



Non Linear Geometric Elastic Analysis of Thin Walled Beam by the Finite Strip Element Method

Master thesis

Faculté Polytechnique de Mons

Alejandro Lifante Mira



Supervised by :

Dr. Prof. IrSélimDatoussaïd, FacultéPolytechnique de Mons

Contents

1. Introduction.....	3
1.1 Background.....	3
1.2. Objectives and scope	3
2. Previous theories.....	5
2.1. Theoretical background	5
2.2. Existing methods	5
2.2.1.Exact methods.....	5
2.2.2. Approximate methods	6
2.2.3. Finite element method.....	6
2.3. Comparative table between FSM and FEM.....	7
3. Finite Element Method (FEM).....	8
3.1. History	8
3.1.2. Creation of the Method	8
3.1.3. Practical use of the method through the centuries	8
3.3. FEM example.....	9
3.3.1. Situation	9
3.3.2 Element solving	10
4. Finite Strip Method (FSM).....	16
4.1. History	16
4.2. Explanation.....	16
4.3. The finite strip analysis.....	17
4.3.1. Degree of freedom and shape functions	17
4.3.2. Elastic stiffness matrix.....	21
4.3.3. Geometric stiffness matrix.....	28
4.3.4. Assembly of the elements.....	31
4.4. Buckling modes	33
4.4.1. Local buckling	33
4.4.2. Distortional buckling	34
4.4.3. Global buckling.....	34
4.4.4. Generalized Beam Theory	35
5. Matlab program	36
5.1. Main program.....	36
4.2. Sub-programs.....	39

4.2.1. Local elastic matrix sub-program	39
4.2.2. Local geometric matrix sub-program	40
4.2.3. Boundary conditions calculator	42
4.2.4. Element properties.....	43
4.2.5. Rotation.....	43
4.2.6. Assembly	44
5.3. Matlab function map.....	45
6. Comparison with other methods	46
6.1. Theoretical comparison.....	46
6.2. Finite Element Method Comparison	48
7. Conclusions.....	49
8. Further work.....	50
9. Bibliography	51
Special thanks.....	51
Annex	52
1. Matlab programs.....	52
1.1. FSMsolver	52
1.2. K_elastic_local.....	53
1.3. K_geometric_local.....	55
1.4. elemprop	56
1.5. BCparameters.....	57
1.6. rotatestrip	59
1.7. assemble_elements.....	60
2. Comparison exercises.....	64
2.1. Theoretical comparison exercise	64
2.2. Finite Element Method Comparison exercise	66

1. Introduction

1.1 Background

The application of numerical methods as a tool to solve and analyze structural problems is widely used at the moment. Lately the Finite Element Method (FEM) has heavily dominated this field; however, other methods such as the Finite Strip Method (FSM) still have their application in some structural problems and have not been discarded in their areas.

A lot of structures maintain geometrical properties constant along an axis, for example plates or bridges. This kind of structures has a transversal section that does not vary in the longitudinal direction, and, if the material properties remain constant as well in the given direction, the analysis can be simplified combining the Finite Element Method (FEM) with developments of the Fourier series to model the behavior.

Even though the FEM is a very versatile tool, it requires discretization in every dimension of the problem, and generally, this implies a lot of unknown variables. The computational advance has given a solution to problems that before seemed untreatable because of their complexity and dimension.

Although the problems have been finally solved with FEM, the cost of these solutions is still computationally extremely high. Furthermore, the availability of computers capable of doing this analysis is still limited to a few privileged people, and this is even more noticeable in developing countries. In those countries, very often analysis have to be carried out in much less powerful machines.

On the other hand, in structural problems with regular geometrical forms and boundary conditions, a complex analysis with finite elements is unnecessary and extravagant. In this situation, to reduce the computational cost and requirements, alternative methods have been developed.

The Finite Strip Method (FSM) is one of the alternative methods. It consists in reducing the problem to a bi-dimensional problem in which longitudinal and transversal directions are separated. Trigonometrical differential functions are used in the longitudinal direction while simple polynomial functions are used in the other directions.

1.2. Objectives and scope

Analysis of structures composed by thin plates such as beams used for prismatic structures, some kinds of plates or even bridges can be done using FSM.

In all the mentioned structures mechanical and material properties are constant in both the transversal and longitudinal direction, and so the problem can be simplified combining FEM and Fourier series to model the transversal and longitudinal behavior.

The main objective of this project is to develop a program capable of using FSM to analyze different kinds of structures. However, in order to create and use this software it is crucial to have a complete understanding of both the structural problem and all the factors that affect this analysis. Therefore, in the project there will also be developed the whole theory behind the Finite Strip Method.

Main objectives

- Develop a program capable of analyzing beams using the Finite Strip Method.
- Create a guide and several examples to understand the functionality and how to use the developed program

Secondary objectives

- Describe the Finite Element Method for straight thin plates.
- Compare the obtained results with the FSM to those that would be obtained with FEM.

2. Previous theories

2.1. Theoretical background

The plate calculus has its origin in the works realized by Euler in the XIII century. It is from this works that the plate theory has been developed [1]*. In order to understand the deformation of plates, the main hypothesis classified plates in both thin and thick plates.

Thin plates accept the hypothesis of Kirchhoff that despises the deformation due to shear strains. It is because of this that we can express the partial differential equations of equilibrium in function of only the deformation. This means that the hypothesis establishes that points that were initially in any normal line to the plain of the plate will remain in the same normal line after the deformation. The next image illustrates this:

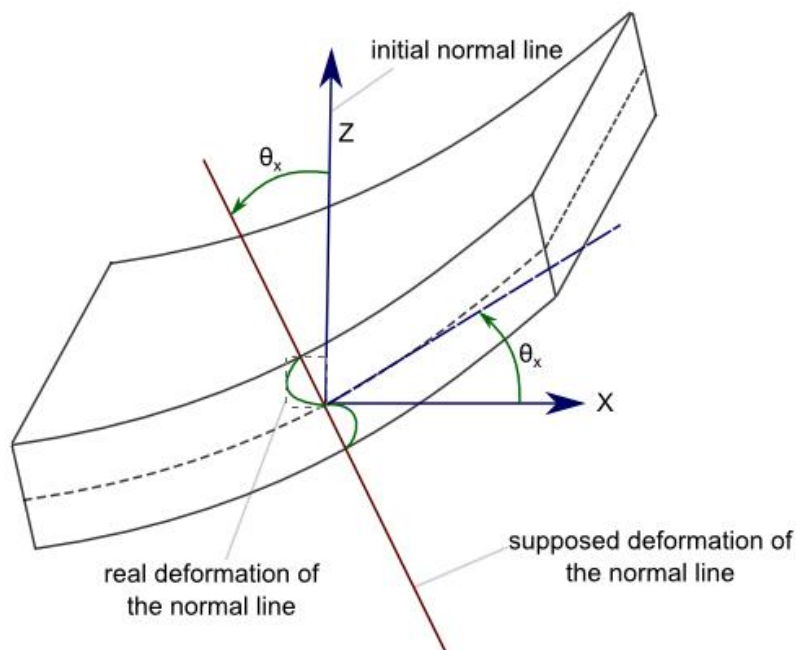


Figure 1: Kirchhoff deformation hypothesis image

2.2. Existing methods

2.2.1.Exact methods

Exact methods are only viable in a very limited quantity of cases; the solution of differential equations in partial derivation is only exact when they generate ordinary differential equations. This only happens with very specific geometries, such as circular thin plates with axial symmetry.

*[1]: Reference to bibliography

2.2.2. Approximate methods

The approximate methods are characterized because they are fast and reliable, and although they do not give an exact solution, they allow us to find sufficiently approximate solutions for the structural problems.

There are several approximate methods. In the case of rectangular plates with uniform loading there is a method created by Grashof and improved by Marcus based in superficies of influence obtained by analytical or experimental methods [1]*.

The solution by series developments has been another procedure heavily used in a lot of thin plate exercises. Timoshenko analyses a great number of plates using this method.

2.2.3. Finite element method

This method is actually part of the approximate methods, but it is the most versatile tool when solving plate exercises. The solving of these problems is based in the hypothesis in the turning of the normal lines to the middle plain and two theories exist. The thin plate theory of Kirchhoff establishes that the normal lines to middle plain stay orthogonal to the deformed form of the middle plain, which allows us to despise the shear strain deformation. The Reissner-Mindlin theory keeps the condition of the straight deformation of the normal line but does not demand the orthogonality after the deformation [2]*.

The FEM requires discretization in every dimension of the exercise, and so requires many more variables than other approximate methods. If the geometrical and mechanical properties are constant in a direction, like in plates or prismatic structures, in which the transversal section does not vary in the longitudinal direction, the analysis can be simplified. Combining the FEM and Fourier series we can solve this kind of structures. This procedure is known as the Finite Strip Method (FSM).

**[1]: Reference to bibliography*

**[2]: Reference to bibliography*

2.3. Comparative table between FSM and FEM

FEM	FSM
Applicable to any type of geometry, boundary conditions or material variation. Extremely powerful and usable in nearly every case.	In static analysis it is used for structures with two opposite simply supported ends. In dynamic analysis it is used with all boundary conditions and with discrete supports.
Implies a great number of equations and extremely big matrix. Can be very expensive and even impossible to use sometimes because of the demanding computing facilities.	Usually has a much smaller quantity of equations and matrix are also smaller. This leads to a much shorter computing time to find a solution with nearly the same accuracy.
Large quantity of input data which can lead to mistakes.	Very small amount of input data due to the smaller number of meshing.
Large quantity of output. Normally displacements of all the nodes are listed.	Easier to specify only those nodes which displacements and stresses are required.
Difficult to program and a very big computational requirement.	Due to the reduction in the number of degrees of freedom, the computational requirements are smaller.

Table 1: Comparison between the Finite Element Method and the Finite Strip Method

3. Finite Element Method (FEM)

As previously stated, finite element method is a part of the approximate methods, and it remains the most versatile tool for solving plate exercises.

3.1. History

3.1.2. Creation of the Method

The Finite Element Method was firstly developed in 1943 by Richard Courant, who used the Ritz method of numerical analysis and minimization of calculus variables to obtain approximate solutions of a vibration system. A little bit later, in 1956, a document published by several scientists established a more wide definition of numerical analysis [1]*.

Although there were different approaches depending on the pioneer who developed the method, all finite element method developments shared one characteristic: mesh discretization of bigger elements into a set of discrete subdomains, usually called elements. After the definition of discretization, an equation system was created to apply the equilibrium equations to every node of every element of the structure. The equations system can be written in the next way:

$$f = K \cdot u$$

Where the unknowns are the displacement of the nodes (u) and they can be found with the forces in the nodes (f) and the stiffness matrix (K).

3.1.3. Practical use of the method through the centuries

With the arrival of the first computers in the 1950s decade, the structures calculus was found in a point where most of the techniques used consisted in iterative methods (like Cross and Kani) which were realized manually and therefore resulted quite tedious. The calculus of a building with several floors could take weeks, which in the end supposed a big cost of time. The arrival of computers allowed for the resurgence of methods for displacement known of previous centuries (Navier, Lagrange, Cauchy) but were too difficult to apply given that the use of them lead to the resolution of enormous equations systems very difficult to approach from the manual point of view.

Between 1960 and 1970 the application of finite elements kept growing, and the requirements for calculus time and memory of the computers also grew. At this point the creation of less demanding and more efficient algorithms became important. In order to solve the equations systems the already know algorithms (Gauss, Cholesky, Crout, etc..) are adapted. It is at this point that the matricial method starts to extend. The development of this method becomes especially known in the structures where discretization into bars is extremely easy [1]*.

**[1]: Reference to bibliography*

However, even though the modeling using bars starts to get developed, there is a great difficulty to solve continuous structures (surfaces and volumes) with complex geometries. It is particularly in the aerospace camp where new FEM techniques are developed.

In the 1970 there is a great deal of development and the method starts to apply to other problems such as the nonlinear. During this decade, FEM was limited to expensive computers and therefore to rich industries such as defense, nuclear, automation or aeronautic.

It is after 1980 that the method finally reaches particular computers, and the use of commercial programs that use this method is extended.

Up until today, the FEM system has acquired a lot of importance due to the great increase in computer capabilities and the reduced economic cost of these. Today, supercomputers are capable of giving exact results for nearly all kind of parameters.

3.3. FEM example

The Finite Element Method (FEM) is a numerical technique used to find approximate solutions to boundary value problems for partial differential equations. The method uses variational methods to minimize an error function and create solutions. To understand the concept, the idea is analogous to saying that many tiny lines can approximate a large circle, although we will never get the exact circle. It is based on the discretization (division in smaller elements), which uses many simple element equations over many small subdomains, named finite elements, to approximate a solution over a very complex equation in a much larger domain.

3.3.1. Situation

There is no best way to explain a method than using an example. In the finite element method we take a structure and divide it into smaller elements, using several nodes for it. In our example a simple beam will be taken, with a force applied in the end.

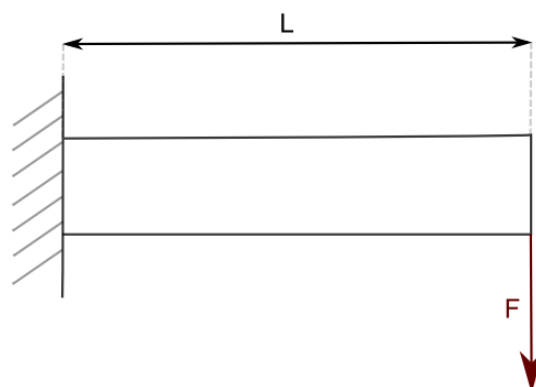


Figure 2: Initial situation

Now, taking the figure into account, we must discretize in order to have the figure divided into different elements. In this example I will discretize it to have two triangular elements.

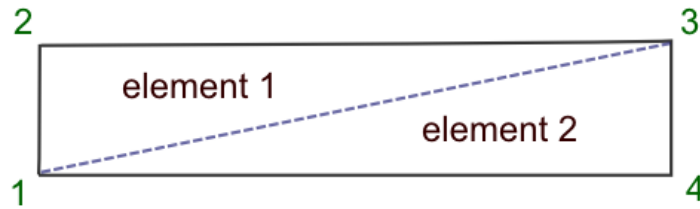


Figure 3: Discretization

So we can see that the figure has been discretized into two elements and four nodes. From now on we will work in the triangular element.

3.3.2 Element solving

3.3.2.1. Displacement functions

We have taken a simple triangular element. In this exercise, we will be working in two dimensions, but the finite element method can be used in three dimensions all the same.

We have then displacements in both the “x” and “y” direction for every node, as displayed in the next figure. We name “u” the displacement in the “x” direction and “v” the displacement in the “y” direction.

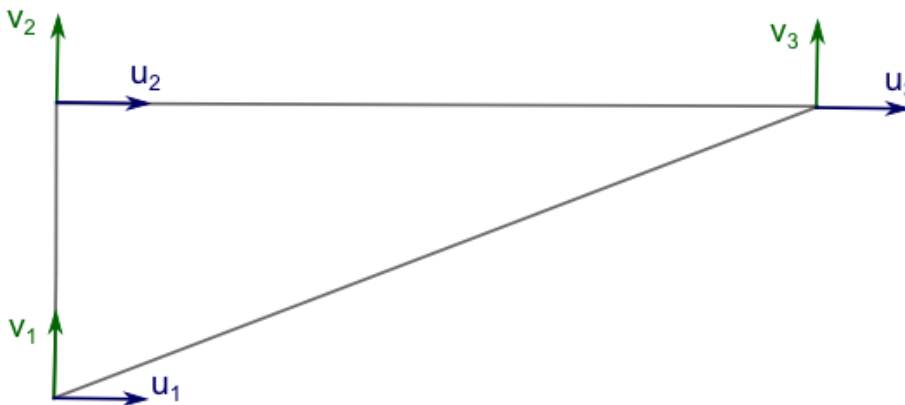


Figure 4: Displacements for every node

These displacements can be written in vector forms as:

$$\begin{Bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \end{Bmatrix} = \begin{Bmatrix} a_1 \\ a_2 \\ a_3 \end{Bmatrix}$$

Where each “a” will correspond to the horizontal and vertical displacement of each node. It is a two vector variable with “u” and “v” inside it.

To know the displacement in any point of the triangle, we create an interpolation function to interpolate the results in each node in order to find the displacement in a given “x, y” coordinate.

$$\begin{aligned}\tilde{u}(x, y) &= N_1(x, y) \cdot u_1 + N_2(x, y) \cdot u_2 + N_3(x, y) \cdot u_3 = \sum_{i=0}^3 N_i(x, y) \cdot u_i \\ \tilde{v}(x, y) &= N_1(x, y) \cdot v_1 + N_2(x, y) \cdot v_2 + N_3(x, y) \cdot v_3 = \sum_{i=0}^3 N_i(x, y) \cdot v_i\end{aligned}$$

Given this equations, we now have to find the interpolation functions N. First we will write the interpolation function as a function depending of both “x” and “y”:

$$\begin{aligned}N_1(x, y) &= \alpha_1 \cdot x + \beta_1 \cdot y + \gamma_1 \\ N_2(x, y) &= \alpha_2 \cdot x + \beta_2 \cdot y + \gamma_2 \\ N_3(x, y) &= \alpha_3 \cdot x + \beta_3 \cdot y + \gamma_3\end{aligned}$$

Once we have this functions, in order to find the coefficients α, β, γ we use the boundary conditions. The next conditions have to be met:

$$\tilde{u}(x_1, y_1) = u_1 \rightarrow N_1(x_1, y_1) = 1 ; N_2(x_1, y_1) = 0 ; N_3(x_1, y_1) = 0$$

$$\tilde{u}(x_2, y_2) = u_2 \rightarrow N_1(x_2, y_2) = 0 ; N_2(x_2, y_2) = 1 ; N_3(x_2, y_2) = 0$$

$$\tilde{u}(x_3, y_3) = u_3 \rightarrow N_1(x_3, y_3) = 0 ; N_2(x_3, y_3) = 0 ; N_3(x_3, y_3) = 1$$

Note that the “x” and “y” are coordinates while “u” and “v” are displacements of those coordinates. We then have 3 equations with 3 unknowns for every interpolation function N. Solving this equations we find that:

$$N_1(x, y) = \frac{1}{2 \cdot \Delta} \cdot [(y_2 - y_3) \cdot x + (x_3 - x_2) \cdot y + (x_2 \cdot y_3 - x_3 \cdot y_2)]$$

$$N_2(x, y) = \frac{1}{2 \cdot \Delta} \cdot [(y_3 - y_1) \cdot x + (x_1 - x_3) \cdot y + (x_1 \cdot y_3 - x_3 \cdot y_1)]$$

$$N_3(x, y) = \frac{1}{2 \cdot \Delta} \cdot [(y_2 - y_1) \cdot x + (x_2 - x_1) \cdot y + (x_1 \cdot y_2 - x_2 \cdot y_1)]$$

Where $2 \cdot \Delta$ is two times the area of the triangle defined as:

$$2 \cdot \Delta = \det \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{bmatrix} = 2 \cdot \text{Area of the triangle}$$

Finally, having all these parameters, we can define our displacement vector as:

$$\begin{Bmatrix} \tilde{u} \\ \tilde{v} \end{Bmatrix} = [N] \cdot \{a\}$$

Where “N” is our interpolation matrix and “a” the deformation of known nodes. Expanding this last equation we would find:

$$\begin{Bmatrix} \tilde{u} \\ \tilde{v} \end{Bmatrix} = \begin{bmatrix} N_1 & 0 & N_2 & 0 & N_3 & 0 \\ 0 & N_1 & 0 & N_2 & 0 & N_3 \end{bmatrix} \cdot \begin{Bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \end{Bmatrix} = [N] \cdot \{a\}$$

3.3.2.2. Deformation

The deformation within a point of our element is defined by the three components that contribute to the internal work of the very element. We therefore have the next equations, where only the linear terms are taken into account:

$$\tilde{\epsilon}_x = \frac{\delta \tilde{u}}{\delta x} ; \tilde{\epsilon}_y = \frac{\delta \tilde{v}}{\delta y} ; \tilde{\gamma}_{xy} = \frac{\delta \tilde{u}}{\delta y} + \frac{\delta \tilde{v}}{\delta x}$$

Therefore, in matricidal form, we can define the deformation as:

$$\begin{Bmatrix} \tilde{\epsilon}_x \\ \tilde{\epsilon}_y \\ \tilde{\gamma}_{xy} \end{Bmatrix} = \underbrace{\begin{bmatrix} \frac{\delta}{\delta x} & 0 \\ 0 & \frac{\delta}{\delta y} \\ \frac{\delta}{\delta y} & \frac{\delta}{\delta x} \end{bmatrix}}_L \cdot \underbrace{\begin{bmatrix} N_1 & 0 & N_2 & 0 & N_3 & 0 \\ 0 & N_1 & 0 & N_2 & 0 & N_3 \end{bmatrix}}_N \cdot \underbrace{\begin{Bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \end{Bmatrix}}_a$$

It is often that matrix B is also defined:

$$[B] = [L] \cdot [N]$$

Once we have our matrix defined we must relate them to the force applied to the object. In order to do that it is crucial that the equilibrium equations of static are understood correctly.

3.3.2.3. Equilibrium

Supposing a simple solid we can find the force applied to it (b) and the tensions within the solid ($\sigma_x, \sigma_y, \gamma_{xy}$).

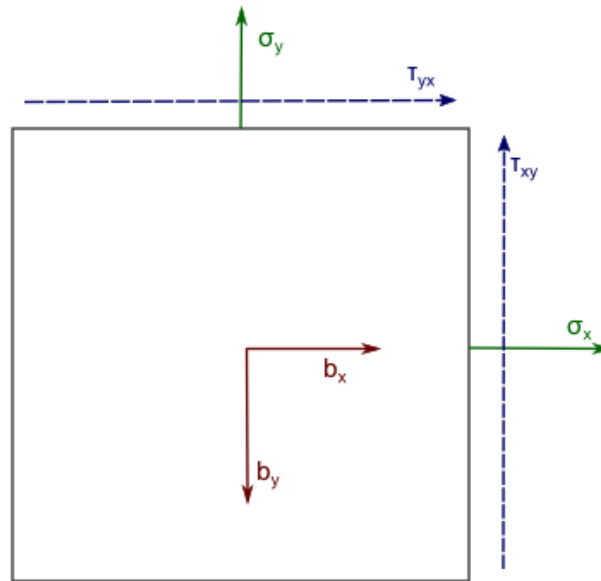


Figure 5: Solid in equilibrium

Seeing all the forces in our system applied, we can define the equilibrium equations as:

$$\frac{\delta \sigma_x}{\delta x} + \frac{\delta \tau_{xy}}{\delta y} + b_x = 0$$

$$\frac{\delta \sigma_y}{\delta y} + \frac{\delta \tau_{yx}}{\delta x} + b_y = 0$$

Having found the deformation in the previous step, the relation between tensions and deformations for the linear elastic behavior can be written like:

$$\varepsilon_x = \frac{\sigma_x}{E} - \frac{\sigma_y \cdot \nu}{E}$$

$$\varepsilon_y = \frac{\sigma_y}{E} - \frac{\sigma_x \cdot \nu}{E}$$

$$\gamma_{xy} = \frac{\tau_{xy}}{G}$$

Where we can define $G = \frac{2 \cdot (1 + \nu)}{E}$. In order to solve our problem, we need to have the expression $\tilde{\sigma} = D \cdot \tilde{\varepsilon}$. Using the last equations we find that:

$$\tilde{\sigma} = \begin{Bmatrix} \tilde{\sigma}_x \\ \tilde{\sigma}_y \\ \tilde{\tau}_{xy} \end{Bmatrix} = \frac{E}{1-\nu^2} \cdot \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{E} \end{bmatrix} \cdot \begin{Bmatrix} \tilde{\epsilon}_x \\ \tilde{\epsilon}_y \\ \tilde{\gamma}_{xy} \end{Bmatrix}$$

Once we have this it is time to define a new concept, the “**virtual work**”. This concept refer to the fact that for every force applied, the object gets an external force which is countered by an internal force created by the deformations in the object.

We can define the external and the internal energy as:

$$internal\ energy = \int_V \{\sigma\}^T \cdot \{\epsilon\} \cdot dV = \int_V \{\epsilon\}^T \cdot \{\sigma\} \cdot dV$$

$$external\ energy = \int_V \{N \cdot a\}^T \cdot \{b\} \cdot dV$$

Thanks to the theory of the virtual works we know that these energies have to counter themselves, so:

$$\int_V \{\epsilon\}^T \cdot \{\sigma\} \cdot dV - \int_V \{N \cdot a\}^T \cdot \{b\} \cdot dV = 0$$

We can now define the deformation and tension as:

$$\epsilon = B \cdot a$$

$$\sigma = D \cdot B \cdot a$$

We then have:

$$\int_V \{B\}^T \cdot \{a\}^T \cdot \{D\} \cdot \{B\} \cdot \{a\} \cdot dV - \int_V \{a\}^T \cdot \{N\}^T \cdot \{b\} \cdot dV = 0$$

Given that the vector $\{a\}$ does not depend on the volume and that none of the variables above depend on the thickness (they only depend on the area), we can write the past equation as:

$$\underbrace{\left[t \int_A \{B\}^T \cdot \{D\} \cdot \{B\} \cdot dA \right]}_{\text{Stiffness matrix (K)}} \cdot \{a\} - \underbrace{\left[t \int_A \{N\}^T \cdot \{b\} \cdot dA \right]}_{\text{Force vector (B)}} = 0$$

Finally we usually write the past equation as:

$$K \cdot a - B = [K] \cdot \{a\} - \{B\} = 0$$

This so-called “standard discrete system” is solved for the whole structure, taking into account the boundary conditions.

4. Finite Strip Method (FSM)

4.1. History

The finite strip method was created for structural analysis in the late 1960s. It was applied to several structures such as bridge, tall buildings, plates or shells. The displacements were described by functions which are given as products of trigonometrical/hyperbolic series and polynomials. This series should at first satisfy the boundary conditions at the end of the strips. [4]*

Although the technique is much less powerful and versatile than the Finite Element Method, it is true that it is much more efficient in computation terms in many situations. It is for this reason that many times it is more appropriate to use this method.

4.2. Explanation

The finite element method requires a discretization in every dimension of the problem, and therefore it requires many more unknowns than other methods. If the structures have geometrical and mechanical properties constant in one of the directions and the transversal section does not vary in the same direction, then the problem can be simplified. In structures such as plates or prismatic structures which satisfy the previous statements, the finite element method can be combined with Fourier series to model the transversal and longitudinal behavior.

In the finite strip method, the Fourier series are used to express the behavior of the longitudinal variables while the transversal direction is modeled with the finite element method. This allows eliminating unknowns associated to the longitudinal direction and solving the problem by solving the one-dimensional finite element method problem where only unknowns associated to the transversal direction discretization intervene.

**[4]: Reference to bibliography*

4.3. The finite strip analysis

4.3.1. Degree of freedom and shape functions

As previously stated, in the finite strip method, the structure is only discretized in the cross-section. The other dimension is usually represented using a shape trigonometrical function. In the next figure we can see the axis taken for the exercise. Local coordinates are named with small letters (x - y - z) and will always be associated with the strip element.

Displacements are represented with the translation U - V - W and the rotation θ for global displacements and u - v - w and φ for local displacements. The subscript p refers to the half-wave number (number of longitudinal terms).

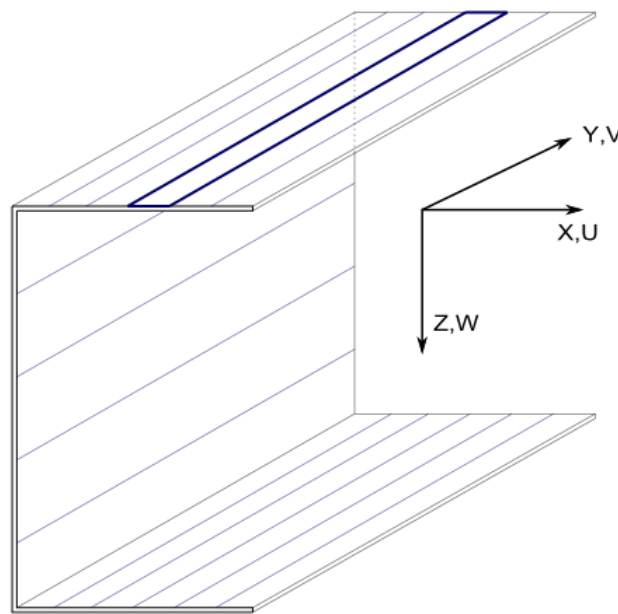


Figure 6: Strip situation

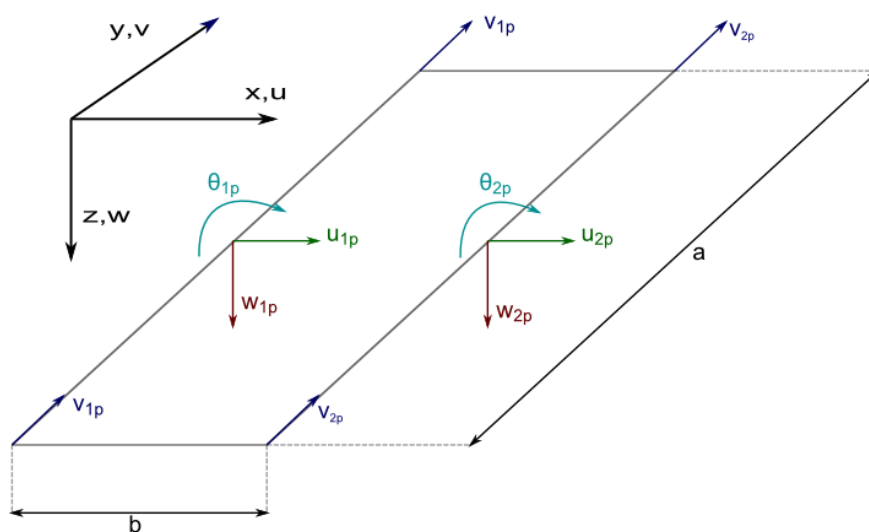


Figure 7: Strip degrees of freedom definition

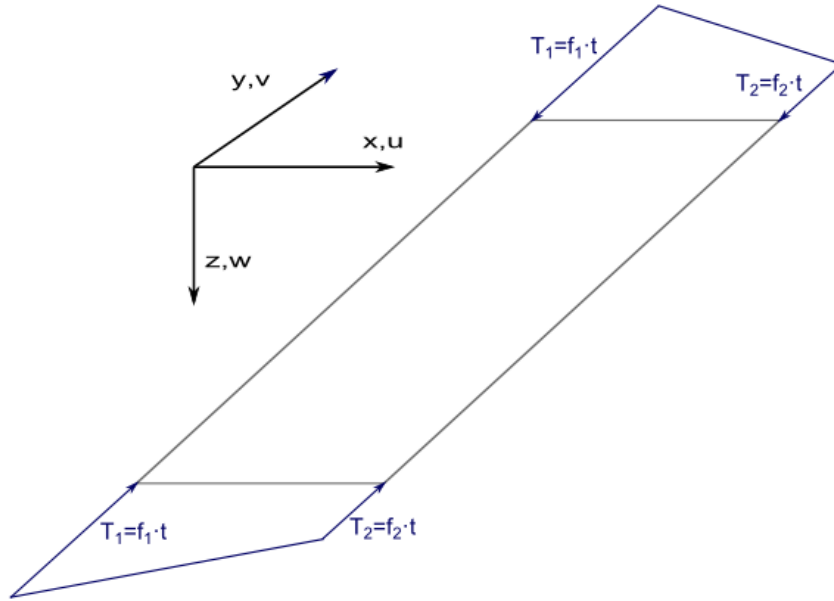


Figure 8: Strip stress distribution

The shape functions for the transverse direction are assumed to be the same polynomial function for every boundary condition. On the other hand, in the longitudinal direction, trigonometrical functions are taken. These functions have to satisfy the pre-set boundary conditions. The out of plane displacement will use a shape cubic polynomial function for all boundary conditions. Therefore, the expressions for general displacements are as follows:

$$u = \sum_{p=1}^m \left[\begin{pmatrix} 1 - \frac{x}{b} & \frac{x}{b} \end{pmatrix} \cdot \begin{Bmatrix} u_{1p} \\ u_{2p} \end{Bmatrix} \cdot Y_p \right]$$

$$v = \sum_{p=1}^m \left[\begin{pmatrix} 1 - \frac{x}{b} & \frac{x}{b} \end{pmatrix} \cdot \begin{Bmatrix} v_{1p} \\ v_{2p} \end{Bmatrix} \cdot Y_p' \cdot \frac{a}{\mu_p} \right]$$

$$w = \sum_{p=1}^m \left[\begin{pmatrix} 1 - \frac{3x^2}{b^2} + \frac{2x^3}{b^3} & x \cdot \left(1 - \frac{2x}{b} + \frac{x^2}{b^2} \right) & \frac{3x^2}{b^2} - \frac{2x^3}{b^3} & x \cdot \left(\frac{x^2}{b^2} - \frac{x}{b} \right) \end{pmatrix} \cdot \begin{Bmatrix} w_{1p} \\ \theta_{1p} \\ w_{2p} \\ \theta_{2p} \end{Bmatrix} \cdot Y_p \right]$$

Where $\mu_p = p \cdot \pi$ and p is the half-wave number. Y_m is the function for the longitudinal direction, which varies depending on the boundary conditions.

Boundary condition	Shape function
Simply-Simply	$Y_p = \sin \frac{p \cdot \pi \cdot y}{a}$
Clamped-Clamped	$Y_p = \sin \frac{p \cdot \pi \cdot y}{a} \cdot \sin \frac{\pi \cdot y}{a}$

Simply-Clamped	$Y_p = \sin \frac{(p+1) \cdot \pi \cdot y}{a} + \left(\frac{p+1}{p} \right) \sin \frac{p \cdot \pi \cdot y}{a}$
Clamped-Free	$Y_p = 1 - \cos \frac{\left(p - \frac{1}{2} \right) \cdot \pi \cdot y}{a}$
Clamped-Guided	$Y_p = \sin \frac{\left(p - \frac{1}{2} \right) \cdot \pi \cdot y}{a} \cdot \sin \frac{p \cdot \pi \cdot y}{2 \cdot a}$

We can put the displacement equations in form of a general vector such that:

$$\begin{Bmatrix} u \\ v \end{Bmatrix} = \sum_{p=1}^m [N_{uv}] \cdot \begin{Bmatrix} u_{1p} \\ v_{1p} \\ u_{2p} \\ v_{2p} \end{Bmatrix} = \sum_{p=1}^m [N_{uv}] \cdot d_{uv}^p$$

$$w = \sum_{p=1}^m [N_w] \cdot \begin{Bmatrix} w_{1p} \\ \theta_{1p} \\ w_{2p} \\ \theta_{2p} \end{Bmatrix} = \sum_{p=1}^m [N_w] \cdot d_w^p$$

m refers to the quantity of half-wave number employed in the analysis. It refers to the shape of the sinus function we will see in the buckling shape after deformation.

For example, for $m=1$ and simply-simply boundary conditions in the loaded edges we will have a full half-sinus in the buckling:

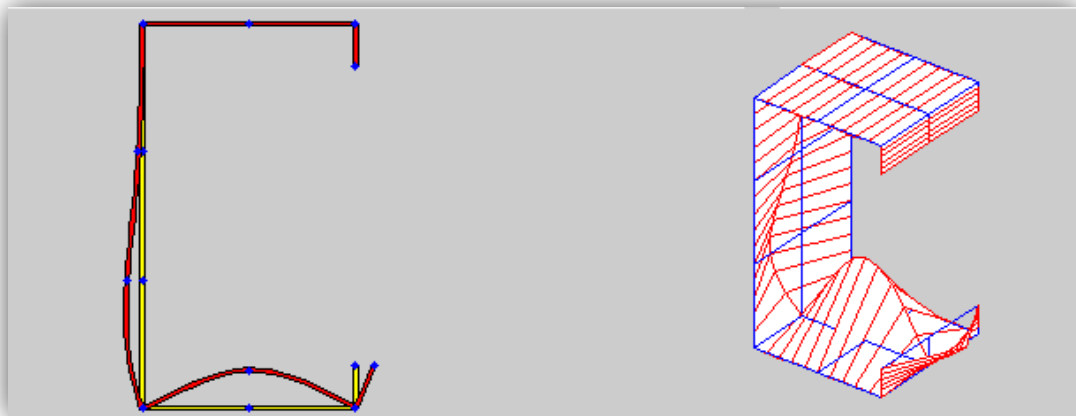
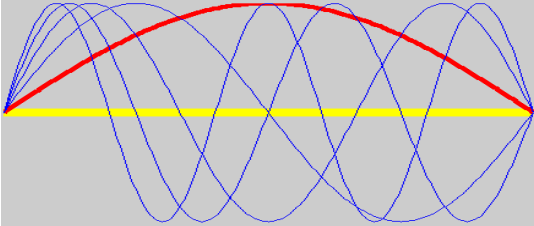
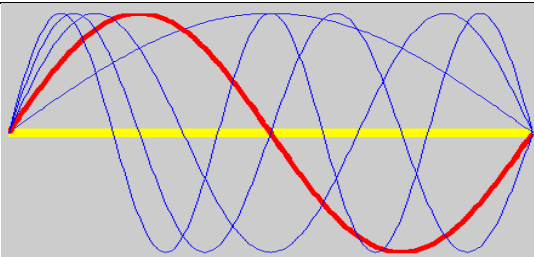
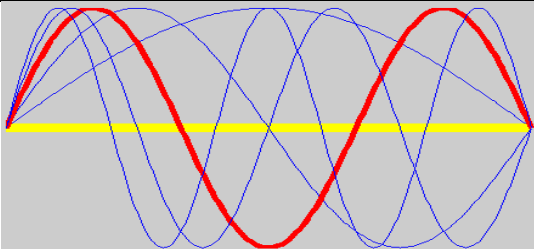
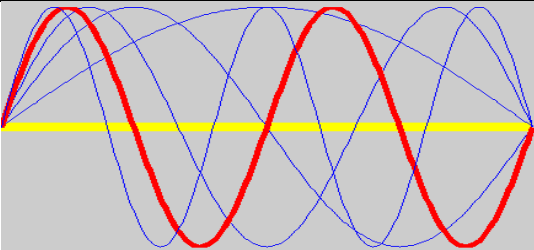


Figure 9: Cufsm4 program "C" section for simply-simply BC and $m=1$

We can see that the deformation in the 3D shape is a sinus form.

The deformation buckling form depends on the value of the m . We can see the first four "m" to understand the idea in the next table:

"m" number	Shape in the longitudinal direction
1	 <p data-bbox="884 501 1265 524">Figure 10: Simply-simply shape for m=1</p>
2	 <p data-bbox="884 786 1265 808">Figure 11: Simply-simply shape for m=2</p>
3	 <p data-bbox="884 1068 1265 1090">Figure 12: Simply-simply shape for m=3</p>
4	 <p data-bbox="884 1348 1265 1370">Figure 13: Simply-simply shape for m=4</p>

For $m=1$ we can see the next boundary conditions figures:

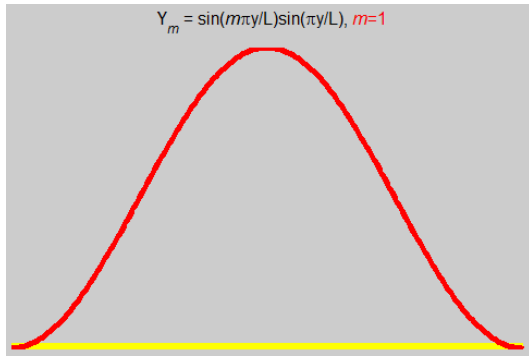


Figure 15: Shape for clamped-clamped boundary conditions

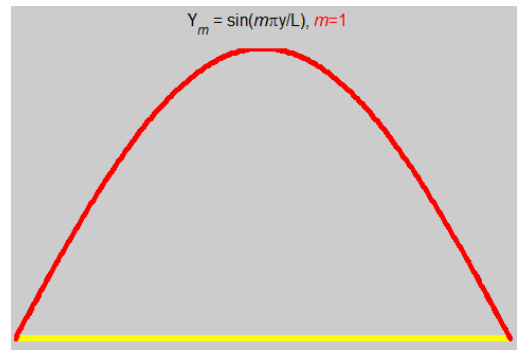


Figure 14: Shape for simply-simply boundary conditions

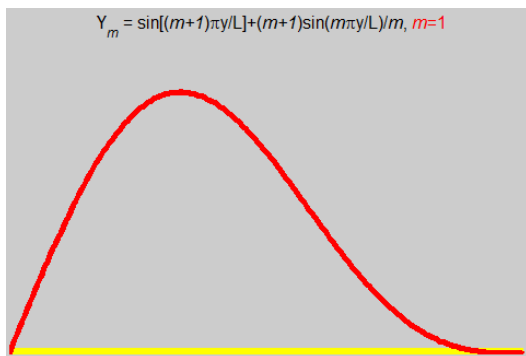


Figure 17: Shape for simply-clamped boundary conditions

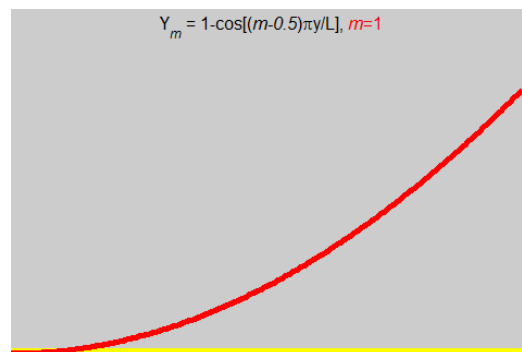


Figure 16: Shape for clamped-free boundary conditions

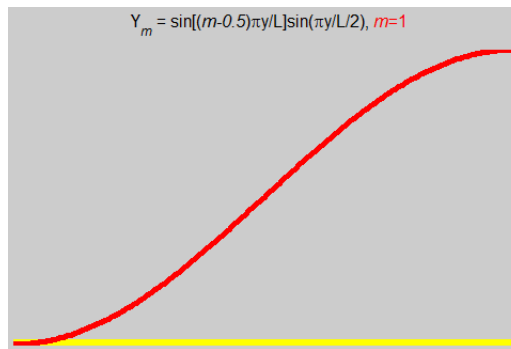


Figure 18: Shape for clamped-guided boundary conditions

4.3.2. Elastic stiffness matrix

In the strip we can distinguish two portions, bending and membrane. Every strip behaves as the superposition of two independent stresses, the membrane and the bending which is due to the normal forces to the plane of the plate.

The membrane strains are at the mid-line of the strip and will be governed by plane stress assumptions. On the other hand, bending strains follows Kirchhoff thin plate theory. We can define the deformation as the sum of both bending and membrane deformations.

$$\varepsilon = \varepsilon_M + \varepsilon_B$$

4.3.2.1. Membrane matrix

The membrane stress is defined as the component of normal stress that is uniformly distributed and equal to the average value of the stress across the thickness of the section under consideration.

In the membrane matrix the plates are submitted to normal and tangential forces in the plane of the plate, for which we consider a uniform distribution due to the fact that the thickness of the plate is considered very small compared to the dimensions of the plate. It is also considered that there is a linear behavior load vs. displacement.

The general expression of the deformation with the linear terms due to membrane stress is:

$$\{\varepsilon_M\} = \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix}_M = \begin{Bmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial v}{\partial y} \\ \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \end{Bmatrix}_M = \sum_{p=1}^m [N'_{uv}] \cdot \begin{Bmatrix} u_{1p} \\ v_{1p} \\ u_{2p} \\ v_{2p} \end{Bmatrix} = \sum_{p=1}^m [B_M^p] \cdot d_{uv}^p$$

It is obvious that the displacement due to membrane stress will depend on u and v , which are the displacements in the plane of the plate, where membrane stress occurs.

Given that the membrane behavior (u, v) is uncoupled from the bending behavior (w), we can define the internal strain energy during buckling for membrane stress as:

$$U_M = \frac{1}{2} \cdot \int_V \{\varepsilon_M\}^T \cdot \{\sigma_M\} \cdot dV$$

For the finite strip method we use a constant thickness, which can therefore stay out of the integration. Also, we can relate σ to ε using the D matrix:

$$\{\sigma_M\} = [D_M] \cdot \{\varepsilon_M\}$$

Where the D matrix is:

$$[D_M] = \frac{1}{1 - \nu_{yx} \nu_{xy}} \begin{bmatrix} E_x & \nu_{xy} \cdot E_x & 0 \\ \nu_{yx} \cdot E_1 & E_y & 0 \\ 0 & 0 & (1 - \nu_{xy} \nu_{yx}) \cdot G \end{bmatrix}$$

Where, if we have an isotropic material:

$$[D_M] = \frac{1}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{(1-\nu)}{2} \end{bmatrix}$$

We can then write the expression for the strain energy as:

$$U_M = \frac{1}{2} \cdot t \cdot \int_0^a \int_0^b \{\varepsilon_M\}^T \cdot [D_M] \cdot \{\varepsilon_M\} dx dy$$

The elastic stiffness matrix for membrane stress can be extracted from the statement for internal energy such as:

$$U_M = \sum_{p=1}^m \sum_{q=1}^m \frac{1}{2} \cdot (d_{uv}^p)^T \cdot \left(t \cdot \int_0^a \int_0^b B_M^{pT} \cdot [D_M] \cdot B_M^p dx dy \right) \cdot d_{uv}^q$$

Where our stiffness matrix is:

$$k_{eM}^{pq} = t \cdot \int_0^a \int_0^b B_M^{pT} \cdot [D_M] \cdot B_M^p dx dy$$

4.3.2.2. Bending matrix

The bending strains are defined as the variable stress across the thickness of the section under consideration, after the subtraction of the membrane stress. As previously stated, unlike the membrane stress, the bending strains follow the Kirchhoff plate theory.

Kirchhoff theory hypothesis

The hypothesis in which Kirchhoff's theory is based for thin plates are the next 4:

- Points in the middle plain only move vertically (u=v=0).
- All the points contained in a normal to the middle plain have the same vertical displacement.
- σ_z is not taken into account.
- Points on the normal lines to the plain stay in the same orthogonal lines to the middle plain after the deformation.

Thanks to the previous hypothesis, we can write our displacements as:

$$\begin{aligned}
 u(x, y, z) &= -z \cdot \theta_x(x, y) \\
 v(x, y, z) &= -z \cdot \theta_y(x, y) \\
 w(x, y, z) &= w(x, y)
 \end{aligned}$$

1st and 4th hypothesis
2nd hypothesis

Where

- w is the vertical displacement.
- θ_x and θ_y are the angles that define the turn of the normal line to the middle plain.

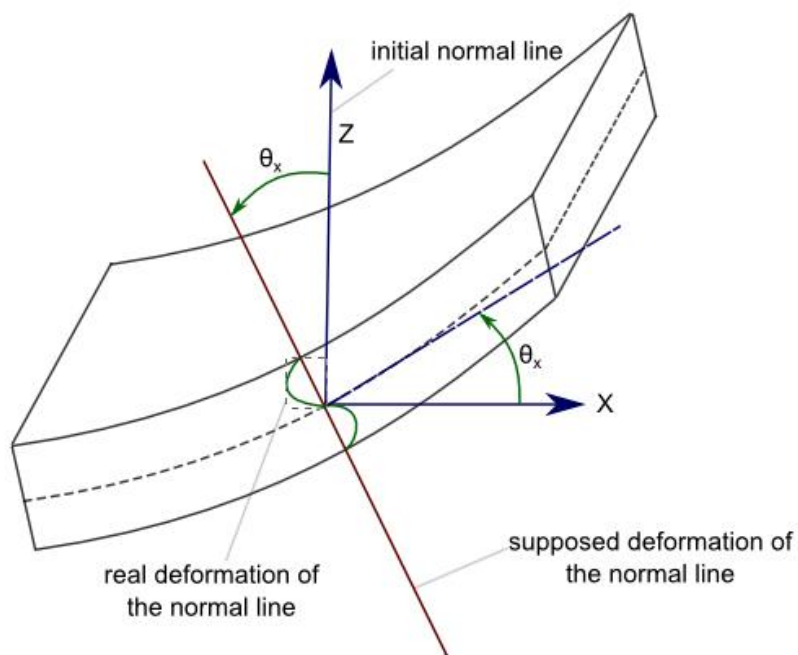


Figure 19: Kirchhoff deformation hypothesis image

Taking this image into account we can see that:

In the xz plain: $\theta_x = \frac{\partial w}{\partial x}$

In the yz plain: $\theta_y = \frac{\partial w}{\partial y}$

Using these equations and the ones written before we can conclude that:

$$u(x, y, z) = -z \cdot \frac{\partial w(x, y)}{\partial x}$$

$$v(x, y, z) = -z \cdot \frac{\partial w(x, y)}{\partial y}$$

$$w(x, y, z) = w(x, y)$$

Taking the last expression, we can define the deformation as:

$$\{\varepsilon_B\} = \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix} = \begin{Bmatrix} -z \cdot \frac{\partial^2 w}{\partial x^2} \\ -z \cdot \frac{\partial^2 w}{\partial y^2} \\ 2z \cdot \frac{\partial^2 w}{\partial x \partial y} \end{Bmatrix} = \sum_{p=1}^m z \cdot [N'_w] \cdot \begin{Bmatrix} w_{1p} \\ w_{2p} \\ \theta_{2p} \end{Bmatrix} = \sum_{p=1}^m z \cdot [B_B^p] \cdot d_w^p$$

As in the case of the membrane stress, we can write the bending term of the internal strain energy during buckling as:

$$U_B = \frac{1}{2} \cdot \int_V \{\varepsilon_B\}^T \cdot \{\sigma_B\} dV$$

Where we can substitute:

$$\{\sigma_B\} = [D_B] \cdot \{\varepsilon_B\}$$

$$[D_B] = \frac{t^3}{12} \cdot \frac{1}{1 - \nu_{yx}\nu_{xy}} \begin{bmatrix} E_x & \nu_{xy} \cdot E_x & 0 \\ \nu_{yx} \cdot E_1 & E_y & 0 \\ 0 & 0 & (1 - \nu_{xy}\nu_{yx}) \cdot G \end{bmatrix}$$

In the case of an isotropic material:

$$[D_B] = \frac{t^3}{12} \cdot \frac{1}{1 - \nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{(1 - \nu)}{2} \end{bmatrix}$$

We can then write the expression for the strain energy as:

$$U_B = \frac{1}{2} \cdot \int_0^a \int_0^b \{\varepsilon_B\}^T \cdot [D_B] \cdot \{\varepsilon_B\} dx dy$$

The elastic stiffness matrix for membrane stress can be extracted from the statement for internal energy such as:

$$U_B = \sum_{p=1}^m \sum_{q=1}^m \frac{1}{2} \cdot (d_w^p)^T \cdot \left(\int_0^a \int_0^b B_B^{pT} \cdot [D_B] \cdot B_B^p dx dy \right) \cdot d_w^q$$

Where our stiffness matrix is:

$$k_{eB}^{pq} = \int_0^a \int_0^b B_B^{pT} \cdot [D_B] \cdot B_B^p dx dy$$

4.3.2.3. Local stiffness matrix

The local stiffness matrix will be a combination of both the membrane and the bending strains. We can define it as:

$$K_e^{pq} = \begin{bmatrix} k_{eM}^{pq} & 0 \\ 0 & k_{eB}^{pq} \end{bmatrix}$$

During the development of the global stiffness matrix the next general expressions for both membrane and bending matrixes have been found. These expressions correspond to the value after the integration:

$$k_{eM}^{pq} = t \cdot \begin{bmatrix} \left(\frac{E_1 I_1}{b} + \frac{G b I_5}{3} \right) & \left(-\frac{E_2 v_x I_2}{2c_1} - \frac{G I_5}{2c_1} \right) & \left(-\frac{E_1 I_1}{b} + \frac{G b I_5}{6} \right) & \left(-\frac{E_2 v_x I_2}{2c_1} + \frac{G I_5}{2c_1} \right) \\ \left(-\frac{E_2 v_x I_2}{2c_1} - \frac{G I_5}{2c_1} \right) & \left(\frac{E_2 b I_4}{3c_1 c_2} + \frac{G I_5}{b c_1 c_2} \right) & \left(\frac{E_2 v_x I_2}{2c_1} - \frac{G I_5}{2c_1} \right) & \left(\frac{E_2 b I_4}{6c_1 c_2} - \frac{G I_5}{b c_1 c_2} \right) \\ \left(-\frac{E_1 I_1}{b} + \frac{G b I_5}{6} \right) & \left(\frac{E_2 v_x I_2}{2c_1} - \frac{G I_5}{2c_1} \right) & \left(\frac{E_1 I_1}{b} + \frac{G b I_5}{3} \right) & \left(\frac{E_2 v_x I_2}{2c_1} + \frac{G I_5}{2c_1} \right) \\ \left(-\frac{E_2 v_x I_2}{2c_1} + \frac{G I_5}{2c_1} \right) & \left(\frac{E_2 b I_4}{6c_1 c_2} - \frac{G I_5}{b c_1 c_2} \right) & \left(\frac{E_2 v_x I_2}{2c_1} + \frac{G I_5}{2c_1} \right) & \left(\frac{E_2 b I_4}{3c_1 c_2} + \frac{G I_5}{b c_1 c_2} \right) \end{bmatrix}$$

$$k_{eB}^{pq} = \frac{1}{420b^3} \begin{bmatrix} \left(\begin{array}{l} 5040D_x I_1 - 504b^2 D_1 I_2 \\ -504b^2 D_1 I_3 + 156b^4 D_y I_4 \\ +2016b^2 D_{xy} I_5 \end{array} \right) & \left(\begin{array}{l} 2520b D_x I_1 - 462b^3 D_1 I_2 \\ -42b^3 D_1 I_3 + 22b^5 D_y I_4 \\ +168b^3 D_{xy} I_5 \end{array} \right) & \left(\begin{array}{l} -5040D_x I_1 + 504b^2 D_1 I_2 \\ +504b^2 D_1 I_3 + 54b^4 D_y I_4 \\ -2016b^2 D_{xy} I_5 \end{array} \right) & \left(\begin{array}{l} 2520b D_x I_1 - 42b^3 D_1 I_2 \\ -42b^3 D_1 I_3 - 13b^5 D_y I_4 \\ +168b^3 D_{xy} I_5 \end{array} \right) \\ \left(\begin{array}{l} 2520b D_x I_1 - 462b^3 D_1 I_2 \\ -42b^3 D_1 I_3 + 22b^5 D_y I_4 \\ +168b^3 D_{xy} I_5 \end{array} \right) & \left(\begin{array}{l} 1680b^2 D_x I_1 - 56b^4 D_1 I_2 \\ -56b^4 D_1 I_3 + 4b^6 D_y I_4 \\ +224b^4 D_{xy} I_5 \end{array} \right) & \left(\begin{array}{l} -2520b D_x I_1 + 42b^3 D_1 I_2 \\ +42b^3 D_1 I_3 + 13b^5 D_y I_4 \\ -168b^3 D_{xy} I_5 \end{array} \right) & \left(\begin{array}{l} 840b^2 D_x I_1 + 14b^4 D_1 I_2 \\ +14b^4 D_1 I_3 - 3b^6 D_y I_4 \\ -56b^4 D_{xy} I_5 \end{array} \right) \\ \left(\begin{array}{l} -5040D_x I_1 + 504b^2 D_1 I_2 \\ +504b^2 D_1 I_3 + 54b^4 D_y I_4 \\ -2016b^2 D_{xy} I_5 \end{array} \right) & \left(\begin{array}{l} -2520b D_x I_1 + 42b^3 D_1 I_2 \\ +42b^3 D_1 I_3 + 13b^5 D_y I_4 \\ -168b^3 D_{xy} I_5 \end{array} \right) & \left(\begin{array}{l} 5040D_x I_1 - 504b^2 D_1 I_2 \\ -504b^2 D_1 I_3 + 156b^4 D_y I_4 \\ +2016b^2 D_{xy} I_5 \end{array} \right) & \left(\begin{array}{l} -2520b D_x I_1 + 462b^3 D_1 I_2 \\ +42b^3 D_1 I_3 - 22b^5 D_y I_4 \\ -168b^3 D_{xy} I_5 \end{array} \right) \\ \left(\begin{array}{l} 2520b D_x I_1 - 42b^3 D_1 I_2 \\ -42b^3 D_1 I_3 - 13b^5 D_y I_4 \\ +168b^3 D_{xy} I_5 \end{array} \right) & \left(\begin{array}{l} -5040D_x I_1 + 504b^2 D_1 I_2 \\ +504b^2 D_1 I_3 + 54b^4 D_y I_4 \\ -2016b^2 D_{xy} I_5 \end{array} \right) & \left(\begin{array}{l} -5040D_x I_1 + 504b^2 D_1 I_2 \\ +504b^2 D_1 I_3 + 54b^4 D_y I_4 \\ -2016b^2 D_{xy} I_5 \end{array} \right) & \left(\begin{array}{l} 1680b^2 D_x I_1 - 56b^4 D_1 I_2 \\ -56b^4 D_1 I_3 + 4b^6 D_y I_4 \\ +224b^4 D_{xy} I_5 \end{array} \right) \end{bmatrix}$$

Where

$$c_1 = \frac{p\pi}{a}$$

$$c_2 = \frac{q\pi}{a}$$

I depends on the boundary conditions and is:

$$I_1 = \int_0^a Y_p \cdot Y_q \cdot dy$$

$$I_2 = \int_0^a Y_p'' \cdot Y_q \cdot dy$$

$$I_3 = \int_0^a Y_p \cdot Y_q'' \cdot dy$$

$$I_4 = \int_0^a Y_p'' \cdot Y_q'' \cdot dy$$

$$I_5 = \int_0^a Y_p' \cdot Y_q' \cdot dy$$

And the coefficient E and D correspond to:

$$E_1 = \frac{E_x}{1 - \nu_x \nu_y}$$

$$E_2 = \frac{E_y}{1 - \nu_x \nu_y}$$

$$D_x = \frac{E_x t^3}{12(1 - \nu_x \nu_y)}$$

$$D_y = \frac{E_y t^3}{12(1 - \nu_x \nu_y)}$$

$$D_1 = \frac{\nu_x E_y t^3}{12(1 - \nu_x \nu_y)} = \frac{\nu_y E_x t^3}{12(1 - \nu_x \nu_y)}$$

$$D_{xy} = \frac{G t^3}{12}$$

Finally we can describe the full local matrix depending on the half-wave number as:

$$K_e = \begin{bmatrix} k_{eM}^{pq} & \cdot \\ \cdot & k_{eB}^{pq} \end{bmatrix}_{m \times m}$$

Where m is the maximum half-wave number. For example for $m = 2$ we will have a matrix:

$$K_e = \begin{bmatrix} K_e^{11} & K_e^{12} \\ K_e^{21} & K_e^{22} \end{bmatrix}$$

Therefore, the dimension of our matrix would be equal to:

$$\text{dimension} = m \cdot \text{nodes} \cdot \text{displacements}$$

Where

$m = \text{maximum halfwave numbers}$

$\text{nodes} = \text{total number of nodes}$

$\text{displacements} = 4 \text{ displacements for every node}$

4.3.3. Geometric stiffness matrix

The geometric stiffness matrix is crucial in order to calculate the correct buckling critical loads. It can be calculated either in terms of higher order strain or by the method of the potential energy. The potential energy method is used here. We distinguish in each strip the stress in both nodes, denominated T_1 and T_2 . You can express the potential due to this two stresses during buckling as:

$$V_p = \frac{1}{2} \int_0^a \int_0^b (T_1 - (T_1 - T_2) \frac{x}{b}) \cdot \left[\left(\frac{\partial u}{\partial y} \right)^2 + \left(\frac{\partial v}{\partial y} \right)^2 + \left(\frac{\partial w}{\partial y} \right)^2 \right] dx dy$$

Like in the case of the elastic stiffness matrix, we can distinguish the bending part (w) and the membrane part (u and v). We can express the derivatives of displacements depending on the shape functions, the nodal displacements and the half-wave number p .

In the case of the bending part we can define the derivative of w as:

$$\left(\frac{\partial w}{\partial y} \right)^2 = \left(\sum_{p=1}^m [N'_w] \begin{Bmatrix} w_{1p} \\ \theta_{1p} \\ w_{2p} \\ \theta_{2p} \end{Bmatrix} \right) = \sum_{p=1}^m \sum_{q=1}^m \begin{Bmatrix} w_{1p} \\ \theta_{1p} \\ w_{2p} \\ \theta_{2p} \end{Bmatrix}^T [G_B^p]^T [G_B^q] \begin{Bmatrix} w_{1p} \\ \theta_{1p} \\ w_{2p} \\ \theta_{2p} \end{Bmatrix}$$

For the membrane, the derivative of u and v are:

$$\begin{Bmatrix} \left(\frac{\partial u}{\partial y} \right)^2 \\ \left(\frac{\partial v}{\partial y} \right)^2 \end{Bmatrix} = \left(\sum_{p=1}^m [N'_{uv}] \begin{Bmatrix} u_{1p} \\ v_{1p} \\ u_{2p} \\ v_{2p} \end{Bmatrix} \right)^2 = \sum_{p=1}^m \sum_{q=1}^m \begin{Bmatrix} u_{1p} \\ v_{1p} \\ u_{2p} \\ v_{2p} \end{Bmatrix}^T [G_M^p]^T [G_M^q] \begin{Bmatrix} u_{1p} \\ v_{1p} \\ u_{2p} \\ v_{2p} \end{Bmatrix}$$

Finally we can write the potential energy V_p as:

$$V_p = \frac{1}{2} \int_0^a \int_0^b (T_1 - (T_1 - T_2) \frac{x}{b}) \left(\sum_{p=1}^m \sum_{q=1}^m d_p^T [G^p]^T [G^q] d_q \right)$$

which can be rewritten as:

$$V_p = \sum_{p=1}^m \sum_{q=1}^m d_p^T k_g^{pq} d_q$$

Where the matrix k_g^{pq} correspond to the half-wave numbers p and q can be divided into two parts, the membrane and the bending, similar to the case of the elastic matrix:

$$k_g^{pq} = \begin{bmatrix} k_{gM}^{pq} & \cdot \\ \cdot & k_{gB}^{pq} \end{bmatrix}$$

Therefore, taking all the p and q half-waves we can create the full geometric matrix which is:

$$k_g = [k_g^{pq}]_{m \times m}$$

Where m is the maximum number of half-waves used in the analysis.

The values of the bending and membrane matrix after integration corresponding to the half-waves p and q can be obtained through the substitution in the next matrices:

$$k_{gM}^{pq} = \begin{bmatrix} \frac{(3T_1 + T_2)bl_5}{12} & 0 & \frac{(T_1 + T_2)bl_5}{12} & 0 \\ 0 & \frac{(3T_1 + T_2)ba^2I_4}{12\mu_p\mu_q} & 0 & \frac{(T_1 + T_2)ba^2I_4}{12\mu_p\mu_q} \\ \frac{(T_1 + T_2)bl_5}{12} & 0 & \frac{(3T_1 + T_2)bl_5}{12} & 0 \\ 0 & \frac{(T_1 + T_2)ba^2I_4}{12\mu_p\mu_q} & 0 & \frac{(3T_1 + T_2)ba^2I_4}{12\mu_p\mu_q} \end{bmatrix}$$

$$k_{gB}^{pq} = \begin{bmatrix} \frac{(10T_1 + 3T_2)bl_5}{35} & \frac{(15T_1 + 7T_2)b^2I_5}{420} & \frac{(T_1 + T_2)bl_5}{140} & -\frac{(7T_1 + 6T_2)b^2I_5}{420} \\ \frac{(15T_1 + 7T_2)b^2I_5}{420} & \frac{(5T_1 + 3T_2)b^3I_5}{840} & \frac{(6T_1 + 7T_2)b^2I_5}{420} & -\frac{(T_1 + T_2)b^3I_5}{280} \\ \frac{(T_1 + T_2)bl_5}{140} & \frac{(6T_1 + 7T_2)b^2I_5}{420} & \frac{(3T_1 + 10T_2)bl_5}{35} & -\frac{(7T_1 + 15T_2)b^2I_5}{420} \\ -\frac{(7T_1 + 6T_2)b^2I_5}{420} & -\frac{(T_1 + T_2)b^3I_5}{280} & -\frac{(7T_1 + 15T_2)b^2I_5}{420} & \frac{(3T_1 + 5T_2)b^3I_5}{840} \end{bmatrix}$$

Where

$$\mu_p = p\pi$$

$$\mu_q = q\pi$$

I depends on the boundary conditions and is:

$$I_1 = \int_0^a Y_p \cdot Y_q \cdot dy$$

$$I_2 = \int_0^a Y_p'' \cdot Y_q \cdot dy$$

$$I_3 = \int_0^a Y_p \cdot Y_q'' \cdot dy$$

$$I_4 = \int_0^a Y_p'' \cdot Y_q'' \cdot dy$$

$$I_5 = \int_0^a Y_p' \cdot Y_q' \cdot dy$$

And T_1 and T_2 are the stresses in the nodes of the element.

Finally we can describe the full local matrix depending on the half-wave number as:

$$K_g = \begin{bmatrix} k_{gM}^{pq} & \cdot \\ \cdot & k_{gB}^{pq} \end{bmatrix}_{m \times m}$$

Where m is the maximum half-wave number. For example for $m = 2$ we will have a matrix:

$$K_g = \begin{bmatrix} K_g^{11} & K_g^{12} \\ K_g^{21} & K_g^{22} \end{bmatrix}$$

4.3.4. Assembly of the elements

We have now found the local elastic stiffness matrix and the local geometric stiffness matrix. In order to create the full global matrices we must first transform the local matrices to the global coordinates and then assemble all the matrices together to find the full global matrices.

4.3.4.1. Rotation

In order to rotate the strip we have to create the matrix for rotation which relates movements between local and global coordinates. In order to understand this matrix we can use the next image:

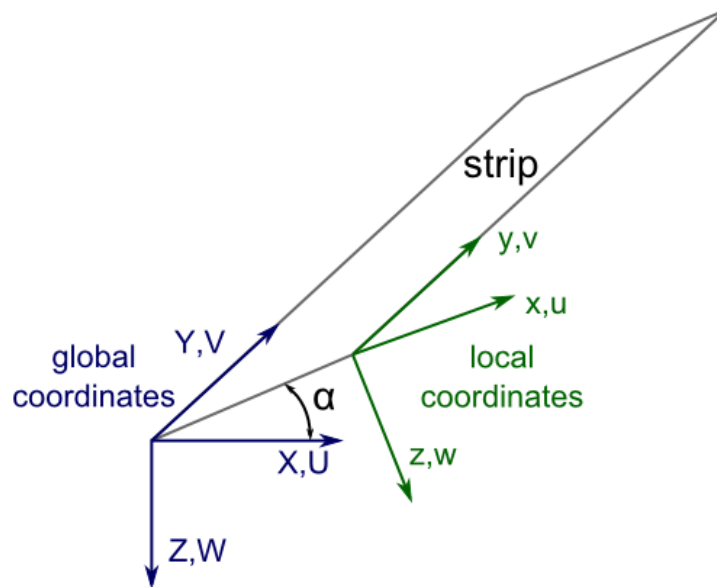


Figure 20: Strip rotation guide image

$$\begin{pmatrix} U_1 \\ V_1 \\ U_2 \\ V_2 \\ W_1 \\ \theta_1 \\ W_2 \\ \theta_2 \end{pmatrix} = \begin{bmatrix} \cos(\alpha) & 0 & 0 & 0 & -\sin(\alpha) & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \cos(\alpha) & 0 & 0 & 0 & -\sin(\alpha) & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \sin(\alpha) & 0 & 0 & 0 & \cos(\alpha) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \sin(\alpha) & 0 & 0 & 0 & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ w_1 \\ \theta_1 \\ w_2 \\ \theta_2 \end{pmatrix}$$

This matrix relates the local coordinates to the global coordinates. This matrix has to be extended to the m number of halfwaves analyzed.

4.3.4.2. Assembly

After changing the local matrices to the global coordinates we must assemble them together to create the full global matrix.

In order to join the matrices we must look at the nodes. In the next image we can see a plate and we will create a matrix for this particular example:

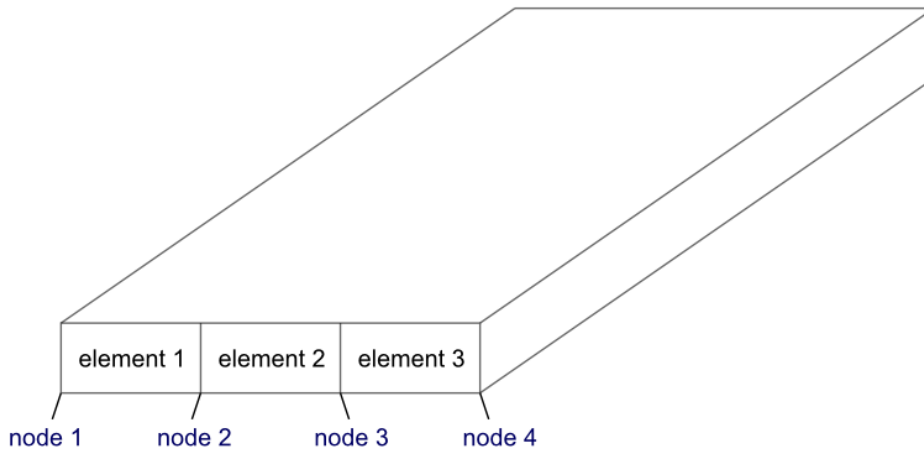


Figure 21: Example image for assembly – This plate with 3 elements

Therefore the global matrix must contain all the local matrices together and position like in the case of the finite element method. The K local matrix has 4 components, relating how every node affects to each other:

$$K_{local} = \begin{bmatrix} K_{ii} & K_{ij} \\ K_{ji} & K_{jj} \end{bmatrix}$$

Where every K_{ii} is a 4x4 matrix with the 4 displacements.

With this matrix we can get the global matrix of our example:

$$K_{global} = \begin{bmatrix} K_{ii} & K_{ij} & 0 & 0 \\ K_{ji} & K_{ii} + K_{jj} & K_{ij} & 0 \\ 0 & K_{ji} & K_{ii} + K_{jj} & K_{ij} \\ 0 & 0 & K_{ji} & K_{jj} \end{bmatrix}$$

In a random section, this matrix will be extended to all the elements and nodes that exist in our section.

4.4. Buckling modes

Thin-walled members, when subjected to a large compressive normal stress, usually fail by the loss of stability, more than due to reaching the material limit parameters. Usually we can distinguish three basic modes of buckling: local, distortional and global buckling. All three modes eventually cause excessive deformation and lead to failure. [7]*

It is very important to understand the buckling modes and the critical stress associated to each mode. Every mode has a different degree of post-buckling capacity, but they all lead to an eventual collapse response. Therefore, we must calculate the critical stress to avoid the loss of stability.

Even when the designs do require the calculation of buckling stresses, there are no clear definitions for each of the modes and it is quite difficult to ultimately distinguish between them correctly.

4.4.1. Local buckling

The local buckling of a section is normally defined as the mode which involves the deformation of the thin plates composing the section, without translation of the intersection lines between the same thin plates. Some examples of this mode are shown:

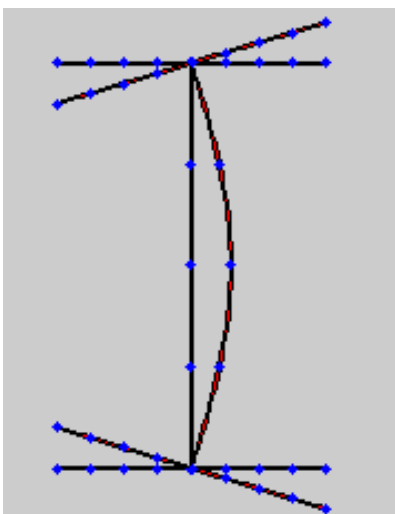


Figure 23: Local buckling for an "I" section

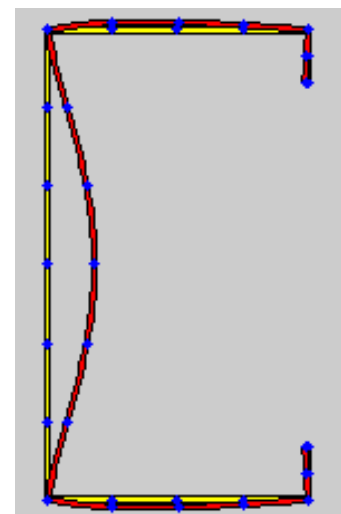
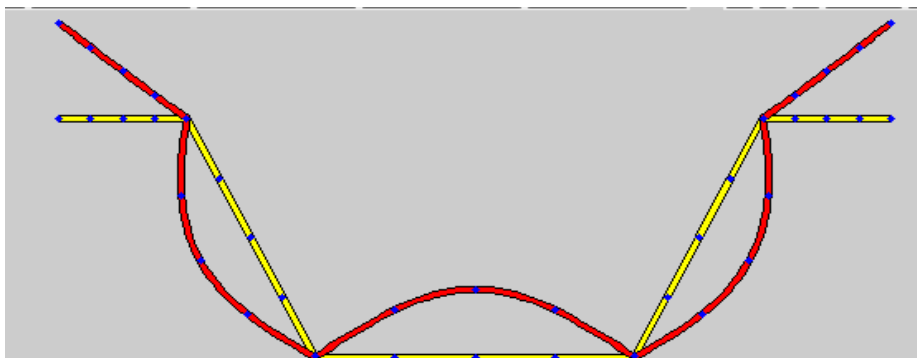


Figure 22: Local buckling for a "C" section



*[7]: Reference to bibliography

Figure 24: Local buckling for a "V" section

4.4.2. Distortional buckling

Distortional buckling is a mode defined as a cross-sectional distortion that involves the translation of some of the fold lines (intersection lines between thin plates). It is a rare mode of buckling, we can see a clear example of it in the C section:

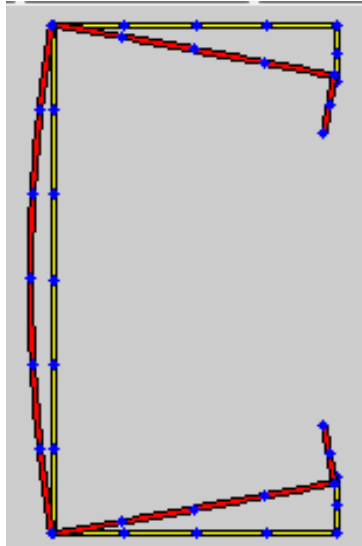


Figure 25: Distortional buckling for a “C” section

4.4.3. Global buckling

Global buckling is a mode where the member deforms with no deformation in its section.

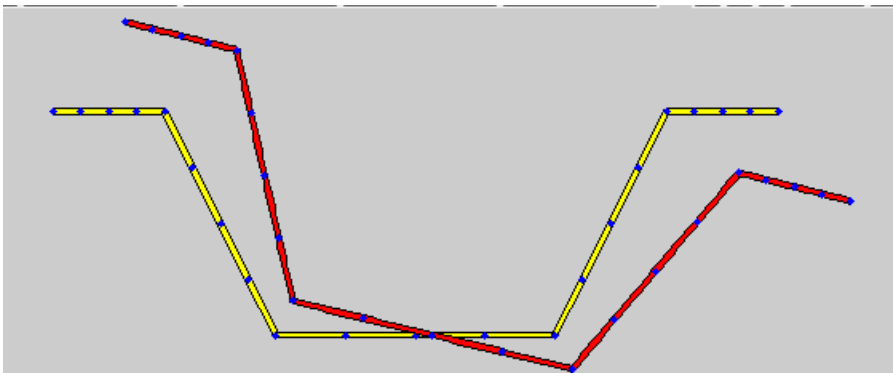


Figure 27: Global buckling for a “V” section

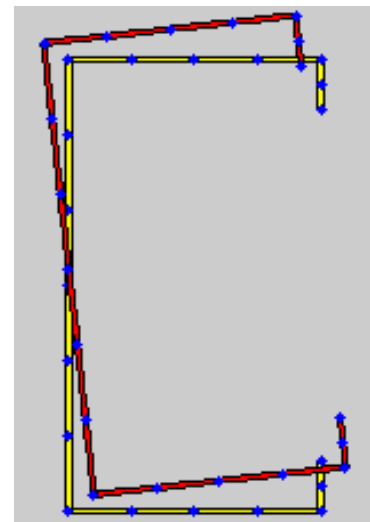


Figure 26: Global buckling for a “C” section

4.4.4. Generalized Beam Theory

Thanks to the Generalized Beam Theory (GBT), the modes can be correctly described and with some programs we can even isolate different buckling modes. In order to distinguish between the buckling modes, the GBT establishes three criteria:

Criteria 1

- $\gamma_{xy} = 0$, membrane (in-plane) shear strains are zero.
- $\varepsilon_x = 0$, membrane transverse strains are zero.
- $v = f(x)$, longitudinal displacements are linear in x within an element.

Criteria 2

- $\varepsilon_y \neq 0$, longitudinal strains/displacements are non-zero along the length.

Criteria 3

- $\kappa_y = 0$, no flexure in the transverse direction.

The buckling modes can then be described as:

- **Global buckling:** Global buckling modes are those that satisfy all the three criteria.
- **Distortional buckling:** Distortional buckling modes are those that satisfy criteria 1 and 2 but do not satisfy criteria 3.
- **Local buckling:** Local buckling modes satisfy criterion 1, but do not satisfy criterion 2. Criterion 3 is irrelevant in this case.

In the next table we can see a resume of the classification of modes.

	Global modes	Distortional modes	Local modes
$\gamma_{xy} = 0, \varepsilon_x = 0, v = f(x)$	YES	YES	YES
$\varepsilon_y \neq 0$	YES	YES	NO
$\kappa_y = 0$	YES	NO	-

The program "CUFSM 4.05" has been used to obtain the images of the previous buckling modes.

5. Matlab program

In order to solve structures using the finite strip method, in this project we have developed a program that will solve any section structures utilizing the finite strip method.

We will explain both the functioning of the program as all the sub-programs that have been implemented. We will then proceed to explain how to use the program and use a few examples to understand the functioning.

5.1. Main program

The main programs that has been developed returns, for several strip lengths and any section, the critical buckling value and the buckling mode. It also calculates the particular displacement caused by the submitted stress.

Our main program is named "FSMsolver" and the function is:

```
[shape,curve,sigmacritical] =
FSMcrit(material_properties,nodes,elements,lengths,boundary_conditions,m_a)
```

INPUTS	OUTPUTS
Material properties: [Ex Eyvxvy G]	Shape
Nodes [node_numberx_positionz_position stress]	Curve
Elements [element_numbernodeinnodej t]	Critical buckling stress
Lengths	Displacement
Boundary conditions	
m_a	

Inputs

- **Material properties:** One of the inputs that are demanded is the material properties.
 - $E_x = E_x$: Young modulus in the "x" direction.
 - $E_y = E_y$: Young modulus in the "y" direction.
 - $\nu_x = \nu_x$: Poisson's ratio in the "x" direction.
 - $\nu_y = \nu_y$: Poisson's ratio in the "y" direction.
 - G_{xy} = The shear modulus for the material. It can be calculated using the Young modulus and the Poisson's ratio:

$$G_{xy} = \frac{E_x}{2 \cdot (1 + \nu_{xy})}$$

- **Nodes:** The nodes of the section we want to analyze
 - node_number = The node number that we will then we can then identify to know the modal displacement in the "shape" output.
 - x_position = The "x" coordinate for the node.
 - z_position = The "z" coordinate for the node.
 - stress = The normal stress we want the node to suffer.

- **Elements:** The elements connecting the nodes must also be defined
 - element_number = The element number
 - nodei and nodej = The two nodes the element is connecting
 - t = The thickness of the element connecting both nodes

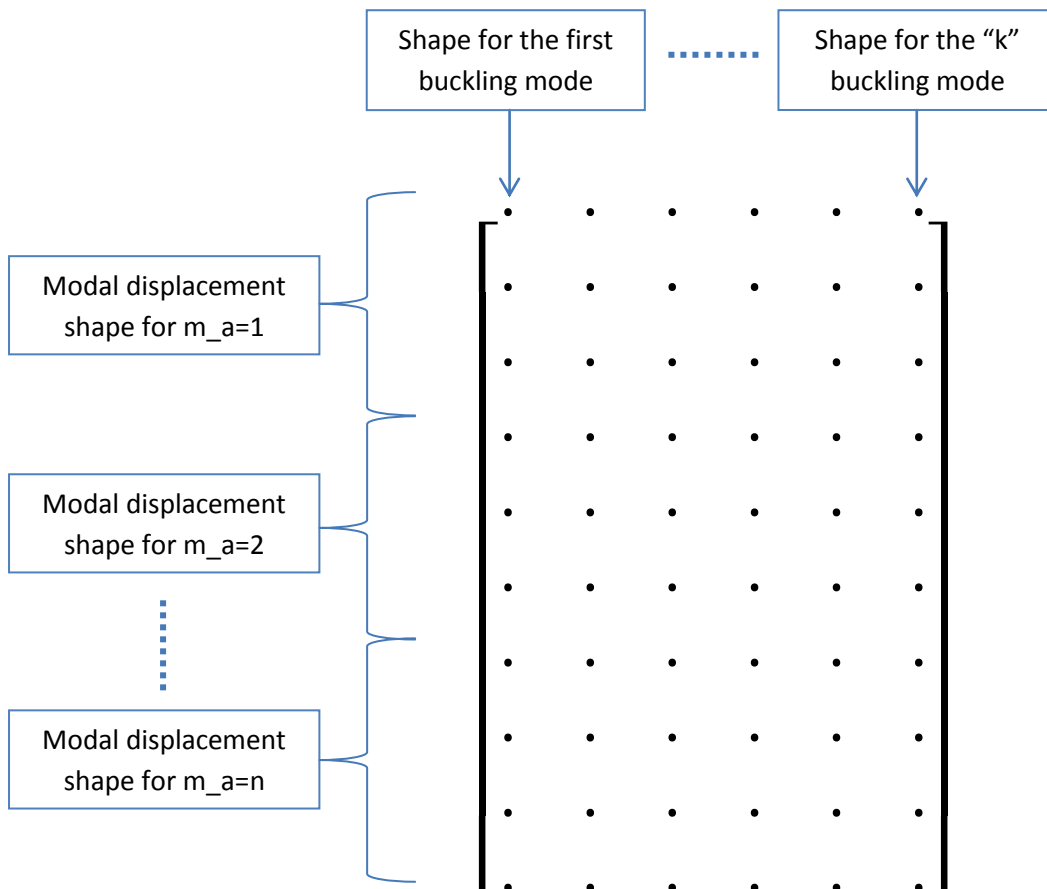
- **Lengths:** The length of the strips we are analyzing.

- **Boundary conditions:** The boundary conditions at the loaded edges.
 - 'S-S': Simply-simply supported boundary condition at loaded edges.
 - 'C-C': Clamped-clamped boundary condition at loaded edges.
 - 'S-C': Simply-clamped boundary condition at loaded edges.
 - 'C-F': Clamped-free supported boundary condition at loaded edges.
 - 'C-G': Clamped-guided supported boundary condition at loaded edges.

- **m_a:** half-wave terms to be analyzed for the length.

Output

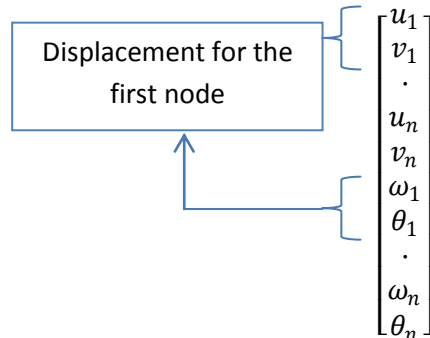
- **Shape:** A matrix containing the modal displacement for all nodes after the finite strip analysis. The matrix will contain the next values.



Where we can distinguish:

- n = The number of half-waves that want to be analyzed, that can be described in the variable m_a , where for example:
 - $m_a=[1\ 2\ 3\ 4\ 5]$ would analyze the first five half-waves, where “ n ” would be 5.
- k = Number of buckling modes that will be analyzed. The default value is 10, however depending on the quantity of nodes and half-waves, more buckling modes may be analyzed.

The modal displacement shape includes in the column the next movements:



Where “ n ” will be the number of nodes in the section that is analyzed.

- **Curve:** Includes the critical buckling stress for the “ k ” buckling modes that were found in the “shape” output. The value we find in the curve will be $\frac{P_{crit}}{stress\ input}$, meaning we have to multiply the value we find with the stress we input to find the critical buckling stress.
- **Critical buckling stress:** The program will automatically output the critical value in the “curve” output, which will be the first buckling mode we will have. Take into account that the value the critical buckling stress output gives us is the value of the critical stress divided between the input stress for the nodes.
- **Displacement:** The program will also return the displacement values for every node as shown in the previous vector for the particular stress input.

5.2. Sub-programs

In order to utilize the previous program, several sub-programs are required. All these programs have been developed utilizing the theory explained in the previous chapter.

5.2.1. Local elastic matrix sub-program

A program to find the local elastic matrix has been created. In this function we will create the local geometric matrix defining it as:

$$K_e = \begin{bmatrix} k_{eM}^{pq} & \cdot \\ \cdot & k_{eB}^{pq} \end{bmatrix}_{mxm}$$

Where

$$k_{eM}^{pq} = t \cdot \begin{bmatrix} \left(\frac{E_1 I_1}{b} + \frac{G b I_5}{3} \right) & \left(-\frac{E_2 v_x I_2}{2c_1} - \frac{G I_5}{2c_1} \right) & \left(-\frac{E_1 I_1}{b} + \frac{G b I_5}{6} \right) & \left(-\frac{E_2 v_x I_2}{2c_1} + \frac{G I_5}{2c_1} \right) \\ \left(-\frac{E_2 v_x I_2}{2c_1} - \frac{G I_5}{2c_1} \right) & \left(\frac{E_2 b I_4}{3c_1 c_2} + \frac{G I_5}{b c_1 c_2} \right) & \left(\frac{E_2 v_x I_2}{2c_1} - \frac{G I_5}{2c_1} \right) & \left(\frac{E_2 b I_4}{6c_1 c_2} - \frac{G I_5}{b c_1 c_2} \right) \\ \left(-\frac{E_1 I_1}{b} + \frac{G b I_5}{6} \right) & \left(\frac{E_2 v_x I_2}{2c_1} - \frac{G I_5}{2c_1} \right) & \left(\frac{E_1 I_1}{b} + \frac{G b I_5}{3} \right) & \left(\frac{E_2 v_x I_2}{2c_1} + \frac{G I_5}{2c_1} \right) \\ \left(-\frac{E_2 v_x I_2}{2c_1} + \frac{G I_5}{2c_1} \right) & \left(\frac{E_2 b I_4}{6c_1 c_2} - \frac{G I_5}{b c_1 c_2} \right) & \left(\frac{E_2 v_x I_2}{2c_1} + \frac{G I_5}{2c_1} \right) & \left(\frac{E_2 b I_4}{3c_1 c_2} + \frac{G I_5}{b c_1 c_2} \right) \end{bmatrix}$$

$$k_{eB}^{pq} = \frac{1}{420b^3} \begin{bmatrix} \left(\begin{array}{l} 5040D_x I_1 - 504b^2 D_1 I_2 \\ -504b^2 D_1 I_3 + 156b^4 D_y I_4 \\ +2016b^2 D_{xy} I_5 \end{array} \right) & \left(\begin{array}{l} 2520b D_x I_1 - 462b^3 D_1 I_2 \\ -42b^3 D_1 I_3 + 22b^5 D_y I_4 \\ +168b^3 D_{xy} I_5 \end{array} \right) & \left(\begin{array}{l} -5040D_x I_1 + 504b^2 D_1 I_2 \\ +504b^2 D_1 I_3 + 54b^4 D_y I_4 \\ -2016b^2 D_{xy} I_5 \end{array} \right) & \left(\begin{array}{l} 2520b D_x I_1 - 42b^3 D_1 I_2 \\ -42b^3 D_1 I_3 - 13b^5 D_y I_4 \\ +168b^3 D_{xy} I_5 \end{array} \right) \\ \left(\begin{array}{l} 2520b D_x I_1 - 462b^3 D_1 I_2 \\ -42b^3 D_1 I_3 + 22b^5 D_y I_4 \\ +168b^3 D_{xy} I_5 \end{array} \right) & \left(\begin{array}{l} 1680b^2 D_x I_1 - 56b^4 D_1 I_2 \\ -56b^4 D_1 I_3 + 4b^6 D_y I_4 \\ +224b^4 D_{xy} I_5 \end{array} \right) & \left(\begin{array}{l} -2520b D_x I_1 + 42b^3 D_1 I_2 \\ +42b^3 D_1 I_3 + 13b^5 D_y I_4 \\ -168b^3 D_{xy} I_5 \end{array} \right) & \left(\begin{array}{l} 840b^2 D_x I_1 + 14b^4 D_1 I_2 \\ +14b^4 D_1 I_3 - 3b^6 D_y I_4 \\ -56b^4 D_{xy} I_5 \end{array} \right) \\ \left(\begin{array}{l} -5040D_x I_1 + 504b^2 D_1 I_2 \\ +504b^2 D_1 I_3 + 54b^4 D_y I_4 \\ -2016b^2 D_{xy} I_5 \end{array} \right) & \left(\begin{array}{l} -2520b D_x I_1 + 42b^3 D_1 I_2 \\ +42b^3 D_1 I_3 + 13b^5 D_y I_4 \\ -168b^3 D_{xy} I_5 \end{array} \right) & \left(\begin{array}{l} 5040D_x I_1 - 504b^2 D_1 I_2 \\ -504b^2 D_1 I_3 + 156b^4 D_y I_4 \\ +2016b^2 D_{xy} I_5 \end{array} \right) & \left(\begin{array}{l} -2520b D_x I_1 + 462b^3 D_1 I_2 \\ +42b^3 D_1 I_3 - 22b^5 D_y I_4 \\ -168b^3 D_{xy} I_5 \end{array} \right) \\ \left(\begin{array}{l} 2520b D_x I_1 - 42b^3 D_1 I_2 \\ -42b^3 D_1 I_3 - 13b^5 D_y I_4 \\ +168b^3 D_{xy} I_5 \end{array} \right) & \left(\begin{array}{l} -5040D_x I_1 + 504b^2 D_1 I_2 \\ +504b^2 D_1 I_3 + 54b^4 D_y I_4 \\ -2016b^2 D_{xy} I_5 \end{array} \right) & \left(\begin{array}{l} -5040D_x I_1 + 504b^2 D_1 I_2 \\ +504b^2 D_1 I_3 + 54b^4 D_y I_4 \\ -2016b^2 D_{xy} I_5 \end{array} \right) & \left(\begin{array}{l} 1680b^2 D_x I_1 - 56b^4 D_1 I_2 \\ -56b^4 D_1 I_3 + 4b^6 D_y I_4 \\ +224b^4 D_{xy} I_5 \end{array} \right) \end{bmatrix}$$

The program is named “k_elastic_local” and the function is:

function [k_elastic_local]=k_elastic_local(Ex,Ey,vx,vy,G,t,a,b,BC,m_a)

INPUTS	OUTPUTS
Ex: Young modulus in the “x” direction. Ey: Young modulus in the “y” direction. vx: Poisson’s ratio in the “x” direction. vy: Poisson’s ratio in the “y” direction. G: Shear modulus for the material. t: Thickness of the analyzed strip element. a: Length of the strip element. b: Width of the strip element. BC: Boundary conditions of the loaded edges. m_a: Analyzed half-waves for the strip element.	k_elastic_local: The elastic stiffness matrix for the defined strip element in local coordinates.

5.2.2. Local geometric matrix sub-program

A program similar to the previous one has also been created to find the local geometric matrix. In this function we will create the local geometric matrix defining it as:

$$K_g = \begin{bmatrix} k_{gM}^{pq} & \cdot \\ \cdot & k_{gB}^{pq} \end{bmatrix}_{m \times m}$$

Where:

$$k_{gM}^{pq} = \begin{bmatrix} \frac{(3T_1 + T_2)bl_5}{12} & 0 & \frac{(T_1 + T_2)bl_5}{12} & 0 \\ 0 & \frac{(3T_1 + T_2)ba^2I_4}{12\mu_p\mu_q} & 0 & \frac{(T_1 + T_2)ba^2I_4}{12\mu_p\mu_q} \\ \frac{(T_1 + T_2)bl_5}{12} & 0 & \frac{(3T_1 + T_2)bl_5}{12} & 0 \\ 0 & \frac{(T_1 + T_2)ba^2I_4}{12\mu_p\mu_q} & 0 & \frac{(3T_1 + T_2)ba^2I_4}{12\mu_p\mu_q} \end{bmatrix}$$

$$k_{gB}^{pq} = \begin{bmatrix} \frac{(10T_1 + 3T_2)bl_5}{35} & \frac{(15T_1 + 7T_2)b^2I_5}{420} & \frac{(T_1 + T_2)bl_5}{140} & -\frac{(7T_1 + 6T_2)b^2I_5}{420} \\ \frac{(15T_1 + 7T_2)b^2I_5}{420} & \frac{(5T_1 + 3T_2)b^3I_5}{840} & \frac{(6T_1 + 7T_2)b^2I_5}{420} & -\frac{(T_1 + T_2)b^3I_5}{280} \\ \frac{(T_1 + T_2)bl_5}{140} & \frac{(6T_1 + 7T_2)b^2I_5}{840} & \frac{(3T_1 + 10T_2)bl_5}{35} & -\frac{(7T_1 + 15T_2)b^2I_5}{420} \\ -\frac{(7T_1 + 6T_2)b^2I_5}{420} & -\frac{(T_1 + T_2)b^3I_5}{280} & -\frac{(7T_1 + 15T_2)b^2I_5}{420} & \frac{(3T_1 + 5T_2)b^3I_5}{840} \end{bmatrix}$$

The program is named "k_geometric_local" and the function is:

```
function [k_geometric_local]=k_geometric_local(a,b,Ty1,Ty2,BC,m_a)
```

INPUTS	OUTPUTS
a: Length of the strip element. b: Width of the strip element. Ty1: Normal stress in the "i" node of the element. Ty2: Normal stress in the "j" node of the element. BC: Boundary conditions of the loaded edges. m_a: Analyzed half-waves for the strip element.	k_geometric_local: The geometric stiffness matrix for the defined strip element in local coordinates.

5.2.3. Boundary conditions calculator

A function to calculate the five undetermined parameters I1, I2, I3, I4 and I5 for the local elastic and geometric stiffness matrix has been created. The parameters are defined as:

$$I_1 = \int_0^a Y_p \cdot Y_q \cdot dy$$

$$I_2 = \int_0^a Y_p'' \cdot Y_q \cdot dy$$

$$I_3 = \int_0^a Y_p \cdot Y_q'' \cdot dy$$

$$I_4 = \int_0^a Y_p'' \cdot Y_q'' \cdot dy$$

$$I_5 = \int_0^a Y_p' \cdot Y_q' \cdot dy$$

As we can see, the I values depend on the two half-waves we analyze “p” and “q”.

The program is called “BCparameters” and the function is:

function [I1,I2,I3,I4,I5] = BCparameters(BC,Nm,Np,a)

INPUTS	OUTPUTS
<p>BC: Boundary conditions as defined previously:</p> <ul style="list-style-type: none"> ➔ 'S-S': Simply-simply supported boundary condition at loaded edges. ➔ 'C-C': Clamped-clamped boundary condition at loaded edges. ➔ 'S-C': Simply-clamped boundary condition at loaded edges. ➔ 'C-F': Clamped-free supported boundary condition at loaded edges. ➔ 'C-G': Clamped-guided supported boundary condition at loaded edges. <p>Nm: Half-wave number “q”.</p> <p>Np: Half-wave number “p”.</p> <p>a: Length of the analyzed strip element.</p>	<p>I1, I2, I3, I4, I5: Undetermined parameters utilized in the geometric and elastic local stiffness matrices.</p> <p>They depend on the half-waves number “p” and “q” which appear in the matrices:</p> <p>k_e^{pq} and k_g^{pq}</p>

5.2.4. Element properties

From the input elements we need to find a few parameters in order to use the other functions. For this the “elemprop” function has been created and the function is:

function [elprop]=elemprop(node,elem,nnodes,nelems)

INPUTS	OUTPUTS
node: The nodes as defined before: [node_number x_position z_position stress] elem: The elements as defined before: [element_number node_i node_j t] nnodes and nelems: Number of nodes and elements in our section.	elprop: A variable including [element_number width alpha] Where the “width” will be used for the local matrices and the “alpha” to change the local coordinates to global coordinates.

5.2.5. Rotation

This program is used to transform the local coordinate matrices to global coordinates. It utilizes a matrix to rotate the strip. The rotation depends on the angle “ α ”.

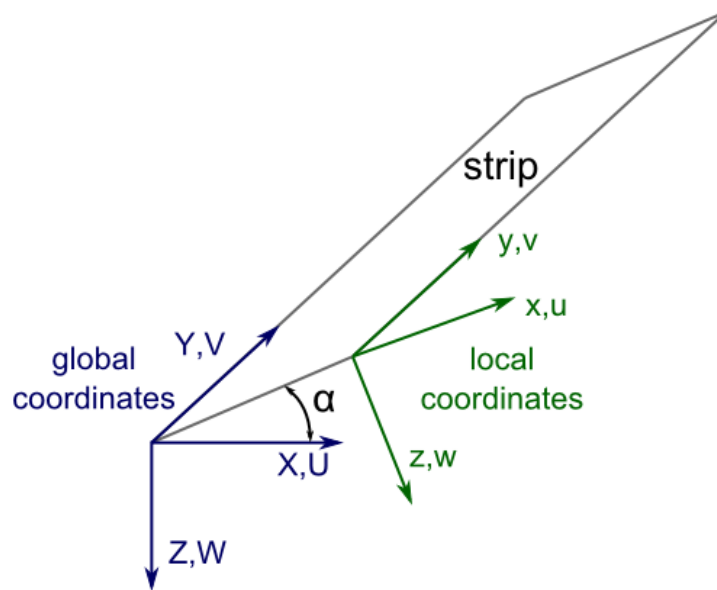


Figure 28: Strip rotation guide image

The program is called “rotatstrip” and the function is:

[k_elastic_global,k_geometric_global]=rotatstrip(alpha,k_elastic,k_geometric,m_a)

INPUTS	OUTPUTS
alpha: angle “ α ” for the strip as defined in the image. k_elastic: elastic matrix in local coordinates. k_geometric: geometric matrix in local coordinates. m_a: Analyzed half-waves for the strip element.	k_elastic_global: elastic matrix in global coordinates. k_geometric_global: geometric matrix in global coordinates.

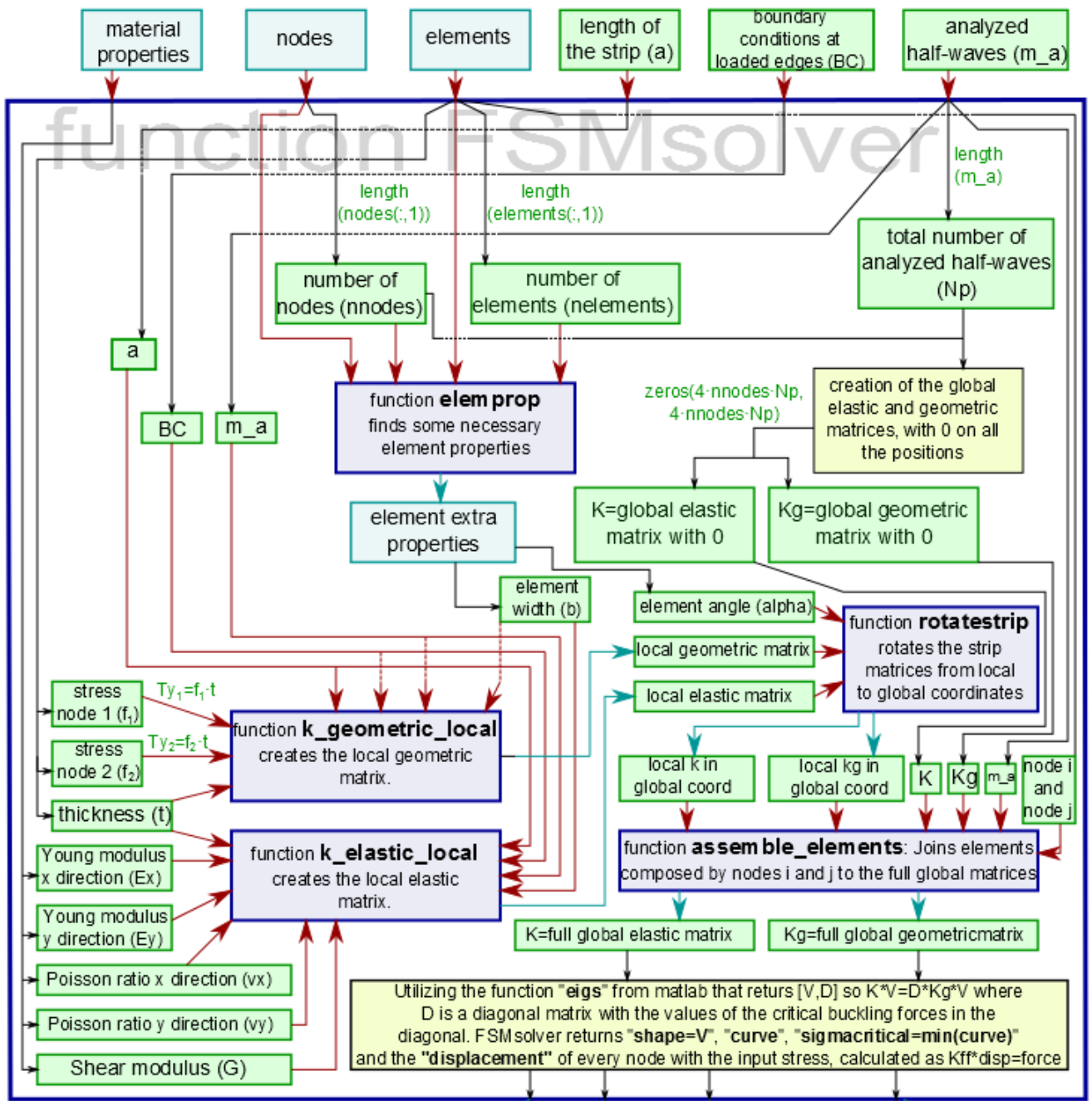
5.2.6. Assembly

Another program has been used to join all the strips. Using the strips elastic and geometric matrices in global coordinates, the function will join the particular strip to the full global matrix. The program is called “assemble_elements” and the function is:

`[K_elastic,K_geometric]=assemble_elements(K_elastic,K_geometric,k_elastic_local,k_geometric_local,nodei,nodej,nnodes,m_a)`

INPUTS	OUTPUTS
<p>K_elastic: Full elastic matrix for the section. K_geometric: Full geometric matrix for the section. k_elastic_global: Elastic matrix for the strip in global coordinates. k_geometric_global: Geometric matrix for the strip in global coordinates. nodei and nodej: Nodes of the element. nnodes: Number of nodes in the section m_a: Analyzed half-waves for the strip element.</p>	<p>K_elastic: Full elastic matrix for the section adding the elastic matrix of the strip element defined by “k_elastic_global”.</p> <p>K_geometric: Full geometric matrix for the section adding the geometric matrix of the strip element defined by “k_geometric_global”.</p>

5.3. Matlab function map



shape = contains the buckling critical shape for every eigenmode and for every half-wave number

curve = contains critical buckling stress for the first 10 buckling modes

sigmacritical = the first critical stress that will be reached

displacement = contains the displacement for the particular input stress of every node

LEGEND

input variables for a particular function	matlab function to obtain some importante variable	contains a matlab function utilized in the program
output variables for a particular function	contains some explanation of the processes the program follows	composed variable: a variable containing more than 1 number
		simple variable: a variable that contains only 1 number

6. Comparison with other methods

In order to verify the usefulness of our program it is crucial that we compare the results we obtain with other programs or theories that we know have the correct results. This way we can compare both the error and the time difference with our program.

As previously stated, our method should be much quicker due to the fact that it has many less freedom degrees than the Finite Element Method. However, we must verify that the error is acceptable.

6.1. Theoretical comparison

Utilizing the classical formulas we can find the critical buckling load for simple compression. In order to verify the level of error the program has, the theoretical values for the next coefficient have been compared with those calculated theoretically in book [3]*. The calculus have been done with a thin plate as the one in the next image.

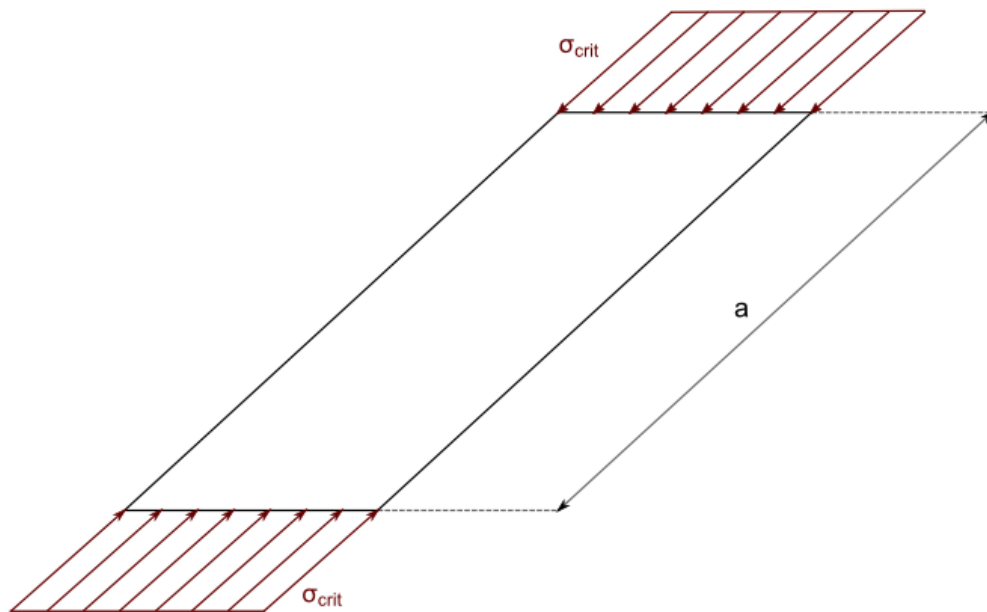


Figure 29: Thin plate used for the calculus

σ_{crit} can be defined as:

$$\sigma_{crit} = K_c \cdot E \cdot \frac{t^2}{b^2}$$

*[3]: Reference to bibliography

Where:

- K_c is a coefficient that depends on the boundary conditions and on the relation between the length and width of the plate.
- E is the Young modulus.
- t is the thickness of the plate.
- b is the width of the plate.
- a is the length of the plate.

In order to find the values for K we must create a thin plate problem. In our case we will do it with 10 nodes in the transversal direction. The complete exercise is added in the annexes. In the next table we will see the results.

$\frac{a}{b}$	Boundary conditions					
	Simply-Simply			Clamped-Clamped		
	Theoretical K	FSM program K	Difference	Theoretical K	FSM program K	Difference
0,5	3,5	3,518	0,5%	15,8	14,347	9,2%
0,6	2,5	2,432	2,7%	10	9,949	0,5%
0,7	1,8	1,778	1,2%	7,3	7,300	0%
0,9	1,2	1,067	11,1%	4,5	4,405	2,1%
1,0	0,9	0,861	4,3%	3,7	3,564	3,7%
1,2	0,7	0,594	15,1%	2,5	2,469	1,2%
1,4	0,5	0,434	13,2%	1,8	1,811	0,6%
1,8	0,25	0,260	4%	1,15	1,090	5,2%
2,0	0,2	0,209	4,5%	0,9	0,884	1,8%
2,4	0,15	0,145	3,3%	0,6	0,613	2,2%
> 3	0,1	0,092	8%	0,4	0,391	2,3%

We can see that the differences between both calculi are small enough. The differences between the values come due to the different formulation between the semi-analytical method and the FSM.

6.2. Finite Element Method Comparison

As previously stated, the Finite Element Method gives a more accurate result than the Finite Strip Method, although it demands much more computer capacity and time.

In order to calibrate the accuracy of the FSM program the values obtained with FEM are utilized and compared to calibrate the program. Values obtained from the article [6]* are utilized in the following comparison.

The whole comparison exercise can be found in the annexes. Here we will only look at the results.

Section	σ_{crit}		Error
	Finite strip method	Finite element method	
a=100, b=110, h=170, lip length=30			
t=1 mm	37,99	37,74	0,66%
t=2 mm	151,85	150,52	0,88%
t=3 mm	341,29	337,07	1,24%
t=4 mm	605,86	595,36	1,73%
t=5 mm	944,85	922,69	2,35%
Degrees of freedom	17 nodes * 4 dof = 68 degrees of freedom	17 nodes in the section * 5 nodes in the longitudinal direction * 4 dof = 340 degrees of freedom	

We can see therefore that the error percentage is very small and acceptable.

The small error that we can find is due to the fact that the finite element method does implement the shear strains into its calculus, while the finite strip method follows the Kirchhoff theory, where the shear strains are not taken into account because they are extremely small.

*[6]: Reference to bibliography

7. Conclusions

The objective of this project was to develop a program capable of implementing the FSM theory in order to solve structural problems. The new formulation of the FSM with the theories of Kirchhoff and Reissner-Mindlin for the analysis of rectangular structures is implemented to the program.

The results are well calibrated in comparison to those in the plate theory. Regarding FEM, the solutions are comparatively quite correct, with a small difference in the buckling critical coefficient of about 1%. This small error is due to the fact that FEM solutions include shear strain effects while the FSM solution does not.

Therefore, we can extract some conclusions:

- The Finite Strip Method, although being a semi-analytical theory that utilizes trigonometric Fourier series in the longitudinal direction and FEM in the transversal direction, can be correctly utilized to solve a rectangular problem with various sections.
- The dimension of the problem is greatly reduced with FSM. It is reduced to a system with two middle nodes with four displacements each. Thanks to the Fourier series, we can define the boundary conditions with trigonometric functions that satisfy them in the loaded edges.
- The boundary conditions are extremely important to determine the stiffness matrices in the FSM, given that changing them will cause some coefficients inside the matrix to vary.
- The behavior of our program is acceptable in many cases and much quicker than the FEM solution to the same problem. The requirements for the computer are greatly decreased because the unknowns in the problem are much lesser than in the finite element method.

8. Further work

Having finished the finite strip method, there are several new things that could be done in order to continue with this investigation.

The program that has been developed only solves a single column. It would be interesting to solve a whole frame utilizing the finite strip method in every column separately. However, in order to do this, further investigation must be made in the unions between the bars. In the next image we can see an illustration of what the next objective could be:

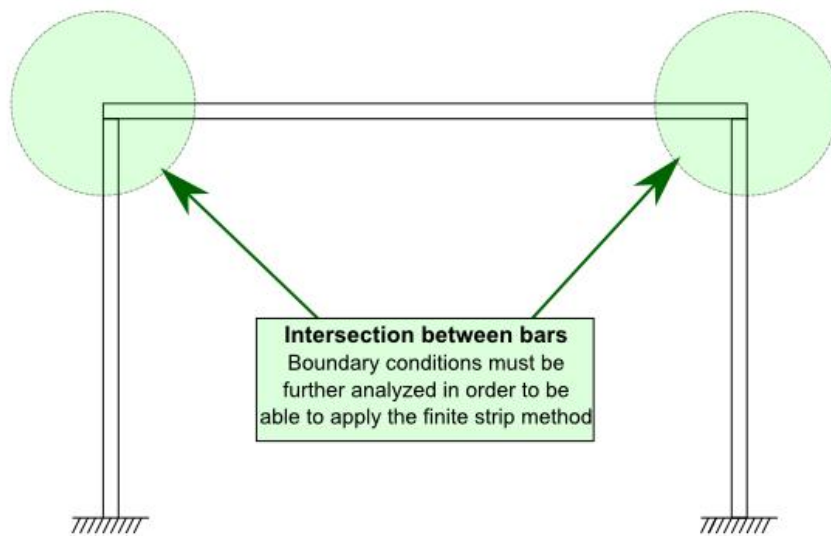


Figure 30: Frame

Although all the bars can correctly be analyzed by the Finite Strip Method when taking them separately, it requires further analysis to know if it is possible to solve a complete frame utilizing this method.

9. Bibliography

1. OÑATE, Eugenio: “Cálculo de Estructuras por el Método de los Elementos Finitos”,
2. JOVICEVIC, J and OÑATE, E: “Analysis of Beams and Shells Using a Rotation”
3. GÉMINARD, Lucien and GIET, Armand: “Stabilité des constructions”
4. Y. K. Cheung, L. G. Tham, “The Finite Strip Method”
5. LI, Z. and SCHAFER, B.W. (2010) “Buckling analysis of cold-formed steel members with general boundary conditions using CUFSM: conventional and constrained finite strip methods.” Proceedings of the 20th Int;l. Spec. Conf. on Cold-Formed Steel Structures, St. Louis, MO. November, 2010.
6. BUI HUNG CUONG, “Analysestatique du comportement des structures a parois minces par la method des elements finis et des bandesfinies de type plaque et coquesurbaisseedeformables en cisaillement”
7. TIMOSHENKO – GOUDIER, “Theory of Elasticity”

Special thanks

To Professor Sélim Datoussaïd, for being so understanding with my initial little knowledge of Finite Element Method and helping me a lot during the whole project.

To Professor Ben Schafer’s thin-walled structures research group, for creating the marvelous program CUFSM 4.05, which has been extremely helpful, together with all the theory behind it, with this project.

Annex

1. Matlab programs

1.1. FSMsolver

```
function [shape,curve,sigmacritical,kvalue] =
FSMsolver(material_properties,nodes,elements,lengths,boundary_conditio
ns,m_a)
% function to find the modal shape and the critical values for the
demanded length.
% It will also find the value of k for the calculus of the sigmacrit.
% INPUTS
% material_properties: [Ex Eyvxy G] One material per row
% with all the properties mentiones.
% nodes: [node_numberx_positionz_position stress] number of nodes x 4.
% elements: [element_numbernodeinnodej t] number of elements x 4.
% lengths=length to be analysed
% boundary_conditions
% m_a: half-waves to be analyzed
% m_a=[1 2 3 4 5] if we want to analyse the first 5 half-wave numbers.

BC=boundary_conditions;
a=lengths;
Np=length(m_a);
nnodes=length(nodes(:,1));
nelements=length(elements(:,1));

[elproperties]=elemprop(nodes,elements,nnodes,nelements);
%elproperties:[element width alfa]

% Defining of the global full matrices
K=zeros(4*nnodes*Np,4*nnodes*Np);
Kg=zeros(4*nnodes*Np,4*nnodes*Np);

fori=1:nelements
%Define the local stiffness and geometric matrixes
t=elements(i,4);
b=elproperties(i,2);
Ex=material_properties(1);
Ey=material_properties(2);
vx=material_properties(3);
vy=material_properties(4);
G=material_properties(5);
[k_el]=k_elastic_local(Ex,Ey,vx,vy,G,t,a,b,BC,m_a);
Ty1=nodes(elements(i,2),4)*t;
Ty2=nodes(elements(i,3),4)*t;
[k_gl]=k_geometric_local(a,b,Ty1,Ty2,BC,m_a);

%Transform matrixes to the global coordinates
alpha=elproperties(i,3);
[k_eg,k_gg]=rotatestrip(alpha,k_el,k_gl,m_a);

%Add the element to the full matrix
nodei=elements(i,2);
nodej=elements(i,3);
[K,Kg]=assemble_elements(K,Kg,k_eg,k_gg,nodei,nodej,nnodes,m_a);
```

```

end

R=eye(4*nnodes*Np);
Kff=R'*K*R;
Kffg=R'*Kg*R;

%function eig: returns [V,D] so K*V=D*Kg*V where D is a diagonal matrix
with
%the values of the critical buckling forces in the diagonal.
neigs=10;
options.disp=0;
options.issym=1;
N=max(min(2*neigs,length(Kff(1,:))),1);
[V,D]=eigs(full(Kffg\Kff),N,'SM',options);
curve=diag(D);
shape=V;

%The critical mode will be that one with the smallest critical value
sigmacritical=min(curve);
b=90;
kvalue=sigmacritical*b^2/(Ex*t^2);
end

```

1.2. K_elastic_local

```

function [k_elastic_local]=k_elastic_local(Ex,Ey,vx,vy,G,t,a,b,BC,m_a)
%
%Create the elastic stiffness matrix in global coordinates

% Inputs:
% Ex,Ey,vx,vy,G: material properties
% t: thickness of the element
% a: length of the strip (longitudinal)
% b: width of the strip (transversal)
% BC: ['S-S'] a string specifying boundary conditions to be analyzed:
%'S-S' simply-simply supported boundary condition
%'C-C' clamped-clamped boundary condition
%'S-C' simply-clamped supported boundary condition
%'C-F' clamped-free supported boundary condition
%'C-G' clamped-guided supported boundary condition
% m_a: longitudinal terms (or half-wave numbers) for this length.
% m_a=[1 2 3 4 5] if we want to analyse the first 5 half-wave numbers.

% Output:
% k: local stiffness matrix, a totalm x totalm matrix of 8 by 8
submatrices.
% k=[kmp]totalm x totalm block matrix
% each kmp is the 8 x 8 submatrix in the DOF order [u1 v1 u2 v2 w1
theta1 w2 theta2]';

E1=Ex/(1-vx*vy);
E2=Ey/(1-vx*vy);
Dx=Ex*t^3/(12*(1-vx*vy));
Dy=Ey*t^3/(12*(1-vx*vy));
D1=vx*Ey*t^3/(12*(1-vx*vy));
Dxy=G*t^3/12;
%
totalm = length(m_a); %Total number of longitudinal terms m
%

```

```

k_elastic_local=sparse(zeros(8*totalm,8*totalm));
z0=zeros(4,4);
for m=1:1:totalm
for p=1:1:totalm
%
km_mp=zeros(4,4);
kf_mp=zeros(4,4);
um=m_a(m)*pi;
up=m_a(p)*pi;
c1=um/a;
c2=up/a;
%
[I1,I2,I3,I4,I5] = BCparameters(BC,m_a(m),m_a(p),a);
%
%assemble the matrix of Km_mp (K membrane for m and p half-waves)
km_mp(1,1)=E1*I1/b+G*b*I5/3;
km_mp(1,2)=E2*vx*(-1/2/c2)*I3-G*I5/2/c2;
km_mp(1,3)=-E1*I1/b+G*b*I5/6;
km_mp(1,4)=E2*vx*(-1/2/c2)*I3+G*I5/2/c2;

km_mp(2,1)=E2*vx*(-1/2/c1)*I2-G*I5/2/c1;
km_mp(2,2)=E2*b*I4/3/c1/c2+G*I5/b/c1/c2;
km_mp(2,3)=E2*vx*(1/2/c1)*I2-G*I5/2/c1;
km_mp(2,4)=E2*b*I4/6/c1/c2-G*I5/b/c1/c2;

km_mp(3,1)=-E1*I1/b+G*b*I5/6;
km_mp(3,2)=E2*vx*(1/2/c2)*I3-G*I5/2/c2;
km_mp(3,3)=E1*I1/b+G*b*I5/3;
km_mp(3,4)=E2*vx*(1/2/c2)*I3+G*I5/2/c2;

km_mp(4,1)=E2*vx*(-1/2/c1)*I2+G*I5/2/c1;
km_mp(4,2)=E2*b*I4/6/c1/c2-G*I5/b/c1/c2;
km_mp(4,3)=E2*vx*(1/2/c1)*I2+G*I5/2/c1;
km_mp(4,4)=E2*b*I4/3/c1/c2+G*I5/b/c1/c2;
km_mp=km_mp*t;
%
%
%assemble the matrix of Kf_mp (K bending for m and p half-waves)
kf_mp(1,1)=(5040*Dx*I1-504*b^2*D1*I2-
504*b^2*D1*I3+156*b^4*Dy*I4+2016*b^2*Dxy*I5)/420/b^3;
kf_mp(1,2)=(2520*b*Dx*I1-462*b^3*D1*I2-
42*b^3*D1*I3+22*b^5*Dy*I4+168*b^3*Dxy*I5)/420/b^3;
kf_mp(1,3)=(-
5040*Dx*I1+504*b^2*D1*I2+504*b^2*D1*I3+54*b^4*Dy*I4-
2016*b^2*Dxy*I5)/420/b^3;
kf_mp(1,4)=(2520*b*Dx*I1-42*b^3*D1*I2-42*b^3*D1*I3-
13*b^5*Dy*I4+168*b^3*Dxy*I5)/420/b^3;

kf_mp(2,1)=(2520*b*Dx*I1-462*b^3*D1*I3-
42*b^3*D1*I2+22*b^5*Dy*I4+168*b^3*Dxy*I5)/420/b^3;
kf_mp(2,2)=(1680*b^2*Dx*I1-56*b^4*D1*I2-
56*b^4*D1*I3+4*b^6*Dy*I4+224*b^4*Dxy*I5)/420/b^3;
kf_mp(2,3)=(-
2520*b*Dx*I1+42*b^3*D1*I2+42*b^3*D1*I3+13*b^5*Dy*I4-
168*b^3*Dxy*I5)/420/b^3;
kf_mp(2,4)=(840*b^2*Dx*I1+14*b^4*D1*I2+14*b^4*D1*I3-
3*b^6*Dy*I4-56*b^4*Dxy*I5)/420/b^3;

kf_mp(3,1)=kf_mp(1,3);
kf_mp(3,2)=kf_mp(2,3);

```

```

kf_mp(3,3)=(5040*Dx*I1-504*b^2*D1*I2-
504*b^2*D1*I3+156*b^4*Dy*I4+2016*b^2*Dxy*I5)/420/b^3;
kf_mp(3,4)=(-2520*b*Dx*I1+462*b^3*D1*I2+42*b^3*D1*I3-
22*b^5*Dy*I4-168*b^3*Dxy*I5)/420/b^3;

kf_mp(4,1)=kf_mp(1,4);
kf_mp(4,2)=kf_mp(2,4);
kf_mp(4,3)=(-2520*b*Dx*I1+462*b^3*D1*I3+42*b^3*D1*I2-
22*b^5*Dy*I4-168*b^3*Dxy*I5)/420/b^3;%not symmetric
kf_mp(4,4)=(1680*b^2*Dx*I1-56*b^4*D1*I2-
56*b^4*D1*I3+4*b^6*Dy*I4+224*b^4*Dxy*I5)/420/b^3;

%assemble the membrane and bending stiffness matrices
kmp=[km_mp z0
z0 kf_mp];
%add it into local element stiffness matrix by corresponding to half-
wave m
k_elastic_local(8*(m-1)+1:8*m,8*(p-1)+1:8*p)=kmp;
end
end

```

1.3. K_geometric_local

```

function [k_geometric_local]=k_geometric_local(a,b,Ty1,Ty2,BC,m_a)
%
%Generate geometric stiffness matrix (kg) in local coordinates

% Inputs:
% a: length of the strip in longitudinal direction
% b: width of the strip in transverse direction
% Ty1, Ty2: node stresses
% BC: a string specifying boundary conditions to be analyzed:
%'S-S' simply-pimply supported boundary condition at loaded edges
%'C-C' clamped-clamped boundary condition at loaded edges
%'S-C' simply-clamped supported boundary condition at loaded edges
%'C-F' clamped-free supported boundary condition at loaded edges
%'C-G' clamped-guided supported boundary condition at loaded edges
% m_a: longitudinal terms (or half-wave numbers) for this length.
% m_a=[1 2 3 4 5] if we want to analyse the first 5 half-wave numbers.

% Output:
% kg: local geometric stiffness matrix, a totalm x totalm matrix of 8
by 8 submatrices.
% kg=[kgmp]totalm x totalm block matrix
% each kgmp is the 8 x 8 submatrix in the DOF order [u1 v1 u2 v2 w1
thetal
% w2 theta2]';

totalm = length(m_a); %Total number of longitudinal terms m
kg=sparse(zeros(8*totalm,8*totalm));
%
for m=1:1:totalm
for p=1:1:totalm
%
gm_mp=zeros(4,4);
z0=zeros(4,4);
gf_mp=zeros(4,4);
um=m_a(m)*pi;
up=m_a(p)*pi;

```



```

%
      [I1,I2,I3,I4,I5] = BCparameters(BC,m_a(m),m_a(p),a);
%
%Membrane geometrical matrix
gm_mp(1,1)=b*(3*Ty1+Ty2)*I5/12;
gm_mp(1,3)=b*(Ty1+Ty2)*I5/12;
gm_mp(3,1)=gm_mp(1,3);
gm_mp(2,2)=b*a^2*(3*Ty1+Ty2)*I4/12/um/up;
gm_mp(2,4)=b*a^2*(Ty1+Ty2)*I4/12/um/up;
gm_mp(4,2)=gm_mp(2,4);
gm_mp(3,3)=b*(Ty1+3*Ty2)*I5/12;
gm_mp(4,4)=b*a^2*(Ty1+3*Ty2)*I4/12/um/up;
%
%Bending geometrical matrix
gf_mp(1,1)=(10*Ty1+3*Ty2)*b*I5/35;
gf_mp(1,2)=(15*Ty1+7*Ty2)*b^2*I5/210/2;
gf_mp(2,1)=gf_mp(1,2);
gf_mp(1,3)=9*(Ty1+Ty2)*b*I5/140;
gf_mp(3,1)=gf_mp(1,3);
gf_mp(1,4)=-(7*Ty1+6*Ty2)*b^2*I5/420;
gf_mp(4,1)=gf_mp(1,4);
gf_mp(2,2)=(5*Ty1+3*Ty2)*b^3*I5/2/420;
gf_mp(2,3)=(6*Ty1+7*Ty2)*b^2*I5/420;
gf_mp(3,2)=gf_mp(2,3);
gf_mp(2,4)=-(Ty1+Ty2)*b^3*I5/140/2;
gf_mp(4,2)=gf_mp(2,4);
gf_mp(3,3)=(3*Ty1+10*Ty2)*b*I5/35;
gf_mp(3,4)=-(7*Ty1+15*Ty2)*b^2*I5/420;
gf_mp(4,3)=gf_mp(3,4);
gf_mp(4,4)=(3*Ty1+5*Ty2)*b^3*I5/420/2;
%assemble the membrane and bending stiffness matrices
kgmp=[gm_mp  z0
z0  gf_mp];
%add it into local geometric stiffness matrix by corresponding to m
%half-wave
k_geometric_local(8*(m-1)+1:8*m,8*(p-1)+1:8*p)=kgmp;
end
end

```

1.4. elemprop

```

function [elprop]=elemprop(node,elem,nnodes,nelems)
% Function to obtain several properties of the element used in the
analysis
% INPUT
% node: [node# x_positionz_position stress] nnodes x 8;
% elem: [elem# nodeinnodej t] nelems x 4;
% OUTPUT
% elprop: [elem# width alpha]
%
elprop=zeros(nelems,3);
%
fori=1:nelems
nodei = elem(i,2);
nodej = elem(i,3);
xi = node(nodei,2);
zi = node(nodei,3);
xj = node(nodej,2);
zj = node(nodej,3);
dx = xj - xi;

```

```

dz = zj - zi;
width = sqrt(dx^2 + dz^2);
alpha = atan2(dz,dx);
elprop(i,:)= [i width alpha];
end

```

1.5. BCparameters

```

function [I1,I2,I3,I4,I5] = BCparameters (BC,Nm,Np,a)
%
% Calculate the 5 undetermined parameters I1,I2,I3,I4,I5 for local
elastic
% and geometric stiffness matrices.
% BC: a string specifying boundary conditions to be analyzed:
%'S-S' simply-pimply supported boundary condition at loaded edges
%'C-C' clamped-clamped boundary condition at loaded edges
%'S-C' simply-clamped supported boundary condition at loaded edges
%'C-F' clamped-free supported boundary condition at loaded edges
%'C-G' clamped-guided supported boundary condition at loaded edges
%Outputs:
%I1,I2,I3,I4,I5
%calculation of I1 is the integration of Ym*Yn from 0 to a
%calculation of I2 is the integration of Ym''*Yn from 0 to a
%calculation of I3 is the integration of Ym*Yn'' from 0 to a
%calculation of I3 is the integration of Ym*Yn'' from 0 to a
%calculation of I4 is the integration of Ym''*Yn'' from 0 to a
%calculation of I5 is the integration of Ym'*Yn' from 0 to a

ifstrcmp (BC,'S-S')
%For simply-pimply supported boundary condition at loaded edges
if Nm==Np
    I1=a/2;
    I2=-Nm^2*pi^2/a/2;
    I3=-Np^2*pi^2/a/2;
    I4=pi^4*Nm^4/2/a^3;
    I5=pi^2*Nm^2/2/a;
else
    I1=0;
    I2=0;
    I3=0;
    I4=0;
    I5=0;
end
elseifstrcmp (BC,'C-C')
%For Clamped-clamped boundary condition at loaded edges
if Nm==Np
if Nm==1
    I1=3*a/8;
else
    I1=a/4;
end
    I2=-(Nm^2+1)*pi^2/4/a;
    I3=-(Np^2+1)*pi^2/4/a;
    I4=pi^4*((Nm^2+1)^2+4*Nm^2)/4/a^3;
    I5=(1+Nm^2)*pi^2/4/a;
else
if Nm-Np==2
    I1=-a/8;
    I2=(Nm^2+1)*pi^2/8/a-Nm*pi^2/4/a;
    I3=(Np^2+1)*pi^2/8/a+Np*pi^2/4/a;

```

```

I4=- (Nm-1) ^2* (Np+1) ^2*pi^4/8/a^3;
      I5=- (1+Nm*Np) *pi^2/8/a;
elseif Nm-Np==-2
I1=-a/8;
      I2=(Nm^2+1) *pi^2/8/a+Nm*pi^2/4/a;
      I3=(Np^2+1) *pi^2/8/a-Np*pi^2/4/a;
      I4=- (Nm+1) ^2* (Np-1) ^2*pi^4/8/a^3;
      I5=- (1+Nm*Np) *pi^2/8/a;
else
      I1=0;
      I2=0;
      I3=0;
      I4=0;
      I5=0;
end
end
elseif strcmp(BC, 'S-C')
%For simply-clamped supported boundary condition at loaded edges
if Nm==Np
      I1=(1+ (Nm+1) ^2/Nm^2) *a/2;
      I2=- (Nm+1) ^2*pi^2/a;
      I3=- (Nm+1) ^2*pi^2/a;
      I4=(Nm+1) ^2*pi^4* ( (Nm+1) ^2+Nm^2) /2/a^3;
      I5=(1+Nm) ^2*pi^2/a;
else
if Nm-Np==1
      I1=(Nm+1) *a/2/Nm;
      I2=- (Nm+1) *Nm*pi^2/2/a;
      I3=- (Np+1) ^2*pi^2* (Nm+1) /2/a/Nm;
      I4=(Nm+1) *Nm* (Np+1) ^2*pi^4/2/a^3;
      I5=(1+Nm) * (1+Np) *pi^2/2/a;
elseif Nm-Np==-1
      I1=(Np+1) *a/2/Np;
      I2=- (Nm+1) ^2*pi^2* (Np+1) /2/a/Np;
      I3=- (Np+1) *Np*pi^2/2/a;
      I4=(Nm+1) ^2*Np* (Np+1) *pi^4/2/a^3;
      I5=(1+Nm) * (1+Np) *pi^2/2/a;
else
      I1=0;
      I2=0;
      I3=0;
      I4=0;
      I5=0;
end
end
%
elseif strcmp(BC, 'C-F')
%For clamped-free supported boundary condition at loaded edges
if Nm==Np
      I1=3*a/2-2*a* (-1) ^ (Nm-1) / (Nm-1/2) /pi;
      I2=(Nm-1/2) ^2*pi^2* ( (-1) ^ (Nm-1) / (Nm-1/2) /pi-1/2) /a;
      I3=(Np-1/2) ^2*pi^2* ( (-1) ^ (Np-1) / (Np-1/2) /pi-1/2) /a;
      I4=(Nm-1/2) ^4*pi^4/2/a^3;
      I5=(Nm-1/2) ^2*pi^2/2/a;
else
      I1=a-a* (-1) ^ (Nm-1) / (Nm-1/2) /pi-a* (-1) ^ (Np-1) / (Np-1/2) /pi;
      I2=(Nm-1/2) ^2*pi^2* ( (-1) ^ (Nm-1) / (Nm-1/2) /pi) /a;
      I3=(Np-1/2) ^2*pi^2* ( (-1) ^ (Np-1) / (Np-1/2) /pi) /a;
      I4=0;
      I5=0;
end

```

```

elseif strcmp(BC, 'C-G')
%For clamped-guided supported boundary condition at loaded edges
if Nm==Np
if Nm==1
I1=3*a/8;
else
I1=a/4;
end
I2=-((Nm-1/2)^2+1/4)*pi^2/a/4;
I3=-((Nm-1/2)^2+1/4)*pi^2/a/4;
I4=((Nm-1/2)^2+1/4)^2*pi^4/4/a^3+(Nm-1/2)^2*pi^4/4/a^3;
I5=(Nm-1/2)^2*pi^2/a/4+pi^2/16/a;
else
if Nm-Np==1
I1=-a/8;
I2=((Nm-1/2)^2+1/4)*pi^2/a/8-(Nm-1/2)*pi^2/a/8;
I3=((Np-1/2)^2+1/4)*pi^2/a/8+(Np-1/2)*pi^2/a/8;
I4=-Np^4*pi^4/8/a^3;
I5=-Np^2*pi^2/8/a;
elseif Nm-Np==-1
I1=-a/8;
I2=((Nm-1/2)^2+1/4)*pi^2/a/8+(Nm-1/2)*pi^2/a/8;
I3=((Np-1/2)^2+1/4)*pi^2/a/8-(Np-1/2)*pi^2/a/8;
I4=-Nm^4*pi^4/8/a^3;
I5=-Nm^2*pi^2/8/a;
else
I1=0;
I2=0;
I3=0;
I4=0;
I5=0;
end
end
end
end

```

1.6. rotatestrip

```

function
[k_elastic_global,k_geometric_global]=rotatestrip(alpha,k_elastic,k_ge
ometric,m_a)

% Transfer the local stiffness matrix into the global coordinates
% OUTPUT
% k_elastic_global: k_elastic input in global coordinates (rotating
input angle alpha)
% k_geometric_global: k_geometric input in global coordinates
(rotating input angle alpha)
% INPUT
% k_elastic and k_geometric: local matrices that must be changed to
global coordinates
% alpha: angle for turning matrices to the global coordinates
% m_a: half-wave numbers to be analysed
% m_a=[1 2 3 4 5] if we want to analyse the first 5 half-wave numbers.

totalm = length(m_a); %Total number of longitudinal terms m
a=alpha;
%
z0=0;
gam=[cos(a)    z0    z0z0    -sin(a) z0    z0z0
      z0    1    z0    z0z0z0z0z0z0

```

```

        z0      z0cos(a)  z0      z0z0 -sin(a)  z0
        z0      z0z0      1      z0      z0z0z0
sin(a) z0      z0z0cos(a)  z0      z0z0
        z0      z0z0z0z0  1      z0      z0
        z0      z0      sin(a)  z0      z0z0cos(a)  z0
z0      z0z0z0z0z0z0z01 ];
%extend to multi-m
fori=1:totalm
gamma(8*(i-1)+1:8*i,8*(i-1)+1:8*i)=gam;
end
%
k_elastic_global=gamma*k_elastic*gamma';
k_geometric_global=gamma*k_geometric*gamma';
end

```

1.7. assemble_elements

```

function
[K_elastic,K_geometric]=assemble_elements(K_elastic,K_geometric,k_elas
tic_global,k_geometric_global,nodei,nodej,nnodes,m_a)
%
% This function adds the local matrices to the global matrices.
% INPUT
% K_elastic: full elastic stiffness matrix without the local input
% K_geometric: full geometric stiffness matrix without the local input
% k_elastic_global: global coordinates elastic stiffness matrix to be
added
% k_geometric_global: global coordinates geometric stiffness matrix to
be added
% nodei: one of the nodes of the analysed strip element
% nodej: the other node of the analysed strip element
% nnodes: number of total node in out section
% m_a: half-waves analysed. m_a=[1 2 3 4 5] if we want to analyse the
first 5 half-waves

% OUTPUTS
% K_elastic: full elastic stiffness matrix with the added input
elastic matrix
% K_geometric: full geometric stiffness matrix with the added input
geometric matrix

totalm = length(m_a); %Total number of half-wave terms m
K2=sparse(zeros(4*nnodes*totalm,4*nnodes*totalm));
K3=sparse(zeros(4*nnodes*totalm,4*nnodes*totalm));
skip=2*nnodes;
fori=1:1:totalm
for j=1:1:totalm
%Submatrices for the initial elastic stiffness
k11=k_elastic_global(8*(i-1)+1:8*(i-1)+2,8*(j-1)+1:8*(j-1)+2);
k12=k_elastic_global(8*(i-1)+1:8*(i-1)+2,8*(j-1)+3:8*(j-1)+4);
k13=k_elastic_global(8*(i-1)+1:8*(i-1)+2,8*(j-1)+5:8*(j-1)+6);
k14=k_elastic_global(8*(i-1)+1:8*(i-1)+2,8*(j-1)+7:8*(j-1)+8);
k21=k_elastic_global(8*(i-1)+3:8*(i-1)+4,8*(j-1)+1:8*(j-1)+2);
k22=k_elastic_global(8*(i-1)+3:8*(i-1)+4,8*(j-1)+3:8*(j-1)+4);
k23=k_elastic_global(8*(i-1)+3:8*(i-1)+4,8*(j-1)+5:8*(j-1)+6);
k24=k_elastic_global(8*(i-1)+3:8*(i-1)+4,8*(j-1)+7:8*(j-1)+8);
k31=k_elastic_global(8*(i-1)+5:8*(i-1)+6,8*(j-1)+1:8*(j-1)+2);
k32=k_elastic_global(8*(i-1)+5:8*(i-1)+6,8*(j-1)+3:8*(j-1)+4);
k33=k_elastic_global(8*(i-1)+5:8*(i-1)+6,8*(j-1)+5:8*(j-1)+6);
k34=k_elastic_global(8*(i-1)+5:8*(i-1)+6,8*(j-1)+7:8*(j-1)+8);

```

```

k41=k_elastic_global(8*(i-1)+7:8*(i-1)+8,8*(j-1)+1:8*(j-1)+2);
k42=k_elastic_global(8*(i-1)+7:8*(i-1)+8,8*(j-1)+3:8*(j-1)+4);
k43=k_elastic_global(8*(i-1)+7:8*(i-1)+8,8*(j-1)+5:8*(j-1)+6);
k44=k_elastic_global(8*(i-1)+7:8*(i-1)+8,8*(j-1)+7:8*(j-1)+8);
%
K2(4*nnodes*(i-1)+nodei*2-1:4*nnodes*(i-1)+nodei*2,4*nnodes*(j-
1)+nodei*2-1:4*nnodes*(j-1)+nodei*2)=k11;
K2(4*nnodes*(i-1)+nodei*2-1:4*nnodes*(i-1)+nodei*2,4*nnodes*(j-
1)+nodej*2-1:4*nnodes*(j-1)+nodej*2)=k12;
K2(4*nnodes*(i-1)+nodej*2-1:4*nnodes*(i-1)+nodej*2,4*nnodes*(j-
1)+nodei*2-1:4*nnodes*(j-1)+nodei*2)=k21;
K2(4*nnodes*(i-1)+nodej*2-1:4*nnodes*(i-1)+nodej*2,4*nnodes*(j-
1)+nodej*2-1:4*nnodes*(j-1)+nodej*2)=k22;
%
K2(4*nnodes*(i-1)+skip+nodei*2-1:4*nnodes*(i-
1)+skip+nodei*2,4*nnodes*(j-1)+skip+nodei*2-1:4*nnodes*(j-
1)+skip+nodei*2)=k33;
K2(4*nnodes*(i-1)+skip+nodei*2-1:4*nnodes*(i-
1)+skip+nodei*2,4*nnodes*(j-1)+skip+nodej*2-1:4*nnodes*(j-
1)+skip+nodej*2)=k34;
K2(4*nnodes*(i-1)+skip+nodej*2-1:4*nnodes*(i-
1)+skip+nodej*2,4*nnodes*(j-1)+skip+nodei*2-1:4*nnodes*(j-
1)+skip+nodei*2)=k43;
K2(4*nnodes*(i-1)+skip+nodej*2-1:4*nnodes*(i-
1)+skip+nodej*2,4*nnodes*(j-1)+skip+nodej*2-1:4*nnodes*(j-
1)+skip+nodej*2)=k44;
%
K2(4*nnodes*(i-1)+nodei*2-1:4*nnodes*(i-1)+nodei*2,4*nnodes*(j-
1)+skip+nodei*2-1:4*nnodes*(j-1)+skip+nodei*2)=k13;
K2(4*nnodes*(i-1)+nodei*2-1:4*nnodes*(i-1)+nodei*2,4*nnodes*(j-
1)+skip+nodej*2-1:4*nnodes*(j-1)+skip+nodej*2)=k14;
K2(4*nnodes*(i-1)+nodej*2-1:4*nnodes*(i-1)+nodej*2,4*nnodes*(j-
1)+skip+nodei*2-1:4*nnodes*(j-1)+skip+nodei*2)=k23;
K2(4*nnodes*(i-1)+nodej*2-1:4*nnodes*(i-1)+nodej*2,4*nnodes*(j-
1)+skip+nodej*2-1:4*nnodes*(j-1)+skip+nodej*2)=k24;
%
K2(4*nnodes*(i-1)+skip+nodei*2-1:4*nnodes*(i-
1)+skip+nodei*2,4*nnodes*(j-1)+nodei*2-1:4*nnodes*(j-1)+nodei*2)=k31;
K2(4*nnodes*(i-1)+skip+nodei*2-1:4*nnodes*(i-
1)+skip+nodei*2,4*nnodes*(j-1)+nodej*2-1:4*nnodes*(j-1)+nodej*2)=k32;
K2(4*nnodes*(i-1)+skip+nodej*2-1:4*nnodes*(i-
1)+skip+nodej*2,4*nnodes*(j-1)+nodei*2-1:4*nnodes*(j-1)+nodei*2)=k41;
K2(4*nnodes*(i-1)+skip+nodej*2-1:4*nnodes*(i-
1)+skip+nodej*2,4*nnodes*(j-1)+nodej*2-1:4*nnodes*(j-1)+nodej*2)=k42;
%
%Submatrices for the initial geometric stiffness
kg11=k_geometric_global(8*(i-1)+1:8*(i-1)+2,8*(j-1)+1:8*(j-
1)+2);
kg12=k_geometric_global(8*(i-1)+1:8*(i-1)+2,8*(j-1)+3:8*(j-
1)+4);
kg13=k_geometric_global(8*(i-1)+1:8*(i-1)+2,8*(j-1)+5:8*(j-
1)+6);
kg14=k_geometric_global(8*(i-1)+1:8*(i-1)+2,8*(j-1)+7:8*(j-
1)+8);
kg21=k_geometric_global(8*(i-1)+3:8*(i-1)+4,8*(j-1)+1:8*(j-
1)+2);
kg22=k_geometric_global(8*(i-1)+3:8*(i-1)+4,8*(j-1)+3:8*(j-
1)+4);
kg23=k_geometric_global(8*(i-1)+3:8*(i-1)+4,8*(j-1)+5:8*(j-
1)+6);

```

```

kg24=k_geometric_global(8*(i-1)+3:8*(i-1)+4,8*(j-1)+7:8*(j-
1)+8);
kg31=k_geometric_global(8*(i-1)+5:8*(i-1)+6,8*(j-1)+1:8*(j-
1)+2);
kg32=k_geometric_global(8*(i-1)+5:8*(i-1)+6,8*(j-1)+3:8*(j-
1)+4);
kg33=k_geometric_global(8*(i-1)+5:8*(i-1)+6,8*(j-1)+5:8*(j-
1)+6);
kg34=k_geometric_global(8*(i-1)+5:8*(i-1)+6,8*(j-1)+7:8*(j-
1)+8);
kg41=k_geometric_global(8*(i-1)+7:8*(i-1)+8,8*(j-1)+1:8*(j-
1)+2);
kg42=k_geometric_global(8*(i-1)+7:8*(i-1)+8,8*(j-1)+3:8*(j-
1)+4);
kg43=k_geometric_global(8*(i-1)+7:8*(i-1)+8,8*(j-1)+5:8*(j-
1)+6);
kg44=k_geometric_global(8*(i-1)+7:8*(i-1)+8,8*(j-1)+7:8*(j-
1)+8);
%
K3(4*nnodes*(i-1)+nodei*2-1:4*nnodes*(i-1)+nodei*2,4*nnodes*(j-
1)+nodei*2-1:4*nnodes*(j-1)+nodei*2)=kg11;
K3(4*nnodes*(i-1)+nodei*2-1:4*nnodes*(i-1)+nodei*2,4*nnodes*(j-
1)+nodej*2-1:4*nnodes*(j-1)+nodej*2)=kg12;
K3(4*nnodes*(i-1)+nodej*2-1:4*nnodes*(i-1)+nodej*2,4*nnodes*(j-
1)+nodei*2-1:4*nnodes*(j-1)+nodei*2)=kg21;
K3(4*nnodes*(i-1)+nodej*2-1:4*nnodes*(i-1)+nodej*2,4*nnodes*(j-
1)+nodej*2-1:4*nnodes*(j-1)+nodej*2)=kg22;
%
K3(4*nnodes*(i-1)+skip+nodei*2-1:4*nnodes*(i-
1)+skip+nodei*2,4*nnodes*(j-1)+skip+nodei*2-1:4*nnodes*(j-
1)+skip+nodei*2)=kg33;
K3(4*nnodes*(i-1)+skip+nodei*2-1:4*nnodes*(i-
1)+skip+nodei*2,4*nnodes*(j-1)+skip+nodej*2-1:4*nnodes*(j-
1)+skip+nodej*2)=kg34;
K3(4*nnodes*(i-1)+skip+nodej*2-1:4*nnodes*(i-
1)+skip+nodej*2,4*nnodes*(j-1)+skip+nodei*2-1:4*nnodes*(j-
1)+skip+nodei*2)=kg43;
K3(4*nnodes*(i-1)+skip+nodej*2-1:4*nnodes*(i-
1)+skip+nodej*2,4*nnodes*(j-1)+skip+nodej*2-1:4*nnodes*(j-
1)+skip+nodej*2)=kg44;
%
K3(4*nnodes*(i-1)+nodei*2-1:4*nnodes*(i-1)+nodei*2,4*nnodes*(j-
1)+skip+nodei*2-1:4*nnodes*(j-1)+skip+nodei*2)=kg13;
K3(4*nnodes*(i-1)+nodei*2-1:4*nnodes*(i-1)+nodei*2,4*nnodes*(j-
1)+skip+nodej*2-1:4*nnodes*(j-1)+skip+nodej*2)=kg14;
K3(4*nnodes*(i-1)+nodej*2-1:4*nnodes*(i-1)+nodej*2,4*nnodes*(j-
1)+skip+nodei*2-1:4*nnodes*(j-1)+skip+nodei*2)=kg23;
K3(4*nnodes*(i-1)+nodej*2-1:4*nnodes*(i-1)+nodej*2,4*nnodes*(j-
1)+skip+nodej*2-1:4*nnodes*(j-1)+skip+nodej*2)=kg24;
%
K3(4*nnodes*(i-1)+skip+nodei*2-1:4*nnodes*(i-
1)+skip+nodei*2,4*nnodes*(j-1)+nodei*2-1:4*nnodes*(j-1)+nodei*2)=kg31;
K3(4*nnodes*(i-1)+skip+nodei*2-1:4*nnodes*(i-
1)+skip+nodei*2,4*nnodes*(j-1)+nodej*2-1:4*nnodes*(j-1)+nodej*2)=kg32;
K3(4*nnodes*(i-1)+skip+nodej*2-1:4*nnodes*(i-
1)+skip+nodej*2,4*nnodes*(j-1)+nodei*2-1:4*nnodes*(j-1)+nodei*2)=kg41;
K3(4*nnodes*(i-1)+skip+nodej*2-1:4*nnodes*(i-
1)+skip+nodej*2,4*nnodes*(j-1)+nodej*2-1:4*nnodes*(j-1)+nodej*2)=kg42;
end
end
K_elastic=K_elastic+K2;

```

$K_{\text{geometric}} = K_{\text{geometric}} + K_3;$

2. Comparison exercises

2.1. Theoretical comparison exercise

In order to do the following exercise we must describe the thin plate we will be using. We can now see the image of the thin plate and the input for the FSM matlab program.

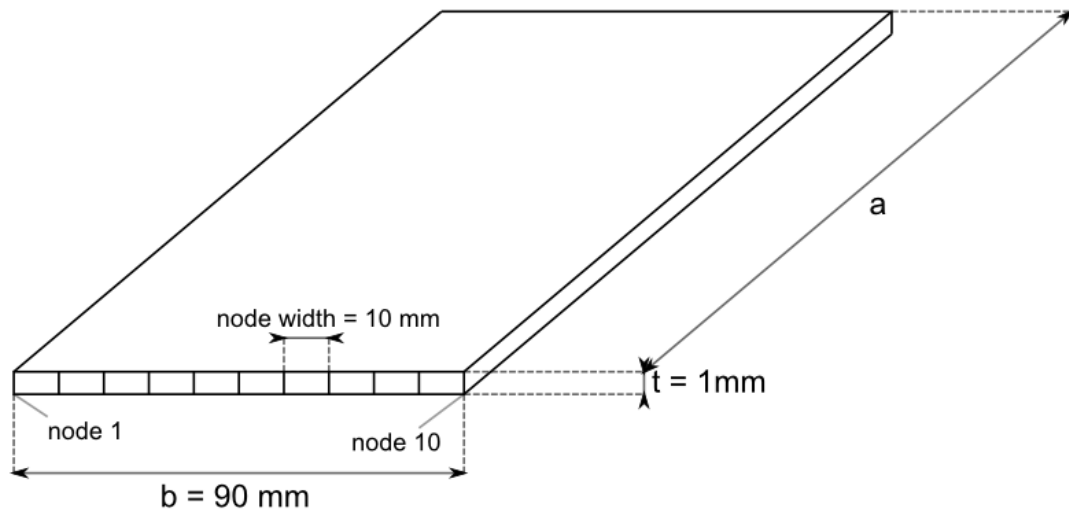


Figure 31: Theoretical comparison thin plate

The input for the matlab FSM program in this case will be:

Thin plate

```
Ex=2.05e5; %Given that we will be working in "N" and "mm" the value of our Young modulus is 205000 N/mm2
```

```
Ey=2.05e5;
```

```
vx=0.3;
```

```
vy=0.3;
```

```
G=Ex/(2*(1+vx));
```

```
t=1;
```

```
m_a=[1]; %We will only analyze the first halfwave buckling modes.
```

```
nodes=[1 0 0 1; 2 10 0 1; 3 20 0 1; 4 30 0 1; 5 40 0 1; 6 50 0 1; 7 60 0 1; 8 70 0 1; 9 80 0 1; 10 90 0 1] %[node_number x_position z_position stress]. In this case we put a normal compression of 1N to obtain directly the critical buckling load in the analysis.
```

```
elements=[1 1 2 t; 2 2 3 t; 3 3 4 t; 4 4 5 t; 5 5 6 t; 6 6 7 t; 7 7 8 t; 8 8 9 t; 9 9 10 t]
```

boundary_conditions=variable;%our boundary conditions will be 'S-S' or 'C-C'
depending on what K we want to calculate.

material_properties=[Ex Eyvxvy G];

lengths=variable;%The length of the strip is "a" and the relation of a/b is what we will
keep changing to calculate our values for "K".

2.2. Finite Element Method Comparison exercise

The following C section will be analyzed

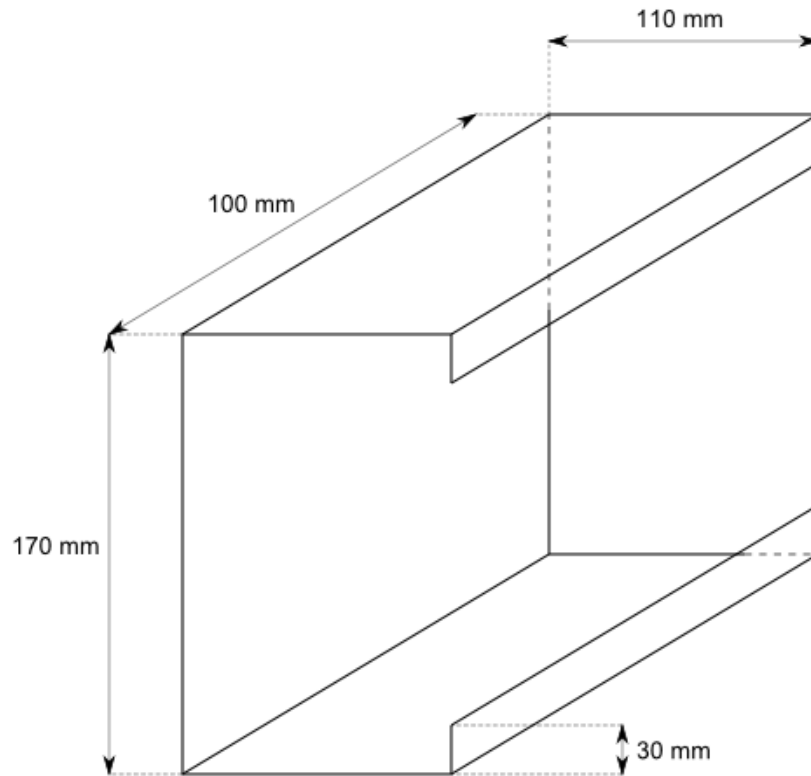


Figure 32: Analyzed figure in the comparison

The section is discretized in the next way.

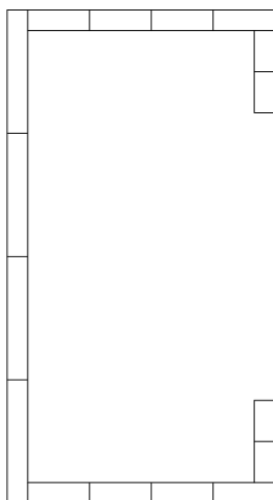


Figure 33: Discretization of the section

Therefore, we have 17 nodes and 16 elements in the cross section. In the case of the finite element method, 5 more nodes have been created in the longitudinal direction.

The freedom degrees for both methods have also been calculated, for they are proportional to the computer power required to solve the equations. The increase of the unknowns increase the time the computer needs to solve the structure.

With the figure described, it can now be introduced to the program in order to make the comparison.

Finite Strip Method program:

```
Ex=2.05e5;
Ey=2.05e5;
vx=0.3;
vy=0.3;
G=Ex/(2*(1+vx));
t=5;%The value of "t" must be changed in order to get the
whole table of the report.
m_a=[1 2 3 4 5];
nodes=[1 110 30 1;2 110 15 1;3 110 0 1;4 82.5 0 1;5 55 0
1;6 27.5 0 1;7 0 0 1;8 0 42.5 1;9 0 85 1;10 0 127.5 1;11 0
170 1;12 27.5 170 1;13 55 170 1;14 82.5 170 1;15 110 170
1;16 110 155 1;17 110 140 1];
elements=[1 1 2 t;2 2 3 t;3 3 4 t;4 4 5 t;5 5 6 t;6 6 7 t;7
7 8 t;8 8 9 t;9 9 10 t;10 10 11 t;11 11 12 t;12 12 13 t;13
13 14 t;14 14 15 t;15 15 16 t;16 16 17 t];
boundary_conditions='S-S';
material_properties=[Ex Eyvxvy G];
lengths=100;
```