Computational modelling and validation of guided wave's dispersion curves in an aluminium flat plate by employing Ansys software

Leonardo Galvis Candelario

Submitted to the School of Mechanical Engineering in partial fulfilment of the requirements for the degree of

Bachelor of Mechanical Engineering

Director
Jabid Quiroga
PhD Civil Engineer

Universidad Industrial de Santander
Facultad Fisico-Mecanica
Escuela de Ingenieria Mecanica
Bucaramanga

2019

Acknowledgements

Throughout the writing of this dissertation, I have received a great deal of support and assistance, mainly from my supervisor, Dr Jabid Quiroga, whose expertise was invaluable in formulating of the research topic and especially in the verification of each of the steps of the methodology.

I would like to thank my colleagues for their support, especially those whom I can call my friends for always keeping my energy up.

And above all, I thank my mother, Iveth, for her love and patience despite all my mistakes and delays. I will always be grateful for your effort, I have everything.

Leonardo Galvis Candelario

CONTENTS

List of Tables

List of Figures

Appendices

Resumen

Título: Modelamiento y validación computacional de las curvas de dispersión de las ondas guiadas en una placa plana de aluminio empleando Ansys. *

Autor: Leonardo Galvis **

Palabras clave: Propagación de ondas guiadas, elementos finitos, curvas de dispersión, formas de modo.

Descripción: Este artículo presenta un enfoque práctico para obtener las curvas de dispersión para una placa mediante modelado numérico utilizando el software comercial ANSYS. El método se basa en el análisis dinámico de la guía de onda para extraer las formas de modo (vectores propios) para cada frecuencia resonante (valores propios) de la ecuación de movimiento no forzado. Cada valor propio proporciona una frecuencia y las dimensiones del modelo (longitud de onda), que son la base de las curvas de dispersión. Luego, se desarrolló un algoritmo en MATLAB para apilar las diferentes soluciones de modo con sus respectivas frecuencias utilizando el Criterio de Garantía Modal (MAC). Finalemente se crea un código para el ploteo de las curvas las cuales son sobre puestas en las curvas a comparar o validar. Los datos obtenidos se comparan con las curvas de dispersión proporcionadas por el software GUIGW el cual obtiene las curvas por medio del método SAFE.

***** Trabajo de grado.

** Facultad de Ingenierías Físico-Mecánicas. Director: Jabid Quiroga Mendez, Doctorado en Ingeniería Civil

Abstract

Title: Computational modelling and validation of guided wave's dispersion curves in an aluminium flat plate by employing Ansys. *

Autor: Leonardo Galvis **

Palabras clave: Guided waves propagation, finite elements, dispersion curves, mode shape.

Description: This paper presents a practical approach to obtain the dispersion curves for a plate via numerical modelling using the commercial software ANSYS. The method is based on the dynamic analysis of the waveguide to extract the mode shapes (eigenvectors) for each resonant frequency (eigenvalues) from the unforced motion equation. Each eigenvalue provides a frequency and the dimensions of the model (wavelength), which are the base of the dispersion curves. Afterwards, an algorithm it's developed in MATLAB to stack the different mode solutions with its respective frequencies using the Modal Assurance Criterion (MAC). Finally, a code is created for the plotting of the curves which are superimposed on the curves to be compared or validated. The obtained data are compared with the dispersion curves provided by the GUIGW software which obtains the curves by using the SAFE method.

***** Bachelor thesis.

** Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingeniería Mecánica Director: Jabid Quiroga Mendez PhD Civil Engineering.

Introduction

Lately, engineering structures require suitable and effective methods for condition monitoring avoiding any damage to the operating personnel and keeping the industrial infrastructure safe. Therefore, Structural Health Monitoring (SHM) becomes an obligation, by guaranteeing a good structural performance, avoiding any type of functioning setback and providing feedback and even a predictive prognosis. Due to this, Guide Waves techniques are getting recognition from the industrial community. The technique is a Non-Destructive Test (NDT) based on the analysis of propagated elastic waves (Lamb waves in the case of plates) that as mentioned by Boutyour & Errkik (2019) have the capacity of propagation over long distances through the element under inspection, given by the resonance between the guided wave frequency and the natural frequency of the plate, which in turn, allows screening bigger areas as the wave is launched in one specific spot or side of the material.

To study the propagation of the guided waves in the material, a large amount of research about this subject is needed, such as a model that roughly represents these behaviours. In their study, Sorohan, Constantin, Gǎvan, & Anghel (2011) point out that the waves travel through the structure intercepting each other, producing stationary interference in the cross-section in the direction of propagation. Thus, it's very convenient to examine the determined interference patterns, modes, which exist in infinite numbers and which only propagate for certain combinations of frequency and wave number, according to the elastodynamic properties of the element under examination.

This project presents a practical way to obtain the guided waves dispersion curves of a plate via a finite element (FE) method. Here, the mass and stiffness matrices are used for a modal analysis using commercial software (ANSYS) where the propagation of time-harmonic disturbances in the plate (isotropic and homogeneous) are obtained by applying periodicity conditions. Additionally, the boundary conditions in the plate are considered stress-free. A computational tool is proposed, which allows determining the dispersion curves (phase velocity, group velocity). A good agreement is achieved between the obtained results of the dynamic analysis by ANSYS (eigenvalues and eigenvectors) organized in dispersion curves and the dispersion curves obtained by the software GUIGW (Bocchini, Marzani, & Viola, 2010). Thus, it can be concluded that the studied method is reliable and will become the base of further studies for being a repeatable code for other materials.

1. Objectives

1.1 General Objective

Develop a computer model in ANSYS for a flat aluminium plate considering the properties of the material and its geometry to generate the dispersion curves via Matlab code: phase velocity and group velocity specific to the specimen studied.

1.2 Specific Objectives

- To develop a computational model in ANSYS of the physical-mechanical behaviour of a guided wave on a flat aluminium plate;
- Obtain the dispersion curves for phase velocity and group velocity as a function of the frequency for a flat aluminium plate of 90 centimetres long and 5 centimetres height (Plain strain);
- Validate the computational model developed by checking the simulated results obtained in the software GUIGW.

2. The Guided Waves in the SHM

A system with integrated structural health management basically consists of a network of sensors equipped to the structure for the acquisition of data, which unitedly with a processor, evaluate the structural health. The ability to determine possible damages or failures does not only increase the reliability of the structure but also guarantees the safety of every operator in certain cases. Achieving such monitoring also reduces downtime and labour costs. Raghavan (2007) indicates that this type of systems can be classified as passive or active, depending on whether they involve the use of actuators, respectively. A comparison of the sensing methods can be seen in Table 1, given by Kessler & Spearing (2003) , where although all of the methods are capable of detecting the same dimension of damage implementing similar size and power sensors, the Lamb wave technique is the only one that can offer full surface coverage for a 1 x 1 m plate. Also, as shown in Fig 1;

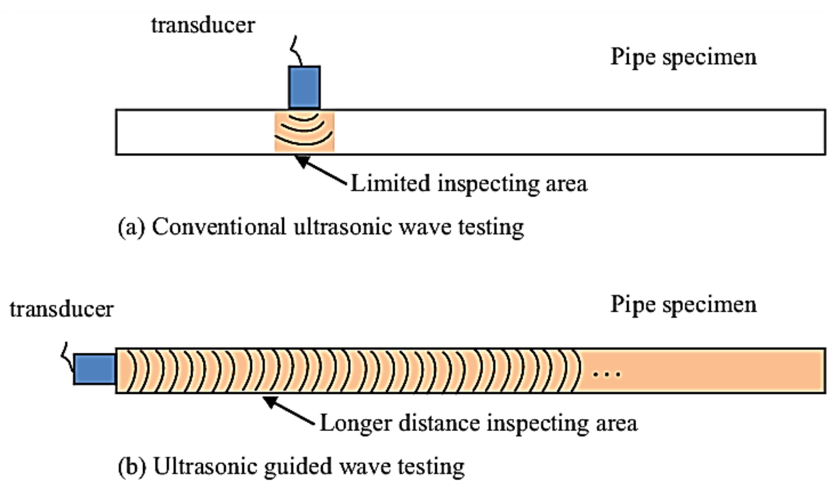


Figure 1. Difference between the conventional ultrasonic wave testing and the guided waves testing. (Pavlakovic, 1998)

Table 1 . Comparison of strengths, limitations and SHM implementation potential for various sensing systems. (Kessler & Spearing, 2003a, p 96)

Method	Strengths	Limitations	SHM Potential
X-radiography	Penetrates surface Small defects with penetrant No data analysis Permanent record of results Simple procedure	Expensive equipment Expensive to implement Human interpretation Can be time-consuming Require access to both sides	Currently none
Strain Gauge	Portable Embeddable Surface mountable Simple procedure Low data rates	Expensive equipment Expensive to implement Data analysis required Localized results	Lightweight Conformable Can be deposited Very low power draw Results for a small area
Optical fibers	Inexpensive equipment Embeddable Quick scan of large area	Expensive to implement Data analysis required High data rates Accuracy in question	Lightweight Large area coverage Must be embedded Requires laser
Ultrasonic	Inexpensive to implement Portable Sensitive to small damage Quick scan of a large area	Very expensive equipment Complex results Specialized software High data rates Require access to both sides	Currently none
Eddy current	Inexpensive to implement Portable Surface mountable Sensitive to small damage	Expensive equipment Very complex results Specialized software Safety hazard Conductive material only	Lightweight Conformable Can be deposited Very high-power draw Results for small area
Acoustic emission	Inexpensive equipment Inexpensive to implement Surface mountable Portable Quick scan of large area Sensitive to small events	Very complex results Very high data rates Specialized software	Lightweight Conformable Can be deposited No power required Results for large area Triangulation capable
Modal analysis	Inexpensive equipment Inexpensive to implement Surface mountable Portable Simple procedure Sensitive to small events	Complex results High data rates Specialized software Results are global	Lightweight Conformable Can be deposited Multi-purpose sensors Low power required Results for small area
Lamb waves	Inexpensive equipment Inexpensive to implement Surface mountable Portable Sensitive to small events Quick scan of linear space	Very complex results Very high data rates Specialized software	Lightweight Conformable Can be deposited Medium power draw Linear scan results Triangulation possible

In conventional UW-based method, the wave propagates in a limited area and attenuates quickly. Therefore, the sensitivity range of the UW-based method is relatively limited. To inspect the condition of a long pipe, the transducer location must be shifted along with the structure. In UGW testing, stress wave with a low frequency propagates along the pipe, therefore, attenuates relatively slowly. (Zhang et al., 2018)

Accordingly, Lamb waves are able to reach complicated or inaccessible locations to be inspected e. g. pipelines inside walls or underground. This fact becomes an advantage since it is possible to select a particular mode shape in order to increase the sensitivity of such mode to specific types of flaws (Rose & Nagy, 2002a), this property reduces the NDT testing time and minimizes the effort. The effectiveness of crack detection using various modes of Lamb wave has been demonstrated through all the references presented in this project.

The rising use of composites in structural components such as fibre-reinforced laminae has led to an increase in the amount of research. Determining the dispersion characteristics of the guided waves is essential for damage detection algorithms. To understand the dynamic response in damage detection applications, it primarily desirable to develop the theory of elastic waves in laminates, including the extraction of dispersion curves. The dispersion curves can be obtained from the root-finding Rayleigh-Lamb equation. The governing equation will be derived under plane strain condition with the assumption of traction-free surfaces. The dispersion equation or in other words, the Rayleigh-Lamb frequency equation can be determined by applying this boundary condition. (Šofer, Ferfecki, & Šofer, 2018).

2.1 Lamb Waves.

Lamb waves are mixed pressure and shear waves, named after Horace Lamb, whom in 1917 presented a closed analytical solution for this type of wave. Lamb waves propagate in solid plates and according to Lamb (1917), they are vibrations of a plate in which deviations occur both in the direction of propagation (longitudinal) and perpendicular (transverse) i.e. the particle motion lies in this two planes. Depending on the excitation frequency, Lamb waves are produced in at least two basic modes, one symmetric and anti-symmetric.

2.1.1 Symmetric Lamb waves

In this case, the upper and lower sides move away from the centre of the plate (or towards them) simultaneously.

2.1.2. Antisymmetric Lamb waves

In these waves, the upper part moves away from the centre while the lower part moves towards the middle at the same position on the plate.

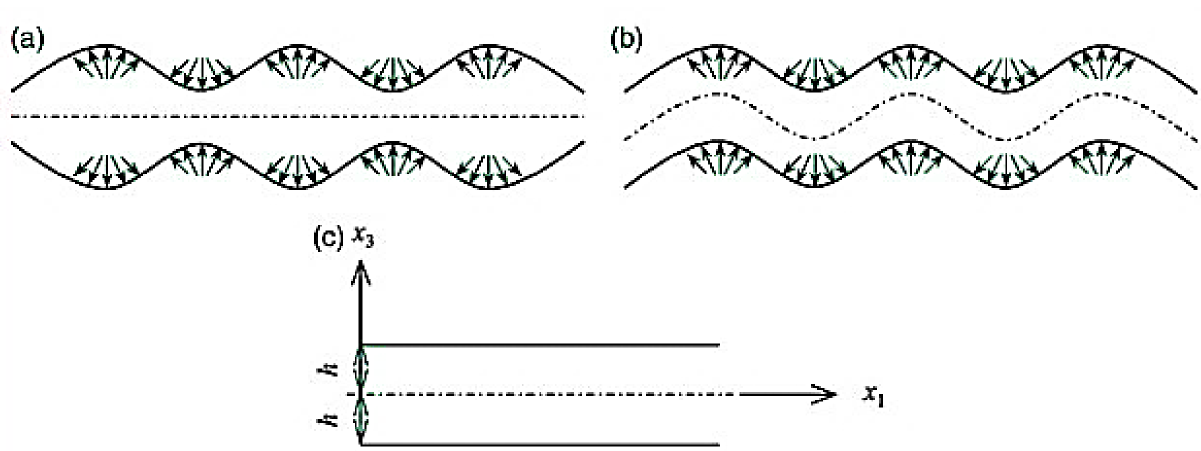


Figure 2. Deformation of S0 and A0 Lamb modes. (a) S0 mode (symmetric mode), (b) A0 mode (antisymmetric mode), (c) Geometry of the free plate problem. (Hosoya, Umino, Kanda, Kajiwara, & Yoshinaga, 2017) .

The propagation speed of the Lamb waves depends on the excitation frequency and thickness of the plate; therefore, they're considered as dispersive, meaning that there's a relation between the wavelength or wavenumber of a wave to its frequency. From this relation, the phase velocity and group velocity of the wave are determined by a geometry-dependent and material-dependent dispersion relation.

Ultrasonic excitation occurs at some point in the plate; as ultrasonic energy from the excitation region encounters the upper and lower bounding surfaces of the plate, mode conversions occur (L wave to T wave, and vice versa). After some travel in the plate, superpositions cause the formation of “wave packets,” or what is commonly called guided wave modes in the plate. Based on the entry angle and frequency used, we can predict how many different modes can be produced in the plate.

(Rose & Nagy, 2002b).

The most popular methods of solution are the displacement potentials and the partial wave techniques (Achenbach 1984 and Auld 1990, respectively) both of them ending in the same exact solution.

$$\frac{\tan(qh)}{\tan(ph)} = \frac{4k^2pq}{(q^2-k^2)^2};\ for\ simmetric\ modes. \qquad (1)$$

$$p^2 = \left(\frac{\omega}{C_L}\right)^2 - k^2 \qquad (2)$$

$$k = \frac{\omega}{C_P} \qquad (3)$$

$$\frac{\tan(qh)}{\tan(ph)} = \frac{(q^2-k^2)^2}{4k^2pq}; for\ anti\ symmetric\ modes. \qquad (4)$$

$$q^2 = \left(\frac{\omega}{C_L}\right)^2 - k^2 \tag{5}$$

$$C_P = \left(\frac{\omega}{2\pi}\right)\lambda \tag{6}$$

Where ω, k and λ define circular frequency, wavenumber and wavelength respectively; C_P phase velocity, and due to the propagation of two types of waves C_L longitudinal velocity and C_T shear velocity. More detailed deductions from these equations are given in the references indicated.

3. Analytical Base

The intention of this project is to present the dispersion curves for an isotropic aluminium plate, it should be noted that this decision is based on the fact that performing analysis for an anisotropic material represents a computational cost. On this Sorohan et al. (2011) point out:

The most significant consequence of elastic anisotropy is the loss of pure wave modes for general propagation directions. This fact also implies that the direction of wave group propagation, i.e. energy flow, does not generally coincide with the wave vector or wave front normal. Moreover, the distinction between wave mode types in general anisotropic plates is somehow artificial, since the equations for classical Lamb and shear horizontal modes and symmetric and anti-symmetric modes are generally coupled.

[...]

The available methods for plotting the dispersion curves are provided by dedicated codes, but for complicated structures like tubes where the number and type of modes are considerably higher than in plates or for noncircular tubes, these codes may prove unsuccessful. (p. 1)

There are four different approaches to determine the dispersion curves: The first one based on the superposition of bulk waves (SPBW), as Lowe (1995) particularises, which includes "Transfer-Matrix" and "Global Matrix" methods. The second one is the Semi-Analytical Finite Element methods (SAFE) or Waveguide Finite Element method (WFE), which resolves the dynamics of the guided wave propagation using a FEM framework. The groundwork for the SAFE method is attributed to Kessler & Spearing (2003) that successfully formulated the SAFE wave equations for elastic layered orthotropic cylinders by means of a one-dimensional cross-section interpolation. As Barski & Pająk (2017) state, it is relatively simple to use in comparison with other methods that were previously mentioned. In the third one, the dispersion curves are obtained in an experimental way requiring great trial effort. Finally, as have been indicated before a numerical approach is proposed by Sorohan et al (2011) in which a dynamic analysis (natural frequencies of the waveguide and its shape modes) provides the wavenumber-frequency relation, which is the foundation of the dispersion curves, highlighting the advantages of the dynamic analysis of FE models in providing a numerical tool able to determine the dispersion curves of complex structures.

3.1 The Finite Element Method

The FEM uses mesh generation techniques and works by dividing a complex problem into small elements. Within each finite element, a polynomial formulation is assumed and assembled into a larger system of equations that models the entire problem. In this case, once the waveguide is discretized by a mesh to compute natural frequencies, the FE environment will not consider applied loadings i.e. a “free in space” situation is considered.

The dynamic analysis to determine the natural frequencies and mode shapes are based on the unforced motion governed by the equation of a set of two coupled second-order differential equations, which is written in the matrix form as:

$$\boldsymbol{M}\ddot{\boldsymbol{u}} + \boldsymbol{C}\dot{\boldsymbol{u}} + \boldsymbol{K}\boldsymbol{u}(t) = \mathbf{f}(t) \tag{7}$$

Which can be considered as generalized Newton's second law. Here $\boldsymbol{M}$ is the mass matrix, $\boldsymbol{C}$ is the damping matrix, $\boldsymbol{K}$ is the stiffness matrix, and then, displacement $\boldsymbol{u}(t)$, velocity $\dot{\boldsymbol{u}}\ (t)$, and acceleration $\ddot{\boldsymbol{u}}(t)$ vectors are defined as follows:

$$\boldsymbol{u}(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} \qquad \dot{\boldsymbol{u}}(t) = \begin{bmatrix} \dot{u}_1(t) \\ \dot{u}_2(t) \end{bmatrix} \qquad \ddot{\boldsymbol{u}}(t) = \begin{bmatrix} \ddot{u}_1(t) \\ \ddot{u}_2(t) \end{bmatrix} \tag{8}$$

$$\mathbf{f}(t) = \begin{bmatrix} f_1(t) \\ f_2(t) \end{bmatrix} \tag{9}$$

Also, the force vector $\mathbf{f}(t)$ is defined by the forces (or the torques) associated with each degree of freedom. A free-vibration ($\ddot{\mathbf{u}} \approx \boldsymbol{u}\,\Lambda$) solution of the harmonic motion is assumed so Equation (7) can be simplified knowing that:

$$\boldsymbol{u}(t) = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} Sin(\omega t + \phi) \tag{10}$$

And differentiating twice respect to time;

$$\ddot{u}(t) = -\omega^2 \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}\ Sin(\omega t + \phi) \tag{11}$$

$$(\boldsymbol{K} - \omega^2 \boldsymbol{M}) \begin{bmatrix} \boldsymbol{A}_1 \\ \boldsymbol{A}_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{12}$$

For a nonzero or a nontrivial solution of A1 and A2,

$$(\boldsymbol{K} - \omega^2 \boldsymbol{M}) \begin{bmatrix} \boldsymbol{A}_1 \\ \boldsymbol{A}_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{13}$$

$$\boldsymbol{K} - \omega^2 \boldsymbol{M} = 0 \tag{14}$$

Ending in the basic equation solved in a typical undamped modal analysis as the classical eigenvalue problem (Equation 14). According to the procedure proposed by Sorohan et al. (2011) a mechanical coupling between the left end and right end cross-sectional areas are required to determine the dynamic behaviour of the waveguide, see (Fig. 3). Thus, the degrees of freedom sequence on boundary **1** is identical to the ones on the boundary **2.**

$$U_1 = U_2 \tag{15}$$

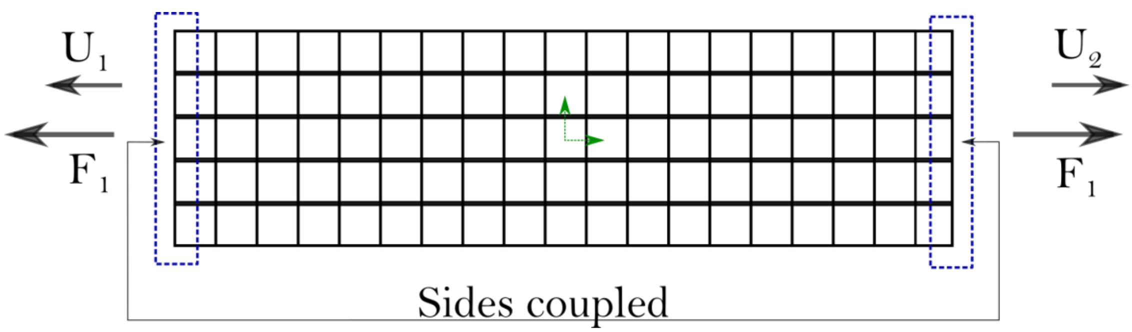


Figure 3. Finite element mesh representation used in plate analysis for Lamb mode computation used in the analysis (plotted in an arbitrary scale).

4. Methodology

The approach presented here starts with the selection of the *Structural* option as a context filter in ANSYS APDL. The dispersion curves for an aluminium plate are desired, hence the PLANE182 element is selected in the *Processor Menu*, which sets up four nodes with two degrees of freedom to each element: translations in the nodal *X* and *Y* as is illustrated in figure 4. (SHARCNET, 2019b). Due to the fact that waves in the *Z* axes are not relevant, the Plain Strain behaviour option must be selected in the *Element Type*.

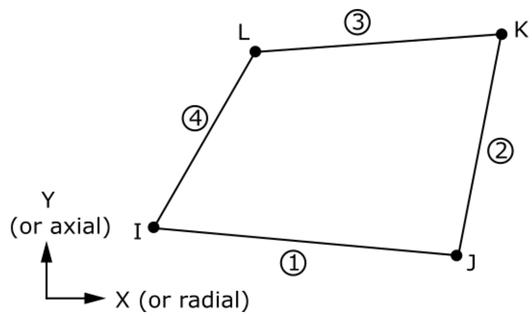


Figure 4. PLANE182 Geometry**.** (SHARCNET, 2019b)

The material model properties for the aluminium are settled. In this case, the aluminium has Young's modulus $E=70$ GPa, Poison's ratio $v=0.33$ and mass density ρ=2700 Kg/m3. The next step will be to create the geometry base with a thickness *h* of 5 mm and a length *L* of 90 mm are defined considering that the wavenumber increment is $\Delta k = \frac{2}{\pi} L$ where *L* must be between 10- and 20-times *h*. In Moser, Jacobs, & Qu (1999), it is recommended to use between 10 - 20 nodes per wavelength, which can be expressed as:

$$l_e = \frac{\lambda_{min}}{10 - 20} \qquad (16)$$

where l_e is the element length and is λ_{min} the shortest wavelength of interest and estimates that if highly accurate numerical results are needed, and a finer meshing might be required. The element size was chosen so that the propagated waves can be accurately described for a frequency range of interest considered (0–1000 kHz).

Subsequently, the *Modal Analysis* is selected. The Block Lanczos algorithm is especially powerful when searching for eigenfrequencies in a given part of the eigenvalue spectrum of a given system, as expressed by (SHARCNET, 2019a). Finally, once the model is submitted to analysis after the computation stage, the results are displayed as a list of natural frequencies (eigenfrequencies) and mode shapes (eigenvectors), this data is saved in an array and will be transferred to the MATLAB Workspace, where the dispersion curves are plotted following a process described next.

5. Post-Processing

Once the Ansys data has been uploaded into MATLAB, the next requirement in order to be able to plot de dispersion curves will be to obtain the wavelength to every mode found in the modal

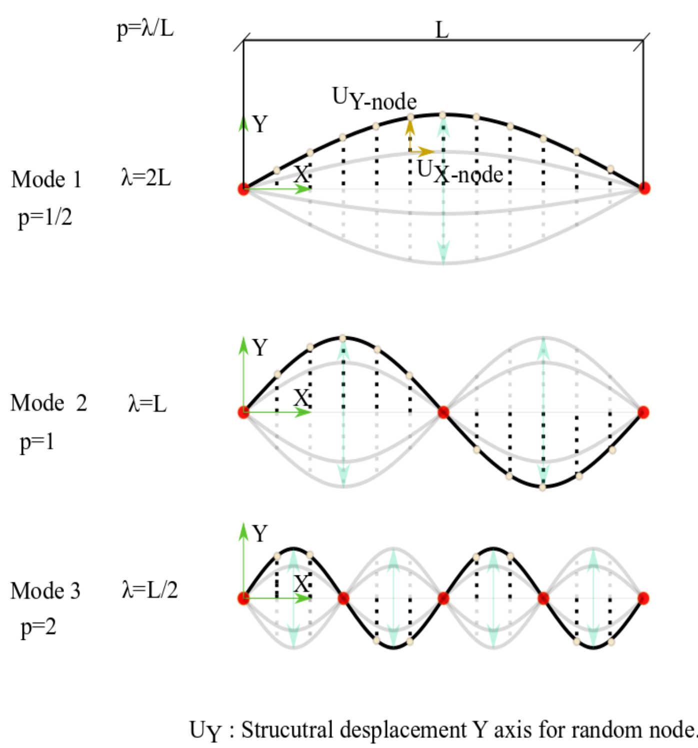


Figure 5. Structural displacements along the wave for a string.

analysis. To solve this situation, all the modes are classified by a mode number, which is an enumeration based on the number of halfwaves in the vibration.

The most common example is the visualization of the vibrating string. Each point of the rope oscillates around its resting position. For each point of the string, its movement can be considered as a dimension of a vector space (linear space); this way the space vector obtained gathers the movements of all the points of the string, in an infinite dimension. From an initial position obtained by the guitarist's finger, the exists of the string follows an equation called a partial differential equation that is linear. The eigenvectors are in this case a vibration that leaves some fixed points, called standing waves or modes. In Figure 5 the vibrating beam with both ends pinned displayed a mode shape of half of a sine wave (one peak on the vibrating string) it would be vibrating in Mode 1. If it had a full sine wave (one peak and one trough) it would be vibrating in Mode 2. To accomplish this numeric sort is necessary to perform an iterative process in which each mode is analysed from the internal displacement of the nodes, which define a specific characteristic to each mode, thereby it is essential to understand the concept of eigenvector and eigenvalue.

5.1 Eigenvectors and Eigenvalues.

An Eigenvector is an algebraic notion that applies to a linear application of space, solving systems of linear equations e.g. dynamic systems. Solving a differential equation:

$$\frac{dy}{dt} = \boldsymbol{A}y \tag{17}$$

Where **A** is a Matrix (**n** by **n**), and thus we have **n** components of y and **n** linear equations, the solution is searched in the form of $y(t) = e^{\lambda t}\, x$, looking for λ to the st solution and multiplied by a vector. Noting that the time dependency is only in the exponential since *x* does not depend on time, the solution is plugged into the differential equations:

$$\lambda e^{\lambda t} = A e^{\lambda t} \tag{18}$$

So $e^{\lambda t}$ is cancelled, since it will never be equal to zero.

$$\mathrm{Ax} = \lambda \mathrm{x} \tag{19}$$

What is sought, is the vector *x* that multiplying by A times *x* gives a number (λ) times *x*. So, in this case, λ (a number), the eigenvalue times the eigenvector *x*. Both are in the same direction (X-direction) and just the length has changed. The value of zero is not desirable for the vector since it contributes nothing to distinguish that the equation is equal to zero. Therefore, *x* must not be zero, and λ can be any number, it can be real, or it can be complex. Hence an eigenvector may be defined as a linear transformation (non-zero vector) that changes by only a scalar factor when that linear transformation is applied to it. The eigenvalue tells whether the special vector *x* is stretched or shrunk or reversed or left unchanged when it is multiplied by A.

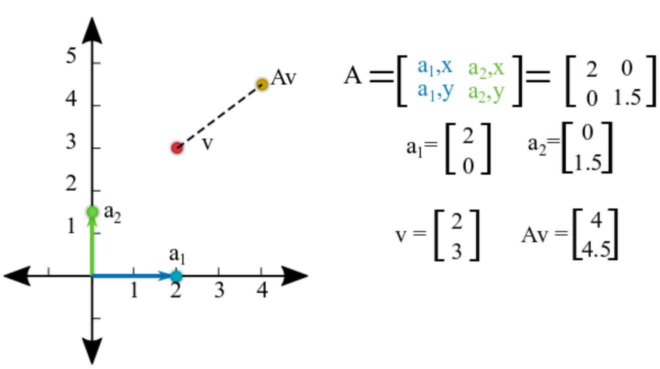


Figure 6. Eigenvector example. Let **v** be a vector (shown as a point) and **A** be a matrix with columns $\mathbf{a_1}$ and $\mathbf{a_2}$ (shown as arrows). If **v** is multiplied by **A**, then **A** sends **v** to a new vector **Av**. Drawing a line through the three points (0,0), v and Av, then **Av** is just v multiplied by a number **λ**; that is, **Av=λv**. Hence **λ** an eigenvalue and **v** an eigenvector.

5.2 Classification of modes by mode number (p).

From this concept, an initial classification by wavelength or "p" number is made for of all vibration modes found in the modal analysis. Basically, it consists in to determine the values of p satisfying $\lambda = L * p$ using the structural displacements given by the software. Knowing that the same eigenvector must be repeated every wavelength, a process of iteration is created. First, the nodes with the maximum displacement in the X and Y axis and their labels from the whole set of nodes are obtained ($U_{Y\text{-}Max}$ & $U_{X\text{-}Max}$; $NodeU_{YMax}$ & $NodeU_{XMax}$) where $NodeU_{YMax}$ and $NodeU_{XMax}$ are the nodes where the maximum displacement $U_{Y\text{-}Max}$ & $U_{X\text{-}Max}$ occurs for each axis respectively. These two scalars $U_{Y\text{-}Max}$ & $U_{X\text{-}Max}$ will be compared with the values of the different nodes by a loop using different values of p validating that the condition $\lambda = L * p$ is accomplished. Once p is stablished the coordinated frequency and lambdas (required to plot dispersion curves) is settled. To locate the node of comparison, the distance from the global coordinate axis (X & Y, coordinated system) are considered. Hence, if the distance of the sum between the position of $U_{Y\text{-}Max}$ & $U_{X\text{-}Max}$ named $NU_{Y\text{-}Max}$ & $NU_{X\text{-}Max}$ (coordinate for X and Y in the active coordinate system) for each axis in the X-axis plus a wavelength does not exceed the length L of the plate, the node can be compared with a node located between the boundaries of the plate, which can be located forward or backwards, named $U_{Y\text{-}Node}$ & $U_{X\text{-}Node}$. In case of $NU_{Y\text{-}Max}$ or $NU_{X\text{-}Max}$ plus a λ/p (where p is an integer in a range between 1 and 20, associated with the mode numbers) exceeds the length of the plate, $U_{Y\text{-}Max}$ & $U_{X\text{-}Max}$ will then be compared with the node located a λ/p backwards. If the absolute difference between the structural displacements for both axes ($U_{Y\text{-}Max}$ & $U_{X\text{-}Max}$) and the displacements for the nodes ($U_{Y\text{-}Node}$ & $U_{X\text{-}Node}$) separated at a distance λ/p is minimal (less than 1%, index by the user) in the two dimensions, the number p is the mode number for that mode shape. Therefore, different *p* values for constant λ are varied from 20 to 2 until they match a mode shape (when the two displacements have match great reliability in the designation of *p* is expected). For the cases where the mode number does not coincide with any of the range (2-20), will be set to *p*=1 in this way the first modes will be determined. However, cut-off frequencies and modes with a very small wavelength ($\lambda/p \geq 20$) will be debugged in an additional process. Part of this process is presented with an illustration Figure 7 and the full iteration process shown in the form of a flowchart in Figure 8. Although it's a useful method, it is not enough to make a discretization of a mode type, also the Ansys software presents a very tedious interface for plotting curves and solves mathematical process. Due to this situation, the use of MATLAB is necessary.

Finally, the results obtained are written into a file that will then be exported to MATLAB where the rest of the post-processing will continue.

Up to this point, the files obtained are:

- NODALINFO_X & NODALINFO_Y is a series of files containing the value of the displacements for every node (hence the name of nodal information) in each of the axes, X and Y respectively.
- FREQLIST is a list of frequency values (100 default).
- P_LIST is a list of the values obtained of P for every frequency i.e. the element (row) 50 in this vector will be the value of *p* for the frequency number 50 (sub-step 50) found.

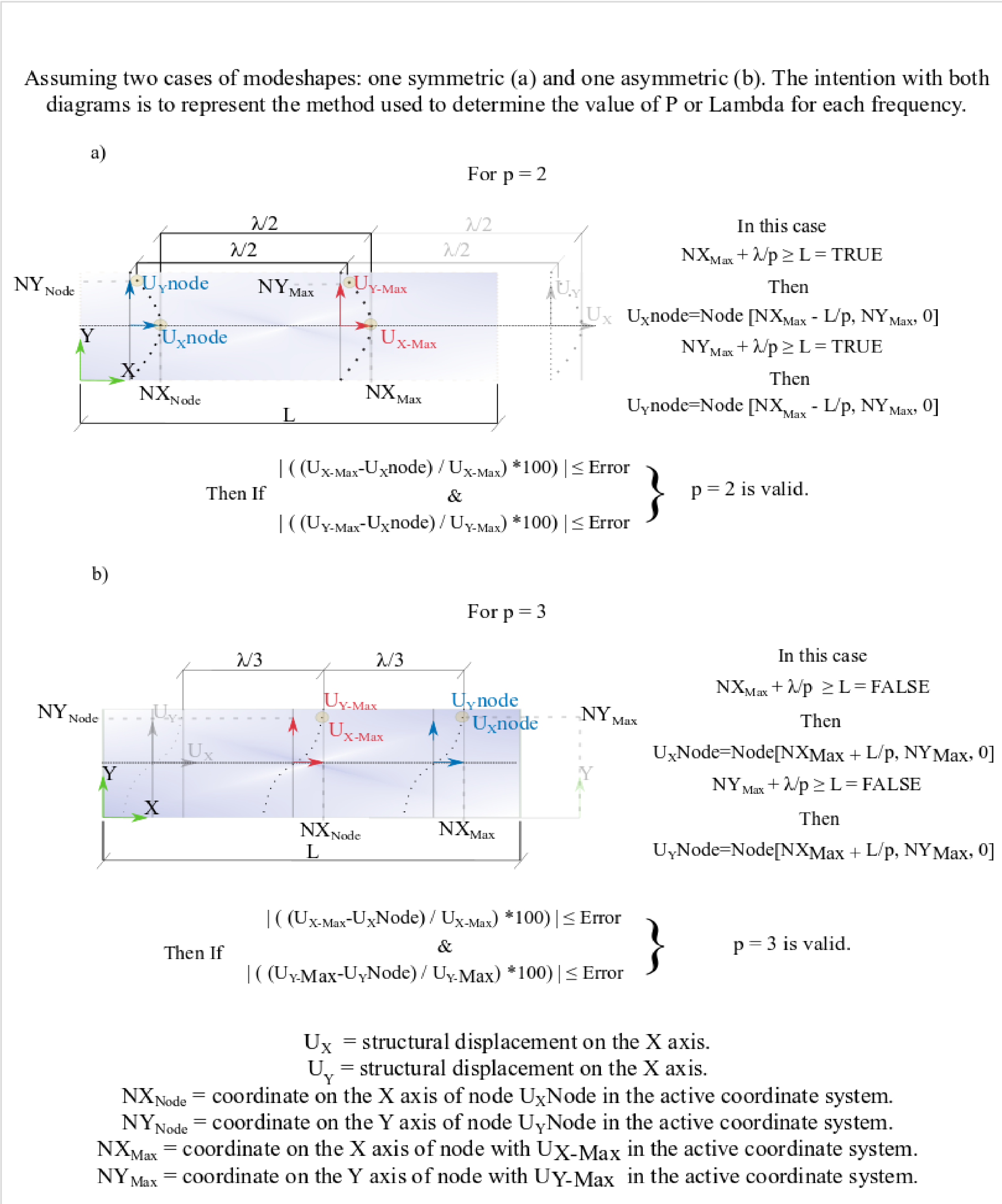


Figure 7. Illustrated example of the iteration process to determine whether the value of *p* in iteration corresponds to the mode shape. Represented without deformation to capture distances more easily.

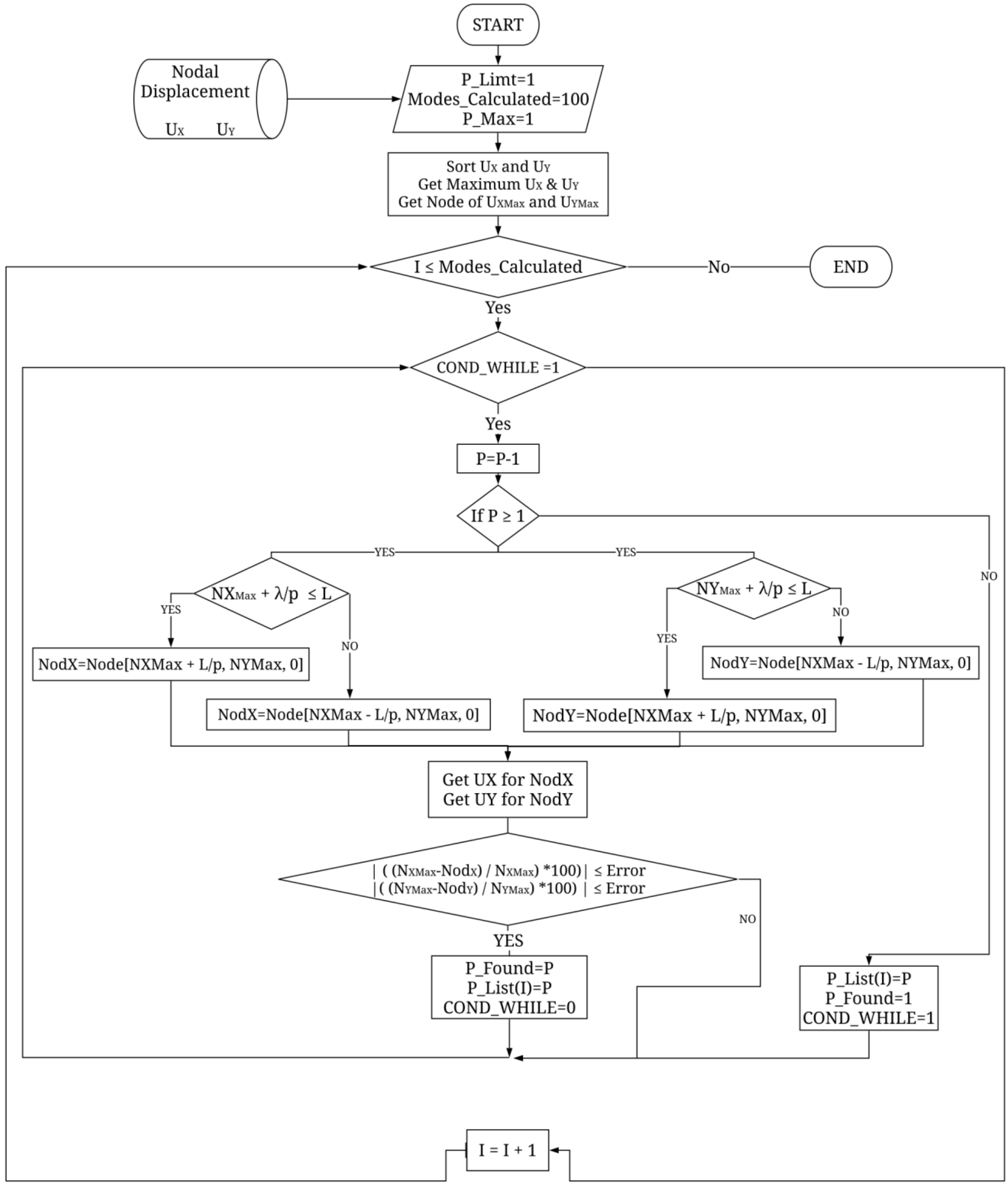


Figure 8. Flowchart of the iteration process classify the modes according to their p-number

5.2 Modal relation through MAC.

The MAC Matrix (Modal Assurance Criterion), also known as a modal correlation matrix, is often used to find a relation between two mode shapes, [Y1] and [Y2]. Where:

$$[Y^1] = \left[\{y_1^1\}..\{y_k^1\}..\{y_{N_1}^1\}\right];\ \text{modal matrix associated to the first series of modes.} \quad (20)$$

$$[Y^2] = \left[\{y_1^2\}..\{y_k^2\}..\{y_{N_2}^2\}\right];\ \text{modal matrix associated to the second series of modes.} \quad (21)$$

The number of modes determined in each series may be different but the size of each vector (number of degrees of freedom) must be the same. The MAC matrix associated with these two series of vectors is defined by:

$$(\mathrm{MAC})_{i,j} = \frac{|^T\{y_i^1\}\{y_j^2\}\ |^2}{\left\|\{y_i^1\}\right\|^2\left\|\{y_j^1\}\right\|^2} \quad (22)$$

This matrix makes possible to quantify the degree of correlation between two vectors by a single number between 0 and 1.

- $(\mathrm{MAC})_{i,j} = 1$; indicates that i^th vector of the first series is colinear to the j^th vector of the second series, i.e. that the own deformities are identical, there is a perfect correlation.
- $(\mathrm{MAC})_{i,j} = 0$; indicates that the two vectors are orthogonal, the proper deformities are very different.

The correlation coefficients lie between these two values and it is necessary to define proximity intervals. The notion of correspondence includes the proximity of the natural frequencies. Two modes can only be "matched" if their natural frequencies are neighbouring.

Consequently, classification of mode types is performed based on the MAC criterion, for that purpose all structural displacements (harmonic displacements) along a vertical line on the plate are selected. The wave structures will become the vector of comparison from every mode shape. The vertical line in the *Y*-axis where all the displacements in the X and Y-axis direction of the cross-section of the plate are acquired. As Figure 9 presents, the value of the node indexed in the code should be one located in the lower edge of the plate, and it will not be fixed to the code so that each user can modify the position of the cutting line where desired.

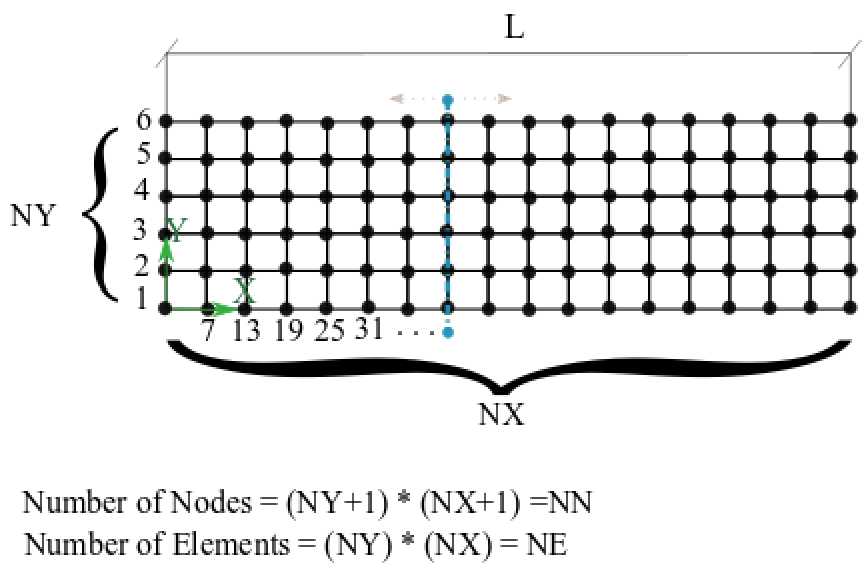


Figure 9. Example of the enumeration process. Any of the nodes on the bottom outer edge can be entered into the code, this number indicates the selection of all nodes on that vertical line (represented by blue) in order to compare mode shapes by MAC, being the wave structure selected the modal matrix.

The wave structure changes for every point on each mode on a dispersion curves as the frequency is swept. The displacement characteristics as *in-plane* or *out of plane*, referred here as the *UX* and *UY* displacements, with *UX* directed along the plate in the wave propagation direction and *UY* normal to the plate. Looking at lamb propagation for a symmetric and antisymmetric modes in Figure 10, the *in-plane* and *out of plane* displacements component is plotted across the thickness of the structure versus frequency. It is easy to see that there is a dominant *in-plane* displacement characteristic across the thickness in both cases.

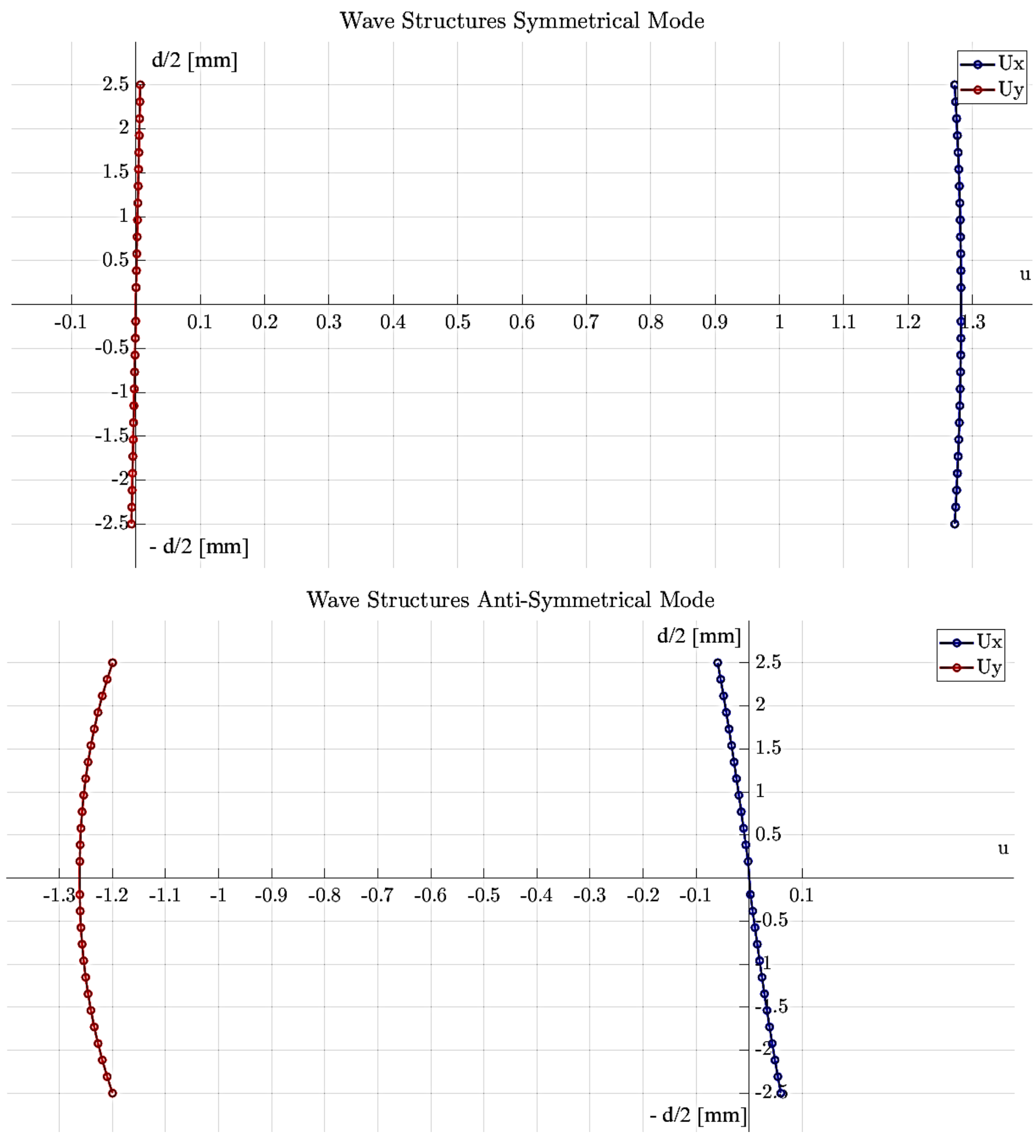


Figure 10. The upper image shows the structural displacements for symmetric mode and the lower one for asymmetric mode. Note that in both cases in the direction of the wave (X axis), the displacements take on greater amplitude. Similarly note that in the case of the symmetric mode the displacements in X reflect a symmetry while those in Y do not. In the opposite case, for the antisymmetric mode, the displacements in Y show symmetry, while the main one in X does not.

The first step in the process will be to input the code (MATLAB), the different values (scalars) necessary for the analysis:

- MOD_CALC: the number of modes found in the modal analysis.
- NEY: number of modes along the Y-axis.
- NEX: number of modes along the X-axis.
- L: the value of the length of the plate.
- CUTTING_LINE_NODE: node number to place the vertical from where the wave structure is obtained.
- TOL_MAC: criteria threshold value desired (tolerance).

Once the cutting line is specified, the selection of displacements within that line starts. Extracting the information from the NODAL_INFO_X and NODAL_INFO_Y matrices for X and Y-axis respectively. The displacement vectors are used to build the MAC matrix, which is the base to compare the different mode shapes and stack them in the different propagation modes such as S0, A0, etc. Although two matrices are created (MAC_X & MAC_Y one for every axis respectively) only the matrix with the displacements in X will be considered, since as shown in Figure 10, the displacements *in-plane* are dominant and are the ones that really generate significant features to compare modes. The MAC_Y is left in the code only with the intention to compare the effectiveness of MAC for each wave structure (MAC_X and MAC_Y).

Two MAC matrices are created from a loop which will repeat the MAC operations between each and every one of the modes (Table 2), since 100 modes were obtained, a matrix of 100 per 100 is obtained. Every row it’s compared to another 100 modes (this number can be modified in both Ansys and MATLAB) including itself. This phenomenon can be seen in the threedimensional bars, Figure 12, where the diagonal line in yellow displays the maximum values of MAC obtained for comparing the same modes (as the bars get closer to a value of one, they will turn into a yellow colour). Also, it can be noticed that for low frequencies (back corner) the number of yellow bars surpasses the number of the ones found near to the high frequencies (front corner). This behaviour was expected since the MAC criterion loses accuracy comparing short wavelengths.

Table 2. MAC Matrix Example Table

MAC_X

	$MODE_i$	$MODE_j$	$MODE_k$	$MODE_l$	$MODE_m$	$MODE_n$
$MODE_i$	$(\text{MAC})_{i,i}$=1	$(\text{MAC})_{j,i} = 0.5$	$(\text{MAC})_{k,i} = 0.99$	$(\text{MAC})_{l,i} = 0{,}4$	$(\text{MAC})_{m,i} = 0{,}97$	$(\text{MAC})_{n,i} = 0{,}33$
$MODE_j$	$(\text{MAC})_{i,j} = 0{,}5$	$(\text{MAC})_{j,j}$=1	$(\text{MAC})_{k,j} = 0{,}3$	$(\text{MAC})_{l,j} = 0.97$	$(\text{MAC})_{m,j} = 0.02$	$(\text{MAC})_{n,j} = 0.05$
$MODE_k$	$(\text{MAC})_{i,k} = 0{,}99$	$(\text{MAC})_{j,k} = 0.33$	$(\text{MAC})_{k,k} = 1$	$(\text{MAC})_{l,k} = 0{,}1$	$(\text{MAC})_{m,k} = 0{,}95$	$(\text{MAC})_{n,k} = 0.01$
$MODE_l$	$(\text{MAC})_{i,l}$ =0,4	$(\text{MAC})_{j,l} = 0.97$	$(\text{MAC})_{k,l} = 0{,}1$	$(\text{MAC})_{l,l} = 1$	$(\text{MAC})_{m,l} = 0.025$	$(\text{MAC})_{n,l} = 0.5$
$MODE_m$	$(\text{MAC})_{i,m} = 0{,}97$	$(\text{MAC})_{j,m} = 0.02$	$(\text{MAC})_{k,m} = 0.95$	$(\text{MAC})_{l,m} = 0.025$	$(\text{MAC})_{m,m} = 1$	$(\text{MAC})_{n,m} = 0.6$
$MODE_n$	$(\text{MAC})_{i,n} = 0{,}33$	$(\text{MAC})_{j,n} = 0.05$	$(\text{MAC})_{k,n} = 0.01$	$(\text{MAC})_{l,n} = 0.5$	$(\text{MAC})_{m,n} = 0{,}6$	$(\text{MAC})_{n,n} = 1$

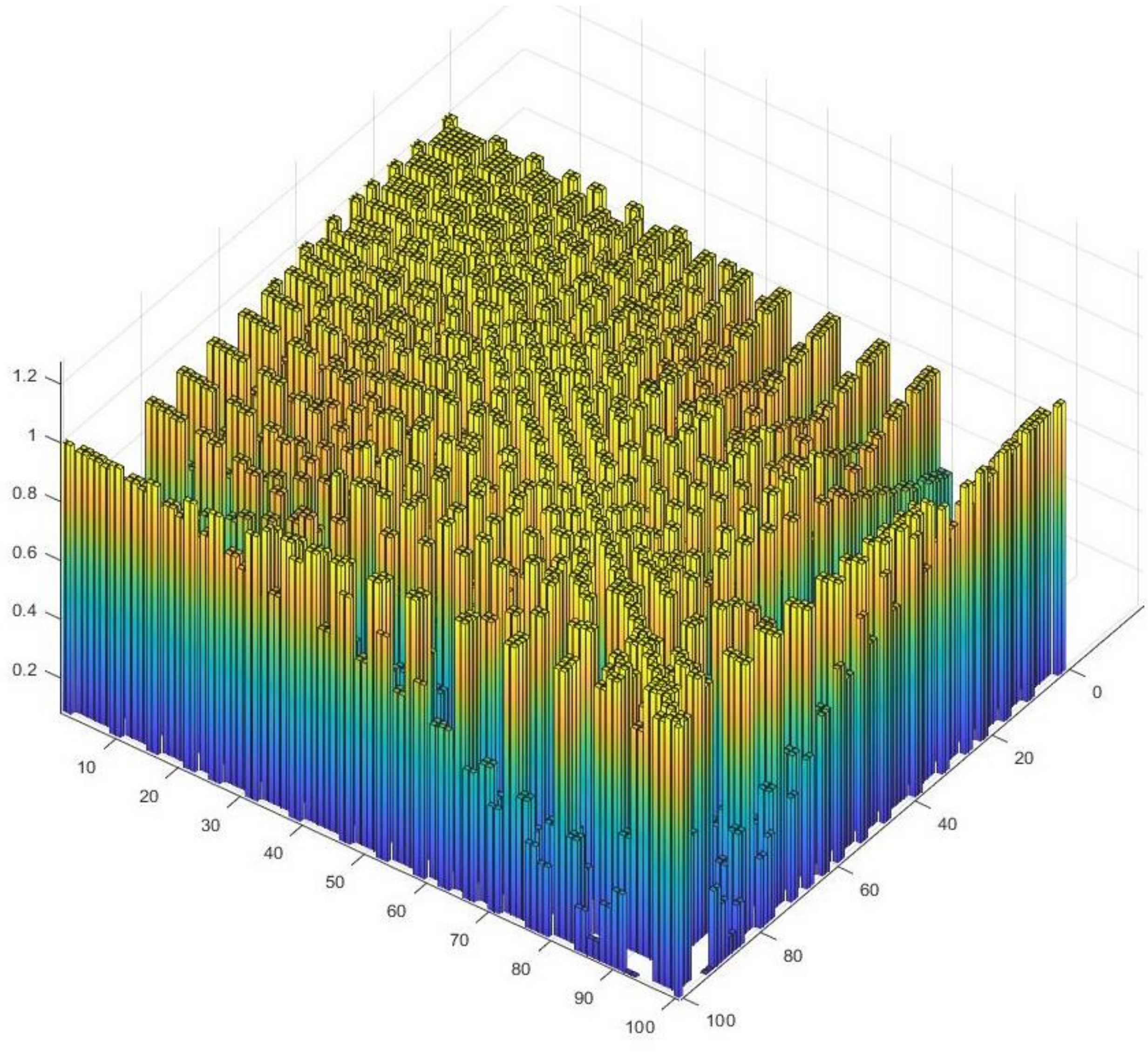


Figure 11. 3D bar plotting of MAC matrix.

The next step will be to organize the MAC matrix by selecting only the sub-steps that surpass the MAC threshold in every column i. e. for column 1 only the rows (sub-steps) that accomplished de MAC criteria will be save in the next matrix named PRE_MODES, name assigned because each column in this matrix will represent a list of frequencies belonging to the same mode type (S0, A0, etc), hence a preview of how the actual results (columns by mode type) are going to look. The order of the elements in each column will correspond proportionally to the respective wave numbers, meaning that, the first element in every column corresponds to the first mode number.

Table 3. PRE_MODES example table. Note that the fact that some columns are repeated confirms the definition of the modes, the number of mode types (in this example 3) and therefore the correct order of the mode numbers.

PRE_MODES

	$MODE_i$	$MODE_j$	$MODE_k$	$MODE_l$	$MODE_m$	$MODE_n$
$MODE_i$	$MODE_i$	$MODE_j$	$MODE_i$	$MODE_j$	$MODE_i$	$MODE_n$
$MODE_j$	$MODE_k$	$MODE_l$	$MODE_k$	$MODE_l$	$MODE_k$	–
$MODE_k$	$MODE_m$	–	$MODE_l$	–	$MODE_l$	–
$MODE_l$	–	–	–	–	–	–
$MODE_m$	–	–	–	–	–	–
$MODE_n$	–	–	–	–	–	–

However, it is possible that different modes (either symmetric or antisymmetric) interfere in the wrong columns, therefore it is recommended to choose a high MAC criterion value (greater than 0,9), where only the frequencies that really belong to the same mode type are merged.

The next step consists to define the number of mode type, this procedure is made by identifying the quantity of initial mode number found i.e. if 3 modes were found with *p*=1, there is a certainty that exists 3 different mode types. For a better understanding, a procedure was added to the code made in Ansys, where the real modes are organized according to their p-number, and then printed in txt format named MATRIX_SUBSTEP (Table 4). Noted that although the modes found with *p*=1 were identified by discarding, the actual frequencies or mode can be debugged by selecting only those who appear in equal pairs. Due to each mode propagates in positive and negative directions, even though some cut-off frequencies may accomplish the *p*=1 in the iteration process, they won't appear in pairs. In other words, having p=1 only fits one wavelength along the plate studied, which corresponds to a shape mode and therefore indicates the main wave mode.

Noting that there are 4 columns it indicates that probably there are at least 4 types of modes, however for the fourth column there is not a mode with *p*=1 and likewise the values of the substeps of the modes are not presented in ascending order in the column, indicating that this type of mode is not fully defined for the length and given frequency . Therefore the mode types will be defined only by the values found for p=1, which will be named from now *main modes* and will be stored in a vector named FIRST_ROW. Based on this argument the list of modes related to these *main modes* is presented.

Table 4. MATRIX_SUBSTEP representation. In the real file, the *p* number column is not presented but the row order indicates the counting of it. Every value in it will represent a specific Sub-step meaning a specif mode found in the modal analysis. Sub-steps *9 30* and *84* will become *main modes.*

p				
1	9	30	84	0
2	5	15	34	78
3	7	19	38	72
4	11	25	46	70
5	13	32	54	74
6	17	40	62	82
7	21	44	76	94
8	23	50	91	0
9	27	56	0	0
10	36	58	0	0
11	42	64	0	0
12	48	66	0	0
13	52	80	0	0
14	60	88	0	0
15	68	96	0	0
16	86	0	0	0
17	98	0	0	0

Table 5. FIRST_ROW example table. Each mode in this vector was classified with a mode number equal to 1 (*p*=1). They are considered as *main modes* since they will relate the rest of the elements for each column (mode type).

FIRST_ROW	$MODE_i$,	$MODE_j$	$MODE_n$	-

Next, the *main modes* will be classified as symmetric or antisymmetric. This differentiation its possible because the first value and the last value for symmetric wave structures should be exactly the same or at least very close, hence the first and last structural displacement of every wave structure (the first and final element of a column) can be selected, and if this condition of equality happens means it is a symmetric type and if not an antisymmetric. This can be seen in Figure 12 where two random frequencies were plotted to compare the wave structures type.

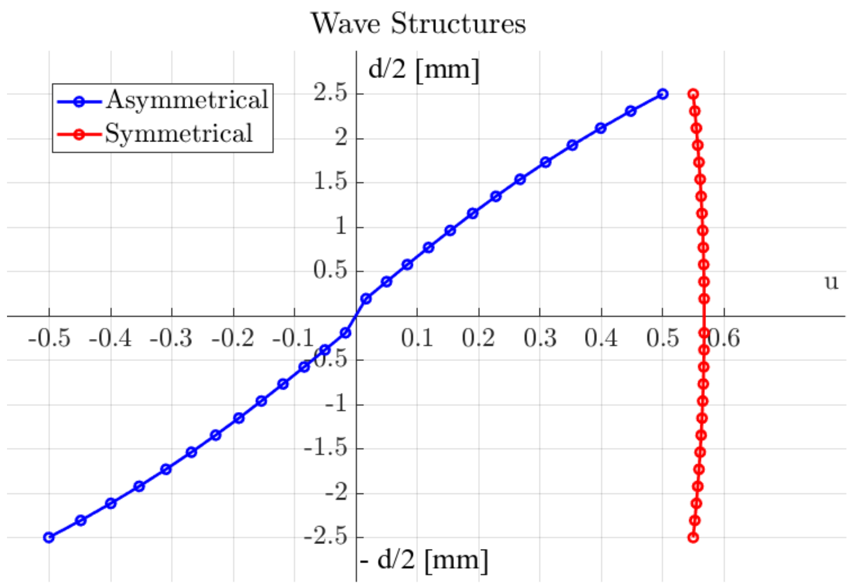


Figure 12. Wave structures plotted from structural displacements on the X-axis direction for two random modes. The extreme values for the symmetrical mode must be equal values.

Following the procedure, as the *main modes* are classified between symmetric or antisymmetric, they will be filling the final vectors (results) through a loop with conditions. Starting by defining whether it is symmetric or not. If the mode is symmetric will fill a vector called S1, if not, one called A1. The code is designed only to fill four types of modes, therefore there are only 2 vectors for each type (A1, A2, S1 and S2). Next, through a conditioned loop each vector of mode type will be filled, all modes present in the designated column with the *main mode* number in the PRE_MODES matrix will be added to the vector, meaning that all the substep highly related to the *main mode* of every vector will be added. To amplify the results the loop will be repeated up to 4 times for each mode type. By taking the last substep to be added to the vector as the new *main mode* to merge a complete list of modes related (highly MAC relation). It is important to note that for the *main modes* (*p=1*) the relation of the value obtained from MAC will lose efficiency as the ratio λ/*p* increases, so taking a sub-step with a higher *p*-value as a reference will merge some modes (or substeps) that were not able to be added in the main mode column but that do correspond to the same type.

For example in the case presented in the tables of this document, the mode *i* will be identified as antisymmetric therefore it will fill the vector A1 with all the modes that belong to its main column in the matrix PRE_MODES, in this case, the number one. While mode *j* will be identified a symmetric mode and therefore will fill a vector called S1 that will be filled with all the modes located in column number two of the matrix PRE_MODES.

Henceforth, the results vectors have been defined and consist of a list of modes of the same type presented for different mode numbers (*p=1, p=2, p=3… etc*) in ascending order. Finally, each mode in the vectors A1, A2, S1 or S2 will be replaced by its corresponding value in Hertz (frequency) and will be debugged by only keeping those frequencies repeated in pairs.

These vectors are the required results to perform the process of plotting the curves, which will be arranged in a general matrix called FINAL_TABLE where the data is presented in an orderly manner for both the frequencies *f* for different mode number and their respective relation *λ/p*, clarifying that as explained by Sorohan et al. (2011) the eigenvectors are orthogonal to the mass and stiffness matrix, so an ascending order of frequencies is expected. Therefore, under the assumption of $kL = \pm 2p\pi$, the frequencies are shown in an order of submultiple of the main length λ_{max}(*p = 0,1,2,3*).

Table 6. FINAL_TABLE representation from the data presented in previous tables.

FINAL_TABLE

$A1$	$A2$	$S1$	λ
f_i	f_n	f_j	$\lambda/1$
f_k	$-$	f_l	$\lambda/2$
f_m	$-$	$-$	$\lambda/3$

5.3 Phase and Group Velocity Calculation.

To produce the database for the dispersion curves generation, some calculations are performed e.g. the wavenumber *k* is obtained from the wavelength which is related to the waveguide length, see Equation 23. On the other hand, the angular velocity is related to the period of oscillation *T* (*f*=(1/*T*) Equation 24, considering that the wave propagates parallel to the largest distance of the plate, a longitudinal wave it's analytically described in time by Equation 25.

$$k = \frac{2\pi}{\lambda} \tag{23}$$

$$\omega = \frac{2\pi}{T} \tag{24}$$

The Phase Velocity can easily be obtained using Equation 26 since the values of frequency and wavelength are available. Mathematically, the Group Velocity is defined as the derivative of the frequency of a partial wave with respect to wavenumber:

$$\psi(x,t) = \sum A_i \cos(k_i x - \omega_i t) \tag{25}$$

$$C_{Ph} = \lambda \cdot \nu = \frac{\omega}{k}; \tag{26}$$

$$C_G = \frac{d\omega}{dk} \tag{27}$$

The wave frequencies (f) are obtained only for a limited series of wavelengths (k), due to this the derivative cannot be accurately obtained. Sorohan et al. (2011) to overcome this limitation, proposes that two further analyses must be performed for new sets of lengths L, with the value of ΔL between (0.5 – 2.5) % of L. In order to achieve an accurate shift in the frequencies with respect to the two sets of wavelengths when a finite difference formulation is used, the ΔL must not be very small. Hence, the central difference theorem was used to estimate each coordinate (Group Velocity, Frequency). The method consists of an approximation of the partial derivatives by algebraic expressions with the values of the dependent variable in a limited number of selected points. As a result of the approximation, the partial differential equation describing the problem is replaced by a finite number of equations algebraic, in terms of the values of the dependent variable at selected points. The value of the selected points becomes the unknowns. The system of algebraic equations must be solved, the central difference theorem can be expressed as:

$$\frac{df(x)}{dx} = \frac{f(x+h) - f(x-h)}{2h} \tag{28}$$

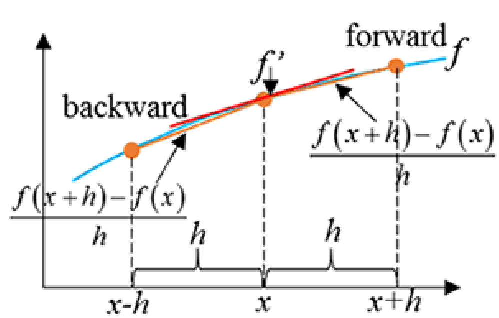


Figure 13. Finite differences as approximations to derivatives. For the first-order derivative, the most common schemes are forward, backward, and central difference schemes. Retrieved from: www.multiphysics.us/FDM.html

Thus, the whole post-processing is repeated twice, one for an added and a minus ΔL as seen in Table 7, obtaining a table data which will be used to plot the dispersion curves.

Table 7. Frequencies (Hz), obtained from an aluminium plate uniformly discretized with *le =2 mm*, and height *h=5 mm*.

	Lamb mode type									
	A0			**A1**			**S0**			**λ**
P	*L=0.081*	*L=0.09*	*L=0.00909*	*L=0.081*	*L=0.09*	*L=0.00909*	*L=0.081*	*L=0.09*	*L=0.00909*	***(mm)***
1	6026,14482	5908,76924	5794,76905	319210,93	319077,986	318948,875	60461,5237	59858,4299	59267,2259	*0,09*
2	22720,596	22301,0413	21892,7904	338216,783	337736,564	337269,397	120447,456	119255,914	118087,569	*0,045*
3	47096,6517	46282,8298	45489,3766	366310,735	365366,362	364445,998	179385,564	177639,75	175926,943	*0,03*
4	76339,7432	75104,8446	73898,878	400481,155	399028,663	397610,901	236482,317	234247,625	232052,633	*0,0225*
5	108477,365	106824,433	105208,261	438487,325	436522,923	434602,978	290553,931	287947,667	285381,805	*0,018*
6	142280,48	140222,632	138208,766	478765,871	476308,562	473903,983	339875,624	337090,039	334336,046	*0,015*
7	177012,427	174564,493	172167,342	520184,256	517269,662	514414,118	382387,293	379672,17	376971,543	*0,01285714*
8	212235,361	209410,871	206643,694	561853,041	558532,119	555274,028	416810,716	414351,73	411894,619	*0,01125*
9	247689,281	244499,62	241373,634	-	590978,188	-	444013,135	441801,345	439594,766	*0,01*
10	283220,456	279674,835	276199,158	-	-	-	466549,427	-	462332,349	*0,009*
11	318739,298	314845,012	311026,891	-	-	-	486957,026	-	482602,652	*0,00818182*
12	354195,288	349958,041	345803,16	-	-	-	506925,255	-	502172,925	*0,0075*
13	389561,828	384986,042	380498,822	-	-	-	527376,823	-	522051,521	*0,00692308*

6. Results

nce the list of frequencies and wavelength are definite, and a list of value for de Phase and Group velocity are obtained, the dispersion curves are plotted joined by the results given by the GUIGW in such a way that it is possible to compare both curves, and the validity of the method proposed for an aluminium plate (E=*70e9*, NU=*0.33*, ρ=*2700* Kg/m^3, Height= *5* mm).

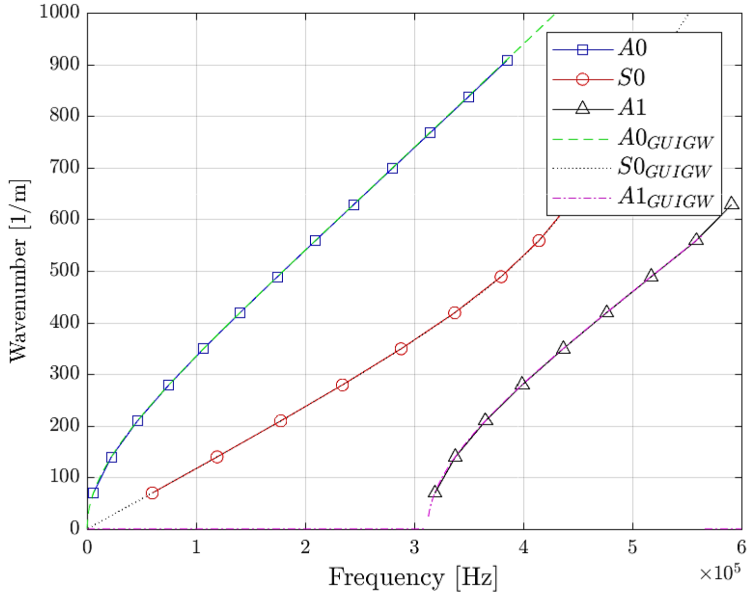


Figure 15. Wavenumber versus frequency for the plate in analysis

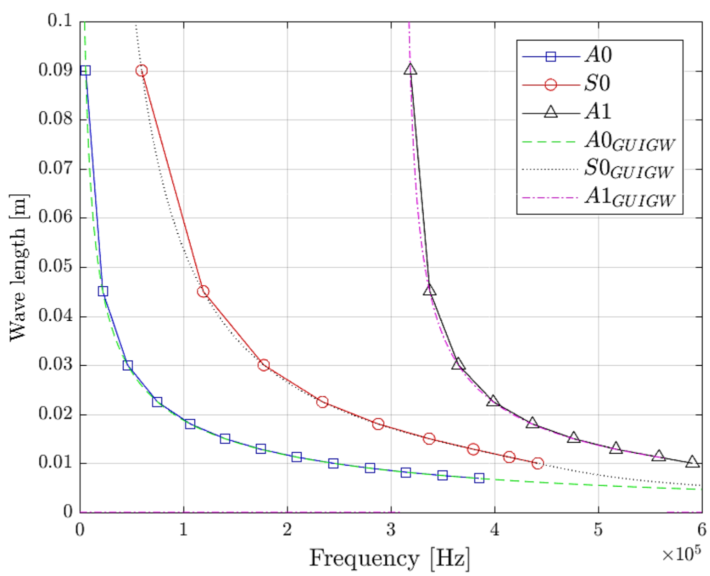


Figure 14. Wavelength versus frequency for the plate in analysis.

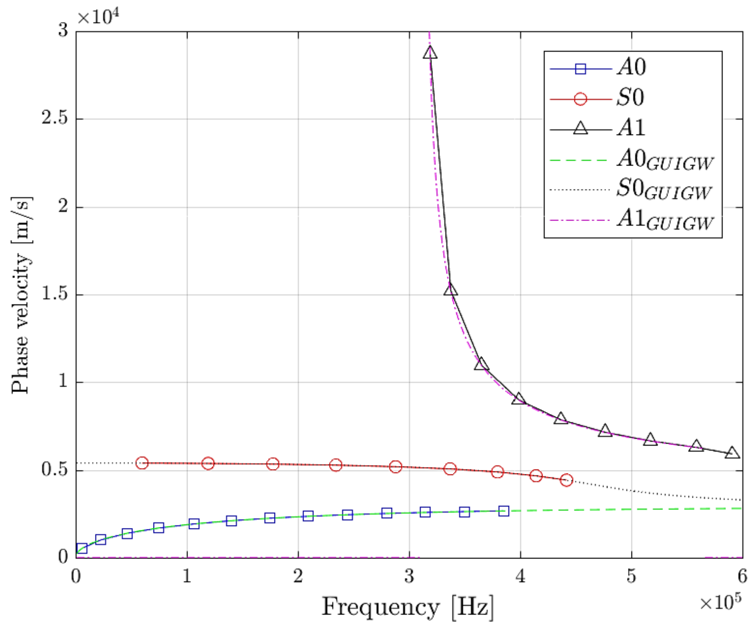


Figure 16. Phase velocity versus frequency for the plate in analysis.

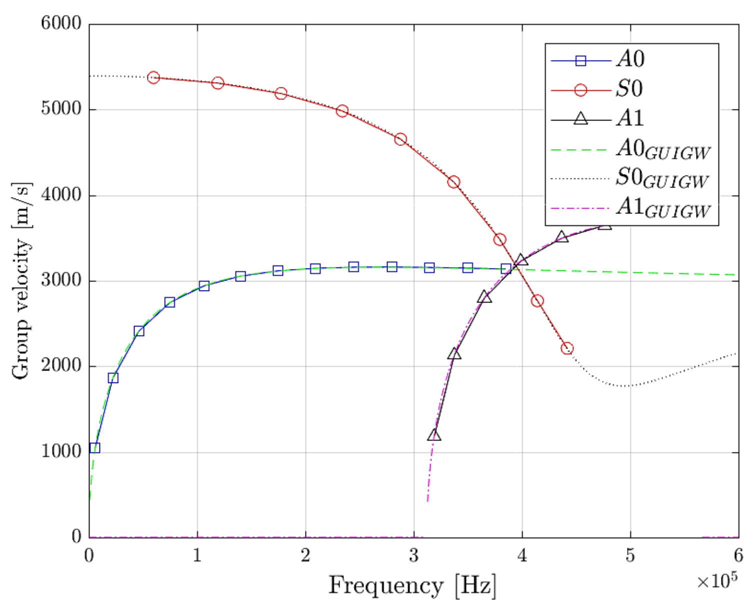
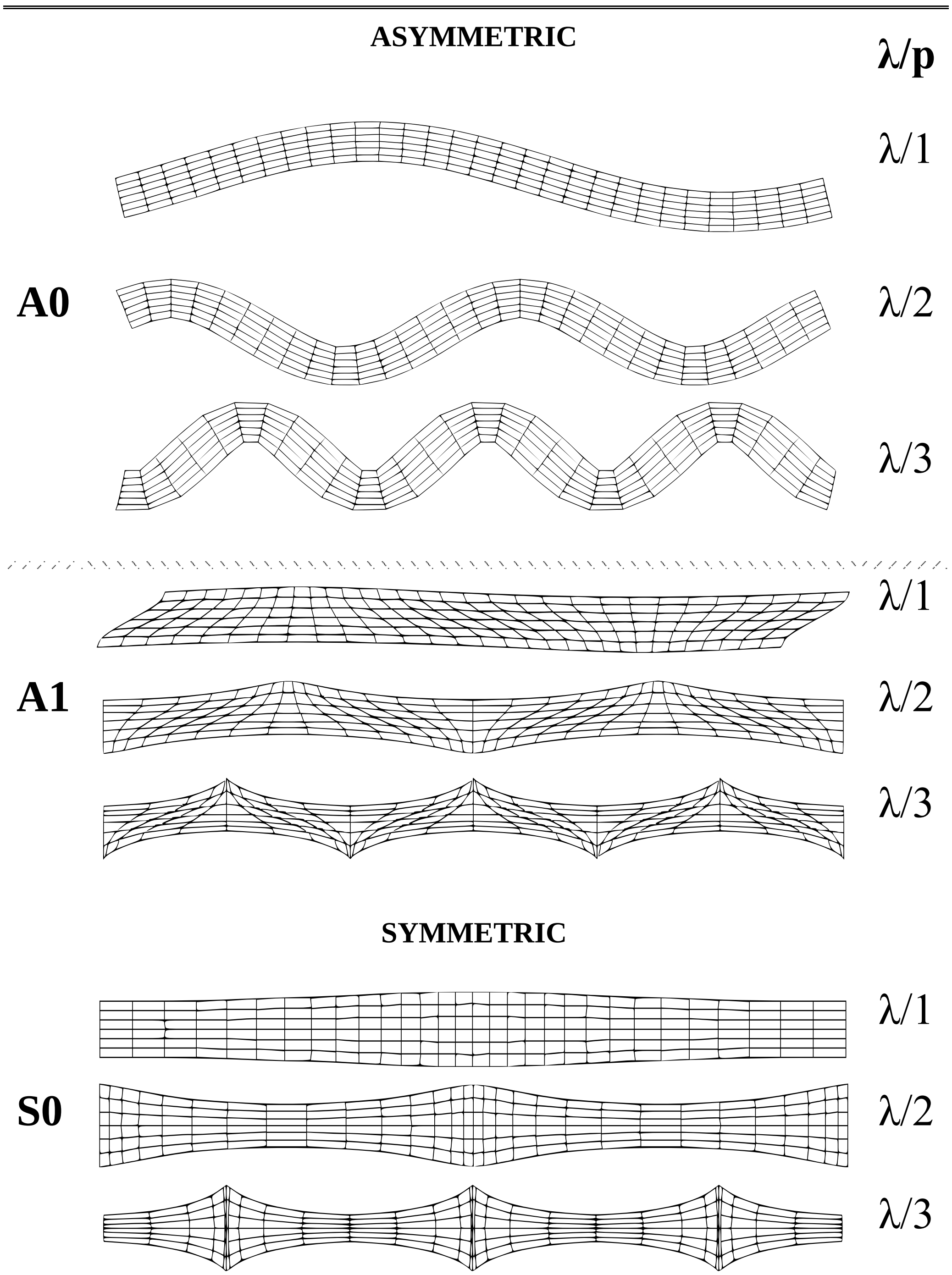


Figure 17. Group velocity versus frequency for the plate in analysis.

As can be seen in the figures the points obtained by modal analysis and the curve obtained analytically are valid according to the GUIGW software.

Table 8. Lamb Modes plotted in Ansys for an isotropic aluminium plate obtained in the modal analysis.

MODE SHAPES



7. Discussions

It is important to demonstrate the importance of choosing a high value for the MAC Matrix threshold, since choosing a value lower than 0.95, for example, will add more points to the curves, however, it is possible that some points do not belong to stacked mode, or that one mode is repeated in two columns. Consequently, the addition of more frequencies (e.g. obtain 200 from the modal analysis) will not be effective if the goal is to define more points in the curve since the code works from the detection of the first mode λ_{Max} (p=1), see Figure 15, 16, 17 and 18. In this regard, the most appropriate way to make proceed is to run the code for different lengths, this way the code will obtain highly related points and therefore it will offer more accuracy to the procedure. However, since the procedure has been divided into two software, repeating these processes is not easy and generates waste of time.

As mentioned above, dispersion curves represent an important element in creating guided wave fault detection systems, the method presented in this project presents a method that is effective within a frequency range, despite the code is designed to obtain at least 4 modes of vibration only 3 were obtained, due to the appearance of the first mode occurs in L away from one entered initially (0.09m), despite the fact that the same frequency range was entered (0-1MHz).

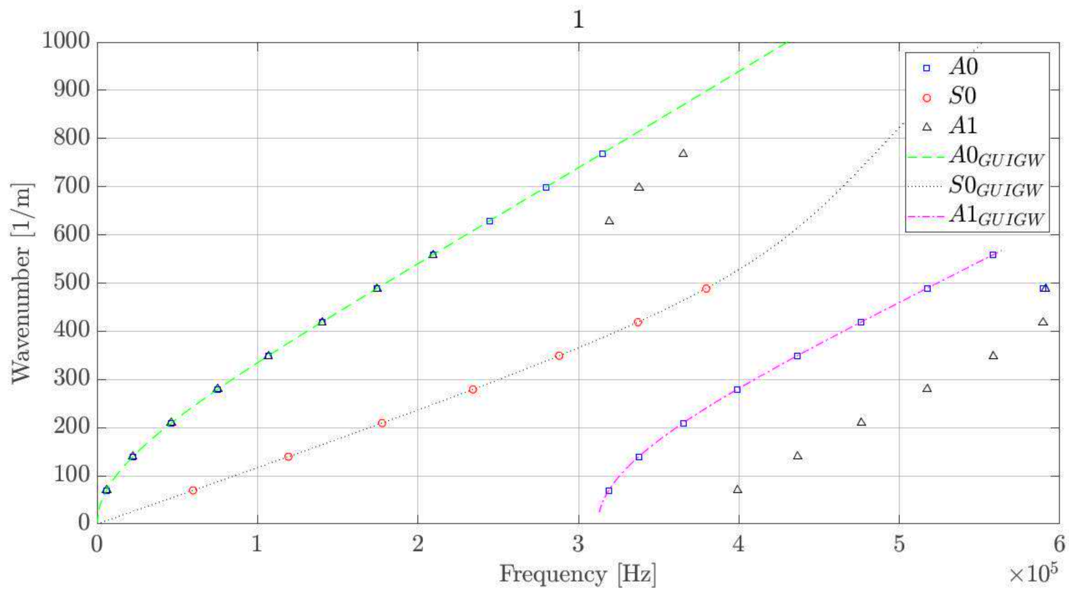


Figure 18. Frequency versus Wavenumber for aluminium plate with low MAC threshold (0,89).

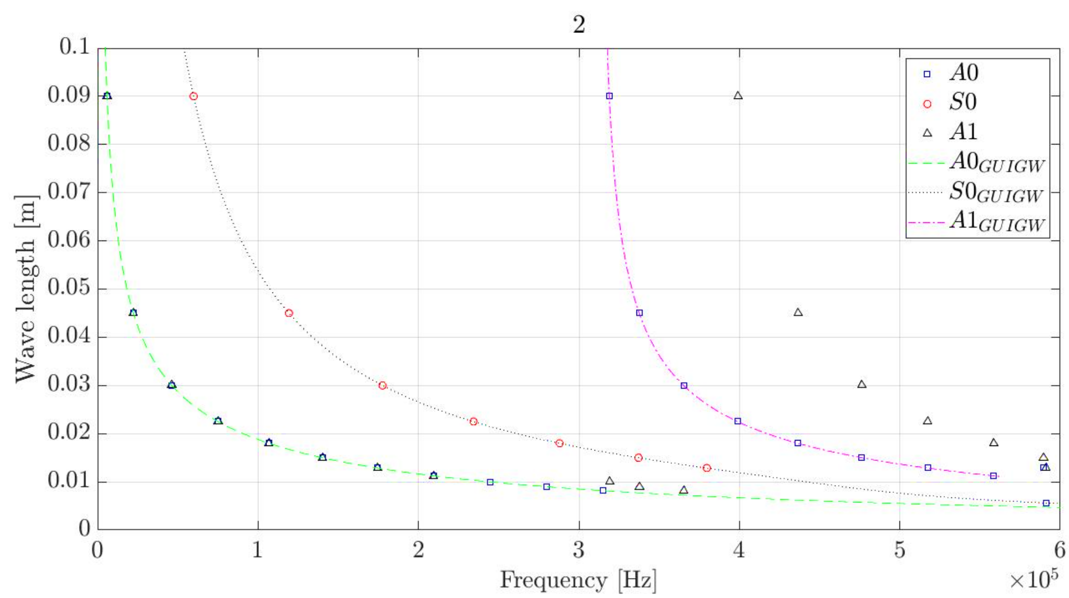


Figure 19. Frequency versus Wave length for aluminium plate with low MAC threshold (0,89).

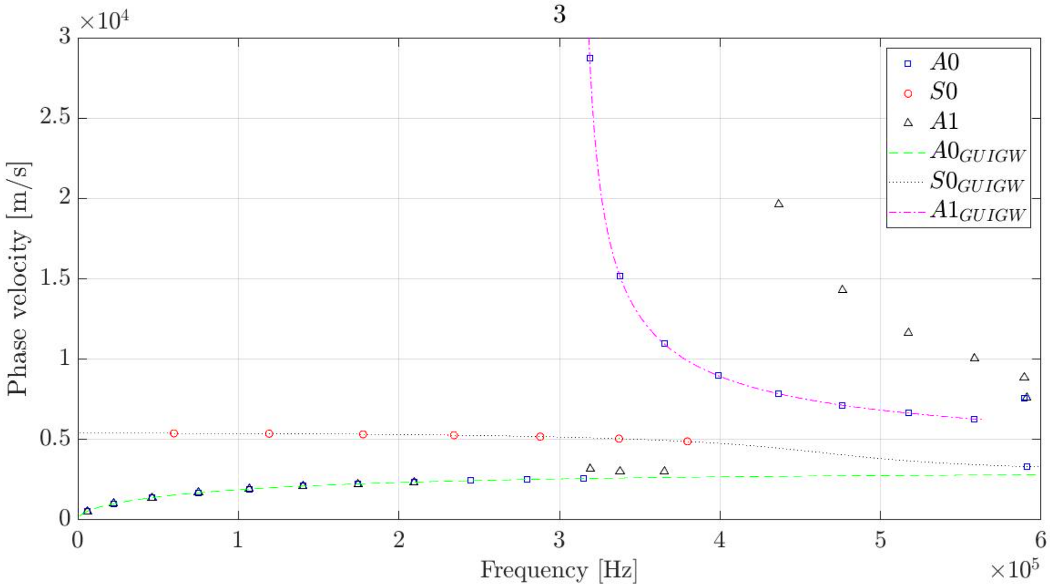


Figure 20. Frequency versus Phase Velocity for an aluminum plate with low MAC threshold (0,89).

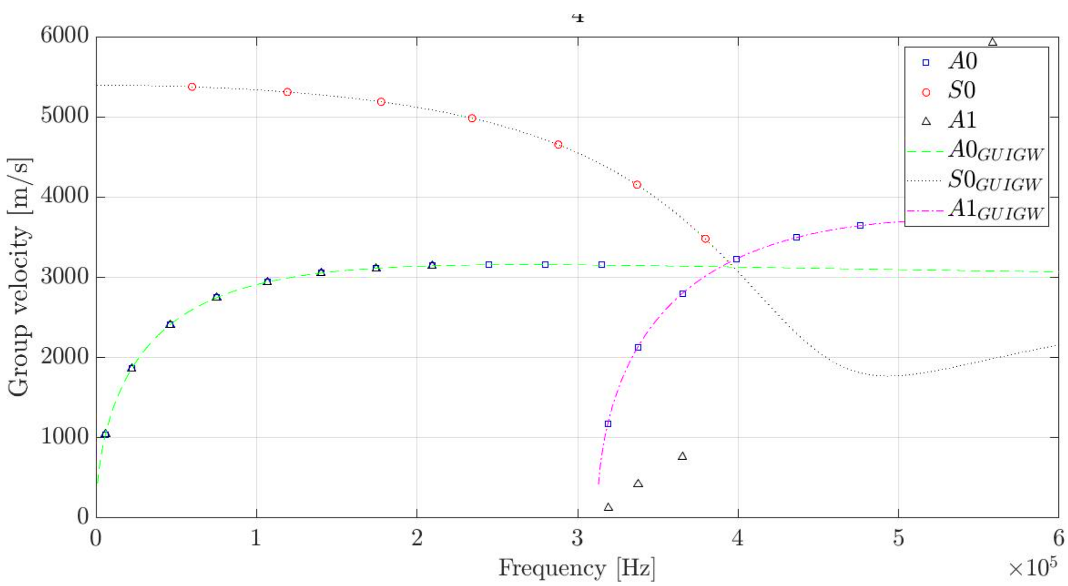


Figure 21. Frequency versus Group Velocty for an aluminum plate with low MAC threshold (0,89).

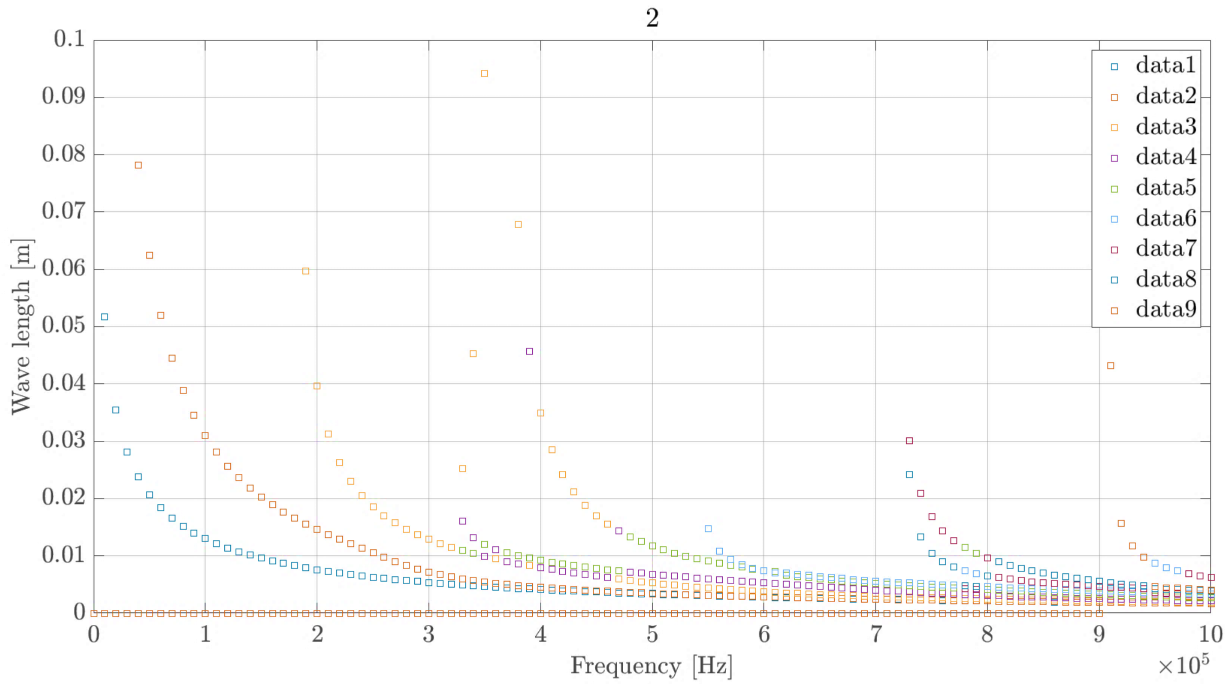


Figure 22. Frequency versus Wavelength for an aluminum plate obtained from GUIGW.

Similarly, it can be noted that for ranges greater than 600KHz the curves tend to combine and eventually ally. The code presented despite being practical presents the disadvantage of having to use two software, which represents another limitation for the code, although it is clear that the correct perform is to repeat for several lengths the same procedure, this method represents a tedious manoeuvre and a lot of time work and attention consumed if it is performed by the Ansys/MATLAB method presented in this document. For this reason, as a future project proposal, the most recommendable option is to create an entire code within MATLAB. This code must perform a modal analysis through finite elements within MATLAB, giving as the main difference with the process developed in Ansys the advantage of being able to automatically variate the lengths of the plate to be studied, obtaining different points for different lengths that will form the base information tables to plot the dispersion curves analytically better.

Finally, it is presented the obtaining of dispersion curves for new material in order to demonstrate the effectiveness of the offered code. The next figures are for an iron plate with, Height *h=0,001 m*, Length *L=0,015*, Density ρ*=7800 Kg/m*^{3}, Young's Module *E= 210 GPa* and Poisson's Ratio *NU=0.33.*

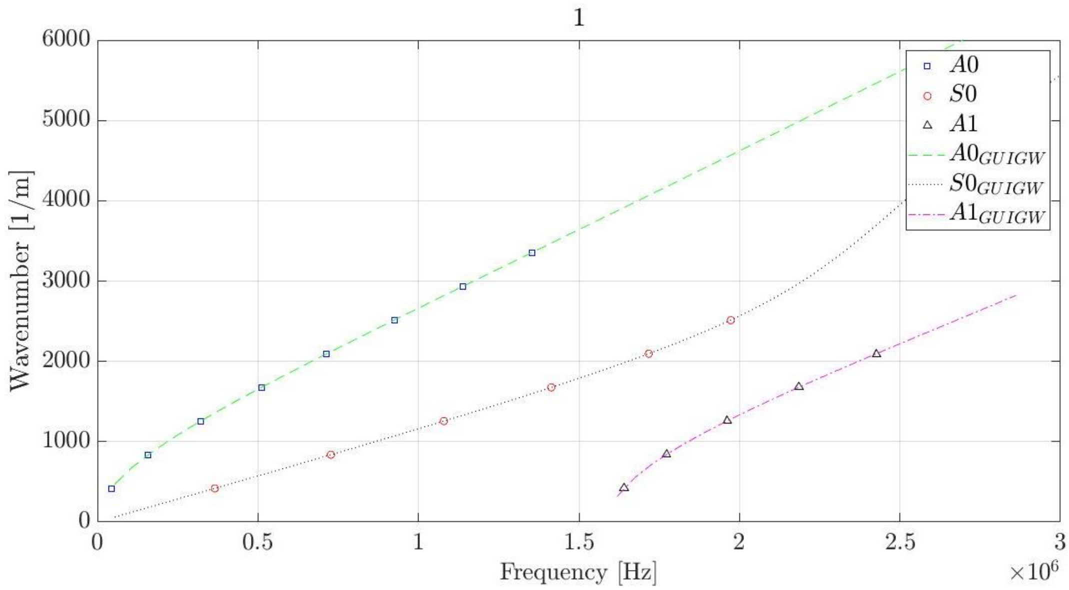


Figure 23. Frequency versus Wavenumber for an iron plate.

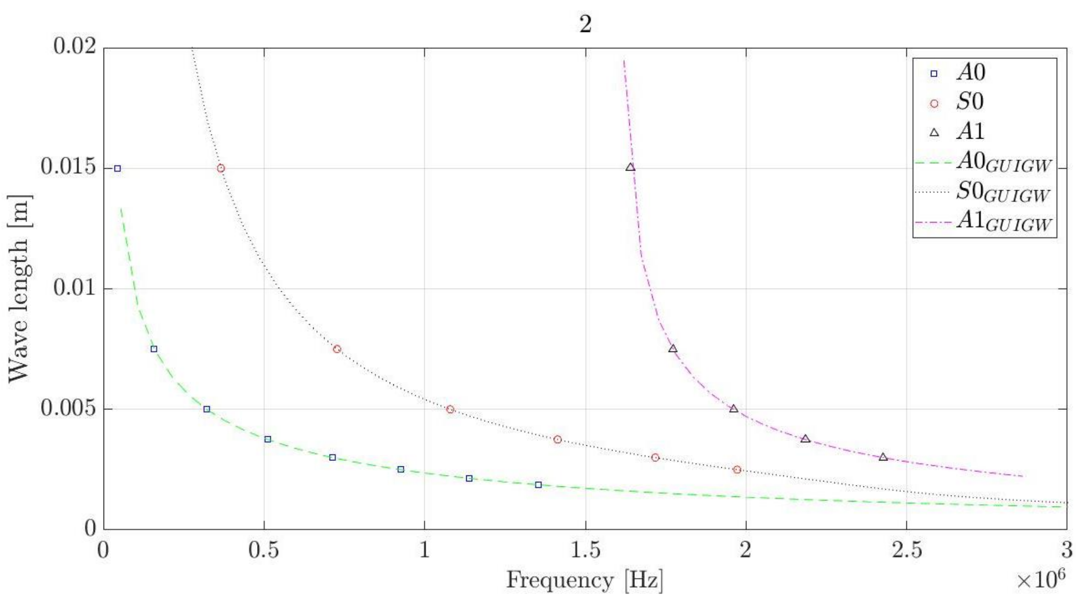


Figure 24. Frequency versus Wave length for an iron plate.

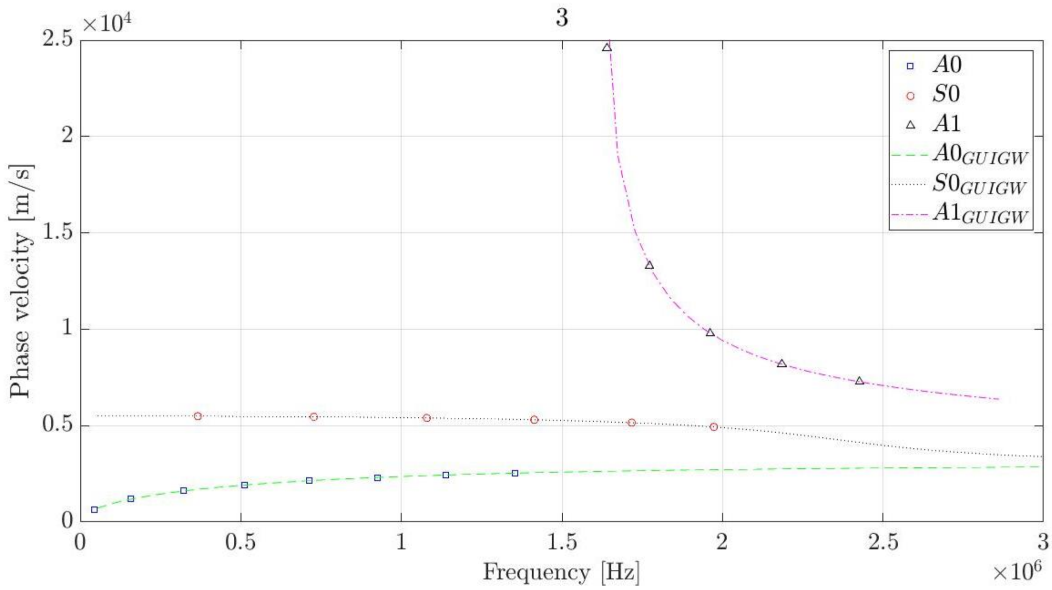


Figure 25. Frequency versus Phase Velocity for an iron plate.

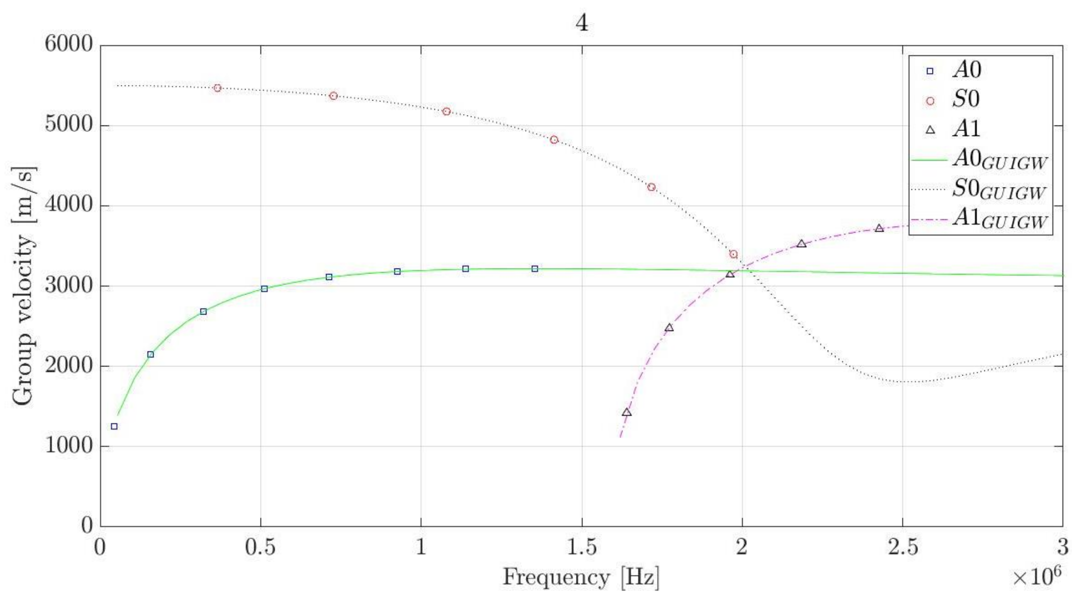


Figure 26. Frequency versus Group Velocity for an iron plate.

8. Conclusions

In this project, the dispersion curves (wavenumber, wavelength, phase velocity and group velocity versus frequency) are obtained for a stress-free, isotropic and homogeneous plate using a modal analysis via a commercial finite element package. The results prove that the studied approach to be assertive in the dispersion curves generation comparing the obtained results with the given by the GUIGW software. To amplifying the resolution and range of the curves more points can be found as the procedure is repeated for different lengths.

As well as it is the advantage of the analytical method since the geometry can be modified to more complex elements that as long as it allows to perform a modal analysis in the software, will allow the dispersion curves to be obtained in a simple way. Therefore, the procedure does not depend on the complexity of the geometry, a situation that occurs when using other types of methods such as SAFE.

References

Barski, M., & Pająk, P. (2017). Determination of dispersion curves for composite materials with the use of stiffness matrix method. *Acta Mechanica et Automatica*, *11*(2), 121–128. https://doi.org/10.1515/ama-2017-0019

Bocchini, P., Marzani, A., & Viola, E. (2010). Graphical User Interface for Guided Acoustic Waves. *Journal of Computing in Civil Engineering*, *25*(3), 202–210. https://doi.org/10.1061/(asce)cp.1943-5487.0000081

Boutyour, E. H., & Errkik, A. (2019). *Lamb waves propagation*. (November 2016).

Hosoya, N., Umino, R., Kanda, A., Kajiwara, I., & Yoshinaga, A. (2017). *Lamb wave generation using nanosecond laser ablation to detect damage*. (December 2016), 3–7. https://doi.org/10.1177/1077546316687904

Kessler, S. S., & Spearing, S. M. (2003a). <title>Design of a piezoelectric-based structural health monitoring system for damage detection in composite materials</title>. *Smart Structures and Materials 2002: Smart Structures and Integrated Systems*, *4701*, 86–96. https://doi.org/10.1117/12.474649

Kessler, S. S., & Spearing, S. M. (2003b). Design of a piezoelectric-based structural health monitoring system for damage detection in composite materials. *Smart Structures and Materials 2002: Smart Structures and Integrated Systems*, *4701*, 86–96. https://doi.org/10.1117/12.474649

Lamb, H. (1917). On Waves in an Elastic Plate. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *93*(648), 114–128. https://doi.org/10.1098/rspa.1917.0008

Load Rayleigh. (1885). On waves propagated along the plane surface of an elastic solid. *Proceedings of the London Mathematical Society*, *17*(11), 4–11. Retrieved from http://library.seg.org/doi/pdf/10.1190/1.9781560801931.ch3i

Lowe, M. J. (1995). Matrix Techniques for Modeling Ultrasonic Waves in Multilayered Media. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, *42*(4), 525–542. https://doi.org/10.1109/58.393096

Moser, F., Jacobs, L. J., & Qu, J. (1999). Modeling elastic wave propagation in waveguides with the finite element method. *NDT and E International*, *32*(4), 225–234. https://doi.org/10.1016/S0963-8695(98)00045-0

Pavlakovic, B. N. (1998). Leaky Guided Ultrasonic Waves in NDT. *Mechanical Engineering*, (October).

Peterie, S. L., Miller, R. D., & Ivanov, J. (2014). Seismology and Its Applications in Kansas. *Kansas Geological Survey*, *37*(July), 1–6.

Raghavan, A. (2007). Guided-wave structural health monitoring. *Dissertation Abstracts International, Volume: 68-11, Section: B, Page: 7465.;Adviser: Carlos E. Cesn*, *975*(2), 91–114.

Rose, J. L. (2003, January). ¨Back to Basics – Dispersion Curves in Guided Wave Testing," Materials Evaluation. *Materials Evaluation, Volume 61, Issue 1*.

Rose, J. L., & Nagy, P. B. (2002a). Ultrasonic Waves in Solid Media . *The Journal of the Acoustical Society of America*, *107*(4), 1807–1808. https://doi.org/10.1121/1.428552

Rose, J. L., & Nagy, P. B. (2002b). Ultrasonic Waves in Solid Media . In *The Journal of the Acoustical Society of America* (Vol. 107). https://doi.org/10.1121/1.428552

SHARCNET. (2019a). *Block Lanczos Method PCG Lanczos Method Supernode (SNODE) Method Subspace Method Unsymmetric Method Damped Method QR Damped Method Method QR Damped Method n n n*. 1–6. Retrieved from https://www.sharcnet.ca/Software/Ansys/17.0/en-us/help/ans_str/Hlp_G_STR3_15.html

SHARCNET. (2019b). Figure 182 . 1 : PLANE182 Geometry Nodal Loading Generalized Plane Strain Element Reference Degrees of Freedom. Retrieved from https://www.sharcnet.ca/Software/Ansys/16.2.3/en-us/help/ans_elem/Hlp_E_PLANE182.html

Šofer, M., Ferfecki, P., & Šofer, P. (2018). Numerical solution of Rayleigh-Lamb frequency equation for real, imaginary and complex wavenumbers. *MATEC Web of Conferences*, *157*, 08011. https://doi.org/10.1051/matecconf/201815708011

Sorohan, Ş., Constantin, N., Găvan, M., & Anghel, V. (2011). Extraction of dispersion curves for waves propagating in free complex waveguides by standard finite element codes. *Ultrasonics*, *51*(4), 503–515. https://doi.org/10.1016/j.ultras.2010.12.003

Zhang, W., Hao, H., Wu, J., Li, J., Ma, H., & Li, C. (2018). Detection of minor damage in structures with guided wave signals and nonlinear oscillator. *Measurement: Journal of the International Measurement Confederation*, *122*(March 2018), 532–544. https://doi.org/10.1016/j.measurement.2017.06.033

Appendix A

MACRO code for modal analysis.

```
!****************************************************************************
****!*          MACRO CODE FOR MODAL ANALYSIS ON A PLATE
!* The eigenvector and eigenvalues obtained from this code should be exported
!* to the workspace in Matlab, for MAC comparison and Dispersion Curves Plotting.
!****************************************************************************
***     !*Working with scalar constants is rather convenient to changes.
                !*INPUTS
            !*Scalar for Material
*SET,E,210E9            !*Young’s modulus E=[GPa]
*SET,niu,0.33           !*Poison's ratio
*SET,ro,7800            !*Density rho=[kg/m^3]
                    !*Geometry
*SET,H,0.001            !*Height[m]
*SET,L,0.015            !*Length[m]
            !*Number of Elements
*SET,NEX,450  !*Number of Elements X axis
*SET,NEY,25   !*Number of Elements Y axis
            !*Frequency Range
*SET,FREQBASE,0       !*Frequency Base
*SET,FREQEND,5E6    !*Frequency range End

        !*Limit on the number of modes to extract, no more than 200
*SET,MOD_LIM_CALC,100
        !*P max, for wavelength calculation
*SET,P_LIM,20   !*Recommended lower than 30
            !*Additional Variables
*SET,NN,(NEX+1)*(NEY+1)  !*Total Number of Nnodes
```

```
*SET,NE,NEX*NEY          !*Total Number of Elements
*SET,ERRP,1              !*Error Percentage for mode estimation.
                         !*Used for P number calculation.
!*************************END OF INPUTS************************
!******************************
!* START OF THE MODAL ANALYSIS
!******************************
/PREP7
!*                       !*ELEMENT TYPE
ET,1,PLANE182,,,2        !*Plane182-Planes Strain Selection (2)
MP,DENS,1,RO             !*Density
MP,EX,1,E                !*!*Young’s modulus E=[GPa]
MP,PRXY,1,NIU            !*Poison's ratio
                      !*NODES check Figure 6 in the extended article to view
N,1,0,0,,,,,
N,NEY+1,0,H,,,,,
FILL,1,NEY+1,
NGEN,NEX+1,NEY+1,ALL,,,L/NEX
                       !*ELEMETS
E,1,NEY+2,NEY+3,2
EGEN,NEY,1,1,,,
EGEN,NEX,NEY+1,ALL
                      !* OFFSET COUPLING check Figure 4 in the extended article to view
CPCYC,ALL,0.00001,0,L,0,0,1
FINISH
/SOLU
!*                    !*TYPE OF ANALYSIS
ANTYPE,2                 !*MODAL
LUMPM,0
PSTRES,0
```

```
MODOPT,LANB,MOD_LIM_CALC,FREQBASE,FREQEND, ,OFF
 !*Block Lanczons selected-Frequency Range
MXPAND,MOD_LIM_CALC, , ,0            !*Number of modes to expand
SOLVE
FINISH
/POST1
!*******************************
!* END OF THE MODAL ANALYSIS
!*******************************

!*******************************
!* PLOTTING NODE NUMBER
!*******************************
!*This with the intention of check the number of the nodes and select the cutting line,
!*to extract the wave structures from the Base data of the modeshape (DOF x & y)
!*check Figure 6 in the extended article to view
!*******************************************************
/PNUM,KP,0
/PNUM,LINE,0
/PNUM,AREA,0
/PNUM,VOLU,0
/PNUM,NODE,1    !*Numbering NODES
/PNUM,TABN,0
/PNUM,SVAL,0
/NUMBER,0
!*
/PNUM,ELEM,0
/REPLOT
!************************************************************
!* THE NODAL INFO (DOF X & Y) and the FREQUENCIES are saved.
```

```
!************************************************************
!* The whole nodal info its saved in NodalInfoX & NodalInfoY
!* The frequencies are also saved in FreqList
*GET,MOD_CALC,ACTIVE,0,SET,NSET,LAST  !*Number of Frequencies found. It has to be equal to the last column for NodalInfo
                                      !* i.e. 80 Frequencies=80 Columns for NodalInfo
allsel,all
*get,NumNd,node,,count            ! Total number of nodes
*dim,NodalInfoX,,NN,MOD_CALC      ! Array size NumberOfNodes x Number of Modes
*do,i,1,MOD_CALC                  ! Loop saving info for every mode
set,1,i
SET,,, ,,, ,i
*VGET,NodalInfoX(1,i),NODE, ,U,X, , ,2
*enddo
allsel,all
*get,NumNd,node,,count                ! Total number of nodes
*dim,NodalInfoY,,NN,MOD_CALC          ! Array size NumberOfNodes x Number of Modes
 *do,i,1,MOD_CALC                     ! Loop saving info for every mode
set,1,i
SET,,, ,,, ,i
*VGET,NodalInfoY(1,i),NODE, ,U,Y, , ,2
*enddo
!***********FREQ**************
*DIM,FREQLIST,ARRAY,MOD_CALC,1,1 , , ,  !* Array with list of frequencies
*DO,I,1,MOD_CALC,1                      !*Defines the beginning of a do-loop.
SET,1,I
*GET,FREQLIST(I),ACTIVE,,SET,FREQ       !*Retrieves a value and stores it as a scalar parameter or as part of an array parameter.
                                        !*Utility Menu>Parameters>Get Scalar Data
*ENDDO
```

```
*CREATE,ansuitmp
*CFOPEN,'FREQLIST',' ',' '
*VWRITE,FREQLIST(1,1)
(30F15.7)
*CFCLOS
*END
/INPUT,ansuitmp
!************************************
!**********NODAL INFO Y *************
!************************************
*CREATE,ansuitmp
*CFOPEN,'NODALINFOY',' ',' '
*VWRITE,NODALINFOY(1,1),NODALINFOY(1,2),NODALINFOY(1,3),NODALINFOY(1,4),NODALINFOY(1,5),NODALINFOY(1,6),NODALINFOY(1,7),NODALINFOY(1,8),NODALINFOY(1,9),NODALINFOY(1,10),NODALINFOY(1,11),NODALINFOY(1,12),NODALINFOY(1,13),NODALINFOY(1,14),NODALINFOY(1,15),NODALINFOY(1,16),NODALINFOY(1,17),NODALINFOY(1,18),NODALINFOY(1,19)
(20F12.7)
*CFCLOS
*END
/INPUT,ansuitmp
*CREATE,ansuitmp
*CFOPEN,'NODALINFOY2',' ',' '
*VWRITE,NODALINFOY(1,20),NODALINFOY(1,21),NODALINFOY(1,22),NODALINFOY(1,23),NODALINFOY(1,24),NODALINFOY(1,25),NODALINFOY(1,26),NODALINFOY(1,27),NODALINFOY(1,28),NODALINFOY(1,29),NODALINFOY(1,30),NODALINFOY(1,31),NODALINFOY(1,32),NODALINFOY(1,33),NODALINFOY(1,34),NODALINFOY(1,35),NODALINFOY(1,36),NODALINFOY(1,37),NODALINFOY(1,38)
(20F12.7)
*CFCLOS
```

```
*END
/INPUT,ansuitmp
*CREATE,ansuitmp
*CFOPEN,'NODALINFOY3',' ',' '
*VWRITE,NODALINFOY(1,39),NODALINFOY(1,40),NODALINFOY(1,41),NODALINFOY(1,42),NODALINFOY(1,43),NODALINFOY(1,44),NODALINFOY(1,45),NODALINFOY(1,46),NODALINFOY(1,47),NODALINFOY(1,48),NODALINFOY(1,49),NODALINFOY(1,50),NODALINFOY(1,51),NODALINFOY(1,52),NODALINFOY(1,53),NODALINFOY(1,54),NODALINFOY(1,55),NODALINFOY(1,56),NODALINFOY(1,57)
(20F12.7)
*CFCLOS
*END
/INPUT,ansuitmp
*CREATE,ansuitmp
*CFOPEN,'NODALINFOY4',' ',' '
*VWRITE,NODALINFOY(1,58),NODALINFOY(1,59),NODALINFOY(1,60),NODALINFOY(1,61),NODALINFOY(1,62),NODALINFOY(1,63),NODALINFOY(1,64),NODALINFOY(1,65),NODALINFOY(1,66),NODALINFOY(1,67),NODALINFOY(1,68),NODALINFOY(1,69),NODALINFOY(1,70),NODALINFOY(1,71),NODALINFOY(1,72),NODALINFOY(1,73),NODALINFOY(1,74),NODALINFOY(1,75),NODALINFOY(1,76)
(20F12.7)
*CFCLOS
*END
/INPUT,ansuitmp
*CREATE,ansuitmp
*CFOPEN,'NODALINFOY5',' ',' '
*VWRITE,NODALINFOY(1,77),NODALINFOY(1,78),NODALINFOY(1,79),NODALINFOY(1,80),NODALINFOY(1,81),NODALINFOY(1,82),NODALINFOY(1,83),NODALINFOY(1,84),NODALINFOY(1,85),NODALINFOY(1,86),NODALINFOY(1,87),NODALINFOY(1,88
```

```
),NODALINFOY(1,89),NODALINFOY(1,90),NODALINFOY(1,91),NODALINFOY(1,92),NODALINFOY(1,93),NODALINFOY(1,94),NODALINFOY(1,95)
    (20F12.7)
    *CFCLOS
    *END
    /INPUT,ansuitmp
    *CREATE,ansuitmp
    *CFOPEN,'NODALINFOY6',' ',' '
    *VWRITE,NODALINFOY(1,96),NODALINFOY(1,97),NODALINFOY(1,98),NODALINFOY(1,99),NODALINFOY(1,100)
    (20F12.7)
    *CFCLOS
    *END
    /INPUT,ansuitmp


    !***********************
    !************ X *********
    !***********************
    *CREATE,ansuitmp
    *CFOPEN,'NODALINFOX',' ',' '
    *VWRITE,NODALINFOX(1,1),NODALINFOX(1,2),NODALINFOX(1,3),NODALINFOX(1,4),NODALINFOX(1,5),NODALINFOX(1,6),NODALINFOX(1,7),NODALINFOX(1,8),NODALINFOX(1,9),NODALINFOX(1,10),NODALINFOX(1,11),NODALINFOX(1,12),NODALINFOX(1,13),NODALINFOX(1,14),NODALINFOX(1,15),NODALINFOX(1,16),NODALINFOX(1,17),NODALINFOX(1,18),NODALINFOX(1,19)
    (20F12.7)
    *CFCLOS
    *END
    /INPUT,ansuitmp
```

```
*CREATE,ansuitmp
*CFOPEN,'NODALINFOX2',' ',' '
*VWRITE,NODALINFOX(1,20),NODALINFOX(1,21),NODALINFOX(1,22),NODALINF
OX(1,23),NODALINFOX(1,24),NODALINFOX(1,25),NODALINFOX(1,26),NODALINFOX(
1,27),NODALINFOX(1,28),NODALINFOX(1,29),NODALINFOX(1,30),NODALINFOX(1,31
),NODALINFOX(1,32),NODALINFOX(1,33),NODALINFOX(1,34),NODALINFOX(1,35),N
ODALINFOX(1,36),NODALINFOX(1,37),NODALINFOX(1,38)
(20F12.7)
*CFCLOS
*END
/INPUT,ansuitmp


*CREATE,ansuitmp
*CFOPEN,'NODALINFOX3',' ',' '
*VWRITE,NODALINFOX(1,39),NODALINFOX(1,40),NODALINFOX(1,41),NODALINF
OX(1,42),NODALINFOX(1,43),NODALINFOX(1,44),NODALINFOX(1,45),NODALINFOX(
1,46),NODALINFOX(1,47),NODALINFOX(1,48),NODALINFOX(1,49),NODALINFOX(1,50
),NODALINFOX(1,51),NODALINFOX(1,52),NODALINFOX(1,53),NODALINFOX(1,54),N
ODALINFOX(1,55),NODALINFOX(1,56),NODALINFOX(1,57)
(20F12.7)
*CFCLOS
*END
/INPUT,ansuitmp
*CREATE,ansuitmp
*CFOPEN,'NODALINFOX4',' ',' '
*VWRITE,NODALINFOX(1,58),NODALINFOX(1,59),NODALINFOX(1,60),NODALINF
OX(1,61),NODALINFOX(1,62),NODALINFOX(1,63),NODALINFOX(1,64),NODALINFOX(
1,65),NODALINFOX(1,66),NODALINFOX(1,67),NODALINFOX(1,68),NODALINFOX(1,69
```

```
),NODALINFOX(1,70),NODALINFOX(1,71),NODALINFOX(1,72),NODALINFOX(1,73),N
ODALINFOX(1,74),NODALINFOX(1,75),NODALINFOX(1,76)
    (20F12.7)
    *CFCLOS
    *END
    /INPUT,ansuitmp
    *CREATE,ansuitmp
    *CFOPEN,'NODALINFOX5',' ',' '
    *VWRITE,NODALINFOX(1,77),NODALINFOX(1,78),NODALINFOX(1,79),NODALINF
OX(1,80),NODALINFOX(1,81),NODALINFOX(1,82),NODALINFOX(1,83),NODALINFOX(
1,84),NODALINFOX(1,85),NODALINFOX(1,86),NODALINFOX(1,87),NODALINFOX(1,88
),NODALINFOX(1,89),NODALINFOX(1,90),NODALINFOX(1,91),NODALINFOX(1,92),N
ODALINFOX(1,93),NODALINFOX(1,94),NODALINFOX(1,95)
    (20F12.7)
    *CFCLOS
    *END
    /INPUT,ansuitmp
    *CREATE,ansuitmp
    *CFOPEN,'NODALINFOX6',' ',' '
    *VWRITE,NODALINFOX(1,96),NODALINFOX(1,97),NODALINFOX(1,98),NODALINF
OX(1,99),NODALINFOX(1,100)
    (20F12.7)
    *CFCLOS
    *END
    /INPUT,ansuitmp
    !*************************************************************
    !***********************LIST OF P [Wavelength]*****************
    !*************************************************************
    *DIM,P_LIST,ARRAY,MOD_CALC,1,1 , , ,      !Array with list of P number for requency list
    P_MAX=1
```

```
NR_MOD_IDENTIF=0
*DO,I,1,MOD_CALC,1    !* Loop for every MODESHAPE found.
SET,1,I    !*Activates to READ the info for every Sub-step (frequencies).
!* The maximum structural displacement is obtained [UxMax & UyMax]
NSORT,U,X                !*Use ascending node-number order X, Y, or Z structural displacement
*GET,MAX_UX,SORT,,MAX      !* Maximum value of last sorted item (NSORT or ESORT
command).
*GET,NOD_MAX_UX,SORT,,IMAX !*Node/Element number where maximum value occurs
NSORT,U,Y
*GET,MAX_UY,SORT,,MAX
*GET,NOD_MAX_UY,SORT,,IMAX
P=P_LIM
COND_WHILE=1
*DOWHILE,COND_WHILE
P=P-1
*IF,P,GT,1,THEN
!* REQUEST A NOD AT X AT L / P DISTANCE TO BE VERIFIED (Check Figure 7)
*IF,NX(NOD_MAX_UX)+L/P,LE,L,THEN    !* NX    X-coordinate of node N in the active
coordinate system.
NOD_UX=NODE(NX(NOD_MAX_UX)+L/P,NY(NOD_MAX_UX),0)
*ELSE                              !* NY    Y-coordinate of node N in the active coordinate system.
NOD_UX=NODE(NX(NOD_MAX_UX)-L/P,NY(NOD_MAX_UX),0)
*ENDIF
!* REQUEST A NOD AT X AT L / P DISTANCE TO BE VERIFIED FOR UY
*IF,NX(NOD_MAX_UY)+L/P,LE,L,THEN
NOD_UY=NODE(NX(NOD_MAX_UY)+L/P,NY(NOD_MAX_UY),0)
*ELSE
NOD_UY=NODE(NX(NOD_MAX_UY)-L/P,NY(NOD_MAX_UY),0)
*ENDIF
```

```
!* The value of the structural displacements for the found node are obtained.
 *GET,UX_NOD,UX,NOD_UX       !* Alternative get functions: UX(N), UY(N), UZ(N).
*GET,UY_NOD,UY,NOD_UY
!*The difference between the structural displacements is verified (must be minimum)
*IF,ABS((MAX_UX-UX_NOD)/MAX_UX*100),LE,ERRP,AND,ABS((MAX_UY-
UY_NOD)/MAX_UY*100),LE,ERRP,THEN
P_FOUND=P
P_LIST(I)=P
COND_WHILE=0
*ENDIF
*IF,P_FOUND,GT,P_MAX,THEN        !* TO SET THE MAXIMUM NUMBER FOR P
COMPUTED
P_MAX=P_FOUND
*ENDIF
*ELSE  !* P=1
P_LIST(I)=1
P_FOUND=1
COND_WHILE=0
*ENDIF
*ENDDO   !* END WHILE
*ENDDO   !* END FOR I
!*******************
!*the vectors are exported
!*******************
*CREATE,ansuitmp
*CFOPEN,'P_LIST',' ',' '
*VWRITE,P_LIST(1,1), , , , , , , , ,
(20F12.2)
*CFCLOS
*END
```

```
/INPUT,ansuitmp
*DIM,LAMBDA,ARRAY,P_MAX,1,1, , ,
*DO,I,1,P_MAX,1
*SET,LAMBDA(I),L/I
*ENDDO
*MWRITE,LAMBDA(1),LAMBDA,TXT
(F12.7)
!*******************************
!* E N D
*****************************************************************
!*TABLE_P organized by P-number, not organized by mode correction.
!*Made with the intention of comparing answers, and obtaining a previous quick solution.
!*****************************************************************
*DIM,NIDF,ARRAY,MOD_CALC,1,1 , , ,
*SET,NFP,0        !*Number of frequencies in pairs
*DO,I,1,MOD_CALC-1,1  !*loop to the penultimate frequency
*IF,FREQLIST(I),EQ,FREQLIST(I+1),THEN
NFP=NFP+1          !*Repeated frequency counter
NIDF(NFP)=I        !*Number of frequencies identified
*ENDIF
*ENDDO
*DIM,P_LIST_DEBUGGED,ARRAY,NFP,1,1 , , ,
P_MAX=1
MOD_IDENTIF=0
*DO,I,1,NFP,1    !* CICLU PENTRU TOATE FRECVENTELE IDENTIFICATE
SET,1,NIDF(I)     !* INCARCA IN MEMORIE REZULTATELE CURENTE
!* EXTRAGEREA DEPLASARII MAX PE X SI Y
NSORT,U,X
*GET,MAX_UX,SORT,,MAX
*GET,NOD_MAX_UX,SORT,,IMAX
```

```
NSORT,U,Y
*GET,MAX_UY,SORT,,MAX
*GET,NOD_MAX_UY,SORT,,IMAX
P=P_LIM
COND_WHILE=1
*DOWHILE,COND_WHILE
P=P-1
*IF,P,GT,1,THEN
*IF,NX(NOD_MAX_UX)+L/P,LE,L,THEN
NOD_UX=NODE(NX(NOD_MAX_UX)+L/P,NY(NOD_MAX_UX),0)
*ELSE
NOD_UX=NODE(NX(NOD_MAX_UX)-L/P,NY(NOD_MAX_UX),0)
*ENDIF
!* GASESTE UN NOD PE X LA DISTANTA L/P CARE SA FIE VERIFICAT PENTRU UY
*IF,NX(NOD_MAX_UY)+L/P,LE,L,THEN
NOD_UY=NODE(NX(NOD_MAX_UY)+L/P,NY(NOD_MAX_UY),0)
*ELSE
NOD_UY=NODE(NX(NOD_MAX_UY)-L/P,NY(NOD_MAX_UY),0)
*ENDIF
*GET,UX_NOD,UX,NOD_UX
*GET,UY_NOD,UY,NOD_UY
*IF,ABS((MAX_UX-UX_NOD)/MAX_UX*100),LE,ERRP,AND,ABS((MAX_UY-
UY_NOD)/MAX_UY*100),LE,ERRP,THEN
P_FOUND_DEBUGGED=P
P_LIST_DEBUGGED(I)=P
COND_WHILE=0
*ENDIF
*IF,P_FOUND_DEBUGGED,GT,P_MAX,THEN
P_MAX_DEBUGGED=P_FOUND_DEBUGGED
*ENDIF
```

```
*ELSE  !* FOR P=1
P_LIST_DEBUGGED(I)=1
!*MOD_IDENTIF=MOD_IDENTIF+1
P_FOUND_DEBUGGED=1
COND_WHILE=0
*ENDIF
*ENDDO
*ENDDO   !*******************
!*END
!********************
```

Appendix B

Matlab code to obtain data table (frequencies and λ/p).

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  F R E Q U E N C Y    L I S T
%                         O B T E N T I O N
%                                       B Y   M O D E   T Y P E
%%%%%%%%%%%%%%%%%%   INPUTS      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
MOD_CALC=100;     % NUMBER OF MODES (FREQUENCIES) FOUND BY ANSYS


NEY=25; %NUMBER OF NODES, THIS DATA WAS ALSO ENTERED IN MACRO CODE IN
NEX=450;                        % ANSYS
L=0.015; %[m] % LENGTH OF THE PLATE
DELTA=0.0002625; %This delta is used to implement the derivation process for
% the …

CUTTING_NODE=313;
TOl_MAC_SYM=0.995;
TOl_MAC_ASYM=0.993;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %THE TABLES OBTAINED FROM THE PROCESS IN ANSYS ARE IMPORTED.
% When opening the MACRO file in ANSYS (Read Input From), the process is
% automatically performed, creating the NODAl information uploaded next.

load NODALINFOY         % D O F Solution
load NODALINFOY2    % Y Component Of Displacement
load NODALINFOY3
load NODALINFOY4
load NODALINFOY5
load NODALINFOY6
load NODALINFOX     % X Component Of Displacement
load NODALINFOX2
load NODALINFOX3
load NODALINFOX4
load NODALINFOX5
load NODALINFOX6

load FREQLIST       % FOUND FREQUENCY LIST
load P_LIST         % LIST OF VALUES OF P CORRESPONDING TO "FREQLIST"

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
       % ALL THE DISPLACEMENTS ARE IMPORTED INTO A SINGLE ARRAY.

Mode_Shape_Y=[NODALINFOY,NODALINFOY2,NODALINFOY3,NODALINFOY4,NODALINFOY5,NODA
LINFOY6];
%Represents the displacements in the 'Y' AXIS of all nodes.
```

```
Mode_Shape_X=[NODALINFOX,NODALINFOX2,NODALINFOX3,NODALINFOX4,NODALINFOX5,NODA
LINFOY6];
%Represents the displacements in the 'X' AXIS of all nodes.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                       % WAVE STRUCTURES are created.

NN=(NEY+1).*(NEX+1); %TOTAL NUMBER OF MODES

for kk=1:MOD_CALC
   rr=CUTTING_NODE:(CUTTING_NODE+NEY);
    Wave_Structure_Y(rr-(CUTTING_NODE-1),kk)=Mode_Shape_Y(rr,kk);
end
%The WaveStructure_Y are taken from the top horizontal line.
%of the plate. Therefore the nodes will be repeated in NEY+1.

for kk=1:MOD_CALC
    rr=CUTTING_NODE:(CUTTING_NODE+NEY);
    Wave_Structure_X(rr-(CUTTING_NODE-1),kk)=Mode_Shape_X(rr,kk); %VERIFICAR

end
%The WaveStructure_Y are taken from a random vertical line.
% on the plate (base on CUTTING_NODE).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% A Matrix is filled with the compared values of M A C between all the
%  frequencies obtained. (The diagonal with 1's is visible)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for j=1:MOD_CALC
    for i=1:MOD_CALC
        AMACX(i,j)=Mac(Wave_Structure_X(:,i),Wave_Structure_X(:,j));
        AMACY(i,j)=Mac(Wave_Structure_Y(:,i),Wave_Structure_Y(:,j));
    end
end

%If the plot of the 3D columns with respect to values of M A C
% the next section must be uncommented. The table for the 'X' AXIS
% presents more concrete values. AMACX by default


%%%%%%%%%%%%%%%%%%%%%%%%        3D PLOT          %%%%%%%%%%%%%%%%%%%%%%%%%

% %Figure
%  Z = AMACX(:,:);
%  b = bar3(Z);
%  colorbar
%  for k = 1:length(b)
%      zdata = b(k).ZData;
%      b(k).CData = zdata;
```

```
%       b(k).FaceColor = 'interp';
%end
% title('MAC')

 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Previously in ANSYS the frequencies with the number 'P' equal to 1 were
obtained.
% Therefore the first value of each type of mode found and by
% the decisive factor to fill each list by MODE TYPE.

POSIBLE_MODE_TYPE(1,:) = find(P_LIST(:,1)==1);
% Fill a row with the Numerals of the Modes(Frequencies) in which P=1

POSIBLE_MODE_TYPE_FREQ(1,:)= FREQLIST(POSIBLE_MODE_TYPE(1,:));
%The Numerals are replaced by the value of the frequency.

%From the frequencies corresponding to P=1.
% Only the values that are repeated will be considered, representing the
% same MODE SHAPE in opposite directions.

[Filas,Col]=size(POSIBLE_MODE_TYPE(1,:));
cont=0;
for zz= 1:Col-1

A=POSIBLE_MODE_TYPE_FREQ(1,zz);
D=POSIBLE_MODE_TYPE_FREQ(1,zz+1);
iswithin = ismembertol(A, D, 0.00005, 'ByRows', true);

 %%NOTE: The value of the tolerance may be modified according to the
 % frequency range.

       if iswithin == true
           cont=cont+1;
        FIRST_ROW(1,cont)=POSIBLE_MODE_TYPE(1,zz+1);
       end

end

% Finally FIRST_ROW will present the first row, i.e. P=1.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for d = 1:MOD_CALC
    FREQUENCIESX_SYM{:,d} =find(AMACX(:,d)>TOl_MAC_SYM) ;
    %Take from AMCAX the sub-steps that surpassed MAC threshold.

    SIZES_SYM(:,d) = size(FREQUENCIESX_SYM{d});
    % Saves the length of each resulting column.

end
```

```
 Rowmax_SYM = max(SIZES_SYM(1,:));
 % Maximum Row Number to define 'PREMODES' Matrix

 PREMODES_SYM = NaN(Rowmax_SYM,d);
 % Empty spaces will be filled NaN

for k1 = 1:d
       PREMODES_SYM(1:SIZES_SYM(1,k1),k1) = FREQUENCIESX_SYM{:,k1};
% LLenado.
end

%%%%%%%%%%%%%%%%%
for d = 1:MOD_CALC
    FREQUENCIESX_ASYM{:,d} =find(AMACX(:,d)>TOl_MAC_ASYM) ;
    %Take from AMCAX the sub-steps that surpassed MAC threshold.

    SIZES_ASYM(:,d) = size(FREQUENCIESX_ASYM{d});
    % Saves the length of each resulting column.
end

 Rowmax_ASYM = max(SIZES_ASYM(1,:));
% Maximum Row Number to define 'PREMODES' Matrix

 PREMODES_ASYM = NaN(Rowmax_ASYM,d);
% Empty spaces will be filled NaN

for k1 = 1:d
       PREMODES_ASYM(1:SIZES_ASYM(1,k1),k1) = FREQUENCIESX_ASYM{:,k1};
% LLenado.
end

%PREMODES %: It presents a table where the indexes of the columns
% represents the number of frequency from the list thrown by
% ANSYS, and in each column the number of modes meeting the criterion of
% MAC, i.e. each item in each column is close, and therefore
% can represent a TYPE OF MODE.
%, In this case, two were created for an easier process of iteration in case
% of being necessary, SYM (symmetrical) and ASYM (asymmetrical).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%       FILLING AND CLASSIFYING  %%%%%%%%%%%%%%%%%%

% This code is designed to only expect approximately 4 types of Modes on
% In case of finding more types, the next Loop must be modified
% and expanded to fill more Vectors.

%Random matrices are created and then filled.
A1=NaN(50,1);
A2=NaN(50,1);
S1=NaN(50,1);
S2=NaN(50,1);

 for ff=1:cont
```

```
     C=Wave_Structure_X(1,FIRST_ROW(1,ff));
     D=Wave_Structure_X(end,FIRST_ROW(1,ff));
     %The symmetry can be easily identified in the Wave Structures
iswithin2 = ismembertol(C, D, 0.00005, 'ByRows', true);


   if iswithin2== true
         %SYMETRIC
                if ff<=2
                   Pre_S(:,1)=FIRST_ROW(:,ff);
                   maxval=max(Pre_S(:,1));
                   ciclo=1;
                               while ciclo<=4
                                     S1=unique(vertcat(S1(:,1),
PREMODES_SYM(:,maxval)));
                                      %tempS=find((AMACX(:,maxval))>0.9);
%(*!*)%%(*!*)%
                                      %maxval=max(tempS(:,1));
                                       maxval=max(S1(:,1));
                                     ciclo=ciclo+1
                               end
                else
                    Pre_S2(:,1)=FIRST_ROW(:,ff);
                    maxval=max(Pre_S2(:,1));
                    ciclo=1
                               while ciclo<=4
                                     S2=unique(vertcat(S2(:,1),
PREMODES_SYM(:,maxval)));
%                                      tempS2=find((AMACX(:,maxval))>0.9);
%(*!*)%%(*!*)%
%                                      maxval=max(tempS2(:,1));
                                      maxval=max(S2(:,1));
                                     ciclo=ciclo+1
                               end
                end
   elseif iswithin2== false
          %ASYMETRIC

               if ff<=2
                   Pre_A(:,1)=FIRST_ROW(:,ff);
                   maxval=max(Pre_A(:,1));
                   ciclo=1;
                               while ciclo<=4
                                     A1=unique(vertcat(A1(:,1),
PREMODES_ASYM(:,maxval)));
                                     maxval=max(A1(:,1));
                                     ciclo=ciclo+1;
                               end
                else
                    Pre_A2(:,1)=FIRST_ROW(:,ff);
                    maxval=max(Pre_A2(:,1));
                    ciclo=1;
                               while ciclo<=4
```

```
                                     A2=unique(vertcat(A2(:,1),
PREMODES_ASYM(:,maxval)));
                                     maxval=max(A2(:,1));
                                     ciclo=ciclo+1;
                               end
                end
   end


 end

 % The sensibility of the MAC method is different in Symmetrical Modes Of
 % Asymmetric for this reason the two arrangements previously indicated
 % were created.

  %%%%%%%% DEBUGGING EQUAL MODES %%%%%%%%
% As stated above, frequencies will be repeated because
% each represents an opposite direction. Therefore the last step before
% presenting the final lists will be to debug the repeated values

 %%%%%%%%%%%% A1 %%%%%%
[ValA1, IdxA1] = max(A1);
  for yy=1:IdxA1
 A1(yy,2)=P_LIST(A1(yy,1),1);
 end
for ff=1:IdxA1
  A1(ff,1)=FREQLIST(A1(ff,1));

end

 cont=0;
for zz= 1:IdxA1-1

A=A1(zz,1);
D=A1(zz+1,1);
iswithin = ismembertol(A, D, 0.00005, 'ByRows', true);

%NOTE: The value of the tolerance may be modified according to the frequency
range
% .

       if iswithin == true
           cont=cont+1;
        A1_Freq(cont,1)=A1(zz+1,1);
        A1_P(cont,1)=A1(zz+1,2);
       end

end
A1_Lambda=(L)./A1_P;
% A1(isnan(A1)) = inf;
% A1 = uniquetol(A1.',0.0001);
% A1(isinf(A1)) = nan;
%  A1=(A1.');
```

```
 [P1_A1, COL2] = find(A1_Freq(:,1)<=1);
A1_Freq(P1_A1,:)=[];
A1_Lambda(P1_A1,:)=[];
%%%%%%%%%%%% A2 %%%%%%

 [ValA2, IdxA2] = max(A2);
  for yy=1:IdxA2
 A2(yy,2)=P_LIST(A2(yy,1),1);
 end
for ff=1:IdxA2
  A2(ff,1)=FREQLIST(A2(ff,1));
end
 cont=0;
for zz= 1:IdxA2-1

A=A2(zz,1);
D=A2(zz+1,1);
iswithin = ismembertol(A, D, 0.00005, 'ByRows', true);

%NOTE: The value of the tolerance may be modified according to the frequency
range

       if iswithin == true
           cont=cont+1;
        A2_Freq(cont,1)=A2(zz+1,1);
        A2_P(cont,1)=A2(zz+1,2);
       end

end
A2_Lambda=(L)./A2_P;
% A2(isnan(A2)) = inf;
% A2 = uniquetol(A2.',0.0001);
% A2(isinf(A2)) = nan;
%  A2=(A2.');

[P1_A2, COL2] = find(A2_Freq(:,1)<=1);
A2_Freq(P1_A2,:)=[];
A2_Lambda(P1_A2,:)=[];
%%%%%%%%%%%% S1 %%%%%%

 [ValS1, IdxS1] = max(S1);
  for yy=1:IdxS1
 S1(yy,2)=P_LIST(S1(yy,1),1);
 end
for ff=1:IdxS1
  S1(ff,1)=FREQLIST(S1(ff,1));
end
 cont=0;
for zz= 1:IdxS1-1

A=S1(zz,1);
D=S1(zz+1,1);
iswithin = ismembertol(A, D, 0.00005, 'ByRows', true);
```

```
%NOTE: The value of the tolerance may be modified according to the frequency
range
       if iswithin == true
           cont=cont+1;
        S1_Freq(cont,1)=S1(zz+1,1);
        S1_P(cont,1)=S1(zz+1,2);
       end

end
S1_Lambda=(L)./S1_P;

[P1_S1, COL2] = find(S1_Freq(:,1)<=1);
S1_Freq(P1_S1,:)=[];
S1_Lambda(P1_S1,:)=[];

%%%%%%%%%%% S2  %%%%%%

 S2(isnan(S2)) = 1;
S2 = uniquetol(S2.',0.0001);
% S2(isinf(S2)) = nan;
%  S2=(S2.');

 [ValS2, IdxS2] = max(S2);
 for yy=1:IdxS2
 S2(yy,2)=P_LIST(S2(yy,1),1);
 end
for ff=1:IdxS2
  S2(ff,1)=FREQLIST(S2(ff,1));
end
cont=0;
for zz= 1:IdxS2-1

A=S2(zz,1);
D=S2(zz+1,1);
iswithin = ismembertol(A, D, 0.00005, 'ByRows', true);

%NOTE: The value of the tolerance may be modified according to the frequency
range

       if iswithin == true
           cont=cont+1;
        S2_Freq(cont,1)=S2(zz+1,1);
        S2_P(cont,1)=S2(zz+1,2);
       else S2_P_=0;

       end

end
S2_Lambda=(L)./S2_P;
% S2(isnan(S2)) = inf;
% S2 = uniquetol(S2.',0.0001);
% S2(isinf(S2)) = nan;
```

```
%  S2=(S2.');

[P1_S2, COL2] = find(S2_Freq(:,1)<=1);
S2_Freq(P1_S1,:)=[];
S2_Lambda(P1_S2,:)=[];


% FINALLY of Each mode two lists are established _Freq and _Lambda
% Which will be the input variables to the plotting code of curves.

 sample=padcat(S1_Lambda,A1_Freq,S1_Freq,A2_Freq);
 %rowNames = {'A1','S1','A2'};
 GeneralTable = array2table(sample,'VariableNames',{'P','A1','S1','A2'});
 TABLE=padcat(S1_Lambda,A1_Freq,S1_Freq,A2_Freq);


 % The same procedure is repeated for the length backwards
 % and for the length forward.
```

Appendix C

Code for plotting curves and compare them with the ones obtained from GUIGW.

```
%%%%%Curve Comparision %%%%%%

%%% The data is loaded
%load plate    %%% DATA GUIGW
%filename = 'DataCurves.xlsx'; %%% Curve Infor From Table
%load DataTablaFinal


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%FREQUENCIES are saved into different vector from the main table FINALTABLE
A1Frec=TABLAFINAL(:,2);
A1Lambda=TABLAFINAL(:,11);

S1Frec=TABLAFINAL(:,8);
S1Lambda=TABLAFINAL(:,11);

A2Frec=TABLAFINAL(:,5);
A2Lambda=TABLAFINAL(:,11);

%%%%%% Data from the GUIGW is extracted in different vectors.
%They are organized by columns ( one columns one mode)
A1_GUI_Frec=Frequency_Hz(1,:);
A2_GUI_Frec=Frequency_Hz(1,:);
S1_GUI_Frec=Frequency_Hz(1,:);

A2_GUI_Frec(A2_GUI_Frec== 0)=NaN;

%Wavenumber
A1_GUI_K=Real_Wavenumber_1_m(1,:);
A2_GUI_K=Real_Wavenumber_1_m(2,:);
S1_GUI_K=Real_Wavenumber_1_m(3,:);

A2_GUI_K(A2_GUI_K== 0)=nan;

%Wavelength
A1_GUI_Lambda=Wavelength_m(1,:);
A2_GUI_Lambda=Wavelength_m(2,:);
S1_GUI_Lambda=Wavelength_m(3,:);

A2_GUI_Lambda(A2_GUI_Lambda== 0)=NaN;

%Phase Velocity
A1_GUI_Vf=Phase_Velocity_m_s(1,:);
A2_GUI_Vf=Phase_Velocity_m_s(2,:);
S1_GUI_Vf=Phase_Velocity_m_s(3,:);
```

```
%Group Velocity
A1_GUI_Vg=Group_Velocity_m_s(1,:);
A2_GUI_Vg=Group_Velocity_m_s(2,:);
S1_GUI_Vg=Group_Velocity_m_s(3,:);

A2_GUI_Vg(A2_GUI_Vg== 0)=NaN;

%%%%%%%%%%%%%%%%%%%                              %%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%      %%%%%%%%%%%%%%%%%%%       %%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% PHASE VELOCITY CALCULATION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

A1w=(A1Frec*2*pi); %[Hz]
A1k=(2*pi)./(A1Lambda); %[m]
A1Cf=(A1Lambda).*(A1Frec); %Vf


S1w=(S1Frec*2*pi);
S1k=(2*pi)./(S1Lambda);
S1Cf=(S1Lambda).*(S1Frec);%Vf

A2w=(A2Frec*2*pi);
A2k=(2*pi)./(A2Lambda);
A2Cf=(A2Lambda).*(A2Frec); %Vf


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Derivative data is obtained
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

A1FrecDeltaForward=TABLAFINAL(:,3);
A1FrecDeltaBackward=TABLAFINAL(:,1);
A1LambdaDeltaForward=TABLAFINAL(:,12);
A1LambdaDeltaBackward=TABLAFINAL(:,10);

A2FrecDeltaForward=TABLAFINAL(:,6);
A2FrecDeltaBackward=TABLAFINAL(:,4);
A2LambdaDeltaForward=TABLAFINAL(:,12);
A2LambdaDeltaBackward=TABLAFINAL(:,10);

S1FrecDeltaForward=TABLAFINAL(:,9);
S1FrecDeltaBackward=TABLAFINAL(:,7);
S1LambdaDeltaForward=TABLAFINAL(:,12);
S1LambdaDeltaBackward=TABLAFINAL(:,10);



%%%%%%%%%%%%%%%%%%%%%%%%%%
% GROUP VELOCITY CALCULATION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%
%     A   1
%%%%%%%%%%%%%%%%%%%
A1wDeltaForward=A1FrecDeltaForward*2*pi;
A1wDeltaBackward=A1FrecDeltaBackward*2*pi;
A1kDeltaForward=(2*pi)./A1LambdaDeltaForward;
A1kDeltaBackward=(2*pi)./A1LambdaDeltaBackward;

A1wDeltaMiddle=abs((A1wDeltaForward)-(A1wDeltaBackward));
A1kDeltaMiddle=abs((A1kDeltaForward)-(A1kDeltaBackward));

%%%%%%%%%%%%%%%%%%%
%     S   1
%%%%%%%%%%%%%%%%%%%
S1wDeltaForward=S1FrecDeltaForward*2*pi;
S1wDeltaBackward=S1FrecDeltaBackward*2*pi;
S1kDeltaForward=(2*pi)./S1LambdaDeltaForward;
S1kDeltaBackward=(2*pi)./S1LambdaDeltaBackward;

S1wDeltaMiddle=abs((S1wDeltaForward)-(S1wDeltaBackward));
S1kDeltaMiddle=abs((S1kDeltaForward)-(S1kDeltaBackward));

%%%%%%%%%%%%%%%%%%%
%     A   2
%%%%%%%%%%%%%%%%%%%
A2wDeltaForward=A2FrecDeltaForward*2*pi;
A2wDeltaBackward=A2FrecDeltaBackward*2*pi;
A2kDeltaForward=(2*pi)./A2LambdaDeltaForward;
A2kDeltaBackward=(2*pi)./A2LambdaDeltaBackward;

A2wDeltaMiddle=abs((A2wDeltaForward)-(A2wDeltaBackward));
A2kDeltaMiddle=abs((A2kDeltaForward)-(A2kDeltaBackward));


%%%&&&%%%%%

lengthA1=size(A1wDeltaMiddle);
for i=1:1:lengthA1-1
    A1Vg(i,1)=(A1wDeltaMiddle(i))./(A1kDeltaMiddle(i));
end

lengthS1=size(S1wDeltaMiddle);
for ii=1:1:lengthS1-1
    S1Vg(ii,1)=(S1wDeltaMiddle(ii))./(S1kDeltaMiddle(ii));
end

lengthA2=size(A2wDeltaMiddle);
for iii=1:1:lengthA2-1
    A2Vg(iii,1)=(A2wDeltaMiddle(iii))./(A2kDeltaMiddle(iii));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%     FREQUENCY VS WAVE NUMBER
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%FIGURA 1
figure(1);
%subplot(2,2,1)
plot(A1Frec,A1k,'b-s',S1Frec,S1k,'r-o',A2Frec,A2k,'k-
^',A1_GUI_Frec',A1_GUI_K','g--
',A2_GUI_Frec',A2_GUI_K','k:',S1_GUI_Frec',S1_GUI_K','m-.');
axis([0 600000 0 1000])
set(gca,'TickLabelInterpreter','latex','Xtick',0:100000:600000,'YTick',0:100:
1000)
legend('$A0$','$S0$','$A1$','$A0_{GUIGW}$','$S0_{GUIGW}$','$A1_{GUIGW}$');
set(legend,'Interpreter','latex');
set(legend,'FontSize',12);
%axis equal
grid on
xlabel('Frequency [Hz]','interpreter','latex','FontSize',13)
ylabel('Wavenumber [1/m]','interpreter','latex')
title('1','interpreter','latex')
%legend('A1','S1','A2')
% %axis tight
set(gca,'color','none')
%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% %FREQUENCY VS WAVE LENGTH
%%%%%%%%%%%%%%%%%%%%%%%%%%%
% %FIGURA 2
 figure(2);
%subplot(2,2,2)
plot(A1Frec,(A1Lambda),'b-s',S1Frec,S1Lambda,'r-o',A2Frec,A2Lambda,'k-
^',A1_GUI_Frec',A1_GUI_Lambda','g--
',A2_GUI_Frec',A2_GUI_Lambda','k:',S1_GUI_Frec',S1_GUI_Lambda','m-.');
axis([0 600000 0 0.1])
set(gca,'TickLabelInterpreter','latex','Xtick',0:100000:600000,'YTick',0:0.01
:0.1)
legend('$A0$','$S0$','$A1$','$A0_{GUIGW}$','$S0_{GUIGW}$','$A1_{GUIGW}$');
set(legend,'Interpreter','latex');
set(legend,'FontSize',12);
%axis equal
grid on
xlabel('Frequency [Hz]','interpreter','latex','FontSize',13)
ylabel('Wave length [m]','interpreter','latex')
title('1','interpreter','latex')
%axis tight
set(gca,'color','none')






% % % %%%%%%%%%%%%%%%%%
% % %FREQUENCY VS PHASE VELOCITY
%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% %FIGURA 3
figure(3);
%subplot(2,2,3)
plot(A1Frec,A1Cf,'b-s',S1Frec,S1Cf,'r-o',A2Frec,A2Cf,'k-
^',A1_GUI_Frec',A1_GUI_Vf','g--
',A2_GUI_Frec',A2_GUI_Vf','k:',S1_GUI_Frec',S1_GUI_Vf','m-.');
axis([0 600000 0 30000])
set(gca,'TickLabelInterpreter','latex','Xtick',0:100000:600000,'YTick',0:5000
:30000)
legend('$A0$','$S0$','$A1$','$A0_{GUIGW}$','$S0_{GUIGW}$','$A1_{GUIGW}$');
set(legend,'Interpreter','latex');
set(legend,'FontSize',12);
%axis equal
grid on
xlabel('Frequency [Hz]','interpreter','latex','FontSize',13)
ylabel('Phase velocity [m/s]','interpreter','latex')
title('1','interpreter','latex')
%axis tight
set(gca,'color','none')

% % % %%%%%%%%%%%%%
%% FREQUENCY VS GROUP VELOCITY
%%%%%%%%%%%%
% % % %FIGURA 4
figure(4);
%subplot(2,2,4)
plot(A1Frec(1:end-1),A1Vg,'b-s',S1Frec(1:end-1),S1Vg,'r-o',A2Frec(1:end-
1),A2Vg,'k-^',A1_GUI_Frec',A1_GUI_Vg','g--
',A2_GUI_Frec',A2_GUI_Vg','k:',S1_GUI_Frec',S1_GUI_Vg','m-.');
%plot(Frequency_Hz,Group_Velocity_m_s)
axis([0 600000 0 6000])
set(gca,'TickLabelInterpreter','latex','Xtick',0:100000:600000,'YTick',0:1000
:6000)
legend('$A0$','$S0$','$A1$','$A0_{GUIGW}$','$S0_{GUIGW}$','$A1_{GUIGW}$');
set(legend,'Interpreter','latex');
set(legend,'FontSize',12);
%axis equal
grid on
xlabel('Frequency [Hz]','interpreter','latex','FontSize',13)
ylabel('Group velocity [m/s]','interpreter','latex')
title('1','interpreter','latex')
%axis tight
set(gca,'color','none')
%%%%% print GUARA LA IMAGEN DE LA FIGURA EN FORMATO PSC PARA LATEX
%%%%%DEPSC A COLOR DEPS
%set(gca,'color','none')
%print -depsc Grafica3SinFondo
%print -deps CurvasBlancoyNegro
h=figure(1);

saveas(h,'PlotGeneral.jpg')
%saveas(h,'PlotGeneral.eps')
```