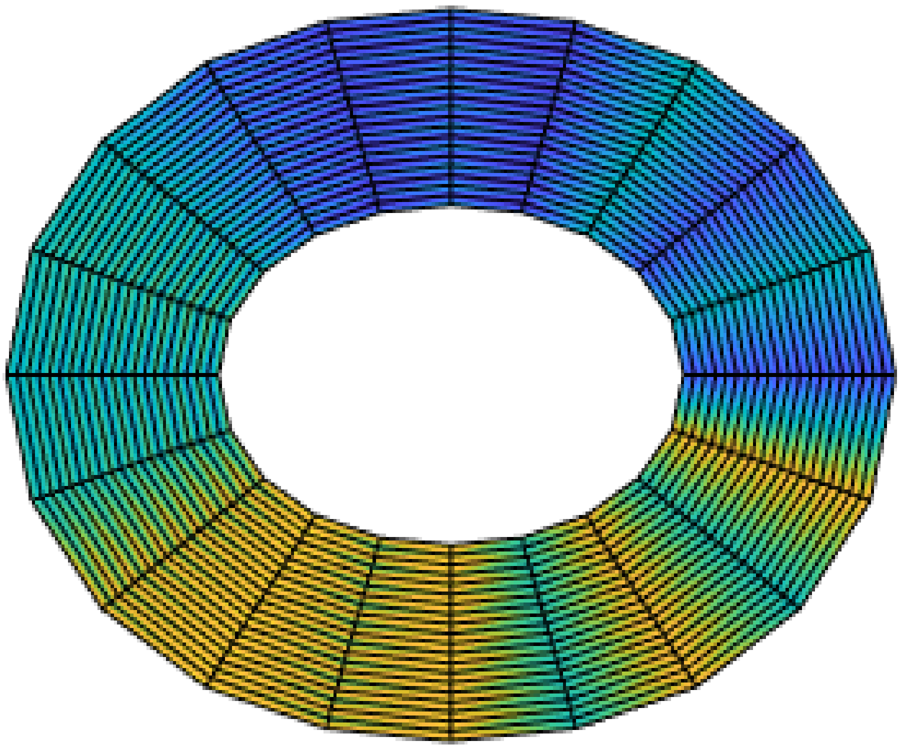
CEE 2333 - FEM Term Project

Group 3

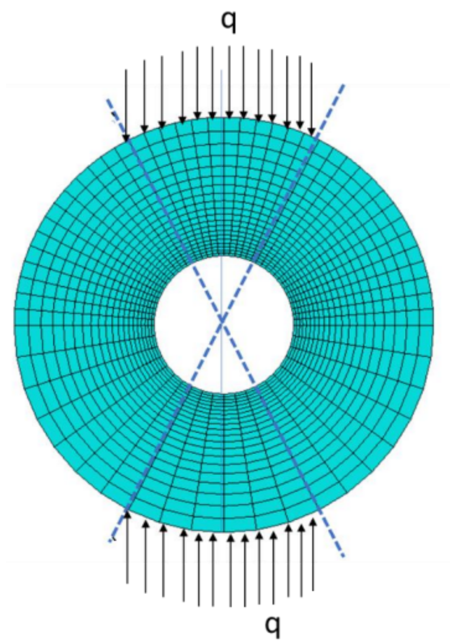
Zach Brody, Aron Griffin, Mason Smetana



1.2 Table of Contents

1.3 Introduction

The problem that is being solved is a cylindrical disk loaded by a vertical symmetrical vertical pressure applied to a portion of the surfaces, as shown below.



By adopting the codes from previous homeworks we were able to develop a new code that analyzed the above structure with these user inputs:

Pre-processor Inputs:

- Quarter or full structure
- External radius
- Internal radius
- Angle of loading
- Number of elements in the radial direction
- Number of elements in the tangential direction
- Material properties
- Magnitude of applied load, q

The user would then be returned a visual model of the structure and its nodes along with a model for displacements, force vectors and stress .

Outputs:

- Displacements (found in local CSV file)
- Stresses (found in local CSV file)
- Plot of the displacement fields

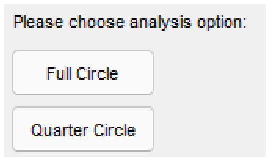
1.4 FEM Model

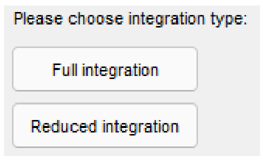
This finite element model allows the user to interface with a Matlab program to solve for the displacements and stresses for each element within the finite element model for a 2-D cylindrical disk. Since this model is symmetric in both loading and boundary conditions, the user has the option to select whether they would like to analyze the entire structure or only a quarter of the structure (as these two methods should produce the same results due to symmetry). In addition, the user has the option to analyze this structure using full or reduced integration. The outputs for this program will include .csv files for the displacements and stresses. In addition, the displacement fields will be plotted to allow the user to imagine how the structure will deform. The methodology described in this report is applied to the user's inputs. The file associated with this report, “CEE2333_Group3_TermProject.zip”, contains 6 files. The “StiffnessMethod2Main.m” file is the main module of code used to solve the specified system in this problem statement. It contains three main sections including:

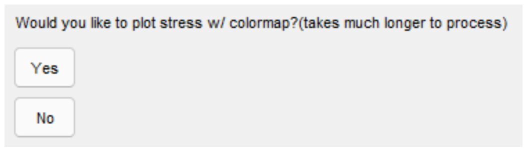
- The preprocessing section where the user is prompted to enter all of the inputs
- The processing step section where the various functions for determining the global stiffness matrix, applying partial surface loads, solving the system for displacements, and solving the system for stresses
- The final section which contains the post-processing section of the code that displays various plots associated with the solution

1.4.1 Preprocessor - User Interface and Input

The user will first input the outer and inner diameters of the tube, the magnitude and angle of loading, Young’s Modulus of Elasticity, and Poisson’s Ratio. It is noted that the maximum applied angle of loading is 180 degrees, or else an error message will occur. These inputs will be carried out through the FEM process. From here, the user now selects which mode of analysis of their preference:


Please choose analysis option:


Please choose integration type:


Would you like to plot stress w/ colormap?(takes much longer to process)

This process proceeds by converting the number of radial elements and tangential elements into global x and y coordinates using a radial sweeping coordinate system. The coordinates are computed by originating from an angle of zero degrees and incrementing by counterclockwise around the axes until a full circle is created (i.e. when the final angle is 360 degrees). The program then assigns global nodal identification numbers to each set of x and y values. It also assigns an element ID to each element within the structure. Each of the elements now contains an ID along with the respective global nodes that are attached to each corner, creating the finite element mesh.

1.4.2 Finite Element Mesh

1.4.2.1 Element Type

The model used in this analysis consisted of the usage of isoparametric elements. Using isoparametric elements allowed the analysis to be run through our code from the previous assignment. The code would develop elements from nodes by transforming sections into a square with each side length being a unit length of 2. The center of the square is placed at the origin of the transformed coordinate system. It is important that the nodes are numbered counter clockwise in the isoparametric element to ensure a Jacobian matrix is produced that is solvable with one solution and non-negative.

1.4.2.2 Creation of the Structure (Meshing)

Full Structure Coordinates

For the full structure, the number of nodes is determined by multiplying the number of radial elements with the number of tangential elements. Each node will have two degrees of freedom due to the 2D geometry of this system. The coordinates are then found by taking the radial distance of the node and multiplying it by the cosine or sine of the angle between the positive x-axis for the x and y coordinates, respectively.

```
%determines coordinates for full or quarter circle
switch(choice)
    case 1 %full
        NumElem = rad_el*tang_el;
        numnodes = (rad_el+1)*(tang_el);
        NumDof = numnodes*2;
        loop2 = tang_el;
    case 2 %quarter circle
        NumElem = rad_el*tang_el/4;
        numnodes = (rad_el+1)*((tang_el)/4+1);
        NumDof = numnodes*2;
        loop2 = tang_el/4 + 1;
end


%determine x and y coordinates of the circle points
xx = zeros(rad_el+1,loop2);
yy = zeros(rad_el+1,loop2);
```

These coordinates are then placed into a global node coordinate matrix:

```
for qz = 1:rad_el+1
    rad1 = OD - ((qz-1)*(OD-ID)/rad_el);
    for jz = 1:loop2
        ang1 = del_rad*(jz-1);
        xx(qz,jz) = rad1*cos(ang1);
        yy(qz,jz) = rad1*sin(ang1);
        nodenum = nodenum + 1;
        temp_el = temp_el + 1;
        globnodecords(nodenum, 1) = nodenum;;
        globnodecords(nodenum, 2) = xx(qz,jz);
        globnodecords(nodenum, 3) = yy(qz,jz);
```

Once the global coordinate system is found, each element within the mesh is turned into an isoparametric element. This is completed within the “StiffnessSpring2.m” code. The isoparametric elements allow the code to determine the displacements and stresses of various shape elements by turning them into square elements with a natural coordinate system of ξ and η. This is discussed in further detail in Section 1.4.5.

Quarter Structure Coordinates

The quarter structure determines the mesh the same way as the full structure. However, the number of tangential elements for the full structure is divided by 4 in order to have a quarter of the structure to have a complete analysis. If the number of tangential elements is not divisible by 4, the resulting structure would have partial elements and the system would have a singularity. It is noted that the code for the quarter structure analysis must have nodes placed on the x-axis and y-axis in order to create a quarter structure. This enforces that there is continuity in the system, and that it is bounded by the rest of the symmetrical structure.

1.4.3 Boundary Conditions and Surface Loading

1.4.3.1 Boundary Conditions

In order to properly restrain the structure , there is one pin on the left side of the disk and one roller on the right side. Both restraints are located on the horizontal axis. This will prevent free body motion of the structure, yet allow the structure to deforme in the vertical and horizontal direction. In the case where a quarter of the structure is selected for analysis, all of the nodes on the y axis will be fixed in only the x direction. All of the nodes situated on the x axis will only be restrained on the right. These boundary conditions ensure that the quarter structure is fully restrained, and no further fixivity is required. The following conditional statement is used to determine how to apply the boundary conditions for both analysis options (full or quarter). The variable choice reflects the option chosen by the user.

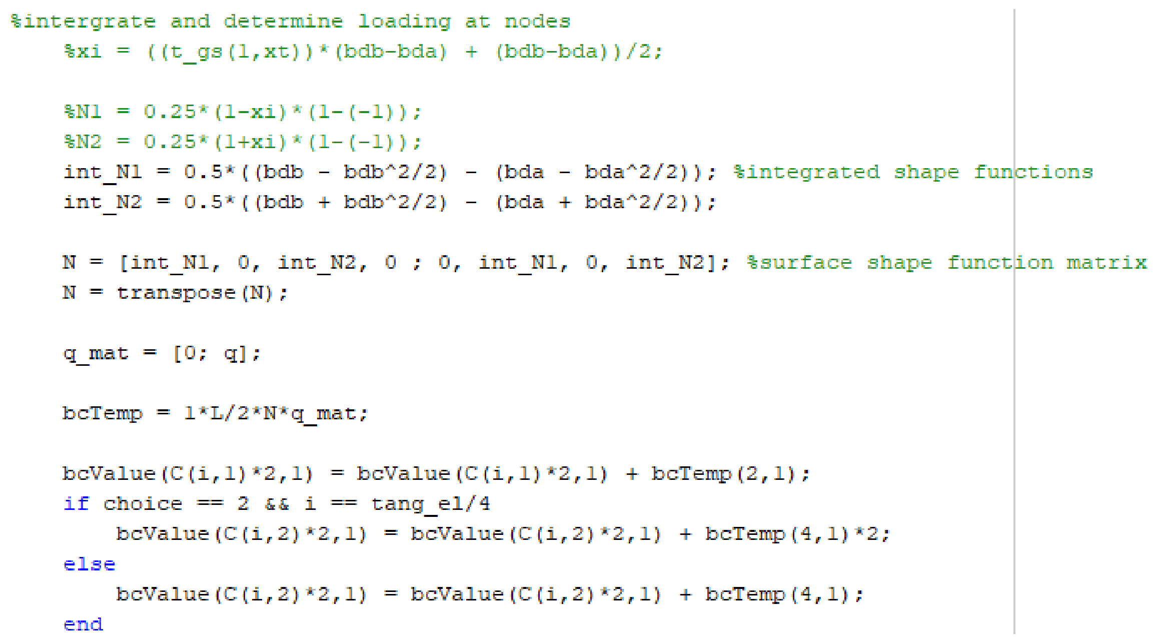
```
if (choice == 1) && (xx(qz,jz) == OD || xx(qz,jz) == -OD)
    if xx(qz,jz) == OD
        bcType(nodenum*2) = 1;
    elseif xx(qz,jz) == -OD
        bcType(nodenum*2) = 1;
        bcType(nodenum*2-1) = 1;
    end
elseif choice == 2
    if angl == 0
        bcType(nodenum*2) = 1;
    elseif angl == pi()/2
        bcType(nodenum*2-1) = 1;
    end
end
```

1.4.3.2 Surface Loading

Within the for preprocessing section of the main code, an algorithm was created to detect which nodes must obtain an equivalent nodal load from the applied surface loading. In the processing step of the main code, the “circloading.m” function is activated in order to calculate the equivalent nodal loads. Surface loading occurred within the angle defined by the user in the vertical direction. First, the global coordinates are transformed into local coordinates. Next, the upper and lower bounds are determined in order to perform the following integration for equivalent nodal forces

$$\{fs\} = \frac{tL}{2}\int_{a}^{b}[Ns]^{T}\{q\}ds$$

Where Ns are the shape functions derived by the natural coordinate system and q is the load vector. In order to determine the bounds of this integral, the location of the loading along the surface also had to be converted to ξ and η coordinates. Then, the length of the member along the outer radius and within the angle of loading were determined. Since the loading conditions remained the same for every element, and occurred over the first 2 nodes in the natural coordinate system, the value of η was always equal to -1. This allowed for a simple calculation and full integration could be achieved. The only surface to integrate across was along the ξ axis. Finally, the value of the load along each member was assigned to the appropriate degree of freedom and stored in the bcValue matrix.



```
%intergrate and determine loading at nodes
    %xi = ((t_gs(1,xt))*(bdb-bda) + (bdb-bda))/2;

    %N1 = 0.25*(1-xi)*(1-(-1));
    %N2 = 0.25*(1+xi)*(1-(-1));
    int_N1 = 0.5*((bdb - bdb^2/2) - (bda - bda^2/2)); %integrated shape functions
    int_N2 = 0.5*((bdb + bdb^2/2) - (bda + bda^2/2));

    N = [int_N1, 0, int_N2, 0 ; 0, int_N1, 0, int_N2]; %surface shape function matrix
    N = transpose(N);

    q_mat = [0; q];

    bcTemp = 1*L/2*N*q_mat;

    bcValue(C(i,1)*2,1) = bcValue(C(i,1)*2,1) + bcTemp(2,1);
    if choice == 2 && i == tang_el/4
        bcValue(C(i,2)*2,1) = bcValue(C(i,2)*2,1) + bcTemp(4,1)*2;
    else
        bcValue(C(i,2)*2,1) = bcValue(C(i,2)*2,1) + bcTemp(4,1);
    end
```

1.4.4 Material Properties

This stage takes the user's input for poisson's ratio and elastic modulus. A matrix is then generated from these properties and is stored as EMAT. The plane stress matrix was used as the [E] matrix because zero stress is assumed in the z direction. It is also assumed that the structure has a thickness of 1 for the plane stress matrix to apply. It is the ideal for thin disks experiencing pressure which is the subject of our analysis.

$$[E] = \frac{1}{1-v^2}\begin{bmatrix} E & vE & 0 \\ vE & E & 0 \\ 0 & 0 & ((1-v)/2)E \end{bmatrix}$$

1.4.5 Reduced or Full Integration

The gauss integration method is used in solving all integrals in future sections. Specifically in generating the local stiffness matrices. Prior to this though the weights and t values for integration are created and stored. The values of these variables are dependent on the user's input of reduced or full integration.

1.4.5.1 For Full Integration:

For n=2:	Gauss Local Coordinates:	Weights
t1 = -0.57735 t2 = 0.57735	x'1 x'2 y'1 y'2	$W_{i1} = W_{i2} = 1$ $W_{j1} = W_{j2} = 1$

1.4.5.2 For Reduced Integration:

For n=1:	Gauss Local Coordinates:	Weights
t1 =0	x'1 x'2 y'1 y'2	$W_{i1} = 2$ $W_{j1} = 2$

1.4.6 Local Stiffness and Global Stiffness Matrices

A coordinate transformation matrix is first generated as zeroes. X and Y values are then retrieved at each element from connectivity data defined in the mesh. They are then translated via the shape functions into the natural coordinate system of xi and eta. Due to the transformation of coordinates, a jacobian must be used in integration in order to properly solve the system. This is where the [B] matrix begins its development in the program.Since the elements were isoparametric bilinear elements the selected shape functions are as follows:

$$N_1(x',y') = 1/4(1 - x'/a)(1 - y'/b)$$
$$N_2(x',y') = 1/4(1 + x'/a)(1 - y'/b)$$
$$N_3(x',y') = 1/4(1 + x'/a)(1 + y'/b)$$
$$N_4(x',y') = 1/4(1 - x'/a)(1 + y'/b)$$

$$* \ Where\ \xi\ is\ defined\ as\ x'/a\ and\ \eta\ as\ y'/b$$

The process of developing the x , y,η and ξ functions prior to the jacobian is seen here:

```
x = locx(1,1)*0.25*(1-xi)*(1-nu) + locx(1,2)*0.25*(1+xi)*(1-nu) + locx(1,3)*0.25*(1+xi)*(1+nu) + locx(1,4)*0.25*(1-xi)*(1+nu);
y = locy(1,1)*0.25*(1-xi)*(1-nu) + locy(1,2)*0.25*(1+xi)*(1-nu) + locy(1,3)*0.25*(1+xi)*(1+nu) + locy(1,4)*0.25*(1-xi)*(1+nu);

%del_x/del_xi = (a + bnu + cxi + dnu*xi)
xb = (-locx(1,1) - locx(1,2) + locx(1,3) + locx(1,4));
xc = (-locx(1,1) + locx(1,2) + locx(1,3) - locx(1,4));
xd = (locx(1,1) - locx(1,2) + locx(1,3) - locx(1,4));
xb = xb * 0.25;
xc = xc * 0.25;
xd = xd * 0.25;

yb = (-locy(1,1) - locy(1,2) + locy(1,3) + locy(1,4));
yc = (-locy(1,1) + locy(1,2) + locy(1,3) - locy(1,4));
yd = (locy(1,1) - locy(1,2) + locy(1,3) - locy(1,4));
yb = yb * 0.25;
yc = yc * 0.25;
yd = yd * 0.25;
```

A zeros matrix is then made for the jacobian and the partial derivatives are directly derived. It is then inverted to become the Γ matrix and placed in the top left and bottom right quadrants to form the $[B_2(\xi, \eta)]$ matrix [B2].

[B1]

```
B1 = zeros(3,4);
B1(1,1) = 1;
B1(3,2) = 1;
B1(3,3) = 1;
B1(2,4) = 1;
```

[B2]

```
J = zeros(2,2);
J(1,1) = xc + xd*nu;
J(1,2) = yc + yd*nu;
J(2,1) = xb + xd*xi;
J(2,2) = yb + yd*xi;

gam = zeros(2,2);
gam = inv(J);
```

[B3]

```
B3 = zeros(3,8);
B3(1,1) = b31;
B3(1,3) = -b31;
B3(1,5) = b32;
B3(1,7) = -b32;
B3(2,1) = b33;
B3(2,3) = -b34;
B3(2,5) = b34;
B3(2,7) = -b33;

B3(3:4,2:8) = B3(1:2,1:7);
B3 = 0.25*B3;
```

The code section for developing the rest of [B] matrix now begins. Then $[B_1]$ and $[B_3]$ matrices are defined [B1, B3]. The matrices are then multiplied to form the general [B] matrix.

$$[B(\xi, \eta)] = [B_1][B_2(\xi, \eta)][B_3(\xi, \eta)]$$

The local stiffness matrix is generated by using: $[K] = t\int_{-1}^{1}\int_{-1}^{1}[B]^T[E][B]det[j]d\xi d\eta$. Where the thickness t is set 1 which is defined in section *1.4.4 Material Properties* . The remaining values are defined previously in this section. The function is then integrated using gaussian integration either reduced or full which is defined in section *1.4.4.2. Reduced or Full Integration.*

```
Ktemp = Wi(1,xt)*Wj(1,xt)*(transpose(B)*EMAT)*B*det(J);
```

The stiffness values in the local [K] matrix are then converted to a global stiffness matrix and then oriented in the global stiffness matrix by values to the corresponding nodes.

```
Ktemp_glob = zeros(NumDof,NumDof);
for ti = 1:8 %row
   for tz = 1:8 %column
        K(globloc(1,ti),globloc(1,tz)) = K(globloc(1,ti),globloc(1,tz)) + Ktemp(ti,tz);
        Ktemp_glob(globloc(1,ti),globloc(1,tz)) = Ktemp(ti,tz);
   end
end

nodenum = nodenum + 1;
%disp(K);
ans11 = ['nu',num2str(xt),' = ',num2str(nu)];
ans12 = ['xi',num2str(zt),' = ',num2str(xi)];
ans3 = ['This is the element B matrix for element NO:',num2str(i),' and gauss Node NO: ',num2str(nodenum)];
ans1 = ['This is the element stiffness matrix for element NO:',num2str(i),' and gauss Node NO: ',num2str(nodenum)];
```

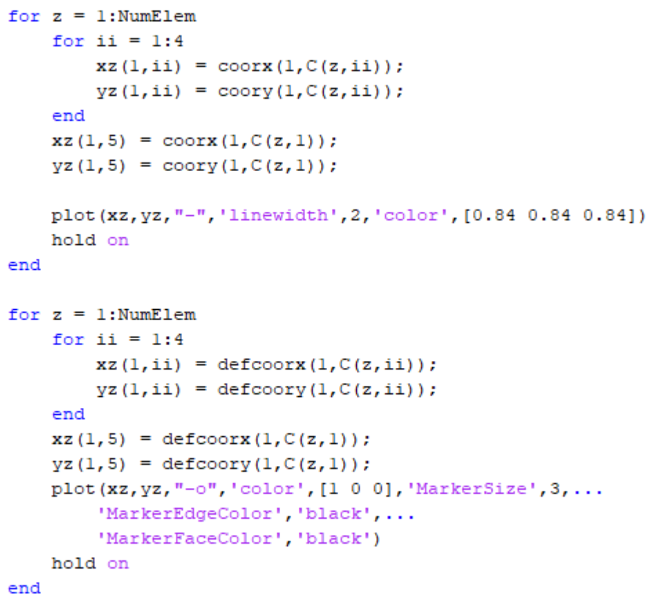
1.4.7 Solving for Displacements and Stresses

The process of calculating the displacements stresses remained the same throughout the previous projects and was not changed for this element. The location of these equations are present in the post-processing step of the main code. The general solution for solving this system of equations for displacements is as follows:

$$[K]\{u\} = \{f\} \rightarrow \{u\} = [K]^{-1}\{f\}$$

1.4.8 Postprocessor - Display of Solution

The element mesh and displacement fields are plotted in this section to provide the user a visual as to how their model is developed. In this section of the program, there are various functions created to be able to plot each of the subplots in a resulting figure. Each function loops through each element and receives x and y coordinates for all of the nodes attached to the element. For the displacement field, the calculated displacements are added to the original x and y coordinates to ultimately show the deformed figure. Initially the deformation scale should be set to 1 to see true deformation, however the user has the option to increase or decrease this scale if they prefer. An example of one of these plotting loops can be shown in the following code:



```
for z = 1:NumElem
    for ii = 1:4
        xz(1,ii) = coorx(1,C(z,ii));
        yz(1,ii) = coory(1,C(z,ii));
    end
    xz(1,5) = coorx(1,C(z,1));
    yz(1,5) = coory(1,C(z,1));

    plot(xz,yz,"-",'linewidth',2,'color',[0.84 0.84 0.84])
    hold on
end

for z = 1:NumElem
    for ii = 1:4
        xz(1,ii) = defcoorx(1,C(z,ii));
        yz(1,ii) = defcoory(1,C(z,ii));
    end
    xz(1,5) = defcoorx(1,C(z,1));
    yz(1,5) = defcoory(1,C(z,1));
    plot(xz,yz,"-o",'color',[1 0 0],'MarkerSize',3,...
        'MarkerEdgeColor','black',...
        'MarkerFaceColor','black')
    hold on
end
```

The first loop in the code above displays the undeformed shape of the structure. The second loop is where the deformed shape is able to be displayed against the original shape. The user has the choice to display stress plots in the x, y, and xy orientations. If this mode is selected, the overall processing time will increase significantly in order to process the heatmaps of the various stresses. This is ultimately to be used at the users discretion, if information about these values is important to their analysis.

1.5 Numerical Analysis

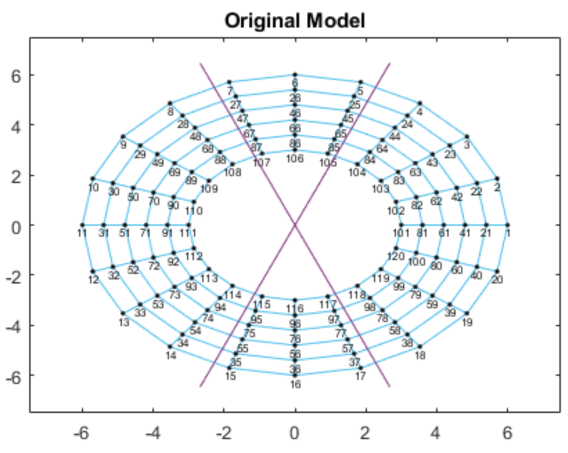
1.5.1 Preprocessor Input

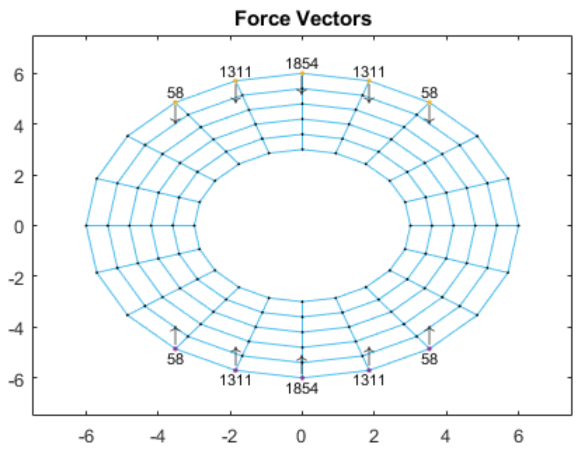
To begin the program must be run using Matlab (this code was written in version Matlab R2019a). Within the command window of Matlab, this is where the user has to input all of the inputs described in previous sections. For the remainder of this section, the following inputs will be demonstrated in this analysis for a full circle and full integration model:

```
Reading input data
Outer radius of the circle (in): 6
Inner radius of the circle (in): 3
Magnitude of surface load q(lbs): 1000
Input angle (in radians) for surface load q: 45
What is the elastic modulus (E - psi) of the material: 36000
What is the materials poisson ratio (v): 0.15
Number of radial elements: 5
Number of tangential elements: 20
Enter deformation scale: 1
```

**Note: The materials properties and geometry used in this analysis may not be of any known material. They are strictly used for demonstration purposes.*

This model contains 100 elements, 120 nodes, and 240 degrees of freedom. After the construction of the finite element mesh is created (as shown in the figure below), the element's global nodal locations are assigned to each corner of the individual element. At a load angle of 45 degrees, on the top of the surface nodes 4-8 have equivalent nodal loads from the surface loading condition. Similarly, nodes 14-18 are loaded with equal but opposite forces as a result of the surface load along the bottom of the structure.





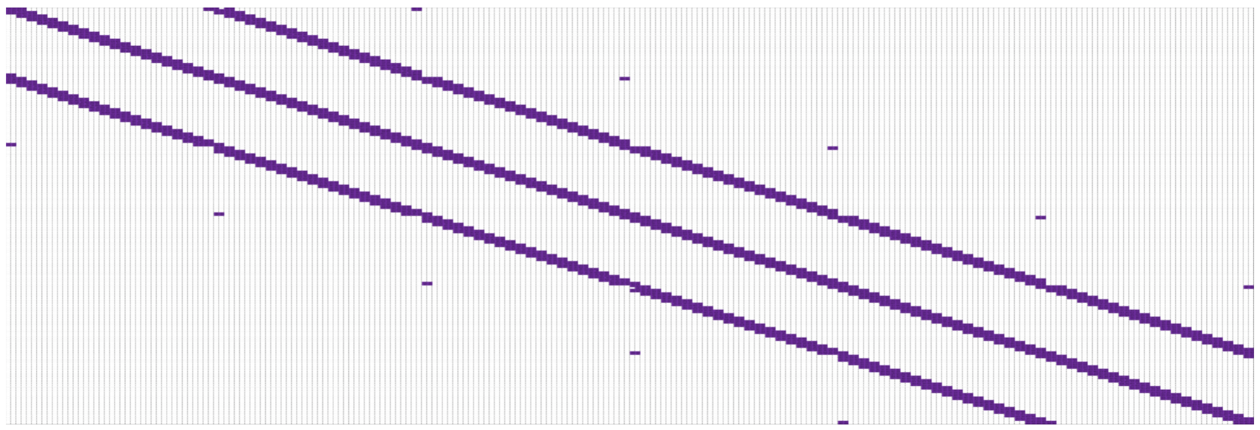
1.5.2 Resulting Stiffness Matrices

1.5.2.1 Local Stiffness Matrix

2662.548	-956.04	4585.484	-1432.91	-6019.35	2005.004	-1228.68	383.9477
-956.04	2208.09	563.9889	87.30791	543.1713	-2272	-151.12	-23.3941
4585.484	563.9889	16313.85	4361.332	-16528	-3756.71	-4371.28	-1168.62
-1432.91	87.30791	4361.332	10152.38	-1759.8	-7519.36	-1168.62	-2720.32
-6019.35	543.1713	-16528	-1759.8	18118.73	745.0948	4428.677	471.5381
2005.004	-2272	-3756.71	-7519.36	745.0948	7776.559	1006.606	2014.807
-1228.68	-151.12	-4371.28	-1168.62	4428.677	1006.606	1171.282	313.1295
383.9477	-23.3941	-1168.62	-2720.32	471.5381	2014.807	313.1295	728.9078

The matrix above is displaying the local stiffness matrix for element number 1. This local stiffness matrix is a direct result of gauss integration as described in the model section of this report. Element number 1 is connected to nodes 1, 2, 22, and 21. The nodes' respective degrees of freedom are (1,2), (3,4), (43,44), and (41,42). When the local stiffness matrix gets placed into the global stiffness matrix, the rows and columns will be assigned to the designated degrees of freedom.

1.5.2.2 Global Stiffness Matrix

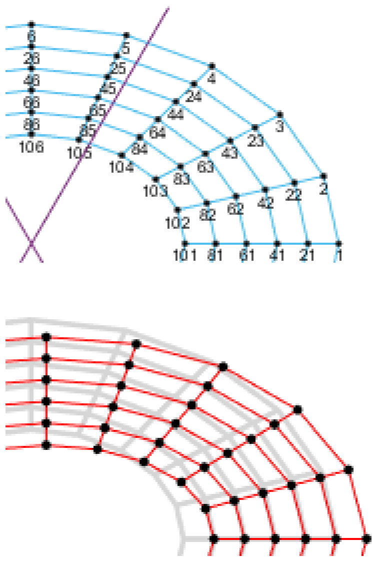


The matrix is formed in the processing step of the matlab function. The resulting global stiffness matrix is a 240x240 matrix (as a product of DOF x DOF).This matrix then applies the boundary conditions defined in the model. The structure is fixed in both degrees of freedom at node 11 and fixed in just the y degree of freedom at node 1.

1.5.3 Resulting Displacements

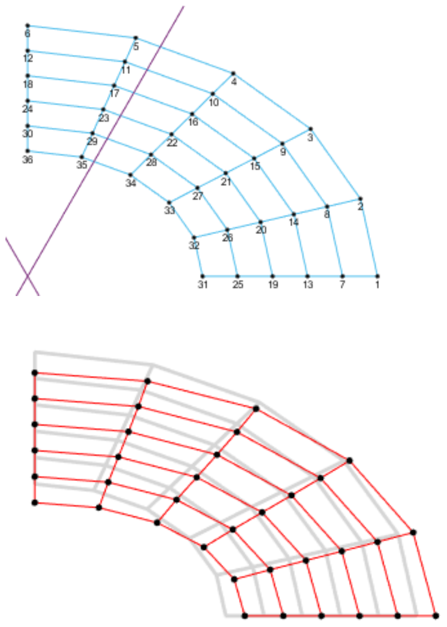
The following displacements are shown for full integration in both structures.

1.5.3.1 Displacements in Full Structure



Displacements		
Node	u	v
1	0.596084	0
2	0.533298	0.043618
3	0.386328	0.003201
4	0.246783	-0.14497
5	0.215229	-0.3704
6	0.298042	-0.47209

1.5.3.2 Displacements in Quarter Structure

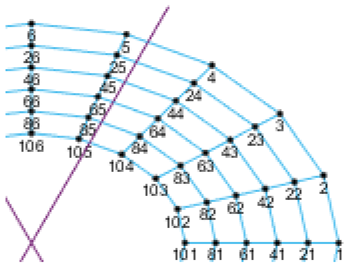


Displacements		
Node	u	v
1	0.298042	0
2	0.235256	0.043618
3	0.088285	0.003201
4	-0.05126	-0.14497
5	-0.08281	-0.3704
6	0	-0.47209

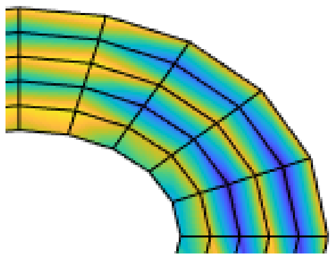
1.5.4 Resulting Stresses

The following stresses are shown for full integration in the both structures. The heatmap displayed stresses in the x direction.

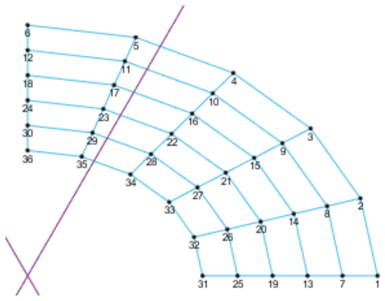
1.5.4.1 Stresses in Full Structure



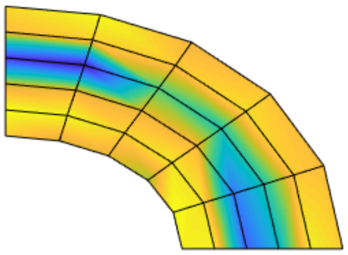
Stress					
Element	Gauss No	Global Nc	Stress-x	Stress-y	Stress-xy
1	1	1	35.63656	796.3165	-382.321
1	2	2	-66.5024	927.5084	11.48352
1	3	22	-16.5759	466.8647	-354.679
1	4	21	-125.117	606.2794	63.80821



1.5.4.2 Stresses in Quarter Structure

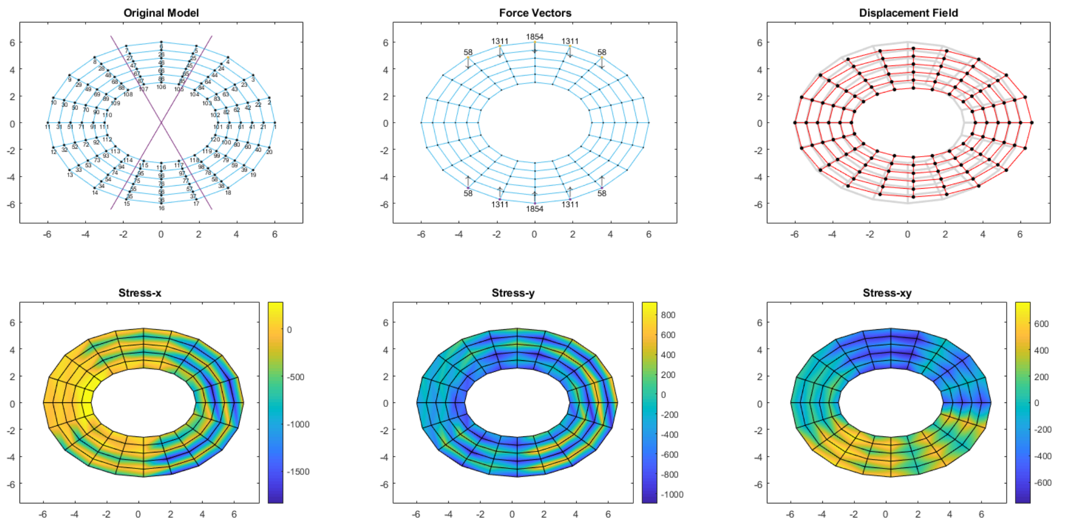


Stress					
Element	Gauss No	Global Nc	Stress-x	Stress-y	Stress-xy
1	1	1	35.63656	796.3165	-382.321
1	2	2	-66.5024	927.5084	11.48352
1	3	8	-16.5759	466.8647	-354.679
1	4	7	-125.117	606.2794	63.80821

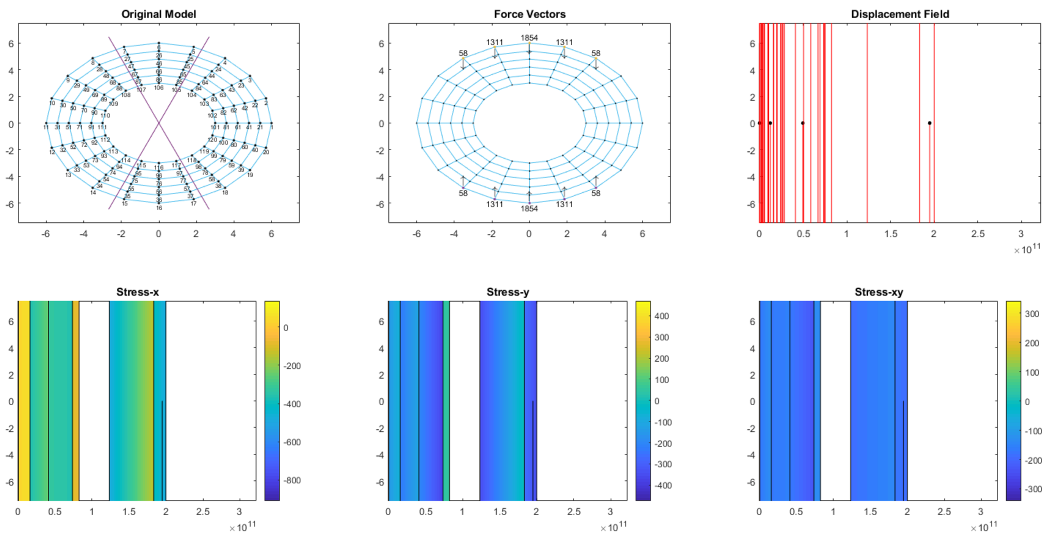


1.5.5 Full Structure Analysis Results

1.5.5.1 Full Integration

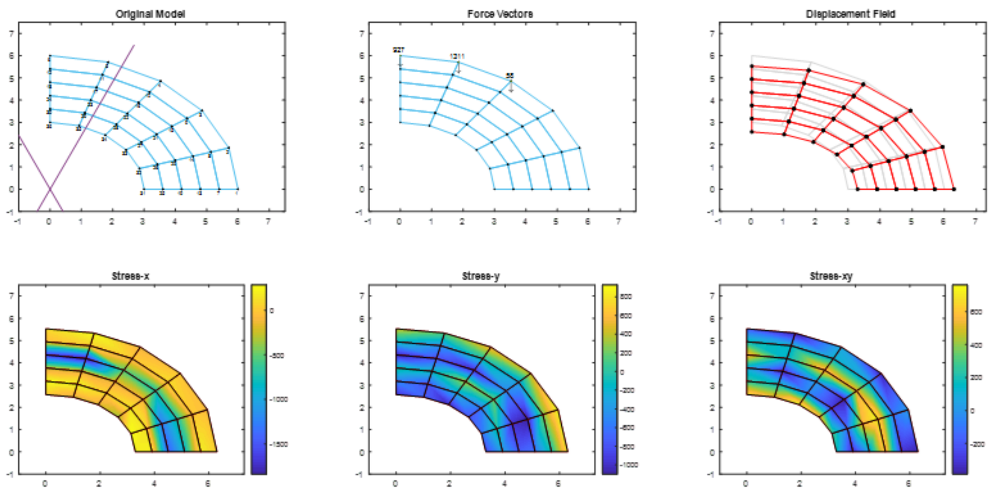


1.5.5.2 Reduced Integration

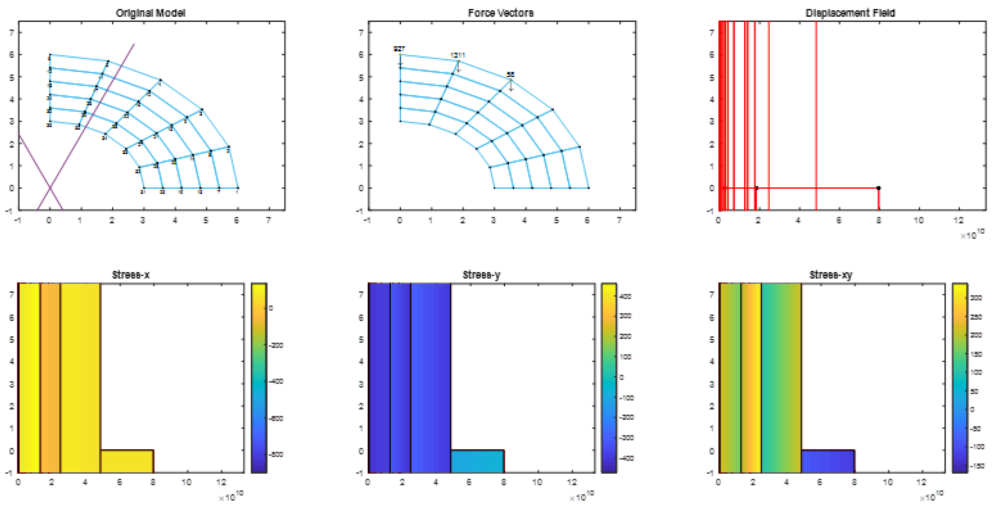


1.5.6 Quarter Structure Analysis Results

1.5.6.1 Full Integration



1.5.6.2 Reduced Integration



1.6 Discussion

The code developed stemmed from projects 6b, 7b and 8b. The first assignment; 6b began with the analysis of a single bilinear element. The second is the analysis of 2 isoparametric bilinear elements. The third required the utilization of symmetry in solving the same structure in 7b. The analysis of this cylindrical disk utilized all of the components of those assignments, the difference was the scale and flexibility of the code. This new code had to provide the user with options so they may analyze this cylindrical disk numerous ways.

When developing this new expansive code we specifically ran into troubles with understanding how the shape function should be defined for each element. It was then recalled how the code automatically places each nodal element into local coordinates, so nothing was needed to be done to the code. This became useful when determining how the partial surface load would be developed into equivalent nodal loads. Knowing that the value of η was always equal to -1 along the surface of an element would further simplify the process of determining the location of the partial loads and their corresponding bounds.

We additionally had differing displacements when comparing our quarter structure to our full structure. We thought it had to do with the boundary conditions since the displacements on nodes that had constraints were nonzeroes. We troubleshooted by adding code that displayed load vectors. This allowed for the comparison of what the loads were in the quarter and full structure. We found out our loading for the symmetric case was not half of the load of the full node. Applying this condition fixed improved our results. The differences between the displacement still returns to the idea of how the full structure is restrained. The resulting deformation allowed the nodes along the y axis to move off of the axis in the full structure, whereas in the quarter structure they were always restrained to this axis.

We lastly had trouble with developing and interpreting the reducing integration displacements. The problem with reduced integration is that it condenses each bilinear element to one point for integration as opposed to 4 which effectively reduces the accuracy of the developed stiffness matrices. When solving the system, matlab would typically report an error about the stiffness matrix solution containing a singularity and that the accuracy of the solution would indeed be compromised. The decrease in accuracy is due to element instability with reduced integration. The reduced integration causes the stresses and displacements to be calculated at the center of each element. This is referred to as a zero-energy mode due to displacements occurring that do not represent a rigid body. While reducing the integration value for developing the stiffness matrices saves computational power it sacrifices accuracy in results. In the case of this project the accuracy was so far off that the displacements could not be properly plotted.

1.7 Conclusion

Throughout the process of developing the code we were able to integrate all of the topics discussed in class. The project gave us insight on what is happening behind the scenes on professional solvers suich as Abaqus CAE. We were able to develop the notion that with the knowledge of finite element method we can better understand how to load, mesh and constrain elements in these professional programs. In this model, the deformed shape agrees with our intuition for the full integration process, and is sufficient for our current understanding of how the finite element method can be used. We would additionally be able to tell now whether our results make sense and the quality of our results if they do make sense. By administering all these new found skills in our career we can become better users of finite element programs. However with all these new skills, the finite element method is still in arduous process affirming the notion that there are no free lunches.