[Date]

Stress and buckling analysis of thin-walled steel plate

Computer assignment 3 – FEM Structures

Chalmers University of technology
LACERDA ANTONIO – BROUSSEAU PAUL

Lacerda Antonio
Brousseau Paul

Table des matières

Problem description

In this assignment, buckling of a thin-walled steel plate is studied. Buckling is an instable phenomenon of a structure that is subjected to a normal compressive force. This causes the latter to bend and deform in a perpendicular direction to the compression axis. In this case, i.e. a thin walled plate, buckling is enhanced by the very small thickness compared to the two other dimensions. However, in this project, the buckling problem will be simplified as a Linearized Pre-Buckling problem resulting in an eigenvalue problem.

In-plane loading is applied on the top boundary of the plate, whereas a transverse distributed load is applied perpendicularly to the plane on the z axis, supported along the entire boundary. The left and right boundaries are fixed in terms of their in-plane motion (figure 1) whereas the out-of-plane displacements are prevented. Moreover, the rotation is free along the whole boundary, representing a simply supported plate.

Thus the boundary conditions are set like in a simply supported plate model. We will consider a mesh building quadrilateral elements where each node presents five degrees of freedom which include three displacements and two rotations.

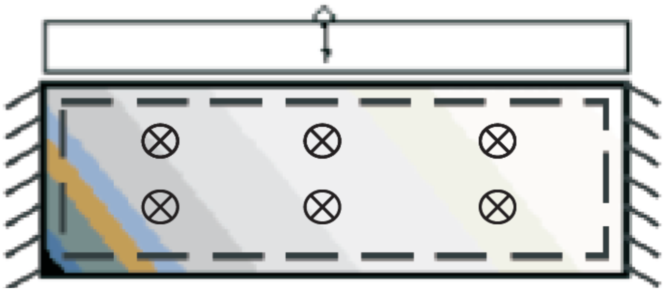


Figure 1 - A plate subjected to in-plane loading applied to the top boundary. The plate is simply supported along the entire boundary and a transverse distributed load qz is applied perpendicular to the plane.

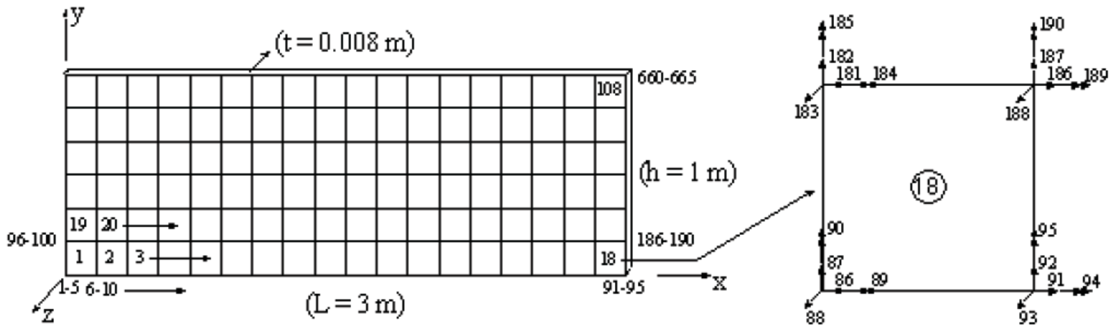


Figure 2 - Mesh and degree of freedom of the plate

Kirchhoff and Mindlin theory are both used to make the stress analysis of the plate. Then we will be able to compute the stress, moment and compressive stress distribution for different thickness. Finally, we make a buckling analysis simplified as a linearized pre-buckling problem resulting in eigenvalue due to the fact that the displacement response below the buckling load (for combined in-plane and lateral loading) can become highly non-linear.

Constants for the problem:

Isotropic steel:
$\mathrm{E} = 210\ \mathrm{GPa}$ and $\vartheta = 0.3$

Dimensions:
$\mathrm{Lx} = 3.0\ \mathrm{m}\ ;\ \mathrm{Ly} = \mathrm{h} = 1.0\ \mathrm{m}\ ;\ \mathrm{t} = 0.008\ \mathrm{m}$

Analysis

In order to realize this analysis, the function *quadmesh* generates a mesh composed of quadrilateral elements and give us access to:

- **Coord** Nodal coordinates matrix.
- **Dof** Nodal dofs matrix.
- **Enode** Topology matrix in terms of nodes.
- **Edof** Topology matrix in terms of dofs, in each row in Edof the dofs are arranged anti-clockwise starting from the bottom left node of each element (5 dofs/node). For each node, first the three translational dofs are counted (ux,uy,uz) and then the two rotational (theta_x,theta_y)
- **Ex** Element nodal x-coordinates, where the nodes are counted anti-clockwise starting from the bottom left node of each element.
- **Ey** Element nodal y-coordinates, where the nodes are counted anti-clockwise starting from the bottom left node of each element.
- **DofRight** Translational dofs (ux,uy,uz) at the right boundary.
- **DofTop** The same as DofRight for the top boundary
- **DofLeft** The same as DofRight for the left boundary
- **DofBottom** The same as DofRight for the bottom boundary
- **cdof** Translational dofs (ux,uy,uz) at the center node of the rectangular domain (only for even number of elements in x- and y-direction)

Stress analysis

In order to realize the buckling analysis, a stress analysis of the plate is required. It is made in both Kirchhoff and Mindlin theories. Thus, stress and moment can be computed to calculate the deflection for the two different theories.

Loads and boundary conditions

As it was sated before, there are two different loads applied on the plate:

- The traction $t_Y = -5.10^6\ \mathrm{N/m^2}$ applied on the top boundary of the plate.
- The transverse and constant distributed load qz = 1000 $\mathrm{N/m^2}$ acting perpendicular to the plate.

The plate is locked on the left and right boundaries since the boundary conditions are the same as if it was a simply supported plate, therefore: $a_x = a_y = a_z = 0$. Finally, on the top and the bottom side, displacement along z axis are also locked, thus $a_z = 0$.

Element stiffness matrix Ke

In order to compute the element stiffness matrix **Ke**, a rectangular plane-shell element subroutine *shell2re* is implemented by using Calfem. It combines the functions *planre* and *platme* or *platre,* respectively for Mindlin or Kirchhoff theory (see figure 3 for illustration). This subroutine enables us to compute **Ke** and **fe** for each element, i.e. 5x4 degrees of freedom, 3 for displacements and 2 for rotations. We can thus build **Ke** and the element load vector **fe** (for all elements) with a *for* loop applied to all of the elements and *shell2re.*

Note that for the Mindlin theory, the vector **ep** needs the integration rules for bending *irb* and shearing *irs* in addition to *ptype* and the thickness *t*.

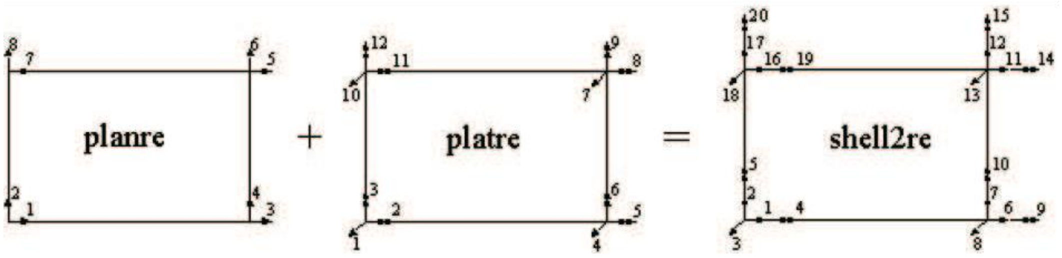


Figure 3 – Subroutine shell2re implementation illustration for Kirchhoff theory

Thanks to shell2re and then by assembling **Ke** and **fe**, we can compute **K** and **f** without forgetting to add the traction force.

Deformation of the plate

Now that we have the matrix **K**, we are able to find the displacement **a**. According to the finite element method we can rearrange **K** to get:

$$K = \begin{bmatrix} K_{FF} & K_{FC} \\ K_{CF} & K_{CC} \end{bmatrix}$$

Indeed, the degrees of freedom are already split in fdofs (free) and cdofs (constraints). We already now a_C because of the boundary conditions so we can compute a_F to deduce **a**:

$$af = K_{ff}{}^{-1}(f_f - K_{fc}a_c)$$

Therefore, we are able to find the element displacement matrix **Ed** with the function *extract*.
Then, we use the given function Dispviz3 to plot the deformations on the plate:

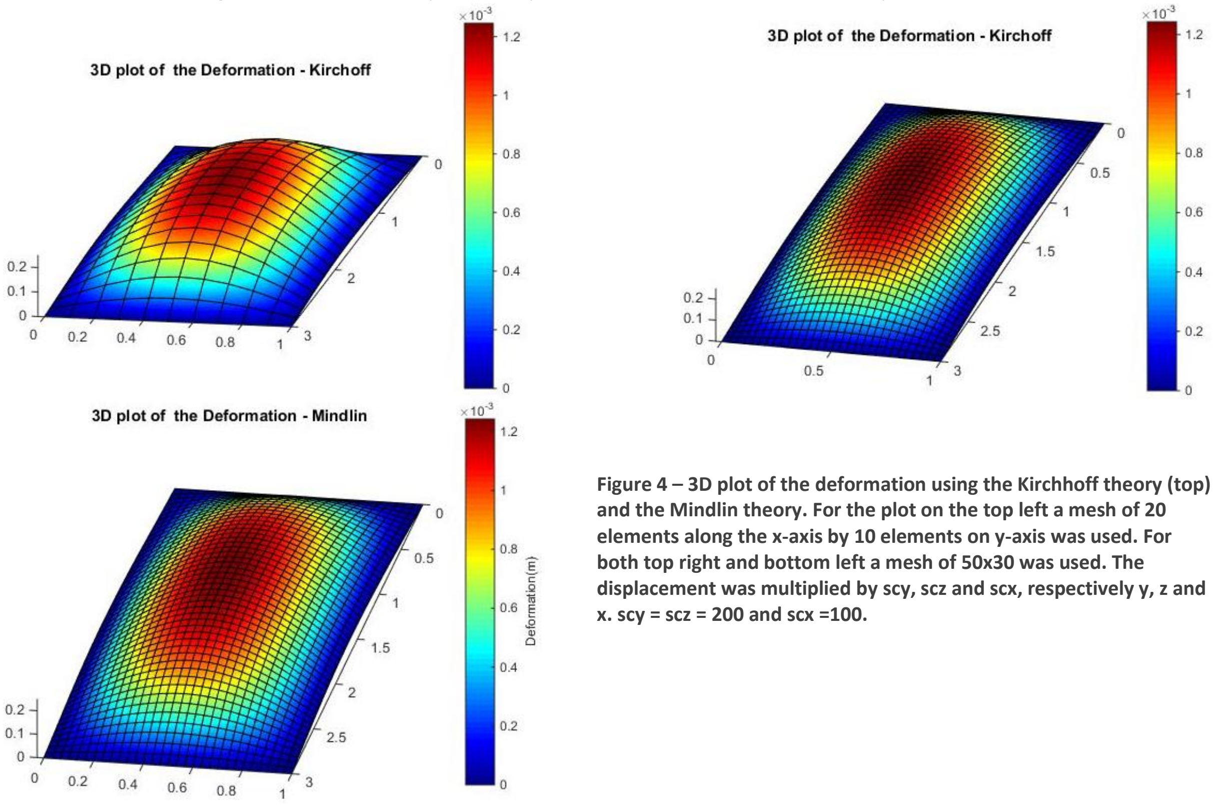


Figure 4 – 3D plot of the deformation using the Kirchhoff theory (top) and the Mindlin theory. For the plot on the top left a mesh of 20 elements along the x-axis by 10 elements on y-axis was used. For both top right and bottom left a mesh of 50x30 was used. The displacement was multiplied by scy, scz and scx, respectively y, z and x. scy = scz = 200 and scx =100.

As one can observe in the image above, there is a deformation along y and a smaller one along the z axis. This is expected due to the stress applied to the plate. Moreover, it can be seen that there is no significant numerical difference between a finer or coarse mesh for the 3D deformation plot. Another observation is that the latter doesn't change whether one is using the Kirchhoff or Mindlin theory.

Stress and moment computation

We created the subroutine *shell2rs* in order to compute the stresses, the strains, the section forces and the curvature. Inside the subroutine, the function planrs is used to compute the stresses $es = \left[\sigma_{xx}\ \ \sigma_{yy}\ \sigma_{xy}\right]$ and the strains $et = \left[\varepsilon_{xx}\ \ \varepsilon_{yy}\ \ \varepsilon_{xy}\right]$ and the function platrs is used to compute the section force $ef = [M_{xx}\ \ M_{yy}\ M_{xy}\ V_{xz}\ V_{yz}]$ and the curvature $ef = [K_{xx}\ \ K_{yy}\ 2K_{xy}]$.

Thus for each element we can compute the matrix **S**: $\bar{\bar{S}} = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & 0 \\ \sigma_{xy} & \sigma_{yy} & 0 \\ 0 & 0 & 0 \end{bmatrix}$. And then plot the stress with the function *fill*:

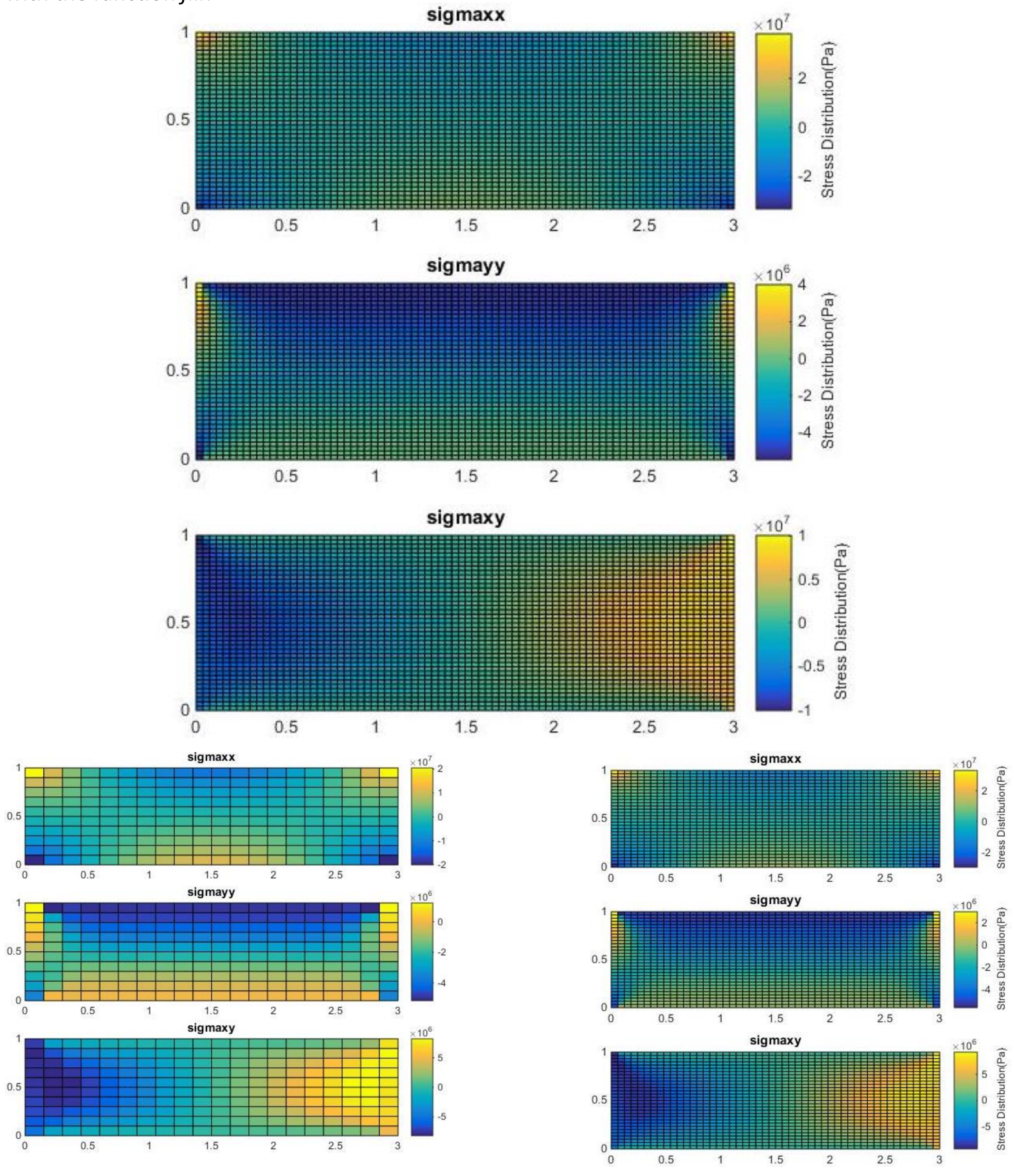


Figure 5 – 2D plot of the Stress distribution. The top was made with a fine mesh 80x40, the bottom right 50x30 and the bottom left 20x10

As one can see, there is a small difference between the values if two plots with different number of elements is used. This decreases as one start to compare finer and finer mesh, i.e. 80x40 versus the values obtained for 50x30. For further plots only the 80x40 mesh will be used.

In the first two stress plots, i.e. $\boldsymbol{\sigma_{xx}}\ and\ \boldsymbol{\sigma_{yy}}$, one can see the right and left corner of the top edge in tension, i.e. they are stretched, while the right and left corner of the edge at the bottom are in compression, i.e. they are pressed against the “walls” due do the boundary (Figure 6). This is explained by the type of deformation resulting from the applied load $\boldsymbol{t_y} = -5.106\, Nm^{-2}$ and from the boundary conditions on the right and left side, which lock the displacement along all directions. Moreover, one can also see that there is a compressive stress on the top edge and a tensile stress on the bottom edge. This can be seen especially at the $\boldsymbol{\sigma_{yy}}$plot and is justified by $\boldsymbol{t_y}$. In the $\boldsymbol{\sigma_{xy}}$ plot one can see the shear stresses along the plate. The latter is smaller in the center and in the bottom and top edges while having its highest and smallest value on the, respectively, right and left corner. This is expected as in the middle the plate will twist less. The change in sign seen as the shear in the right and left border is expected.

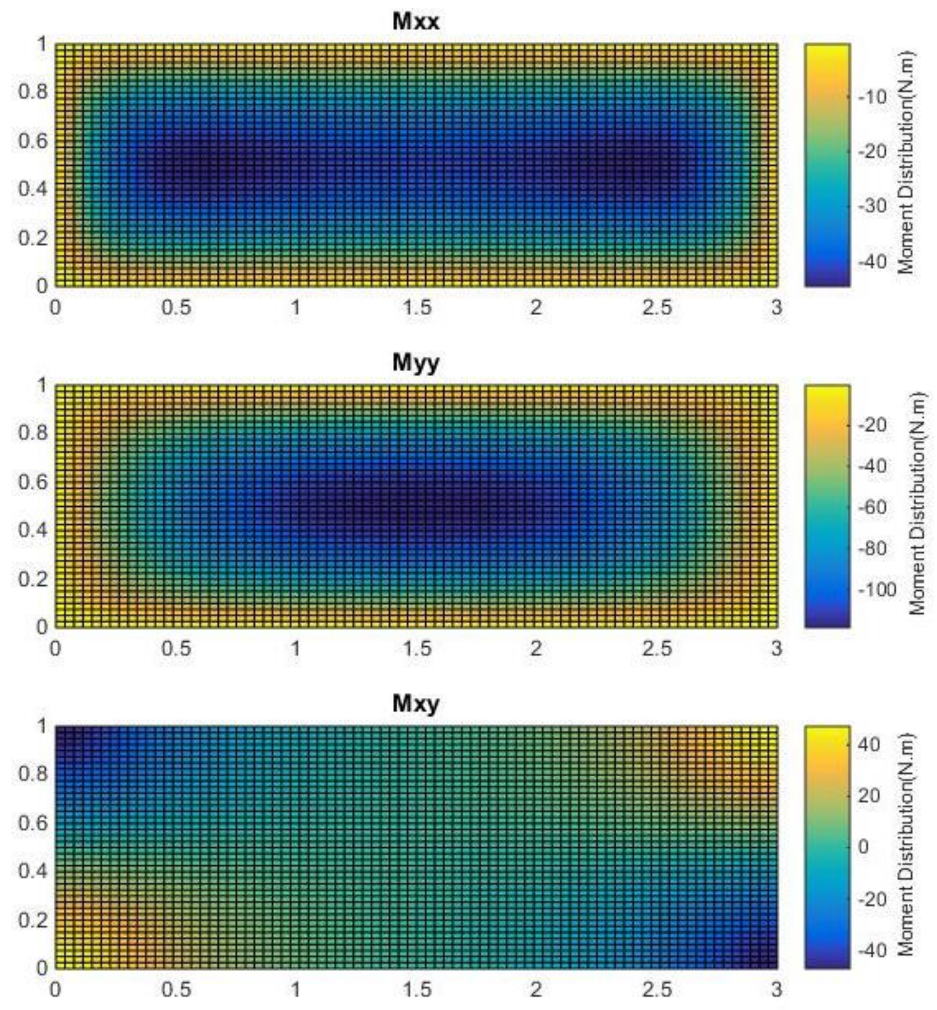


Figure 6 - 2D plot of the moment distribution at the plate. This is the plot corresponding to a 80x40 mesh.

Since the biggest deflection is on the center of the plate, one can understand that the biggest moment $\boldsymbol{Mxy}$ and $\boldsymbol{Mxy}$ will also be located in the center. Moreover, due to the fact that $\boldsymbol{Mxy}$ is the twisting moment, that we are applying two perpendicular loads and that the plate cannot move in any direction at the right and left corner, it makes sense that the biggest values for $\boldsymbol{Mxy}$ will be at opposite sides of the corner of the plate. The same line of though can be applied for the minimum value of $\boldsymbol{Mxy}$ ($\boldsymbol{Mxy} \cong -40$) and for $\mathbf{Mxy} = 0$.

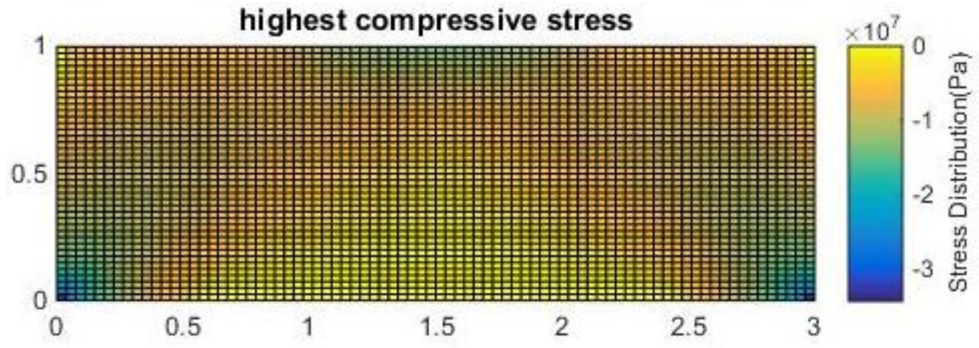


Figure 7 : Highest compressive stress distribution at the plate. Mesh size is 80x40.

As one can see, in the middle of the top edge the compressive stress is high while on the bottom edge it is nearly zero. This is a result of the deformation caused by $\boldsymbol{t_y}$, where the top is being compressed. Moreover, one can also observe a high compression at the corner of the bottom edge. This happens for the same reason explained above.

Finally, since the thickness used here is $0.008m$, there is no significant difference between the plots using the Kirchhoff theory and the Mindlin theory, especially for the fine mesh used for the simulations above. This will be further discussed in the section bellow.

Deflection vs thickness for Mindlin and Kirchhoff

In order to analyze the difference in results between the Kirchhoff and the Mindlin plate, we create a loop to plot the deflection versus the thickness for t between 0,001m and 0,5m. We need to multiply the deflection by t^3 because the deflection for Kirchhoff theory is linked as $w = {}^{1}/_{t^3}$.

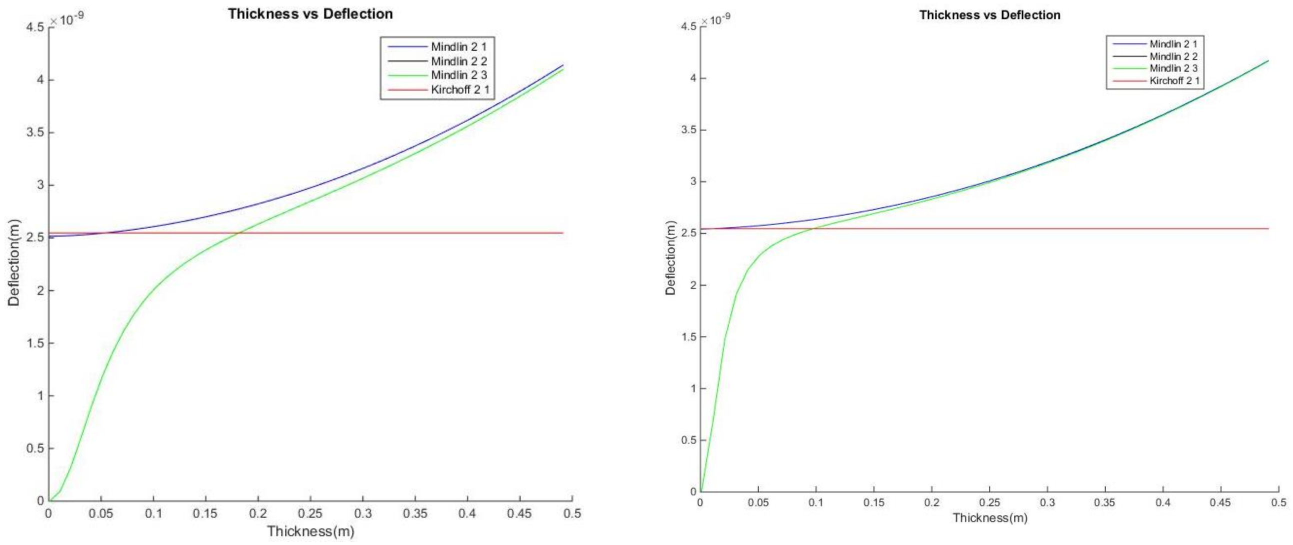


Figure 8 – Thickness vs Deflection plotted for four different cases. On the left we have for a 30x10 mesh and on the right for a 50x30 mesh

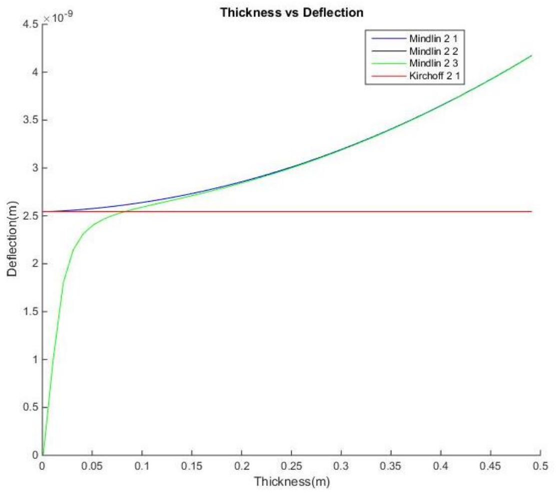


Figure 9– Thickness vs Deflection plotted for four different cases for a mesh with a size of 80x40

Initially one can see that Kirchhoff was successfully scaled when using the t^3 factor, as it was expected by the explanation above. Moreover, it can be seen that the deformation plot for the Mindlin theory is not constant, which, again, is expected since the deformation, for the latter, is not linked as $1/_{t^3}$.

As it is known, the main difference between the Kirchhoff and Mindlin theory is that, for the latter, shear deformations is taken in account. This might not prove to be a significant factor if the thickness is small in relation to the other dimensions of the plate, nevertheless, if the plate is thick, that factor proves itself important. This is because the thicker the plate, more important the shear becomes in relation to the bending. The statement above can be clearly seen in the plot for the Thickness vs Deflection. There, we can see little difference between the Mindlin (blue) curve and the Kirchhoff (red) curve. This also explains what has been seen in the previous section, where the results for both theories were extremely similar.

Another important observation is the convergence of the Mindlin simulation (blue) for small thickness towards then Kirchhoff value. One can see that for finer meshes the former approach the latter. This is predictable, since both theories are expected to give good approximations for this thickness interval. Although this is the case for the blue curve, where the integration points are 2 for bending and 1 for shear, the same cannot be concluded from the green and black curves, which have for bending 2 integration points and for shearing, respectively, 2 and 3 integration points. Indeed, for when the thickness goes towards zeros one find that the approximations for the deflection and rotation are incompatible. This incompatibility, called shear locking, can be avoided by using a reduced integration, i.e. 2 bending and 1 shearing points, and causes the constrained bending seen in this plot.

The Shear locking also explains why we find that for bigger values of thickness the green and black curve will converge towards the blue curve. One other interesting observation is that for finer meshes the green and black curve will converge towards the blue faster than for coarse meshes. This can be understood as that by increasing the number of elements used we will also decrease de thickness for which the constrained bending will be seen.

As the reader might have realized, we talked here about the green and black curve, although the latter cannot be seen in any of the three plots above. This might show that there is a perfect rectangle for our mesh element. Indeed, the polynomial found with a 2 points integration is of a smaller degree compared to the one found for a three point integration. In the case that we have a perfect rectangle, there will be no difference between the two polynomials associated with each integration.

Finally, one can state that the Mindlin theory is better for thick plates and can be used when the thickness is small compared to the other dimensions if a reduced integration is used. The Kirchhoff model is good for plates with a small thickness.

Buckling analyses

It is possible to find the parameter λ for which the structure start to buckle since the existence of compressible stresses indicates a risk for buckling. This is actually the load λq_z and λt_y the structure can carry out before it buckles.

The stress analysis will be the bases as a reference response in a linearized pre-buckling analysis. Thus, the critical value of the load parameter λ is solved from the pertinent eigenvalue problem. This is done under Kirchhoff plate assumption.

Element material and geometrical stiffness matrices KMe and KGe

First we need to compute the element material matrix KMe and the element geometrical matrix KGe , which affects the bending stiffness because of the in-plane stress state in the plate. To do so, a new subroutine is created. The *shell2rg* subroutine is implemented by adding the second-order effects to the element stiffness routine shell2re and is expressed as:

$$[\mathrm{KMe}, \mathrm{fe}, \mathrm{KGe}] = \mathrm{shell2rg}(\mathrm{ex}, \mathrm{ey}, \mathrm{ep}, \mathrm{D}, \mathrm{eq}, \mathrm{es})$$

With the matrix es containing the stresses calculated by the stress routine shell2rs.

Thus, KMe is computed with *shell2re* while KGe is computed by according to the following 3x3 Gauss points numerical integration:

$$K_G^e \approx \sum_{i=1}^{3}\sum_{j=1}^{3}\left\{\left(\nabla N^e(x_i', y_j')\right)^T t\sigma(\nabla N^e(x_i', y_j'))H_iH_j\right\}$$

Where:

- $x_i',\ y_{j,}'$ H_i and H_j are, respectively, the Gauss points and Gauss weights given in Ottosen&Petersson book. In order to get the expression of the shape-functions they are scaled accordingly to our element's dimensions, with the **C** matrix expressed in the CALFEM manual for the *platre* function:

$$\mathrm{C} = \begin{bmatrix}
1 & -a & -b & a^2 & ab & b^2 & -a^3 & -a^2b & -ab^2 & -b^3 & a^3b & ab^3 \\
0 & 0 & 1 & 0 & -a & -2b & 0 & a^2 & 2ab & 3b^2 & -a^3 & -3ab^2 \\
0 & -1 & 0 & 2a & b & 0 & -3a^2 & -2ab & -b^2 & 0 & 3a^2b & b^3 \\
1 & a & -b & a^2 & -ab & b^2 & a^3 & -a^2b & ab^2 & -b^3 & -a^3b & -ab^3 \\
0 & 0 & 1 & 0 & a & -2b & 0 & a^2 & -2ab & 3b^2 & a^3 & 3ab^2 \\
0 & -1 & 0 & -2a & b & 0 & -3a^2 & 2ab & -b^2 & 0 & 3a^2b & b^3 \\
1 & a & b & a^2 & ab & b^2 & a^3 & a^2b & ab^2 & b^3 & a^3b & ab^3 \\
0 & 0 & 1 & 0 & a & 2b & 0 & a^2 & 2ab & 3b^2 & a^3 & 3ab^2 \\
0 & -1 & 0 & -2a & -b & 0 & -3a^2 & -2ab & -b^2 & 0 & -3a^2b & -b^3 \\
1 & -a & b & a^2 & -ab & b^2 & -a^3 & a^2b & -ab^2 & b^3 & -a^3b & -ab^3 \\
0 & 0 & 1 & 0 & -a & 2b & 0 & a^2 & -2ab & 3b^2 & -a^3 & -3ab^2 \\
0 & -1 & 0 & 2a & -b & 0 & -3a^2 & 2ab & -b^2 & 0 & -3a^2b & -b^3
\end{bmatrix}$$

- According to the handout $\nabla N^e = \nabla N * C^{-1}$, where ∇N is the first derivative of N vector expressed in CALFEM :
  $$N = [1 \quad x \quad y \quad x^2 \quad xy \quad y^2 \quad x^3 \quad x^2y \quad xy^2 \quad y^3 \quad x^3y \quad xy^3]$$
  by $\nabla^T = [\frac{\partial}{\partial x} \frac{\partial}{\partial y}]$
- $\sigma = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{xy} & \sigma_{yy} \end{bmatrix}$
- t the thickness of the plate

Furthermore, we assume that:

- $\widetilde{N}^{(R)}$ is constant within the element
- Only pure buckling of the plate according to Runesson & Larsson will be studied so that the geometrical stiffness can be neglect for the plane elasticity element.
- The Gauss points and weights are the following[1] :

```
X  =[0 0.774596 0.774596]*a;
Y  =[0 0.774596 0.774596]*b;
Hx =[0.8889 0.5556   0.5556]*a;
Hy =[0.8889 0.5556   0.5556]*b;
```

  Where a = (ex(3) – ex(1))/2 and b = (ey(3) – ey(1))/2.

Influence of q_z in the reference load on the buckling analysis

Buckling is subjected to compressive stress, in our case t_y. Our analysis is run for several values of the distributed load $\boldsymbol{q_z}$ and it appears that the reference load is not affected by $\boldsymbol{q_z}$.

Global material and geometrical stiffness matrices KM and KG

Since we already compute KMe and KGe, we can assemble them respectively in KM and KG in order to solve the eigenvalue problem:

$$(KM + \lambda KG).z = 0$$

The command lines given in the handout allow us to compute:

- The eigenvalues L corresponding to the load parameter λ for each buckling mode:

$$\lambda = \frac{-1}{diag(L)}$$

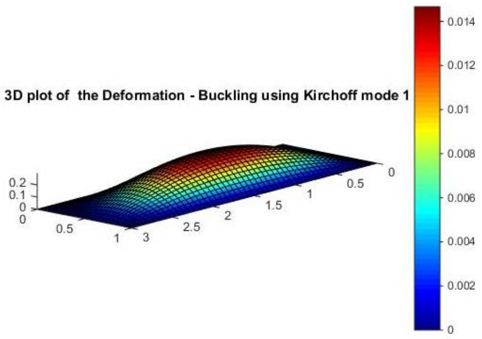
It means that the loads applied in each mode are q= λ_i*q_z and t= λ_i*t_y where q_z and t_y are the reference loads

- The eigenvectors X giving the displacements in the different buckling mode.

The critical load for buckling is λ*t_y. It is a constant. Indeed, if the reference compression load t_y is changed, then the load parameter λ changes too and keep the critical load constant.

[1] The values for the Gauss points used in the Matlab code had more numbers after zero in order to improve the simulation. Here they were simplified for reasons of clarity.

First and Second mode for Buckling



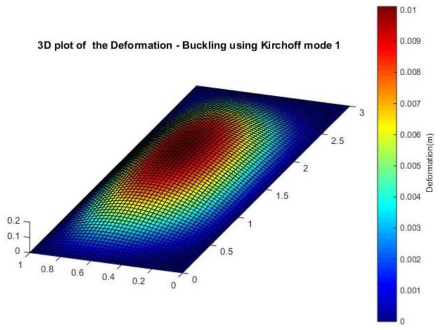


Figure 10 – First Buckling mode for $t_y = -5.106\ Nm^{-2}$. The plot on the right is for a mesh with 80x0 and on the left for a mesh with 50 x30 elements.

Here are plotted the first buckling mode (Figure 10) for two different meshes. We can see that the highest value between the two plots are not exactly the same, e.g. the values for the deformation in the coarse mesh (20x10 – see Appendix A) are a lot bigger than the ones in Figure 10 . Nevertheless, with a fine enough mesh, the behaviour of the two methods is similar, i.e. max deformation 0,014 for the 50x30 mesh while we have a value of 0,01 for the 80x50 mesh. Note that the results are quite close to the ones obtained with Abaqus (Figure 11). However, there is no traction force applied on the plate and the dimensions are not the same.

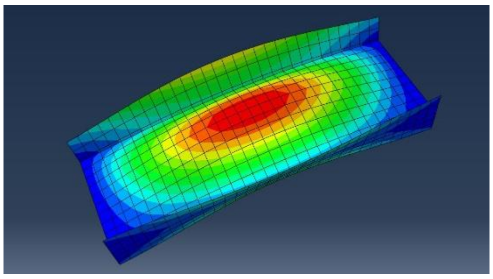
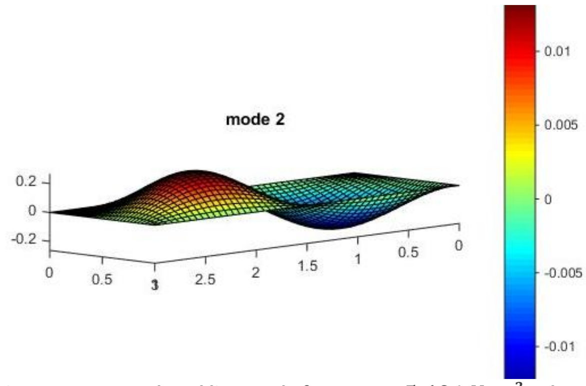


Figure 11 – First buckling mode with Abaqus



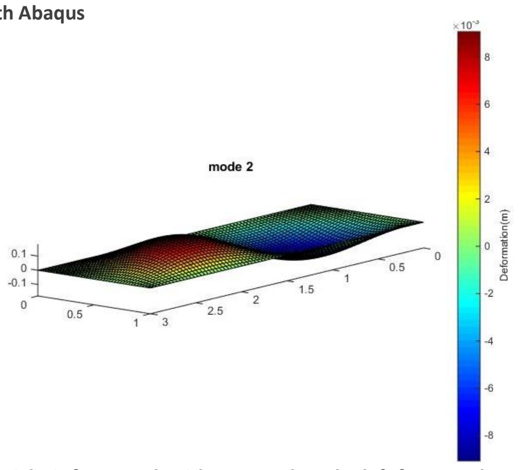


Figure 12 – Second Buckling mode for $t_y = -5.106\ Nm^{-2}$. The plot on the right is for a mesh with 80x0 and on the left for a mesh with 50 x30 elements.

The same conclusion can be made for the second buckling mode. The finest mesh gives better result.

Concerning the values for the applied loads, we need to multiply all forces by the scaling factor for each buckling mode. The scaling factor is the lambda calculated using Matlab. Therefore we will obtain the following values for the applied loads:

Table 1 – Table with the values for the applied loads for each buckling mode.

	Mode 1 - $\lambda\ = 5.1231$	Mode 2 - $\lambda\ = 7.3687$
$q_z = 10^3$	$q_{1z}\ = 5.1231.10^3$	$q_{2z}\ = 7.3687.10^3$
$t_y = 5.10^6$	$t_{1y} = 2562.10^4$	$t_{2y} = 36844.10^3$

Since the load q_z, as explained before, doesn't affect the buckling of the plate, the loading case that is critical for buckling is the one corresponding to t_y. The latter has to be scaled by λ as it was done in the previous paragraph. The buckling mode to be taken in consideration is the one with the lowest lambda, i.e. the first mode with a $\lambda\ = 5.1231$.

Conclusion

- The deflection for Kirchhoff is constant while for Mindlin it varies a lot when changing the thickness.
- Since they exhibit similar behavior for thin components, similar values of deformation are obtained for both Kirchhoff and Mindlin model (when using few integration points).
- Kirchhoff model is a good assumption for thin plate, however Midlin theory is more accurate and the FEM solution agrees with the analytical one for a thick structure. However, for the thin plate the FEM solution, when using more integration points than 2 for bending and one for shearing, in the case of Midlin, is very poor. This error occurs because the Q4 elements cannot accurately approximate the strain distribution with bending. This phenomenon is known as shear locking.

Appendix A

Here one can see the 3D deformation plot for the two buckling modes calculated using a coarse mesh, i.e.20x10.

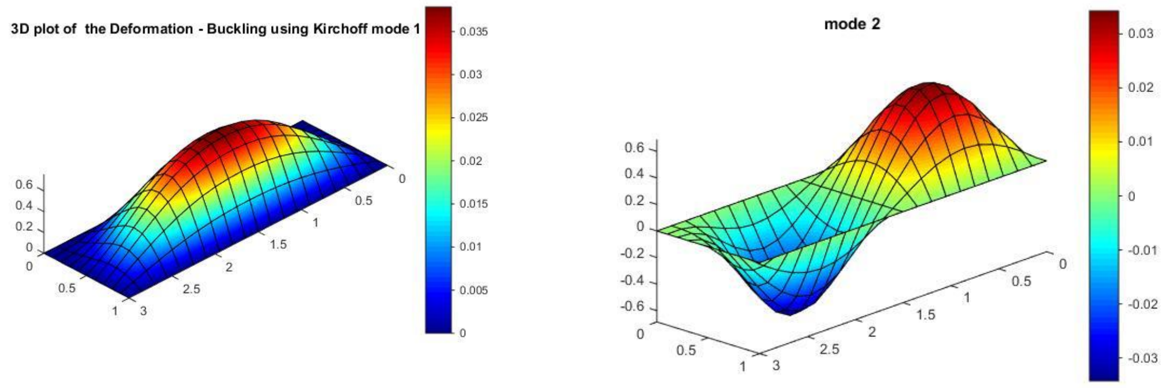


Figure 12 – First and Second Buckling mode for $t_y = -5.106\ Nm^{-2}$ using a mesh with the following dimensions: 20x10.

Appendix B

Matlab code used in this Assignment. There is a main function CA3 and 6 sub functions called/used by CA3.

```
%
%
%
%
%
%
close all
clear all

plate = 1; % Kirchoff

ptype = 1;
t = 0.008; % m
Lx = 3; % m
Ly = 1; % m
E = 210e9; % PA
v = 0.3;
qz =1000; % in Newton / m^ 2
tractiony = -5e6; % Newton / m^ 2
irb = 2;
irs = 1;
ep = [ptype t irb irs];
eq = [0 0 qz];

nx = 20;
lx = Lx/nx;
ny = 10;


D = hooke (ptype,E,v);

G = (E*eye(2,2))/(2*(1+v)); %



[ Coord, Dof, Enode, Edof, Ex, Ey, DofRight, DofTop, ...
    DofLeft, DofBottom, cdof ] = quadmesh( Lx, Ly, nx, ny, 'no' );


nelement = size(Ex,1);
ndofs = max(max(Edof));

% find the dofs for the constrained noeuds
% For the Dofs along z where we have boundary conditions along Top and
% Bottom boundaries

DofBoundary = [DofBottom(:,1);
    DofTop(:,1)];
```

```
DofBoundaryZ = DofBoundary(3:3:end,1);

constraineddofs = [DofLeft(:,1);
    DofRight(:,1);
    DofBoundaryZ(:,1)];


% Find the free dofs

fdofs = 1:ndofs;
fdofs(constraineddofs) = [];

% set the bc's

bc = zeros(length(constraineddofs),2);
bc(:,1) = constraineddofs;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculate A and Ed -Deformation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[A,Ed,F]  = displacementcalc( ndofs,nelement,Edof, Ex, Ey,  DofTop, ...
    ep,eq,D,plate,G,bc,fdofs,constraineddofs,tractiony*ep(2),lx);
% PLotting the deformations
scx = 100;
scy = 200;
scz = 200;

xdofs = [1 6 11 16];
ydofs = [2 7 12 17];
zdofs = [3 8 13 18];

figure;
Dispviz3(Ex + Ed(:,xdofs)*scx,Ey + Ed(:,ydofs)*scy,Ed(:,zdofs)*scz,Edof,A,3,2);
colorbar
shading flat
colormap(jet);
title('3D plot of  the Deformation - Kirchoff');
xlabel( colorbar,'Deformation(m)');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% stress
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

es=zeros(nelement,3);
et=zeros(nelement,3);
ef=zeros(nelement,8);
ec=zeros(nelement,3);
s_min = zeros(nelement,1);
Mxx=zeros(nelement,1);
Myy=zeros(nelement,1);
Mxy=zeros(nelement,1);



for teta=1:nelement
    [es(teta,:),et(teta,:),ef(teta,:),ec(teta,:)]=shell2rs(Ex(teta,:)...
        ,Ey(teta,:),[ep(1) ep(2)],D,Ed(teta,:));
    S=[es(teta,1) es(teta,3) 0;    %Building of the stress matrix
        es(teta,3) es(teta,2) 0;
        0 0 0];

    eigen=eig(S);            %principal stresses

    s_min(teta)=min(eigen);
    Mxx(teta)=ef(teta,1);          %value of Mxx Myy Mxy
    Myy(teta)=ef(teta,2);
    Mxy(teta)=ef(teta,3);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Second Part calculation using A and Ed / Buckling
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% We need to use kirchoff
plate2nd = 1;

% Just so we recall what are we using
ep = [ptype t irb irs];
eq = [0 0 qz];


[ X,~,posvec,freedof,~] = bucklingcalc(Ex,Ey,ep,D,es,G,plate2nd...
                                       ,eq,bc,ndofs,Edof,nelement );

% Retrieving the data for each mode usign the posvec, where we will obtain
% the position of each lambada in a crescenting order

u_mode = zeros(ndofs,6);
u_mode(freedof,:) = X(:,posvec(:,1));

Ed_mode1 = extract(Edof,u_mode(:,1));
Ed_mode2 = extract(Edof,u_mode(:,2));
Ed_mode3 = extract(Edof,u_mode(:,3));
Ed_mode4 = extract(Edof,u_mode(:,4));
Ed_mode5 = extract(Edof,u_mode(:,5));
Ed_mode6 = extract(Edof,u_mode(:,6));

% Plotting

scx = 50;
scy = 20;
scz = 20;

xdofs = [1 6 11 16];
ydofs = [2 7 12 17];
zdofs = [3 8 13 18];

figure;
Dispviz3(Ex + Ed_mode1(:,xdofs)*scx,Ey +
Ed_mode1(:,ydofs)*scy,Ed_mode1(:,zdofs)*scz,Edof,u_mode(:,1),3,2);
colorbar
colormap(jet);
title('3D plot of  the Deformation - Buckling using Kirchoff mode 1');
```

```
xlabel( colorbar,'Deformation(m)');

figure;

Dispviz3(Ex + Ed_mode2(:,xdofs)*scx,Ey +
Ed_mode2(:,ydofs)*scy,Ed_mode2(:,zdofs)*scz,Edof,u_mode(:,2),3,2);
colorbar
colormap(jet);
title('mode 2');
xlabel( colorbar,'Deformation(m)');

figure;

Dispviz3(Ex + Ed_mode3(:,xdofs)*scx,Ey +
Ed_mode3(:,ydofs)*scy,Ed_mode3(:,zdofs)*scz,Edof,u_mode(:,3),3,2);
colorbar
colormap(jet);
title(' mode 3');
xlabel( colorbar,'Deformation(m)');

figure;

Dispviz3(Ex + Ed_mode4(:,xdofs)*scx,Ey +
Ed_mode4(:,ydofs)*scy,Ed_mode4(:,zdofs)*scz,Edof,u_mode(:,4),3,2);
colorbar
colormap(jet);
title(' mode 4');
xlabel( colorbar,'Deformation(m)');

figure;

Dispviz3(Ex + Ed_mode5(:,xdofs)*scx,Ey +
Ed_mode5(:,ydofs)*scy,Ed_mode5(:,zdofs)*scz,Edof,u_mode(:,5),3,2);
colorbar
colormap(jet);
title(' mode 5');
xlabel( colorbar,'Deformation(m)');

figure;

Dispviz3(Ex + Ed_mode6(:,xdofs)*scx,Ey +
Ed_mode6(:,ydofs)*scy,Ed_mode6(:,zdofs)*scz,Edof,u_mode(:,6),3,2);
colorbar
colormap(jet);
title('6');
xlabel( colorbar,'Deformation(m)');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Last part Kirchoff vs Mindlin plate
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

deflectionKtotal = zeros(50,1);
deflectionMtotal1 = zeros(50,1); % 2 and 1
deflectionMtotal2 = zeros(50,1); % 2 and 2
deflectionMtotal3 = zeros(50,1); % 2 and 3

counter = 1;
% In order to change the thickness, it will fo from 0.001 to 0.5001 in an
% 0.01 interval

for i= 0.001:0.01:0.5001

    ep = [ptype i 2 1];

    % calculate for Kirchoff

    [ AKK,EdKK ] = deflectioncalc( ndofs,nelement, Edof, Ex, Ey,  DofTop, ...
        ep,eq,D,1,G,bc,fdofs,constraineddofs,tractiony,cdof,lx );

    deflectionKtotal(counter,1) = (i^3)*sum(EdKK);

    % Calculate for Mindling and irb = 2 nad irs = 1

    [AMM1, EdMM1 ] = deflectioncalc( ndofs,nelement, Edof, Ex, Ey,  DofTop, ...
        ep,eq,D,2,G,bc,fdofs,constraineddofs,tractiony,cdof,lx);

     deflectionMtotal1(counter,1) = (i^3)*sum(EdMM1);

    % Calculate for Mindling and irb = 2 nad irs = 2

    ep = [ptype i 2 2];

    [ AMM2,EdMM2 ] = deflectioncalc( ndofs,nelement, Edof, Ex, Ey,  DofTop, ...
        ep,eq,D,2,G,bc,fdofs,constraineddofs,tractiony,cdof,lx);

    deflectionMtotal2(counter,1) = (i^3)*sum(EdMM2);

    % Calculate for Mindling and irb = 2 nad irs = 3

    ep = [ptype i 2 3];

    [AMM3, EdMM3 ] = deflectioncalc( ndofs,nelement, Edof, Ex, Ey,  DofTop, ...
        ep,eq,D,2,G,bc,fdofs,constraineddofs,tractiony,cdof,lx);

    deflectionMtotal3(counter,1) = (i^3)*sum(EdMM3);

    counter = counter +1;
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Plotting
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% plot the moments
figure
subplot(3,1,1)
fill(Ex',Ey',Mxx');
colorbar
xlabel( colorbar,'Moment Distribution(N.m)');
title('Mxx')
subplot(3,1,2)
fill(Ex',Ey',Myy');
colorbar
xlabel( colorbar,'Moment Distribution(N.m)');
title('Myy')
```

```
subplot(3,1,3)
fill(Ex',Ey',Mxy');
colorbar
xlabel( colorbar,'Moment Distribution(N.m)');
title('Mxy')

% plot the stresses
figure
subplot(3,1,1)
fill(Ex',Ey',es(:,1)');
colorbar
xlabel( colorbar,'Stress Distribution(Pa)');
title('sigmaxx')
subplot(3,1,2)
fill(Ex',Ey',es(:,2)');
colorbar
xlabel( colorbar,'Stress Distribution(Pa)');
title('sigmayy')
subplot(3,1,3)
fill(Ex',Ey',es(:,3)');
colorbar
xlabel( colorbar,'Stress Distribution(Pa)');
title('sigmaxy')

% plot the highest compressive stress distribution
figure
fill(Ex',Ey',s_min')
colorbar
title('highest compressive stress')
xlabel( colorbar,'Stress Distribution(Pa)');
axis equal
axis([0 Lx 0 Ly])


% plot the Deflection versus thickness

thicknessvect = 0.001:0.01:0.5001;
figure


figure
plot(thicknessvect',deflectionMtotal1,'b-');
xlabel('Thickness(m)')
ylabel('Deflection(m)')
title('Thickness vs Deflection for Mindlin 2 1')
figure
plot(thicknessvect',deflectionMtotal2,'k-');
xlabel('Thickness(m)')
ylabel('Deflection(m)')
title('Thickness vs Deflection for Mindlin 2 2')
figure
plot(thicknessvect',deflectionMtotal3,'g-');
xlabel('Thickness(m)')
ylabel('Deflection(m)')
title('Thickness vs Deflection for Mindlin 2 3')
figure
plot(thicknessvect',deflectionKtotal,'r-');
xlabel('Thickness(m)')
```

```
ylabel('Deflection(m)')
title('Thickness vs Deflection for Kirchoff 2 1')

figure
hold on
plot(thicknessvect',deflectionMtotal1,'b-');
plot(thicknessvect',deflectionMtotal2,'k-');
plot(thicknessvect',deflectionMtotal3,'g-');
plot(thicknessvect',deflectionKtotal,'r-');

xlabel('Thickness(m)')
ylabel('Deflection(m)')
legend('Mindlin 2 1','Mindlin 2 2','Mindlin 2 3','Kirchoff 2 1')
title('Thickness vs Deflection')

hold off
```

Published with MATLAB® R2014b

```
function [ KMe,fe,KGe] = shell2rg( ex,ey,ep,D,es,G,plate,eq)
%Purpose : Compute the Geometrical stiffness matrix, global material matrix
%and the force fe matrix for Blucking
%
% Input
% ex = [x1 x2 x3 x4]     element coordinates
% ey = [y1 y2 y3 y4]
% ep = [ptype t ]             ptype: analysis type,    t: thickness
% G
% D                              constitutive matrix
% ed = [u1 u2 .. u8;          element displacement vector
%                 ...........]          one row for each element
% plate                       Minldlin if 1 and Kirchoff if 2
% eq =[qx qy qz]
%
% Output
% KMe = 20x20
% fe = 20x1                   Global material matrix
% KGe = 20x20                 Geometrical stiffness matrix,
% ec = [Kxx Kyy 2Kxy]     curvature

sigma = [ es(1) es(3);
    es(3) es(2)];

% C matrix

a = (1/2)*(ex(3)-ex(1));
b = (1/2)*(ey(3)-ey(1));

C = [1 -a -b   a^2   a*b   b^2      -a^3  -(a^2)*b  -(b^2)*a      -b^3      (a^3)*b     a*(b^3);
    0   0   1     0    -a   -2*b         0      (a^2)    2*b*a    3*(b^2)      -(a^3)  -3*a*(b^2);
    0  -1   0   2*a     b      0   -3*(a^2)   -2*a*b    -(b^2)         0    3*(a^2)*b       (b^3);
    1   a  -b   a^2  -a*b    b^2       a^3  -(a^2)*b   (b^2)*a      -b^3     -(a^3)*b    -a*(b^3);
    0   0   1     0     a   -2*b         0      (a^2)   -2*b*a   3*(b^2)        (a^3)   3*a*(b^2);
    0  -1   0  -2*a     b      0  -3*(a^2)     2*a*b    -(b^2)         0    3*(a^2)*b       (b^3);
    1   a   b   a^2   a*b    b^2       a^3   (a^2)*b   (b^2)*a       b^3      (a^3)*b     a*(b^3);
    0   0   1     0     a    2*b         0      (a^2)    2*b*a   3*(b^2)        (a^3)   3*a*(b^2);
    0 -1  0 -2*a   -b     0 -3*(a^2)   -2*a*b    -(b^2)       0  -3*(a^2)*b     -(b^3);
    1 -a  b  a^2 -a*b   b^2      -a^3  (a^2)*b -(b^2)*a      b^3    -(a^3)*b   -a*(b^3);
    0  0  1    0   -a   2*b         0     (a^2)  -2*b*a  3*(b^2)      -(a^3) -3*a*(b^2);
    0 -1  0  2*a   -b     0 -3*(a^2)    2*a*b    -(b^2)       0  -3*(a^2)*b     -(b^3)];

% Gauss points for a 3x3 integration

xGauss = [0 0.774596669241483  -0.774596669241483]*a;
yGauss = [0 0.774596669241483 -0.774596669241483]*b;

Hx = [0.888888888888889 0.555555555555556   0.555555555555556]*a;
Hy = [0.888888888888889 0.555555555555556   0.555555555555556]*b;

% calculating deltaN , SEE IF THE -1 IS CORRECT IN THIS WAY

KGeep = zeros(12,12);

% Calculatig the components of KGeep for each gauss points in a 3x3
% integration

for i = 1:3
    for j = 1:3

        % Calculating the delta*N

        deltaN1 = [0 1 0 2*xGauss(i) yGauss(j) 0 3*(xGauss(i)^2) 2*xGauss(i)*yGauss(j)
yGauss(j)^2 0 3*(xGauss(i)^2)*yGauss(j) yGauss(j)^3];

        deltaN2 = [0 0 1 0 xGauss(i) 2*yGauss(j) 0 (xGauss(i)^2) 2*yGauss(j)*xGauss(i)
3*(yGauss(j)^2) (xGauss(i)^3) 3*xGauss(i)*(yGauss(j)^2)];

        deltaN = [deltaN1;deltaN2];


        deltaN = deltaN*(C^-1);

        Nr = ep(2)*sigma;

        KGeep = KGeep + deltaN'*Nr*deltaN*Hx(i)*Hy(j);

    end

end

posep = [3 4 5 8 9 10 13 14 15 18 19 20];

% Putting KGeep in their proper palces
KGe(posep,posep) = KGeep;

% Calculating KMe and fe
[KMe,fe] = shell2re(ex,ey,ep,D,G,eq,plate);
```

Published with MATLAB® R2014b

```
function [ X,L,posvec,freedof,realambdaoriginal ] = bucklingcalc( Ex,Ey,ep,D,es,G...
                                                     ,plate2nd,eq,bc,ndofs,Edof,nelement )
% function[ X,L,posvec,freedof,realambdaoriginal ] = bucklingcalc( Ex,Ey,ep,D,es,G...
%          ,plate2nd,eq,bc,ndofs,Edof,nelement )
% -------------------------------------------------------------------------
% Purpose : This function will calculate the eingenvalues L, the realeingenvalues
% and the matrix X representes the eigenmodes. It will give the position of
% the real eingenvalues , ordered from the lowest to the biggest eigenvalue,
% in the posvec vector.
%
% -------------------------------------------------------------------------
% Input
%
% ndofs                     Total number of degrees of freedom
% nelement                  Total number of elements
% Ex                        Element nodal x-coordinates, where the nodes are
%                           counted anti-clockwise starting from the bottom left
%                           node of each element.
% Ey                        Element nodal y-coordinates, where the nodes are
%                           counted anti-clockwise starting from the bottom left
%                           node of each element.
% bc                        Boundary Conditions
% G
% es = [sigxx sigyy sigxy] stresses
% eq = [qx qy qz]
% ep = [ptype t irb irs]    ptype: analysis type,    t: thickness
% D                         Hook matrix
% plqte2nd = 1              1 for Kirchoff
%
% ------------------------------------------------------------------------
% Output
%
% X                  eingenvectors
% L                  enigenvalues
% posvec             vector with the positions of the real eigenvectors
%                    ordered from the smallest to the biggest
% realambdaoriginal  real eigenvectors
% freedof            free degrees of freedom




KM = zeros(ndofs,ndofs);
KG = zeros(ndofs,ndofs);
FB = zeros(ndofs, 1);
% Compute KM and KG for each element and assemble them

for gamma = 1:nelement

    elemdofs = Edof(gamma,2:end);

    [KMe,fe,KGe] = shell2rg(Ex(gamma,:),Ey(gamma,:),ep,D,es(gamma,:),G,plate2nd,eq);

    KM(elemdofs,elemdofs) = KM(elemdofs,elemdofs) + KMe;

    KG(elemdofs,elemdofs)= KG(elemdofs,elemdofs) +KGe;
    FB(elemdofs,1) = FB(elemdofs,1) + fe;

end


% Solving the eingenvalue problem

bc1 = bc(:,1);

freedof = setdiff(1:ndofs,bc1);

Kred = sparse(KM(freedof,freedof));
Gred = sparse(KG(freedof,freedof));

Knew = Kred\Gred;

[X,L] = eigs(Knew);

realambda = -1./diag(L);

realambdaoriginal = zeros(1,length(realambda));
posvec = zeros(length(realambda),1);


% Recovering the position and ordering it from the minimum to the biggest
% lambda

for u = 1:length(realambda)

    lambdamin = min(abs(realambda));
    posmin = find(realambda == lambdamin);
    realambdaoriginal(1,u) = realambda(posmin);
    realambda(posmin) = [];
    posvec(u,1) = posmin + (u-1);
end
```

Published with MATLAB® R2014b

```
function [ A, Ed ,F] = displacementcalc(ndofs,nelement, Edof, Ex, Ey,  DofTop, ...
                                      ep,eq,D,plate,G,bc,fdofs,constraineddofs,tractiony,Lx)
% function [ A, Ed ,F] = displacementcalc(ndofs,nelement, Edof, Ex, Ey,  DofTop, ...
% ep,eq,D,plate,G,bc,fdofs,constraineddofs,tractiony,Lx)
%
% -------------------------------------------------------------------------
% Purpose : This function will calculate the Displacement for a give
% thickness and stress. Mindlin and Kirchoff methods can be used
%
% -------------------------------------------------------------------------
% Input
%
% ndofs                      Total number of degrees of freedom
% nelement                   Total number of elements
% Edof                       Topology matrix in terms of dofs, size = (NoElem x 21)
%                            In each row in Edof, the dofs are arranged
%                           anti-clockwise starting from the bottom left node
%                           of each element (5 dofs/node). For each node, first
%                           the three translational dofs are counted (ux,uy,uz)
%                           and then the two rotational (theta_x,theta_y)
% Ex                        Element nodal x-coordinates, where the nodes are
%                           counted anti-clockwise starting from the bottom left
%                           node of each element.
% Ey                        Element nodal y-coordinates, where the nodes are
%                           counted anti-clockwise starting from the bottom left
%                           node of each element.
% DofTop                    Translational dofs (ux,uy,uz) at the top boundary,
%                           size = (NoNodesRightBoundary x 3)
% fdofs                     Free degrees of freedom
% bc                        Boundary Conditions
% G
% constraineddofs           Contrained degrees of freedom
% eq = [qx qy qz]
% ep = [ptype t irb irs]    ptype: analysis type,    t: thickness
% D                         Hook matrix
% tractiony                 Traction along y direction
% cdof                      Dofs at the centermost point
% Lx                        Length along x of each element
%
% -----------------------------------------------------------------------
% Output
%
% Ed = extract(Edof,AMM);
% A = ndofs x 1           Displacements
% F

A = zeros(ndofs,1);

K = zeros(ndofs,ndofs);
F = zeros(ndofs,1);

for n = 1:nelement

    elemdofs = Edof(n,2:end);
    [Ke,Fe ] = shell2re(Ex(n,:),Ey(n,:),ep,D,G,eq,plate);

    K(elemdofs,elemdofs) = K(elemdofs,elemdofs) + Ke;
    F(elemdofs) = F(elemdofs) + Fe;

end


% tractionydof = (tractiony)/length(DofTop);

F(DofTop(5:3:(end-3),1)) = F(DofTop(5:3:(end-3),1))  + ((tractiony*(Lx)));
F(DofTop(2:(end-3):end,1)) = F(DofTop(2:(end-3):end,1)) + ((tractiony*Lx)/2);


A(fdofs) = (K(fdofs,fdofs))\(F(fdofs) - K(fdofs,constraineddofs)*bc(:,2));

Ed = extract(Edof,A);

end
```

Published with MATLAB® R2014b

```
function [AMM, Ed ] = deflectioncalc( ndofs,nelement, Edof, Ex, Ey,  DofTop, ...
    ep,eq,D,plate,G,bc,fdofs,constraineddofs,tractiony,cdof,Lx )
% function [AMM, Ed ] = deflectioncalc( ndofs,nelement, Edof, Ex, Ey,  DofTop,
% ep,eq,D,plate,G,bc,fdofs,constraineddofs,tractiony,cdof,Lx )
%
% -------------------------------------------------------------------------
% Purpose : This function will calculate the Deflection for a give thickness
% and stress for the centermost degrees of freedom (cdof)
%
% -------------------------------------------------------------------------
% Input
%
% ndofs                       Total number of degrees of freedom
% nelement                    Total number of elements
% Edof                        Topology matrix in terms of dofs, size = (NoElem x 21)
%                             In each row in Edof, the dofs are arranged
%                             anti-clockwise starting from the bottom left node
%                             of each element (5 dofs/node). For each node, first
%                             the three translational dofs are counted (ux,uy,uz)
%                             and then the two rotational (theta_x,theta_y)
% Ex                          Element nodal x-coordinates, where the nodes are
%                             counted anti-clockwise starting from the bottom left
%                             node of each element.
% Ey                          Element nodal y-coordinates, where the nodes are
%                             counted anti-clockwise starting from the bottom left
%                             node of each element.
% DofTop                      Translational dofs (ux,uy,uz) at the top boundary,
%                             size = (NoNodesRightBoundary x 3)
% fdofs                       Free degrees of freedom
% bc                          Boundary Conditions
% G
% constraineddofs             Contrained degrees of freedom
% eq = [qx qy qz]
% ep = [ptype t irb irs]      ptype: analysis type,    t: thickness
% D                           Hook matrix
% tractiony                   Traction along y direction
% cdof                        Dofs at the centermost point
% Lx                          Length along x of each element
% plate                       Kirchoff = 1 Mindlin = 2
%
% ------------------------------------------------------------------------
% Output
%
% Ed = extract(Edof,AMM);
% AMM = ndofs x 1             Displacements


[AM,~,~]  = displacementcalc( ndofs,nelement,Edof, Ex, Ey,  DofTop, ...
    ep,eq,D,plate,G,bc,fdofs,constraineddofs,tractiony*ep(2),Lx);

AMM = zeros(ndofs,1);
AMM(cdof) = AM(cdof);

Ed = extract(Edof,AMM);
Ed = Ed(:,3:5:end);

% Since I only calculated the displacements for cdof
poszeros  = find(Ed == 0);

Ed(poszeros) = [];

end
```

Published with MATLAB® R2014b

```
function [ Ke,Fe ] = shell2re(ex,ey,ep,D,G,eq,plate)
%
% Purpose : Calculate the Element stiffness matrix and the Element load
% vector
%
% --------------------------------------------------------
% OUTPUT -
%
% Ke = Element stiffness matrix (20 x 20)
% Fe = ELement load vector (20 x 1)
%
% INPUT -
% ex = [x1 x2 x3 x4] X -Coordinates
% ey = [x1 x2 x3 x4] Y- coordinates
% ep = [ptype,t,irb,irs] - ptype = 1 gives plane stress
%                                  2 gives plane strain
%                          t thickness of the plate
%                          irb and irs integration rules for bending and
%                          shearing respectively (only for Mindlin plate)
% D = Constitutive matrix
% G = Constitutive matrix for bending mode (Mindlin plate)
% eq = [bx,by,qz]
% plate = 1 for Kirchoff plate and 2 for Mindlin plate

Ke = zeros (20,20);
Fe = zeros (20,1);
[keplanre,feplanre] = planre ([ex(1) ex(3)],[ey(1) ey(3)],[ep(1) ep(2)],D,[eq(1) eq(2)]);

% TO swicth betwehn Krichof and Mildlin theory
switch plate

    case 1 % Kirchoff theory
        [kep,fep] = platre (ex,ey,ep(2),D,eq(3));

    case 2 % Mindlin theory
        [kep,fep] = platme (ex,ey,[ep(2) ep(3) ep(4)],D,G,eq(3));
end

% To add Fe and Ke the good position - Easier way exits

posplanre = [1 2 6 7 11 12 16 17];
posep = [3 4 5 8 9 10 13 14 15 18 19 20];

Ke(posep,posep) = kep;
Ke(posplanre,posplanre) = keplanre;
```

```
Fe(posep) = fep ;
Fe(posplanre) =feplanre;

end
```

Published with MATLAB® R2014b

```
function [es,et,ef,ec] = shell2rs(ex,ey,ep,D,ed)

% Purpose : Compute element stresses, strains, forces and curvature
%
% Input
% ex = [x1 x2 x3 x4]     element coordinates
% ey = [y1 y2 y3 y4]
% ep = [ptype t ]             ptype: analysis type,    t: thickness
% D                           constitutive matrix
% ed = [u1 u2 .. u8;          element displacement vector
%                ...........]          one row for each element
%
% Output
% es = [sigxx sigyy sigxy]    stresses
% et = [epsxx epsyy gamxy]     strains
% ef = [ Mxx Myy Mxy Vxz Vyz] section forces
% ec = [Kxx Kyy 2Kxy]     curvature


% element force matrix and curvature with platrs
[ef,ec]=platrs(ex,ey,ep(2),D,ed([3 4 5 8 9 10 13 14 15 18 19 20]));

% element stresses and strains with planrs
[es,et]=planrs([ex(1) ex(3)],[ey(1) ey(3)],[ep(1) ep(2)],D,ed([1 2 6 7 11 12 16 17]));



end
```

Published with MATLAB® R2014b