VK-Gong

User's manual

Version 1.0 - March 2017

Àngels Aragonès
Cédric Camier
Michele Ducceschi
Olivier Thomas
Cyril Touzé

HTTPS://VKGONG.ENSTA-PARISTECH.FR/

First publication, March 2017

Contents

IV Example cases

Introduction

Foreword

VK-gong find its origin in the article : C. Touzé, O.Thomas and A. Chaigne : Asymmetric nonlinear forced vibrations of free-edge circular plates, part I: theory , Journal of Sound and Vibration, vol 258, No 4, pp. 649-676, 2002 [29]. This paper was devoted to large-amplitude vibrations of free-edge circular plates. The von Kármán theory is used to model the geometrically nonlinear vibrations with moderate amplitudes, a framework which is sufficient for a number of applications ranging from engineering to musical acoustics. One of the key points addressed by the research initiated at this time by Antoine Chaigne who supervises the two PhD works by myself and Olivier Thomas (respectively defended in 2000 and 2001), was to simulate the sound produced by gongs and cymbals [24, 26]. These percussion instruments, though very simple in shape, present a very rich bright and shimmering sound with a broadband Fourier spectrum, manifestation of the strongly nonlinear dynamics that develops during the vibration. From this very first idea, numerous studies have been led in order to understand the nonlinear dynamics, the choice of mechanical models able to simulate such systems and finally the choice of adequate numerical methods that could be able to have both the accuracy, the stability and reasonable computing times. VK-gong thus benefits from all these developments, knowledge and comprehension gained around the strongly nonlinear vibrations of plates and shells with dynamics displaying from linear to wave turbulence regime.

VK-gong relys on the von Kármán model for thin plates and shells. This model can be viewed as the first nonlinear perturbation of the Kirchhoff-Love equations. It contains a number of approximations, but has been proven to be sufficient to model strngly nonlinear dynamics including turbulence. In particular, it has been used in [4, 5, 11, 13, 15, 19, 21, 27, 28] for all the works dedicated to a better understanding of the transition to turbulence and the energy transfer and repartition among wavelength once the turbulence is settled down.

With our goal of sound synthesis in mind, important features with regard to the numerical methods used to integrate the problem in time have been selected. This features makes VK-gong different from standard structural codes for dynamics, generally based on Finite Element procedures. Some peculiarities linked to the human auditory perception conducted to specific

choices. In particular:

- **Description of losses:** A fine and frequency-dependent description of losses is needed in order to recover correctly the temporal and spectral evolution and a realistic timbre.
- **Accuracy:** The characteristics of the plates, in particular the eigenfrequencies, need to be numerically finely reproduced without numerical dispersion or computational inaccuracies in the high-frequency range that could blur the sound numerically simulated.

For these reasons a modal approach has been selected for the space discretization of the nonlinear problems. The main advantages of such approach are the following:

- the accuracy of the computation of the eigenfrequencies can be finely controlled. In some cases where analytical solutions exist (*e.g.* for circular plates) the solution can be considered as exact.
- the losses can be tuned at ease by defining modal damping factors for each mode retained in the simulation.

Since the first paper in 2002, the models and methods have been upgraded by considering:

- a static, geometric imperfection in the shape of the plate, in order to take into account more easily shallow shells and different profiles. This has been realized in the PhD by Cédric Camier, defended in 2009 [6].
- the case of rectangular plates and energy-conserving scheme for the modal approach. The code has gained in computational time and generality. This parts were treated in the PhD by Michele Ducceschi, defended in 2014 [9].

The article : M. Ducceschi and C. Touzé: Modal approach for nonlinear vibrations of damped impacted plates: Application to sound synthesis of gongs and cymbals, Journal of Sound and Vibration, vol. 344, 313-331, 2015 [10]; summarizes the main outcomes and give a general framework adopted in this documentation. The first sounds of gongs and cymbals have been obtained in 2014 for circular and rectangular perfect plates.

The generality of the code and its performance in terms of robustness, accuracy, ease in the implementation of losses, makes it a very good candidate for simulations of large amplitudes of plates and shells that can be used in a number of engineering contexts. It has then be decided to offer an open source version. This work has been realized by Angels Aragones during a post-doc position funded by the european project ITN Batwoman. Translation of matlab code into C/C++, new functionnalities (*e.g.* elastically restrained boundary conditions for the circular case, imperfection for the rectangular plate, ...) has been added and the first release of VK-gong has been possible in March 2017.

Palaiseau, March 2017,
Cyril Touzé

1. Introduction

What is VK-gong ?

VK-gong is a numerical software for the time simulations of nonlinear (large amplitude) vibrations of thin plates and shallow shells. The code relies on the von Kármán theory for geometrically nonlinear vibrations, the main assumptions of which are recalled in chapter 2. In short, von Kármán theory is developed for moderate vibrations amplitudes, it neglects in-plane and rotary inertia, and assumes Kirchhoff-Love kinematics without shear. It covers nonetheless a large panel of situations, for vibration amplitude up to 10-50 times the thickness, depending on the application (static/dynamic) and the frequency range.

What are the main features ?

VK-gong relies on a modal approach for the space discretisation of the problem, in contrast to a number of other methods such as finite elements or finite difference. The advantages of using the modal approach are the following :

- the accuracy of the linear and nonlinear characteristics can be finely controlled and in some cases can be considered as exact, thanks to the use of analytical expressions of mode shapes, when available.
- the damping coefficients can be tuned at ease so that any frequency dependence of the losses can be implemented.
- The computation of the eigenfrequencies and the nonlinear coupling coefficients, are treated as off-line calculations that are done once and for all, once the geometry and material parameters have been selected. This crucial step can be realized with the desired accuracy, and is then used for each time simulation.
- Analytical treatments are pushed as far as possible to maintain the accuracy. For particular cases where analytical solutions are available for the linear part (see *e.g.* the circular plate), this leads to a refined computations of all the coefficients feeding the model.

The second main feature of the code is the use of two different numerical methods for time integration :

- The Störmer-Verlet scheme (also known as leap-frog). This scheme is symplectic, second-order accurate and conditionnally stable [14].
- An energy-conserving scheme especially designed for the case of plates with a modal approach. The scheme is also second-order accurate and conditionnally stable [10].

VK-gong has been primarily developed in Matlab language. It has recently be translated into a C/C++ version, so that both Matlab and C/C++ version are given to the user.

What VK-gong can (and cannot) do ?

The core of VK-gong is the expression of the linear and nonlinear characteristics (coupling coefficients) of the von Kármán plate PDE of motions, expressed in the basis of the eigenmodes, giving rise to a finite-dimensional system, the linear part of which is diagonal. This core can be used for a number of applications, be they static or dynamic. For instance, the use of such an approach for solving the buckling of circular plates is shown in [18, 20]. However in its present 1.0 version, VK-gong is only oriented toward time-domain simulations of vibrating plates with large amplitude. The inputs to be defined for running a simulation are the following:

choice of the plate : one has to define the geometry, the material parameters, the boundary conditions, the presence of an imperfection, and the damping law.

choice of the external forcing : to define the forces acting on the plate.

choice of the simulation parameters : selection of appropriate values for numerical integration.

About the plate selection, VK-gong 1.0 can handle the following cases:

geometry : only circular and rectangular plates are possible at present. Plates of arbitrary shape are not allowed. In the two cases the thickness is assumed to be constant and is denoted as h_d. The circular geometry is then defined by its radius R_d, while the rectangular geometry is defined by L_x and L_y.

material : as isotropic linear elasticity is assumed, the material parameters are the Young modulus E and the Poisson ratio ν. Other behaviour laws are not allowed.

Boundary conditions : three different cases of boundary conditions are possible for the circular plate : completely free edge, clamped edge, and elastically restrained with in-plane movable edge. For the rectangular case, only simply-supported transverse boundary conditions with in-plane movable edges are possible at present.

imperfection : it is possible to define a purely geometrical imperfection of the shape of the plate without pre-stress. This allows to extend the cases treated to shallow shells.

losses : any frequency-dependent frequency law can be input to the code.

External forces are given as pointwise excitations with different temporal contents (raised cosine for simulating an impact, harmonic sine excitation, random noise, ...). Preset values are offered in the code, however a user can extend the code to take into account other forces.

Structure of this documentation

The documentation begins by recalling the general theory for solving the von Kármán PDE of motions using a modal approach. Definitions af all related quantities are introduced, in a consistent manner with the notations used in the code. In particular all the formulas explaining how to compute all the linear and nonlinear characteristics are reviewed.

Then a full documentation explaining all the functions used in the two versions (MATLAB and C/C++) is given so as to have a detailed view of how the code is structured. Finally a number of examples are given. These should be a good starting point for a user to take charge of the code.

Words of caution

VK-gong is a code developed from a research effort. It is freely available for scientific use. It is a typical research program which is provided "as is", with no guarantee whatsoever. The code is also completely open. A user can be free to take some parts of the code here and there to develop his own computations and applications, while still referencing the source VK-gong.

The computational burden associated to the simulations can be very heavy. In particular the number of transverse modes retained in the simulations is a critical parameter and has to be taken with care. It is generally advised to begin with small number so as to test the capacity of the computer and slowly increase the numbers.

VK-gong has been first coded in Matlab language. Consequently the MATLAB version has to be taken as a reference. It has been tested in a number of context and is thus provided as the most reliable. The C version has been translated from the Matlab sources. It has not been tested in a systematic manner as the matlab version has been. Moreover the Störmer-Verlet version is not stabilized and, in this current version 1.0, has been found to diverge for some tuning parameters. On the other hand the results given with the energy-conserving scheme are more reliable. Current effort is put to solve this problems which should be fixed in the next available version.

II Theory

2. General equations

2.1 Von Kármán equations

The equations used to model the nonlinear (large amplitude) vibrations of thin plates in this work are the so-called von Kármán equations [2, 10, 16, 17, 22, 25, 29]. Von Kármán equations can be used as long as the following set of assumptions is fulfilled [8, 25],

- The plate thickness must be significantly smaller than the other plate dimensions.
- The Kirchhoff-Love hypotheses are fulfilled and thus, the transverse shear is neglected.
- The strain tensor is limited to the second order terms.
- in-plane and rotatory inertia are neglected so that an Airy stress function can be used.

Given a plate of volume mass density ρ, thickness h and flexural rigidity $D = Eh^3/12(1-\nu^2)$, with E standing for the Young modulus and ν for the Poisson ratio, they read

$$\rho h\ddot{w} + D\Delta\Delta w = \mathscr{L}(w,F) + p(\mathbf{x},t) - R(\dot{w}), \tag{2.1a}$$

$$\Delta\Delta F = -\frac{Eh}{2}\mathscr{L}(w,w). \tag{2.1b}$$

where the plate vibration is expressed by means of two main variables, the transverse displacement $w(\mathbf{x},t)$ and the Airy stress function $F(\mathbf{x},t)$, with $\mathbf{x}$ denoting the space variable and t the time. The operator Δ represents the two dimensional Laplacian operator, $p(\mathbf{x},t)$ a time-depending excitation and $R(\dot{w})$, a generic term to express the plate damping, that can vary with frequency.

The operator $\mathscr{L}(w,F)$ is known as the von Kármán operator and accounts for the coupling between the transverse and in-plane displacements. For two functions $f(\mathbf{x})$ and $g(\mathbf{x})$, it may be expressed in intrinsic coordinates as

$$\mathscr{L}(f,g) = \Delta f\Delta g - \nabla\nabla f : \nabla\nabla g \tag{2.2}$$

where : denotes the doubly contracted product of two tensors.

R Particular cases of von Kármán operator.

- Polar coordinates (θ, r):

$$\mathscr{L}(f,g) = f_{,rr}\left(\frac{g_{,r}}{r} + \frac{g_{,\theta\theta}}{r^2}\right) + g_{,rr}\left(\frac{f_{,r}}{r} + \frac{f_{,\theta\theta}}{r^2}\right) - 2\left(\frac{f_{,r\theta}}{r} + \frac{f_{,\theta}}{r^2}\right)\left(\frac{g_{,r\theta}}{r} + \frac{g_{,\theta}}{r^2}\right) \tag{2.3}$$

- Cartesian coordinates (x, y):

$$\mathscr{L}(f,g) = f_{,xx}g_{,yy} + f_{,yy}g_{,xx} - 2f_{,xy}g_{,xy} \tag{2.4}$$

2.2 Imperfect plate case

Imperfect plates are those that present a static deformation in their profile at equilibrium, either because of their shape or due to factory defects. The variations in the plate shape may affect the nonlinear behaviour and thus, they must be included in the model.

This can be done by redefining the transverse displacement $w(\mathbf{x},t)$ in eq. (2.1) as a combination of two terms [7, 23, 25],

$$w(\mathbf{x},t) = w_0(\mathbf{x}) + \tilde{w}(\mathbf{x},t). \tag{2.5}$$

As shown in fig. 2.1, $w_0(\mathbf{x})$ corresponds to the static displacement without pre-stress, i.e. the imperfection height at every point, whereas $\tilde{w}(\mathbf{x},t)$ equals to the dynamic transverse displacement with respect to $w_0(\mathbf{x})$.

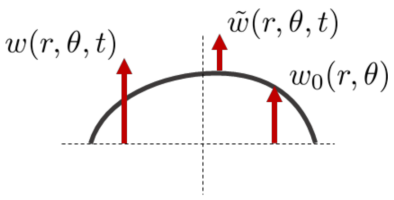


Figure 2.1: Redefinition of transverse displacement to include imperfection.

Definition eq. (2.3) is introduced in eq. (2.1). After a static equilibrium, the equations of motion for the new unknown $\tilde{w}(\mathbf{x},t)$ read:

$$\rho h\ddot{\tilde{w}} + D\Delta\Delta\tilde{w} = \mathscr{L}(\tilde{w},F) + \mathscr{L}(w_0,F) + p(\mathbf{x},t) - R(\dot{w}), \tag{2.6a}$$

$$\Delta\Delta F = -\frac{Eh}{2}\left[\mathscr{L}(\tilde{w},\tilde{w}) + 2\mathscr{L}(\tilde{w},w_0)\right]. \tag{2.6b}$$

Equation (2.1) presented linear and cubic nonlinear terms. After the incorporation ot the imperfection, new quadratic terms appear proving that the profile variation directly affects the nonlinear behaviour of the plate.

2.3 Modal approach for the perfect plate

A modal approach is used to discretize the von Kármán equations [10]. This procedure implies several advantages. First, it enables the introduction of a frequency dependent damping into the model, resulting in a broader applicability of the method. This is especially interesting in sound synthesis applications where the ear is particularly sensitive to losses and associated decay times. It repesent also a very interesting feature as compared to *e.g.* finite element codes where one has

to work with Rayleigh dissipation only, not offering such a broad range of applications. Second, there is a set of parameters that can be computed off-line. This means that if the simulation must be repeated or the accuracy must be increased, they do not have to be recalculated. Furthermore their accuracy can be finely controlled and in general exact converged values are used, giving to the code a great accuracy without any numerical dispersion, neither in the eigenfrequencies, nor in the nonlinear coupling coefficients. This feature is also of paramount importance for sound synthesis in order to obtain realistic sounds.

Let $\{\Phi_k(\mathbf{x})\}_{k\geq 1}$ be the eigenmodes of the transverse displacement and $\{\Psi_i(\mathbf{x})\}_{i\geq 1}$ the eigenmodes for the Airy stress function, obtained by solving the eigenproblems

$$\left(\boldsymbol{\Delta\Delta} - \xi^4\right)\boldsymbol{\Phi} = 0, \tag{2.7a}$$

$$\left(\boldsymbol{\Delta\Delta} - \zeta^4\right)\boldsymbol{\Psi} = 0. \tag{2.7b}$$

As we first focus on the perfect plate case, the main variable for the transverse displacement is $w(\mathbf{x},t)$, which is expanded together with the Airy stress function $F(\mathbf{x},t)$ as

$$w(\mathbf{x},t) = S_W \sum_{p=1}^{N_\Phi} \frac{\Phi_p(\mathbf{x})}{\|\Phi_p\|} q_p(t), \tag{2.8a}$$

$$F(\mathbf{x},t) = S_F \sum_{p=1}^{N_\Psi} \frac{\Psi_p(\mathbf{x})}{\|\Psi_p\|} \eta_p(t), \tag{2.8b}$$

where $q_p(t)$ and $\eta_p(t)$ are respectively the transverse and in-plane modal variables and S_W and S_F the normalization constants of the family modes, i.e. $\bar{\Phi} = S_W\Phi_k(\mathbf{x})/\|\Phi_k\|$ and $\bar{\Psi} = S_F\Psi_k(\mathbf{x})/\|\Psi_k\|$. The values N_Φ and N_Ψ determine the number of modes included in the truncated series. These modal basis will be computed differently depending on the plate shape and the boundary conditions. This will be seen in the following chapters.

After replacing $w(\mathbf{x},t)$ and $F(\mathbf{x},t)$ by eq. (2.8), eq. (2.1a) and eq. (2.1b) are multiplied respectively by Φ_s and Ψ_s and integrated over the plate surface S, leading to the temporal system of Ordinary Differential Equations (ODEs),

$$\ddot{q}_s + \omega_s^2 q_s + 2\mu_s\omega_s\dot{q}_s = \frac{S_F}{\rho h}\sum_{k=1}^{N_\Phi}\sum_{l=1}^{N_\Psi} E_{kl}^s q_k \eta_l + p_s(t), \tag{2.9a}$$

$$\eta_l = -\frac{Eh}{2\zeta_l^4}\frac{S_W^2}{S_F}\sum_{m,n}^{N_\Phi} H_{mn}^l q_m q_n. \tag{2.9b}$$

with $\omega_s = \xi_s^2$ standing for the angular frequency of transverse mode s, μ_s the damping coefficient associated to mode s and ζ_l the eigenvalue correspondent to in-plane mode l. E_{kl}^s and H_{mn}^l are the modal coupling coefficients defined in the next section. Note that the external force $p(t)$ has also been expressed in modal terms as

$$p_s(t) = \frac{1}{\rho h S_W \|\Phi_s\|}\int_S p(\mathbf{x},t)\Phi_s(\mathbf{x})\mathrm{d}S. \tag{2.10}$$

2.3.1 Coupling coefficients: H and Γ

The modal approach also leads to the introduction of the modal coupling coefficients $E^s_{k,l}$ and $H^l_{m,n}$, defined as

$$E^s_{kl} = \frac{\int_S \Phi_s \mathscr{L}(\Phi_k, \Psi_l)\,\mathrm{d}S}{\|\Phi_s\|\|\Phi_k\|\|\Psi_l\|}, \tag{2.11a}$$

$$H^l_{mn} = \frac{\int_S \Psi_l \mathscr{L}(\Phi_m, \Phi_n)\,\mathrm{d}S}{\|\Psi_l\|\|\Phi_m\|\|\Phi_n\|}, \tag{2.11b}$$

which quantify the non-linear coupling between transverse and in-plane triads of modes. Note that thanks to the bilinearity of the von Kármán operator, the symmetry allows that $H^l_{mn} = H^l_{nm}$. In addition, depending on the boundary conditions at the edge of the plate, it may be fulfilled that $E^s_{kl} = H^l_{sk}$ [10, 25]. This is the case of all the configurations considered in this work.

On the other hand, although for some time stepping schemes both equations in eq. (2.9) are necessary, a closed form of the system, only depending on the transverse displacement $w(\mathbf{x})$, can be obtained with the substitution of η_l in eq. (2.9a) by eq. (2.9b),

$$\ddot{q}_s + \omega_s^2 q_s + 2\mu_s\omega_s\dot{q}_s = -\frac{ES_w^2}{\rho}\sum_{k,m,n=1}^{N_\Phi}\left[\sum_{l=1}^{N_\Psi}\frac{E^s_{kl}H^l_{mn}}{2\zeta_l^4}\right] q_k q_m q_n + p_s(t) \tag{2.12}$$

leading to the definition of the fourth-order tensor,

$$\Gamma^s_{kmn} = \sum_{l=1}^{N_\Psi}\frac{E^s_{kl}H^l_{mn}}{2\zeta_l^4}. \tag{2.13}$$

2.3.2 Modal approach for imperfect plates

In order to introduce the imperfection in the modal equations, the static displacement $w_0(\mathbf{x})$ is also expressed in terms of the transverse mode basis by projecting the profile on this family of modes [7],

$$w_0(\mathbf{x}) = S_W\sum_{p=1}^{N_0}\frac{\Phi_p(\mathbf{x})}{\|\Phi_p\|}a_p + z_g \tag{2.14}$$

where the projection coefficients a_p and the center of mass z_g are computed thanks to the orthogonality property of the eigenmodes,

$$a_p = \iint_S w_0(\mathbf{x})\Phi_p(\mathbf{x})\,\mathrm{d}S, \tag{2.15}$$

$$z_g = \frac{\iint_S w_0(\mathbf{x})\,\mathrm{d}S}{A_p}, \tag{2.16}$$

where S is the plate surface domain and A_p is the plate area.

The same procedure used for the perfect plate is now used in eq. (2.6), adding in this case the definition of w_0 in eq. (2.14). The resulting expression is

$$\ddot{q}_s + \omega_s^2 q_s + 2\mu_s\omega_s\dot{q}_s = \frac{S_F}{\rho h}\sum_{k=1}^{N_\Phi}\sum_{l=1}^{N_\Psi}E^s_{kl}(q_k + a_k)\eta_l + p_s(t) \tag{2.17a}$$

$$\eta_l = -\frac{Eh}{2\zeta_l^4}\frac{S_W^2}{S_F}\sum_{m,n}^{N_\Phi}H^l_{mn}\left(q_m q_n + 2q_m a_n\right). \tag{2.17b}$$

Observe that in case of a perfect plate, the projection coefficients are null and the perfect plate eqs. (2.9a) and (2.9b) are recovered.

2.4 Eigenfrequencies of the imperfect plate

The eigenfrequencies of the imperfect plate can be calculated from the eigenfrequencies of the equivalent perfect plate [7].

Let

$$\mathbf{A} = \{\varepsilon\alpha_m^s + \omega_s^2\delta_{sm}\}, \tag{2.18}$$

denote the linear part of (2.12) in matrix form with

$$\alpha_m^s = \sum_m^{N_\Phi}\sum_n^{N_\Phi} 2\Gamma_{kmn}^s a_k a_n, \tag{2.19}$$

δ_{sm} the usual Kronecker delta symbol and ε a constant that depends on the shape and materials of the plate. (See section 5.2.)

The diagonalization of $\mathbf{A}$ by means of

$$\{\Omega_s\delta_{sk}\}_{s,k\in[1,N_\Phi]} = \mathbf{P}^{-1}\mathbf{A}\mathbf{P} \tag{2.20}$$

provides the eigenfrequencies of the imperfect plate in Ω_s and the mode shapes in $\mathbf{P}$.

One can observe in eq. (2.20) that the variation of the imperfect plate's eigenfrequencies with respect to the perfect plate not only depends on the magnitude of the imperfection (expressed through the $\{a_p\}$ factors), but also on the coupling coefficients that involve the projected modes via the Γ_{kmn}^s.

3. Circular plates

Besides material parameters and thickness, circular plates are characterized by their radius R_d. In this chapter, particularities of circular plates are explained and followed by the summary of boundary conditions included in the code.

3.1 Non-dimensioning of variables

The quantities involved in the model of circular plates are of significantly different orders of magnitude. For this reason, their redefinition in a dimensionless form is advised to avoid numerical problems.

Amongst the multiple possibilities to apply the non-dimensioning of variables, the simplest one is by using the plate thickness h and radius R_d as follows [6, 7],

$$\tilde{w} = h\bar{\tilde{w}},\ w_0 = h\bar{w}_0,\ r = R_d\bar{r}. \tag{3.1}$$

Accordingly, for the Airy stress function F, time variable t, input force p and modal damping μ_s,

$$\tilde{F} = Eh^3\bar{\tilde{F}},\ t = \sqrt{\frac{\rho h R_d^4}{D}}\bar{t},\ p = \frac{h^4 E}{R_d^4}\bar{p},\ \mu = \frac{Eh^3}{2R_d^2}\sqrt{\frac{\rho h}{D}}\bar{c}. \tag{3.2}$$

where the bars $\bar{(\cdot)}$ denote dimensionless variables.

Replacing the new definitions in (2.12) and omitting the over-bars to simplify the notation, the equations of motion of the imperfect plate can be rewritten as

$$\ddot{\tilde{w}} + \Delta\Delta\tilde{w} = \varepsilon\left[\mathscr{L}(\tilde{w}, F) + \mathscr{L}(w_0, F)\right] + p(r,\theta,t) - 2c\dot{\tilde{w}}, \tag{3.3a}$$

$$\Delta\Delta F = -\frac{1}{2}\left[\mathscr{L}(\tilde{w},\tilde{w}) + 2\mathscr{L}(\tilde{w}, w_0)\right] \tag{3.3b}$$

with $\varepsilon = 12(1-\nu^2)$.

Equation (2.18) can be used to compute the dimensionless eigenfrequencies of the imperfect plate by using ε instead of ε_d.

The use of dimensionless variables implies a positive side effect. Computation of H and Γ coupling coefficients as well as pairs of eigenmodes and eigenfrequencies can be made once an reused for every plate with the same boundary conditions, with the sole application of a multiplicative factor.

3.2 Boundary conditions

This section recalls the equations that must be fulfilled to reproduce the boundary conditions in every direction of displacement. For further details, the reader can refer to [12, 25, 29, 30].

3.2.1 General boundary conditions

Boundary conditions in term of forces and displacements

We use the classical orthonormal basis for the cylindrical coordinate system $(\mathbf{e}_r, \mathbf{e}_\theta, \mathbf{e}_z)$, with $\mathbf{e}_r$, $\mathbf{e}_\theta$ respectively normal and tangent to the circular boundary of the plate and $\mathbf{e}_z$ normal to the plate mid-plane.

In the most general case, the boundary is subjected to an external forcing, represented by a force field $T_e\mathbf{e}_z + \mathbf{N}_e$ and a moment field $\mathbf{M}_e$, with $\mathbf{N}_e$ and $\mathbf{M}_e$ two vectors parallel to the plate mid-plane. One can also impose an in-plane displacement $\mathbf{u}_e$, a transverse displacement w_e or a normal rotation of the edge φ_{re}.

At any point of the plate boundary, the boundary conditions write:

$$\mathbf{u} = \mathbf{u}_e \quad \text{or} \quad \mathsf{N}\mathbf{e}_r = \mathbf{N}_e \tag{3.4a,b}$$

$$w = w_e \quad \text{or} \quad V_r = Q_r + \frac{1}{r}\frac{\partial M_{r\theta}}{\partial \theta} = T_e + \frac{1}{r}\frac{\partial}{\partial \theta}\left(\mathbf{M}_e \cdot \mathbf{e}_\theta\right), \tag{3.4c,d}$$

$$\frac{\partial w}{\partial r} = \varphi_{re} \quad \text{or} \quad M_{rr} = \mathbf{M}_e \cdot \mathbf{e}_r. \tag{3.4e,f}$$

where Q_r is the internal shear force, V_r is the Kirchhoff shear force, N is the membrane force tensor and M the bending moment tensors, whose useful components write:

$$N_{rr} = \frac{1}{r}\frac{\partial F}{\partial r} + \frac{1}{r^2}\frac{\partial^2 F}{\partial \theta^2}, \qquad N_{r\theta} = -\frac{1}{r}\frac{\partial^2 F}{\partial r \partial \theta} + \frac{1}{r^2}\frac{\partial F}{\partial \theta}; \tag{3.5a}$$

$$Q_r = -D\frac{\partial \nabla^2 w}{\partial r} + N_{rr}\frac{\partial w}{\partial r} + \frac{N_{r\theta}}{r}\frac{\partial w}{\partial \theta}; \tag{3.5b}$$

$$M_{rr} = -D\left(\frac{\partial^2 w}{\partial r^2} + \frac{\nu}{r}\frac{\partial w}{\partial r} + \frac{\nu}{r^2}\frac{\partial^2 w}{\partial \theta^2}\right), \qquad M_{r\theta} = -D(1-\nu)\left(\frac{1}{r}\frac{\partial^2 w}{\partial r \partial \theta} - \frac{1}{r^2}\frac{\partial w}{\partial \theta}\right) \tag{3.5c}$$

In-plane simple boundary conditions in term of F and w

The in-plane part of the above boundary conditions, in simple cases of free or immovable edge, can be rewritten as **sufficient** conditions in term of F only [25]:

- Free edge ($N_{rr} = N_{r\theta} = 0$ on the boundary):

$$F = \frac{\partial F}{\partial r} = 0. \tag{3.6}$$

- Immovable edge, ($\mathbf{u} = \mathbf{0}$ on the boundary):

$$\frac{\partial^2 F}{\partial r^2} - \nu\left(\frac{1}{r}\frac{\partial F}{\partial r} + \frac{1}{r^2}\frac{\partial^2 F}{\partial \theta^2}\right) + \mathcal{N}_1(w) = 0, \tag{3.7a}$$

$$\frac{\partial^3 F}{\partial r^3} + \frac{1}{r}\frac{\partial^2 F}{\partial r^2} - \frac{1}{r^2}\frac{\partial F}{\partial r} + \frac{2+\nu}{r^2}\frac{\partial^3 F}{\partial r \partial \theta^2} - \frac{3+\nu}{r^3}\frac{\partial^2 F}{\partial \theta^2} + \mathcal{N}_2(w) + \mathcal{N}_F(\partial w/\partial r) = 0. \tag{3.7b}$$

  with $\mathcal{N}_1$, $\mathcal{N}_2$ and $\mathcal{N}_F$ nonlinear functions of the transverse displacement w and slope and $\partial w/\partial r$ [25]).

For more general cases, for instance in the case of an elastic edge that imposes a relation between N and $\mathbf{u}$, the corresponding boundary conditions include nonlinear terms functions of w.

3.2.2 Particular boundary conditions

Due to some nonlinear terms in the boundary conditions (the terms $\mathcal{N}_1(w)$, $\mathcal{N}_2(w)$ and $\mathcal{N}_F(\partial w/\partial r)$ in Eqs. (3.7a,b) and the term N*vgradw* in the expression of Q_r in Eq. (3.5b)), only the ones for which those terms vanish are considered here.

Completely free edge

The external loads vanish at the plate edge ($N_{rr} = N_{r\theta} = M_{rr} = Q_r + \partial M_{r\theta}/(r\partial\theta) = 0$), so that:

$$F = \frac{\partial F}{\partial r} = 0 \tag{3.8a}$$

$$\frac{\partial^2 w}{\partial r^2} + \nu\left(\frac{1}{r}\frac{\partial w}{\partial r} + \frac{1}{r^2}\frac{\partial^2 w}{\partial \theta^2}\right) = 0, \tag{3.8b}$$

$$\frac{\partial^3 w}{\partial r^3} + \frac{1}{r}\frac{\partial^2 w}{\partial r^2} - \frac{1}{r^2}\frac{\partial^2}{\partial r} + \frac{2-\nu}{r^2}\frac{\partial^3 w}{\partial r \partial \theta^2} - \frac{3-\nu}{r^3}\frac{\partial^2 w}{\partial \theta^2} = 0. \tag{3.8c}$$

Completely clamped edge

The displacements vanish at the plate edge ($u_r = u_\theta = w = \partial w/\partial r = 0$), so that:

$$\frac{\partial^2 F}{\partial r^2} - \nu\left(\frac{1}{r}\frac{\partial F}{\partial r} + \frac{1}{r^2}\frac{\partial^2 F}{\partial \theta^2}\right) = 0, \tag{3.9a}$$

$$\frac{\partial^3 F}{\partial r^3} + \frac{1}{r}\frac{\partial^2 F}{\partial r^2} - \frac{1}{r^2}\frac{\partial F}{\partial r} + \frac{2+\nu}{r^2}\frac{\partial^3 F}{\partial r \partial \theta^2} - \frac{3+\nu}{r^3}\frac{\partial^2 F}{\partial \theta^2} = 0, \tag{3.9b}$$

$$w = \frac{\partial w}{\partial r} = 0. \tag{3.9c}$$

Transversely clamped and in-plane free edge

The transverse displacement and slope vanish at the plate edge ($w = \partial w/\partial r = 0$) whereas the membrane displacements are free ($N_{rr} = N_{r\theta}$), so that :

$$F = \frac{\partial F}{\partial r} = 0, \tag{3.10a}$$

$$w = \frac{\partial w}{\partial r} = 0. \tag{3.10b}$$

Transversely elastic and in-plane free edge

Some distributed stiffness in translation K_T (expressed in [N·m/m/rad=N·rad^{-1}]) and in rotation K_R (expressed in [N/m/m=N·m^{-2}]) are prescribed at the edge of the plate, so that the bending moment

M_{rr} and the Kirchhoff shear force V_r are related to their conjugated displacement variables, the deflection and the slope:

$$M_{rr} = K_R \frac{\partial w}{\partial r}, \qquad V_r = -K_T w. \tag{3.11}$$

To avoid the nonlinear terms in the equations, the in-plane boundary conditions must be prescribed free. On then obtains:

$$F = \frac{\partial F}{\partial r} = 0, \tag{3.12a}$$

$$D\left[\frac{\partial^3 w}{\partial r^3} + \frac{1}{r}\frac{\partial^2 w}{\partial r^2} - \frac{1}{r^2}\frac{\partial^2}{\partial r} + \frac{2-\nu}{r^2}\frac{\partial^3 w}{\partial r \partial \theta^2} - \frac{3-\nu}{r^3}\frac{\partial^2 w}{\partial \theta^2}\right] - K_T w = 0, \tag{3.12b}$$

$$D\left[\frac{\partial^2 W}{\partial r^2} + \nu\left(\frac{1}{r}\frac{\partial W}{\partial r} + \frac{1}{r^2}\frac{\partial^2 W}{\partial \theta^2}\right)\right] + K_r \frac{\partial w}{\partial r} = 0. \tag{3.12c}$$

3.3 Mode families and modal coupling coefficients for some combinations of boundary conditions

Throughout this section, the modal families for the previous boundary conditions are computed by solving eigenproblems in eq. (2.7). The most common combinations of transverse / in-plane / rotational boundary conditions are revisited and some examples of coupling coefficients are included. Note that these computations correspond to perfect plates.

3.3.1 Circular plate with free edge

Given a plate with free edge, eqs. (3.8a) to (3.8c) must be fulfilled. Regarding the transverse direction, equations eqs. (3.8b) and (3.8c) are rewritten in terms of the eigenmodes $\{\Phi_i(\mathbf{x})\}_{i\geq 1}$ in dimensionless form, so that for all θ and t,

$$\boldsymbol{\Phi}_{,rr} + \nu\boldsymbol{\Phi}_{,r} + \nu\boldsymbol{\Phi}_{,\theta\theta} = 0 \quad \text{at } r = 1, \tag{3.13}$$

$$\boldsymbol{\Phi}_{,rrr} + \boldsymbol{\Phi}_{,rr} - \boldsymbol{\Phi}_{,r} + (2-\nu)\boldsymbol{\Phi}_{,r\theta\theta} - (3-\nu)\boldsymbol{\Phi}_{,\theta\theta} = 0 \quad \text{at } r = 1, \tag{3.14}$$

$$\boldsymbol{\Phi}(r=0) \quad \text{is bounded.} \tag{3.15}$$

Note that the term $\mathcal{N}_w$ in eq. (3.8c) is canceled because of eq. (3.8a).
Solutions can be separated in r and θ, so that

$$\boldsymbol{\Phi}_{0n}(r,\theta) = R^f_{0n}(r) \quad \text{for } k = 0, \tag{3.16}$$

$$\begin{vmatrix}\boldsymbol{\Phi}_{kn1}(r,\theta)\\ \boldsymbol{\Phi}_{kn2}(r,\theta)\end{vmatrix} = R^f_{kn}(r)\begin{vmatrix}\cos k\theta\\ \sin k\theta\end{vmatrix} \quad \text{for } k > 0, \tag{3.17}$$

where

$$R^f_{kn}(r) = \kappa_{kn}\left[J_k(\xi_{kn}r) - \frac{\tilde{J}^f_k(\xi_{kn})}{\tilde{I}^f_k(\xi_{kn})} I_k(\xi_{kn}r)\right] \tag{3.18}$$

with $J_k(x)$ being the k-th order Bessel function of the first kind and $I_k(x) = J_k(ix)$ the k-th order modified Bessel function of the first kind. κ_{kn} is the normalization constant of mode Φ set to fulfill $\|\Phi_{kn}\| = 1$. Special terms $\tilde{J}^f_k(x)$ and $\tilde{I}^f_k(x)$ are computed as follows,

$$\tilde{J}_k^f(x) = x^2 J_{k-2}(x) + x(\nu - 2k + 1) J_{k-1}(x) + k(k+1)(1-\nu) J_k(x) \tag{3.19}$$

$$\tilde{I}_k^f(x) = x^2 I_{k-2}(x) + x(\nu - 2k + 1) I_{k-1}(x) + k(k+1)(1-\nu) I_k(x) \tag{3.20}$$

The eigenvalues ξ_{kn} are found as the $\tilde{n}$-th solution of

$$\begin{aligned}\tilde{I}_k^f(\xi)\left[\xi^3 J_{k-3}(\xi) + \xi^2(4-3k) J_{k-2}(\xi) + \xi k(k(1+\nu)-2) J_{k-1}(\xi)\right.\\ \left.+k^2(1-\nu)(1+k) J_k(\xi)\right] - \tilde{J}_k^f(\xi)\left[\xi^3 I_{k-3}(\xi) + \xi^2(4-3k) I_{k-2}(\xi)\right.\\ \left.+\xi k(k(1+\nu)-2) I_{k-1}(\xi) + k^2(1-\nu)(1+k) I_k(\xi)\right] = 0\end{aligned} \tag{3.21}$$

where k is the number of nodal diameters and n the number of nodal circles. Given that the edge is set in free condition, for $k = 1$, $n = \tilde{n}$, whereas for $k \neq 1$, $n = \tilde{n} - 1$. In addition, for every frequency associated to a mode with $k \neq 0$, i.e. a non-axisymmetric mode, there are two independent modes. These are known as the two preferential configurations, sharing the same radial dependence but differing in the position of nodal diameters. In perfect plates, the mode shapes are shifted $\pi/2$ rad and for this reason, the two configurations are denoted cos or sin.

Table 3.1 includes the data related to the first 20 modes of a plate with a free edge made of a material with Poisson ratio $\nu = 0.38$. Values are shown in dimensionless form.

Index	$\boldsymbol{\xi}$	$\boldsymbol{k}$	$\boldsymbol{n}$	**Conf.**	$\boldsymbol{\omega = \xi^2}$
1, 2	2.2568	2	0	cos / sin	5.0933
3	3.0290	0	1	cos	9.1751
4, 5	3.4493	3	0	cos / sin	11.8973
6, 7	4.5361	1	1	cos / sin	20.5762
8, 9	4.5798	4	0	cos / sin	20.9744
10, 11	5.6814	5	0	cos / sin	32.2777
12, 13	5.9341	2	1	cos / sin	35.2134
14	6.2138	0	2	cos	38.6118
15, 16	6.7654	6	0	cos / sin	45.7706
17, 18	7.2647	3	1	cos / sin	52.7766
19, 20	7.7419	1	2	cos / sin	59.9374

Table 3.1: Transverse modes and dimensionless frequencies for a plate with a free edge and $\nu = 0.38$

Analogously, for the in-plane family of modes, eq. (3.8a) is written in dimensionless form in terms of $\{\Psi_i(\mathbf{x})\}_{i\geq 1}$. Thus, Ψ must fulfill for all θ and t,

$$\boldsymbol{\Psi} = 0 \quad \text{at } r = 1, \tag{3.22}$$

$$\boldsymbol{\Psi}_{,r} = 0 \quad \text{at } r = 1, \tag{3.23}$$

$$\boldsymbol{\Psi}(r = 0) \quad \text{is bounded.} \tag{3.24}$$

Mode shapes can be written separately in terms of r and θ,

$$\boldsymbol{\Psi}_{0m}(r,\theta) = S_{0m}^f(r) \quad \text{for } l = 0, \tag{3.25}$$

$$\left.\begin{matrix}\boldsymbol{\Psi}_{lm1}(r,\theta)\\ \boldsymbol{\Psi}_{lm2}(r,\theta)\end{matrix}\right| = S_{lm}^f(r) \left|\begin{matrix}\cos l\theta\\ \sin l\theta\end{matrix}\right. \quad \text{for } l > 0, \tag{3.26}$$

with

$$S^f_{lm}(r) = \lambda_{lm}\left[J_l(\zeta_{lm}r) - \frac{J_l(\zeta_{lm})}{I_l(\zeta_{lm})} I_l(\zeta_{lm}r)\right] \tag{3.27}$$

and λ_{lm}, the normalization constant for modes Ψ that fulfills $\|\Psi\| = 1$. The values of ζ_{lm} are found as the m-th solution of

$$J_{l-1}(\zeta)I_l(\zeta) - I_{l-1}(\zeta)J_l(\zeta) = 0. \tag{3.28}$$

Unlike the transverse modes, in this case, l and m correspond to the numbers of nodal diameters and circles respectively. Some dimensionless values are included in table 3.2.

Index	ξ	k	n	**Conf.**	$\omega = \xi^2$
1	3.1962	0	1	cos	10.2158
2, 3	4.6109	1	0	cos / sin	21.2604
4, 5	5.9057	2	0	cos / sin	34.8770
6	6.3064	0	2	cos	39.7711
7, 8	7.1435	3	0	cos / sin	51.0300
9, 10	7.7993	1	1	cos / sin	60.8287
11, 12	8.3466	4	0	cos / sin	69.6658
13, 14	9.1969	2	1	cos / sin	84.5826
15	9.4395	0	3	cos	89.1041
16, 17	9.5257	5	0	cos / sin	90.7390
18, 19	10.5367	3	1	cos / sin	111.0214
20	10.6870	6	0	cos	114.2125

Table 3.2: In-plane modes and dimensionless frequencies for a plate with free edge

$[i, j, k]$	H^i_{jk}
$[1,1,1]$	-19.8615
$[2,4,1]$	-26.0895
$[1,4,2]$	0
$[2,1,4]$	-26.0895
$[1,3,3]$	42.1456
$[3,8,5]$	76.4045
$[2,9,5]$	-76.4045
$[10,10,10]$	0

Table 3.3: Some H coefficients for a plate with a free edge and Poisson ratio $\nu = 0.38$

3.3.2 Circular plate with clamped edge

The equations that must be fulfilled for a plate with a clamped edge are eqs. (3.10a) and (3.10b). The equations for the transverse and rotational displacement are equivalent to eq. (3.8a) of the free-edge plate. Therefore, the mode shapes in the transverse direction of the clamped plate are analogous to eqs. (3.26) and (3.27). They are rewritten here for the sake of completeness.

$[p,q,r,s]$	Γ^p_{qrs}
$[1,1,1,1]$	1.8983
$[2,2,2,2]$	1.8983
$[3,3,3,3]$	8.5747
$[1,2,3,4]$	0
$[4,3,2,1]$	0
$[1,1,2,2]$	$3.9121e-06$
$[1,2,2,1]$	1.8983
$[1,2,1,2]$	$3.9121e-06$

Table 3.4: Some Γ coefficients for a plate with a free edge and Poisson ratio $\nu = 0.38$

$$\boldsymbol{\Phi}_{0n}(r,\theta) = R^c_{0n}(r) \quad \text{for } k = 0, \tag{3.29}$$

$$\left.\begin{matrix}\boldsymbol{\Phi}_{kn1}(r,\theta)\\ \boldsymbol{\Phi}_{kn2}(r,\theta)\end{matrix}\right| = R^c_{kn}(r) \left|\begin{matrix}\cos k\theta\\ \sin k\theta\end{matrix}\right. \quad \text{for } k > 0, \tag{3.30}$$

$$R^c_{kn}(r) = \lambda_{kn}\left[J_k(\xi_{kn}r) - \frac{J_k(\xi_{kn})}{I_k(\xi_{kn})} I_k(\xi_{kn}r)\right] \tag{3.31}$$

$$J_{k-1}(\xi)I_k(\xi) - I_{k-1}(\xi)J_k(\xi) = 0 \tag{3.32}$$

On the other hand, the equation for the in-plane direction eq. (3.10a) is similar to eq. (3.8c) but they differ in the sign previous to ν. Thus, mode shapes must be recomputed. For all θ and t,

$$\boldsymbol{\Psi}_{,rr} + \nu\boldsymbol{\Psi}_{,r} + \nu\boldsymbol{\Psi}_{,\theta\theta} = 0 \quad \text{at } r = 1, \tag{3.33}$$

$$\boldsymbol{\Psi}_{,rrr} + \boldsymbol{\Psi}_{,rr} - \boldsymbol{\Psi}_{,r} + (2+\nu)\boldsymbol{\Psi}_{,r\theta\theta} - (3+\nu)\boldsymbol{\Psi}_{,\theta\theta} = 0 \quad \text{at } r = 1, \tag{3.34}$$

$$\boldsymbol{\Psi}(r=0) \quad \text{is bounded.} \tag{3.35}$$

The solutions of this system of equations are

$$\boldsymbol{\Psi}_{0n}(r,\theta) = S^c_{0n}(r) \quad \text{for } l = 0, \tag{3.36}$$

$$\left.\begin{matrix}\boldsymbol{\Psi}_{lm1}(r,\theta)\\ \boldsymbol{\Psi}_{lm2}(r,\theta)\end{matrix}\right| = S^c_{lm}(r) \left|\begin{matrix}\cos k\theta\\ \sin k\theta\end{matrix}\right. \quad \text{for } l > 0, \tag{3.37}$$

where

$$S^c_{lm}(r) = \kappa_{lm}\left[J_l(\zeta_{lm}r) - \frac{\tilde{J}^c_l(\zeta_{lm})}{\tilde{I}^c_l(\zeta_{lm})} I_l(\zeta_{lm}r)\right] \tag{3.38}$$

and terms $\tilde{J}^c_l(x)$ and $\tilde{I}^c_l(x)$ are computed as follows,

$$\tilde{J}^c_l(x) = x^2 J_{l-2}(x) + x(-\nu - 2l + 1)J_{l-1}(x) + (l(l+1) + \nu l(1-l))J_l(x) \tag{3.39}$$

$$\tilde{I}^c_l(x) = x^2 I_{l-2}(x) + x(-\nu - 2l + 1)I_{l-1}(x) + (l(l+1) + \nu l(1-l))I_l(x) \tag{3.40}$$

The eigenvalues ζ_{lm} are found as the $\tilde{m}$-th solution of

$$\tilde{I}_l^c(\zeta)\left[\zeta^3 J_{l-3}(\zeta)+\zeta^2(4-3l)J_{l-2}(\zeta)+\zeta l(l(1-\nu)-2)J_{l-1}(\zeta)\right.$$
$$\left.+l^2(1+l)(1+\nu)J_l(\zeta)\right]-\tilde{J}_l^c(\zeta)\left[\zeta^3 I_{l-3}(\zeta)+\zeta^2(4-3l)I_{l-2}(\zeta)\right.$$
$$\left.+\zeta l(l(1-\nu)-2)I_{l-1}(\zeta)+l^2(1+l)(1+\nu)I_l(\zeta)\right]=0 \quad (3.41)$$

First 20 values are displayed in table 3.5.

Index	ξ	k	n	Conf.	$\omega=\xi^2$
1, 2	2.3032	2	1	cos / sin	5.3047
3	2.6251	0	1	cos	6.8912
4, 5	3.6549	3	1	cos / sin	13.3584
6, 7	4.3582	1	1	cos / sin	18.9938
8, 9	4.8712	4	1	cos / sin	23.7286
10, 11	5.8741	2	2	cos / sin	34.5054
12, 13	6.0358	5	1	cos / sin	36.4304
14	6.0658	0	2	cos	36.7944
15, 16	7.1715	6	1	cos / sin	51.4300
17, 18	7.2814	3	2	cos / sin	53.0190
19, 20	7.6418	1	2	cos / sin	58.3975

Table 3.5: In-plane modes and dimensionless frequencies for a plate with clamped edge

$[i,j,k]$	H^i_{jk}
$[1,1,1]$	0
$[1,4,1]$	-12.0352
$[1,1,4]$	-12.0352
$[10,7,2]$	206.3190
$[10,2,7]$	206.3190
$[1,10,10]$	-79.2939
$[2,9,10]$	79.2939
$[10,10,10]$	-420.4716

Table 3.6: Some H coefficients for a plate with a clamped edge and Poisson ratio $\nu=0.38$

3.3.3 Circular plate with elastic edge in the transverse direction and free edge in the in-plane direction

In this case, elastic edge is set for the transverse direction combined with the free edge boundary conditions in the in-plane direction. **????** are rewritten in terms of $\{\Phi_i(\mathbf{x})\}_{i\geq 1}$. For all θ and t,

$$\boldsymbol{\Phi}_{,rr}+\nu\boldsymbol{\Phi}_{,r}+\nu\boldsymbol{\Phi}_{,\theta\theta}+\frac{K_R}{D}\boldsymbol{\Phi}_{,r}=0 \quad \text{at } r=1, \quad (3.42)$$

$$\boldsymbol{\Phi}_{,rrr}+\boldsymbol{\Phi}_{,rr}-\boldsymbol{\Phi}_{,r}+(2-\nu)\boldsymbol{\Phi}_{,r\theta\theta}-(3-\nu)\boldsymbol{\Phi}_{,\theta\theta}-\frac{K_T}{D}\boldsymbol{\Phi}=0 \quad \text{at } r=1, \quad (3.43)$$

$$\boldsymbol{\Phi}(r=0) \quad \text{is bounded.} \quad (3.44)$$

$[p,q,r,s]$	Γ^p_{qrs}
$[1,1,1,1]$	9.0374
$[2,2,2,2]$	64.4130
$[3,3,3,3]$	64.4130
$[1,2,3,4]$	0
$[4,3,2,1]$	0
$[1,1,2,2]$	4.3122
$[1,2,2,1]$	20.5191
$[1,2,1,2]$	4.3122

Table 3.7: Some Γ coefficients for a plate with a clamped edge and Poisson ratio $\nu = 0.38$

Solutions can be separated in r and θ, so that

$$\boldsymbol{\Phi}_{0n}(r,\theta) = R^e_{0n}(r) \quad \text{for } k = 0, \tag{3.45}$$

$$\left.\begin{matrix}\boldsymbol{\Phi}_{kn1}(r,\theta)\\ \boldsymbol{\Phi}_{kn2}(r,\theta)\end{matrix}\right| = R^e_{kn}(r) \left|\begin{matrix}\cos k\theta\\ \sin k\theta\end{matrix}\right. \quad \text{for } k > 0, \tag{3.46}$$

where

$$R^e_{kn}(r) = \kappa_{kn}\left[J_k(\xi_{kn}r) - \frac{\tilde{J}^e_k(\xi_{kn})}{\tilde{I}^e_k(\xi_{kn})} I_k(\xi_{kn}r)\right] \tag{3.47}$$

with $\tilde{J}^e_k(x)$ and $\tilde{I}^e_k(x)$ computed as,

$$\tilde{J}^e_k(x) = x^2 J_{k-2}(x) + x(\nu - 2k + 1) J_{k-1}(x) + \left(k(k+1)(1-\nu) - \frac{K_R}{D}k\right) J_k(x) \tag{3.48}$$

$$\tilde{I}^e_k(x) = x^2 I_{k-2}(x) + x(\nu - 2k + 1) I_{k-1}(x) + \left(k(k+1)(1-\nu) - \frac{K_R}{D}k\right) I_k(x) \tag{3.49}$$

The eigenvalues ξ_{kn} are found as the $\tilde{n}$-th solution of

$$\begin{aligned}&\tilde{I}^e_k(\xi)\left[\xi^3 J_{k-3}(\xi) + \xi^2(4-3k) J_{k-2}(\xi) + \xi k(k(1+\nu)-2) J_{k-1}(\xi)\right.\\ &\left. + \left(k^2(1-\nu)(1+k) - \frac{K_T}{D}\right) J_k(\xi)\right] - \tilde{J}^e_k(\xi)\left[\xi^3 I_{k-3}(\xi) + \xi^2(4-3k) I_{k-2}(\xi)\right.\\ &\left. + \xi k(k(1+\nu)-2) I_{k-1}(\xi) + \left(k^2(1-\nu)(1+k) - \frac{K_T}{D}\right) I_k(\xi)\right] = 0\end{aligned} \tag{3.50}$$

Similarly to the free case, the number of nodal circles n corresponds to $n = \tilde{n}$ for $k = 1$ and to $n = \tilde{n} - 1$ for $k \neq 1$.

Note that when $K_R = K_T = 0$, the above equations correspond to eqs. (3.18) to (3.21) for a free edge. Analogously, when $K_R = K_T = \infty$, **????** can be rewritten to obtain the equation for the clamped case.

In order to generalize results, the rotational K_R and transverse K_T stiffness can be normalized in terms of the flexural rigidity D, such that

$$KR = \frac{K_R}{D}, \tag{3.51a}$$

$$KT = \frac{K_T}{D}. \tag{3.51b}$$

Table 3.8 contains the first eigenmodes and dimensionless frequencies for a plate with Poisson ratio $\nu = 0.38$ and normalized stiffnesses $KR = 10$ and $KT = 1000$.

Index	ξ	k	n	**Conf.**	$\omega = \xi^2$
1	2.7477	0	1	cos	7.5498
2, 3	4.0176	1	1	cos / sin	16.1412
4, 5	5.2448	2	1	cos / sin	27.5075
6	5.6130	0	2	cos	31.5062
7, 8	6.4487	3	1	cos / sin	41.5851
9, 10	7.0689	1	2	cos / sin	49.9688
11, 12	7.6335	4	1	cos / sin	58.2707
13, 14	8.4501	2	2	cos / sin	71.4042
15	8.6829	0	3	cos	75.3928
16, 17	8.8024	5	1	cos / sin	77.4822
18, 19	9.7822	3	2	cos / sin	95.6908
20	9.9578	6	1	cos	99.1583

Table 3.8: Transverse modes and dimensionless frequencies for a plate with an elastic edge, $KR = 10$, $KT = 1000$ and $\nu = 0.38$

$[i, j, k]$	H^i_{jk}
$[1,1,1]$	26.9817
$[5,3,2]$	126.2545
$[5,2,3]$	126.2545
$[8,8,6]$	-451.6667
$[8,6,8]$	-451.6667
$[6,8,8]$	434.6719
$[8,1,10]$	0
$[1,10,8]$	0

Table 3.9: Some H coefficients for a plate with an elastic edge, distributed rotational stiffness $KR = 10$, distributed translational stiffness $KT = 1000$ and Poisson ratio $\nu = 0.38$

$[p,q,r,s]$	Γ^p_{qrs}
$[1,1,1,1]$	3.6068
$[2,2,2,2]$	19.7141
$[3,3,3,3]$	19.7141
$[1,2,3,4]$	0
$[4,3,2,1]$	0
$[7,8,9,10]$	-7.9968
$[7,10,8,9]$	7.9968
$[7,9,8,10]$	-7.9968

Table 3.10: Some $\boldsymbol{G}$ coefficients for a plate with an elastic edge, distributed rotational stiffness $KR = 10$, distributed translational stiffness $KT = 1000$ and Poisson ratio $\nu = 0.38$

4. Rectangular plates

Rectangular plates are defined by the length of their sides L_x and L_y.

4.1 Boundary conditions

This section summarizes the equations for the boundary conditions of the rectangular plate. For further details refer to [25].

Let $(\mathbf{n}, \boldsymbol{\tau})$ be respectively the normal and tangent unitary vectors with respect to the plate boundary and s the coordinate along the edge ($s = x, y$ depending on the edge considered).

The only case treated in VK-Gong is the simply supported case, for which the plate is in-plane free ($N_{nn} = N_{n\tau} = 0$), the transverse displacement is blocked ($w = 0$) and the normal angle is free to move ($M_{nn} = 0$). This leads to the following boundary conditions, in any of the four edges of the plate:

$$F = \frac{\partial F}{\partial n} = 0, \tag{4.1a}$$

$$w = 0, \tag{4.1b}$$

$$\frac{\partial^2 w}{\partial n^2} + \nu \frac{\partial^2 w}{\partial s^2} = 0. \tag{4.1c}$$

Notice that in this case, the nonlinear term $\mathcal{N}_F(\partial w/\partial r)$ in Eq. (3.7b) is zero because of the zero curvature of the edge (Eq. (31) of [25]) so that the boundary conditions are fully linear.

4.2 Mode families and modal coupling coefficients for some combinations of boundary conditions

This section revisits the computation of the mode families for the boundary conditions included in the code.

4.2.1 Simply supported edge

A simply supported plate is characterized by being fixed in the direction of support and free in the others. Although some literature also includes the fixed edge for the in-plane direction [1]. Thus, the equations that must be fulfilled are **??????**.

For the transverse direction, the family of mode that solves the problem in eq. (2.7a), is [9, 10],

$$\Phi_k(\mathbf{x}) = \sin\frac{k_x\pi x}{L_x}\sin\frac{k_y\pi y}{L_y} \tag{4.2}$$

where k_x and k_y are integers and correspond to the number of nodal lines in the mode shape. The angular eigenfrequency ω_k is obtained with

$$\omega_k^2 = \frac{D}{\rho h}\left[\left(\frac{k_x\pi}{L_x}\right)^2 + \left(\frac{k_y\pi}{L_y}\right)^2\right]^2. \tag{4.3}$$

On the in-plane direction, in order to simplify the computations, the boundary conditions are restricted to

$$F = F_{,n} = 0 \;\; \forall \mathbf{x} \in \delta S. \tag{4.4}$$

This way, the problem becomes equivalent to the clamped edge case in the transverse direction and thus, the Rayleigh-Ritz method can be used to solve it [10]. The mentioned procedure consists on transforming the continuous eigenvalue problem in eq. (2.7b) into the discrete domain so that it can be solved by a common eigenvalue routine. For that, a generic eigenmode $\Psi_k(\mathbf{x})$ is rewritten as a series expansion

$$\Psi_k(\mathbf{x}) = \sum_{n=1}^{N_\Lambda} a_n \Lambda_n(\mathbf{x}), \tag{4.5}$$

where a_n are unknown weighting coefficients and Λ_n is a set of functions specifically chosen for the problem. In this case,

$$\Lambda_n = X_{n_1}(x)Y_{n_2}(y), \tag{4.6}$$

with

$$X_{n_1}(x) = \cos\left(\frac{n_1\pi x}{L_x}\right) + \frac{15(1+(-1)^{n_1})}{L_x^4}x^4 - \frac{4(8+7(-1)^{n_1})}{L_x^3}x^3 + \frac{6(3+2(-1)^{n_1})}{L_x^2}x^2 - 1, \tag{4.7}$$

and analogous for $Y_{n_2}(y)$.

The values of a_n along with the eigenvalues ζ_k^4 are found by solving the eigenproblem[10, 12]

$$\mathbf{Ka} = \zeta^4\mathbf{Ma}, \tag{4.8}$$

where $\mathbf{K}$ and $\mathbf{M}$ are respectively the stiffness and mass matrices with dimensions $\Lambda_n \times \Lambda_n$. They are defined as

$$\begin{aligned} K(i,j) = K(mn,pq) = &\int_0^{Lx} X_m''(x)X_p''(x)\mathrm{d}x \int_0^{Ly} Y_n(y)Y_q(y)\mathrm{d}y \\ + &\int_0^{Lx} X_m(x)X_p(x)\mathrm{d}x \int_0^{Ly} Y_n''(y)Y_q''(y)\mathrm{d}y + 2\int_0^{Lx} X_m'(x)X_p'(x)\mathrm{d}x \int_0^{Ly} Y_n'(y)Y_q'(y)\mathrm{d}y \end{aligned} \tag{4.9}$$

and

$$M(i,j) = M(mn,pq) = \int_0^{Lx} X_m(x)X_p(x)\mathrm{d}x \int_0^{Ly} Y_n(y)Y_q(y)\mathrm{d}y. \tag{4.10}$$

The integrals in eqs. (4.9) and (4.10) can be calculated analytically, with

$$\int_0^{Lx} X_m''(x)X_p''(x)\mathrm{d}x = \begin{cases} 720/L_x^3 & \text{if } m=p=0 \\ \left(\pi^4 m^4 - 672(-1)^m - 768\right)/\left(2L_x^3\right) & \text{if } m=p\neq 0 \\ 0 & \textbf{if } m \textbf{ or } p=0 \text{ and } m\neq p \\ -24\left(7(-1)^m + 7(-1)^p + 8(-1)^m(-1)^p + 8\right)/L_x^3 & \text{otherwise} \end{cases} \tag{4.11}$$

$$\int_0^{Lx} X_m(x)X_p(x)\mathrm{d}x = \begin{cases} 10L_x/7 & \text{if } m=p=0 \\ 67L_x/70 - (-1)^m L_x/35 - 768L_x/\left(\pi^4 m^4\right) - 672(-1)^m L_x/\left(\pi^4 m^4\right) & \text{if } m=p\neq 0 \\ 3L_x\left((-1)^p+1\right)\left(\pi^4 p^4 - 1680\right)/\left(14\pi^4 p^4\right) & \text{if } m=0 \text{ and } p\neq 0 \\ 3L_x\left((-1)^m+1\right)\left(\pi^4 m^4 - 1680\right)/\left(14\pi^4 m^4\right) & \text{if } p=0 \text{ and } m\neq 0 \\ -\left(L_x\left(11760(-1)^m + 11760(-1)^p - 16\pi^4 m^4 + 13440(-1)^m(-1)^p + \right.\right. & \\ \left.\left.(-1)^m\pi^4 m^4 + (-1)^p\pi^4 m^4 - 16(-1)^m(-1)^p\pi^4 m^4 + 13440\right)\right)/\left(70\pi^4 m^4\right) & \\ -\left(L_x\left(13440m^4 + 11760(-1)^m m^4 + 11760(-1)^p m^4 \right.\right. & \\ \left.\left. +13440(-1)^m(-1)^p m^4\right)\right)/\left(70\pi^4 m^4 p^4\right) & \text{otherwise} \end{cases} \tag{4.12}$$

$$\int_0^{Lx} X_m''(x)X_p(x)\mathrm{d}x = \begin{cases} -120/(7Lx) & \text{if } m=p=0 \\ -\left(768\pi^2 m^2 - 47040(-1)^m + 35\pi^4 m^4 + 432(-1)^m\pi^2 m^2 \right. & \\ \left. -53760\right)/\left(70L_x\pi^2 m^2\right) & \text{if } m=p\neq 0 \\ -\left(60\left((-1)^p+1\right)\left(\pi^2 p^2 - 42\right)\right)/\left(7L_x\pi^4 p^4\right) & \text{if } m=0 \text{ and } p\neq 0 \\ -\left(60\left((-1)^m+1\right)\left(\pi^2 m^2 - 42\right)\right)/\left(7L_x\pi^4 m^4\right) & \text{if } p=0 \text{ and } m\neq 0 \\ \left(24\left(m^2+p^2\right)\left(7(-1)^m + 7(-1)^p + 8(-1)^m(-1)^p + 8\right)\right)/\left(L_x\pi^2 m^2 p^2\right) & \\ -\left(\left(108(-1)^m + 108(-1)^p + 192(-1)^m(-1)^p + 192\right)\right)/\left(35L_x\right) & \text{otherwise} \end{cases} \tag{4.13}$$

and similarly for the integrals involving the functions $Y(x)$.

5. Time integration schemes

The use of a modal approach as shown in chapter 2, provides a spatial discretization of the von Kármán equations and reduces the problem to a system of coupled Ordinary Differential Equations which are time dependent.

This chapter presents the two methods that are implemented in VK-Gong. The first one respects the conservation of energy whereas the second does not but implies lower computational cost. Prior to that, the operators used in their developing are introduced.

5.1 Operators

The following time schemes make use of a set of operators acting on the state vector $\mathbf{q}(n)$ at time step n [3, 12].

The backward e_{t-} and forward e_{t+} shift operators are

$$e_{t-}\mathbf{q}(n) = \mathbf{q}(n-1), \quad e_{t+}\mathbf{q}(n) = \mathbf{q}(n+1). \tag{5.1}$$

Backward δ_{t-}, centered $\delta_{t\cdot}$ and forward δ_{t+} approximations to first order time derivatives,

$$\delta_{t-} \equiv \frac{1}{k}(1-e_{t-}), \quad \delta_{t\cdot} \equiv \frac{1}{2k}(e_{t+}-e_{t-}), \quad \delta_{t+} \equiv \frac{1}{k}(e_{t+}-1), \tag{5.2}$$

where $k = 1/f_s$ is the time step corresponding to the sampling frequency f_s. They can be combined to obtain an approximation to the second order derivative δ_{tt},

$$\delta_{tt} \equiv \delta_{t+}\delta_{t-} = \frac{1}{k^2}(e_{t+}-2+e_{t-}). \tag{5.3}$$

Finally, the backward μ_{t-}, centered $\mu_{t\cdot}$ and forward μ_{t+} averaging operators are also introduced as

$$\mu_{t-} \equiv \frac{1}{2}(1+e_{t-}), \quad \mu_{t\cdot} \equiv \frac{1}{2}(e_{t+}+e_{t-}), \quad \mu_{t+} \equiv \frac{1}{2}(e_{t+}+1). \tag{5.4}$$

5.2 Energy conserving scheme

This section presents a en energy conserving scheme. This finite difference approach was first developed in [3] to fully solve the von Kármán equations and later applied to the modal approach for perfect and imperfect plates in [12]. Departing from eq. (2.11), it is built as

$$\delta_{tt} q_s(n) + \omega_s^2 q_s(n) = \varepsilon \sum_k^{N_\Phi} \sum_l^{N_\Psi} E_{kl}^s \left(q_k(n) + a_k\right) \mu_t . \eta_l - 2c_s \omega_s \delta_t . q_s(n) + p_s(n) \tag{5.5a}$$

$$\mu_{t-} \eta_l(n) = -\frac{1}{2\zeta_l^4} \sum_{i,j=1}^{N_\Phi} H_{i,j}^l \left(q_i(n) e_{t-} q_j(n) + 2a_j \mu_{t-} q_i(n)\right) \tag{5.5b}$$

where the definition of ε depends on the shape of the plate. For circular plates, the involved variables are dimensionless and

$$\varepsilon_c = 12(1-\nu^2), \tag{5.6}$$

with ν standing for the Poisson ratio.

For rectangular plates, the variables in eq. (5.5) are dimensioned and

$$\varepsilon_r = -\frac{E S_w^2}{\rho}, \tag{5.7}$$

where E is the Young modulus, ρ the volumetric mass density and S_W is the constant of normalization of the transverse vectors.

Replacing the operators by their definitions, the equations to solve for every mode s at every time step n,

$$\begin{aligned}
&\left(\frac{1}{k^2} + \frac{C_{ss}}{2k}\right) q_s(n+1) + \varepsilon \sum_{i,j,k}^{N_\Phi} \sum_l^{N_\Psi} \frac{E_{kl}^s H_{i,j}^l}{2\zeta_l^4} q_i(n+1) \left(q_j(n) + a_j\right) \left(q_k(n) + a_k\right) = \\
&-\varepsilon \sum_{i,j,k}^{N_\Phi} \sum_l^{N_\Psi} \frac{E_{kl}^s H_{i,j}^l}{2\zeta_l^4} q_i(n) a_j \left(q_k(n) + a_k\right) + \left(\frac{2}{k^2} - K_{ss}\right) q_s(n) + \\
&\left(\frac{C_{ss}}{2k} - \frac{1}{k^2}\right) q_s(n-1) + \varepsilon \sum_{i,j,k}^{N_\Phi} \sum_l^{N_\Psi} E_{kl}^s \left(q_k(n) + a_k\right) \left(\frac{\eta_l(n-1) - \eta_l(n)}{2}\right) + p_s(n)
\end{aligned} \tag{5.8}$$

$$\eta_l(n+1) = -\eta_l(n) - \sum_{i,j}^{N_\Phi} \frac{H_{ij}^l}{\zeta_l^4} \left[q_i(n+1) q_j(n) + a_j \left(q(n+1) + q(n)\right)\right] \tag{5.9}$$

where

$$C_{ss} = 2c_s \omega_s, \tag{5.10}$$

and

$$K_{ss} = \omega_s^2. \tag{5.11}$$

Note that this is an implicit scheme and every iteration requires a matrix inversion to update the value of the state vector $\boldsymbol{q}(t)$. Although this may increase the computational cost with respect to the next method, the fact that this scheme is energy conserving provides higher accuracy.

R This scheme proves to be energy conservative and stable as long as

$$f_s \geq \pi f_{N_\Phi}. \tag{5.12}$$

Where $f_s = 1/k$ is the sampling rate and f_{N_Φ} is the largest eigenfrequency retained in the truncation for transverse motions [10].

5.3 Störmer-Verlet scheme

The use of the Störmer-Verlet scheme appears as a less time consuming alternative to solve the von Kármán equations [3, 9, 12].

Using the definitions of ε in eqs. (5.6) and (5.7), it is built over eq. (2.13) as

$$\left(\frac{1}{k^2}+\frac{C_{ss}}{2k}\right)q_s(n+1) = \left(\frac{2}{k^2}-K_{ss}\right)q_s(n)+\left(\frac{C_{ss}}{2k}-\frac{1}{k^2}\right)q_s(n-1) - \varepsilon'\sum_{i,j,k}^{N_\Phi}\sum_{l}^{N_\Psi}\frac{E^s_{kl}H^l_{i,j}}{2\zeta_l^4}\left(q_k(n)+a_k\right)q_i(n)\left(q_j(n)+2a_j\right)+p_s(n). \tag{5.13}$$

The same stability condition defined in eq. (5.12) holds for this integration scheme. In this case, the expression is explicit which in terms of computational cost and memory usage is more advantageous.

III

User's manual

6. Matlab code

6.1 Installation and general description

In order to use the Matlab version of the code, "VK-Gong" folder and its subfolders should be added to the Matlab search path. In addition, for simplicity, the *Current folder* should be changed to "VK-Gong". To that end, user shall use the following command lines,

```
addpath(genpath('<VK-Gong Path>/VK-Gong'));
cd('<VK-Gong Path>/VK-Gong');
```

"VK-Gong" is divided in two directories, one for the code and the other for the parameters. In its turn, the code is structured in seven groups, classifying the functions according to their use,

Linear characteristics functions contains the functions related to the computation of eigenmodes and eigenfrequencies of the perfect plate, in both transverse and in-plane directions. It also includes a function to display the transverse eigenfrequencies of any plate, either perfect or imperfect.

Nonlinear characteristics functions includes the routines that compute the $\boldsymbol{H}$ and $\boldsymbol{\Gamma}$ modal coupling coefficients.

Imperfection functions contains the functions devoted to compute the imperfection profile and the projection coefficients used in the time integration process.

Excitation and damping function includes the functions that generate the damping and excitation vectors.

Time integration functions contains the functions related to the time simulation.

Parsers contains the functions that load and read the input parameter files and create all the variables necessary for the time simulation.

Output plot functions includes the functions used to display the simulation results such as the displacement time signal, with the corresponding fast Fourier transform or spectrogram.

On the other hand, the parameters folder contains the preset variables and also stores the newly computed ones.

H files Containing the H and Γ coefficients for every combination of parameters.

Mode files Containing the transverse and in-plane mode files as explained below.
Input files Containing the files with the input parameters for the simulation.

Note that each of the previous folders organize their contents in three subdirectories: Circular, Rectangular and Common, in order to distinguish the applicability of every function or file.

6.2 How to use the program

The final purpose of this software is the time simulation of the nonlinear response of thin plates. To that end, the code should be executed in two phases. First, the calculation of all the necessary parameters and second, the time integration itself. The *main* scripts provided with the code follow this process and give a black box alternative where only the proper introduction of the input parameters is required.

The program execution is performed using the appropriate main file, "mainCircular.m" for circular plates or "mainRectangular.m" for rectangular. This script is headed by the input parameter file names and should be modified by user if necessary. When the specified files are not found, the preset values will be loaded. The contents of these files are described in Section 6.10.

```
%% Input parameters files
PlateCharacteristicsFileName = 'PlateCharacteristics.mat'; % Physical characteristics of the plate: Dimensions, imperfection profile, material and boundary conditions.
SimulationParametersFileName = 'SimulationParameters.mat'; % Parameters related to the simulation: Time length, scheme, number of modes, output points, accuracy.
GammaFileName = 'GammaCircular.mat'; % Name of the file containing the Gamma Tensor.
ScoreFileName = 'ScoreParameters.mat'; % Characteristics of the excitation.
OutputFileName = 'Results'; % Name of the results files and folder.
...
```

Next, the script calls the parser functions *plate_def* and *score* in charge of reading the input files and preparing the variables for the time simulation which is performed in the following step. Finally, results are saved as audio and / or binary files.

Nevertheless, if the user is interested in the intermediate results such as the plate eigenfrequencies or nonlinear coupling coefficients, the preliminary functions can be executed independently as described in the following sections.

6.3 Linear characteristics functions

6.3.1 Circular

ComputeTransverseEigenfrequenciesCircular.m

This function is used to obtain the eigenfrequencies of the transverse vibration modes of a perfect plate. Depending on the boundary conditions of the problem, a different eigenproblem is considered. When `BC = 'clamped'`, the function solves eq. (3.32). For `BC = {'free','elastic'}`, eq. (3.50) is used, setting `KR = 0` and `KT = 0` for the free case.

The routine calculates the roots ξ_i of the aforementioned equations up to the value introduced in *xmax*. The precision of the solutions is tuned by means of the subinterval size `dx`. Note that these variables must be introduced in dimensionless form. Typically, `dx = 1e-3` provides enough accuracy for the later performance of the code. On the other hand, `xmax` should be set large enough to obtain the desired number of eigenfrequencies. As a reference, when `dx = 1e-3`, `xmax = 100` and `BC = 'free'`, 2579 frequencies are obtained.

In order to find the zeros, function `FindZeros.m` is used. Given a function f evaluated at every value of x, it finds the values of x where f is null.

Results are shown in two output variables. Matrix *Zeros* contains the values of ξ_i sorted in ascending order so that every row corresponds to the number of nodal diameters k and every column to a number of nodal circles n_i. Note that in case of clamped edge the minimum number of nodal circles is 1 so $n \geq 1$; whereas in case of free or elastic edge, $n \geq 0$. This matrix is no longer necessary in the code but might be used for checking the function results.

In vector `mode_t`, modes are listed in ascending order according to its eigenfrequency. Every row contains the index i of the mode, the position of the root ξ_i, the number of nodal diameters k_i, the number of nodal circles n_i, the configuration of the mode, $c_i = 1$ for cos and $c_i = 2$ for sin, and finally, the eigenfrequency $\omega_i = \xi_i^2$. *mode_t* is built by `SortZeros.m` and saved in a binary file in The output is saved and located in *'./Parameters/Mode files/Circular/'*.

Function 6.3.1 — ComputeTransverseEigenfrequenciesCircular.m.

Short description

Computation of transverse eigenvalues and dimensionless eigenfrequencies.

Call

[mode_t, Zeros] = ComputeTransverseEigenfrequenciesCircular(dx, xmax, BC, nu, KR, KT)

Input parameters

dx Accuracy step.

xmax Top boundary value for x.

BC Type of boundary conditions at the edge. Possible values: 'free', 'clamped', 'elastic'.

nu Poisson ratio.

KR Rotational stiffness normalized with respect to bending stiffness, $KR = Kr/D$. Only used when BC = 'elastic'.

KT transverse stiffness normalized with respect to bending stiffness, $KT = Kt/D$.. Only used when BC = 'elastic'.

Output parameters

mode_t Vector containing the information corresponding to the transverse vibration modes with eigenfrequencies up to xmax. For every mode,

Column 1	Column 2	Column 3	Column 4	Column 5	Column 6
$< i >$	$< \xi_i >$	$< k_i >$	$< n_i >$	$< c_i >$	$< \omega_i >$

Zeros Matrix containing the zeros found up to xmax, i.e. ξ_i, sorted in ascending order. Every row corresponding to a value of k and every column to a value of n.

Function 6.3.2 — FindZeros.m.

Short description

Localization of roots of a given function.

Call

[zer] = FindZeros(x, f)

Input parameters

x Vector of abscissa points.

f Function to be evaluated

Output parameters

zer Vector containing the values of x where f is null

Function 6.3.3 — SortZeros.m.

Short description

Building of vector TAB containing the sorted list of eigenfrequencies with their corresponding number of nodal diameters and circles.

Call

[TAB] = SortZeros(Zeros)

Input parameters

Zeros Matrix containing zeros of a certain function, sorted in ascending order. Every row corresponds to a value of k and every column to a value of n.

Output parameters

TAB Table that contains the values of matrix Zeros sorted in ascending order with the following additional information,

Column 1	Column 2	Column 3	Column 4	Column 5	Column 6
$< i >$	$< x_i >$	$< k_i >$	$< n_i >$	$< c_i >$	$< x_i^2 >$

ComputeInplaneEigenfrequenciesCircular.m

This function is analogue to *ComputeTransverseEigenfrequenciesCircular.m* for the in-plane modes of vibration of a perfect plate. In this case, eq. (3.28) is considered for *BC = {'free','elastic'}* and eq. (3.41) for *BC = {'clamped'}*. The recommended input values are the same than for the previous function, so `dx = 1e-3` and `xmax = 100` will provide enough results for typical computations.

Function 6.3.4 — ComputeInplaneEigenfrequenciesCircular.m.

Short description

Computation of in-plane eigenvalues and dimensionless eigenfrequencies.

Call

[mode_l, Zeros] = ComputeInplaneEigenfrequenciesCircular(dx, xmax, BC, nu)

Input parameters

dx Discretization step.

xmax Top boundary value for x.
BC Type of boundary conditions at the edge. Possible values: 'free', 'clamped', 'elastic'.
nu Poisson ratio.

Output parameters

mode_l Vector containing the information corresponding to the longitudinal vibration modes with eigenfrequencies up to xmax. For every mode,

Column 1	Column 2	Column 3	Column 4	Column 5	Column 6
$< i >$	$< \zeta_i >$	$< k_i >$	$< n_i >$	$< c_i >$	$< \omega_i >$

Zeros Matrix containing the zeros found up to xmax sorted in ascending order. Every row corresponds to a value of k and every column to a value of n.

ModeShapeCircular.m

This function is used to compute the shape of a single transverse vibration mode as a 3D surface. In this way, for every pair of coordinates `[U, V]` the value of `W` is calculated.

It is used by the main code to compute the projection coefficients between the imperfection and the transverse modal basis. However, it may also be used independently to plot the modeshape of a given mode by means of the Matlab function `surf(U, V, W)`.

Input variables `k`, `c` and `xkn` can be obtained by executing `ComputeTransverseEigenfrequenciesCircular.m` and retrieving the contents of `mode_t`.

Next variables, radius `R` and Poisson ratio `nu` depend on the plate characteristics. Note that if `xkn` is introduced in dimensionless form, `R` should be set to unity. On the other hand, `KR` corresponds to the normalized rotational stiffness of the edge. In case that the boundary conditions do not correspond to elastic edge, `KR` should be set to 0 or infinity as explained in the table below.

Finally, `kmax` and `nmax` specify the output resolution, i.e. the number of points in every direction. Note that `kmax`$\gg$`k` and `nmax`$\gg n$ to guarantee that all the variations in the profile are displayed. The final value depends on the user needs, however values around `kmax = nmax = 500` are recommended to guarantee accurate results.

Figure 6.1 shows the mode shape of the 98-th transverse mode of a perfect plate with free edge and Poisson ratio $\nu = 0.38$. It is calculated by executing: `[U,V,W] = ModeShapeCircular( 4, 1, 18.4232, 1, 0.38, 0, 500, 500);`.

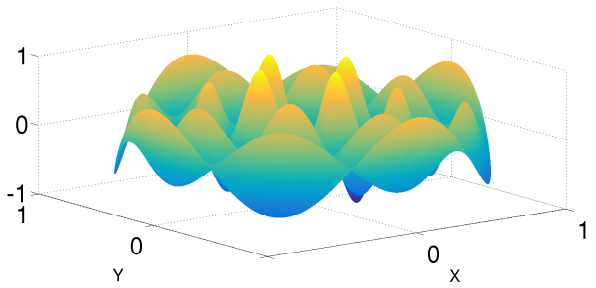


Figure 6.1: Transverse mode shape obtained when `k=4`, `c=1`, `xkn=18.4232`, `R=1`, `nu=0.38`, `KR=0`, `kmax=500` and `nmax=500`.

Function 6.3.5 — ModeShapeCircular.m.

Short description

Computation of the mode shape as a 3D surface.

Call

[U, V, W] = ModeShapeCircular(k, c, xkn, R, nu, KR, kmax, nmax)

Input parameters

k Number of nodal diameters.
c Configuration of the mode cos/sin.
xkn Eigenvalue of the mode.
R Radius of the plate.
nu Poisson ratio.
KR Normalized rotational stiffness. Also used to declare the boundary conditions. Set $KR = 0$ for free boundary and $KR = \infty$ for clamped boundary.
kmax Number of discretization points in θ dimension.
nmax Number of discretization points in r dimension.

Output parameters

U Matrix of x components of the points in the mode shape surface. (Cartesian coordinates)
V Matrix of y components of the points in the mode shape surface. (Cartesian coordinates)
W Values of the mode shape surface in points $[U,V]$.

norm_modes.m

This function is used to calculate the norm of a 2D eigenvector. This value is used in further computations of the code to normalize the eigenvectors. For this reason, the returned variable corresponds to the inverse of the norm, i.e. $Kkn = \|\Phi\|^{-1}$

Function 6.3.6 — norm_modes.m.

Short description

Function used to compute the inverse norm of a 2D vector.

Call

[Kkn] = norm_modes(k_t,xkn,R,dr, nu, KR, BC)

Input parameters

k_t Number of nodal diameters.
xkn Square root of the angular frequency $\xi_s = \sqrt{\omega_s}$.
R Plate radius.
dr Discretization step.
nu Poisson ratio
KR Rotational stiffness normalized with respect to bending stiffness, $KR = Kr/D$. Only used when BC = 'elastic'.
BC Type of boundary conditions at the edge. Possible values: 'free', 'clamped', 'elastic'.

Output parameters

Kkn Inverse of the vector norm.

DisplayEigenfrequenciesCircular.m

This is an utility function used to independently compute and display the eigenfrequencies of a given plate. This means that this function is not called by the main code but provides a fast way to obtain the plate transverse eigenfrequencies without having to execute the main script.

However, it should be executed once the *mode_t.mat* file that corresponds to the plate characteristics has been created by *ComputeTransverseEigenfrequenciesCircular.m*. Remember that in most of the cases, thanks to the variable nondimensioning, the file will already exist from previous simulations.

The input parameters of the function correspond to those in the main script, i.e. the names of the files containing the plate characteristics and the simulation parameters. Their contents are described in Section 6.10. Again, if the file indicated in `GammaFileName` does not exist, it will be created during the execution.

The function returns three vectors, being `om_dim` the angular eigenfrequencies in rad/s, `f_dim` the eigenfrequencies in Hz and `om_ndim` the angular eigenfrequencies in dimensionless form.

Note that if the plate is perfect, i.e. $H = 0$, the values in `om_dim` correspond to those in the last column of `mode_t`.

If the plate is imperfect, the eigenfrequencies will be calculated combining the projection coefficients A_i and the nonlinear coupling coefficients $\mathbf{\Gamma}$, as explained in Section 2.4.

Function 6.3.7 — DisplayEigenfrequenciesCircular.m.

Short description

Function used to compute the eigenfrequencies of a perfect or imperfect circular plate.

Call

[om_dim, f_dim, om_ndim] = DisplayEigenfrequenciesCircular(PlateCharacteristicsFileName, SimulationParametersFileName, GammaFileName)

Preconditions

ComputeTransverseEigenfrequenciesCircular.m has already been executed or the corresponding *mode_t_XXXX.mat* file exists.

Input parameters

PlateCharacteristicsFileName Name of the file containing the plate characteristics parameters.

SimulationParametersFileName Name of the file containing the simulation parameters.

GammaFileName Name of the file containing the $\mathbf{\Gamma}$ tensor, `G`. If the file does not exist, the function will create it using this name.

Output parameters

om_dim Sorted vector containing the angular eigenfrequencies in rad/s.

f_dim Sorted vector containing the eigenfrequencies in Hz.

om_ndim Sorted vector containing the dimensionless angular eigenfrequencies.

6.3.2 Rectangular

ComputeTransverseEigenfrequenciesRectangular.m

This function is used to calculate the transverse eigenfrequencies of a perfect rectangular plate. Given the plate dimensions `Lx` and `Ly` in meters and the boundary conditions `BC`, the function computes the first `Nphi` angular eigenfrequencies and creates the `mode_t` matrix as described below. The output is saved and located in *'./Parameters/Mode files/Rectangular/*.

The first column of `mode_t` corresponds to the mode index, the second and third columns to the integers `kx` and `ky`, and the forth column corresponds to $\left[\left(\frac{k_x\pi}{L_x}\right)^2+\left(\frac{k_y\pi}{L_y}\right)^2\right]$.

Note that this expression does not depend on the material characteristics, in order to obtain the real angular eigenfrequencies, the values in `mode_t(:,4)` should be multiplied by $\sqrt{\frac{D}{\rho hd}}$, where D is the bending stiffness, ρ the mass density and hd the plate thickness.

The reason for omitting this term in `mode_t` is that in this way, the file can be reused for all the plates with the same size. Once the file is created, the user shall use *DisplayEigenfrequenciesRectangular.m* to obtain the eigenfrequencies in the proper unities.

Function 6.3.8 — ComputeTransverseEigenfrequenciesRectangular.m.

Short description

Computation of transverse eigenfrequencies.

Call

[mode_t] = ComputeTransverseEigenfrequenciesRectangular(BC, Lx, Ly, Nphi)

Input parameters

BC Type of boundary conditions at the edge. Possible values: 'SimplySupported'.
Lx Plate dimension X in meters.
Ly Plate dimension Y in meters.
Nphi Number of transverse modes and number of eigenfrequencies that will be computed.

Output parameters

mode_t Vector containing the information corresponding to the first *Nphi* transverse vibration modes. For every mode,

Column 1	Column 2	Column 3	Column 4
$< i >$	$< k_x >$	$< k_y >$	$< \omega_i >$

ModeShapeRectangular.m

This function computes the shape of a transverse mode as a 3D surface. For every pair of coordinates `[X, Y]` the value of `phi` is calculated.

It is used by the main code to compute the projection coefficients between the imperfection and the transverse modal basis. However, it may also be used independently to plot the mode shape of a given mode by means of the Matlab function `surf(X, Y, phi)`.

The function requires the introduction of the boundary conditions `BC`, the mode coefficients `kx` and `ky`, the plate dimensions `Lx` and `Ly` and the number of discretization points in every direction `Nx` and `Ny`. Note that the number of nodal lines in the mode shape will be `kx + 1` and `ky + 1`.

The values of `Nx` and `Ny` should be set to ensure that all the variations in the imperfection profile are displayed. The final value depends on the user needs, however values around `Nx = Ny = 500` are recommended to guarantee accurate results.

Figure 6.2 shows the mode shape of a mode of a simply supported plate, with `kx=5`, `ky=3`, `Lx=0.3`, `Ly=0.5` and `Nx = Ny = 500`. It is calculated by executing: `[X,Y,phi] = ModeShapeRectangular( 'SimplySupported', 5, 3, 0.3, 0.5, 500, 500);`.

Function 6.3.9 — ModeShapeRectangular.m.

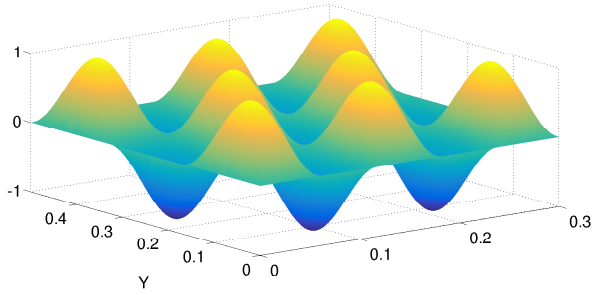


Figure 6.2: Transverse mode shape obtained when `BC='SimplySupported'`, `kx=5`, `ky=3`, `Lx=0.3`, `Ly=0.5`, `Nx = 500` and `Ny = 500`.

Short description

Computation of the mode shape as a 3D surface.

Call

[X, Y, phi] = ModeShapeRectangular(BC, kx, ky, Lx, Ly, Nx, Ny)

Input parameters

BC Boundary conditions. Possible values: 'SimplySupported'.
kx Mode coefficient in direction x. The number of nodal lines in this direction will be `kx + 1`.
kx Mode coefficient in direction y. The number of nodal lines in this direction will be `ky + 1`.
Lx Plate dimension X in meters.
Ly Plate dimension Y in meters.
Nx Number of discretization points in x dimension.
Ny Number of discretization points in y dimension.

Output parameters

X Matrix of x components of the points in the mode shape surface.
Y Matrix of y components of the points in the mode shape surface.
phi Values of the mode shape surface in points $[X,Y]$.

6.4 Nonlinear characteristics functions

6.4.1 Circular

H_tensorCircular.m

This function builds the matrices that contain the coupling coefficients H^i_{pq}. It is only called by the main script when there is not an *H file* with the characteristics required by the simulation, i.e. with the same boundary conditions and parameters and enough number of considered modes. Given that it implies a long computational time, it is recommended to call it independently prior to the time simulation.

The function requires the number of modes included in the simulation, `Nphi` and `Npsi`, the boundary conditions `BC` and the Poisson ratio `nu`.

The rotational `KR` and transverse `KT` normalized stiffnesses are only used if the boundary conditions are set to `'elastic'`.

The last parameter, `dr_H` is used to tune the precision of the computations and corresponds to the discretization interval. It is recommended to set it around `dr_H = 1e-4`.

Function 6.4.1 — H_tensorCircular.m.

Short description

Computation of H matrices.

Call

[H0, H1, H2] = H_tensorCircular(Nphi, Npsi, BC, nu, KR, KT, dr_H)

Input parameters

Nphi Number of transverse modes included in the truncation.

Npsi Number of in-plane modes included in the truncation.

BC Type of boundary conditions at the edge. Possible values: 'free', 'clamped', 'elastic'.

nu Poisson ratio.

KR Rotational stiffness normalized with respect to bending stiffness, $KR = Kr/D$. Only used when BC = 'elastic'.

KT transverse stiffness normalized with respect to bending stiffness, $KT = Kt/D$. Only used when BC = 'elastic'.

dr_H Discretization step for the computation of H.

Output parameters

H0 Matrix of size $Npsi \times Nphi \times Nphi$ containing the H coupling coefficient tensor, $H(i,p,q) = H^i_{pq}$.

H1 Matrix that contains matrix H0 divided by the eigenfrequency of the corresponding in-plane mode, i.e. $H1(i,p,q) = H^i_{pq}/\omega_i = H^i_{pq}/\zeta_i^2$.

H2 Matrix that contains matrix H0 divided by the squared eigenfrequency of the corresponding in-plane mode, i.e. $H2(i,p,q) = H^i_{pq}/\omega_i^2 = H^i_{pq}/\zeta_i^4$.

For every combination of indexes, `i`, `p`, `q`, *H_tensorCircular.m* calls function *HcoefficientCircular.m* to compute a single value of H^i_{pq} according to eq. (2.11b).

The surface integral in eq. (2.11b) can be separated in two terms, one depending on the radius r and the other, on the angle θ. For certain combinations of modes, the H coefficient is directly null and does not need to be computed. This will depend on the number of nodal diameters `kp`, `kq` and `ki` and the mode configurations `cp`, `cq` and `ci`, as detailed in Table 6.1. On the other cases, functions *CosCosCosIntegration.m* and *CosSinSinIntegration.m* are used.

	$\mathbf{c_p}$	$\mathbf{c_q}$	$\mathbf{c_i}$
$k_i \neq \{k_p + k_q, \lvert k_p - k_q \rvert\}$	-	-	-
$k_i = \{k_p + k_q, \lvert k_p - k_q \rvert\}$	cos	cos	sin
	sin	sin	sin
	sin	cos	cos
	cos	sin	cos

Table 6.1: Cases where the combination of k_i, k_p and k_q makes the nonlinear coupling coefficient be null, i.e. $H^i_{pq} = 0$

The value computed by *HcoefficientCircular.m* is placed in `H` matrix, so that $H0(i,p,q) = H^i_{pq}$. To reduce the computational cost of other functions in the code, *H_tensorCircular.m* also creates matrices *H1* and *H2*, such that $H1(i,p,q) = H^i_{pq}/\zeta_i^2$ and $H2(i,p,q) = H^i_{pq}/\zeta_i^4$.

Function 6.4.2 — HcoefficientCircular.m.

Short description

Computation of a single H coefficient.

Call

[H] = HcoefficientCircular(kp, kq, cp, cq, xip, xiq, ki, ci, zeta, nu, KR, dr_H)

Input parameters

kp Number of nodal diameters of mode p, i.e. k_p.
kq Number of nodal diameters of mode q, i.e. k_q.
cp Configuration of mode p, i.e. c_p.
cq Configuration of mode q, i.e. c_q.
xip Eigenvalue of mode p, i.e. ξ_p.
xiq Eigenvalue of mode q, i.e. ξ_q.
ki Number of nodal diameters of mode i, i.e. k_i.
ci Configuration of mode i, i.e. c_i.
zeta Eigenvalue of mode i, i.e. ζ_i.

nu Poisson ratio.
KR Normalized rotational stiffness. Force for free edge boundary conditions, $KR = 0$, for clamped edge boundary conditions $KT = \infty$.
dr_H Discretization step.

Output parameters

H Coupling coefficient between in-plane mode i and transverse modes p, and q, i.e. H^i_{pq}.

Function 6.4.3 — CosCosCosIntegration.m.

Short description

Analytical computation of $\int_0^{2\pi} \cos(k\theta)\cos(l\theta)\cos(m\theta)\,d\theta$

Call

[S] = CosCosCosIntegration(k, l, m)

Input parameters

k Coefficient of the first $\cos(k\theta)$. It can be a single value or a matrix.
l Coefficient of the first $\cos(l\theta)$. It can be a single value or a matrix.
m Coefficient of the first $\cos(m\theta)$. It can be a single value or a matrix.

Output parameters

S Result of the integral. The size of S is equal to the size of $\{k, l, m\}$.

Function 6.4.4 — CosSinSinIntegration.m.

Short description

Analytical computation of $\int_0^{2\pi} \cos(k\theta)\sin(l\theta)\sin(m\theta)\,d\theta$

Call

[S] = CosSinSinIntegration(k, l, m)

Input parameters

k Coefficient of the first $\cos(k\theta)$. It can be a single value or a matrix.
l Coefficient of the first $\sin(l\theta)$. It can be a single value or a matrix.
m Coefficient of the first $\sin(m\theta)$. It can be a single value or a matrix.
Output parameters
S Result of the integral. The size of S is equal to the size of $\{k,l,m\}$.

6.4.2 Rectangular

H_TensorRectangular.m

This function is used to compute the $\boldsymbol{H}$ tensor of the rectangular plate as defined in section 2.3.1.

The integrals in eq. (2.11b) can be decomposed by separating the terms that depend on the in-plane or the transverse modes. Specific functions have been implemented for every group of terms.

AiryStressFactorsCalculation.m, described below, contains the computations related to the in-plane direction. The outputs of this function `coeff0, coeff1, coeff2` are required as inputs of *H_TensorRectangular.m* and thus, this function should be executed before.

The transverse eigenfrequencies must be also computed in advance if the `mode_t` vector has not been created yet. For that, execute *ComputeTransverseEigenfrequenciesRectangular.m*.

The rest of the input parameters correspond to the plate and simulation characteristics. They are the number of transverse `Nphi` and in-plane `Npsi` modes, the plate dimensions `Lx` and `Ly` and the boundary conditions `BC`.

Function 6.4.5 — H_tensorRectangular.m.
Short description
Computation of H matrices.
Call
[H0, H1, H2] = H_tensorRectangular(coeff0, coeff1, coeff2, Nphi, Npsi, Lx, Ly, mode_t, BC)

Preconditions
Execute *ComputeTransverseEigenfrequenciesRectangular.m* to obtain `mode_t` and *AiryStressFactorsCalculation.m* for `coeff0, coeff1, coeff2`.
Input parameters
coeff0 Coefficient containing the terms relative to the in-plane mode for the computation of the H coefficient.
coeff1 Vector containing coeff0 divided by the angular frequency of the in-plane mode, $coeff1(:,i) = coeff0(:,i)/\omega_i$.
coeff2 Vector containing coeff0 divided by the squared angular frequency of the in-plane mode, $coeff2(:,i) = coeff0(:,i)/\omega_i^2$.
Nphi Number of transverse modes included in the truncation.
Npsi Number of in-plane modes included in the truncation.
Lx Plate dimension X in meters.
Ly Plate dimension Y in meters.
mode_t Vector containing the information corresponding to the first *Nphi* transverse vibration modes. For every mode,

Column 1	Column 2	Column 3	Column 4
$< i >$	$< k_x >$	$< k_y >$	$< \omega_i >$

BC Type of boundary conditions at the edge. Possible values: 'SimplySupported'.

Output parameters

H0 Matrix of size $Npsi \times Nphi \times Nphi$ containing the H coupling coefficient tensor, $H(i,p,q) = H^i_{pq}$.

H1 Matrix that contains matrix H0 divided by the eigenfrequency of the corresponding in-plane mode, i.e. $H1(i,p,q) = H^i_{pq}/\omega_i = H^i_{pq}/\omega_i$.

H2 Matrix that contains matrix H0 divided by the squared eigenfrequency of the corresponding in-plane mode, i.e. $H2(i,p,q) = H^i_{pq}/\omega_i^2$.

Function 6.4.6 — Auxiliary functions: gi.m, i = { 1, 2, 3, 4, 5, 6}.

Short description

Integration terms for the computation of H coefficients.

Call

```
[ m ] = g1( Npsi, Nphi, S, Lx, mode_t)
[ m ] = g2( Npsi, Nphi, S, Lx, mode_t)
[ m ] = g3( Npsi, Nphi, S, Ly, mode_t)
[ m ] = g4( Npsi, Nphi, S, Ly, mode_t)
[ m ] = g5( Npsi, Nphi, S, Lx, mode_t)
[ m ] = g6( Npsi, Nphi, S, Ly, mode_t)
```

Input parameters

Nphi Number of transverse modes included in the truncation.

Npsi Number of in-plane modes included in the truncation.

S Number of computed in-plane eigenvalues.

Lx Plate dimension X in meters.

Ly Plate dimension Y in meters.

mode_t Vector containing the information corresponding to the first *Nphi* transverse vibration modes. For every mode,

Column 1	Column 2	Column 3	Column 4
$< i >$	$< k_x >$	$< k_y >$	$< \omega_i >$

Output parameters

m Solution of the integral.

Function 6.4.7 — Auxiliary functions: ik_mat.m, k = { 1, 2, 3, 4, 5, 9, 10, 11, 12, 13}.

Short description

Auxiliary functions used by gi.m.

Call

```
[ s ] = ik_mat( Npsi, Nphi, L)
```

Input parameters
Nphi Number of transverse modes included in the truncation.
Npsi Number of in-plane modes included in the truncation.
L Plate dimension in meters.

Output parameters
s Solution of calculation.

AiryStressFactorsCalculation.m
This function solves the eigenproblem explained in **??**.

The input values are the boundary conditions `BC`, the number of in-plane modes that will be considered in the simulation `Npsi` and the plate dimensions `Lx` and `Ly`.

The output values `coeff0, coeff1, coeff2` will be used for the computation of the nonlinear tensor $\boldsymbol{H}$ and correspond to the terms of the integral in eq. (2.11b) that depend on the in-plane modes.

In order to solve the eigenproblem, the function calls the subroutines *int1.mat-int4.mat* which solve the integrals eqs. (4.11) to (4.13) respectively. They are described below for the sake of completeness.

Function 6.4.8 — AiryStressFactorsCalculation.m.
Short description
Solves the eigenproblem associated to the Airy stress function for the in-plane direction and returns the factors necessary for the computation of the H coupling coefficients.
Call
[coeff0, coeff1, coeff2] = AiryStressFactorsCalculation(BC, Npsi, Lx, Ly)

Input parameters
BC Type of boundary conditions at the edge. Possible values: 'SimplySupported'.
Npsi Number of in-plane modes in the model.
Lx Plate dimension X in meters.
Ly Plate dimension Y in meters.

Output parameters
coeff0 Coefficient containing the terms relative to the in-plane mode for the computation of the H coefficient.
coeff1 Vector containing coeff0 divided by the angular frequency of the in-plane mode, $coeff1(:,i) = coeff0(:,i)/\omega_i$.
coeff2 Vector containing coeff0 divided by the squared angular frequency of the in-plane mode, $coeff2(:,i) = coeff0(:,i)/\omega_i^2$.

Function 6.4.9 — int1.m .
Short description
Function that solves integral eq. (4.10) for the Airy stress eigenproblem.
Call
[y] = int1(m, p, L)

Input parameters
m Index of the first function.
p Index of the second function.
L Plate dimension in meters.

Output parameters
y Solution of the integral.

Function 6.4.10 — int2.m .
Short description
Function that solves integral eq. (4.11) for the Airy stress eigenproblem.
Call
[y] = int2(m, p, L)

Input parameters
m Index of the first function.
p Index of the second function.
L Plate dimension in meters.

Output parameters
y Solution of the integral.

Function 6.4.11 — int4.m .
Short description
Function that solves integral eq. (4.12) for the Airy stress eigenproblem.
Call
[y] = int4(m, p, L)

Input parameters
m Index of the first function.
p Index of the second function.
L Plate dimension in meters.

Output parameters
y Solution of the integral.

Function 6.4.12 — int2_mat.m .
Short description
Function that solves integral eq. (4.11) and returns a matrix used to compute the norm of the eigenvectors.
Call
[y] = int2_mat(N, L)

Input parameters
N Number of functions to consider and size of the output matrix.
L Plate dimension in meters.

Output parameters
y Solution of the integral. Squared matrix of size $N \times N$.

6.4.3 Common

GammaTensor.m

This function computes the 4-th order tensor Γ^p_{qrs} using the matrix `H1` and saves it in a file specified by `filename`. The Gamma file is used to compute the eigenfrequencies of the imperfect plate. Like *H_tensorCircular.m*, *GammaTensor.m* is only called if there is not an existing file with the required characteristics for the simulation.

The matrix `H1` is obtained after executing *H_tensorCircular.m* or *H_TensorRectangular*, depending on the plate shape, or by loading it from the H file if it already exists. The values of `Nphi` and `Npsi` should be set equal or less than the size of `H1`. Note that, `size(H1) = [Npsi, Nphi, Nphi]`.

This function requires a large amount of memory since it works with a variable of size `Nphi`^{4}. The user is advised to set `Nphi<100`. The value of `Npsi` is generally inferior, so it should not be restricted by memory limitations. However, it is been proved that the values of Gamma converge when $N_\Psi \approx 50$ [10]. Set, `Npsi`$\in$`(50,100)`.

Function 6.4.13 — GammaTensor.m.
Short description
Computation of Γ tensor using H matrices.
Call
[G] = GammaTensor(H1, filename, Nphi, Npsi)

Input parameters
H1 H coefficients matrix divided by the eigenfrequency of the corresponding in-plane mode, i.e. $H1(i,p,q) = H^i_{pq}/\omega_i = H^i_{pq}/\zeta_i^2$.
filename Name of the output file.
Nphi Number of transverse modes to be included. The size of H1 must include at least Nphi transverse modes.
Npsi Number of in-plane modes to be included. The size of H1 must include at least Npsi in-plane modes.

Output parameters
G Matrix of size $Nphi \times Nphi \times Nphi \times Nphi$ containing the $\mathbf{\Gamma}$ coupling coefficient tensor, $G(p,q,r,s) = \Gamma^p_{qrs}$.

6.5 Imperfection functions

The current model can not only deal with perfect plates but also with those that present imperfections in their profile. The way to introduce them in the time integration calculation is by expressing the imperfection shape as a series expansion of the modes in the transverse modal basis multiplied by the projection coefficients as written in eq. (2.14).

The code offers the choice of introducing manually the projection coefficients along to the respective mode indexes or choosing one of the preset shapes and compute automatically the necessary values. When the second option is selected, the following functions are used.

6.5.1 Circular

ProjectionOfTheImperfectionCircular.m

When the modeled plate is imperfect, i.e. when `H>0` and the projection coefficients are not introduced, this function calculates the imperfection profile, builds the 3D surface and computes the projection coefficients that will be used by the time stepping function to expand w_0 as in eq. (2.14).

To that end, the function calls successively *AxisymmetricCap.m* and *ComputationOfTheProjectionCoefficientsCircular.m*. The input and output parameters are described below.

Function 6.5.1 — ProjectionOfTheImperfectionCircular.m.

Short description

Computation of the imperfection profile and calculation of the projection coefficients from the transverse modal basis of the perfect plate.

Call

[U, V, Imperfection, proj, modeIndices, Approximation, Rc] = ProjectionOfTheImperfectionCircular(H, hd,Rd, ImperfectionType, tau2, Nr, Nth, nu, KR, error_coef, ModeType, mode_t)

Input parameters

H Height of the imperfection.
hd Thickness of the plate.
Rd Radius of the plate in meters.
ImperfectionType Shape of the imperfection profile. Possible values: 'Spherical', 'Parabolic'.
tau2 When ImperfectionType = 'Parabolic', tau2 is the parabola order. $tau2 \in \left(-10^{250}, 10^{250}\right)$
Nr Number of discretization points for the radius variable r.
Nth Number of discretization points for the angle variable θ.
nu Poisson ratio.
KR Rotational stiffness normalized with respect to bending stiffness, $KR = Kr/D$. Only used when BC = 'elastic'.
error_coef Top error admitted in the approximation of the imperfection. $error\_coef \in [0,1]$
ModeType Type of modes considered in the approximation of the imperfection. Possible values: 'All', 'Axisymmetric'.
mode_t Vector containing the information corresponding to the transverse vibration modes. For every mode,

Column 1	Column 2	Column 3	Column 4	Column 5	Column 6
$< i >$	$< \xi_i >$	$< k_i >$	$< n_i >$	$< c_i >$	$< \omega_i >$

Output parameters

U Matrix of x components of the points in the Imperfection. (Cartesian coordinates)
V Matrix of y components of the points in the Imperfection. (Cartesian coordinates)
Imperfection Values of the imperfection profile in points $[U,V]$.
proj Vector of projection coefficients.
modeIndices Indices of the modes included in vector *proj* as sorted in vector *mode_t*.
Approximation Approximated imperfection profile obtained as a linear combination of the

projection coefficients and the corresponding mode shapes.
Rc Curvature radius of the plate. Only used for spherical profile.

AxisymmetricCap.m

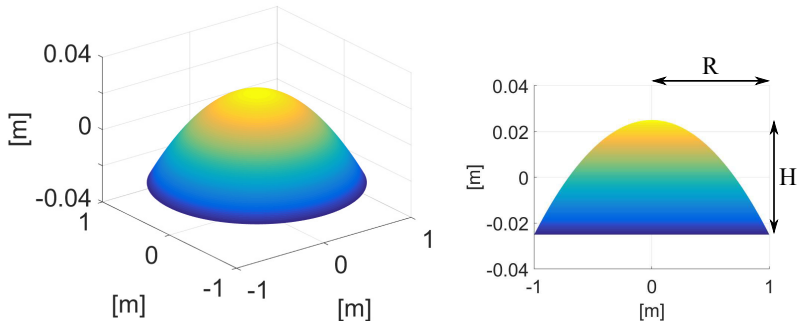
For the construction of the imperfection profile, *AxisymmetricCap.m* is called. There are two shapes included: spherical and parabolic, each one configured by different parameters.

- Spherical: Spherical cap shaped imperfection (See fig. 6.3). It is parametrized by
  H Height of the imperfection
  R Plate radius
  The profile is computed using

$$y(r) = -R_c + \sqrt{R_c^2 - r^2} \tag{6.1}$$

  where the radius of curvature *Rc* is computed as

$$R_c = \frac{H^2 + R^2}{2H}. \tag{6.2}$$



(a) (b)

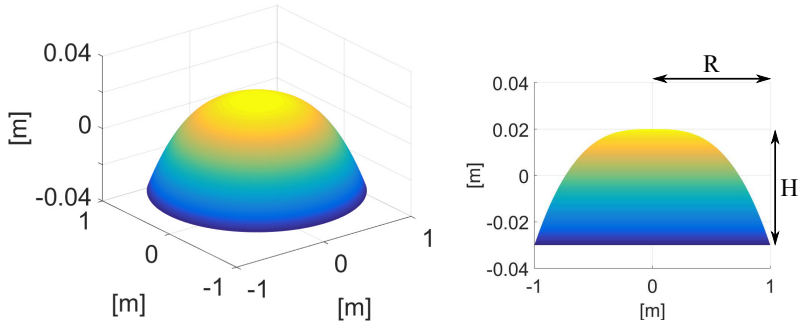
Figure 6.3: Spherical imperfection with `Rd=1`, `H=0.05`. This shape has been plot setting `Nr=400` and `Nth=500`.

- Parabolic: The imperfection profile is a paraboloid as the one displayed in fig. 6.4, defined by

$$y = -Hr^{\tau_2} \tag{6.3}$$

  and parametrized by
  H Height of the imperfection
  R Plate radius
  tau2 Parabola order. Mathematically, τ_2 can take any real value but `tau2` should be bounded to $\pm 10^{250}$ to avoid under/overflows.

`Nr` and `Nth` are the number of discretization of points for variables r and θ respectively. They should be chosen so that all the variations in the shape can be appreciated. If there are no memory limitations, the user is advised to set both of them to values larger than 400, e.g. `Nr=400` and `Nth=500`.



(a) (b)

Figure 6.4: Parabolic imperfection with `R=1`, `H=0.05`, `tau2 = 3`. This shape has been plot setting `Nr=400` and `Nth=500`.

Function 6.5.2 — AxisymmetricCap.m.

Short description

Building of the imperfection profile as a 3D surface.

Call

[U, V, Imperfection] = AxisymmetricCap(H, R, ImperfectionType, Nr, Nth, tau2)

Input parameters

H Height of the imperfection.

R Radius of the plate.

ImperfectionType Shape of the imperfection profile. Possible values: 'Spherical', 'Parabolic'.

tau2 When ImperfectionType = 'Parabolic', tau2 is the parabola order. $tau2 \in \left(-10^{250}, 10^{250}\right)$. (Not used when ImperfectionType ='Spherical', any value can be introduced.)

Nr Number of discretization points for the radius variable r.

Nth Number of discretization points for the angle variable θ.

Output parameters

U Matrix of x components of the points in the Imperfection. (Cartesian coordinates)

V Matrix of y components of the points in the Imperfection. (Cartesian coordinates)

Imperfection Values of the imperfection profile in points $[U,V]$.

ComputationOfTheProjectionCoefficientsCircular.m

Next *ComputationOfTheProjectionCoefficientsCircular.m* calculates the projection coefficients to approximate the shape in `Imperfection` as stated in eq. (2.15).

Incrementally, the function takes the values of row i in `mode_t`, computes the shape of that mode with *ModeShapeCircular.m* and calculates the scalar product between `Imperfection` and the obtained mode shape, using *PolarScalarProduct.m*. The result of this operation is the projection coefficient a_i which is added to vector *proj*, whereas the mode index i is added to *modeIndices*. The process is repeated until the relative error between the original and the approximated shape is lower than `error_coef` or until `Nphi` modes have been considered.

The function also requires the Poisson ratio `nu` and the boundary conditions to calculate the transverse mode shapes. The boundary conditions are introduced by selecting `KR` accordingly. When `BC='elastic'`, `KR` equals the normalized rotational stiffness. On the other cases, `KR=0` indicates `BC='free'` and `KR=inf`, `BC='clamped'`.

In order to accelerate the computation process, when an axisymmetric profile must be approximated, the `ModeType` can be set to *'Axisymmetric'* so that only this kind of modes are considered

for the calculations.

For instance, we want to approximate the shape in Figure 6.3, for a plate with Poisson ratio $\nu = 0.38$, thickness $hd = 1e-3$ and free edge boundary conditions. The error is bounded to `error_coef=0.01`. The rotational stiffness is set to `KR=0` to indicate that the boundary conditions are `BC='free'`. Given that this is an axisymmetric shape, `ModeType='Axisymmetric'`. `mode_t` is obtained after running *ComputeTransverseEigenfrequenciesCircular.m*. Only the first `Nphi = 1000` rows of `mode_t` will be passed as argument.

The function is called using: `[proj, modeIndices, Approximation, error] = ComputationOfTheProjectionCoefficientsCircular( Imperfection, error_coef, nu, KR, ModeType, mode_t(1:Nphi,:));` Note that `Imperfection` must be divided by `hd` to obtain dimensionless projection coefficients. This is not necessary when using dimensioned magnitudes.

The function returns the projection coefficients displayed in Table 6.2 and the approximated shape shown in Figure 6.5. See that the series expansion only needs 9 terms to approximate the shape with the desired error. Using these values, the committed error is `error=0.005`. In contrast, if the number of considered transverse modes was smaller, for instance `Nphi =100`, only the first 6 values of `proj` would have been obtained and the error would have been above `error_coef`, i.e. `error=0.0994`. Therefore, complex profile shapes require a compromise between accuracy and computational cost.

i	modeIndices(i)	proj(i)
1	3	2.54e-02
2	14	-2.95e-03
3	29	8.57e-04
4	50	-3.59e-04
5	73	1.84e-04
6	104	-1.07e-04
7	141	6.77e-05
8	178	-4.65e-05
9	225	3.29e-05

Table 6.2: Projection coefficients calculated for the approximation of the spherical imperfection in Figure 6.3, when `nu=0.38`, `KR=0`, `ModeType='Axisymmetric'`, `Nphi=1000` and `error_coef=0.01`.

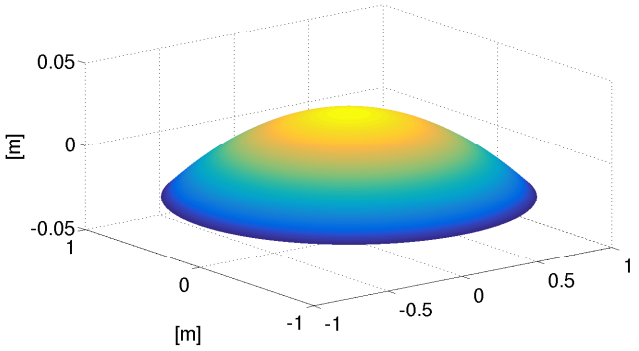


Figure 6.5: Approximation of the spherical imperfection in Figure 6.3, when `nu=0.38`, `KR=0`, `ModeType='Axisymmetric'`, `Nphi=1000` and `error_coef=0.01`.

Function 6.5.3 — ComputationOfTheProjectionCoefficientsCircular.m.

Short description

Computation of the projection coefficients.

Call

[proj, modeIndices, Approximation, error] = ComputationOfTheProjectionCoefficientsCircular(Imperfection, error_coef, nu, KR, ModeType, mode_t)

Input parameters

Imperfection Values of the imperfection profile in points $[U,V]$.

error_coef Top error admitted in the approximation of the imperfection. $error\_coef \in [0,1]$

nu Poisson ratio.

KR Normalized rotational stiffness. Only used when BC = 'elastic'. Set KR=0 for BC = 'free' and KR = inf for BC = 'clamped'.

ModeType Type of modes considered in the approximation of the imperfection. Possible values: 'All', 'Axisymmetric'.

mode_t Vector containing the information corresponding to the transverse vibration modes. For every mode,

Column 1	Column 2	Column 3	Column 4	Column 5	Column 6
$< i >$	$< \xi_i >$	$< k_i >$	$< n_i >$	$< c_i >$	$< \omega_i >$

Output parameters

proj Vector of projection coefficients.

modeIndices Indexes of the modes included in vector *proj* as sorted in vector *mode_t*.

Approximation Approximated imperfection profile obtained as a linear combination of the projection coefficients and the corresponding mode shapes.

error Maximum error committed in the approximation. $error \in [0,1]$.

Function 6.5.4 — PolarScalarProduct.m.

Short description

Function that computes the scalar product between two functions expressed in polar coordinates using

$$< f1, f2 >= \int_{r_{min}}^{r_{max}} \int_{\theta_{min}}^{\theta_{max}} f1(r,\theta) f2(r,\theta) r \, dr \, d\theta$$

Call

[I] = PolarScalarProduct(f1, f2, r_min, r_max, theta_min, theta_max)

Input parameters

f1 First function.

f2 Second function

r_min Lower boundary for variable r.

r_max Upper boundary for variable r

theta_min Lower boundary for variable θ.

theta_max Upper boundary for variable θ

Output parameters

I Result of the integral.

6.5.2 Rectangular

ProjectionOfTheImperfectionRectangular.m

When the modeled plate is imperfect, i.e. when `H>0` and the projection coefficients are not introduced, this function calculates the imperfection profile, builds the 3D surface and computes the projection coefficients that will be used by the time stepping function to expand w_0 as in eq. (2.14).

To that end, the function calls successively *RectangularImperfection.m* and *ComputationOfTheProjectionCoefficientsRectangular.m*. The input and output parameters are described below.

Function 6.5.5 — ProjectionOfTheImperfectionRectangular.m.

Short description

Computation of the imperfection profile and calculation of the projection coefficients from the transverse modal basis of the perfect plate.

Call

[U, V, Imperfection, proj, modeIndices, Approximation] = ProjectionOfTheImperfectionRectangular(H, Lx, Ly, Nx, Ny, error_coef, ModeType, ImperfectionType, xWidth, yWidth, Nphi)

Input parameters

H Height of the imperfection.
Lx Plate dimension X in meters.
Ly Plate dimension Y in meters.
Nx Number of discretization points in x.
Ny Number of discretization points in y.
error_coef Top error admitted in the approximation of the imperfection. $error\_coef \in [0,1]$
ModeType Type of modes considered in the approximation of the imperfection. Possible values: 'All'.
ImperfectionType Shape of the imperfection profile. Possible values: '2DRaisedCosine'.
xWidth Half-width of the 2D raised cosine in the X direction.
yWidth Half-width of the 2D raised cosine in the Y direction.
Nphi Number of transverse modes considered for the approximation.

Output parameters

U Matrix of x components of the points in the Imperfection. (Cartesian coordinates)
V Matrix of y components of the points in the Imperfection. (Cartesian coordinates)
Imperfection Values of the imperfection profile in points $[U,V]$.
proj Vector of projection coefficients.
modeIndices Indices of the modes included in vector *proj* as sorted in vector *mode_t*.
Approximation Approximated imperfection profile obtained as a linear combination of the projection coefficients and the corresponding mode shapes.

RectangularImperfection.m

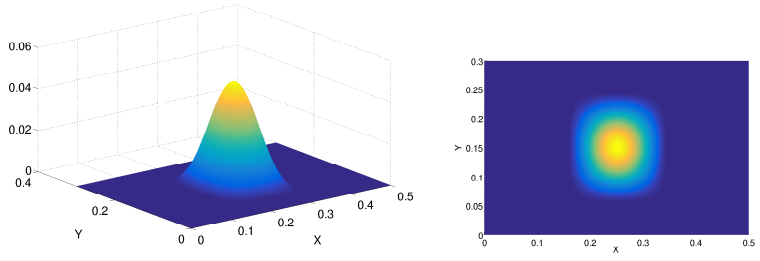
The preset shape offered for the rectangular plate is a centered 2D raised cosine, defined by

$$g(x,y) = \begin{cases} \frac{H}{4}\left(1+\cos\left(\frac{\pi(x-L_x/2)}{x_{width}}\right)\right)\left(1+\cos\left(\frac{\pi(y-L_y/2)}{y_{width}}\right)\right), & \text{if } |x-L_x/2| \le x_{width} \\ & and\, |y-L_y/2| \le y_{width} \\ 0 & \text{otherwise,} \end{cases} \tag{6.4}$$

and parametrized by

H Height of the imperfection
L_x Plate dimension X.
L_y Plate dimension Y.
x_{width} Half-width of the 2D raised cosine in the X direction.
y_{width} Half-width of the 2D raised cosine in the Y direction.

Figure 6.6 shows an example of a plate with a 2D raised cosine imperfection. The plate dimensions are $L_x = 0.5$ m and $L_y = 0.3$ m. The imperfection height is $H = 0.05$ m. The raised cosine half-width is 0.1 m in both directions. Thus, for the x direction, it goes from $x = 0.15$ m to $x = 0.35$ m and for the y direction, from $y = 0.5$ m to $y = 2.5$ m. This shape is obtained by executing `[U,V,Imperfection] = RectangularImperfection( 0.5,0.3,0.05,500,500,'2DRaised-Cosine',0.1,0.1);`



(a) (b)

Figure 6.6: Rectangular plate with `Lx=0.5`, `Ly=0.3` with a 2D raised cosine imperfection where $H = 0.05$ m `xWidth=yWidth=0.1`
. The imperfection has been discretized with `Nx=Ny=500`.

Function 6.5.6 — RectangularImperfection.m.

Short description

Building of the imperfection profile as a 3D surface.

Call

[U, V, Imperfection]=RectangularImperfection(Lx, Ly, H, Nx, Ny, ImperfectionType, xWidth, yWidth)

Input parameters

Lx Plate dimension X in meters.
Ly Plate dimension Y in meters.
H Height of the imperfection.
Nx Number of discretization points in x.
Ny Number of discretization points in y.
ImperfectionType Shape of the imperfection profile. Possible values: '2DRaisedCosine'.
xWidth Half-width of the 2D raised cosine in the X direction.
yWidth Half-width of the 2D raised cosine in the Y direction.

Output parameters

U Matrix of x components of the points in the Imperfection. (Cartesian coordinates)
V Matrix of y components of the points in the Imperfection. (Cartesian coordinates)
Imperfection Values of the imperfection profile in points $[U,V]$.

ComputationOfTheProjectionCoefficientsRectangular.m

This function computes the projection coefficients to express the imperfection profile as a series expansion in terms of the transverse modal basis of the plate.

The profile is introduced by means of `Imperfection` which is obtained after executing *RectangularImperfection.m*. However, the user can modify the code to manually introduce any other shape that is consistent with the selected boundary conditions in `BC`.

The accuracy of the series is tunned with `error_coef` which fixes the maximum admitted error in the approximation. The routine will compute projection coefficients by calculating the Cartesian scalar product of the Imperfection with every mode shape in the modal basis, until the approximation error is below `error_coeff` or until `Nphi` modes are considered. For the rectangular case, the computation of the error uses a less strict formula than in the circular case given that the series convergence would require a larger number of modes. The user is advised to set `error_coef = 0.01` which equals a maximum relative error of 1%.

`ModeType` indicates the group of modes that will be considered for the approximation. Up to this version of the code, the only possible value is `ModeType = 'All'`. However, the parameter is introduced aiming at possible updates or expansions of the program.

The function returns two main variables `proj` and `modeIndices`. Every position of `proj` contains the projection coefficient that corresponds to the mode indicated by the same position in `modeIndices`. The values in `modeIndices` contain the index of the mode in `mode_t`, i.e. the index of the transverse modes when sorted according to their eigenfrequencies.

On the other hand, `Approximation` contains the shape obtained when expressing the imperfection profile by means of the coefficients in `proj` and the modes in `modeIndices`. Finally, `error` is the error committed in the approximated profile.

As an example, the imperfection in Figure 6.6 is approximated by setting `error_coef=0.01`, `ModeType='All'`, `BC='SimplySupported'` and `Nphi=1000`. The function provides a vector `proj` containing 157 values, first ten are displayed in Table 6.3. The approximated shape is shown in Figure 6.7.

i	modeIndices(i)	proj(i)
1	1	2.35e-03
2	2	-4.10e-05
3	3	-1.90e-03
4	4	-3.55e-05
5	5	6.19e-07
6	6	5.90e-05
7	7	2.87e-05
8	8	-1.26e-03
9	9	-8.91e-07
10	10	1.20e-03

Table 6.3: First 10 projection coefficients calculated for the approximation of the 2D raised cosine imperfection in Figure 6.6, when `ModeType='All'`, `Nphi=10` and `error_coef=0.01`.

Function 6.5.7 — ComputationOfTheProjectionCoefficientsRectangular.m.
Short description
Computation of the projection coefficients.
Call

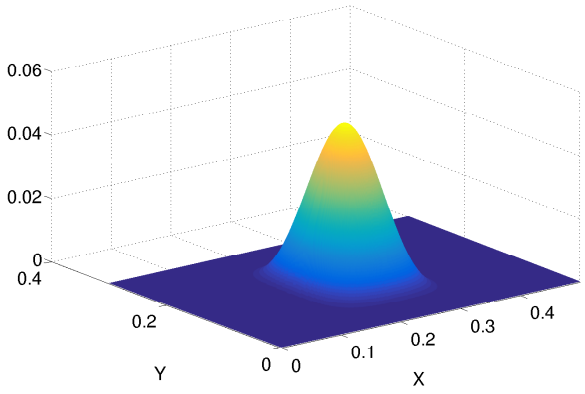


Figure 6.7: Approximation of the 2D raised cosine imperfection in Figure 6.6, when `ModeType='All'`, `Nphi=1000` and `error_coef=0.01`

[proj, modeIndices, Approximation, error] = ComputationOfTheProjectionCoefficientsRectangular(Imperfection, error_coef, ModeType, BC, Nphi, Lx, Ly)

Input parameters

Imperfection Values of the imperfection profile in points $[U,V]$.

error_coef Top error admitted in the approximation of the imperfection. $error\_coef \in [0,1]$

ModeType Type of modes considered in the approximation of the imperfection. Possible values: 'All'.

BC Boundary conditions. Possible values: 'SimplySupported'.

Nphi Number of transverse modes.

Lx Plate dimension X in meters.

Ly Plate dimension Y in meters.

Output parameters

proj Vector of projection coefficients.

modeIndices Indexes of the modes included in vector *proj* as sorted in vector *mode_t*.

Approximation Approximated imperfection profile obtained as a linear combination of the projection coefficients and the corresponding mode shapes.

error Maximum error committed in the approximation. $error \in [0,1]$.

Function 6.5.8 — CartesianScalarProduct.m.

Short description

Function that computes the scalar product between two functions expressed in cartesian coordinates using

$$< f1, f2 >= \int_{x_{min}}^{x_{max}} \int_{y_{min}}^{y_{max}} f1(x,y) f2(x,y)\, dx\, dy$$

Call

[I] = CartesianScalarProduct(f1, f2, x_min, x_max, y_min, y_max)

Input parameters

f1 First function.

f2 Second function

x_min Lower boundary for variable x.
x_max Upper boundary for variable x
y_min Lower boundary for variable y.
y_max Upper boundary for variable y
Output parameters
I Result of the integral.

6.5.3 Common

ComputeEigenfrequenciesImperfectPlate.m

This function is used to compute the eigenfrequencies of the imperfect plate from the values for the perfect plate and the Γ tensor as described in section 2.4. It is called by the main script when the user has demanded a list of the eigenfrequencies of the imperfect plate. In addition, it is also run when the *DisplayEigenfrequenciesCircular/Rectangular.m* is executed to obtain such eigenfrequencies independently.

Since the function is devoted to calculate the eigenfrequencies of the imperfect plate, it requires the parameters related to the approximation of the profile. The input values `proj` and `modeIndices` are obtained from *ProjectionOfTheImperfectionCircular/Rectangular.m* or can be manually introduced by the user. The angular eigenfrequencies are obtained from the last column of the corresponding `mode_t` and `e` is the constant ε used to adapt the von Kármán equations to the dimensioned or dimensionless form.

The `GammaFileName` indicates the name of the file that should contain the `Gamma` matrix corresponding to the characteristics of the problem. The user should be specially careful and avoid introducing a filename with the wrong contents. If this file does not exist, the function will calculate the `Gamma` matrix and save it in a file named by `GammaFileName`.

Note that `Gamma` is a four-dimension matrix that might require a large amount of memory and disk space. Given that the computer capacity may not be able to deal with matrices of size proportional to the value of `Nphi` set in the main simulation, this function permits introducing a lower value `NA` for the computation of the eigenfrequencies. This value corresponds to the size of the **A** matrix in eq. (2.18) and therefore, to the number of computed eigenfrequencies. It is also used as a flag to indicate to the main script that the eigenfrequencies should not be calculated. When `NA=0`, the main script will skip this function.

Function 6.5.9 — ComputeEigenfrequenciesImperfectPlate.m.
Short description
Calculation of the eigenfrequencies of the imperfect plate using the eigenfrequencies of the equivalent perfect plate and the *Gamma* factor.
Call
[Omega , A_matrix] = ComputeEigenfrequenciesImperfectPlate(proj, NA, modeIndices, om, e, GammaFileName)

Input parameters
proj Vector of projection coefficients.
NA Number of eigenfrequencies to compute.
modeIndices Indices of the modes included in vector *proj* as sorted in vector *mode_t*.
om Angular eigenfrequencies of the perfect plate.
e Constant for dimensioning or non-dimensioning the von Kármán equations, corresponding to ε'.

GammaFileName Name of the file containing the Γ coefficient.
Output parameters
Omega Eigenfrequencies of the imperfect plate.
A_matrix Matrix corresponding to eq. (2.18).

6.6 Excitation and damping functions

6.6.1 Common

c_preset.m

This function creates a damping vector `c` of `Nphi` positions based on the modal angular frequencies $\omega_s \equiv om(s)$ and a preset function selected by `X`. The possible values of `X` are `'Undamped'` and `'PowerLaw'`. The former, creates an array of zeros whereas the latter builds a vector defined by this expression

$$c_s = d_{Fac}\omega_s^{d_{Exp}} + d_{Cons}. \tag{6.5}$$

The damping vector `c` is used to create the matrices `C,C1,C2` that are input in the time integration functions. Check Section 6.7 and Section 6.9 for further details.

Function 6.6.1 — c_preset.m.
Short description
Function that loads the damping preset values.
Call
[c] = c_preset(X, om, Nphi, dFac, dExp, dCons)

Input parameters
X Switch value to select the preset. Possible values: 'Undamped', 'PowerLaw'.
om Modal angular frequencies.
Nphi Number of modes and vector length.
dFac Multiplicative factor of the damping function.
dExp Exponent of the damping function.
dCons Additive constant to the damping function.
Output parameters
c Vector with the damping coefficients.

StrikeExcitation.m

This function creates a raised cosine time signal trying to reproduce the excitation generated by the strike of a mallet. The curve $g(t)$ is defined by

$$g(t) = \begin{cases} \frac{p_m}{2}\left[1+\cos\left(\frac{\pi(t-t_0)}{T_{wid}}\right)\right], & \text{if } |t-t_0| \le T_{wid} \\ 0, & \text{if } |t-t_0| > T_{wid}, \end{cases} \tag{6.6}$$

where p_m is the force amplitude, t_0 the initial time of the pulse and T_{wid} the half-time-width of the pulse. In the code, $t_0 \equiv T0$ and $p_m \equiv fm$.

The output `fex` is a vector of `Tn` time steps sampled at sampling rate `fs`. If the excitation lasts longer than `Tn`, i.e. `(T0+2*Twid)>(Tn/fs)`, the function will neglect the samples after `Tn`.

The function works for both dimensioned and dimensionless magnitudes. If the inputs are introduced in dimensionless form, the output will be also dimensionless. This should be the case for the circular case.

Figure 6.8 shows an example of strike excitation in a signal of $Ts = 1$s equivalent to $Tn = 40000$ samples when the sampling rate is $fs = 40$kHz. The pulse starts at $T0 = 200$ms and lasts until $t = 400$ms given that $Twid = 100$ms.

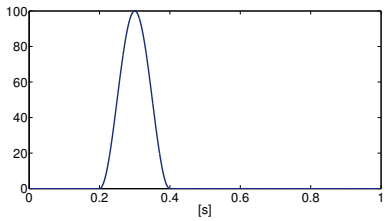


Figure 6.8: Raised cosine signal with $T0 = 0.2$s, $Twid = 0.1$s, $fm = 100$N, $fs = 40$kHz and $Tn = 40000$.

Function 6.6.2 — StrikeExcitation.m.

Short description

Function that generates a raised cosine time signal.

Call

[fex] = StrikeExcitation(T0, fm, Twid, fs, Tn)

Input parameters

T0 Initial time delay.
fm Strike amplitude.
Twid Raised cosine half time width.
fs Dimensionless time frequency.
Tn Number of samples of the output signal.

Output parameters

fex Output time signal.

HarmonicSignal.m

This function generates a sinusoid signal that cant have varying amplitude and frequency. The signal starts at time `T0`, has `Tn` time steps and is sampled at rate `fs`.

The vector `Times` indicates the instants where every variation of frequency or amplitude should happen. Note that the values are relative to `T0`. For every value in `Times`, a value of `Amplitude` and frequency `f` must be given. This means that `Times`, `Amplitude` and `f` must have the same length. In addition, the first value of `Times` should be 0.

Let us illustrate this with an example. A variable harmonic signal of 5 seconds will be generated. The sample rate is set at `fs=40000`, thus the number of samples is `Tn=200000`. The initial time and phase are `T0=1`s and `phase=0`. The sinusoid should last 3s and change every second, thus `Times=[0 1 2 3]`. The frequency should increase from 1Hz to 3Hz at the first interval, stay constant at the second interval, and decrease again to 1Hz at the last interval. For the amplitude, it will stay constant at 1N at the first interval, rise to 5N at the second interval and finally, decrease to 1N again. Therefore, the input parameters are `f=[1 3 3 1]` and `Amplitude=[1 1 5 1]`. The result is plot in Figure 6.9.

On the other hand, if the user prefers a sinusoid signal with constant amplitude and frequency, they should introduce a single value in `Times`, that will correspond to the time length of the signal, and also single values in `Amplitude` and `f`.

Figure 6.10 shows an example of sine signal with constant frequency and amplitude. The

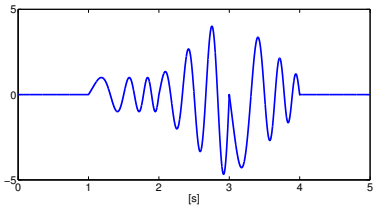


Figure 6.9: Harmonic signal with `T0=1s`, `phase=0`, `f=[1 3 3 1]`, `Amplitude=[1 1 5 1]`, `Times=[0 1 2 3]`, `fs=40000Hz` and `Tn=200000`.

initial time is set at `T0=2.5s`, the sampling rate and time length equal the previous example. The frequency is set to `f=2Hz` and the amplitude to `Amplitude=3N`. The excitation should last 2.5s so `Times=2.5`.

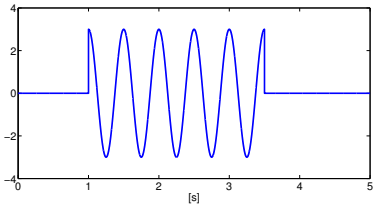


Figure 6.10: Harmonic signal with `T0=2.5s`, `phase=pi/2`, `f=2`, `Amplitude=3`, `Times=2.5`, `fs=40000Hz` and `Tn=200000`.

Note that for the circular case, the parameters should be introduced in dimensionless form. In addition, if the excitation lasts longer than `Tn`, i.e. `(T0+Times(end))>(Tn/fs)`, the function will neglect the samples after `Tn`.

Function 6.6.3 — HarmonicSignal.m.

Short description

Function that generates a harmonic wave time signal.

Call

[fex] = HarmonicSignal(T0, f, Amplitude, phase, Times, fs, Tn)

Input parameters

T0 Initial time delay.

f Vector containing the frequencies at every time position introduced in `Times`.

Amplitude Vector containing the amplitudes at every time position introduced in `Times`.

phase Initial phase of the harmonic signal.

Times Vector containing the times where the signal has a variation of amplitude or frequency. If the length of `Times` is larger than unity, the initial value of this vector should always be 0.

fs Sampling rate.

Tn Number of samples of the output signal.

Output parameters

fex Output time signal.

ColoredNoiseSignal.m

This function generates a colored noise signal for the plate excitation. A pseudo-random white noise signal is first generated by means of the series expansion,

$$y = \sum_{i=0}^{N_f} \cos(2\pi(f_{min} + i\Delta f) + \phi_i), \quad N_f = \frac{f_{max} - f_{min}}{\Delta f}, \tag{6.7}$$

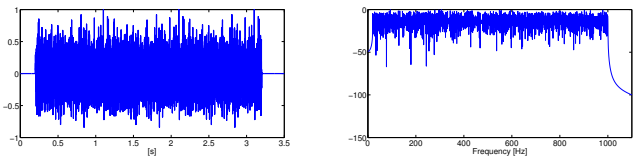
where f_{min} and f_{max} are the frequency limits of the signal, Δf is the distance between two consecutive frequencies in the series, and ϕ_i is the phase value that is obtained pseudo-randomly. Next, the function is filtered so that the frequency spectrum corresponds to one of the colored noise types below.

Thus, the input parameters of this function are the `Color`, the initial time `T0`, the `Amplitude`, the frequency boundaries `fmin` and `fmax`, the distance between two frequencies `deltaf` and the duration of the signal *TimeLength*. The function is sampled at sampling frequency *fs* and placed in a vector of Tn samples. The possible values for the *Color* parameter are

'White' The output signal has equal power in bands with the same bandwidth.
'Pink' The output signal has equal power in frequency bands that are proportionally wide.
'Blue' The power density of the output signal increases 3dB per octave with increasing frequency.
'Red' The power density of the output signal decreases 6dB per octave with increasing frequency.
'Purple' The power density of the output signal increases 6dB per octave with increasing frequency.

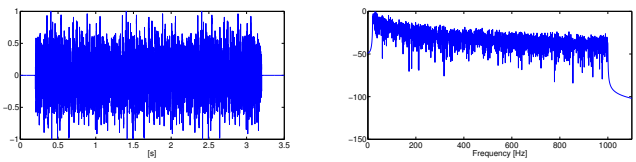
Note that for the circular case, the parameters should be introduced in dimensionless form. In addition, if the excitation lasts longer than `Tn`, i.e. `(T0+TimesLength)>(Tn/fs)`, the function will neglect the samples after `Tn`.

Figures 6.11 to 6.15 show colored noise signals accompanied with their spectrum. They are parametrized with the same values but with different colors.



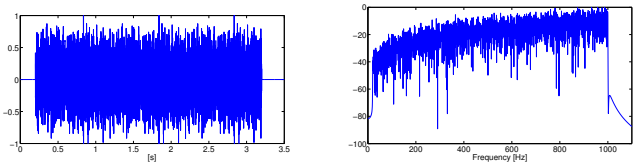
(a) (b)

Figure 6.11: White noise with `T0 =0.2s`, `Amplitude=1`, `fmin=20Hz`, `fmax=1000Hz`, `deltaf=1Hz`, `TimeLength=3s`, `fs=40kHz` and `Tn=140000`.



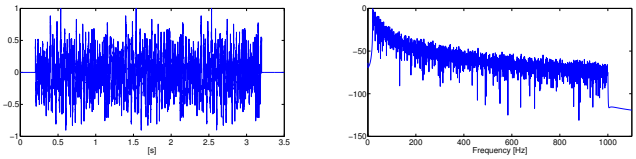
(a) (b)

Figure 6.12: Pink noise with `T0 =0.2s`, `Amplitude=1`, `fmin=20Hz`, `fmax=1000Hz`, `deltaf=1Hz`, `TimeLength=3s`, `fs=40kHz` and `Tn=140000`.



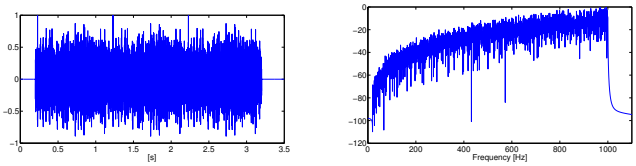
(a) (b)

Figure 6.13: Blue noise with `T0 =0.2s`, `Amplitude=1`, `fmin=20Hz`, `fmax=1000Hz`, `deltaf=1Hz`, `TimeLength=3s`, `fs=40kHz` and `Tn=140000`.



(a) (b)

Figure 6.14: Red noise with `T0 =0.2s`, `Amplitude=1`, `fmin=20Hz`, `fmax=1000Hz`, `deltaf=1Hz`, `TimeLength=3s`, `fs=40kHz` and `Tn=140000`.



(a) (b)

Figure 6.15: Purple noise with `T0 =0.2s`, `Amplitude=1`, `fmin=20Hz`, `fmax=1000Hz`, `deltaf=1Hz`, `TimeLength=3s`, `fs=40kHz` and `Tn=140000`.

Function 6.6.4 — ColoredNoiseSignal.m.

Short description

Computes a vector of colored noise.

Call

[fex] = ColoredNoiseSignal(Color, T0, Amplitude, fmin, fmax, deltaf, TimeLength, fs, Tn)

Input parameters

Color Type of colored noise.
T0 Initial time delay.
Amplitude Amplitude of the harmonic or colored noise signal.
fmin Lower boundary of the noise frequency bandwidth.
fmax Upper boundary of the noise frequency bandwidth.
deltaf Distance between two consecutive frequencies in the cosine series used to generate the noise.
TimeLength Duration of the excitation.
fs Dimensionless time frequency.
Tn Number of samples of the output signal.

Output parameters
fex Output time signal.

6.7 Time integration functions

6.7.1 Common

The resolution of the system in the time domain is performed by *ftime_imperfect_ECS.m* or *ftime_imperfect_verlet.m* depending on the value of `scheme`.

a) `scheme = 'ECS'` stands for Energy Conserving Scheme. In this case, *ftime_imperfect_ECS.m* solves eqs. (5.8) and (5.9).

b) `scheme = 'verlet'` stands for Störmer-Verlet scheme. In this case, *ftime_imperfect_verlet.m* solves eq. (5.13).

Both functions can be used for rectangular or circular plates but the variables must be adjusted for every case. The construction and adaptation of the parameters is performed by the parsers *plate_def_circ.m*/*plate_def_rect.m* and *score_circ.m*/*score_rect.m*. Although the user is strongly advised to use the parsers rather than create the variables manually, the input arguments are shortly described below.

`Nphi` is the number of transverse modes that will be considered for the simulation. It corresponds to N_Φ, the upper boundary of the series in eq. (2.8a). The value of `Nphi` determines the frequency range of the simulation. It should be chosen so that the frequency corresponding to this mode, i.e. f_{N_Φ} is high enough to cover the energy cascade in the plate response.

`Npsi` is the number of in-plane modes that will be considered for the simulation. It corresponds to N_Ψ, the upper boundary of the series in eq. (2.8b). The value of `Npsi` determines the accuracy on the analysis of the nonlinear response. In [10], it was proved that the results converge when `Npsi>50`.

`Ai` is the vector containing the projection coefficients that, in case of an imperfect plate, are used to express the profile shape. If the plate is perfect, this vector should only contain 0's. Note that in case of an imperfect circular case, the values in `Ai` must be dimensionless. Refer to *ProjectionOfTheImperfectionCircular/Rectangular.m* for further details.

`H0, H1, H2` are the $\boldsymbol{H}$ tensor matrices. They are computed in dimensionless form for the circular case by *H_tensorCircular.m* and in dimensioned form for the rectangular case by *H_TensorRectangular.m*.

`C,C1,C2` are the auxiliary matrices used in the intermediate calculations of the time integration. They are defined in eqs. (5.10) and (5.11). Some remarks:

- In the circular case, the values used to calculate these matrices are dimensionless.
- For the Störmer-Verlet scheme, to accelerate the computations, every element in `C1` and `C2` is divided by the corresponding element in `C`.

Tn is the total number of time steps of the current simulation. It is obtained by `Tn=fs*Ts`, where `fs` is the sampling rate and `Ts` is the simulation time length.

e corresponds to ε_r in eq. (5.7) and it is a constant that permits adapting the von Kármán equations to the rectangular dimensioned form or the circular dimensionless form.

- Circular case: `e = 12*(1-nu^2)` where `nu` is the Poisson ratio.
- Rectangular case: `e = (Lx*Ly/4)/rho*E` where `Lx` and `Ly` are the plate dimensions, `rho` is the volumetric mass density and `E` is the Young modulus.

f_time is a matrix containing the modal excitations. The size of `f_time` is `size(f_time) = [Nphi, Tn]` so every row corresponds to a modal component and every column to a time step. Some remarks:

- In the circular case, the values used to calculate this vector are dimensionless.

- For the Störmer-Verlet scheme, to accelerate the computations, every element in `f_time` is divided by the corresponding element in `C`.

rp is the vector containing the modal shape at every output point. This will be multiplied by the state vector of the time integration scheme to obtain the output displacement. The size of `rp` is `size(rp) = [Nop,Nphi]` where `Nop` corresponds to the number of output points. Thus, every row of `rp` corresponds to an output point and every column to a modal component. Remark:

- In the circular case, the values used to calculate this vector are dimensionless.

The returned variable *out* is the output displacement at the selected output points. Every row of this matrix corresponds to a time step and every column to an output point, e.g. $out(n,i)$ corresponds to the displacement at sample n and point $op(i)$.

Note that for circular plates, the output variables are dimensionless and must be multiplied by the plate thickness *hd* to express them in meters or m/s.

The energy conserving scheme function *ftime_imperfect_ECS.m* provides more precision but also requires a large amount of memory, since the three tensors `H0, H1, H2` must be loaded. On the contrary, *ftime_imperfect_verlet.m* is less memory and time demanding but the results are slightly less accurate. The choice of the time integration scheme will should be a compromise between the computer capacities and the parameters that determine the accuracy of the simulation, these are the number of considered modes `Nphi` and `Npsi`, the simulation time `Ts` and the sampling rate `fsd`.

ftime_imperfect_ECS.m

Function 6.7.1 — ftime_imperfect_ECS.m.

Short description

Function for the time integration of the von Kármán equations for imperfect plates using an energy conserving scheme.

Call

[out] = ftime_imperfect_ECS(Nphi, Npsi, Ai, H0, H1, H2, C, C1,C2, Tn, e, f_time, rp)

Input parameters

Nphi Number of transverse modes included in the truncation.

Npsi Number of in-plane modes included in the truncation.

Ai Full vector of projection coefficients.

H0 Matrix of size $Npsi \times Nphi \times Nphi$ containing the H coupling coefficient tensor, $H(i,p,q) = H^i_{pq}$.

H1 Matrix that contains matrix H0 divided by the eigenfrequency of the corresponding in-plane mode, i.e. $H1(i,p,q) = H^i_{pq}/\omega_i = H^i_{pq}/\zeta_i^2$.

H2 Matrix that contains matrix H0 divided by the squared eigenfrequency of the corresponding in-plane mode, i.e. $H2(i,p,q) = H^i_{pq}/\omega_i^2 = H^i_{pq}/\zeta_i^4$.

C Matrix corresponding to $\left(\frac{1}{k^2} + \frac{C_{ss}}{2k}\right)$ as defined in eq. (5.10).

C1 Matrix corresponding to $\left(-\frac{2}{k^2} + K_{ss}\right)$ as defined in eq. (5.11).

C2 Matrix corresponding to $\left(\frac{1}{k^2} - \frac{c_s}{k}\right)$.

Tn Number of samples of the output signal.

e Dimensioning or non-dimensioning constant corresponding to ε'.

f_time Time vector containing the excitation signal for every mode.

rp Vector containing the modal deformation at the output points.

Output parameters

out Displacement at the output points.

ftime_imperfect_verlet.m

Function 6.7.2 — ftime_imperfect_verlet.m.

Short description

Function for the time integration of the von Kármán equations for imperfect plates using the Störmer-Verlet scheme.

Call

[out] = ftime_imperfect_verlet(Nphi, Npsi, Ai, H1, C, C1, C2, Tn, e, f_time, rp)

Input parameters

Nphi Number of transverse modes included in the truncation.

Npsi Number of in-plane modes included in the truncation.

Ai Full vector of projection coefficients.

H1 Matrix that contains matrix H0 divided by the eigenfrequency of the corresponding in-plane mode, i.e. $H1(i,p,q) = H^i_{pq}/\omega_i = H^i_{pq}/\zeta_i^2$.

C Matrix corresponding to $\left(\frac{1}{k^2} + \frac{C_{ss}}{2k}\right)$ as defined in eq. (5.10).

C1 Matrix corresponding to $\left(-\frac{2}{k^2} + K_{ss}\right)$ as defined in eq. (5.11).

C2 Matrix corresponding to $\left(\frac{1}{k^2} - \frac{c_s}{k}\right)$.

Tn Number of samples of the output signal.

e Dimensioning or non-dimensioning constant corresponding to ε'.

f_time Time vector containing the excitation signal for every mode.

rp Vector containing the modal deformation at the output points.

Output parameters

out Displacement at the output points.

out_vel Velocity at the output points.

6.8 Ouput plot functions

This function is used to display the results of the time integration process. The possible representations are the time signal plot, the Fast Fourier transform and the spectrogram.

The vector to be plotted, i.e. the displacement or velocity at a single point, is placed in `signal`. The user must next introduce the sampling frequency `fsd` expressed in Hz and the maximum frequency `Fc` that will be displayed in the FFT representation or the spectrogram.

The next three parameters are used as flags:

`TimeSignal`

- `TimeSignal = 0`. The time signal is not plotted.
- `TimeSignal = 1`. The time signal is plotted.

`FFT`

- `FFT = 0`. The FFT is not plotted.
- `FFT = 1`. The FFT is plotted in terms of the dimensioned frequency, i.e. in Hertz, up to `Fc`.
- `FFT > 1`. The FFT is plotted in terms of the dimensionless angular frequency, up to the dimensionless equivalent of `Fc`.

`Spectrogram`

- `Spectrogram = 0`. The spectrogram is not plotted.
- `Spectrogram = 1`. The spectrogram is plotted in terms of time (seconds) and dimensioned frequency up to `Fc`.

The remaining parameters are only used if `FFT = 2` , so the nondimensioning constant must be computed. These are the Young modulus `E`, the plate thickness `hd`, the Poisson ratio `nu`, the dimensioned plate radius `Rd` and the volumetric mass density `rho`.

The output figures are saved in file *OutputPlots.fig*.

6.8.1 Common

Function 6.8.1 — DisplayResults.m.

Short description

Function that plots the results of the time integration.

Call

[h] = DisplayResults(signal, fsd, Fc, TimeSignal, FFT, Spectrogram, E, hd, nu, Rd, rho)

Input parameters

signal Vector containing the signal for a single point.
fsd Sampling frequency in Hertz.
Fc Maximum frequency in Hertz contained in the spectrogram and the FFT plot.
TimeSignal Flag to select if the time signal must be plotted. Set *TimeSignal = 1* to display it.
FFT Switch to select if the FFT must be computed. Set *FFT = 1* to display the FFT in terms of frequency in Hertz and *FFT = 2* to display the FFT in terms of dimensionless angular frequency.
Spectrogram Flag to select if the spectrogram must be shown. Set *Spectrogram = 1* to display it.
E Young modulus. Only used if *FFT = 2*.
hd Plate thickness. Only used if *FFT = 2*.
nu Poisson ratio. Only used if *FFT = 2*.
Rd Plate radius in meters. Only used if *FFT = 2*.
rho volumetric mass density. Only used if *FFT = 2*.

Output parameters

h Figure object containing the output plots.

6.9 Parsers

6.9.1 Circular

plate_def_circ.m

This function reads the files defined in the main script and builds the variables necessary for the time simulation. Refer to **??** for a detailed description of the file contents.

First of all, the files introduced in *PlateCharacteristicsFileName* and *SimulationParameters-FileName* are loaded, if they do not exist, the user is informed and preset values are assigned.

The mode files are next loaded, if they do not exist, the mode variables `mode_t` and `mode_l` will be computed and saved.

Next, *LoadHTensorCircular.m* is called. This function looks for a file containing the `H` matrices that correspond to the same boundary conditions and parameters, i.e. `nu`, `KR` and `KT` depending on the case. If such a file does not exist or the found ones do not include enough modes, i.e. less than `Nphi` and `Npsi`, the `H` matrix is computed and the file is created.

The following steps concern the imperfect plate. If the studied plate does not involve any defect, they are skipped. As seen in the previous sections, there are two ways to parametrize the imperfect plate, either by introducing the projection coefficients and modes or by selecting one of the preset shapes and the corresponding characteristics. In the latter case, the projection coefficients are

computed. If the user has required it, the eigenfrequencies of the imperfect plate are also calculated and saved.

Finally, variables are adapted for the dimensionless equations and the auxiliary vectors are computed. Although some of them will not be used independently, the output arguments are briefly described in the table below for the sake of completeness.

Function 6.9.1 — plate_def_circ.m.

Short description

This function parses the files indicated in the main script in order to define the plate and create the necessary variables for the code.

Call

[Rd, hd, E, BC, e, Nphi, Npsi, scheme, H0, H1, H2, filename, Ai, C, C1, C2, k_t, c_t, xkn, JJ, II, Kkn, rp, tnd, fs, Tsd] = plate_def_circ(PlateCharacteristicsFileName, SimulationParametersFileName, OutputFileName, GammaFileName)

Input parameters

PlateCharacteristicsFileName Name of the file containing the plate characteristics parameters.

SimulationParametersFileName Name of the file containing the simulation parameters.

OutputFileName Name used to save the results of the program execution.

GammaFileName Name of the file containing the Gamma matrix. If a file with this name does not exist, it will be created during the performance of the code.

Output parameters

Rd Plate radius in meters.

hd Dimensioned plate thickness in meters.

E Young modulus in Pa.

BC Type of boundary conditions at the edge. Possible values: 'free', 'clamped', 'elastic'.

e Non-dimensioning factor corresponding to ε.

Nphi Number of transverse modes kept in the truncation.

Npsi Number of in-plane modes kept in the truncation.

scheme Time integration scheme. Possible values: 'ECS', 'verlet'.

H0 Matrix of size $Npsi \times Nphi \times Nphi$ containing the H coupling coefficient tensor, $H(i,p,q) = H^i_{pq}$.

H1 Matrix that contains matrix H0 divided by the eigenfrequency of the corresponding in-plane mode, i.e. $H1(i,p,q) = H^i_{pq}/\omega_i = H^i_{pq}/\zeta_i^2$.

H2 Matrix that contains matrix H0 divided by the squared eigenfrequency of the corresponding in-plane mode, i.e. $H2(i,p,q) = H^i_{pq}/\omega_i^2 = H^i_{pq}/\zeta_i^4$.

filename String containing the path and the filename to save the results of the code.

Ai Full vector of projection coefficients.

C Matrix corresponding to $\left(\frac{1}{k^2} + \frac{C_{ss}}{2k}\right)$ as defined in eq. (5.10).

C1 Matrix corresponding to $\left(-\frac{2}{k^2} + K_{ss}\right)$ as defined in eq. (5.11).

C2 Matrix corresponding to $\left(\frac{1}{k^2} - \frac{c_s}{k}\right)$.

k_t Vector containing the number of nodal diameters of the transverse modes.

c_t Vector containing the configuration of the transverse modes.

xkn Vector containing the eigenvalues ξ of the transverse modes.

JJ Vector containing $\tilde{J}^f_k(x)$, $\tilde{J}^c_k(x)$ or $\tilde{J}^e_k(x)$ depending on the value of BC.

II Vector containing $\tilde{I}^f_k(x)$, $\tilde{I}^c_k(x)$ or $\tilde{I}^e_k(x)$ depending on the value of BC.

Kkn Vector containing the inverse of the norm of the transverse modes.
rp Vector containing the modal deformation at the output points.
tnd Non-dimensioning time factor.
fs Dimensionless sampling frequency.
Tsd Time length of the output signal in seconds.

LoadHTensorCircular.m

Function 6.9.2 — LoadHTensorCircular.m.

Short description

Function that loads the H file or creates it if necessary.

Call

[H0, H1, H2] = LoadHTensorCircular(BC, Nphi, Npsi, nu, KR, KT, dr_H, scheme)

Input parameters

BC Type of boundary conditions at the edge. Possible values: 'free', 'clamped', 'elastic'.

Nphi Number of transverse modes kept in the truncation.

Npsi Number of in-plane modes kept in the truncation.

nu Poisson ratio.

KR Rotational stiffness normalized with respect to bending stiffness, $KR = Kr/D$. Only used when BC = 'elastic'.

KT transverse stiffness normalized with respect to bending stiffness, $KT = Kt/D$. Only used when BC = 'elastic'.

dr_H Integration step used for computing the H coefficients.

scheme Time integration scheme. Possible values: 'ECS', 'verlet'.

Output parameters

H0 Matrix of size $Npsi \times Nphi \times Nphi$ containing the H coupling coefficient tensor, $H(i,p,q) = H^i_{pq}$.

H1 Matrix that contains matrix H0 divided by the eigenfrequency of the corresponding in-plane mode, i.e. $H1(i,p,q) = H^i_{pq}/\omega_i = H^i_{pq}/\zeta_i^2$.

H2 Matrix that contains matrix H0 divided by the squared eigenfrequency of the corresponding in-plane mode, i.e. $H2(i,p,q) = H^i_{pq}/\omega_i^2 = H^i_{pq}/\zeta_i^4$.

score_circ.m

This function loads the score file `ScoreFileName` introduced in the main script. If it is not found, the preset values are loaded. The other arguments are obtained after executing *plate_def_circ.m*.

Next, the variable *score_cell* described in section 6.10.1 is read and the functions that generate the excitation signals are called. The output parameter *f_time* corresponds to the excitation term $p(n)$ in eq. (5.8) and eq. (5.13).

Function 6.9.3 — score_circ.m.

Short description

Parser for the excitation file.

Call

[f_time, Tn] = score_circ(ScoreFileName, Rd, hd, E, BC, e, Nphi, scheme, C, k_t, c_t, xkn, JJ, II, Kkn, tnd, fs, Tsd)

Input parameters

ScoreFileName Name of the file containing the excitation variable.

Rd Plate radius in meters.

hd Dimensioned plate thickness in meters.

E Young modulus in Pa.

BC Type of boundary conditions at the edge. Possible values: 'free', 'clamped', 'elastic'.

e Dimensioning or non-dimensioning factor corresponding to ε'.

Nphi Number of transverse modes kept in the truncation.
scheme Time integration scheme. Possible values: 'ECS', 'verlet'.
C Matrix corresponding to $\left(\frac{1}{k^2}+\frac{C_{ss}}{2k}\right)$ as defined in eq. (5.10).
k_t Vector containing the number of nodal diameters of the transverse modes.
c_t Vector containing the configuration of the transverse modes.
xkn Vector containing the eigenvalues ξ of the transverse modes.
JJ Vector containing $\tilde{J}_k^f(x)$, $\tilde{J}_k^c(x)$ or $\tilde{J}_k^e(x)$ depending on the value of BC.
II Vector containing $\tilde{I}_k^f(x)$, $\tilde{I}_k^c(x)$ or $\tilde{I}_k^e(x)$ depending on the value of BC.
Kkn Vector containing the inverse of the norm of the transverse modes.
tnd Non-dimensioning time factor.
fs Dimensionless sampling frequency.
Tsd Time length of the output signal in seconds.

Output parameters
f_time Time vector containing the excitation signal for every mode.
Tn Number of samples of the output signal.

6.9.2 Rectangular

plate_def_rect.m

This function reads the files defined in the main script and builds the variables necessary for the time simulation. Refer to **??** for a detailed description of the file contents.

First of all, the files introduced in *PlateCharacteristicsFileName* and *SimulationParameters-FileName* are loaded, if they do not exist, the user is informed and preset values are assigned.

The mode file is next loaded and if it does not exist, the variable `mode_t` is computed and saved.

Next, *LoadHTensorRectangular.m* is called. This function looks for a file containing the `H` matrices that correspond to the same boundary conditions and plate size, i.e. `Lx` and `Ly`. If such a file does not exist or the found ones do not include enough modes, i.e. less than `Nphi` and `Npsi`, the `H` matrix is computed and the file is created. Note that the execution of *H_tensorRectangular.m* is preceded by *AiryStressFactorsCalculation.m*.

The following steps concern the imperfect plate. If the studied plate does not involve any defect, they are skipped. As seen in the previous sections, there are two ways to parametrize the imperfect plate, either by introducing the projection coefficients and modes or by selecting one of the preset shapes and the corresponding characteristics. In the latter case, the projection coefficients are computed. If the user has required it, the eigenfrequencies of the imperfect plate are also calculated and saved.

Finally, variables are adapted for the dimensionless equations and the auxiliary vectors are computed. Although some of them will not be used independently, the output arguments are briefly described in the table below for the sake of completeness.

Function 6.9.4 — plate_def_rect.m.
Short description
This function parses the files indicated in the main script in order to define the plate and create the necessary variables for the code.
Call
[Lx, Ly, hd, E, rho, BC, e, Nphi, Npsi, scheme, H0, H1, H2, filename, Ai, C, C1, C2, kx, ky, om_dim, rp, tnd, fsd, Tsd] = plate_def_rect(PlateCharacteristicsFileName, SimulationParametersFileName, OutputFileName, GammaFileName)

Input parameters

PlateCharacteristicsFileName Name of the file containing the plate characteristics parameters.

SimulationParametersFileName Name of the file containing the simulation parameters.

OutputFileName Name used to save the results of the program execution.

GammaFileName Name of the file containing the Gamma matrix. If a file with this name does not exist, it will be created during the performance of the code.

Output parameters

Lx Plate dimension X in meters.

Ly Plate dimension Y in meters.

hd plate thickness in meters.

E Young modulus in Pa.

rho volumetric mass density.

BC Type of boundary conditions at the edge. Possible value: 'SimplySupported'.

e Constant factor that corresponds to ε for the rectangular plate.

Nphi Number of transverse modes kept in the series truncation.

Npsi Number of in-plane modes kept in the series truncation.

scheme Time integration scheme. Possible values: 'ECS', 'verlet'.

H0 Matrix of size $Npsi \times Nphi \times Nphi$ containing the H coupling coefficient tensor, $H(i,p,q) = H^i_{pq}$.

H1 Matrix that contains matrix H0 divided by the eigenfrequency of the corresponding in-plane mode, i.e. $H1(i,p,q) = H^i_{pq}/\omega_i = H^i_{pq}/\zeta_i^2$.

H2 Matrix that contains matrix H0 divided by the squared eigenfrequency of the corresponding in-plane mode, i.e. $H2(i,p,q) = H^i_{pq}/\omega_i^2 = H^i_{pq}/\zeta_i^4$.

filename String containing the path and the filename to save the results of the code.

Ai Full vector of projection coefficients.

C Matrix corresponding to $\left(\frac{1}{k^2} + \frac{C_{ss}}{2k}\right)$ as defined in eq. (5.10).

C1 Matrix corresponding to $\left(-\frac{2}{k^2} + K_{ss}\right)$ as defined in eq. (5.11).

C2 Matrix corresponding to $\left(\frac{1}{k^2} - \frac{c_s}{k}\right)$.

kx Vector containing the X modal indexes of the transverse eigenmodes.

ky Vector containing the Y modal indexes of the transverse eigenmodes.

om_dim Angular eigenfrequencies of the transverse direction. Note that `om_dim = sqrt(D/rho/hd)*((kx*pi/Lx).^2 + (ky*pi/Ly).^2)`.

xkn Vector containing the eigenvalues ξ of the transverse modes.

JJ Vector containing $\tilde{J}^f_k(x)$, $\tilde{J}^c_k(x)$ or $\tilde{J}^e_k(x)$ depending on the value of BC.

II Vector containing $\tilde{I}^f_k(x)$, $\tilde{I}^c_k(x)$ or $\tilde{I}^e_k(x)$ depending on the value of BC.

Kkn Vector containing the inverse of the norm of the transverse modes.

rp Vector containing the modal deformation at the output points.

tnd Non-dimensioning time factor. For the rectangular plate, `tnd=1`. It is only added because it is necessary for the time integration function.

fsd Sampling rate.

Tsd Time length of the output signal in seconds.

LoadHTensorRectangular.m

Function 6.9.5 — LoadHTensorRectangular.m.

Short description

Function that loads the H file or creates it if necessary.

Call

[H0, H1, H2] = LoadHTensorRectangular(BC, Nphi, Npsi, Lx, Ly, mode_t)

Input parameters

BC Type of boundary conditions at the edge. Possible values: 'SimplySupported'.

Nphi Number of transverse modes kept in the truncation.

Npsi Number of in-plane modes kept in the truncation.

Lx Plate dimension X in meters.

Ly Plate dimension Y in meters.

mode_t Vector containing the information corresponding to the first *Nphi* transverse vibration modes. For every mode,

Column 1	Column 2	Column 3	Column 4
$< i >$	$< k_x >$	$< k_y >$	$< \omega_i >$

Output parameters

H0 Matrix of size $Npsi \times Nphi \times Nphi$ containing the H coupling coefficient tensor, $H(i,p,q) = H^i_{pq}$.

H1 Matrix that contains matrix H0 divided by the eigenfrequency of the corresponding in-plane mode, i.e. $H1(i,p,q) = H^i_{pq}/\omega_i = H^i_{pq}/\zeta_i^2$.

H2 Matrix that contains matrix H0 divided by the squared eigenfrequency of the corresponding in-plane mode, i.e. $H2(i,p,q) = H^i_{pq}/\omega_i^2 = H^i_{pq}/\zeta_i^4$.

score_rect.m

This function loads the score file `ScoreFileName` introduced in the main script. If it is not found, the preset values are loaded. The other arguments are obtained after executing `plate_def_rect.m`.

Next, the variable `score_cell` described in section 6.10.1 is read and the functions that generate the excitation signals are called. The output parameter `f_time` corresponds to the excitation term $p(n)$ in eq. (5.8) and eq. (5.13).

Function 6.9.6 — score_rect.m.

Short description

Parser for the excitation file.

Call

[f_time, Tn] = score_rect(ScoreFileName, Lx, Ly, hd, rho, kx, ky, BC, Nphi, scheme, C, fsd, Tsd)

Input parameters

ScoreFileName Name of the file containing the excitation variable.

Lx Plate dimension X in meters.

Ly Plate dimension Y in meters.

hd Plate thickness in meters.

rho volumetric mass density.
kx Vector containing the X modal indexes of the transverse eigenmodes.
ky Vector containing the Y modal indexes of the transverse eigenmodes.
BC Type of boundary conditions at the edge. Possible values: 'SimplySupported'.
Nphi Number of transverse modes kept in the truncation.
scheme Time integration scheme. Possible values: 'ECS', 'verlet'.
C Matrix corresponding to $\left(\frac{1}{k^2}+\frac{C_{ss}}{2k}\right)$ as defined in eq. (5.10).
fsd Sampling rate.
Tsd Time length of the output signal in seconds.

Output parameters
f_time Time vector containing the excitation signal for every mode.
Tn Number of samples of the output signal.

6.10 Input file contents

6.10.1 Circular plate

Plate characteristics

This file includes the parameters used for the plate modeling. First two blocks refer to material and geometrical description. Third block, contains the details referred to the imperfection. Next group of variables concerns the damping values and last block refers to the boundary conditions at the edge.

Special remark must be made with regard to the imperfection characteristics block. When a perfect plate is being modeled, the imperfection height `H` should be set to 0. Otherwise, the imperfection profile can be described in two different ways.

a) If the linear combination of modeshapes that configure the imperfection profile is known, these should be introduced in `proj` and `modeIndices`.
b) If the projection coefficients are not known, `proj` and `modeIndices` should be left empty and the imperfection must be approximated to one of the shapes listed in `ImperfectionType`, by filling the values of `tau1`, `tau2`, `ModeType` and `error_coef`. This way, the computation of `proj` and `modeIndices` will be performed automatically by the code.

Damping can be defined manually, by defining vector `c` and leaving `X` empty or automatically, by choosing one of the presets included in function `c_preset`. When the preset option is chosen, the values of `dFac`, `dExp` and `dCons` must be also introduced.

File 6.10.1 — PlateCharacteristics-Circular.mat.

Material parameters

Name	Type	Size	Description	Recommended values
nu	Real	1×1	Poisson ratio	$[-1,0.5]$
E	Real	1×1	Young modulus in Pascal.	$(0,300\times10^9)$
rho	Real	1×1	Volumetric mass density in kg/m^3.	$(0,10000)$

Geometrical characteristics

Name	Type	Size	Description	Recommended values
Rd	Real	1×1	Dimensioned plate radius in meters	$(0, \infty)$
hd	Real	1×1	Dimensioned plate thickness in meters	$(0, O\left(10^{-3}\right))$

Imperfection characteristics

Name	Type	Size	Description	Recommended values
H	Real	1×1	Height of the imperfection in meters. Set $H = 0$ for a perfect plate.	$[0, O\left(10^{-2}\right)]$
ImperfectionType	String	1×1	Shape of the imperfection.	'Spherical' 'Parabolic'
tau2	Real	1×1	When `ImperfectionType = 'Parabolic'`, `tau2` is the parabola order.	$\left(-10^{250}, 10^{250}\right)$
ModeType	String	1×1	Type of modes considered in the approximation of the imperfection.	'All' 'Axisymmetric'
error_coef	Real	1×1	Top error admitted in the imperfection approximation.	$[0, 1]$
proj	Real	$N_p \times 1$	Projection coefficients corresponding to the N_p projected modes. Leave empty if the imperfection must be approximated according to the other parameters.	$(-100, 100)$
modeIndices	Integer	$N_p \times 1$	Indexes of the Np projected modes included in proj. Leave empty if the imperfection must be approximated according to the other parameters.	$[1, Nphi]$

Damping parameters

Name	Type	Size	Description	Recommended values
X	String	1×1	Damping value preset selector. Leave empty for manual introduction.	'Undamped' 'PowerLaw'
c	Real	$N \times 1$	Damping value vector.	$[0, \infty)$
dFac	Real	1×1	Multiplicative factor of the damping function.	$[0, \infty)$
dExp	Real	1×1	Exponent of the damping function.	$(-\infty, \infty)$
dCons	Real	1×1	Additive constant to the damping function.	$[0, \infty)$

Boundary condition parameters

Name	Type	Size	Description	Recommended values
BC	String	1×1	Type of boundary conditions at the edge.	'free' 'clamped' 'elastic'
KR	Real	1×1	Normalized rotational stiffness, $KR = Kr/D$. Where Kr is the dimensioned rotational stiffness and D is the plate flexural stiffness. Only used when BC = 'elastic'. ($KR_{free} = 0$ and $KR_{clamped} = \infty$)	$(0, \infty)$
KT	Real	1×1	Normalized transverse stiffness, $KT = Kt/D$. Where Kt is the dimensioned transverse stiffness and D is the plate flexural stiffness. Only used when BC = 'elastic'. ($KT_{free} = 0$ and $KT_{clamped} = \infty$)	$(0, \infty)$

Simulation parameters

This file includes the variables related to the simulation. The first block affects the accuracy of the results. `Nphi` corresponds to the number of transverse modes that will be considered for the modal approach, i.e. the number of modes included before the truncation of the series in eq. (2.8a). Thus, the larger `Nphi`, the wider the frequency range covered by the simulation. To ensure that the whole plate response is analyzed, `Nphi` should be set to a value such that f_{N_Φ} is higher than the maximum frequency reached with the energy cascade. Generally, to cover the plate response in the audible range `Nphi`$\in (1000, 1500)$. To test if the selected `Nphi` was the appropriate, plot the spectrogram of the results (output displacement or velocity) and check the maximum frequency reached by the response. On the other hand, `Npsi` corresponds to the number of the in-plane modes considered before the truncation of the series in eq. (2.8b). The value of `Npsi` affects to the coupling accuracy. It has been proved that convergence is reached for values of `Npsi` larger than 50, thus, it is advised to set `Npsi` $\in (50, 100)$.

`NA` determines the number of eigenfrequencies of the imperfect plate that will be computed. Thus, it is not necessary when the plate does not present any profile imperfection. When set to 0, the computation of the eigenfrequencies is skipped. Mathematically, `NA` corresponds to the size of $\mathbf{A}$ in eq. (2.18). This matrix is computed from the $\mathbf{\Gamma}$ coefficients. Given that the creation of the `Gamma` matrix requires a large amount of memory, since it is a 4D vector of size `NA`^{4}, the code offers the possibility of setting a low value for `NA` so that memory problems are avoided.

The time simulation parameters include the election of the time stepping scheme, either `'ECS'` for the energy conserving scheme or `'verlet'` for the Störmer-Verlet scheme. The sampling frequency `fsd` must be set above the stability limit, i.e. $f_s \geq \pi f_{N_\Phi}$. The recommended value is three times the stability limit. `Tsd` determines the time-length of the output signal.

Finally, the geometrical parameters are used to specify the accuracy of spatial discretization in some routines. The values of `Nr` and `Nth` are used in routines such as the computation of the mode shapes or the construction of the imperfection profile. Values around 500 provide enough resolution for the typical computations. On the other hand, `dr_H` corresponds to the discretization step used for the computation of the `H` tensor. The recommended value for this parameter is `dr_H=1e-4`. The

last input argument corresponds to the group of output points. The number of output points is limited by the computer capacity. If a large amount of outputs is needed, consider decreasing the simulation time or the sampling frequency.

File 6.10.2 — SimulationParameters-Circular.mat.

Accuracy parameters

Name	Type	Size	Description	Recommended values
Nphi	Integer	1×1	Number of transverse modes included in the truncation	$(0, 1500)$
Npsi	Integer	1×1	Number of in-plane modes included in the truncation.	$(50, 100)$
NA	Integer	1×1	Number of modes considered to compute the eigenfrequencies of the imperfect plate.	$[0, Nphi]$

Time simulation parameters

Name	Type	Size	Description	Recommended values
scheme	String	1×1	Type of time stepping scheme	'ECS' 'verlet'
fsd	Real	1×1	Dimensioned sampling rate in seconds	$3\pi f_{N_\Phi}$
Tsd	Real	1×1	Dimensioned simulation time in seconds	$(0, \infty)$

Geometrical model parameters

Names	Type	Size	Description	Recommended values
Nr	Integer	1×1	Number of discretization points for the radius variable r.	$O(500)$
Nth	Integer	1×1	Number of discretization points for the angle variable θ.	$O(500)$
dr_H	Real	1×1	Discretization step used for computing the H coefficients	1×10^{-4}
op	Real	$N_{op} \times 2$	Matrix containing the N_{op} output points where the displacement and velocity must be predicted. First column corresponds to the angle and second column to the normalized radius. $op(i,:) = [\theta_i,\ r_i/Rd]$.	$[(0, 2\pi), [0, 1]]$

Score parameters

The score parameters file includes a single variable `score_cell` that contains all the excitations that will be input to the plate. There are three possible types, 'Strike' for a raised cosine excitation reproducing the strike of a mallet, 'Harmonic' for a sinusoidal input and 'ColoredNoise' for a wide band input.

Each type is accompanied with a specific set of values. The common parameter `T0` defines the

time shift between the start of the current excitation and the previous one. This way, first input starts at time $t_{0,1} = T0_1$, second excitation at time $t_{0,2} = T0_1 + T0_2$ and so on. The rest of the parameters are described below. For a detailed description of every type of excitation refer to Section 6.6.

File 6.10.3 — ScoreParameters-Circular.mat.

Name	Type	Size	Description	Recommended values
score_cell	Cell	$Ns \times 2$	List of excitation inputs. $score\_cell(i,:) = [ExcitationType_i,\ Parameters_i]$	'Strike' 'Harmonic' 'ColoredNoise'

***score_cell* syntax**

Excitation Type	Parameters
'Strike'	[T0 fm Twid fp_th fp_r]
'Harmonic'	{T0 [Frequency] [Amplitude] Phase [Times] fp_th fp_r}
'ColoredNoise'	{Color [T0 Amplitude fmin fmax deltaf TimeLength fp_th fp_r]}

Parameters

Name	Type	Size	Description	Recommended values
T0	Real	1×1	Initial time delay. $t_{0,i} = \sum_{j=1}^{i} T0_j$	$(0, Tsd - t_{0,i-1})$
fm	Real	1×1	Strike amplitude in Newton.	$(0, 1000)$
Twid	Real	1×1	Raised cosine half time width	$(0, Tsd - 2Twid)$
fp_th	Real	1×1	Angular component of the excitation point.	$(0, 2\pi)$
fp_r	Real	1×1	Radial component of the excitation point in normalized units.	$[0, 1]$
Times	Real	$N_T \times 1$	Vector containing the times where the harmonic signal has a variation of amplitude or frequency. If $N_T > 1$, the initial value of this vector should always be 0.	$(0, Tsd - T0)$
Frequency	Real	$N_T \times 1$	Vector containing the frequencies at every time position introduced in `Times`.	$(0, fsd/2)$
[Amplitude]	Real	$N_T \times 1$	Vector containing the amplitudes at every time position introduced in `Times`.	$(0, \infty)$
Phase	Real	1×1	Initial phase of the harmonic signal	$(0, 2\pi)$
Amplitude	Real	1×1	Amplitude of the colored noise signal.	$(0, 1000)$
TimeLength	Real	1×1	Duration of the excitation.	$(0, Tsd - T0)$
Color	String	1×1	Color of the noise	'White' 'Pink' 'Blue' 'Red' 'Purple'
fmin	Real	1×1	Lower frequency of the noise signal spectrum.	$(0, fsd/2)$
fmax	Real	1×1	Upper frequency of the noise signal spectrum.	$(0, fsd/2)$
deltaf	Real	1×1	Frequency spacing between to consecutive frequencies on the generation of the noise signal.	$(0, fmax - fmin)$

6.10.2 Rectangular plate

The variables included in the input files for rectangular plates are analogous to those for circular plates with the exception of the coordinates. Whereas circular plates are defined in polar coordinates, rectangular ones are described with Cartesian. Refer to section 6.10.1 for further explanations.

Plate characteristics

File 6.10.4 — PlateCharacteristics-Rectangular.mat.

Material parameters

Name	Type	Size	Description	Recommended values
nu	Real	1×1	Poisson ratio	$[-1, 0.5]$
E	Real	1×1	Young modulus in Pascal.	$(0, 300 \times 10^9)$
rho	Real	1×1	Volumetric mass density in kg/m^3.	$(0, 10000)$

Geometrical characteristics

Name	Type	Size	Description	Recommended values
Lx	Real	1×1	Plate dimension X in meters	$(0, \infty)$
Ly	Real	1×1	Plate dimension Y in meters	$(0, \infty)$
hd	Real	1×1	Dimensioned plate thickness in meters	$(0, O(10^{-3}))$

Imperfection characteristics

Name	Type	Size	Description	Recommended values
H	Real	1×1	Height of the imperfection in meters.	$(0, O(10^{-2}))$
ImperfectionType	String	1×1	Shape of the imperfection.	'2DRaisedCosine'
xWidth	Real	1×1	Half-width of the raised cosine shape in direction X	$(0, Lx/2)$
yWidth	Real	1×1	Half-width of the raised cosine shape in direction Y	$(0, Ly/2)$
ModeType	String	1×1	Type of modes considered in the approximation of the imperfection.	'All'
error_coef	Real	1×1	Top error admitted in the imperfection approximation.	$[0, 1]$
proj	Real	$N_p \times 1$	Projection coefficients corresponding to the N_p projected modes. Leave empty if the imperfection must be approximated according to the other parameters.	$(-\infty, \infty)$
modeIndices	Integer	$N_p \times 1$	Indexes of the Np projected modes included in proj. Leave empty if the imperfection must be approximated according to the other parameters.	$[1, Nphi]$

Damping parameters

Name	Type	Size	Description	Recommended values
X	String	1×1	Damping value preset selector. Leave empty for manual introduction.	'Undamped' 'PowerLaw'
c	Real	$N \times 1$	Damping value vector.	$[0, \infty)$

dFac	Real	1×1	Multiplicative factor of the damping function.	$[0, \infty)$
dExp	Real	1×1	Exponent of the damping function.	$(-\infty, \infty)$
dCons	Real	1×1	Additive constant to the damping function.	$[0, \infty)$

Boundary condition parameters

Name	Type	Size	Description	Recommended values
BC	String	1×1	Type of boundary conditions at the edge.	'SimplySupported'

Simulation parameters

File 6.10.5 — SimulationParameters-Rectangular.mat.

Accuracy parameters

Name	Type	Size	Description	Recommended values
Nphi	Integer	1×1	Number of transverse modes included in the truncation	$(0, O(10^3))$
Npsi	Integer	1×1	Number of in-plane modes included in the truncation.	$(0, O(10))$
NA	Integer	1×1	Number of modes considered to compute the eigenfrequencies of the imperfect plate.	$(0, O(10^2))$

Time simulation parameters

Name	Type	Size	Description	Recommended values
scheme	String	1×1	Type of time stepping scheme	'ECS' 'verlet'
fsd	Real	1×1	Dimensioned sampling rate in seconds	$3\pi f_{N_\Phi}$
Tsd	Real	1×1	Dimensioned simulation time in seconds	$(0, \infty)$

Geometrical model parameters

Names	Type	Size	Description	Recommended values
Nx	Integer	1×1	Number of discretization points for the x dimension.	$(0, \infty)$
Ny	Integer	1×1	Number of discretization points for the y dimension.	$(0, \infty)$
op	Real	$N_{op} \times 2$	Matrix containing the N_{op} output points where the displacement and velocity must be predicted. $op(i,:) = [x_i,\ y_i]$.	$[[0, Lx], [0, Ly]]$

Score parameters

File 6.10.6 — ScoreParameters-Rectangular.mat.

Name	Type	Size	Description	Recommended values
score_cell	Cell	$Ns \times 2$	List of excitation inputs. $score\_cell(i,:) = [ExcitationType_i,\ Parameters_i]$	'Strike' 'Harmonic' 'ColoredNoise'

***score_cell* syntax**

Excitation Type	Parameters
'Strike'	[T0 fm Twid fp_x fp_y]
'Harmonic'	[T0 [Frequency] [Amplitude] Phase [Times] fp_x fp_y]
'ColoredNoise'	{Color [T0 Amplitude fmin fmax deltaf TimeLength fp_x fp_y]}

Parameters

Name	Type	Size	Description	Recommended values
T0	Real	1×1	Initial time delay. $t_{0,i} = \sum_{j=1}^{i} T0_j$	$(0, Tsd - t_{0,i-1})$
fm	Real	1×1	Strike amplitude in Newton.	$(0, 1000)$
Twid	Real	1×1	Raised cosine half time width	$(0, Tsd - 2Twid)$
fp_x	Real	1×1	x component of the excitation point.	$(0, Lx)$
fp_y	Real	1×1	y component of the excitation point.	$[0, Ly]$
Times	Real	$N_T \times 1$	Vector containing the times where the harmonic signal has a variation of amplitude or frequency. If $N_T > 1$, the initial value of this vector should always be 0.	$(0, Tsd - T0)$
Frequency	Real	$N_T \times 1$	Vector containing the frequencies at every time position introduced in `Times`.	$(0, fsd/2)$
[Amplitude]	Real	$N_T \times 1$	Vector containing the amplitudes at every time position introduced in `Times`.	$(0, \infty)$
Phase	Real	1×1	Initial phase of the harmonic signal	$(0, 2\pi)$
Amplitude	Real	1×1	Amplitude of the colored noise signal.	$(0, 1000)$
TimeLength	Real	1×1	Duration of the excitation.	$(0, Tsd - T0)$
Color	String	1×1	Color of the noise	'White' 'Pink' 'Blue' 'Red' 'Purple'
fmin	Real	1×1	Lower frequency of the noise signal spectrum.	$(0, fsd/2)$
fmax	Real	1×1	Upper frequency of the noise signal spectrum.	$(0, fsd/2)$
deltaf	Real	1×1	Frequency spacing between to consecutive frequencies on the generation of the noise signal.	$(0, fmax - fmin)$

6.10.3 Common

Gamma tensor file

The Gamma file contains the coupling coefficient Γ. Preferably, this tensor should be computed before the execution of the code. If the file does not exist, the matrix *Gamma* will be computed during the performance of the program.

File 6.10.7 — Gamma.mat.

Name	Type	Size	Description	Recommended values
Gamma	Real	$NA \times NA \times NA \times NA$	Gamma coefficients $Gamma[s,p,q,r] = \Gamma^s_{pqr}$	$(-\infty, \infty)$

7. C++ code

The C++ code has been finalized recently and must be considered as a beta preliminary version. In particular the Verlet scheme has shown some instabilities in the preliminary tests that is not yet resolved. The documentation is still under preparation and should be ready in summer 2017.

IV

Example cases

8. Matlab cases

8.1 Case C1: Perfect circular plate with free edge

8.1.1 Input parameters

File 8.1.1 — PlateCharacteristics-CaseC1.mat.

Material parameters
- **nu** 0.38
- **E** 2e11
- **rho** 7860

Geometrical characteristics
- **Rd** 0.4
- **hd** 1e-3

Imperfection characteristics
- **H** 0
- **proj** []
- **Imperfection type** []
- **tau2** 0
- **ModeType** []
- **modeIndices** []
- **error_coef** []

Damping characteristics
- **X** 'PowerLaw'
- **c** []
- **dFac** 0.05
- **dExp** 0.6
- **dCons** 0.5

Boundary condition parameters
- **BC** 'free'
- **KR** 0

KT 0

File 8.1.2 — SimulationParameters-CaseC1.mat.
Accuracy parameters
Nphi 1000
Npsi 100
NA 0
Accuracy parameters
scheme 'ECS'
fsd 0
Tsd 10
Accuracy parameters
Nr 400
Nth 400
dr_H 1e-4
op [1.0472, 0.8]

File 8.1.3 — ScoreParameters-Circular-CaseC1.mat.
Excitation type 'Strike'
T0 0.1
fm 200
Twid 0.005
fp_th 0.7854
fp_r 0.9

8.1.2 Results

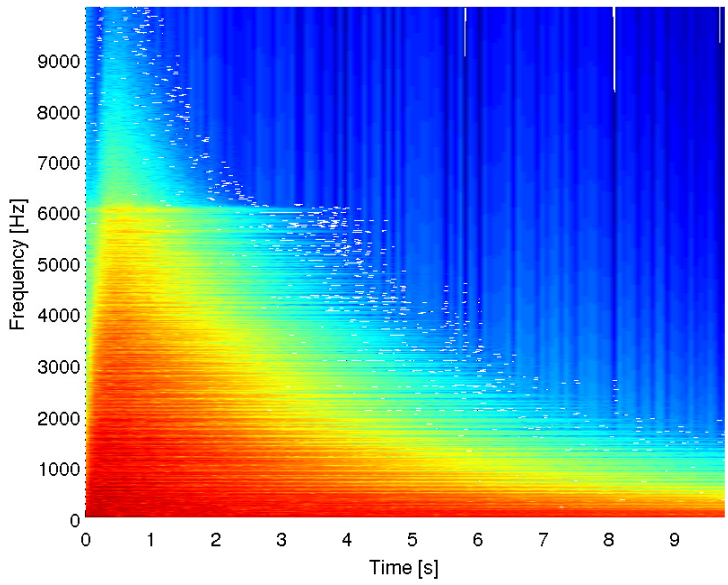


Figure 8.1: Spectrogram of the velocity at the output point in case C1.

8.2 Case C2: Perfect circular plate with clamped edge

8.2.1 Input parameters

File 8.2.1 — PlateCharacteristics-CaseC2.mat.

Material parameters
- **nu** 0.3
- **E** 2e11
- **rho** 7860

Geometrical characteristics
- **Rd** 0.2
- **hd** 0.8e-3

Imperfection characteristics
- **H** 0
- **proj** []
- **Imperfection type** []
- **tau2** 0
- **ModeType** []
- **modeIndices** []
- **error_coef** []

Damping characteristics
- **X** 'PowerLaw'
- **c** []
- **dFac** 0.008
- **dExp** 0.6
- **dCons** 0

Boundary condition parameters
- **BC** 'clamped'
- **KR** 0
- **KT** 0

File 8.2.2 — SimulationParameters-CaseC2.mat.

Accuracy parameters
- **Nphi** 600
- **Npsi** 100
- **NA** 0

Accuracy parameters
- **scheme** 'ECS'
- **fsd** 0
- **Tsd** 10

Accuracy parameters
- **Nr** 400
- **Nth** 400
- **dr_H** 1e-4
- **op** [1.0472, 0.2]

File 8.2.3 — ScoreParameters-Circular-CaseC2.mat.

Excitation type 'Strike'

T0 0.1

fm 250
Twid 0.005
fp_th 0.7854
fp_r 0.35

8.2.2 Results

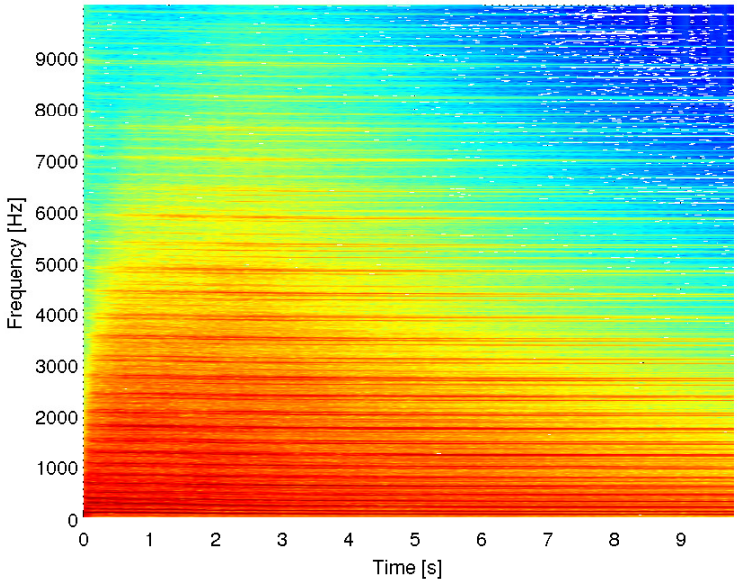


Figure 8.2: Spectrogram of the velocity at the output point in case C2.

8.3 Case C3: Imperfect spherical circular plate with elastic edge

8.3.1 Input parameters

File 8.3.1 — PlateCharacteristics-CaseC3.mat.
Material parameters
nu 0.38
E 2e11
rho 7860
Geometrical characteristics
Rd 0.4
hd 1e-3
Imperfection characteristics
H 0.02
proj []
Imperfection type 'Spherical'
tau2 0
ModeType 'Axisymmetric'
modeIndices []
error_coef 0.1
Damping characteristics

X 'PowerLaw'
c []
dFac 0.008
dExp 0.6
dCons 0
Boundary condition parameters
BC 'elastic'
KR 10
KT 1000

File 8.3.2 — SimulationParameters-CaseC3.mat.
Accuracy parameters
Nphi 800
Npsi 60
NA 100
Accuracy parameters
scheme 'verlet'
fsd 0
Tsd 10
Accuracy parameters
Nr 400
Nth 400
dr_H 1e-4
op [1.0472, 0.2]

File 8.3.3 — ScoreParameters-Circular-CaseC3.mat.
Excitation type 'Strike'
T0 0.1
fm 200
Twid 0.005
fp_th 0.7854
fp_r 0.5167

8.3.2 Results

8.4 Case R5: Imperfect spherical circular plate with elastic edge

8.4.1 Input parameters

File 8.4.1 — PlateCharacteristics-CaseR5.mat.
Material parameters
nu 0.3
E 2e11
rho 7860
Geometrical characteristics
Lx 0.6
Ly 0.6
hd 0.8e-3
Imperfection characteristics
H 0

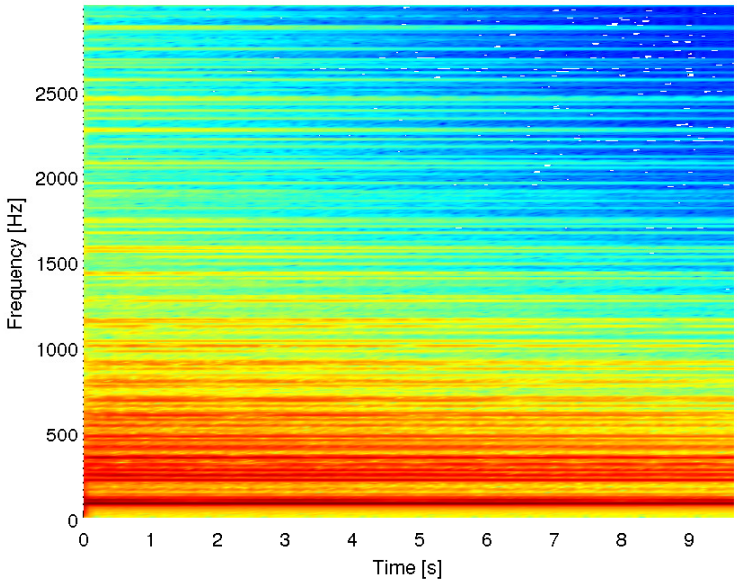


Figure 8.3: Spectrogram of the velocity at the output point in case C2.

proj []
Imperfection type []
xWidth 0
yWidth 0
ModeType 'All'
modeIndices []
error_coef []
Damping characteristics
X 'PowerLaw'
c []
dFac 0.008
dExp 0.6
dCons 0.05
Boundary condition parameters
BC 'SimplySupported'

File 8.4.2 — SimulationParameters-CaseR5.mat.
Accuracy parameters
Nphi 800
Npsi 60
NA 100
Accuracy parameters
scheme 'verlet'
fsd 0
Tsd 5
Accuracy parameters
Nx 400

Ny 400
op [1.0472, 0.2]

File 8.4.3 — ScoreParameters-Circular-CaseR5.mat.
Excitation type 'Strike'
T0 0.1
fm 10
Twid 0.005
fp_x 0.1425
fp_y 0.0963
Excitation type 'Strike'
T0 1
fm 30
Twid 0.005
fp_x 0.1425
fp_y 0.0963

8.4.2 Results

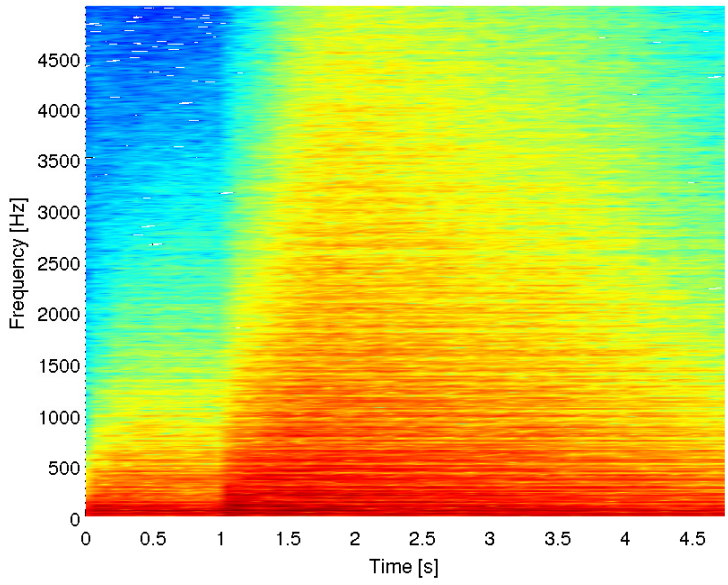


Figure 8.4: Spectrogram of the velocity at the output point in case C2.

8.5 Case CT1: Perfect circular plates with varying boundary conditions at the edge

8.5.1 Input parameters

File 8.5.1 — PlateCharacteristics-CaseCT1.mat.
Material parameters
nu 0.38
E 2e11

rho 7860
Geometrical characteristics
Rd 0.4
hd 1e-3
Imperfection characteristics
H 0
proj []
Imperfection type []
tau2 0
ModeType []
modeIndices []
error_coef []
Damping characteristics
X 'PowerLaw'
c []
dFac 0.05
dExp 0.6
dCons 0.5
Boundary condition parameters
BC 'free', 'clamped', 'elastic'
KR {0, 0, 100, 0, 1000}
KT {0, 0, 0, 1000, 10000}

File 8.5.2 — SimulationParameters-CaseCT1.mat.
Accuracy parameters
Nphi 1000
Npsi 60
NA 10
Accuracy parameters
scheme 'verlet'
fsd 0
Tsd 10
Accuracy parameters
Nr 400
Nth 500
dr_H 1e-4
op [0.5192, 0.8962]

File 8.5.3 — ScoreParameters-Circular-CaseCT1.mat.
Strikes 1-4
Excitation type 'Strike'
T0 {0.1, 0.6, 0.6, 0.5}
fm 0.1
Twid 7e-4
fp_th 3.1
fp_r 0.9
Strike 5

Excitation type 'Strike'
T0 0.1
fm 0.5
Twid 2e-3
fp_th 3.1
fp_r 0.9

Strikes 6-7
Excitation type 'Strike'
T0 {0.6, 0.5}
fm 0.1
Twid 7e-4
fp_th 3.1
fp_r 0.9

Strike 8
Excitation type 'Strike'
T0 0.1
fm 0.5
Twid 2e-3
fp_th 3.1
fp_r 0.9

8.5.2 Results

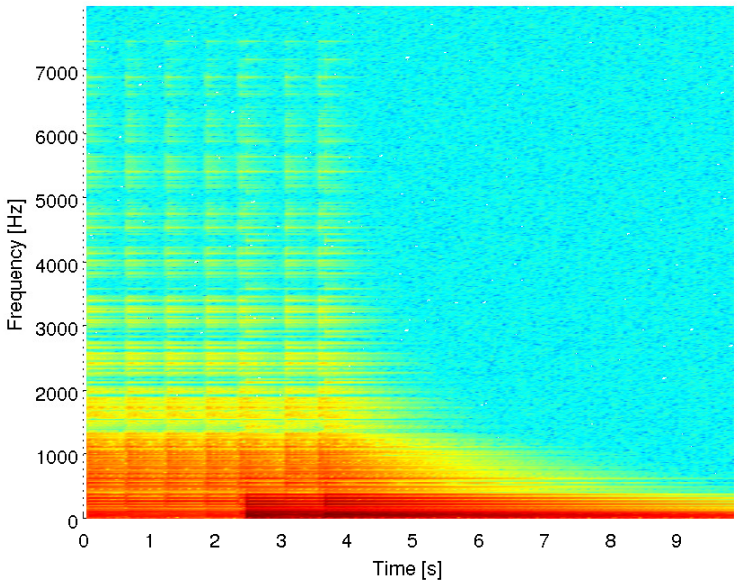


Figure 8.5: Spectrogram of the velocity at the output point in case CT1: Free edge.

8.6 Case CT3: Perfect circular plates with free edge varying thickness

8.6.1 Input parameters

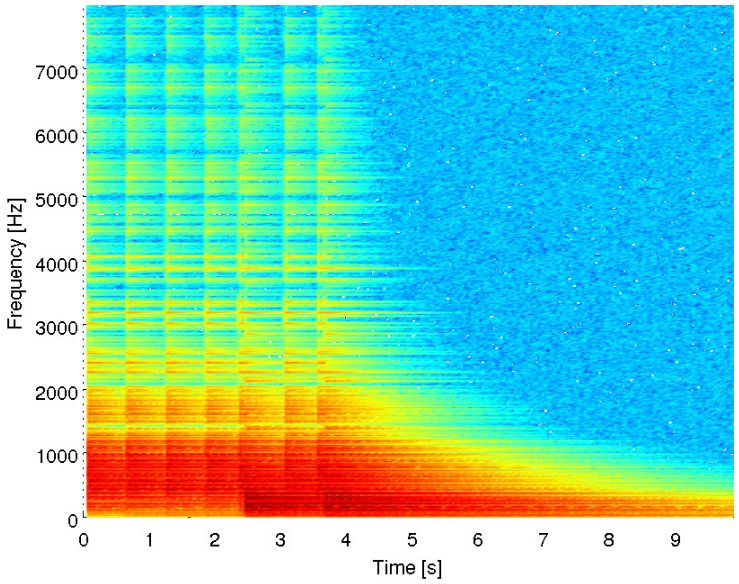


Figure 8.6: Spectrogram of the velocity at the output point in case CT1: Clamped edge.

File 8.6.1 — PlateCharacteristics-CaseCT3.mat.

Material parameters
- **nu** 0.38
- **E** 1e11
- **rho** 7860

Geometrical characteristics
- **Rd** 0.3
- **hd** { 0.3e-3, 0.6e-3, 1e-3, 2e-3, 3e-3}

Imperfection characteristics
- **H** 0
- **proj** []
- **Imperfection type** []
- **tau2** 0
- **ModeType** []
- **modeIndices** []
- **error_coef** []

Damping characteristics
- **X** 'PowerLaw'
- **c** []
- **dFac** 0.01
- **dExp** 0.6
- **dCons** 0.5

Boundary condition parameters
- **BC** 'free'
- **KR** 0
- **KT** 0

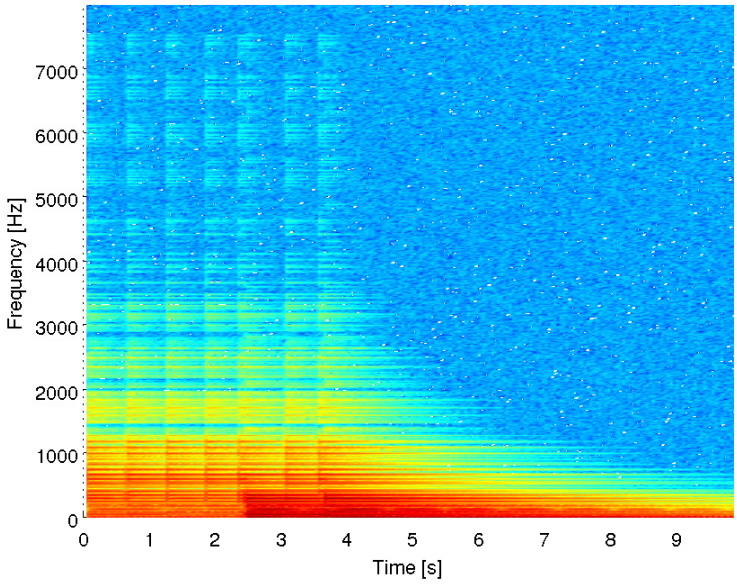


Figure 8.7: Spectrogram of the velocity at the output point in case CT1: Elastic edge with KR = 100 and KT = 0.

File 8.6.2 — SimulationParameters-CaseCT3.mat.
Accuracy parameters
Nphi {1000, 1000, 1000, 800, 600}
Npsi 60
NA 10
Accuracy parameters
scheme 'verlet'
fsd 0
Tsd 10
Accuracy parameters
Nr 400
Nth 500
dr_H 1e-4
op [0.5192, 0.8962]

File 8.6.3 — ScoreParameters-Circular-CaseCT3.mat.
Strikes 1
Excitation type 'Strike'
T0 0.1
fm 0.5
Twid 1e-3
fp_th 3.1
fp_r 0.1
Strike 2
Excitation type 'Strike'
T0 1

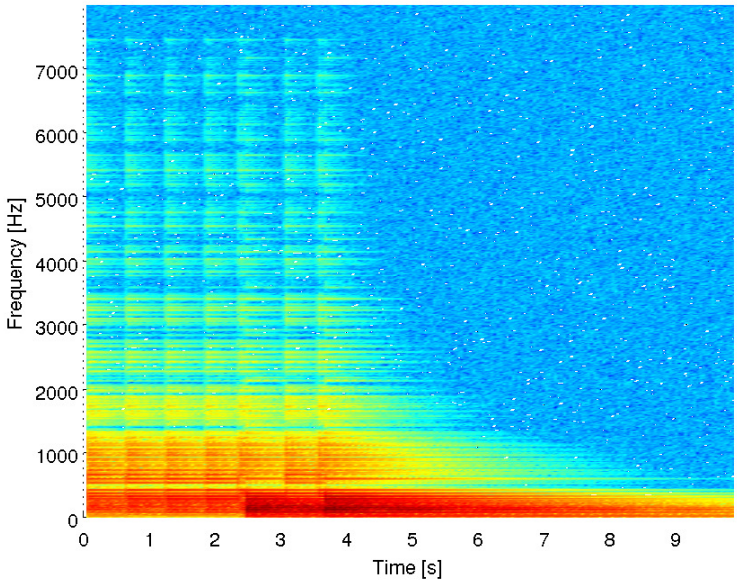


Figure 8.8: Spectrogram of the velocity at the output point in case CT1: Elastic edge with KR = 0 and KT = 1000.

fm 1
Twid 2e-3
fp_th 3.1
fp_r 0.5
Strike 3
Excitation type 'Strike'
T0 1
fm 2
Twid 3e-3
fp_th 3.1
fp_r 0.9

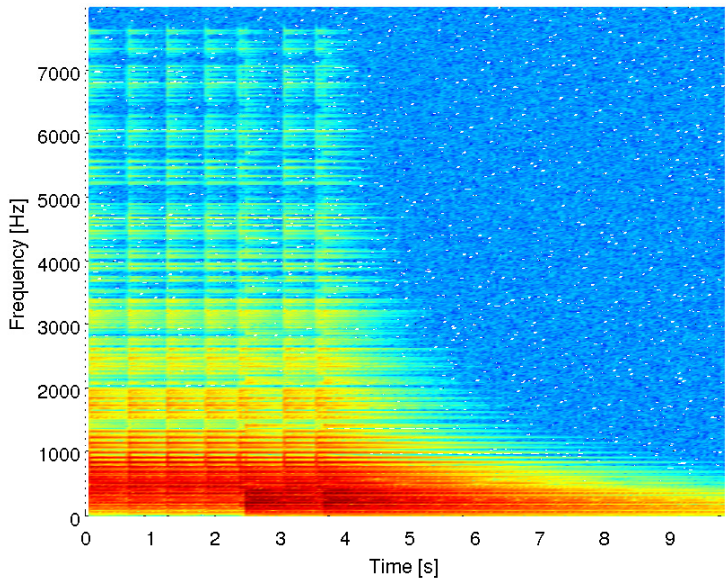


Figure 8.9: Spectrogram of the velocity at the output point in case CT1: Elastic edge with KR = 1000 and KT = 10000.

Bibliography

[1] M Amabili. “Nonlinear vibrations of rectangular plates with different boundary conditions: theory and experiments”. In: *Computers & structures* 82.31 (2004), pages 2587–2605 (cited on page 36).

[2] Z.P. Bazant and L. Cedolin. *Stability of structures*. third edition. Singapore: World Scientific, 2010 (cited on page 17).

[3] S. Bilbao. “A family of conservative finite difference schemes for the dynamical von Karman plate equations”. In: *Numerical Methods for Partial Differential Equations* 24.1 (2008), pages 193–216 (cited on pages 39–41).

[4] A. Boudaoud et al. “Observation of Wave Turbulence in Vibrating Plates”. In: *Physical Review Letters* 100 (2008), page 234504 (cited on page 9).

[5] O. Cadot et al. “Wave turbulence in vibrating plates”. In: *Handbook of applications of chaos theory*. Edited by C. H. Skiadas and C. Skiadas. Chapman and Hall/CRC, 2016 (cited on page 9).

[6] C. Camier. “Modélisation et étude numérique des vibrations non-linéaires de plaques circulaires minces imparfaites: application aux cymbales”. PhD thesis. Ecole Polytechnique X, 2009 (cited on pages 10, 23).

[7] C. Camier, C. Touzé, and O. Thomas. “Non-linear vibrations of imperfect free-edge circular plates and shells”. In: *European Journal of Mechanics-A/Solids* 28.3 (2009), pages 500–515 (cited on pages 18, 20, 21, 23).

[8] C.Y. Chia. *Nonlinear analysis of plates*. McGraw-Hill International Book Company, 1980 (cited on page 17).

[9] M. Ducceschi. “Nonlinear Vibrations of Thin Rectangular Plates. A numerical investigation with application to wave turbulence and sound synthesis.” PhD thesis. ENSTA-ParisTech, 2014 (cited on pages 10, 36, 41).

[10] M. Ducceschi and C. Touzé. “Modal approach for nonlinear vibrations of damped impacted plates: Application to sound synthesis of gongs and cymbals”. In: *Journal of Sound and Vibration* 344 (2015), pages 313–331 (cited on pages 10, 12, 17, 18, 20, 36, 41, 60, 76).

[11] M. Ducceschi et al. “Dynamics of the wave turbulence spectrum in vibrating plates: A numerical investigation using a conservative finite difference scheme”. In: *Physica D* 280-281 (2014), pages 73–85 (cited on page 9).

[12] M. Ducceschi et al. “Nonlinear dynamics of rectangular plates: investigation of modal interaction in free and forced vibrations”. In: *Acta Mechanica* 225.1 (2014), pages 213–232 (cited on pages 24, 36, 39–41).

[13] G. Düring, C. Josserand, and S. Rica. “Weak Turbulence for a Vibrating Plate: Can One Hear a Kolmogorov Spectrum?” In: *Physical Review Letters* 97 (2006), page 025503 (cited on page 9).

[14] E. Hairer, C. Lubich, and G. Wanner. *Geometric numerical integration: structure-preserving algorithms for Ordinary differential equations*. second edition. Springer, 2006 (cited on page 12).

[15] T. Humbert et al. “Wave turbulence in vibrating plates: The effect of damping”. In: *EPL (Europhysics Letters)* 102.3 (2013), page 30002 (cited on page 9).

[16] T. von Kármán. “Festigkeitsprobleme im Maschinenbau”. In: *Encyklopädie der Mathematischen Wissenschaften* 4.4 (1910), pages 311–385 (cited on page 17).

[17] L.D. Landau and E.M. Lifschitz. *Theory of Elasticity*. third edition. Elsevier Butterworth Heinemann, 1986 (cited on page 17).

[18] J. Frelat M. Monteil O. Thomas and C. Touzé. “Effet de la mise en forme sur les vibrations d’une coque mince : application au steelpan”. In: *Proceedings of the 11th colloque National en Calcul de structures, CSMA 2013, Giens*. 2013 (cited on page 12).

[19] B. Miquel, A. Alexakis, and N. Mordant. “The role of dissipation in flexural wave turbulence: from experimental spectrum to Kolmogorov-Zakharov spectrum”. In: *Physical Review E* 89.6 (2014), page 062925 (cited on page 9).

[20] M. Monteil. “Comportement vibratoire du steelpan : effet des procédés de fabrication et dynamique non linéaire”. PhD thesis. ENSTA ParisTech, 2013 (cited on page 12).

[21] N. Mordant. “Are there waves in elastic wave turbulence ?” In: *Physical Review Letters* 100 (2008), page 234505 (cited on page 9).

[22] A. H. Nayfeh and D. T. Mook. *Nonlinear oscillations*. New-York: John Wiley & sons, 1979 (cited on page 17).

[23] G. L. Ostiguy and S. Sassi. “Effects of initial geometric imperfections on dynamic behavior of rectangular plates”. In: *Nonlinear Dynamics* 3.3 (1992), pages 165–181 (cited on page 18).

[24] O. Thomas. “Analyse et modélisation de vibrations non-linéaires de milieux minces élastiques. Application aux instruments de percussion (Analysis and modelisation of nonlinear vibrations of thin elastic media. Application to nonlinear percussion instruments)”. PhD thesis. Université Pierre et Marie Curie, Paris VI, 2001 (cited on page 9).

[25] O. Thomas and S. Bilbao. “Geometrically nonlinear flexural vibrations of plates: In-plane boundary conditions and some symmetry properties”. In: *Journal of Sound and Vibration* 315.3 (2008), pages 569–590 (cited on pages 17, 18, 20, 24, 25, 35).

[26] C. Touzé. “Analyse et modélisation de signaux acoustiques et vibratoires chaotiques. Application aux intruments de percussion non-linéaires (Analysis and modelisation of vibratory and acoustics chaotic signals. Application to nonlinear percussion instruments)”. PhD thesis. Université Pierre et Marie Curie, Paris VI, 2000 (cited on page 9).

[27] C. Touzé, S. Bilbao, and O. Cadot. “Transition scenario to turbulence in thin vibrating plates”. In: *Journal of Sound and Vibration* 331.2 (2012), pages 412–433 (cited on page 9).

[28] C. Touzé, O. Thomas, and M. Amabili. “Transition to chaotic vibrations for harmonically forced perfect and imperfect circular plates”. In: *International Journal of Non-linear Mechanics* 46.1 (2011), pages 234–246 (cited on page 9).

[29] C. Touzé, O. Thomas, and A. Chaigne. “Asymmetric non-linear forced vibrations of free-edge circular plates. Part 1: Theory”. In: *Journal of Sound and Vibration* 258.4 (2002), pages 649–676 (cited on pages 9, 17, 24).

[30] A. Zagrai and D. Donskoy. “A “soft table” for the natural frequencies and modal parameters of uniform circular plates with elastic edge support”. In: *Journal of sound and vibration* 287.1 (2005), pages 343–351 (cited on page 24).