Escuela Técnica Superior de Ingenieros Industriales - Universidad Politécnica de Madrid

Proyecto Fin de Carrera

Nonlinear Normal Modes in Mechanical Systems: Concept and Computation with Numerical Continuation

Autor:

Alejandro Silva Bernárdez

Director del Proyecto:

Dr. Alejandro Zarzo Altarejos

Un Proyecto remitido para cumplir con los requisitos de la obtención del título de Ingeniero Industrial, especialidad de Mecánica - Máquinas

March 2016

This text has been written in LaTeX
with a template from the website
www.latextemplates.com

Abstract

Nonlinear Normal Modes in Mechanical Systems: Concept and Computation with Numerical Continuation

by Alejandro Silva Bernárdez

The purpose of this Project is, first and foremost, to disclose the topic of nonlinear vibrations and oscillations in mechanical systems and, namely, nonlinear normal modes –NNMs– to a greater audience of researchers and technicians. To do so, first of all, the dynamical behavior and properties of nonlinear mechanical systems is outlined from the analysis of a pair of exemplary models with the harmonic balanced method. The conclusions drawn are contrasted with the Linear Vibration Theory. Then, it is argued how the nonlinear normal modes could, in spite of their limitations, predict the frequency response of a mechanical system. After discussing those introductory concepts, I present a Matlab package called 'NNMcont' developed by a group of researchers from the University of Liège. This package allows the analysis of nonlinear normal modes of vibration in a range of mechanical systems as extensions of the linear modes. This package relies on numerical methods and a 'continuation algorithm' for the computation of the nonlinear normal modes of a conservative mechanical system. In order to prove its functionality, a two degrees of freedom mechanical system with elastic nonlinearities is analized. This model comprises a mass suspended on a foundation by means of a spring-viscous damper mechanism –analogous to a very simplified model of most suspended structures and machines– that has attached a mass damper as a passive vibration control system. The results of the computation are displayed on frequency-energy plots showing the NNMs branches along with modal curves and time-series plots for each normal mode. Finally, a critical analysis of the results obtained is carried out with an eye on devising what they can tell the researcher about the dynamical properties of the system.

El propósito de este trabajo es, en primer lugar, mostrar el concepto de las vibraciones y oscilaciones no lineales en sistemas mecánicos y, concretamente, los modos de vibración no lineales –NNMs– a una amplia audiencia de investigadores y técnicos. Para ello, en primer lugar se analiza el comportamiento dinámico y las propiedades de los sistemas mecánicos no lineales tomando dos modelos de ejemplo que son analizados mediante el método del balance harmónico. Las conclusiones tomadas del análisis son contrastadas con la teoría lineal de vibraciones. Después, se discute cómo los modos normalen no lineales, a pesar de sus limitaciones, pueden predecir la respuesta en frecuencia de un sistema mecánico. Una vez que he discutido estos conceptos introductorios, presento un paquete de Matlab llamado 'NNMcont' desarrollado por un grupo de investigadores de la Universidad de Lieja. Este paquete permite el estudio de los modos de vibración no lineales de un rango de sistemas mecánicos. Este paquete emplea métodos numéricos y un algoritmo de continuación para computar los modos normales no lineales de un sistema mecánico conservativo como extensiones de los modos lineales. Para probar su funcionalidad, el modelo de un sistema mecánico de dos grados de libertad con nolinealidades elásticas es analizado numericamente. Dicho sistema se compone de una masa suspendida sobre un soporte fijo mediante un conjunto muelle-amortiguador –equivalente a un modelo muy simplificado de numerosas máquinas y estructuras suspendidas– que se encuentra unida a una masa amortiguadora como sistema de control pasivo de la vibración. Los resultados son mostrados mediante diagramas frecuencia-energía que muestren las diferentes ramas de modos normales del sistema junto con los diagramas de curvas modales y de series temporales de cada modo. Finalmente se lleva a cabo el análisis crítico de los resultados con la mirada puesta en comprender lo que éstos pueden indicar al investigador sobre el comportamiento dinámico del sistema.

Keywords: Nonlinear Normal Modes, Mechanical Systems, Vibration Theory, Modal Analysis, Nonlinear Dynamics, Harmonic Balance, Numerical Continuation, Matlab, NNMcont.

Acknowledgements

A mi tutor, Dr. Alejandro Zarzo Altarejos, cuya experiencia y conocimientos, reflejados en su ayuda y sus consejos, me han guiado a lo largo de los últimos pasos de este viaje. Han sido horas de reuniones, conversaciones y discursiones constructivas de las que he aprendido y tomado ejemplo. Acabo el proyecto mucho más sabio, mejor persona y con una mente más abierta gracias a él, y también, de forma tal vez indirecta, mucho más paciente.

Al Dr. Manuel Álvarez Fernández, profesor del Departamento de Matemáticas Aplicadas de la ETSII, y desde un lugar más lejano, al Dr. Gaëtan Kerschen, catedrático de Ingeniería Mecánica y Aeroespacial de la Universidad de Lieja. Sin ellos hubiera sido imposible que este Proyecto llegase a buen puerto, pues con su ayuda en momentos puntuales me han indicado el camino a seguir y los límites a los que podía llegar con el tema elegido. También, a los Drs. Jose Luís Muñoz Sanz y Juan Manuel Muñoz Guijosa, del Departamento de Ingeniería Mecánica de la ETSII, por el interés mostrado en mi trabajo.

A todos los profesores que me han acompañado y formado a lo largo de tantos años de estudio. Me quedo con un pedacito de cada uno de ellos y de sus clases.

A todas las personas con las que haya compartido un lazo dentro de esta Escuela. Aunque sólo pueda contar con los dedos de una mano aquellos con los que haya construido una profunda amistad, de ellos mantendré los recuerdos de grandes momentos vividos. Éstas son las personas que han dado auténtico significado a mi vida, y aunque mis rarezas sean en ocasiones obstáculo, saber que hay gente al otro lado que confía en mí y en la que yo pueda confiar me mantiene vivo.

Por esto último, he de destacar de entre todos a mi familia. Si ahora soy quien soy es porque ellos sabían de qué era capaz, y me cuidaron, me aportaron disciplina y tenacidad y me mantuvieron alejado de influencias negativas mientras yo me construía a mi mismo.

A todos, gracias.

Contents

List of Figures

Abbreviations

dof	**d**egrees **o**f **f**reedom
NNM	**N**onlinear **N**ormal **M**ode
LNM	**L**inear **N**ormal **M**ode
ODE	**O**rdinary **D**iferential **E**quation
SODE	**S**ystem of **O**rdinary **D**iferential **E**quations
SDOF	**S**ingle **D**egree **O**f **F**reedom
MDOF	**M**ultiple **D**egree **O**f **F**reedom
FRD	**F**requency **R**esponse **D**iagram
FEP	**F**requency **E**nergy **P**lot
WEMS	**W**heel **E**lastomer **M**ounting **S**ystem
NES	**N**onlinear **E**nergy **S**ink
LO	**L**inear **O**scillator

Symbols

t time, s

$[\mathbf{M}]$ mass matrix, kg

$[\mathbf{C}]$ damping matrix, N s m^{-1}

$[\mathbf{K}]$ linear stiffness displacement, N m^{-1}

$\mathbf{x} \equiv \mathbf{x}(t)$ vector function of nodes/dofs displacements in the domain of time, m

$\mathbf{f_{nl}}(\mathbf{x})$ nonlinear stiffness vector function, N

$\mathbf{f}(t)$ forced excitation vector function in the time domain, N

M SDOF oscillator mass, kg

C SDOF oscillator damping coefficient, N s m^{-1}

K SDOF oscillator linear stiffeness, N m^{-1}

F SDOF oscillator forced excitation coefficient, F

ω angular frequency, $rads^{-1}$

β Duffing stiffness nonlinearity coefficient. N m^{-3}

M_i mass component of the i-th node or dof, kg

C_{ij} i-th row, j-th column damping matrix component, N s m^{-1}

K_{ij} i-th row, j-th column linear stiffness matrix component, N m^{-1}

F_{ij} i-th node/dof, j-th forced excitation component, N

$\mathbf{x_{N},}_j$ mode shape of the j-th natural –linear– mode, m

$\omega_{N,j}$ natural frequency of the j-th natural –linear– mode, $rads^{-1}$

C_j, coefficient of the j-th component of the free undamped linear response, N s m^{-1}

$\mathbf{a}_m$, $\mathbf{b}_m$ coefficient vectors of the m-th harmonic of a Fourier vector series, m

$\mathbf{a}$, $\mathbf{b}$ coefficient vectors of the fundamental harmonic of a Fourier vector series, m

$A_{i,m}$, $B_{i,m}$ coefficient of the m-th harmonic of the i-th component of a Fourier vector series, m

A_i, B_i coefficient of the fundamental harmonic of the i-th component of a Fourier vector series, m

$\mathbf{R}(t)$ residual vector function, N

$\mathbf{G}(\mathbf{a}, \mathbf{b}, \omega)$ linear system for the approximate forced response, N

$\mathbf{G}_{\mathbf{a},\mathbf{b}}(\mathbf{a}, \mathbf{b}, \omega)$ jacobian matrix of the linear system for the approximate forced response, N m^{-1}

$\mathbf{G}^{-1}_{\mathbf{a},\mathbf{b}}(\mathbf{a}, \mathbf{b}, \omega)$ inverse of the jacobian matrix of the linear system for the approximate forced response, m N^{-1}

$\mathbf{G}(A, B, \omega)$ linear system for the approximate forced response of a SDOF oscillator, N m^{-1}

$\mathbf{G}_{A,B}(A, B, \omega)$ jacobian of the linear system for the approximate forced response of a SDOF oscillator, N m^{-1}

$\mathbf{G}^{-1}_{A,B}(A, B, \omega)$ inverse of the jacobian of the linear system for the approximate forced response of a SDOF oscillator, N m^{-1}

$\mathbf{G}(\mathbf{a}, \omega)$, system of algebraic equations for the approximate modal curves of a system, N

$\mathbf{G}_{\mathbf{a}}(\mathbf{a}, \omega)$, jacobian of the linear system for the approximate modal curves, N m^{-1}

$\mathbf{G}^{-1}_{\mathbf{a}}(\mathbf{a}, \omega)$ inverse of the jacobian of the linear system for the approximate modal curves, m N^{-1}

λ homotopy parameter

λ_0 initial value of the homotopy parameter

$\mathbf{z} \equiv (\mathbf{x}, \mathbf{x}')^T$ state vector of a mechanical system, (m, m s^{-1})

$\mathbf{z}_0$ initial state vector of a mechanical system, (m, m s^{-1})

T period, s

T_0 initial period, s

$\mathbf{z_N}$ natural mode as a state vector, (m, m s^{-1})

ν frequency, Hz

ν_N natural frequency, Hz

$T_N = \frac{1}{\nu_N}$ natural period, s

$\mathbf{g}(\mathbf{z})$ equivalent first-order SODE for mechanical systems, (m s^{-1}, m s^{-2})

$\mathbf{z_p}$ initial state vector of a trajectory in the phase space, (m, m s^{-1})

$\mathbf{z}(\mathbf{z_p}, T)$ continuation duple initial state vector plus period as homotopy parameter, (m, m s^{-1})

$\mathbf{S}(\mathbf{z_p}, T)$ shooting function to measure degree of periodicity of system state vector, m

$h(\mathbf{z_p})$ phase condition of a curve of NNMs points in the invariant manifold of NNMs trajectories in the phase space, (m, m s^{-1})

$\mathbf{F}(\mathbf{z_p}, T)$ augmented shooting function with the phase condition to the points of the phase space

$(\mathbf{z}_i, T_i)$ continuation duple of the i-th NNM on the curve, (m, m s^{-1}, s)

$(\bar{\mathbf{z}}_{i+1}, \bar{T}_{i+1})$ output of the predictor phase in the i-th continuation step, (m, m s^{-1}, s)

$\mathbf{F}_{\mathbf{z},T}(\mathbf{z}_i, T_i)$ jacobian of the augmented shooting function

$\mathbf{v}_i$ tangent vector on the i-th curve point, (m, m s^{-1}, s)

$\mathbf{w} \equiv (\mathbf{z}, T)$ initial state vector-period, (m, m s^{-1}, s)

h_i predictor stepsize value at the i-th step of the continuation algorithm

$(\mathbf{z}_{i+1}, T_{i+1})$ output of the corrector phase in the i-th continuation step, (m, m s^{-1}, s)

$(\mathbf{z}^k_{i+1}, T^k_{i+1})$ k-th iteration of the corrector phase in the i-th continuation step, (m, m s^{-1}, s)

$(\mathbf{z}^{k+1}_{i+1}, T^{k+1}_{i+1})$ output of the k-th iteration of the corrector phase in the i-th continuation step, (m, m s^{-1}, s)

$o(\mathbf{w}_{i+1}^k)$ convergence direction constraint for the corrector phase, (m, m s^{-1}, s)

$\mathbf{H}(\mathbf{w}_{i+1}^k)$ augmented shooting function with the shooting condition plus the corrector direction constraint

$\mathbf{H}_{\mathbf{w}}(\mathbf{w}_{i+1}^k)$ jacobian of the augmented shooting function with the shooting condition plus the corrector direction constraint

$\mathbf{H}_{\mathbf{w}}^{-1}(\mathbf{w}_{i+1}^k)$ inverse of the jacobian of the augmented shooting function with the shooting condition plus the corrector direction constraint

N_{opt} optimum number of corrector iterations

N_i number of corrector iterations of the i-th step of the continuation algorithm

h_{max} maximum stepsize value

h_{min} minimum stepsize value

N_{max} maximum number of corrector iterations

$h_{initial}$ initial stepsize value

ϵ convergence factor of the corrector phase

$\Delta\mathbf{z}(\mathbf{z}_i, \Delta\mathbf{z}_i, T_i)$ perturbation vector of a trajectory in the phase space after a perturbation of the initial state vector, m

$\Phi_T(\mathbf{z}_i, T_i)$ monodromy matrix of the i-th NNM of a curve in the phase space, m m^{-1}

E total mechanical energy, J

T kinetical energy, J

L_L linear stiffness energy, J

L_{NL} nonlinear stiffness energy, J

A todos aquellos que, pese a todo, siempre creyeron en mí. No hace falta decir sus nombres. Ellos se verán reflejados en estas palabras.

Chapter 1

INTRODUCTION

1.1 Background and motivation

Vibration Engineering is a wide brach of Mechanics with a key role in the design and study of any structural or mechanical system. It deals with the analysis of the dynamical effects and variable oscillating motions caused by external forces of disturbances on those systems. Those oscillations are called 'vibrations'. An uncontrolled state of vibration on a structure or component could lead to disastrous consequences to its integrity, examples of this can be found in the oscillation of unbalaned rotors and shafts, in natural phenomena such as earthquakes or wind gusts or in machine tool chatter –an example of self-induced vibration caused by friction–. However, vibrations can also be used by the designer and operator with positive effects: one of those effects is isolation: it should be convenient to isolate the system from any possible disturbance from the outside. Examples of this are found in car suspesions and machine vibration isolators to protect either the machine or the structure of attachment, as in passive mass dampers that benefit from energy transfer in order to control and threshold the motion of the mechanism to protect. Another beneficial exploitation of vibration can take place by means of a 'energy harvesting system', a mechanism that takes residual or external energy from a source and turns it into a useful energy, generally electric power.

The analysis of the dynamics of any system requiers the build-up of a physical and mathematical model, which delivers a differential equation or system of equations in the domains of time and space. Traditionally engineers have always taken a linear approach to it, considering the linear hypothesis in their models and dismissing any nonlinear effect that could arise, even linearizing when neccesary around the equilibrium points. In certain systems this could mean the loss of important information about the system behavior, resulting in an unexpected behavior. In other circumstances, the designer could introduce nonlinearities into a system intentionally in order to alter its dynamical behavior with beneficial purposes, such as energy harvesting and energy transfer by localization. During the last decades there has been a huge amount of research on useful applications of nonlinearity in Engineering design and exploitation –see [1], [2], [3], [4], [5], [6] and [7]–, and in consequence, a turnaround in the design paradigms could be expected.

The theory of linear dynamical systems has been widely developed since the XIX century and is very well known by practising engineers, who use it today in the analysis of the dynamics of mechanical systems and the determination of the model parameters. When the anaysis is conducted with computational techniques such as finite element analysys –FEM– it conforms the subject of 'computational modal analysis' –[8]–. All this data obtained computationally could –and should– be contrasted with experimental results obtained with methods of 'experimental modal analysis'. But the computational tools and algorithms used for linear study are useless when it comes to models containing any nonlinear component. Nonlinear dynamical systems have a richness and variety of behaviors and phenomena that does not exist in linear systems and that I describe in the context of nonlinear vibrations. Moreover, the motion of nonlinear systems do not comply with the principle of linear superposition. This makes their analysis much more complex because, in the general case, a nonlinear motion cannot be split into a set of simple synchronous motions –the so-called 'normal modes' which are the cornerstone of modal analysis– unlike the linear models. Due to all those reasons, nonlinear physics researchers had to develop unconventional techniques and methods for the analysis of nonlinear dynamical systems and, namely, of nonlinear oscillations –see [9], [10] and [11]–. In this context is where 'nonlinear normal modes' –NNMs– have come into the picture. They can be deemed as an extension of the normal modes of linear systems for systems

with nonlinearities, inheriting some of the properties of the linear modes that make them useful to have an overall view of the dynamical motion and behavior of the system, even though some desirable properties such as mode superposition, as it was mentioned before, do not comply in nonlinear systems –see [12], [13], [14], [15], [16], [17] and [18]–.

Linear modal analysis, with the computation of linear normal modes, takes advantage of the theory of eigenvalues and eigenvectors for their calculation. However, the computation on nonlinear normal modes is far less trivial. Analytic methods such as the invariant manifold approach or the harmonic balance require to approximate the real solution by a power series. Meanwhile, numerical methods, which have not been exploited for this purpose until recent years, are limited by numerical accuracy, computational power and convergence rates. In this text I focus on these two techniques for the computation of NNMs: the harmonic balance method, with the purpose of introducing some basic themes on nonlinear vibration and shedding light on the physical and mathematical meaning of NNMs; and a numerical algorithm based on arclength continuation methods and the numerical computation of trajectories. This algorithm can be found thoroughly described in this text, as well as in [19], [20] and [21]. Numerical techniques such as this have many advantages over the analytical methods: they are suitable for systems with strong nonlinearities and a large number of degrees of freedom, and of course, a computer can be programmed with those algorithms taking the model of the system to study as an input of the computation. This method turns out to be extraordinarily efficient with complex systems on which analytical methods could never be applied feasibly –see [22], [23], [24] and [25]–.

I have made an extensive use of numerical computing tools as a part of this text, namely, Matlab and Maple. For the preliminaries I have written my own codes for computational assistance on the application of the harmonic balance method and the output of numerical results and figures. On the other hand, I have made use of a Matlab package developed by the Structural Dynamics Research Group of the University of Liège called 'NNMcont' that can compute the nonlinear normal modes of any conservative nonlinear 2nd-order SODE, given the model definition, with the continuation algorithm I introduce in this text –see [26]–. The use, installation and execution of that program are described in depth to fill the gap left by the lack of thorough documentation about 'NNMcont'. Those two are

the main original contributions made in this Project. All my original source codes used are found in appendices A and B.

The uppermost application of the theory of nonlinear normal modes found in the literature is in mechanical analysis and design, and that is the focus of this text. Nevertheless, the same concepts that I present can be applied on any analogous dynamical system such as electrical systems, and that application could also be a topic of Engineering interest.

1.2 Aims of the Project

My aims for this Project can be enumerated as follows:

1. To bring the subjects of nonlinear vibrations and NNMs and the most important academic work on it to light to more researchers and engineers, gathering some comprehensive literature on the topic and enphasizing the practical Engineering aplications of those concepts.

2. To present methods of calculation and computation of the dynamical and oscillatory behavior of mechanical systems, including the computation of NNMs.

3. To build models of simple mechanical system that can serve as a practical example where the concepts treated in this text could be applied for mechanical design or dynamical analysis purposes.

4. To apply one of the analytical methods presented –the harmonic balance method– on them with computational assistance, contrasting the results with the classical Linear Vibration Theory.

5. To shift my focus to numerical continuation methods for the computation of NNMs, describing the algorithm as it is described in the literature and providing reference to a further insight in the topic.

6. To present and use a Matlab Package called 'MMNcont', developed by researchers of the University of Liège for the computation and representation of NNMs of conservative systems by means of a continuation algorithms, and to apply it on a exemplary two degrees of freedom model, describing its usage and execution.

7. To perform a critical analysis of the results from the simulations and analysis, with the focus on their degree of accuracy and reliability and on determining how the dynamical behavour of the system analized can be infered from those results.

8. To prove my skills and competences as a future Engineer. A list of the skills that I pretend to show with this text can be found in chapter 6.

1.3 Methodology

I have followed the following steps during the realization of this Project:

1. First of all, I did a deep search of bibliography on the topic of Vibration Theory, nonlinear vibration and NNMs, which included books, articles, conference proceedings, lectures and slideshows among other documents. Some of this references put the focus on the physical and mathematical theory, others in analytical and numerical computation and calculation techniques. Another proportion of the bibliography was dedicated to a range of practical applications of vibrations and nonlinearity –control, localization, energy harvesting, etc– and a few references covered several of those approaches.

2. The next step was to make a selection of the range of topics and methods to cover in this text and the depth of treatment of the topics presented. This included to make a search of systems and models that could be studied under these new theory and methods. All this work aimed, in short, to define the scope of the Project, already described in the previous section 1.2.

3. Then, I got hold of the computational tools I would require to comply with the aims of this Project: Matlab and Maple. Moreover, I required the Matlab package 'NNMcont', freely available in this website: `www.ltas-vis.ulg-ac.be/cmsms/uploads/File/NNMcont.zip`. I give a further insight in the software I used in the corresponding chapters.

4. Some of the goals of this Project required building up small Matlab executable files to carry out the computation of the solution of certain equations. Those equations are derived from different mechanical models and their equations of motion. These codes allow to compute some figures and plots that are included in this text. Hence a considerable amount of time has been invested in coding and in collecting the graph outputs of the computation in a comprehensive format.

   The computational work can be split in two parts: The first one has required the application of the harmonic balance method on the one and two degrees of freedom mechanical models shown in this text in order to obtain the approximate free and forced responses of these systems –see chapter 2–. Meanwhile, the second one has taken advantage of the numerical continuation techniques described on the chapter 3 and has been made with the Matlab package 'NNMcont' introduced in chapter 4, with its results shown in chapter 5.

5. Before beginning to write this text I considered it was very convenient to devise its structure and organization, bearing in mind that the LaTeX format would be the most appropiate for it. This text has been structured in this manner:

   (a) In the first main section –chapter 2– I introduce the basics of Nonlinear Vibration Theory, contrasting it with the well-known Linear Theory. Then, nonlinear normal modes –NNMs–, the most important tool of nonlinear modal analysis, are defined and compared with their linear counterpart: linear modes of vibration. The nonlinear phenomena that NNMs can display in nonlinear systems is described, as well as a short discussion on the practicality of their Engineering analysis and consideration. In the same chapter, the most important analytic methods developed for the heuristic calculation of NNMs are shown. One of them, the harmonic balance method, is applied on one degree of freedom –the

Duffing oscillator– and two degrees of freedom models to obtain their frequency response diagrams, along with the NNMs branches or backbones caracterizing the systems. A very clear relationship between the NNMs branches and the resonance peaks of the frequency response curves is graphically established and visible from these simple exemplary systems. Finally, I switch to numerical techniques, outlining their advantages over the analytical methods presented before and explaining which outputs are expected when the NNMs numerical computation is carried out in Matlab and the physical phenomena they quantify.

(b) Chapter 3 describes the numerical method that makes another of the cornerstones of this text: a method based on a continuation algorithm on a shooting function quantifying periodicity, requiring the numerical computation of the system orbits with Runge-Kutta or Newmark techniques. Along with this method the control parameters of the algorithm are mentioned, as well as the numerical outputs obtained.

(c) The group of researchers who presented the numerical algorithm previously described in proceedings and publications –see bibliography– has made available online a Matlab package called 'NNMcont'. This tool computes and displays graphically the NNMs branches of any nonlinear mechanical model defined by the user in a matlab file. In chapter 4 I describe the usage of this package accounting its possibilities and its limitations.

(d) Finally the two degrees of freedom model that was studied in previous section with harmonic balance is studied in the 'NNMcont' software. In chapter 5 NNMs branches of this system are computed, comparing the results obtained with those from the harmonic balance method and doing a critical analysis of those, figuring out information about the dynamical response of the system from this analysis..

6. The most important stage of this Project, and the longest, has been the writing of this text. It involved the composition of all the graphs, figures and images in it. I opted for very simple software, such as Paint or Microsoft Power Point to edit all the figures included, but the results are fairly good and satisfactory.

7. As an Engineering Project, it was compulsory to elaborate a Project schedule and planing of all its stages, taking account of the time invested in its completition and putting it into a Gantt chart. A Project budjet was also included in this text.

8. Another sections included in this text are abstract, introduction, conclusions, bibliography and apendices, among others.

As a final remark, any original contribution present along this text is remarked to distinguish it from what is taken from external references, knowledge and research.

Chapter 2

PRELIMINARIES

In this chapter the dynamics in forced and free nonlinear mechanical systems will be introduced, taking the harmonic balance method as a technique to approximate the solutions of the equations of motion, then describing the graphic results obtained in Matlab for the amplitude and phase of the steady-state vibration. Later on, nonlinear modal analysis will be contrasted with its linear counterpart, posing a definition to the concept of 'nonlinear normal mode' –NNMs–, the cornerstone of this text. The rich nonlinear behavior of NNMs, unprecedented in linear normal modes, will be described, along with a short discussion on the practicality of their Engineering analysis and consideration.

The reader is presumed to have a basic knowledge of linear free and forced vibrations in single and multiple degree of freedom systems, and desirably, of nonlinear vibration, as well as in the most elementary Nonlinear Dynamics. A very good reference on vibration can be found in [27]. Another interesting monograph about modal analysis, both experimental and computational or analytical, is [8]. Some classical references on nonlinear dynamics and vibrations that cover most sorts of systems that can be found in mechanical modeling, and of course, in this text, are [9] and [11].

2.1 Modelization of mechanical systems: the fundamental equation

In the most general form, a mechanical system with N degrees of freedom –dofs– with nonlinear stiffness can be modeled by the following second-order SODE:

$$[\mathbf{M}]\mathbf{x}'' + [\mathbf{C}]\mathbf{x}' + [\mathbf{K}]\mathbf{x} = \mathbf{f}(\mathbf{x}, t) \tag{2.1}$$

where $[\mathbf{M}]$ is the mass matrix, $[\mathbf{C}]$ is the damping matrix and $[\mathbf{K}]$ is the matrix of linear stiffness, all in $\mathbb{R}^{NxN}$; $\mathbf{x}$ represents a function in the domain of time, $\mathbf{x}(t)$, that quantifies the evolution of the displacement of each node with respect to their equilibrium position; and $\mathbf{f}(\mathbf{x}, t)$ is a component that depends on the displacements of the nodes and on time, comprising all the external forces on the nodes and nonlinear elastic forces in the model.

Remember that if we take a continuous model to define a mechanical system –such as the Navier-Stokes equation– it can be discretizised by means of approximation methods such as Galerkin –see [28]–.

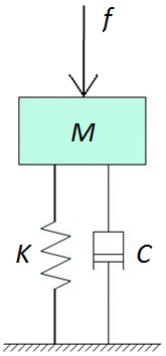


FIGURE 2.1: This figure represents an elementary single degree of freedom –SDOF– mechanical oscillator. It comprises a mass M –whose vertical displacement from equilibrium is quantified by the function **x**–, a linear spring of stiffness K, an absorber with a damping coeficient C and a external vertical force on the mass, **f**, that could be constant or variable in time.

However, in this Project I will only study models of one and two degrees of freedom because they are sufficient to understand the utility and applications of nonlinear vibration analysis to be disclose in this text as stated in section 1.2.

2.1.1 Model of a single degree of freedom oscillator

The single equation –ODE– determining the motion of a SDOF oscillator with nonlinear stiffness –figure 2.2– and a sinusoidal excitation can be written as:

$$Mx'' + Cx' + Kx + f_{nl}(x) - F\sin\omega t = 0 \tag{2.2}$$

where M is the mass, C is the viscous damping coefficient, K is the linear stiffness term, $f_{nl}(x)$ is a nonlinear stiffness function, F is the forcing amplitude and ω the forcing pulsation.

If the nonlinear stiffness is a cubic power of the displacement, then $f_{nl}(x) = \beta x^3$, where β is a nonlinear stiffness coefficient.

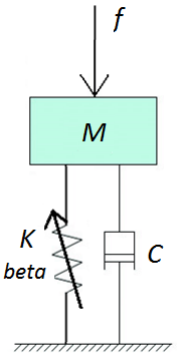


FIGURE 2.2: Forced SDOF nonlinear damped oscilator.

The equation of the undamped oscillator in free –unforced– motion can be derived from 2.2:

$$Mx'' + Kx + \beta x^3 = 0 \tag{2.3}$$

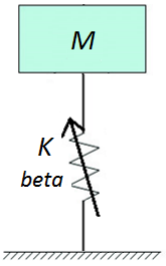


FIGURE 2.3: Free SDOF nonlinear undamped oscilator.

2.1.2 Model of a two degrees of freedom oscillator

A two degrees of freedom oscillator will be modeled according to figure 2.4. The SODE that defines its motion is:

$$\begin{aligned} M_1 x_1'' + C_1 x_1' - C_2(x_2' - x_1') + K_1 x_1 - K_2(x_2 - x_1) - \beta(x_2 - x_1)^3 \\ -F_{11}\sin\omega t - F_{12}\cos\omega t = 0 \\ M_2 x_2'' + C_2(x_2' - x_1') + K_2(x_2 - x_1) + \beta(x_2 - x_1)^3 \\ -F_{21}\sin\omega t - F_{22}\cos\omega t = 0 \end{aligned} \tag{2.4}$$

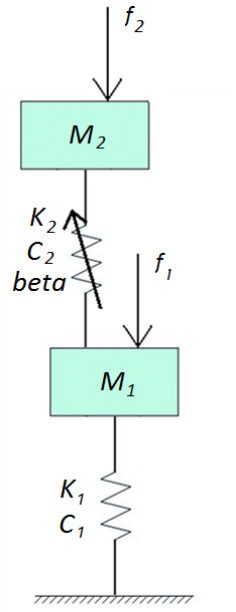


FIGURE 2.4: Forced 2dofs nonlinear damped oscillator. It comprises two masses with vertical displacement. The first mass is attached through a damped elastic suspenson to a fixed foundation, while the second is attached to the first mass by means of the same viscoelastic suspension but holding a stiffness nonlinearity with coefficient β. The two excitations on the dofs f_1 and f_2 are sinusoidal of pulsation ω and are allowed to have different phases.

with masses M_1 and M_2, viscous damping coefficients C_1, C_2, linear stiffnesses K_1 and K_2, nonlinear stiffness coefficient β and forcing coefficients F_{11}, F_{12}, F_{21} and F_{22}.

The equations of the free undamped motion of the 2dofs oscillator 2.4 are given by:

$$\begin{aligned} M_1 x_1'' + K_1 x_1 - K_2(x_2 - x_1) - \beta(x_2 - x_1)^3 = 0 \\ M_2 x_2'' + K_2(x_2 - x_1) + \beta(x_2 - x_1)^3 = 0 \end{aligned} \tag{2.5}$$

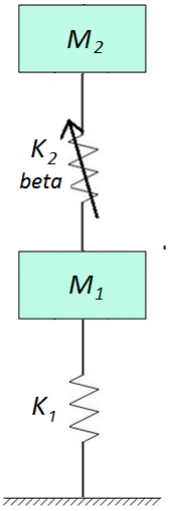


FIGURE 2.5: Free 2dofs nonlinear undamped oscillator.

2.2 The linear oscillators

2.2.1 A short digest on linear vibration

Linear Theory of Vibrations deals with the dynamics of mechanical systems whose equations of motion have the general form:

$$[\mathbf{M}]\mathbf{x}'' + [\mathbf{C}]\mathbf{x}' + [\mathbf{K}]\mathbf{x} = \mathbf{f}(t) \tag{2.6}$$

This second-order SODE is linear and, if the forcing on each dof is a sine wave: $f_i(t) = F_{i1}\sin\omega t + F_{i2}\cos\omega t$, its steady-state solutions for every dof will be given by:

$$x_i(t) = A_i\sin\omega t + B_i\cos\omega t \tag{2.7}$$

These solutions are pure sinusoidal waves of frequency ω. The oscillations of the dofs are not necessarily synchronous to each other, unless there is no damping –$[\mathbf{C}] = 0$– and all the forcings on the dofs are synchonous, that is, $f_i = F_i\sin\omega t$. Remember that the notion of synchronicity assumes that the motion of all the dofs reach their extremes and zero amplitude values at the same time.

In the latter case, the resulting equation:

$$[\mathbf{M}]\mathbf{x}'' + [\mathbf{K}]\mathbf{x} = \mathbf{F}\sin\omega t \tag{2.8}$$

has this as the steady-state solution:

$$x_i(t) = A_i \sin \omega t \tag{2.9}$$

which proves the synchronicity of all the dofs of the model.

The linear free undamped –conservative– equation has the form:

$$[\mathbf{M}]\mathbf{x}'' + [\mathbf{K}]\mathbf{x} = \mathbf{0} \tag{2.10}$$

Any solution to the last equation can be written as a linear combination of a set of basis functions called 'normal modes'. Each of those normal modes characterized each by a mode shape vector $\mathbf{x_N}_{,j}$ and a 'natural pulsation' $\omega_{N,j}$.

Each j-th normal mode is a solution of 2.10 by itself and has this basic form:

$$\mathbf{x}(t) = C_j \mathbf{x_N}_{,j} e^{i\omega_{N,j}t} = C_j \mathbf{x_N}_{,j} \sin \omega_{N,j} t \tag{2.11}$$

The set of normal modes of the system 2.10 can be calculated from the eigenvalues and eigenvectors given by the following generalized eigenproblem:

$$[\mathbf{K}]\mathbf{x_N} = \omega_N^2 [\mathbf{M}]\mathbf{x_N} \tag{2.12}$$

Normal modes in linear systems are sine waves whose pulsation is one of the natural pulsation and whose shape is the mode shape associated, proportional to a real or complex

coefficient C_j that depends on the amount of energy carried by that mode. If we could displace all the dofs of a system to the shape of a mode $(\mathbf{x_N}_{,j}, \omega_{N,j})$, and then release the system, it would oscillate with the motion given by 2.11. It is also noticable that when the system oscillates on a normal mode the motion of its nodes in synchronous –they vibrate 'in unison'–. Obtaining the normal modes of a mechanical system is one of the aims of modal analysis, both experimentally or computationally. The resonance peaks of any forced mechanical system will always arise in the vicinities of its natural frequencies, while its shape of oscillation will resemble the associated mode shape.

2.2.2 Steady-state solutions of the linear oscillators

It was seen in 2.2.1 that if the mechanical model is linear and the forcing is sinusoidal, then the steady-state response –the displacements of the degrees of freedom– is harmonic with same frequency than the excitation, therefore, the following solution:

$$x_i(t) = A_i \sin \omega t + B_i \cos \omega t \tag{2.13}$$

is exact for 2.2 and 2.4, with β set to zero.

In the case of a linear free undamped oscillator –such as those modeled by equations 2.3 and 2.5–, the solution of each degree of freedom for each mode of oscillation has the general form:

$$x_i(t) = A_i \sin \omega t \tag{2.14}$$

resulting is a synchronous, periodic harmonic oscillation of all the dofs.

Setting β to zero in the SDOF forced oscillator equation we have:

$$Mx'' + Cx' + Kx - F\sin\omega t = 0 \tag{2.15}$$

Substituting 2.13 into 2.15 for the single degree of freedom and balancing coefficients around $\sin\omega t$ and $\cos\omega t$ results in the following linear system of algebraic equations with dependent variables A and B, and independent parameter ω, the forcing pulsation:

$$\begin{aligned} -4F - 4CB\omega + 4KA - 4MA\omega^2 = 0 \\ -4MB\omega^2 + 4CA\omega + 4KB = 0 \end{aligned} \tag{2.16}$$

Solutions A and B could be analytically cleared from the equations above as functions of ω, however, in this Project I have opted for the Newton-Raphson numerical algorithm for any equation solving, linear or nonlinear, as I will describe in further sections.

The amplitude of the harmonic response is given by the values of its two coefficients as $\sqrt{A^2 + B^2}$, while the phase angle between response and excitation is a function of $\arctan B/A$ with a correction to make it lay in the interval $[-\pi, 0]$:

$$\begin{aligned} A > 0 \Rightarrow \phi = -\arctan\frac{B}{A} \\ A < 0 \Rightarrow \phi = \arctan\frac{B}{A} - \pi \end{aligned} \tag{2.17}$$

The minus sign of the phase angle in the whole interval would indicate a delay of the response with respect to the excitation.

Now, we look for synchronous, periodic harmonic solutions for the steady-state free undamped oscillations of the linear SDOF oscillator:

$$Mx'' + Kx = 0 \tag{2.18}$$

Substituting 2.14 into equation 2.18, and balancing terms around $\sin \omega t$ setting them to zero, results in:

$$4KA - 4MA\omega^2 = 0 \tag{2.19}$$

The last equation has no solution unless $\omega = \sqrt{\frac{K}{M}}$. This value of the pulsation is called 'natural pulsation' and is characteristic of the mechanical system for certain constraints M and K. It determines the 'natural frequency' of the system by the expression $\nu = \frac{\omega}{2\pi}$. The physical meaning of this equation is that for any amplitude of free oscillation of 2.3 its frequency remains unchanged and is equal to its natural frequency.

Notice that equation 2.19 can be derived from the first equation of 2.16 by setting F, B and C to zero.

Following the same procedure, we can derive the linear algebraic equations that determine the dependence between the coefficients of the harmonic solutions and the forcing pulsation of the linear two degrees of freedom oscilator in 2.4. It suffices to set β to zero in SODE 2.4 and the harmonic solutions 2.13 for the displacement of each i-th dof are exact:

$$\begin{aligned}
-4F_{11} - 4C_1B_1\omega - 4C_2B_1\omega - 4M_1A_1\omega^2 + 4C_2B_2\omega - 4K_2A_2 \\
+4K_2A_1 + 4K_1A_1 = 0 \\
4C_1A_1\omega + 4C_2A_1\omega - 4M_1B_1\omega^2 - 4C_2A_2\omega + 4K_2B_1 + 4K_1B_1 \\
-4K_2B_2 - 4F_{12} = 0 \\
-4C_2B_1\omega + 4M_2A_2\omega^2 + 4C_2B_2\omega - 4K_2A_2 + 4F_{21} + 4K_2A_1 = 0 \\
4F_{22} - 4K_2B_2 + 4K_2B_1 - 4C_2A_2\omega + 4M_2B_2\omega^2 + 4C_2A_1\omega = 0
\end{aligned} \tag{2.20}$$

The amplitudes of the harmonic responses of degrees of freedom one and two are derived from the values of the coefficients of $\sin\omega t$ and $\cos\omega t$:

$$\begin{aligned}
Amplitude\ dof\ 1 = \sqrt{A_1^2 + B_1^2} \\
Amplitude\ dof\ 2 = \sqrt{A_2^2 + B_2^2}
\end{aligned} \tag{2.21}$$

Meanwhile, the phase angle of the response of each i-th dof depends on the values and signs of the coefficients A_i and B_i.

Equally, taking equations one and three from 2.20 and setting C_1, C_2, B_1, B_2 and all the F_{ij} coefficients to zero will result in the equations determining the free linear undamped oscillation of 2.5:

$$\begin{aligned}
-4M_1A_1\omega^2 - 4K_2A_2 + 4K_2A_1 + 4K_1A_1 = 0 \\
4M_2A_2\omega^2 - 4K_2A_2 + 4K_2A_1 = 0
\end{aligned} \tag{2.22}$$

The algebraic system above is analogous to the eigenproblem of the 2dofs undamped mechanical system described by 2.12, which has the following matricial expression:

$$\omega^2[\mathbf{M}](A_1, A_2)^T = [\mathbf{K}](A_1, A_2)^T \tag{2.23}$$

2.3 The effects of nonlinearity

In this section I will proceed to describe the most elementary dynamics of nonlinear mechanical systems whose behavior is given by the SODE:

$$[\mathbf{M}]\mathbf{x}'' + [\mathbf{C}]\mathbf{x}' + [\mathbf{K}]\mathbf{x} + \mathbf{f_{nl}}(\mathbf{x}) = \mathbf{f}(t) \tag{2.24}$$

where $\mathbf{f_{nl}}(\mathbf{x})$ is the function representing the stiffness nonlinearities given by the system.

Harmonic balance has been the analytical method taken in this text to study the steady-state dynamics of these systems, and will be described in the next section.

2.3.1 Steady-state solutions with the harmonic balance method

This is a very useful method when the interest is in the periodic solutions of a dynamical system. It supposes the solution as a Fourier series with the form:

$$\mathbf{x}(t) = \sum_{m=1}^{\infty} (\mathbf{a}_m \sin m\omega t + \mathbf{b}_m \cos m\omega t) \tag{2.25}$$

where the first term of the harmonic sum –$m = 1$– is the 'fundamental harmonic', whose period, $\frac{2\pi}{\omega}$, is the minimal period of the solution in the phase space of the system.

The systems treated in this text have the following fundamental form, and we will constrain to it in what concerns to this method: a discrete mechanical system with a nonlinear stiffness term and an harmonic excitation of pulsation ω.

$$[\mathbf{M}]\mathbf{x}'' + [\mathbf{C}]\mathbf{x}' + [\mathbf{K}]\mathbf{x} + \mathbf{f_{nl}}(\mathbf{x}) - \mathbf{F_1}\sin\omega t - \mathbf{F_2}\cos\omega t = \mathbf{0} \tag{2.26}$$

where $\mathbf{x} = (x_1, x_2, ..., x_N)^T$ is any function of the dofs displacements in the domain of time that satisfies the SODE shown above in the whole configuration space.

For each i-th degree of freedom we will suppose the solution for the displacement as an harmonic series as follows:

$$x_i(t) = \sum_{m=1}^{\infty}(A_{i,m}\sin m\omega t + B_{i,m}\cos m\omega t) \tag{2.27}$$

where the fundamental frequency matches the frequency of the forcing term of the system, and $A_{i,m}$ and $B_{i,m}$ are coefficients of the series.

For simplicity, we will truncate the harmonic series to the fundamental harmonic, leading to a solution in the form of a pure sine wave that approximates the real periodic solutions of 2.26:

$$x_i(t) \approx A_i\sin\omega t + B_i\cos\omega t \tag{2.28}$$

with undetermined coefficients A_i and B_i for each i-th degree of freedom.

2.28 is only an approximation, therefore, when introduced into 2.26 a residual function $\mathbf{R}(t)$ can be defined. It quantifies how close 2.28 is to the real solutions of 2.26:

$$\mathbf{R}(t) = [\mathbf{M}]\mathbf{x}(\mathbf{t})'' + [\mathbf{C}]\mathbf{x}(\mathbf{t})' + [\mathbf{K}]\mathbf{x}(\mathbf{t}) + \mathbf{f_{nl}}(\mathbf{x}(\mathbf{t})) - \mathbf{F_1}\sin\omega t - \mathbf{F_2}\cos\omega t \tag{2.29}$$

Another way to see the series 2.27 is as a linear combination of a basis harmonic continuous test functions $\sin\omega t, \cos\omega t, \sin 2\omega t, \cos 2\omega t, ...$ with coefficients $A_1, B_1, A_2, B_2, ...$ These test functions are all pairwise orthogonal for the inner product, therefore, any continuous periodic function in the domain $t \in [0, \frac{2\pi}{\omega}]$ can be univocably expressed as the infinite series 2.27.

That infinite basis in unfeasible to work with and one must ask if any of those functions $\mathbf{x}(t)$ could be approximated by a finite sum of terms from the harmonic basis such as 2.28. The answer is yes, and now a method must be taking into account to yield the coefficients of the linear combinations leading to the best, or most acceptable, approximation. We will resort to the Galerkin method of weighted residuals: The coefficients will be derived from the orthogonalization of the residuals with respect to a set of 'weight functions'. Those weight functions are the same harmonic test functions that form the basis of the function space of all periodic $\mathbf{x}(t)$.

If we choose the truncated series 2.28 as an approximate solution of the SODE 2.26 for each i-th dof, the orthogonalization of the residuals in the interval $[0, \frac{2\pi}{\omega}]$ will lead to a set of $2N$ algebraic equations –for the $2N$ dofs– that determine the values of all the coefficients A_i and B_i of the approximate solution 2.28 as a function of the parameter ω:

$$\begin{aligned} \int_0^{\frac{2\pi}{\omega}} \mathbf{R}(t)\sin\omega t \; dt = 0 \\ \int_0^{\frac{2\pi}{\omega}} \mathbf{R}(t)\cos\omega t \; dt = 0 \end{aligned} \tag{2.30}$$

More on Galerkin methods in reference [28].

As said, the result of the integrations is the following algebraic system of $2N$ equations with $2N$ solutions –the coefficients $A_1, A_2, ..., A_N, B_1, B_2, ..., B_N$– and the parameter ω, the forcing pulsation, also the pulsation of the approximate solution:

$$\mathbf{G}(A_1, A_2, ..., A_N, B_1, B_2, ..., B_N, \omega) = \mathbf{G}(\mathbf{a}, \mathbf{b}, \omega) = \mathbf{0} \tag{2.31}$$

where $\mathbf{a} = (A_1, A_2, ..., A_N)^T$ and $\mathbf{b} = (B_1, B_2, ..., B_N)^T$.

The algebraic system 2.31 can be interpreted as an implicit curve in the $\mathbb{R}^{2N+1}$ field with ω as the independent parameter. In the context of mechanical systems modeled by equation 2.26 the resulting curve should be interpreted as the frequency response curve for non-zero excitations, and a set of modal backbones –periodic free undamped motions called 'modes'– for null excitations –$\mathbf{F_1} = \mathbf{0}$ and $\mathbf{F_2} = \mathbf{0}$– and damping coefficients –C_1 and $C_2 = 0$–.

Computationally, the curves can only be obtained as sequence of points for discrete values of ω. The method used here to compute the curves is a Newton-Raphson algorithm for different values of ω: Suppose a j-th value of ω from a list or sequence of values. Sustituting in 2.31 we have a determined system of $2N$ equations and $2N$ solutions. The following Newton-Raphson formula:

$$(\mathbf{a}^{k+1}, \mathbf{b}^{k+1})^T = (\mathbf{a}^k, \mathbf{b}^k)^T - \mathbf{G}^{-1}_{\mathbf{a},\mathbf{b}}(\mathbf{a}^k, \mathbf{b}^k)\mathbf{G}(\mathbf{a}^k, \mathbf{b}^k) \tag{2.32}$$

must be applied iteratively on a initial vector of solutions $(\mathbf{a}^0, \mathbf{b}^0)$ in an attempt to converge to the solution of 2.31 for the j-th value of the parameter ω. Several initial solutions for each ω should be considered because if 2.26 is nonlinear –$\mathbf{f_{nl}}(\mathbf{x}) \neq 0$– for some intervals of ω equations 2.31 have multiple solutions $(\mathbf{a}^0, \mathbf{b}^0)$.

The solutions define an approximate steady-state displacement for each i-th dof in the form of a trigonometric wave with an associated oscillation amplitude:

$$Amplitude\ dof\ i = \sqrt{A_i^2 + B_i^2} \tag{2.33}$$

and the phase angle of the response of each degree of freedom depends on the values and signs of A_i and B_i.

Remember that as seen in section 2.2.2 if 2.26 is linear –$\mathbf{f_{nl}}(\mathbf{x}) = \mathbf{0}$– the fundamental harmonic 2.28 is an exact solution of the displacement of each degree of freedom.

The harmonic balance method is broadly treated in texts on Nonlinear Dynamics and Vibration and in academic papers and literature. Some of the references that mention or described the method are [11], [3], [1] or [5].

2.3.1.1 An example in one degree of freedom: the Duffing oscillator

Remember the ODE for the nonlinear SDOF oscillator represented in figure 2.2:

$$Mx'' + Cx' + Kx + f_{nl}(x) - F\sin\omega t = 0 \tag{2.34}$$

The steay-state solution can be written as a Fourier series:

$$x(t) = \sum_{m=1}^{\infty}(A_m \sin m\omega t + B_m \cos m\omega t) \tag{2.35}$$

For simplicity, the series is truncated to the fundamental harmonic:

$$x(t) \approx A \sin \omega t + B \cos \omega t \tag{2.36}$$

Substituting, a residual can be defined as the degree of accuracy with respect to the real solution:

$$R(t) = Mx''(t) + Cx'(t) + Kx(t) + f_{nl}(x(t)) - F \sin \omega t \tag{2.37}$$

Applying the Galerkin method, the residual is orthogonalized with respect to the trial functions of the harmonic approximation –$\sin \omega t$ and $\cos \omega t$–

$$\begin{aligned} \int_0^{\frac{2\pi}{\omega}} R(t) \sin \omega t dt = 0 \\ \int_0^{\frac{2\pi}{\omega}} R(t) \cos \omega t dt = 0 \end{aligned} \tag{2.38}$$

In the Duffing oscillator, the nonlinear stiffness term is proportional to the cube of the mass displacement –$f_{nl}(x) = \beta x^3$–. Now, the residual 2.37 can be established:

$$\begin{aligned} R(t) = -MA\omega^2 \sin \omega t - MB\omega^2 \cos \omega t + CA\omega \cos \omega t \\ -CB\omega \sin \omega t + KA \sin \omega t + KB \cos \omega t + \beta A^3 \sin \omega t \\ -\beta A^3 \sin \omega t \cos \omega t^2 + 3\beta A^2 B \cos \omega t - 3\beta A^2 B \cos \omega t^3 \\ +3\beta AB^2 \sin \omega t \cos \omega t^2 + \beta B^3 \cos \omega t^3 - F \sin \omega t \end{aligned} \tag{2.39}$$

The orthogonalization of the residual by means of 2.38 will result in the following system of algebraic equations:

$$\begin{aligned} 3\beta A^3 - 4F - 4CB\omega + 4KA - 4MA\omega^2 + 3\beta AB^2 = 0 \\ 3\beta B^3 - 4MB\omega^2 + 4CA\omega + 3\beta A^2 B + 4KB = 0 \end{aligned} \tag{2.40}$$

The resulting system, $\mathbf{G}(A, B, \omega) = \mathbf{0}$, is a set of two equations with two dependent variables –A and B– with an independent variable –ω–. For a given ω we could apply Newton-Raphson on a set of initial values of $(A^0, B^0)^T$ expecting to converge to the steady-state approximate solutions of 2.2:

$$(A^{k+1}, B^{k+1})^T = (A^k, B^k)^T - \mathbf{G}_{A,B}^{-1}(A^k, B^k)\mathbf{G}(A^k, B^k) \tag{2.41}$$

where $\mathbf{G}_{A,B}$ is the jacobian matrix of $G(A, B)$:

$$\mathbf{G}_{A,B}(A, B) = \begin{pmatrix} \frac{\partial \mathbf{G_1}}{\partial A}(A, B) & \frac{\partial \mathbf{G_1}}{\partial B}(A, B) \\ \\ \frac{\partial \mathbf{G_2}}{\partial A}(A, B) & \frac{\partial \mathbf{G_2}}{\partial B}(A, B) \end{pmatrix} \tag{2.42}$$

The amplitude of the harmonic oscillation of the mass is derived from the values of the coefficients of the trigonometrical response 2.36:

$$Amplitude = \sqrt{A^2 + B^2} \tag{2.43}$$

Those same coefficients determine the delay between the response and the excitation through the phase angle of the harmonic response. The phase angle, $\arctan \frac{B}{A}$, has to be corrected according to the sign of A in the same way than in the linear SDOF oscillator, that is, as stated by the expression 2.17.

Nonlinear free undamped oscillators such as the SDOF one portrayed in 2.3 and modeled by the following ODE always have a similar treatment in modal analysis:

$$Mx'' + Kx + f_{nl}(x) = 0 \tag{2.44}$$

The solutions of interest are periodic solutions with minimal period $T = \frac{2\pi}{\omega}$, that is, with fundamental frequency $\nu = \frac{1}{\omega}$. Those solutions can be written as a sum of sine waves with frequencies multiples of the fundamental frequency:

$$x(t) = \sum_{i=1}^{\infty} A_i \sin i\omega t \tag{2.45}$$

Truncating to the fundamental harmonic:

$$x(t) \approx A \sin \omega t \tag{2.46}$$

And again considering a Duffing oscillator $-f_{nl}(x) = \beta x^3-$ applying Galerkin on the residual for the trial function $\sin \omega t$ yields the same algebraic equation than simply taking the first equation from system 2.40 and setting to zero B, C and F:

$$3\beta A^3 + 4KA - 4MA\omega^2 = 0 \tag{2.47}$$

Equations 2.40 define a frequency response curve of 2.2, while the equation 2.47 determines the modal backbones of 2.44, that is, the amplitude of its free response with respect to its pulsation or frequency. The curves obtained as outputs after applying Newton-Raphson on them numerically are described on 2.3.2, and the Matlab codes used to compute and plot the numerical results of the Newton-Raphson procedure will be shown in appendix A.

The aforementioned algebraic equations have been derived with computational assistance with a computer algebra system –CAS– environment called Maple. The codes of the Maple sheet written with that purpose are found in appendix B.

2.3.1.2 An example in two degrees of freedom

The set of two differential equations that model our nonlinear forced two degrees of freedom oscillator shown in 2.4 are:

$$\begin{aligned} M_1x_1'' + C_1x_1' - C_2(x_2' - x_1') + K_1x_1 - K_2(x_2 - x_1) - \beta(x_2 - x_1)^3 \\ -F_{11}\sin\omega t - F_{12}\cos\omega t = 0 \\ M_2x_2'' + C_2(x_2' - x_1') + K_2(x_2 - x_1) + \beta(x_2 - x_1)^3 \\ -F_{21}\sin\omega t - F_{22}\cos\omega t = 0 \end{aligned} \tag{2.48}$$

Suppose an approximate solution for the steady-state displacement of the two masses as harmonic oscillations:

$$\begin{aligned} x_1(t) &\approx A_1 \sin \omega t + B_1 \cos \omega t \\ x_2(t) &\approx A_2 \sin \omega t + B_2 \cos \omega t \end{aligned} \tag{2.49}$$

where A_1, B_1, A_2 and B_2 are the coefficients determining amplitude and phase of the displacements of the masses.

Applying Galerkin orthogonalization on the residuals of the SODE:

$$\begin{aligned} \int_0^{\frac{2\pi}{\omega}} R_1(t) \sin \omega t dt &= 0 \\ \int_0^{\frac{2\pi}{\omega}} R_1(t) \cos \omega t dt &= 0 \\ \int_0^{\frac{2\pi}{\omega}} R_2(t) \sin \omega t dt &= 0 \\ \int_0^{\frac{2\pi}{\omega}} R_2(t) \cos \omega t dt &= 0 \end{aligned} \tag{2.50}$$

results in a system of four algebraic equations with four dependent variables and the coefficient ω: $\mathbf{G}(A_1, A_2, B_1, B_2, \omega) = \mathbf{0}$. The four equations conforming $\mathbf{G}$ are displayed below:

$$
\begin{aligned}
&-3\beta A_2 B_1^2 - 4F_{11} - 3\beta A_2 B_2^2 + 9\beta A_2^2 A_1 - 6\beta B_2 A_1 B_1 + 6\beta A_2 B_2 B_1 \\
&-4C_1 B_1\omega - 4C_2 B_1\omega - 4M_1 A_1\omega^2 + 4C_2 B_2\omega - 4K_2 A_2 - 3\beta A_2^3 \\
&+3\beta A_1^3 + 4K_2 A_1 + 4K_1 A_1 + 3\beta A_1 B_1^2 - 9\beta A_2 A_1^2 + 3\beta B_2^2 A_1 = 0 \\
\\
&-6\beta A_2 A_1 B_1 + 6\beta A_2 B_2 A_1 + 4C_1 A_1\omega + 4C_2 A_1\omega - 4M_1 B_1\omega^2 \\
&-4C_2 A_2\omega + 9\beta B_2^2 B_1 - 9\beta B_2 B_1^2 - 3\beta A_2^2 B_2 + 3\beta A_2^2 B_1 - 3\beta B_2 A_1^2 \\
&+3\beta A_1^2 B_1 + 4K_2 B_1 + 4K_1 B_1 + 3\beta B_1^3 - 3\beta B_2^3 - 4K_2 B_2 - 4F_{12} = 0 \\
\\
&-3\beta A_2 B_1^2 - 3\beta A_2 B_2^2 + 9\beta A_2^2 A_1 - 6\beta B_2 A_1 B_1 + 6\beta A_2 B_2 B_1 \\
&-4C_2 B_1\omega + 4M_2 A_2\omega^2 + 4C_2 B_2\omega - 4K_2 A_2 - 3\beta A_2^3 + 3\beta A_1^3 \\
&+4F_{21} + 4K_2 A_1 + 3\beta A_1 B_1^2 - 9\beta A_2 A_1^2 + 3\beta B_2^2 A_1 = 0 \\
\\
&3\beta B_1^3 - 6\beta A_2 A_1 B_1 + 6\beta A_2 B_2 A_1 + 4F_{22} - 3\beta B_2^3 - 4K_2 B_2 \\
&+4K_2 B_1 - 3\beta A_2^2 B_2 + 9\beta B_2^2 B_1 - 9\beta B_2 B_1^2 - 4C_2 A_2\omega \\
&+4M_2 B_2\omega^2 + 4C_2 A_1\omega - 3\beta B_2 A_1^2 + 3\beta A_1^2 B_1 + 3\beta A_2^2 B_1 = 0
\end{aligned}
\tag{2.51}
$$

This set of equations was obtained with the same computational assistance of Maple as the one-dimensional algorithm described in the previous section –2.3.1.1–. See Maple sheets in the appendix B.

Another Matlab code was written to compute the solutions of the equations by means of a Newton-Raphson algorithm, obtaining a plottable forced response curve. Those curves and diagrams are depicted and described in the following section 2.3.2. The codes written are found in appendix A.

Analogously to the linear case, the amplitude of each of the harmonic solutions is derived from the coefficients of $\sin\omega t$ and $\cos\omega t$:

$$\begin{aligned} Amplitude\ dof\ 1 &= \sqrt{A_1^2 + B_1^2} \\ Amplitude\ dof\ 2 &= \sqrt{A_2^2 + B_2^2} \end{aligned} \tag{2.52}$$

while the phase angle of the response of each degree of freedom depend on the values and signs of the coefficients.

The modal backbones of the nonlinear two degrees of freedom oscillator 2.5 can be computed from the equations of the free undamped model, which can be derived from 2.4:

$$\begin{aligned} M_1 x_1'' + K_1 x_1 - K_2(x_2 - x_1) - \beta(x_2 - x_1)^3 &= 0 \\ M_2 x_2'' + K_2(x_2 - x_1) + \beta(x_2 - x_1)^3 &= 0 \end{aligned} \tag{2.53}$$

We seek for periodic solutions in the phase space, approximated to sinewaves with a single fundamental harmonic of pulsation ω:

$$\begin{aligned} x_1(t) &\approx A_1 \sin \omega t \\ x_2(t) &\approx A_2 \sin \omega t \end{aligned} \tag{2.54}$$

Once more, applying Galerkin on the two residuals derived from introducing the solutions of 2.54 into 5.1 results in the following two algebraic equations with variables A_1 and A_2, and independent parameter ω. Notice that they can be also obtained from equations one and three of 2.51 setting B_1, B_2, C_1, C_2 and all the F_{ij} coefficients to zero:

$$\begin{aligned} 9\beta A_2^2 A_1 - 4M_1 A_1 \omega^2 - 4K_2 A_2 - 3\beta A_2^3 \\ +3\beta A_1^3 + 4K_2 A_1 + 4K_1 A_1 - 9\beta A_2 A_1^2 = 0 \\ 9\beta A_2^2 A_1 + 4M_2 A_2 \omega^2 - 4K_2 A_2 - 3\beta A_2^3 \\ +3\beta A_1^3 + 4K_2 A_1 - 9\beta A_2 A_1^2 = 0 \end{aligned} \tag{2.55}$$

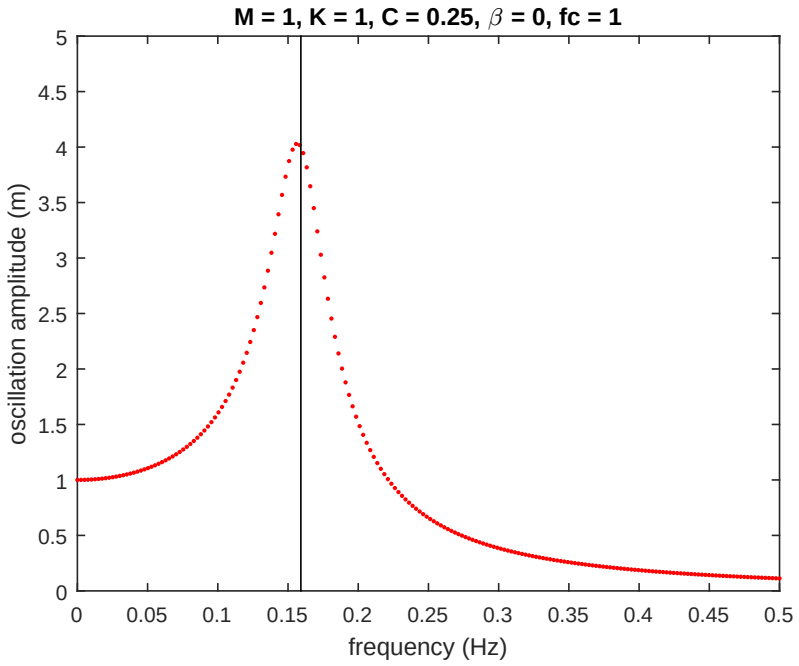
The modal backbones these equations define for the model shown in 2.5 are computed along with the forced response curves that the set of equations 2.51 determine for the equivalent undamped model on 2.4 with the same Matlab codes shown in the appendices.

As a final remark, notice that all the linear algebraic equations derived in 2.2.2 defining the frequency and modal response of the linear SDOF and 2dofs oscillators can be obtained directly from the nonlinear algebraic equations seen in this section that approximate the motions of the same nonlinear oscillators, simply setting β to zero.

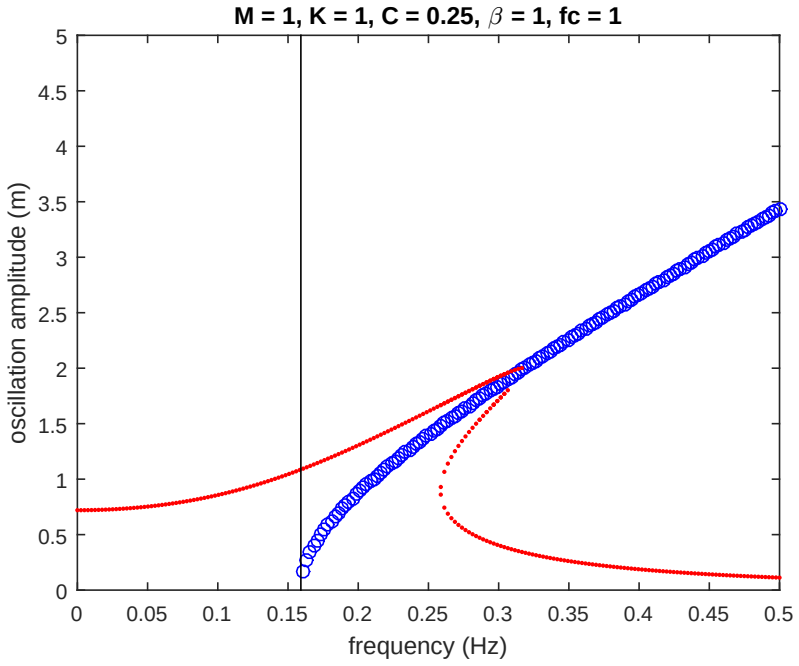
2.3.2 Nonlinear vibration

From equations 2.40, 2.47 –for the SDOF oscillator–, 2.51 and 2.2 –for the 2dofs oscillator– the frequency response curves and the modal backbones can be computed by setting the parameter ω –the pulsation– to a number of values and finding the coefficients that solve the equations. The results can be plotted on frequency response diagrams –FRDs– such as the ones shown in this text.

First, the dynamical behavior of a single degree of freedom oscillator with certain model parameters will be described from the FRDs shown below. The linear oscillators –$\beta = 0$, upper figure– are contrasted with the nonlinear oscillator –lower figure– with same model parameters but non-zero β:



(A) Linear SDOF oscillator.

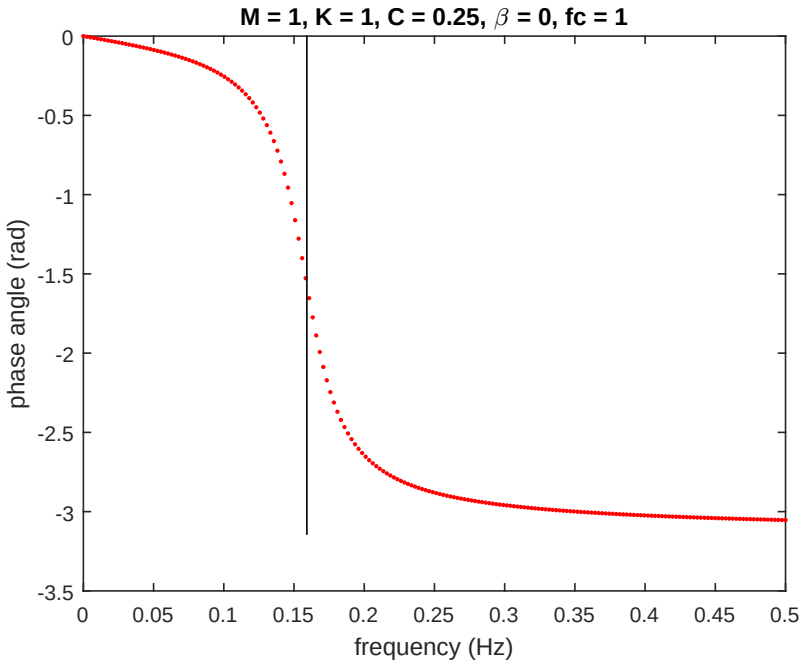


(B) Nonlinear SDOF oscillator.

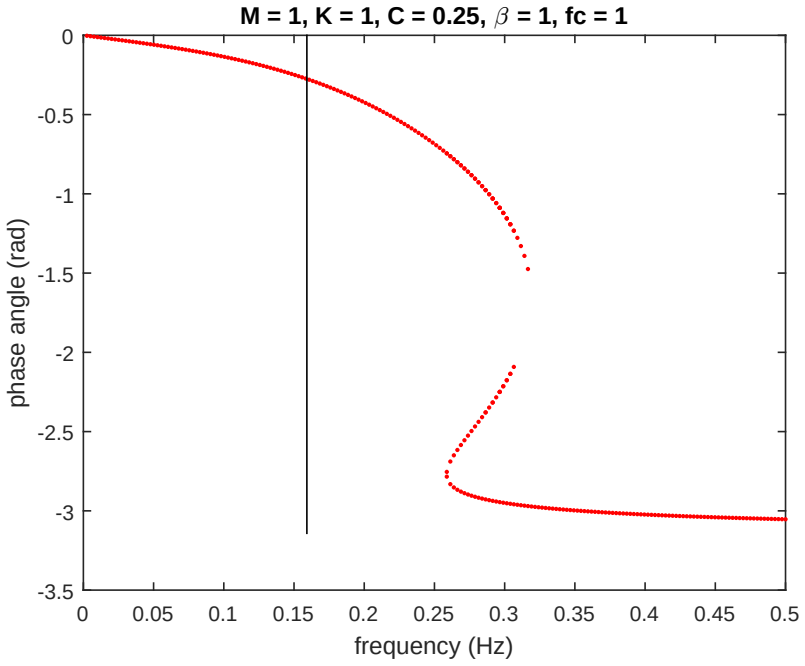
FIGURE 2.6: FRDs with the frequency response curves and the modal backbones of the SDOF oscillator.

The dotted lines on the FRDs represent the forced response curves of the models. Blue circles or crosses –the latter on the 2dofs model diagrams– represent the modal backbones of the same model with no damping –periodic free motions called modes–. The amplitude of mass displacement is plotted with respect to forcing frequency –for the frequency response curves– or frequency of free motion –for the modal backbones–. In the particular case that the systems are linear –$\beta = 0$–, the frequencies of free periodic oscillations are invariant to displacement amplitudes or energies and the modal backbones are vertical lines on the FRD over the natural frequencies. The later is a well-known property of linear systems related to the invariance of their normal modes –see 2.2.1 or any reference on Vibration Theory such as [27]–. Linear backbones have been depicted as black lines on all FRDs displayed here.

In a linear system the frequency response curve has its resonance peaks in the vicinity of the natural frequencies as in 2.6a. However, when nonlinear stiffnesses are present in the system, the resonance peak of the forced response 'bends' due to the presence of nonlinear terms in the equations –2.6b–. For a positive β –hardening nonlinearity– the resonance bends to the right side of the FRD as in the figure above while in case of a negative β –softening nonlinearity– it would bend to the left side. The bending has a direct consequence: there exist an interval of forcing frequencies between the natural frequency and a certain greater frequency –or smaller if $\beta \leq 0$– where the forced response has three different solutions for the amplitude of the responses; in other words, there are two bifurcation points in the frequency response where the number of solutions switches from one to three and viceversa. The larger the absolute value of β the greater that frequency interval. If a stability analysis was performed on 2.2 at those multiple forced steady-state solutions in that interval of frequencies we would find out that for each forcing frequency, the solutions with the largest and the smallest amplitude are stable –i.e. a small perturbation on the system while the motion is at that state would bring the system back to the same state of motion– but the intermediate solution is unstable –a small perturbation from that regime of motion would force the system out of that state only to converge to one of the two stable solutions–. For any other forcing frequency out of that interval the steady-state solution is unique and stable –see [9] and [11]–.



(A) Linear SDOF oscillator.

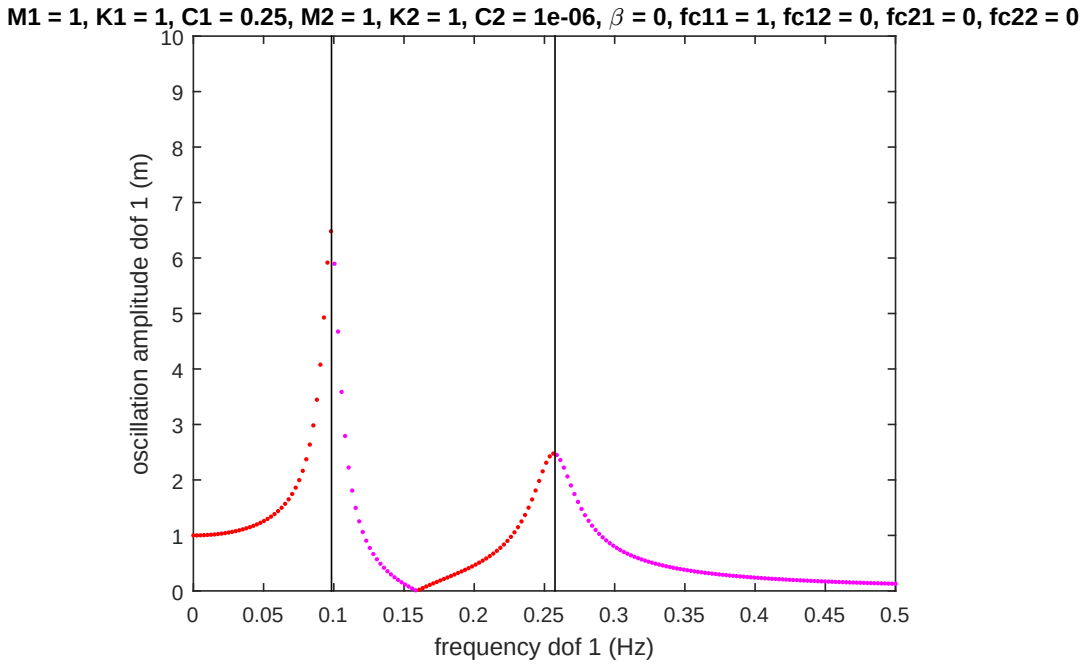


(B) Nonlinear SDOF oscillator.

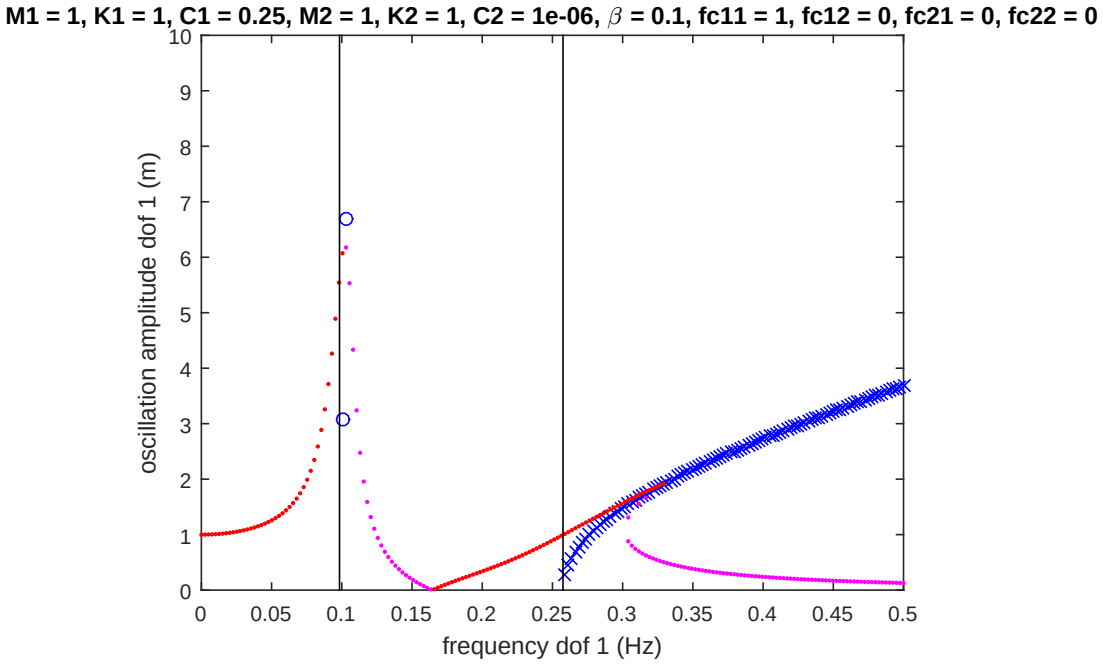
FIGURE 2.7: FRDs with the response delay to excitation –phase angle– of the SDOF oscillator.

The backbones of the conservative system also have the same 'bending' property, thus reflecting the dependence between the amplitude of the periodic free vibration of the oscillator and its frequency. This dependence and can be positive or negative according to the sign of the β coefficient, and its sharpness depends on its absolute value. It can be clearly seen on diagram 2.6b that the orbits of the free periodic oscillation –with their frequency and amplitude– are a very good approximation to the frequencies and shapes the system could attain at the resonance peaks. This quality is shared by systems with any number of degrees of freedom and determines a very useful application of modal backbones: to predict the shapes and frequencies that the system could adopt while passing through a resonance peak.

The phase angle of the same forced responses from linear and nonlinear models are portrayed on 2.7. The response lags the excitation at any forcing frequency, while the phase difference is $\frac{\pi}{2}$ at the resonance peak. The less damping in the system, the sharper the shift in phase angle around the resonance. The unstable branch of the response curve contains solutions with phase angles greater than $\frac{\pi}{2}$ while the branch of stable solutions with the lowest steady-state amplitudes displays the greatest phase lags of all the possible solutions. For large frequencies the phase angle converges to π –out-of-phase oscillations– while the phase angle remains close to zero for small frequencies. The black vertical lines represent the natural undamped frequencies of the system.

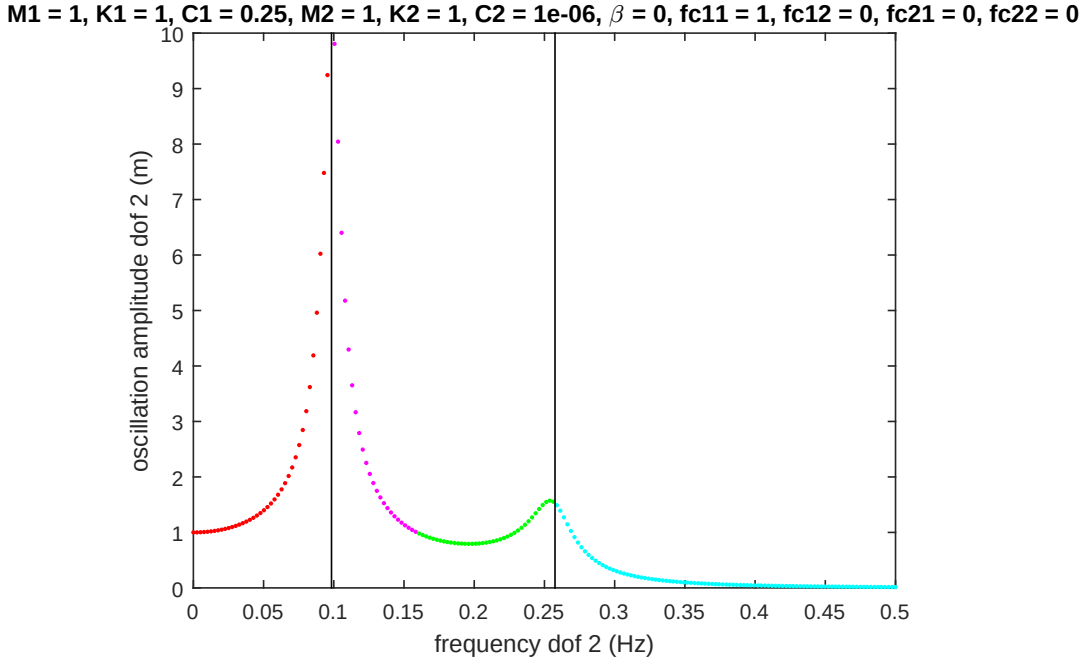


(A) Linear 2dofs oscillator, mass one.

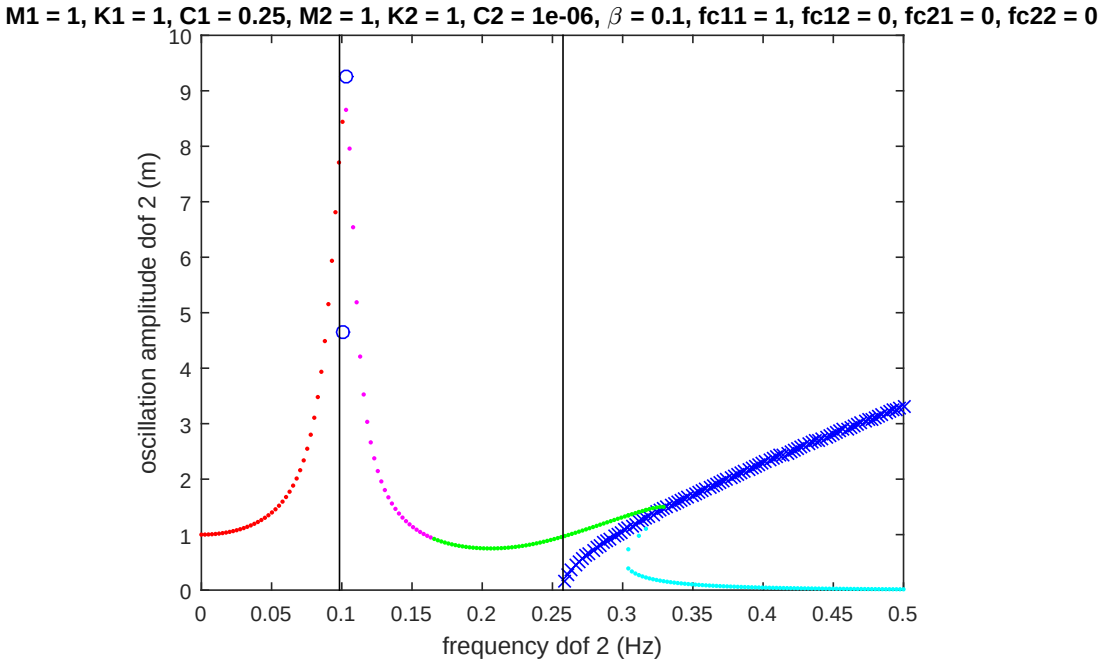


(B) Nonlinear 2dofs oscillator, mass one.

FIGURE 2.8: FRDs with the frequency response curves and the modal backbones of the 2dofs oscillator, portraying amplitude and phase of mass one



(A) Linear 2dofs oscillator, mass two.



(B) Nonlinear 2dofs oscillator, mass two.

FIGURE 2.9: FRDs with the frequency response curves and the modal backbones of the 2dofs oscillator, portraying amplitude and phase of mass two

For systems with multiple degrees of freedom the properties of the nonlinear frequency response and its differences with the linear response are analogous with the SDOF oscillator. Nonetheless, it is worth mentioning that each resonance has an associated modal backbone, although all modal backbones are the solutions of the same equations. And for nonlinear components such as a Duffing term –$f_{nl}(x) = \beta x^3$– if the amplitudes –i.e. the energy of the oscillations– are small enough, the system mode shapes do not differ much from those of the linear system $\beta = 0$ and the properties of linear systems hold with great accuracy.

2dofs oscillator 2.5 has these two linear normal modes: an in-phase low-frequency first mode of shape $(0.618, 1)$ and an out-of-phase high-frequency second mode of shape $(1, -0.618)$. If the system is undamped or slightly damped, a forcing at one of the resonance frequencies would make its motion tend to huge amplitudes in the long term while the amount of mechanical energy absorbed by each mass remains similar to the mode shape –same oscillation amplitudes for this particular case with the given parameter values–, being in-phase or out-of phase, depending on the excited mode. However, the presence of greater damping would force the two masses to oscillate with energies out of proportion with the mode shapes, changing the system shape at the resonance peaks with respect to the modes of the conservative system. Damping also reduces the resonance frequencies with respect to the natural frequencies.

Different colours of the forced response curve mean different phases of the responses of each degree of freedom:

$$\begin{aligned} A_i > 0; B_i \leqslant 0 &\rightarrow Red\ spot \\ A_i \leqslant 0; B_i < 0 &\rightarrow Magenta\ spot \\ A_i < 0; B_i \geqslant 0 &\rightarrow Green\ spot \\ A_i \geqslant 0; B_i > 0 &\rightarrow Cyan\ spot \end{aligned} \tag{2.56}$$

From the coefficients signs and numerical values the phase angles between the oscillations of each dof –or between excitations and responses– can be infered. Both masses oscillate in-phase at the first resonance, and out-of-phase at the second resonance. Equally, both masses oscillate in-phase along the first modal backbone and out-of-phase along the second modal backbone. On the figures shown –2.8b and 2.9b– the in-phase modal backbone has been plotted with blue circles, while the out-of-phase mode backbone is represented by blue crosses.

Notice that there is a frequency value between the two resonances for which the steady-state amplitude of mass one is zero, that is, it remains steady while all the mechanical energy is absorbed by mass two and the dampings. This phenomenon is called 'tuning' and is broadly exploited in mechanical design for vibration control.

In the nonlinear cases it could happen that as the energy increases along the modal curve the energy is transfered from one mode to another, therefore changing the mode shapes. This phenomenon is called 'mode localization' and is characteristic of nonlinear systems. Of course, due to the similarity between the dynamics of the normal modes represented by the backbones and the resonances, the latter are also affected by localization phenomena.

Another important distinction from linear sistems has to do with the principle of linear superposition, which simply does not hold in nonlinear dynamical systems. Superposition means that if the system has several inputs –forces on the dof– the solution can be written as the sum of the solutions of the system motion under each force. Forced solutions can also be obtained as the sum of so-called 'modal terms' –[8], [27], [3]–. Another application of superposition comes for free undamped linear systems, where any motion could be expressed as a linear combination of its normal modes. However, the normal modes –periodic free oscillations– of nonlinear systems cannot be added and combined to form solutions in the phase space of the free undamped system.

The bending of the forced resonance peak causes a phenomenon called 'hysteresis', as portrayed on figure 2.10. Suppose a system with $\beta \geq 0$ with FRDs such as 2.8b and 2.9b under an excitation with a forcing frequency well under the resonance –Point five of the diagram on 2.10 for $\beta \geq 0$–. Now the frequency of the input is increased very slowly until the resonance peak has been passed and a bifurcation point has been reached

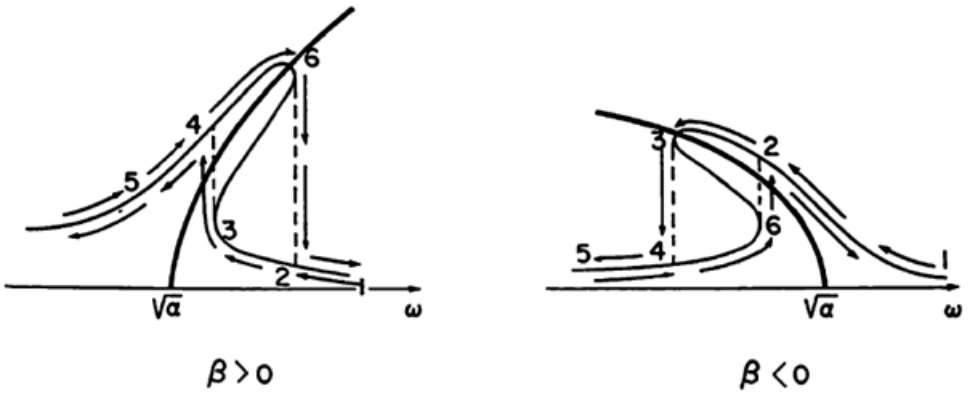


FIG. 4.1. Jump, or hysteresis, phenomena.

FIGURE 2.10: Jump and hysteresis phenomenon around a resonance in a nonlinear forced system. This figure has been taken from reference [9]

–point six–. At this instant there is a sudden reduction in the amplitudes of the frequency response as a result of a jump from one branch of the curve to another with lower energy for greater frequencies –point two –. Then, the forcing frequency is slowly diminished to frequencies smaller than the resonance peak, however, the system vibrates according to the branch of lowest oscillations until a second bifurcation is reached –point three– and the motion 'jumps' to a state of greater oscillations given by the highest branch –point four–. These discontinuous energy jumps between different regimes of vibration when the forcing frequency is increased or decreased is what is called 'hysteresis' and is present in many real physical systems in Engineering and nature.

The last important property of nonlinear systems that will be described in this text is the presence of harmonics and subharmonics in the response: the motion could contain frequencies different to the frequency of excitation, in contrast to linear systems where the frequencies of the response were the frequencies of the excitation. Those new frequencies are multiples of the fundamental frequency of excitation. Those terms with frequencies lower than the forcing frequency are called 'subharmonics' and are less common than higher harmonics –[16]–. This phenomenon is easy to understand if we recall that the solution of a nonlinear system such as 2.24 can be expressed as a sum of harmonics –or Fourier series– where each term is an harmonic of the motion of the system:

$$x_i(t) \approx A_i \sin\omega t + B_i \cos\omega t + C_i \sin 2\omega t + D_i \cos 2\omega t + E_i \sin 3\omega t + F_i \cos 3\omega t + ...$$

Harmonic terms that appear in both free and forced periodic motions represented by the modal backbones can cause a phenomenon called 'internal resonance': One of the harmonics of an NNM can resonate with a higher-frequency nonlinear mode thus producing large amplitudes in that harmonic term. Very often, those resonant motions are confined to a few degrees of freedom of a system, thus breaking the synchronicity that the modes held in their motion and that was characteristic of the linear modes –see 2.5.3.2–. Internal resonances are generated through bifurcations –see 2.5.3.3–, as does another feature characterizing nonlinear dynamics: the ramification of NNMs.

Finally, we have to keep in mind that the frequency response curves and the modal backbones of the nonlinear systems studied in this section are no more than heuristic approximations of the solutions to the first harmonic. Nonetheless, numerical methods such as sequential continuation –section 2.5– have the potential to find solutions of huge accuracy with any number of harmonics.

Matlab codes for the computation and plotting of the diagrams shown are found in appendix A.

References on the topic of nonlinear dynamics, vibrations and oscillations have been given throughout the chapter. For a further insight on the elementary concepts presented in this section there is a wide range of open access literature such as lecture notes in [16].

2.4 Nonlinear modal analysis: Nonlinear normal modes

'Modal analysis' refers to the analysis of the dynamic caracteristics of mechanical systems. It is based on the determination of natural frequencies, damping factors and mode shapes –[8]–. In the linear case there is energy invariance of both natural frequencies and mode shapes –[8] and [27]–, however, this property does not hold in nonlinear systems as it was shown numerically and graphically in the former section.

Nonetheless, the computation of the system modal backbones would still provide very useful information about its forced dynamics. Those modal backbones comprised a sequence of points representing the periodic solutions of the free conservative –undamped– system called 'modes', approximated to the fundamental frequency, and displayed on a FRD according to the oscillation frequency and amplitude. The set of those solutions constitute the 'nonlinear normal modes' –NNMs– of the conservative free undamped system. In the following section a more rigorous definition will be given.

2.4.1 What nonlinear normal modes –NNMs– stand for

In the last section the dynamics of nonlinear mechanical systems have been discussed, performing first an heuristic computation of the forced response diagrams for 1dof and 2dofs oscillators and then overlaying on the diagrams a particular set of solutions or trajectories of the equivalent free undamped –conservative– system for each model. These sets of solutions had a fundamental property over the rest: the motion of the degrees of freedom in the configuration space was periodic. This particular choice was not made at random, these solutions of the free undamped model had properties that made them very useful. In the one-dimensional case these solutions were all possible solutions in the phase space because there are not other nodes to synchronize to, but in MDOF systems the distinction makes a mathematical sense.

For linear mechanical systems we could also obtain their frequency response curves, but on top of that the interest was on their free undamped response, neglecting damping and excitation. Free response was the linear combination of basis functions that were

for themselves solutions of the conservative model. These particular solutions were called 'normal modes' and were the product of a vector called 'mode shape' and a sine wave with a particular frequency called 'natural frequency'. Normal modes were particular solutions of the conservative unforced models but could help to understand the dynamics of both free and forced systems in a way that has been discussed earlier in the text.

From this moment, normal modes of linear systems will be called in this text 'linear normal modes' or LNMs while normal modes derived from the dynamical analysis of nonlinear conservative systems will be refered to as 'nonlinear normal modes' or 'NNMs'. In short, nonlinear normal modes in those systems are an analogous concept to linear normal modes in linear conservative systems, but they differ in many of their properties due to the nature of nonlinear dynamics. There are two main definitions for the nonlinear normal modes:

Rosenberg's definition The first definition was purposed by Rosenberg –[29]–. He defined NNMs as a 'vibration is unison' or 'synchronized vibration' where all the nodes vibrate with the same frequencies and with a fixed displacement ratio. In other words, all dofs must reaches extreme values and pass through zero at the same time. In this sense, that synchronism would be a property shared with linear normal modes. However, this definition has two main restrictions: first, it cannot be extended to nonconservative systems –a case that is not a matter of discussion in this text, treating only conservative undamped systems–. Secondly, if two or more MMNs interact and internal resonance takes over in the system some nodes of it can vibrate with different dominant frequency than others, In that case, synchronicity is broken.

In the latter case Rosenberg's definition can be extended to 'non-necessarily synchronous periodic motion of the system', because in the case of internal resonances the motion is still periodic. This latter definition is the closest to what we had understood as 'NNMs' in the previous section: the syncronicity, which is characteristic of linear modes, may be lost, but their periodicity remains.

Shaw and Pierre's definition Another definition that extended the notion of NNMs to damped systems was purposed by Shaw and Pierre –[30]–, this time based on a

geometric approach. They defined an NNM as an two-dimensional invariant manifold in the phase space. It is invariant because any orbit that starts in that manifold remains on it at any instant of time. This is a property shared with linear normal modes. The invariant manifold can be parametrized with two state variables called 'master coordinates' such as the position and the velocity of a particular degree of freedom. The position of any point on that manifold could then be expressed as a function of those two parameters, making it possible to compute any orbits on it. This is the foundation of the 'invariant manifold method' to approximate NNMs analytically –see section 2.4.3–.

NNM invariant manifolds can be regarded as an extension of LNMs, which are hyperplanes in the phase space due to their invariability and non-dependence on their energy –more in 2.4.2–. A graphical example showing the concept of a NNM invariant manifold along with its linear counterpart has been displayed on figure 2.11. It can be noticed that for small energy levels LNMs and NNMs overlap due to the insignificance of nonlinearities for small vibration amplitudes. This property may spur us to define nonlinear normal modes as the nonlinear extensions of the linear normal modes of any system. In fact, the Lyapunov's Theorem –1909– confirms that any N-dof conservative system with no internal resonances has at least N families of normal modes around the equilibrium point. Internal resonances between modes lead to bifurcations that increase the number of NNMs –see 2.5.3.2–.

A good introductory reference on nonlinear normal modes and nonlinear normal analysis can be found in –[13]–. This conference proceedings –[17] and [18]– and in this website: `http://www.ltas-vis.ulg.ac.be/cmsms/index.php?page=sicon_tc5`. It posts slideshows of lectures given in SICON courses by researchers of the University of Liège –Golinval, Vakakis, Kerschen et al– on the topic of Vibration and NNMs.

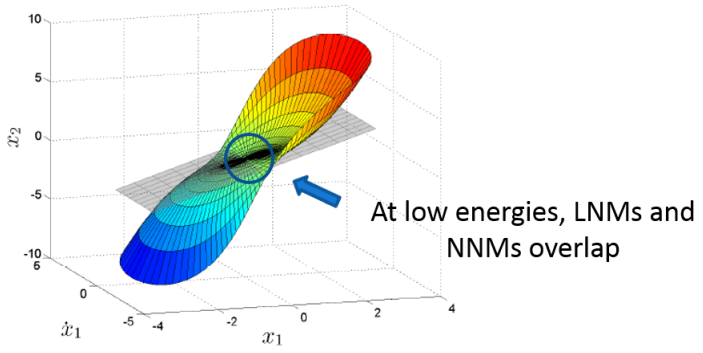


FIGURE 2.11: NNM invariant manifold in the phase space –in colour– and invariant manifold of the same linearized mode –in grey–. Both manifolds overlap near the origin, where nonlinearities can be neglected. The LNM manifold is an hyperplane, while the NNM manifold has a curvature in the phase space due to frequency-energy and shape-energy dependence. This figure was originally shown on reference [17]

2.4.2 Linear normal modes versus nonlinear normal modes

The similarities and differences between modes in linear and nonlinear systems can be summed up as follows:

Superposition principle Superposition is a fundamental property of linear systems. It simply states that the solution of a system whose input is a sum of different inputs –the forces exciting the system– is the sum of the individual solutions of the same system subjected to each of the individual inputs conforming the global input. In free conservative systems superposition also allows to write any possible solution in the phase space as the linear combination of a basis of solutions called 'linear normal modes'.

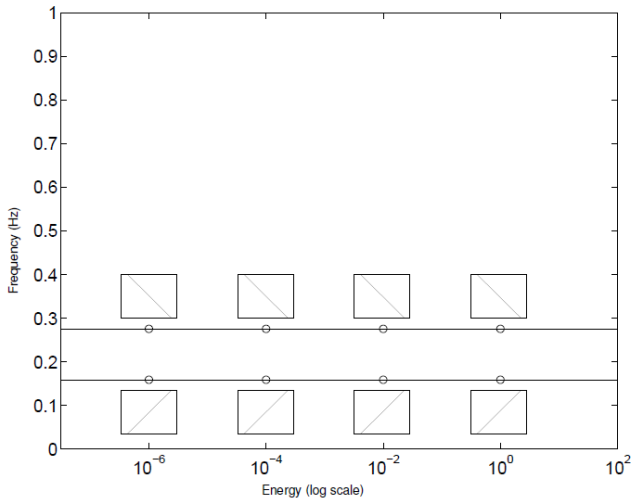
Orthogonality Linear normal modes are orthogonal. That means the dot product of two different modes in the phase space of a linear conservative system will be always zero. The direct algebraic consequence is that the modes form a linearly independent basis where all possible linear combinations from the basis can span the whole phase space. Nonlinear normal modes, however, are not orthogonal.

Modal superposition Superposition principle allows to obtain any oscillation of the free conservative linear system from a linear combination of modes. This is simply unfeasible in nonlinear systems.

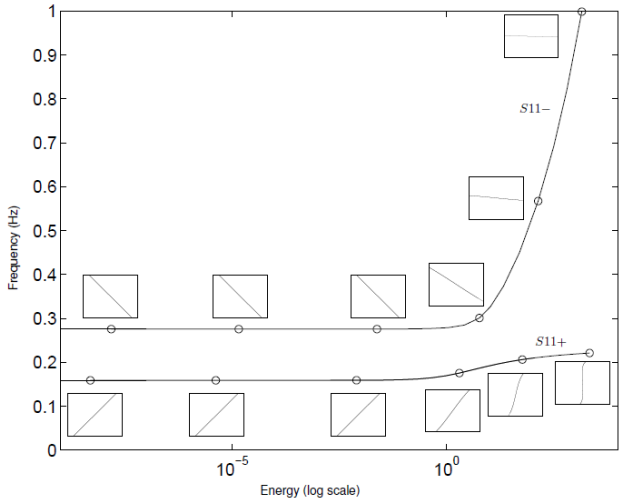
Frequency-energy dependence Linear normal modes are invariant to energy: both the mode shapes and their associated vibration frequencies remain unchanged at any energy level. Moreover, mode shapes of linear normal modes are straight lines elipses in the phase space as they are sinusoidal functions with only one fundamental harmonic. However, shapes and frequencies of NNMs are dependent on the energy of oscillation –see figures in 2.12–. This can be noticed on the modal backbones displayed on the frequency-energy diagrams of the models –2.6, 2.8 and 2.9–. A depencence between the frequency of the free oscillations and the amplitude of those oscillations in the dofs is noticable.

In contrast, modal backbones of linear normal modes –figure 2.12a– are horizontal on a frequency-energy plot or FEP –more on FEPs in section 2.5.2.1– while their mode shapes remain invariant. At low energies, however, LNMs –the black vertical asymptotes on the FRDs– resemble the NNMs. All mode shapes of LNMs shown on the phase diagrams of a sequence of modes on the curves are straight lines.

Invariance This is a property NNMs and LNMs share, derived from the Shaw and Pierre definition of NNM. Trajectories that begin in the invariant manifold remain on that manifold for all times. This quality holds in both conservative and non-conservative –damped– systems, Even though in the latter case the motion decays and converges to the origin of the phase space. Invariance is very useful in experimental modal analysis because it makes possible to reconstruct experimentally the modal backbones of a frequency-energy plot. This is done by progressively forcing a system until a resonance peak has been isolated before turning off the excitation and letting the vibration decay naturally in time.



(A) This is a FEP of the normal modes of a linear conservative system. Mode frequencies and shapes are invariant here for any energy level.



(B) This FEP shows how the frequences and mode shapes of the NNMs of a 2dofs system evolve and change with the energy level at each point of the branches. The changes in shape seen here are a sign of the ocurrence of 'mode localization' –see 2.5.3.1– while the increase of the frequency of vibration with energy is related to a hardening stiffness.

FIGURE 2.12: Frequency-energy plots –FEP, see 2.5.2.1– of a linear and nonlinear mechanical system. These figures have been taken from reference [17]

Stability Linear normal modes are always stable. Meanwhile, NNMs can be stable or unstable. A point of stability change on a backbone or 'branch' of NNMs will be a 'bifurcation point'.

This dychotomy between stable and unstable solutions or trajectories in a system is a feature that characterizes nonlinear dynamics. In Dynamical Systems Theory, a trajectory is said to be 'stable' if a small perturbation of the initial conditions lead to the motion of the new solution to converge to the previous motion, and 'unstable' if small perturbations lead to trajectories that drift apart from the previous trajectory as time goes on.

In nonlinear forced response there were frequency intervals at the neighborhoods of the resonances with three possible steady-state solutions: two were stable and one unstable –2.3.2–. In free response there is also a distinction between stable and unstable orbits which has to be accounted for during the calculation or computation of the NNMs of a system. Stability analysis is very important because the unstable modes are motions that should never arise in the free motion of real-life systems. More on computational algorithms for stability analysis in section 3.9 of the next chapter.

Number of modes In linear systems the number of modes equals to the number of dof or coordinates. However, internal resonances or the ramification of new NNMs branches from bifurcation points increase the number of system modes over its number of degrees of freedom.

Synchronism Those internal resonances can also break the synchronism of the oscillation of all the degrees of freedom, forcing some of them to oscillate with frequencies other than the fundamental frequency. Linear normal modes always oscillate 'in unison'.

Forced resonance Both linear and nonlinear normal modes can result in good approximations of the oscillation shape and displacements –the system response– in the neighborhood of a resonance peak while the system sustains a forced sinusoidal excitation on any of its points, even if the real system is slightly damped. Also, due to bifurcation and internal resonances, the number of resonances of a nonlinear system could be greater than its number of degrees of freedom, unlike in linear systems.

2.4.3 Other analytical methods for NNMs calculation

Here I will outline very briefly other analytical techniques for the estimation of NNMs in the phase space of a nonlinear mechanical dynamical system apart from the harmonic balance method.

2.4.3.1 The energy-based formulation

This method results in an expression of a NNM in the configuration space. The foundation of the method is to choose one of the degrees of freedom of the system and to consider the motion of the remaining coordinates as different functions of that dof. Then these functions have to be used through time chain differentiation in order to eliminate the time derivatives.

The elimination of those time derivatives would require at some step to integrate the equations of motion applying energy conservation. The integration limits determine the maximum amplitude reached by the independent degree of freedom chosen at the beginning of the method and therefore determine the energy of the trajectory sought. This integrals must be accompanied by a boundary condition in order to have it univocally solve. The boundary condition imposes on the solution an important property that normal modes and periodic solutions in conservative systems have: their trajectories in the configuration space of displacements are normal to the maximum equipotencial ellipsoid –a surface in the phase space with a constant energy, equal to the potential energy of the NNM calculated–.

The solution to the integral can be expressed as a power series.

2.4.3.2 The invariant manifold approach

This method is very similar to the energy formulation. The main difference is that the motion of all system variables: positions and velocities, in the phase space is expressed

as a function of two variables, a position and a velocity, called 'master coordinates': $x_0(t)$ and $x_0'(t)$.

$$\begin{aligned} \mathbf{x}(s,t) &= \mathbf{X_1}[s, x_0(t), x_0'(t)] \\ \mathbf{x}'(s,t) &= \mathbf{X_2}[s, x_0(t), x_0'(t)] \end{aligned} \tag{2.57}$$

This system defines a two-dimensional manifold in the phase space. Elimination of the time variable through chain rule differenciation leads to a set of partial differential equations governing modal functions $\mathbf{X_1}$ and $\mathbf{X_2}$. The solutions to these equations can be approximated with power series.

2.4.3.3 The method of multiple-scales

Multiple-scales method belongs to the family of perturbation methods –[10] is a classical reference on perturbation methods– for dynamical systems and partial differential equations solving. The idea behind this method is to consider the solution for each dof as an asymptotic expansion in terms that are functions of independent time scales $T_0, T_1, T_2, ...$ that replace the variable time. A small parameter ϵ is introduced as a multiplying factor of the nonlinear terms to indicate smallness.

$$\begin{aligned} x_i(s,t) = \epsilon X_{i1}(s, T_0, T_1.T_2, ..., T_N) + \epsilon^2 X_{i2}(s, T_0, T_1.T_2, ..., T_N) \\ +... + \epsilon^{iN} X_N(s, T_0, T_1.T_2, ..., T_N),\ with\ T_k = \epsilon^k T \end{aligned} \tag{2.58}$$

where N is the order of the approximation and T_0 is the time scale for the fast motion of the system, that is, the motion that occurs at the NNM dominant frequency. Slower time scales T_1, T_2... are neccesary because, as we already know, NNM motions are not harmonic. The more time scales added, the more accuracy. Aproximating functions

$X_i(s, T_0, T_1.T_2, ..., T_N)$ are determined by substituting each coordinate with its asymptotic expansion and solving the resulting system of ODEs for the functions $X_{i1}, X_{i2}, ...$ that approximate the motion of the i-th degree of freedom.

2.4.4 Reasons for the nonlinear analysis

Linearity is just a mathematical construct, an idealization. In reality there is no physical system conforming exactly to the hypotesis of linearization. However, many natural processes and engineering systems behave in a way very close to linearity, and mathematical models can be constructed under that assumption. Some examples in Mechanics are the constitutive laws of elastic materials such as metals, structures sustaining small deformations or linear viscous dampings. Engineering has traditionally taken a linear approach to design, modelization and analysis that has kept things simple for centuries.

Nevertheless, the Industry is continuously evolving and demanding more efficient designs, lighter materials, better reliability and lower costs in order to remain competitive and fulfill the demands of the materials. In certain fields, such as Aerospace or High Tech, there is a need to attain performances over what is feasible with today's know-how, turning the concepcion and design of the newest products a big challenge to the old paradigms of Engineering.

Nonlinearity may be present in many ways: hyperelasticity, elastoplasticity, large mechanical deformation, nonlinear boundary conditions of service, fluid-structure interactions, self-sustained forces or simply designs of such nature that cannot be subject to any sort of linearization without great loss of information. In those situations, the linear hypotesis would underestimate the deepest model dynamics and becomes unreliable. Nonlinear analysis is the alternative, even though it is a much more complicated and less developed field, to attain a reliable analysis that could predict the real behavior of a product, structure, mechanism or commodity in the design phase of its life cycle.

Designers can also capitalize on nonlinear dynamics in the concepcion of an engineering system with the purpose of improving performance, reliability, functionality or for other

purposes, and furtheremore, nonlinear analysis tools need to be undertaken for a thorough design according to the product or the system specifications.

Nonlinear modal analysis has been succesfully applied on models of real-life mechanical systems with a large number of dofs, such as on the structures of an aircraft –[22]– and a satellite –[23] and [24]–, with both experimental and computational analysis methods.

2.4.4.1 Practical applications of nonlinearity

In this text I will single out two of the most important practical applications of nonlinear vibration in Mechanics with a huge body of undergone research and published bibliography. Both have been, at the present moment, been successfully considered on structural and mechanical design:

Vibration control The tradictional approach to vibration control in Engineering has consisted in the use of passive systems of protection, such as dissipative elements or mass dampers. Nonetheless, today's standard of performance and reliability cannot be fully reached with these techniques. so nowadays 'active' protection systems are widely used instead. Active control systems rely on the use of actuators that vary stiffness and damping in real time, but they have very high costs and require an energy supply –[31]–. Moreover, those active control systems have unnegligible dimensional constraints leading to a reduction of the mechanism deflections on the linear elastic region –[3]–. A common cause of nonlinearity in those elements is found on constructive clearances between elements or elastic limiting stoppers.

On the other hand, linear passive mass dampers have many limitations that hinder their use for certain applications: they fail to damp out several modes of systems with multiple dofs, being limited to the neighborhood of a vibration mode, and they are unable to mitigate vibrations in nonlinear structures. Those drawbacks can be circumvented with the use of 'nonlinear vibration absorbers' –[31]–. Due to their frequency-energy dependence they are effective in mitigating vibration in a much larger interval of frequencies. The 2dofs oscillator shown on figure 2.4

could be the model of a mechanical structure attached to a nonlinear vibration absorber. To understand this one can compare the frequency response diagrams in figures 2.8 and 2.9. If the masses are attached with a nonlinear spring the vibration mitigation spans a larger interval of frequencies around the 'tuning' frequency than if the attachment was linear. Also, nonlinear phenomena such as energy transfer –or 'mode localization'– and internal resonance can be exploited with the purpose of vibration control –[12] and [2]–.

Some reference books on vibration control with a generalistic approach are [1] and [3].

Nonlinear energy harvesting Energy harvesting is the conversion of environmental energy into electrical energy –[32]–. There is a broad range of applications to energy harvesting, mainly energy powering –in very small scales: miliwatts or microwatts– and monitoring sensors. In what concerns to us, our interest is to pick up the vibrational energy from a mechanical system by means of an oscillator attached to a system of electrical generation –of electromagnatic, piezoelectric or capacitative type, among others– that could convert mechanical energy into electrical energy.

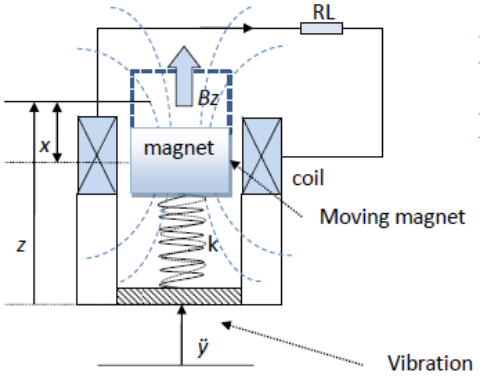
A very considerable limitation of vibration harvesting systems is that they are only effective when the mechanical oscillations are in the neighborhood of a resonance of the global structure of attachment of the harvester. In such conditions the system oscillates close to a local maximum of energy and amplitude and the amount of energy transfered for conversion is optimized, but the neighborhoods of linear resonance are usually very narrow as seen in section 2.3.2.

Consider the two degrees of freedom oscillator represented on 2.4. Let the first mass –first dof– be a mechanism or mechanical structure or system, while the second mass –second dof– represents a vibration energy harvester elastically attached to the first mass, constituting a vibrational energy source. This is an integrated system whose dynamics should be analized as a whole for the design and prediction of the so-called 'frequency bandwidths', which are the frequency intervals around the resonance points maximizing the amount of energy that could be transfered to the harvesting system during a forced regime of vibration. The energy transfered to the

harvester –the second mass– can be converted to electrical power by different means already cited.

Now compare the linear and nonlinear frequency responses of oscillators 2.2 and 2.4 shown in 2.3.2. When a nonlinear stiffness is added to the elastic attachment between the main mechanical structure –mass one– and the harvester –mass two– the frequency bandwith can be greately broadened, that is, more mechanical energy could be effectively harvested in a wider range of frequencies around the resonances. This is a considerable advantage because the most mechanical systems experience multiple excitations at very variable frequencies. Furtheremore, mode localization can also be an advantage if it leads to an increased amount of energy in the harvester.

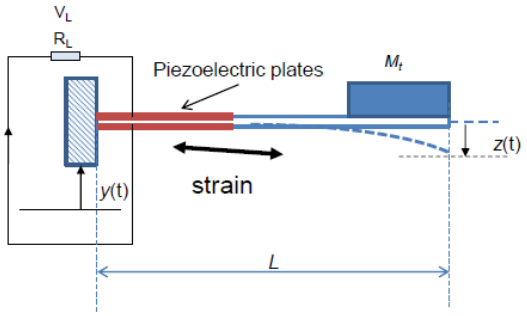
There is a wide amount of research and literature on the topic of nonlinear harvesting and the design of harvesting systems. For a brief introduction one can see references [6], [5] and [33].



(A) Elementary model of the electrodynamic vibration energy harvester –image from [6]–.

(B) Electrodynamic energy harvester Perpetuum PMG17: hppt://www.perpetuum.com/pmg17.asp.

FIGURE 2.13: In an electrodynamic vibration energy harvester, the mechanical energy absorbed by a sprung mass is transfered to an electrical circuit through an inductor. Nonlinearities can be considered in the elements that suspend the sprung mass, the magnetic field around the inductor or the electrical circuit itself.



(A) Model of the piezoelectric vibration energy harvester –image from [6]–.

(B) Piezoelectric energy harvester Midé Volture: http://www.mide.com.

FIGURE 2.14: In a piezoelectric vibration energy harvester, the deformation of a suspended beam with a mass on its tip is converted to electrical power by a piezoelectric plate attached to the beam. The models should be deemed as nonlinear for large deformations of the suspended beam or for nonlinear elasticity or piezoelectricity laws.

2.5 Numerical methods for NNMs

2.5.1 Why to use them

Numerical techniques are those which directy deal with the differential equations of the models relying on extensive computation to obtain the branches of NNMs. One of the approaches developed with that purpose is based on the direct integration of the equations of motion with methods such as Runge-Kutta or Newmark, minimizing the condition of periodicity of the trajectory computed in the phase space –The definition of NNM given in 2.4.1–. Low energy modes and frequencies, which are very similar to the linear normal modes of the linearized model, are computed first. Then, the energy is gradually increased as modes with greater energy are computed taking the previous one as an initial guess in each step, making for an iterative scheme that results in the computation of a branch on NNMs. This technique is called 'sequencial continuation'. A variant of this method is a combination of sequential continuation with shooting algorithms, this being the numerical method that will be described and used in this text for the computation of NNMs branches.

The analytic techniques seen above have several limitations. First, it is only plausible to apply them on simple systems with a low number of dofs. Moreover, they are all heuristic

techniques, and the approximate results that come with their application are only valid for small amplitudes of the dofs displacements. All those methods would require at some extent the use of computational assistance to calculate coefficients of power series or perform integrations, as it was seen when we applied the harmonic balance method on several models. Harmonic balance method has the advantage that it has the potential to deliver solutions valid over larger amplitude ranges –[18]– but such degree of accuracy requires extense computational support, making the harmonic balance method more of a numerical than an analytical technique.

On the other hand, numerical techniques are very suitable to systems of great dimensionality or with strong nonlinearities, and have the potential of providing exact numerical solutions to the NNMs problem –[18]–, while paying the price of requiring great computational power with the methods today available.

The numerical technique that will be considered in this text is the sequential continuation of periodic solutions with a shooting algorithm, and will be covered in chapter 3. With the assistance of a computational environment such as Matlab this method can be automatized and the practising engineer would only need to introduce the model.

2.5.2 Outputs of the NNMs numerical analysis

2.5.2.1 The frequency-energy plot –FEP– diagrams

The convention used by Vakakis, Kerschen, Peeters et al in their publications, papers and proceedings –see in bibliography– to represent graphically the results of the numerical computation of the NNMs is to build a frequency-energy plot or FEP.

FEPs display the computed NNMs branches as sequences of discrete points representing each NNM. These points are plotted according to the fundamental frequency of oscillation and the total energy of the NNM, the latter in a decimal log scale. Remember that due to the energy conservativeness of the equations of free undamped motion the sum of kinetic and potential energy remains constant in any trajectory –3.11–.

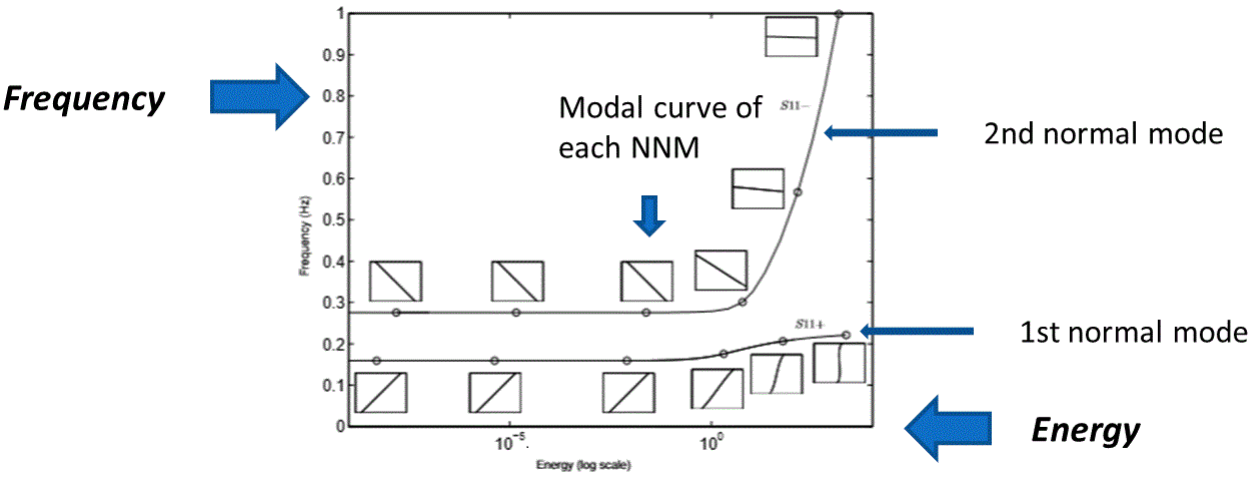


FIGURE 2.15: This is an example of a frequency-energy plot –FEP– that displays the two main branches or NNMs backbones of a 2dofs system. The diagram plots the NNMs energy on the x axix in decimal log scale versus the NNMs fundamental frequency on the y axis. For each point on the branches one can compute the 'modal curve' of the NNM chosen showing the NNM motion on a phase portrait, a 'time-series plot' and the mode stability.

2.5.2.2 Modal curves and time-series plots

Nonlinear normal modes can be defined by their total energy and their fundamental frequency, and each of them has a trajectory in the phase space of the dynamical system considered. The trajectories of the NNMs could be visualized with two different plots:

Modal curves They are analogous to the 'phase portraits' where orbits of dynamical systems can be displayed. The aim is to plot the trajectory of a NNM in the phase space, where each of the cartesian axles represents one coordinate or degree of freedom.

Time-series plots This is a graph where the evolution of the NNM motion of any of the degrees of freedom over time is plotted. The timespan of the representation is the period of oscillation of the NNM, which is the inverse of its fundamental frequency: $T = \frac{1}{\nu}$.

2.5.3 Possible nonlinear behavior of NNMs

In this chapter, I will pay a more detailed view on each nonlinear phenomenon that could affect NNMs and how they should be identified in a frequency-energy plot –More on FEPs in 2.5.2.1–.

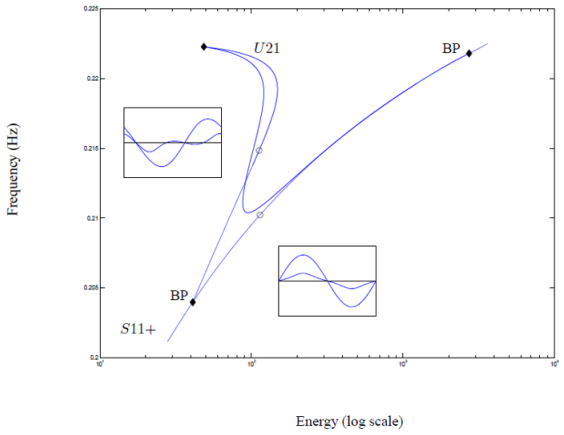


FIGURE 2.16: This is a typical shot of what a FEP should look like after the computation of a modal backbone –a curve representing the mechanical energy of the periodic solutions, NNMs, of the nonlinear free conservative system as a function of its fundamental frequency– in a system as an extension of a linear normal mode, plus new branches that emerge on bifurcation points and the time-series plots –amplitude versus time– that display the trajectories of each degree of freedom in two different points of the curves –white circles– during the minimal period of the modes. This figure has been taken from reference [34].

2.5.3.1 Mode localization

When nonlinearities are present in the model, with energy input into the system is increased that energy will tend to confine to a part of the structure, hence those particular coordinates vibrate with more energy –amplitudes– than other parts of it.

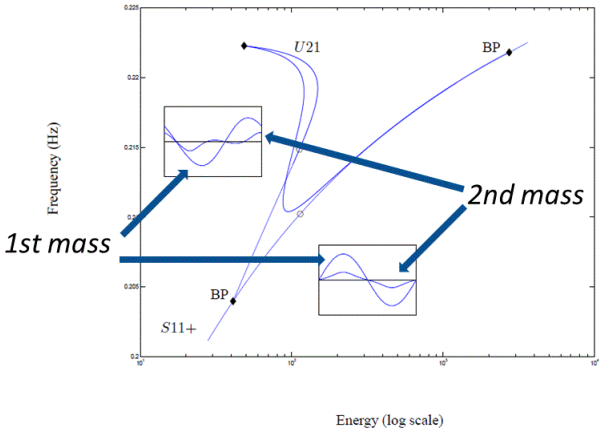


FIGURE 2.17: In both NNMs shown, one of the dofs –the first mass– oscillates with a greater amplitude than the other dof –the second mass–.

2.5.3.2 Internal resonances

When the harmonics of a mode resonate with another NNM those harmonics can sustain an increase in energy and become dominant. This is called 'internal resonance' and leads to the bifurcation of new NNMs branches or modal backbones. Internal resonance can also break the synchronicity of the oscillations of the degrees of freedom if the dominant harmonics are confined to a few system dofs.

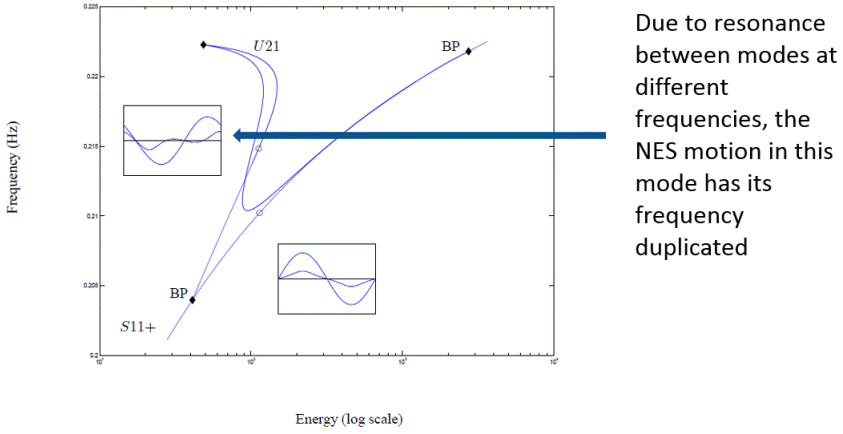


FIGURE 2.18: One of the NNMs whose time series has been displayed on this figure has a dof oscillating with dominant frequency twice the fundamental frequency of the NNM.

2.5.3.3 Ramification of modes

The ramification of the NNMs branches is a result of internal resonances and symmetry-breaking bifurcations. New branches of NNMs –possible periodic solutions of the nonlinear system– could emerge from points on the main NNMs branches. Those particular points can be labeled 'BP' in most bifurcation diagrams, which stands for 'branching point'.

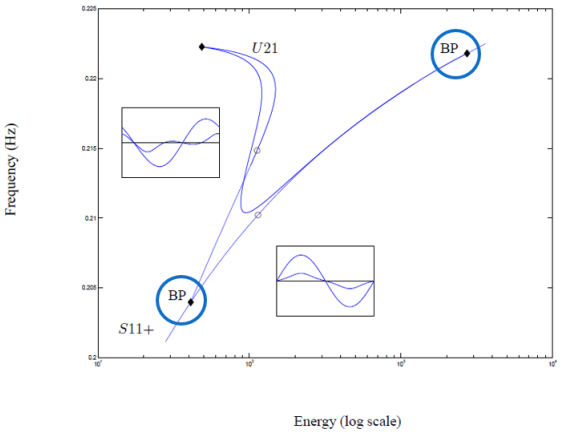


FIGURE 2.19: This diagram shows two bifurcation points from one of the two main NNMs branches. A new modal backbone branches out from one of them and reengages on the main curve on the following point.

All NNMs branches can be labeled with the following notation. It coincides with the convention taken in the vast majority of the bibliography consulted regarding NNMs, including [2], [12] and [13]:

Consider this template of a NNMs branch label: *Smn+*

The first letter represent the symmetry of the modes

- S: symmetrical mode.
- U: unsymmetrical mode.

The numbers that follow the letter, m and n, have this meaning:

- m: number of half waves in the oscillation of the nonlinear attachment of the system between $t = 0$ and $t = \frac{T}{2}$.
- n: number of half waves in the oscillation of the linear part of the system between $t = 0$ and $t = \frac{T}{2}$.

The sign at the end can be a plus sign if the linear and nonlinear parts of the system oscillate in-phase, and a negative sign if they oscillate out-of-phase.

2.5.3.4 Turning points

In Bifurcation Theory, a turning point –also called 'fold'– is a point where a curve changes direction with respect to the curve parameters, i.e. a point where the curve steepness tends to infinity before it turns backwards. Turning points are very important in Bifurcation theory because they emerge for a particular type of bifurcation called 'tangent bifurcation': they give rise to a pair of new solutions for certain values of the parameters. For our particular problem, turning points are those where the curve changes direction with respect to the total energy.

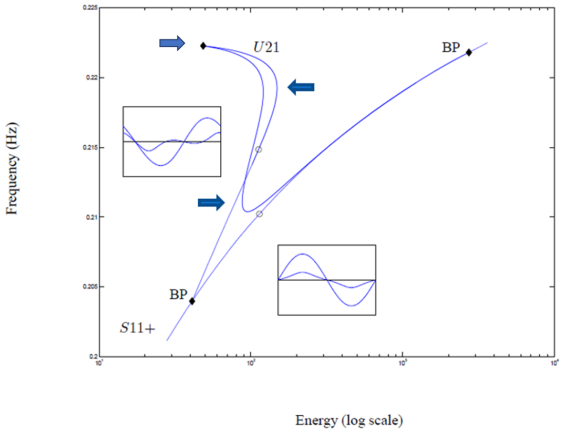


FIGURE 2.20: This figure shows some turning points tagged by blue arrows. On those points the curve reverses course with respect to the total energy in either direction.

Turning points are bifurcation points where stability changes.

2.5.3.5 Symmetry breaking

A nonlinear normal mode is said to be 'symmetrical' if one half of the oscillating motion of a NNM through its period of minimum periodicity 'mirrors' the motion of the other half of it, and 'asymmetrical' in the other way round. Points on the MMNs branches where symmetry changes are called 'symmetry-breaking points'.

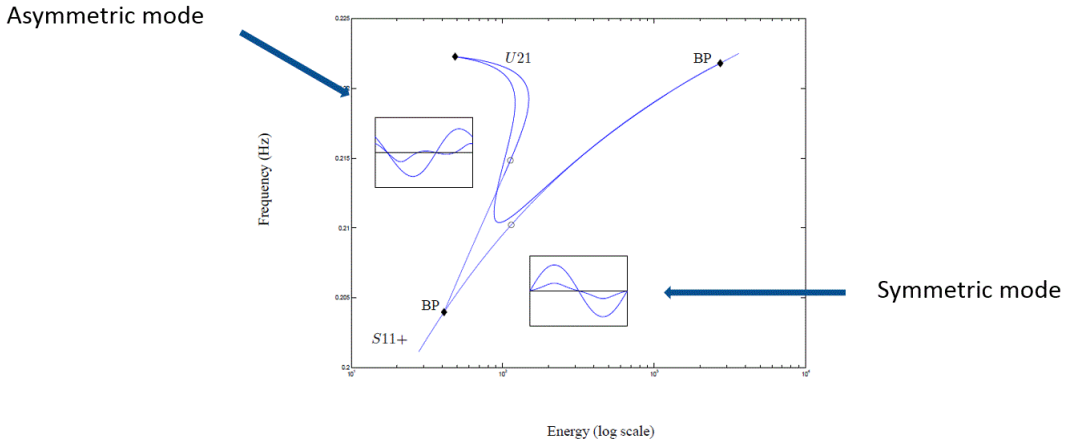


FIGURE 2.21: One of the NNM displayed on the time series plots is unsymmetrical, while the other one shows a symmetrical motion. More precisely, the whole branch of NNMs that sprouts from the main branch is one of unsymmetrical nonlinear normal modes.

A symmetry-breaking point is a bifurcation point, thus involving a stability change.

2.5.3.6 Stability changes

Unlike linear modes, nonlinear normal modes could be stable or unstable. NNMs can be displayed on the FEP on different colours depending on their stability.

Points of change in stability on the modal backbones will always coincide with branching points, turning points and symmetry-breaking points.

Chapter 3

NUMERICAL CONTINUATION ALGORITHM FOR THE COMPUTATION OF NONLINEAR NORMAL MODES

From this chapter, the will divert our attention to numerical algorithm to compute the NNM branches and, consequently, the frequency-energy plots of conservative mechanical systems with nonlinear stiffnesses. This technique is based on 'sequencial continuation', a method that is introduced in the next sections. The reader is presumed a knowledge of basic concepts of differential equations and dynamical autonomous –without an explicit dependence of time– systems.

The most important references considered to write this chapter are conference proceedings [21], [20] and [19], where the shooting algorithm with sequencial continuation is described in great detail.

3.1 Continuation methods: the concept

Briefly defined, 'continuation methods' are a set of numerical algorithms with the potential to compute a sequence of solutions of equations that have a dependence on an independent parameter: $\mathbf{f}(\mathbf{z}, \lambda) = \mathbf{0}$. The independent parameter λ is usualy called 'homotopy variable'. The computation begins with an initial solution $\mathbf{z}_0$ for a particular parameter value λ_0 as the initial value.

Continuation methods originally emerged in the field of Nonlinear Dynamics and Nonlinear Equations in order to compute how the position and stability of the equilibrium points of a dynamical system –or the solutions of an algebraic equation– varied with the value of a particular parameter. These techniques have the potential of approximating the branches of all the possible solutions of a system from a skeleton of sparsed discrete points in one single run of the algorithm. Other by-products from the computation are the points where the stability of the fixed points –if it is applied on a differential equation or SODE– change, or where new branches of solutions emerge. These points are called 'bifurcation points' and the results of the computation are usually displayed on plots called 'bifurcation diagrams'.

Another application is the resolution of nonlinear algebraic equations through 'homotopy': a simpler system of equations with known solution is constructed, and a homotopy parameter is introduced into the equations to connect the equations with known solution to the original system with unknown solution.

There are lots of bibliographic references dedicated to numerical continuation algorithms. Most of them describe it in the context of bifurcation analysis of dynamical systems or for algebraic equation solving with homotopy. A classic on numerical continuation can be found on [35], while [36] is a good introductory text to bifurcation and stability analysis, homotopy and continuation methods.

3.1.1 Convenience of a continuation algorithm

As stated by the Lyapunov theorem in 2.4.1, any conservative system of N degrees of freedom such as 2.3 and 2.5 has at least N different families of periodic orbits. Each of those families is an 'invariant manifold' of trajectories in the phase space as shown on figure 2.11. A NNMs branch is a sequence of points on each of those manifolds where each point is the initial point of a periodic orbit with a certain energy and minimal period T. For the systems referenced above the main NNMs branches, those which emerge from the linear normal modes at very low energies, were approximated with harmonic balance and depicted on their respective FRDs in 2.3.2.

If a point on any of the invariant manifolds is chosen close enough to the equilibrium point –the origin– its energy is very low and the trajectory of the periodic solutions of the nonlinear system virtually coincides with the linear normal modes of the system with no nonlinearities –see figure 3.1–. Therefore, the most intuitive initial points for the computation of a whole branch of NNMs are its linear normal modes with very low energies and the natural frequencies of the linear conservative system:

$$(\mathbf{z_0}, T_0) = (\mathbf{z_N}, T_N = \frac{1}{\nu_N})$$

Now, the aim is to compute, in each i-th iteration of the continuation algorithm, a discrete sequence of periodic points along the invariant manifold that give shape to a branch of NNMs in the frequency-energy plots: $(\mathbf{z_1}, T_1), (\mathbf{z_2}, T_2)$... In the meantime, we mesure the stability of each one of those nonlinear normal modes and identify which phenomenon causes each stability change: a turning point, branching point or a symmetry-breaking point.

Once those NNMs have been obtained, the energy and oscillation frequency ν –where $\nu = \frac{1}{T}$– of each of them determine a point on the frequency-energy plot –FEP–. A whole set of points on the FEP define the frequency-energy dependence along a NNMs branch.

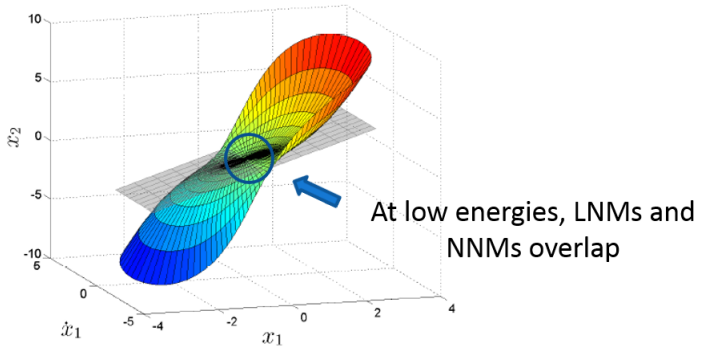


FIGURE 3.1: A LNM –portrayed in grey– and its associated NNM –in color– overlap in a neighborhood of the origin.

3.2 Outputs and inputs of the algorithm

As explained in the previous section, the computation of the n-th NNMs branch requires the n-th linear normal mode and its associated natural frequency in order to set an initial periodic point $(\mathbf{z}_0, T_0)$ on the branch, with $\mathbf{z} = (\mathbf{x}', \mathbf{x})^T$ as the state vector of the dynamical system governing the body –$\mathbf{x}$ as the displacements of the dofs, $\mathbf{x}'$ as their velocities–. LNMs were obtained from the eigenproblem 2.12. A necessary condition is that the mechanical energy of the system in the state $\mathbf{z}_0$ must be small enough to make the system nonlinearities negligible. Under this condition, LNMs and NNMs can be deemed as equal.

Therefore, the linear normal mode $(\mathbf{z_N}, T_N = \frac{1}{\nu_N})$ must be scaled out, multiplying $\mathbf{z_N}$ by a shrinking parameter, before taking it as the initial point of the sequencial continuation of the MMNs branch.

Another input parameters for the algorithm are the so-called 'control parameters', which are presented in 3.5.

The most important numerical output of the computation is the set of points $(\mathbf{z_1}, T_1), (\mathbf{z_2}, T_2)$... along the invariant manifold that conform the corresponding NNMs branch and the minimal periods of each nonlinear normal mode. Each of those NNMs has an associated energy, which must be calculated in order to plot the branches on a FEP. Other outputs are the Floquet multipliers for the quantification of the stability of each NNM –see 3.9–

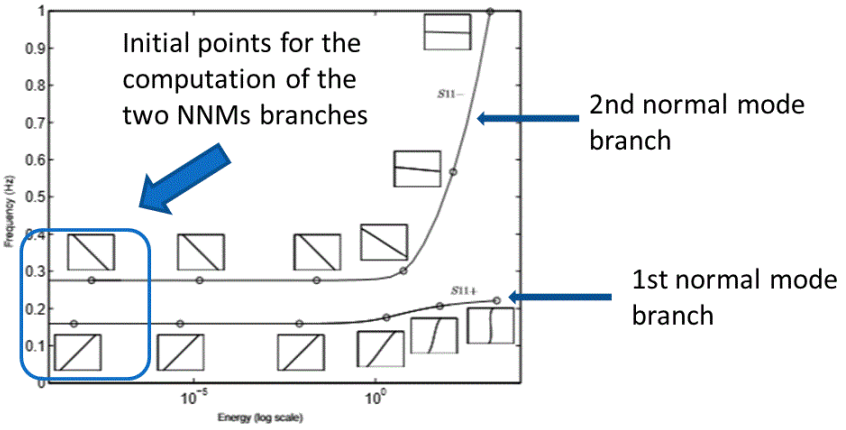


FIGURE 3.2: Yet another exemplifying FEP of a two degrees of freedom system. On this occasion the two main NNMs branches have been outlined, along with the discrete sequences of NNM points computed with continuation that make them up. The modal curve of each NNM and the initial points of computation –points of lower energy whose trajectories coincide with the LNMs of the conservative linear system– are also noted.

and a set of parameters quantifying the computing performance and accuracy of the implementation in each step, also presented in this chapter.

The graphic outputs were already described in 2.5.2 and sre depicted again on 3.2

3.3 Definition of a shooting function to determine periodic orbits

The aim is to find the periodic orbits of a conservative system of the form:

$$[\mathbf{M}]\mathbf{x}'' + [\mathbf{K}]\mathbf{x} + \mathbf{f_{nl}}(\mathbf{x}) = \mathbf{0} \tag{3.1}$$

First, the SODE 3.1 is expressed as a first order system by defining a new phase space for the new first order system as $\mathbf{z} = (\mathbf{x}', \mathbf{x})^T$. $\mathbf{x}$ is the vector of displacement functions for each dof, and $\mathbf{x}'$ its time derivative, resulting in:

$$\mathbf{z}' = \mathbf{g}(\mathbf{z}) = \begin{pmatrix} \mathbf{x}' \\ -[\mathbf{M}]^{-1}[\mathbf{K}\mathbf{x} + \mathbf{f_{nl}}(\mathbf{x})] \end{pmatrix} \tag{3.2}$$

This is a system with $2N$ unknowns: the displacement of all the dofs and their velocities.

Given an initial point $\mathbf{z_p}$ in the phase space, a unique function $\mathbf{z}(t)$, solution of the system 3.2, with $\mathbf{z_p} = \mathbf{z}(0)$, exists, and at any instant T of time the system is in the space state $\mathbf{z}(\mathbf{z_p}, T)$.

If $\mathbf{z}(t)$ is periodic of period T then $\mathbf{z}(0) = \mathbf{z}(T)$, and $\mathbf{z_p} = \mathbf{z}(0)$ is a point in the phase space of 3.2 periodic with period T. With this in mind, a 'shooting function' can be defined as the difference between an initial state $\mathbf{z_p}$ and the system response –space state– after a period T determined by the initial value problem with equation 3.2 and initial point $\mathbf{z_p}$:

$$\begin{aligned} \mathbf{S}(\mathbf{z_p}, T) &\equiv \mathbf{z}(\mathbf{z_p}, T) - \mathbf{z_p} \\ \mathbf{S} &: \mathbb{R}^{N+1} \rightarrow \mathbb{R}^N \end{aligned} \tag{3.3}$$

If $\mathbf{z_p}$ is a periodic point with period T, the shooting function of the duple $(\mathbf{z_p}, T)$ is zero.

$$\mathbf{S}(\mathbf{z_p}, T) = \mathbf{0} \tag{3.4}$$

It is obvious that if $\mathbf{z}(t)$ is a solution of 3.2 with period T, any function $\mathbf{z}(t+\Delta t)$ –the same solution defined in a translated time domain– is also periodic of period T. This arbitrariness of the initial condition must be avoided forcing the shooting function to determine only a one-dimensional curve of points on the invariant manifold of periodic orbits. This curve constitutes the NNMs branch to be computed in each i-th invariant manifold of 3.1. To do so a 'phase condition' $h(\mathbf{z_p})$ is included as a constraint:

$$\begin{aligned} h(\mathbf{z_p}) &= 0 \\ h &: \mathbb{R}^N \rightarrow \mathbb{R} \end{aligned} \tag{3.5}$$

With this constraint the 'augmented' shooting function $\mathbf{F}(\mathbf{z_p}, T)$ is defined as:

$$\mathbf{F}(\mathbf{z_p}, T) = \begin{pmatrix} \mathbf{S}(\mathbf{z_p}, T) \\ h(\mathbf{z_p}) \end{pmatrix} = \mathbf{0} \tag{3.6}$$

The most commonly chosen phase condition is to set to zero the velocities of any i-th degree of freedom or several of them. This can be done taking the norm of a vector containing the velocities of those degrees of freedom as the phase function:

$$h : \|\mathbf{x}'_\mathbf{p}\| = 0 \tag{3.7}$$

An evaluation of the shooting function $\mathbf{S}$ in a point $\mathbf{z_p}$ of the phase space of 3.1 requires the computation of the trajectory in a timespan T from that initial point to obtain the final state $\mathbf{z}(\mathbf{z_p}, T)$. This can be done with numerical methods for resolution of ODEs and SODEs. Some of the most popular algorithms are Runge-Kutta and Newmark.

3.4 Description of the algorithm

Any continuation algorithm consists in an iterative process where a sequence of points $(\mathbf{z_1}, \lambda_1), (\mathbf{z_2}, \lambda_2)...$, solutions of a system $\mathbf{f}(\mathbf{x}, \lambda) = \mathbf{0}$ with homotopy parameter λ. In each i-th iteration the solution $(\mathbf{z_{i+1}}, \lambda_{i+1})$ is calculated from the previous solution: $(\mathbf{z_i}, \lambda_i)$. The computation of any sequence of points requires an initial point $(\mathbf{z_0}, \lambda_0)$, solution of $\mathbf{f} = \mathbf{0}$, as an input.

In this particular case, the function is the augmented shooting function 3.6 and the homotopy parameter is the orbit minimal period T.

Every iteration from point i to point $i + 1$ comprises two steps: a predictor step and a corrector. Those steps are described below.

3.4.1 Predictor step

The input point of the i-th iteration is $(\mathbf{z}_i, T_i)$. The aim of the predictor is to estimate the following point in the curve:

$$(\mathbf{z}_i, T_i) \rightarrow (\bar{\mathbf{z}}_{i+1}, \bar{T}_{i+1})$$

where $(\bar{\mathbf{z}}_{i+1}, \bar{T}_{i+1})$ is the estimated next solution of $\mathbf{F}$.

This is done by linearizing the function $\mathbf{F}(\mathbf{z}, T)$ around the i-th point in the solution curve: $(\mathbf{z}_i, T_i)$. The new linear function at $(\mathbf{z}_i, T_i)$ has a jacobian matrix $\mathbf{F}_{\mathbf{z},T}(\mathbf{z}_i, T_i)$, and any new solution of the linearized function can be found in the direction of a vector $\mathbf{v}_i \in \mathbb{R}^{N+1}$ that is tangent to the curve $\mathbf{F}(\mathbf{z}, T) = \mathbf{0}$ in $(\mathbf{z}_i, T_i)$. Therefore $\mathbf{v}_i$ is obtained from the homogeneous system of linear equations:

$$\mathbf{F}_{\mathbf{z},T}(\mathbf{z}_i, T_i)\mathbf{v}_i = \mathbf{0} \tag{3.8}$$

To make notation simpler, $\mathbf{w}_i$ will represent the duple $(\mathbf{z}_i, T_i)$.

The length of the tangent vector $\mathbf{v}_i$ solution of 3.4.1 is undetermined, and must be explicitly fixed into 3.4.1. The most common practise, followed by convention in this algorithm, is to set the norm of $\mathbf{v}_i$ to the unit:

$$\|\mathbf{v}_i\| = 1 \tag{3.9}$$

The new jacobian matrix –3.17– that results from this unitary condition is square, and if it is a regular matrix, the system is determined with unique solution $\mathbf{v}_i$.

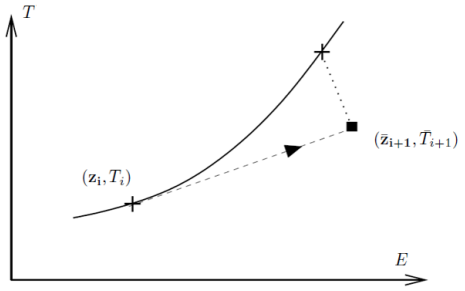


FIGURE 3.3: Schematic representation of the predictor phase of a continuation algorithm.

With the tangent vector $\mathbf{v}_i$ of the solution curve of $\mathbf{F}(\mathbf{z}, T)$ at $(\mathbf{z}_i, T_i)$, a new point in the curve can be 'predicted', i.e. approximated, as $(\bar{\mathbf{z}}_{i+1}, \bar{T}_{i+1})$, as figure 3.3 reflects. All what is left to do is to multiply the tangent vector by a 'stepsize parameter' h, which is a control parameter of the algorithm –see 3.5.1–:

$$\bar{\mathbf{w}}_{i+1} = \mathbf{w}_i + h\mathbf{v}_i \tag{3.10}$$

3.4.2 Corrector step

The predicted solution must be 'corrected' to deliver a final point which is close enough to the solution curve of $\mathbf{F} = \mathbf{0}$:

$$(\bar{\mathbf{z}}_{i+1}, \bar{T}_{i+1}) \rightarrow (\mathbf{z}_{i+1}, T_{i+1})$$

To do so, the Newton-Raphson method can be applied to $\mathbf{F}(\mathbf{z}, T)$ with $(\bar{\mathbf{z}}_{i+1}, \bar{T}_{i+1})$ as the initial point. $(\mathbf{z}_{i+1}^{k+1}, T_{i+1}^{k+1})$ is the result of the k-th iteration of the Newton-Raphson method, which should converge over the solution curve of $\mathbf{F} = \mathbf{0}$.

If the method is convergent, for some k-th iteration, the point $(\mathbf{z}_{i+1}^{k+1}, T_{i+1}^{k+1})$ is a solution of $\mathbf{F} = \mathbf{0}$ good enough and therefore the next point in the curve: $(\mathbf{z}_{i+1}, T_{i+1})$.

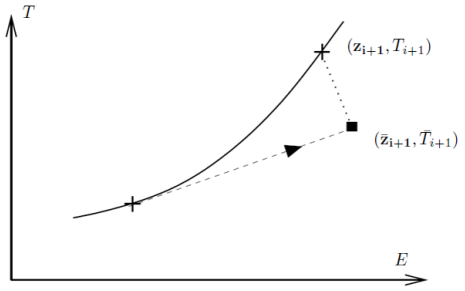


FIGURE 3.4: During the corrector step, the predictor solution is corrected until an acceptable error tolerance is satisfied, that is, until the (i+1)-th point computed during this continuation step is close enough to the curve of exact solutions of $\mathbf{F} = \mathbf{0}$.

However, the function $\mathbf{F}$ has not a unique solution. Instead, the solution is dependent on one parameter and the resulting set of solutions constitutes one or multiple curves. Hence this Newton-Raphson scheme is undetermined: for any initial point $(\bar{\mathbf{z}}_{i+1}, \bar{T}_{i+1})$ there are multiple directions of convergence in the space $(\mathbf{z}, T)$.

A unique direction of convergence can be established with an extra condiction $o(\mathbf{w}_{i+1}^k)$ that makes the system $\mathbf{F} = \mathbf{0}$ determined.

$$\begin{aligned} o(\mathbf{w}_{i+1}^k) &= 0 \\ o &: \mathbb{R}^{N+1} \to \mathbb{R} \end{aligned} \tag{3.11}$$

If pseudo-arclength continuation is chosen as continuation method, the convention is to converge to the real solution in an direction orthogonal to that of the predictor step, given by the tangent vector $\mathbf{v}_i$:

$$o(\mathbf{w}_{i+1}^k) = \langle \mathbf{w}_{i+1} - \mathbf{w}_{i+1}^k, \mathbf{v}_i \rangle \tag{3.12}$$

where $\langle u, v \rangle$ represents the dot product of vectors u and v.

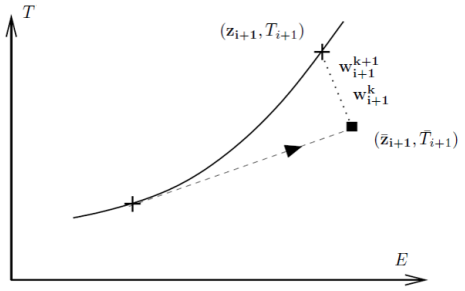


FIGURE 3.5: In each k-th Newton-Raphson iteration a (k+1)-th solution $\mathbf{w}$ is obtained, and if the method is convergent this new solution is closer to the solution curve of $\mathbf{F} = \mathbf{0}$.

The augmented function on which the corrector step is applied in pseudo-arclength continuation is:

$$\mathbf{H}(\mathbf{w}_{i+1}^{k}) = \begin{pmatrix} \mathbf{F}(\mathbf{w}_{i+1}^{k}) \\ o(\mathbf{w}_{i+1}^{k}) \end{pmatrix} \tag{3.13}$$

and the Newton-Raphson formula is, for the k-th corrector step:

$$\mathbf{w}_{i+1}^{k+1} = \mathbf{w}_{i+1}^{k} - \mathbf{H}_{\mathbf{w}}^{-1}(\mathbf{w}_{i+1}^{k})\mathbf{H}(\mathbf{w}_{i+1}^{k}) \tag{3.14}$$

where $\mathbf{H}_{\mathbf{w}}^{-1}(\mathbf{w}_{i+1}^{k})$ is the inverse of the jacobian matrix of $\mathbf{H}(\mathbf{w}_{i+1}^{k})$, $\mathbf{H}_{\mathbf{w}}(\mathbf{w}_{i+1}^{k})$. This matrix, shown in 3.18, is not square, therefore its inverse is in reallity a pseudo-inverse.

3.5 Control parameters

3.5.1 Predictor stepsize control

The stepsize parameter h is recalculated in each predictor step to conform to a desired number of Newton-Raphson iterations –3.14– in the corrector step. The longer the stepsize of the predictor, the more iterations needed in the corrector phase to converge to the solution curve of $\mathbf{F}$.

There are as many methods and conventions of stepsize control as computational implementations of the continuation algorithm. For the particular application of the algorithm described in this text, the stepsize control method implemented, described in further detail in [20] and [19], is defined here:

Let N_{opt} be an ideal number of iterations, input parameter of the whole continuation algorithm, and let h_{i-1} and N_{i-1} be the stepsize and number of corrector iterations, respectively, taken during the former, (i-1)-th, continuation iteration. A stepsize h_i for the present i-th continuation stage can be established as:

$$h_i = (\frac{N_{opt}}{N_{i-1}})h_{i-1} \tag{3.15}$$

At the same time, upper and lower thresholds, h_{max} and h_{min}, are applied on the stepsize as input continuation parameters, together with an upper limit to the number of Newton-Raphson corrector iterations, N_{max}, and an initial stepsize value that is applied in step one of the continuation: $h_{initial}$.

If the corrector step does not converge to a solution on the curve or if it required more iterations than the upper limit set, the stepsize is halved and the corrector step is repeated from a different point until the convergence condition is reached in an adecuate number of corrector step iterations.

3.5.2 Factor of convergence (corrector stopping condition)

Any implementation of a Newton-Raphson algorithm requires one or several stopping conditions. The iterative process of finding a zero in a function should be stopped when the value of the zero obtained numerically is sufficiently close to one of the solutions of the system.

This is attained by defining a 'factor of convergence'. For our particular application, the corrector phase stops when the value of the following expression:

$$\frac{\|\mathbf{S}(\mathbf{z}_{i+1}^k, T_{i+1}^k)\|}{\|\mathbf{z}_{i+1}^k\|} = \frac{\|\mathbf{z}(\mathbf{z}_{i+1}^k, T_{i+1}^k) - \mathbf{z}_{i+1}^k\|}{\|\mathbf{z}_{i+1}^k\|}$$

is smaller than a predefined factor of convergence ϵ –another control parameter of the continuation implementation–.

If there was no convergence for any value of the stepsize greater than the minimum stepsize the computation must be stopped and an error message should be shown.

3.6 Detection of turning points

The sign of the stepsize h must be corrected in each predictor step in order to prevent any reversal in the computation of the solution curve of $\mathbf{F} = \mathbf{0}$ in case a turning point has been passed during the i-th iteration of the continuation scheme.

For that purpose, the sign of the current stepsize h_i must be chosen so the following expression has a positive value:

$$[h_i \mathbf{v}_i]^T [h_{i-1} \mathbf{v}_{i-1}]$$

where $\mathbf{v}_i$ and $\mathbf{v}_{i-1}$ are the tangent vectors, and h_i and h_{i-1} the stepsize values, of the current and the last continuation stages respectively.

It the stepsize parameter has its sign changed during an i-th continuation step, there is a turning point in the NNMs branch –the solution curve of $\mathbf{F} = \mathbf{0}$– somewhere between $(\mathbf{z}_i, T_i)$ and $(\mathbf{z}_{i+1}, T_{i+1})$. The estimation of the exact position of the turning point is not of special interest for us.

3.7 Halved period shooting function for symmetrical modes

All symmetrical NNMs branches –remember the concept of 'symmetrical' and 'unsymmetrical' mode in 2.5.3.5– conform to the following condition:

$$\mathbf{z}(\mathbf{z_p}, \frac{T}{2}) + \mathbf{z_p} = 0$$

hence we can calculate the shooting function with half the period of those orbits, reducing the time of computation of the trajectory from the initial point $\mathbf{z_p}$ by a factor close to two:

$$\mathbf{S}(\mathbf{z_p}, T) \equiv \mathbf{z}(\mathbf{z_p}, \frac{T}{2}) - \mathbf{z_p} \tag{3.16}$$

3.8 Jacobian matrix computation

The jacobian matrix of the augmented shooting function $\mathbf{F}(\mathbf{z}_i, T_i)$ for the i-th predictor step is given by:

$$\mathbf{F}_{\mathbf{z},T}(\mathbf{z}_i, T_i) = \begin{pmatrix} \frac{\partial \mathbf{S}}{\partial \mathbf{z}_i}(\mathbf{z}_i, T_i), & \frac{\partial \mathbf{S}}{\partial T}(\mathbf{z}_i, T_i) \\ \frac{\partial h}{\partial \mathbf{z}_i}(\mathbf{z}_i), & 0 \end{pmatrix} \tag{3.17}$$

and is a square matrix with $N+1$ rows and columns.

The augmented function $\mathbf{H}(\mathbf{z}_i, T_i)$ taken during the k-th corrector iteration of the i-th continuation step has this expression for its jacobian:

$$\mathbf{H}_{\mathbf{z},T}(\mathbf{z}_i^k, T_i^k) = \begin{pmatrix} \frac{\partial \mathbf{S}}{\partial \mathbf{z}_i}(\mathbf{z}_i^k, T_i^k), & \frac{\partial \mathbf{S}}{\partial T}(\mathbf{z}_i^k, T_i^k) \\ \frac{\partial h}{\partial \mathbf{z}_i}(\mathbf{z}_i^k), & 0 \\ \multicolumn{2}{c}{\mathbf{v}_i^T} \end{pmatrix} \tag{3.18}$$

With its $(N+1)$ columns and $(N+2)$ rows, its inverse requires the Moose-Penrose pseudo-inverse.

These matrices can be computed with two different methods: finite differences and sensitivity analysis.

3.8.1 Finite differences method

The time derivatives of $\mathbf{S}(\mathbf{z}_i, T_i)$:

$$\frac{\partial \mathbf{S}}{\partial T}(\mathbf{z}_i, T_i)$$

where the function $\mathbf{S}(\mathbf{z}_i, T_i)$ is given by definition 3.3, can be calculated straightforwardly from the first time derivatives that conform the main SODE of the model. Those time derivatives must be evaluated in the space state given by the trajectory with initial position $\mathbf{z}_i$ after a determined period of time T_i:

$$\frac{\partial \mathbf{S}}{\partial T}(\mathbf{z}_i, T_i) = \frac{\partial \mathbf{z}(\mathbf{z}_i, T_i)}{\partial t} = \mathbf{z}'(\mathbf{z}_i, T_i) = \mathbf{g}(\mathbf{z}(\mathbf{z}_i, T_i)) \tag{3.19}$$

where, with the notation of the previous sections, $\mathbf{z}(\mathbf{z}_i, T_i)$ is the system state in a trajectory given by $\mathbf{z}_i$ as an initial condition, after a timespan T_i; and $\mathbf{z}'(\mathbf{z}_i, T_i)$ is the time derivative of the system in that state –how that space state varies in time–.

Meanwhile, its space derivatives are calculated with a finite difference scheme, computing perturbed trayectories with respect to each j-th space state variable z_j of the initial position $\mathbf{z}_i$ of the trajectory concerned, $z_{i,j}$, to calculate numerically each j-th first derivative. the final submatrix $\frac{\partial \mathbf{S}}{\partial \mathbf{z}_i}(\mathbf{z}_i, T_i)$ of the jacobian matrix calculated with that method has the form:

$$\frac{\partial \mathbf{S}}{\partial \mathbf{z}_i}(\mathbf{z}_i, T_i) = \frac{\partial \mathbf{z}(\mathbf{z}_i, T_i)}{\partial \mathbf{z}_i} - \mathbf{I} \tag{3.20}$$

However, this apprach may be very slow for large systems because it takes at least two numerical integrations of the system trajectory to compute each column of the jacobian matrix.

3.8.2 Sensitivity analysis

An alternative arises when we reflect on this property of the space derivatives of the time derivatives in the phase space given by the SODE of the model:

$$\frac{\partial}{\partial \mathbf{z_p}}[\mathbf{z}'(\mathbf{z_p}, T)] = \frac{\partial}{\partial \mathbf{z_p}}[\mathbf{g}(\mathbf{z}(\mathbf{z_p}, T))] \tag{3.21}$$

Applying the chain rule to the previous expression we obtain the following SODE:

$$\begin{aligned} \frac{d}{dt}[\frac{\partial \mathbf{z}(\mathbf{z}_i, t)}{\partial \mathbf{z}_i}] &= \frac{\partial \mathbf{g}(\mathbf{z})}{\partial \mathbf{z}}|_{\mathbf{z}(\mathbf{z}_i,t)}[\frac{\partial \mathbf{z}(\mathbf{z}_i, t)}{\partial \mathbf{z}_i}] \\ \frac{\partial \mathbf{S}}{\partial \mathbf{z}_i}(\mathbf{z}_i, T_i) &= \frac{\partial \mathbf{z}(\mathbf{z}_i, T_i)}{\partial \mathbf{z}_i} - \mathbf{I} \end{aligned} \tag{3.22}$$

where $\frac{\partial \mathbf{g}(\mathbf{z})}{\partial \mathbf{z}}|_{\mathbf{z}(\mathbf{z}_i,t)}$ is the space derivative of the function $\mathbf{g}(\mathbf{z})$ defining the SODE of the model, evaluated in the space state of the system in a trajectory with initial point $\mathbf{z}_i$ after a time t –thus meaning that 3.22 is a non-autonomous dynamical system with this term varying with time t–.

From 3.22 we can compute the space derivatives of $\mathbf{S}(\mathbf{z}_i, T_i)$ with respect to the initial vector in the space state, $\mathbf{z}_i$, with just one time integration of the term $\frac{\partial \mathbf{z}(\mathbf{z}_i,t)}{\partial \mathbf{z}_i}$:

As an initial condition for that system we have:

$$\frac{\partial \mathbf{z}(\mathbf{z}_i, 0)}{\partial \mathbf{z}_i} = \mathbf{I} \tag{3.23}$$

Integrating 3.22 with initial conditions 3.23 in the time interval $[0, T_i]$ the submatrix $\frac{\partial \mathbf{S}}{\partial \mathbf{z}_i}(\mathbf{z}_i, T_i)$ of jacobians 3.17 and 3.18 can be computed.

3.9 Calculation of the mode stability: the Floquet multipliers

After the computation of a whole NNMs branch of points $(\mathbf{z}_i, T_i)$, the stability of each NNM is calculated with the following procedure:

Perturbing the initial conditions of a periodic orbit of period T_i –$(\mathbf{z}_i, T_i)$– by $\Delta\mathbf{z}_i$ we get a 'perturbation vector' $\Delta\mathbf{z}(\mathbf{z}_i, \Delta\mathbf{z}_i, T_i)$ of the trajectory after the period T_i:

$$\Delta\mathbf{z}(\mathbf{z}_i, \Delta\mathbf{z}_i, T_i) = \mathbf{z}(\mathbf{z}_i + \Delta\mathbf{z}_i, T_i) - \mathbf{z}(\mathbf{z}_i, T_i) \tag{3.24}$$

Now, a 'monodromy matrix' of derivatives of the space state $\mathbf{z}(\mathbf{z}_i, T_i)$ after the period T_i with respect to the initial condition vector –$\mathbf{z}_i$– is defined:

$$\Phi_T(\mathbf{z}_i, T_i) = \frac{\partial\mathbf{z}(\mathbf{z}_i, t)}{\partial\mathbf{z}_i}|_{t=T_i} \tag{3.25}$$

The monodromy matrix of the i-th point on the NNMs branch can be obtained as a subproduct of the jacobian submatrix $\frac{\partial\mathbf{S}}{\partial\mathbf{z}_i}(\mathbf{z}_i, T_i)$ by applying the expression 3.22:

The perturbation vector of the trajectory can be expressed in Taylor series as a function of the monodromy matrix:

$$\Delta\mathbf{z}(\mathbf{z}_i, \Delta\mathbf{z}_i, T_i) = \Phi_T(\mathbf{z}_i, T_i)\Delta\mathbf{z}_i + \mathcal{O}(\|\Delta\mathbf{z}_i\|^2) \tag{3.26}$$

If we take m times a given period T_i the perturbation vector may converge to zero –stable mode– or diverge –unstable mode–:

$$\Delta\mathbf{z}(\mathbf{z}_i, \Delta\mathbf{z}_i, mT_i) = [\Phi_T(\mathbf{z}_i, T_i)]^m \Delta\mathbf{z}_i + \mathcal{O}(\|\Delta\mathbf{z}_i\|^2) \tag{3.27}$$

Therefore, the eigenvalues of the monodromy matrix, called 'Floquet multipliers', quantify the ratio of convergence or divergence of a system oscillation mode after a perturbation:

- All eigenvalues with real parts between -1 and 1: stable NNM mode.
- An eigenvalue has a real part with absolute value greater than one: unstable NNM mode.

A point of stability change in a NNMs branch coincides with a point of bifurcation: branching point, turning point or symmetry-breaking bifurcation.

3.10 Branch switching

Once the branching points have been located by means of stability change different strategies can be taken to compute the NNMs branches that could emerge from them. The tangent vector of the new branch can be calculated with the hessian matrix of $\mathbf{F}(\mathbf{z}_i, T_i)$ evaluated in the branching point, and then, a perturbation technique can be applied to the system $\mathbf{F}(\mathbf{z}_i, T_i)$ to force the continuation of new branches –see [35] and [36]–.

Unfortunately, no formal branching strategy has been implemented in the algorithms used in this Project, as no specific method has been found in the bibliography. The alternative is to find a combination of continuation parameters that forces the curve computation to divert and follow a different path than the usual when a branching point has been passed.

The implementation of a formal branching technique is a pending task for future developments of the method.

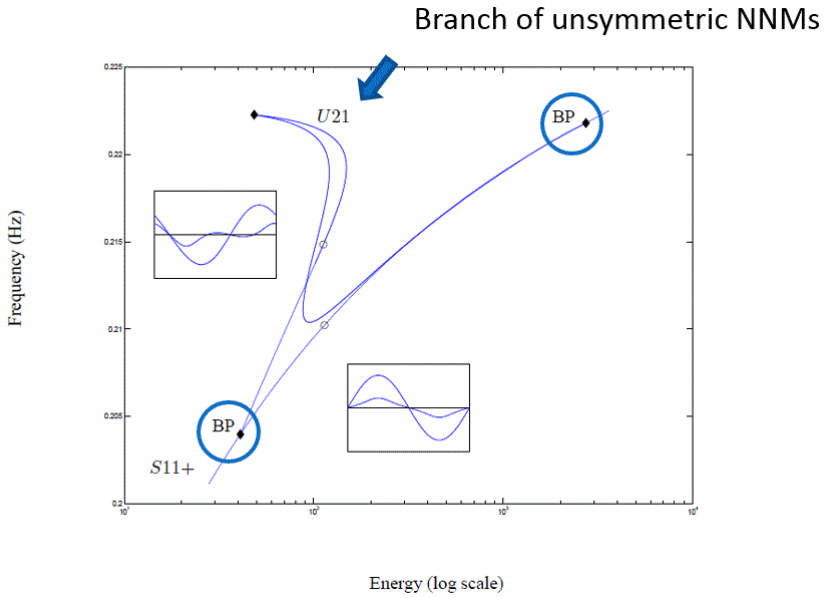


FIGURE 3.6: In this FEP, two branching points have been found by means of stability analysis. An appropiate branching technique should allow to compute the NNMs branches bifurcating from the main NNMs branch. In the case shown, the bifurcating branch contains unsymmetrical NNMs –see 2.5.3.5–

3.11 Mode energy calculation

It is required to calculate the energy of each i-th point computed on the desired NNMs branch, $(\mathbf{z}_i, T_i)$, for their symbolic representation on a FEP plot.

The total energy of the conservative system modeled by equation 3.1 at any state $\mathbf{z} = (\mathbf{x}', \mathbf{x})^T$ is:

$$E = T + L_L + L_{NL} \tag{3.28}$$

with:

$$Kinetic\ energy\ T = \frac{1}{2}\mathbf{x}'^T[\mathbf{M}]\mathbf{x}' \tag{3.29}$$

$$Potential\ energy\ given\ by\ the\ linear\ stiffness\ L_L = \frac{1}{2}\mathbf{x}^T[\mathbf{K}]\mathbf{x} \tag{3.30}$$

$$Potential\ energy\ given\ by\ the\ nonlinear\ force\ component\ L_{NL} = \oint_{\mathbf{0}}^{\mathbf{x}} \mathbf{f_{nl}}(\mathbf{x}) \cdot d\mathbf{x} \tag{3.31}$$

Due to the conservativeness of undamped mechanical systems and their elastic force fields the value of L_{NL} is independent of the path taken for the calculation of the line integral defining it, and $\mathbf{f_{nl}}(\mathbf{x}) = \nabla \cdot L_{NL} = (\frac{\partial L_{NL}}{\partial x_1}, \frac{\partial L_{NL}}{\partial x_2}, ..., \frac{\partial L_{NL}}{\partial x_N})^T$.

The last property makes it possible to calculate L_{NL} analitically from the nonlinear elastic force $\mathbf{f_{nl}}(\mathbf{x})$ in case this function is continuous in the domain of $\mathbf{x}$. That can be done following these steps:

1. We calculate L_{NL} as the integral of any of the terms –the i-th term– of $\mathbf{f_{nl}}(\mathbf{x})$ with respect to the i-th degree of freedom:

$$L_{NL} = \int f_{nl,i}(\mathbf{x})\, dx_i = F_{nl,i}(\mathbf{x}) + g(x_1, x_2, .., x_{i-1}, x_{i+1}, ..., x_N) \tag{3.32}$$

   where $F_{nl,i}$ is the primitive of $f_{nl,i}$ with respect to the i-th degree of freedom.

2. We find a scalar function $g(x_1, x_2, .., x_{i-1}, x_{i+1}, ..., x_N)$ satisfying the condition:

$$\frac{\partial L_{NL}}{\partial x_j} = \frac{\partial F_{nl,i}}{\partial x_j} + \frac{\partial g}{\partial x_j} = f_{nl,j} \tag{3.33}$$

   for all $j = 1, 2, ..., N; j \neq i$.

Chapter 4

THE MMNCONT MATLAB PACKAGE

In this new chapter, I present a Matlab package called 'NNMcont', developed by researchers from the Structural Dynamics Research Group in the Aerospace and Mechanical Engineering Department of the University of Liège. It was developed with the purpose to implement computationally the algorithm described in the previous chapter in a interface of the well-known numerical environment Matlab.

The package –version 1.1– can be downloaded from this webpage: http://www.ltas-vis.ulg.ac.be/cmsms/index.php?page=nnm. Throughout this chapter, this version of NNMcont is described with its features and capabilities. The aim is to provide a simple and comprehensive guide about the usage of this package, while applying it to compute the main NNMs branches of the two degrees of freedom nonlinear conservative oscillator represented in 2.5 and modeled by the SODE 2.5. The computational results from this analysis are shown in chapter 5. At the same time, possible shortcomings or limitations that I could encounter are noted. Unfortunately, NNMcont source code has been compiled into Matlab protected p-files, thus it is not possible to access it. An open source code may have been useful for a better understanding of the algorithm.

For the execution of NNMcont I have used the Matlab 2015a release in the Windows 10 operating system with a 64-bit Intel processor. The package was compiled with a Matlab version prior to 7.5 –R2007b– and executed by myself on a Matlab version as old as 6.0. NNMcont should then be executable in a wide range of Matlab versions, but it may not be executable in future versions unless its p-codes are compiled with a more recent version than 7.5.

The reader and potential user of this package should have at least a basic knowledge on the Matlab environment and on the Matlab programming language, as well as object-oriented programming in Matlab.

4.1 About NNMcont

The NNMcont package performs the computation of the main NNMs branches of a conservative system by numerical continuation, taking the shooting algorithm and input parameters described in the previous chapter, and returning the same outputs described, both numerical and graphical, inluding stability analysis by means of Floquet multipliers.

The computed branches are displayed on a FEP in the main NNMcont graphic user interface. By clicking on any of the discrete points that conform a branch one can compute the modal curves and time series of that particular nonlinear normal mode.

The mass and stiffness matrices and the nonlinear elements of the model considered are introduced and loaded from m-files. Nonlinear elements are defined by objects of the Matlab classes available in the package and conform to a particular "nonlinear law' object. New nonlinear element or law classes can be created from scratch if necessary. All numerical results are saved and loaded from Matlab data files.

For a preliminar introduction on NNMcont the package comes with the tutorial [26] in the directory 'Tutorial'.

4.2 Installation and execution

By clicking the download link of the package –https://www.ltas-vis.ulg.ac.be/cmsms/uploads/File/NNMcont.zip– a zip file is saved in your computer. The next step is to unzip it in any memory location. The package files and directories are contained in a folder called 'NNMcont'.

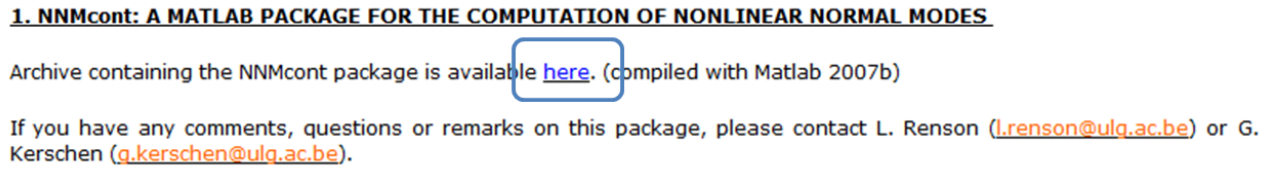
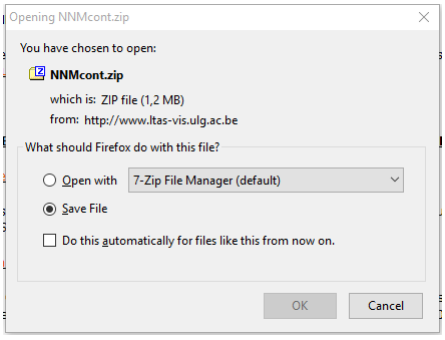
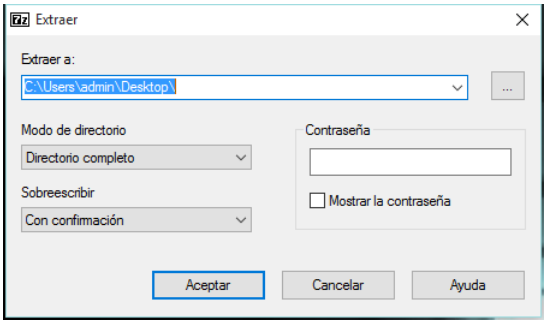


FIGURE 4.1: Download link in the webpage http://www.ltas-vis.ulg.ac.be/cmsms/index.php?page=nnm.



(A) Saving the zip file.



(B) Unzipping it in the desktop.

FIGURE 4.2: NNMcont Installation.

Next, we execute our version of Matlab and add all folders and subfolders to the Matlab search path by clicking the 'set path' button on the home tab.

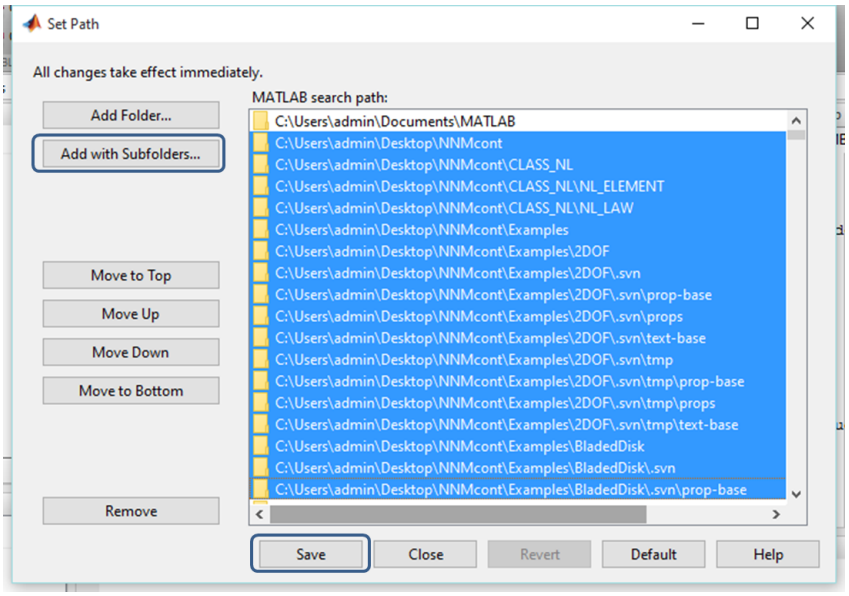


FIGURE 4.3: Adding the 'NNMcont' folder to the path, with subfolders.

Then, the current folder must be changed to the folder containing the m-files that define our mechanical model: *defsys.m* and *paramdisp.m*.

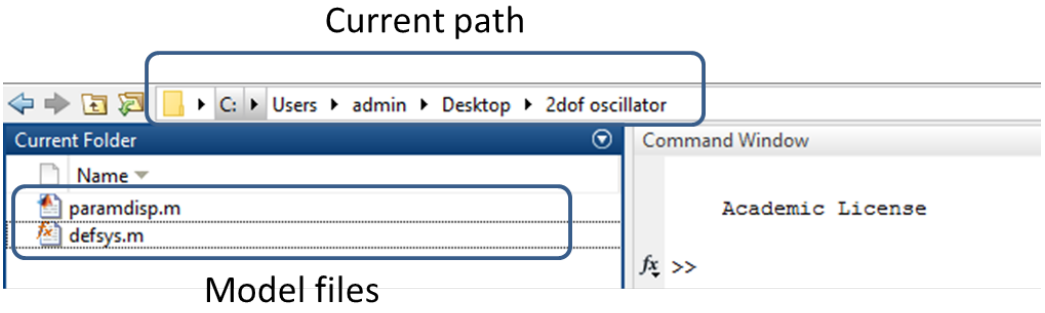


FIGURE 4.4: The current path is set to the directory that containes the two model files. the folder '2dof oscillator' contains the model for our two degrees of freedom oscillator 2.5.

The environment is now prepared to run the program. We simply have to type the command 'NNMcont' and its graphic user interface –GUI– is launched:

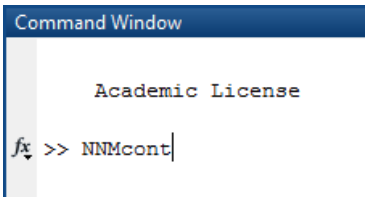


FIGURE 4.5: 'NNMcont' command.

4.3 The NNMcont graphic user interface –GUI–

The MMNcont GUI is comprised of different panels of text fields, buttons, radio buttons, drop-down lists and a plotting window as depicted on figure 4.6:

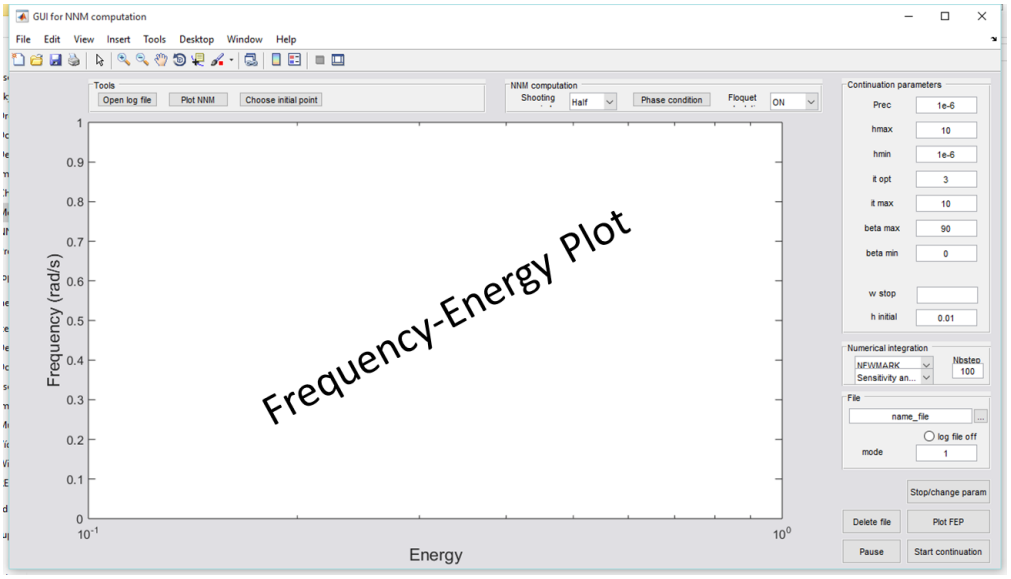


FIGURE 4.6: Graphic user interface of NNMcont with the graph display for the FEP.

Lists and fields mainly allow for the introduction of the inputs and outputs of the program and the algorithm. Inputs and outputs are described in subsection 4.3.1. Most buttons are for computation control -*Start continuation*, *Pause*, *Stop/change parameter* and *Chose initial point* buttons–. while others allow the user to choose a desired graphical output to compute –*Plot FEP* and *Plot NNM*– or to open a 'log file' with the record of the sequencial continuation events during each step of the algorithm –*Open log file*–. The *Phase condition* button opens a window where the user can establish on which degrees of freedom to force the null phase condition –More on this in the next section–. The

main graph display on the GUI serves as a graphical output of the NNMs branches on a frequency-energy plot –FEP–.

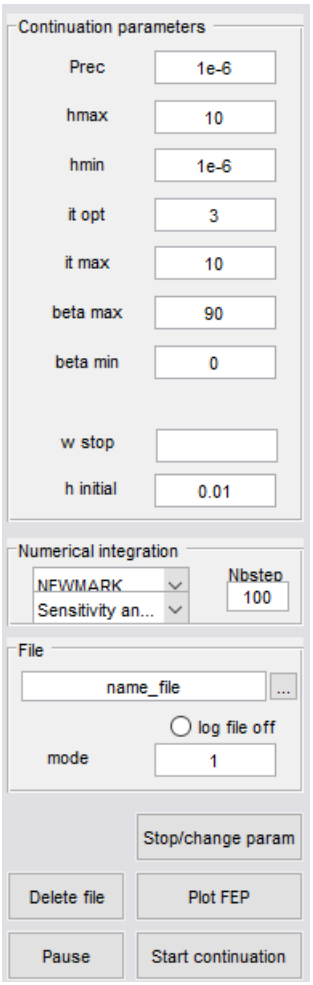


FIGURE 4.7: Right hand side of the NNMcont GUI, with the 'continuation parameters', 'numerical integration' and 'file' panels for the input parameters, and some control buttons for the NNMs computation and file and parameter control.

FIGURE 4.8: More computation options and tool buttons on top of the NNMcont GUI.

The usage of the NNMcont GUI and each of its elements are detailed in chapter 5 during the computation of NNMs branches of the 2dofs nonlinear oscillator 2.3.

4.3.1 NNMcont inputs and outputs

The input parameters of the numerical continuation that can be set on the NNMcont GUI are enlisted below:

NNMs computation parameters:

- *Shooting period* –full/half–: If the interest is on the computation of the symmetrical NNMs the 'half' period option should be chosen. That establishes 3.16 as the shooting function instead of 3.3, calculating shooting functions with only half the trajectory period. This would also cut the computation time by the order of a half.
- *Phase condition*: By clicking this button a window is displayed –figure 4.9– where we select those degrees of freedom whose velocities are set to zero as a part of the phase condition –remember 3.7– for the computation of the NNMs branches.

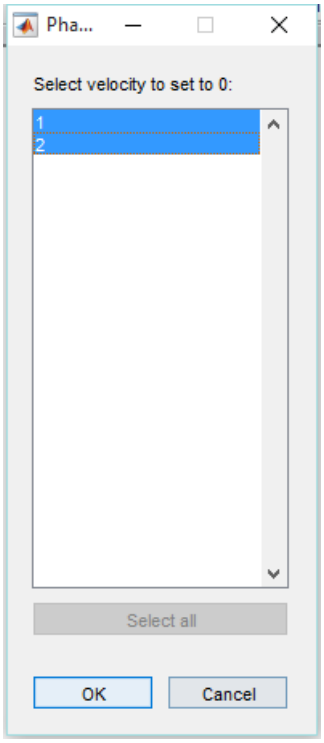


FIGURE 4.9: Window displayed after clicking *Phase condition*, where the degrees of freedom with velocities set to zero are selected.

- *Floquet calculation* –yes/no–: We choose whether to calculate or not the stability of the NNMs computed. Stable modes are displayed on the FEP in blue color, while the unstable modes are represented in red on the same plot.

Continuation parameters:

- *Prec*, convergence precision of the corrector: as noted in 3.5.2 the Newton-Raphson algorithm implemented in the corrector requires a factor of convergence ϵ. Its value must be introduced in this field.

- *hmax* and *hmin*: these values set the maximum and minimal predictor stepsizes h_{min} and h_{max}, defined in 3.5. Their values determine the level of detail and smoothness of the discretization of the NNMs branches numerically computed. Too large values of h_{max} can cause convergence problems or worse, a jump from one curve of solutions –the NNMs branch– to a different one very close to the previous curve during an i-th continuation step.

- *it opt* and *it max*, optimal –N_{opt}– and maximum –N_{max}– number of corrector iterations: these parameters determine the predictor stepsize in each continuation step along with h_{max} and h_{min} as noted in section 3.5.1.

- *beta max* and *beta min*, maximum and minimal angle between consecutive predictor vectors: these options allow to set boundaries to the angles between the predictor vectors of two consecutive continuation steps –i-th and (i-1)-th– whether for branch switching hindrance β_{min} or for stepsize control –β_{max}–.

- *w max*: This is an option whose function is unknown, if it had any. No clue was found even in [26].

- *h initial*, initial stepsize $h_{initial}$: this is the stepsize parameter value taken during the predictor phase of the first step of the continuation sequence.

Numerical integration parameters:

- *Numerical integration* –Runge-Kutta/Newmark–: this option allows to choose the method of integration of the trajectories for the evaluation of the shooting function in the phase space.
- *Nbstep*, number of time integration steps along one shooting period: more time steps brings more precision to the numerical computation of trajectories of dynamical systems such as those calculated for the shooting function, but increases computation time.
- Jacobian calculation method –sensitivity analysis/finite difference–: jacobian matrices of $\mathbf{F}(\mathbf{z}_i, T_i)$ for the predictor phase –3.17– and of $\mathbf{H}(\mathbf{z}_i, T_i)$ for the corrector phase –3.18– of the sequencial continuation can be numerically computed by any of these techniques as explained in 3.8.

File parameters:

- *File*, name of MAT-file: the name of the Matlab data file where the numerical outputs are saved. This file is saved in the working path directory –where the model definition files are contained–.
- *mode*: mode number –out of the system linear normal modes– whose nonlinear extension has to be computed. The lowest order modes are those with lower frequencies, and viceversa with the highest order modes.
- *Log file off* –radio button–: if selected, a log file that records the computation events is not produced.

On the other hand, the following are the numerical and matricial outputs given by the computation of NNMs with this package. These results are saved in a MAT.file or 'Matlab data file' in each computation run.

- *Angle_beta*: row array that contains, for the i-th NNM computed, the beta angle between the two consecutive i-th and (i-1)-th predictor vectors –in degrees–.
- *Energy*: the total mechanical energy of each i-th NNM.
- *Floq*: the $2N$ complex Floquet multipliers of each NNM, where N is the number of dofs of the system.
- *Itnumber*: number of corrector iterations in each i-th continuation step.
- *Prediction*: tangent vector $\mathbf{v}_i$ computed on each i-th continuation step during the predictor phase.
- *Pulsation*: pulsation $-\omega-$ of each NNM motion in $rad\ s^{-1}$.
- *Sol*: two-dimensional array where each i-th column is the mode shape of the i-th NNM on the branch with coordinates $\mathbf{z}_i = (\mathbf{x}_i, \mathbf{x}_i')^T$.
- *Stepsize*: stepsize of the predictor phase of each i-th continuation step.

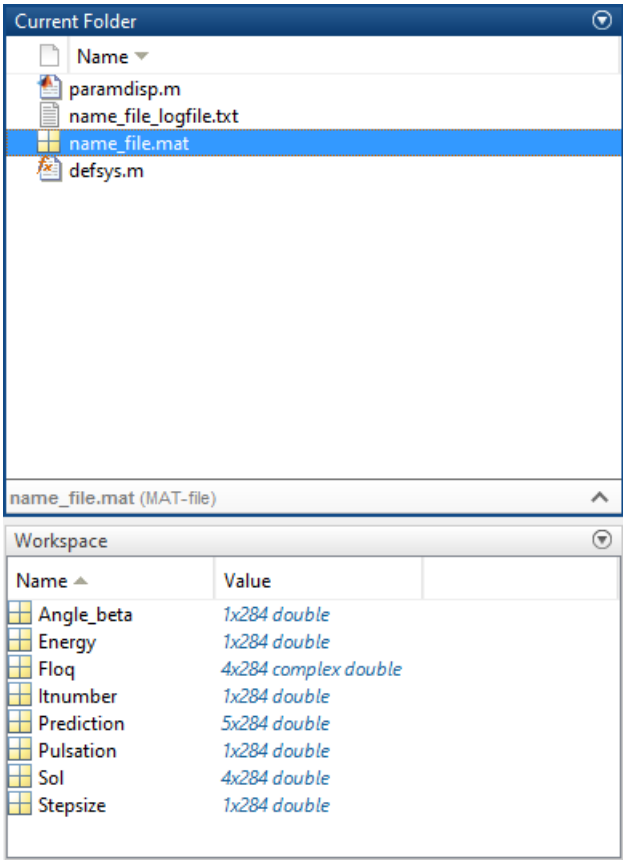


FIGURE 4.10: Contents of the Matlab data file with the outputs of the computation.

4.4 Model definition

The system whose nonlinear analysis has to be conducted is defined in a function m-file called *defsys.m* inside the current path directory with the following structure elements as an output:

- *sys.Klin*, *sys.Mlin*: linear stiffness matrix [**K**] and mass matrix [**M**].
- *sys.nl*: vector of objects of a class defining the nonlinear stiffness elements contained in our system and modeled by the nonlinear component $\mathbf{f_{nl}}(\mathbf{x})$.

As well as other optional elements:

- *sys.norm*: Norm of the linear mode used as an initial guess for determining a low energy NNM motion.
- *sys.IGcont*: another initial guess for the continuation algorithm that contains the vector *sys.IGcont.x0* of initial displacements –velocities are assumed to be equal to zero– and an initial pulsation *sys.IGcont.w*.
- *sys.vitfix*: numbers of the degrees of freedom whose velocities are set to zero as phase condition.

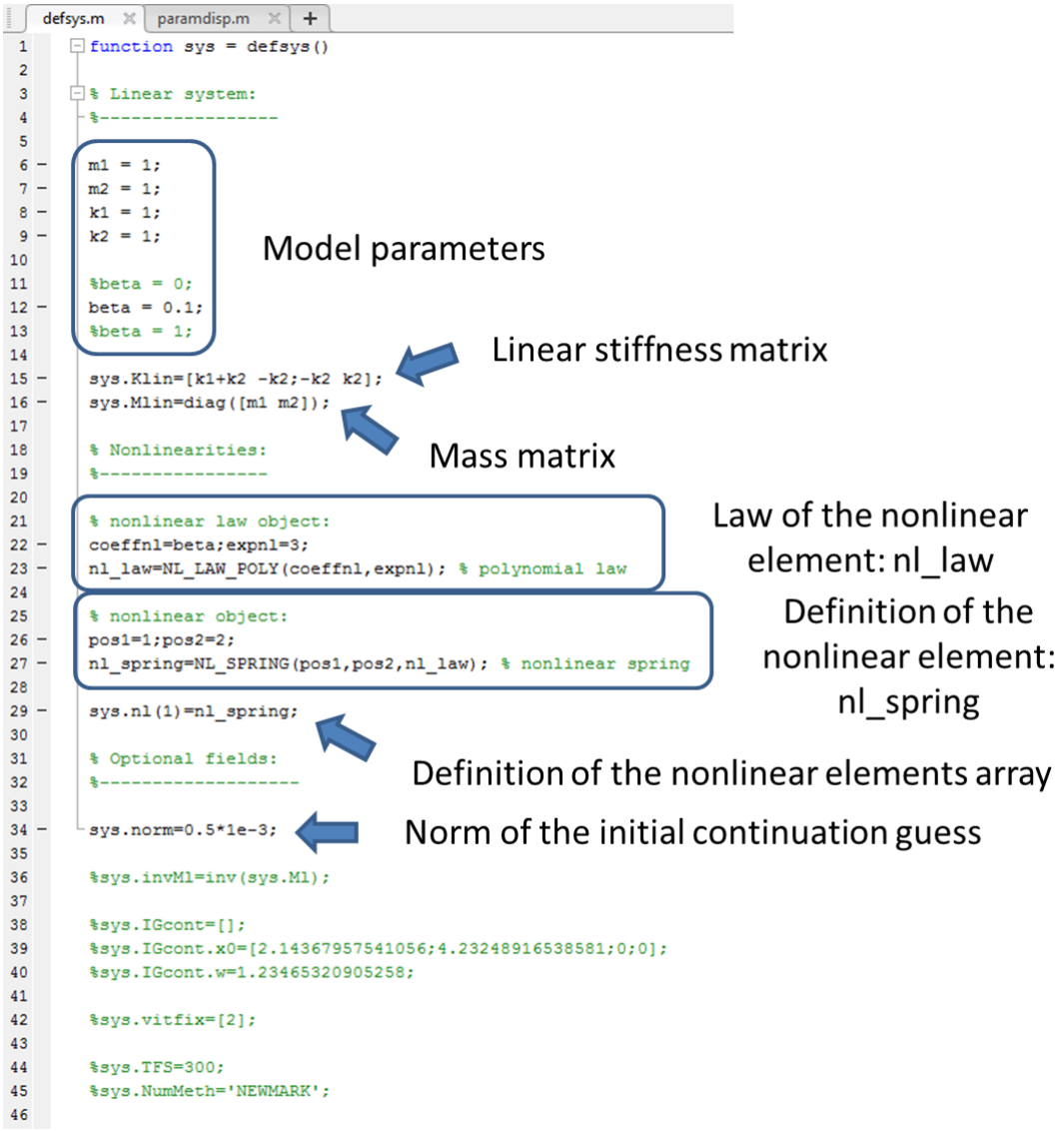


FIGURE 4.11: This is the *defsys.m* file where the 2dofs system 2.5 is defined for the numerical continuation of its NNMs. Model parameters and stiffness and mass matrices are straightforward to introduce. Nonlinear element definition is the object of the following section. *sys.norm* determines the norm of the initial guess –the corresponding linear mode– for the start of the continuation algorithm.

Another m-file in the same directory called *paramdisp.m* contains the properties of the results display.

defsys.m | paramdisp.m

```
% User Display for "NNM Plot" button:
% Inputs:
% - x is the response (state vector) of the system over one time period
% x(1:end/2,:), displacements (time evolution);
% x(end/2+1:end,:), velocities (time evolution);
% - t, time vector.

figure
% Motion in configuration space
plot(x(1,:),x(2,:))
xlabel('x_1')
ylabel('x_2')
```

FIGURE 4.12: The *paramdisp.m* file for the same model. This function m-file defines which graphic output is produced when clicking the *Plot NNM* button as it is seen in more detail next chapter. In this case, the modal curve defined by the displacements of the two dof –x_1 and x_2– is plotted.

When NNMcont is started in the Matlab command window those two function files are automatically loaded from the current path directory.

4.5 Definition of the nonlinearities

4.5.1 Nonlinear elements

Any nonlinear elements that our mechanical models could contain are defined by Matlab classes contained in a folder called *NL_ELEMENT* inside the directory *CLASS_NL*.

In version 1.1 of NNMcont this folder contains two subfolders, *@NL_SPRING* and *@NL_USER*, each holding a Matlab class.

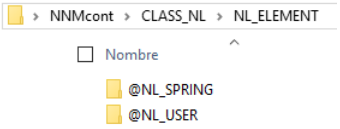


FIGURE 4.13: The two default nonlinear element classes in NNMcont 1.1.

The user can define the classes he needs. Each of these classes contains three methods: *fint_nl.m*, *dfint_nl.m* and *Energy_nl.m*. The purpose of these methods is to return, respectively, the global nonlinear elastic force vector $\mathbf{f_{nl}}(\mathbf{x})$, the global matrix of spatial derivatives of $\mathbf{f_{nl}}(\mathbf{x})$ with respect to the displacements and the nonlinear elastic energy L_{NL} conveyed by the element defined by means of that class.

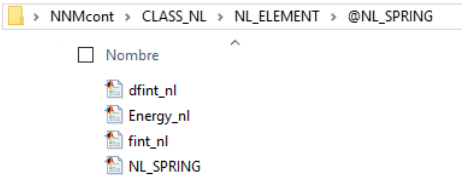


FIGURE 4.14: The three methods of each nonlinear element class, plus the Matlab class constructor: *@NL_SPRING*.

The *@NL_SPRING* class is the classic spring with two nodes. The *@NL_USER* class is a sort of template that can be taken to define any nonlinear element class needed.

The class *@NL_SPRING* is defined by these attributes –figure 4.15–:

- *DDL1* and *DDL2*, its first two attributes: dof numbers of the spring nodes. If one of those nodes is attachfed to the ground it is assigned a dof number of zero.
- *nl_law*, its third attribute: a nested object of a class that defines which nonlinear elasticity law our spring element follows. More on nonlinear laws for nonlinear elements in the next section.

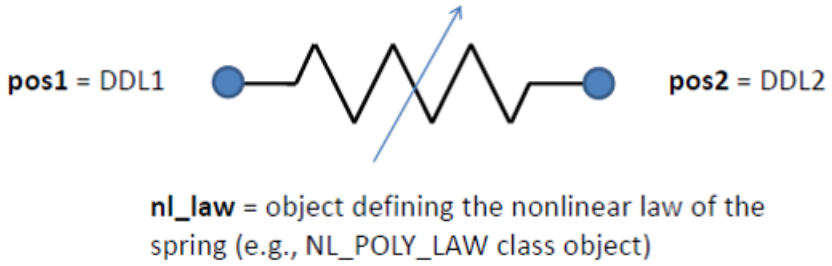


FIGURE 4.15: The three attributes of any nonlinear element object of the class *@NL_SPRING*: the position of the nodes in the model and the elasticity law it follows. This image was taken from the reference [26].

The introduction of a nonlinear element in the mechanical model requires the definition of the variable *sys.nl* inside the function m-file *defsys.m*. This parameter is a one-dimensional array where each element is an object of any nonlinear class of those defined in the folder *NL_ELEMENT*. As an example, next figure depicts how the nonlinear spring of our two degrees of freedom oscillator has been defined in NNMcont:

```
% nonlinear law object:
coeffnl=beta;expnl=3;
nl_law=NL_LAW_POLY(coeffnl,expnl); % polynomial law

% nonlinear object:
pos1=1;pos2=2;
nl_spring=NL_SPRING(pos1,pos2,nl_law); % nonlinear spring

sys.nl(1)=nl_spring;
```

FIGURE 4.16: Definition of a polynomial cubic –*expnl*= 3– law, *nl_law*, of the class *NL_LAW_POLY* with coefficient *coeffnl*= β. And then, of a nonlinear element object *@NL_SPRING* –the nonlinear spring of 2.5– of class *@NL_SPRING* and the position of its nodes and the nonlinear law previously defined as attributes. The object *@NL_SPRING* is introduced in the array *sys.nl*.

4.5.2 Nonlinear laws

The *nl_law* attribute plays a key role in the definition of the elastic law that the nonlinear element defined from a nonlinear element class follows. The attributes of this class are the law parameters and coefficients, while its methods, enlisted below, return the numerical values of nonlinear force, spatial derivatives and nonlinear energy as a function of those law parameters, plus the deformation values of the degrees of freedom of the element.

- *f_nl.m*: returns the local nonlinear elastic restoring forces of the element.
- *df_nl.m*: returns the local submatrix of spatial derivatives of the nonlinear restoring laws
- *energy_nl.m*: : returns the nonlinear elastic energy of the element, L_{NL}.

The methods of the nonlinear element class –*fint_nl.m*, *dfint_nl.m* and *Energy_nl.m*– call each of the nonlinear law methods from above and add the local values they return to the global nonlinear force vector $\mathbf{f_{nl}}(\mathbf{x})$, spatial derivatives matrix and nonlinear energy value, respectively.

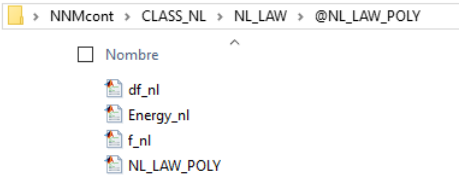


FIGURE 4.17: The methods of the nonlinear law class *@NL_LAW_POLY* and its class constructor.

All nonlinear laws that our nonlinear elements can follow are defined in the folder *NL_LAW*. The default NNMcont classes, all contained in the folder cited above, in version 1.1 are:

- *@NL_LAW_POLY*: Polinomial law, the stiffness is proportional to the element deformation with a polynomical dependence: $f_{nl} = kx^e$, with $k =$*coeffnl* and $e =$ *expnl* as class attributes, and x as the deformation between the element nodes

- *@NL_LAW_LINPIECEWISE*: Piecewise law, the element has different linear stiffness coeficients in different deformation intervals. This class allows two stiffness intervals with coefficients k_1 and k_2. The element is assigned coefficient k_1 for deformations in the interval $[-a, a]$ and coefficient k_2 in $(-\infty, -a) \cup (a, \infty)$. Therefore its attributes are k_1, k_2 and a.

- *@NL_LAW_UNILAT*: Unilateral law, the element has non-zero linear stiffness coefficient *coeffnl* if and only if its deformation x is positive. *coeffnl* is its single class attribute.

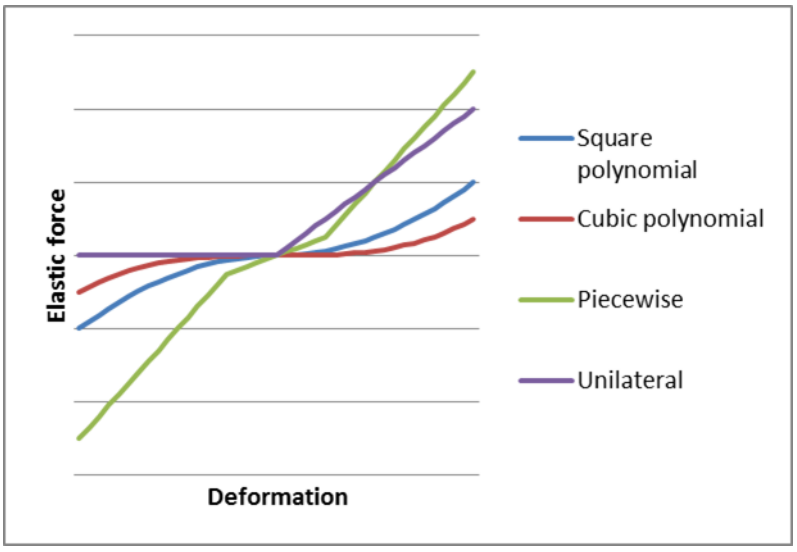


FIGURE 4.18: Elastic curves for each nonlinear law.

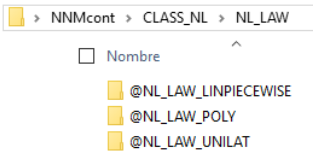


FIGURE 4.19: Nonlinear law classes in NNMcont 1.1.

Nonlinear law classes can be defined by the user if needed to suit the problem to be solved.

Chapter 5

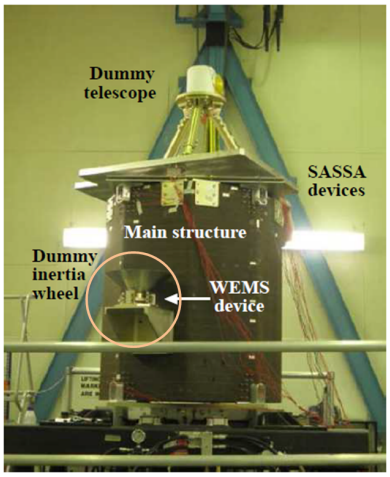
A PRACTICAL EXAMPLE: A TWO DEGREES OF FREEDOM OSCILLATOR

In this chapter, the NNMcont package presented in 4 is applied to find NNMs branches in the equations modeling the two degrees of freedom system presented in section 2.1.2.

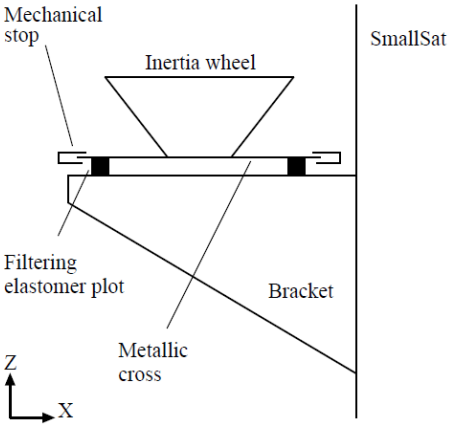
As argued in 2.4.4 this system was chosen for analysis despite its simplicity because it is an accurate enough model of the two most notorious applications of nonlinear vibrations: vibration control with passive or active nonlinear mass dampers, and energy harvesting with nonlinear vibration harvesting.

The methodology followed is to perform several analysis of each NNMs main branches –the mode one and mode two branches emerging from the LNMs– with different continuation parameters, and then trying to force branch switching by experimenting with different parameter values –remember that neither the algorithm nor the software used has no implemented branching strategy–. The analysis and interpretation of the results are treated along this chapter after each computation. On the other hand, the comparison between the analytical method implemented in chapter 2 and the continuation algorithm is an aim of the Project conclusions in the next chapter –6–.

Throughout this chapter, the steps needed for the computation and post-processing of the results are remarked. This is part of the commitment to make this text serve as an aid to any person who wishes to replicate the results or to apply the software on another model. It must be noted that NNMcont can be applied on systems with any number of degrees of freedom, and that the continuation algorithm with shooting function has been successfully applied on the modal analysis of models of huge size such as satellite structures with vibration and motion insulators –[25]– and full-scale aircraft fusselages –[22], [24]–.

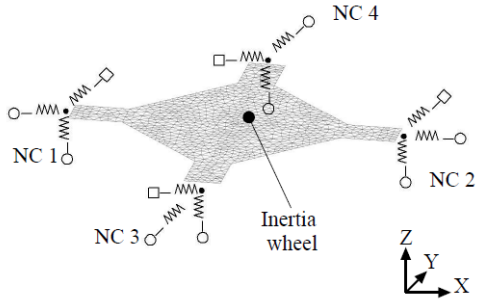


(A) Picture of the SmallSat spacecraft with its inertial wheel attached to the satellite structure through an filtering device called 'WEMS': Wheel Elastomer Mounting System, designed to insulate the structure from the high-frequency motion.

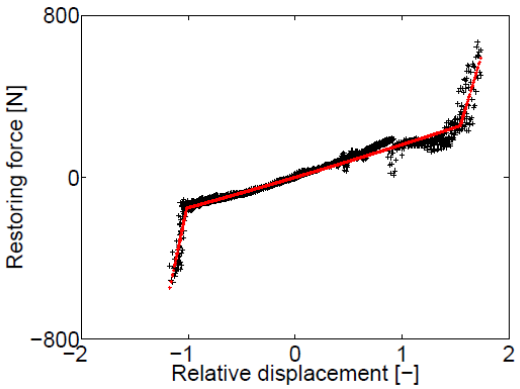


(B) Schematic of the vibration insulation device called 'WEMS': the inertia wheel is attached to a metallic cross supported by four filtering elastomer plots on its tips. Mechanical stops limit the motion of the plate. Those stops are responsible of the elastic nonlinearities of the attachment for large deformations as their stiffness is much higher than that of the elastomer plots.

FIGURE 5.1: Reference [25] compares the results of the experimental and computational modal analysis on the structure of the SmallSat satellite. Strong nonlinearities on the structure could not be dismissed from the computational analysis as nonlinear phenomena was reported on the experimental analysis. Therefore a nonlinear modal analysis was carried out on both forced damped –frequency response analysis– and free undamped –nonlinear normal modes computation– models. The most significative results diagrams are shown through the paper.



(A) Simplified model of the WEMS mechanism with the inertial wheel as a mass point and the elastic attachments as springs. mobile –linear springs– and fixed –nonlinear springs– attachments are represented by squares and circles respecively.



(B) Experimental stiffness curve –black crosses– of one of the plate attachments, and the model fitted to the stiffness curve, in red.

FIGURE 5.2: Modelization of the nonlinear WEMS of the SmallSat satellite.

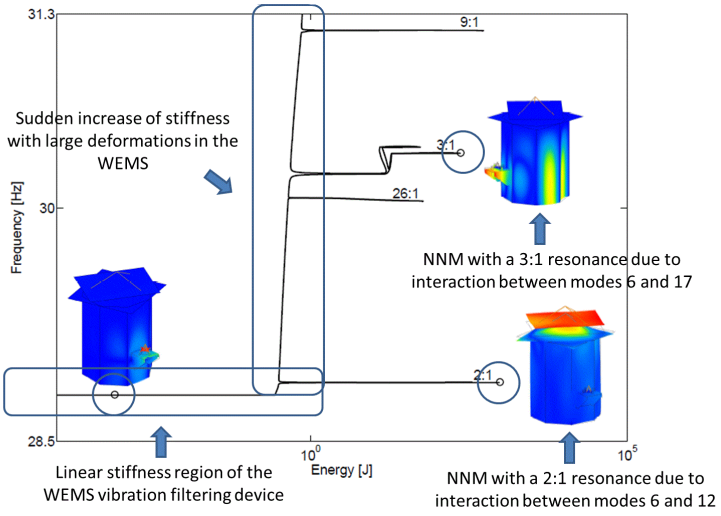


FIGURE 5.3: FEP of the sixth mode of the SmallSat structure, obtained with computational nonlinear modal analysis. This is the sixth NNMs main branch. At low energies of oscillation, or small deformations of the WEMS, the systems behaves according to the linear model. When deformations are large enough to make mechanical stops limiting the motion of the metallic cross be hit, the linear threshold given by the stiffness curves is broken. Under such conditions nonlinear phenomena emerge including jumps, resonance, unstability and frequency-energy dependence emerge.

5.1 System description and modelization

Here, we remind the equations that model the conservative –undamped– two degrees of freedom oscillator in free vibration –unforced– with a nonlinear spring attached to masses one and two. This model is portrayed in 5.4:

$$\begin{aligned} M_1 x_1'' + K_1 x_1 - K_2(x_2 - x_1) - \beta(x_2 - x_1)^3 = 0 \\ M_2 x_2'' + K_2(x_2 - x_1) + \beta(x_2 - x_1)^3 = 0 \end{aligned} \tag{5.1}$$

For the purpose of its implementation in the model files of NNMcont, this system of equations is rewritten in matricial format:

$$\begin{pmatrix} M_1, & 0 \\ 0, & M_2 \end{pmatrix} \begin{pmatrix} x_1'' \\ x_2'' \end{pmatrix} + \begin{pmatrix} K_1 + K_2, & -K_2 \\ -K_2, & K_2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} -\beta(x_2 - x_1)^3 \\ \beta(x_2 - x_1)^3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{5.2}$$

From this SODE the nonlinear modal analysis of the system can be carried out.

5.2 Definition of the model in NNMcont

Following the steps noted in the last chapter –4– that model can be taken into NNMcont for NNMs computation. In this section the parameters defining the model described by the system 5.2 in the model file *defsys.m* are outlined. Remember first that *defsys.m* is a function m-file that returns a Matlab structure named 'sys' with the following elements: the linear stiffness matrix, the mass matrix and an array of nonlinear stiffness elements –where each element of the array is an object with attributes–.

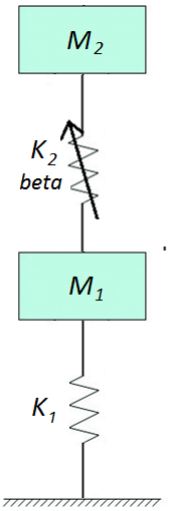


FIGURE 5.4: The model of the free 2dofss nonlinear undamped oscillator for its modal analysis.

The linear stiffness and mass matrices –*sys.Klin* and *sys.Mlin*– of 5.2 are introduced in the model file as:

```
sys.Klin=[k1+k2 -k2;-k2 k2];
sys.Mlin=diag([m1 m2]);
```

Meanwhile, the nonlinear stiffness element uniting both masses is defined in *defsys.m* as:

```
% nonlinear law object:
coeffnl=beta;expnl=3;
nl_law=NL_LAW_POLY(coeffnl,expnl); % polynomial law

% nonlinear object:
pos1=1;pos2=2;
nl_spring=NL_SPRING(pos1,pos2,nl_law); % nonlinear spring

sys.nl(1)=nl_spring;
```

where the first and only element of the array *sys.nl* is an object –*nl_spring*– of the class *NL_SPRING* with attibutes the two spring nodes and the nonlinear law of the element. The nonlinear law is another Matlab object. More details on the implementation in section 4.5.1.

The different model parameters required can be defined on top of the function file or along the code itself:

```
m1 = 1;
m2 = 1;
k1 = 1;
k2 = 1;

%beta = 0;
beta = 0.1;
%beta = 1;
```

The analysis of the 2dofs oscillator is conducted here with the following generic values for the model parameters: $M_1 = 1$, $M_2 = 1$, $K_1 = 1$, $K_2 = 1$, $\beta = 0.1$.

Finally, some optional elements can be defined in the data structure *sys*. In this example, *sys.norm* determines a default norm for the linear mode shape –one of the LMMs of the system– taken as the initial guess for the start of the conputation of a NNMs main branch.

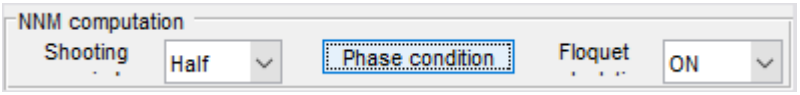
```
% Optional fields:
%--------------------

sys.norm=0.5*1e-3;
```

When the command 'NNMcont' is executed in the command line with the model files in the current Matlab path –see 4.2– the system is loaded into the program. At low energies, nonlinearities are negligible and LNMs and NNMs overlap. In consequence, it is convenient to choose small norms to the initial NNMs branch points.

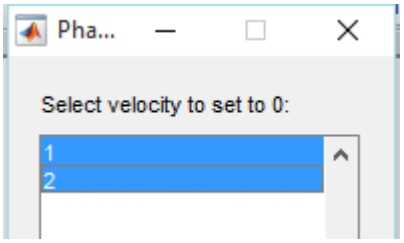
5.3 Computation of the system NNMs branches with NNMcont

5.3.1 Default parameter values

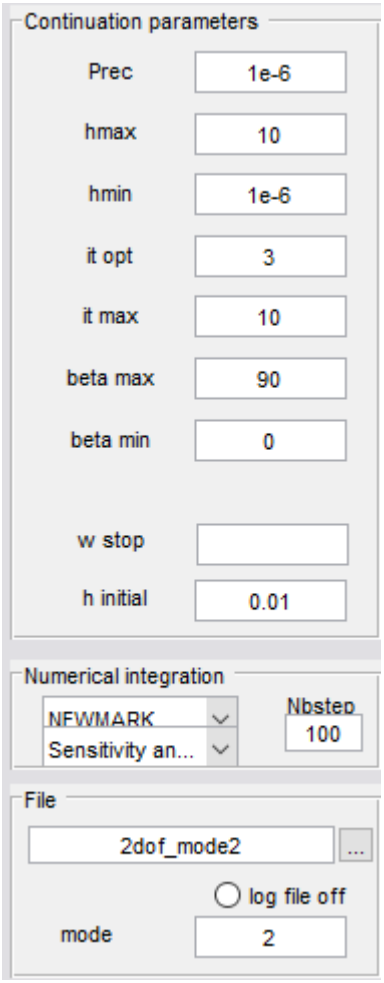
The first computations of the NNMs branches are performed with the default input parameter values shown on the NNMcont window the first time the program is executed:



The shooting period is set by default to the *half period* option, computing only a main branch of symmetrical modes while cutting off the computation times. Also by default, the Floquet multipliers calculation appears switched on, and the velocities of all degrees of freedom are set to zero on the NNMs branch points. Those phase conditions can be checked and ensure by clicking the *phase condition* button and selecting the two dofs:

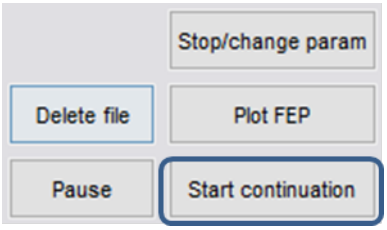


The numerical results are saved in files named *2dofs_mode1* for the first NNMs main branch and *2dofs_mode2* for the second main branch, in the current Matlab directory along with the model files.



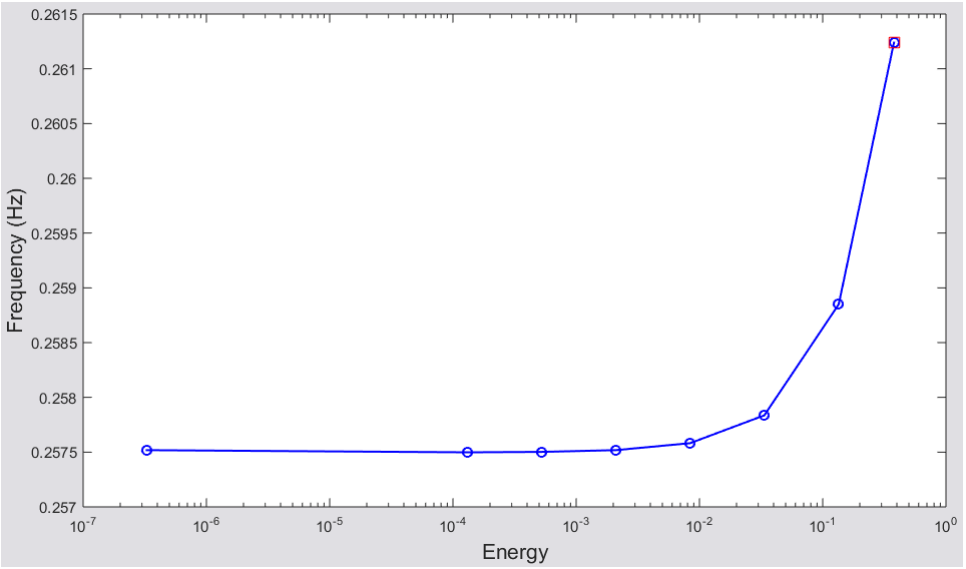
We begin with the computation of the second main branch –the branch emerging from the linear mode with highest natural frequency–, so the text field 'mode' is set with the value two.

To start the computation we simply click the button *Start continuation*:



5.3.1.1 Second mode branch

Step after step, a succession of NNMs belonging to the main branch are calculated. When an i-th point $(\mathbf{z_i}, T_i)$ has just been computed its frequency of oscillation $-\nu_i = \frac{1}{T_i}-$ is plotted on the FEP with respect to the mechanical energy it conveys: E.



To ensure that the stability of the points is depicted on the plot, we have to click on the FEP with the mouse rigth button and select the option *Stability display* on the unfolding list. We also choose the option *Freq. (Hz)* of the same list to represent the frequency in hertzs on the y-axis of the FEP.

The continuation algorithm continues until the button *Pause* is clicked. By doing this the modes computation freezes.

In an advanced point of the computation the second NNMs main branch has this shape on the FEP:

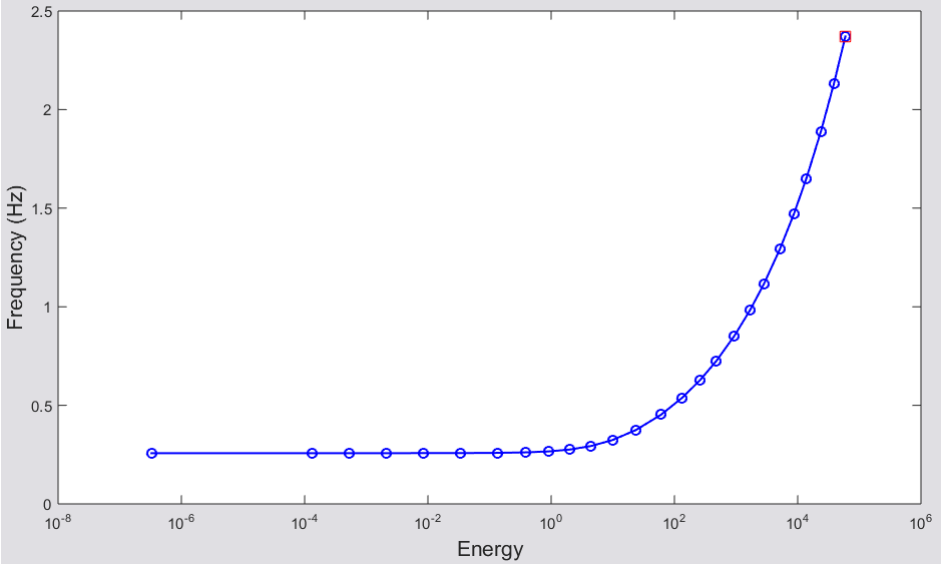
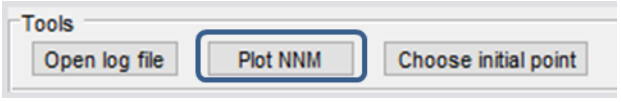


FIGURE 5.5: Second NNMs main branch. As the nonlinear spring follows a hardening law the oscillation frequency of the free periodic oscillations increases with the oscillation amplitude. Also, all the branch modes are asymptotically stable as the points on the FEP are displayed on blue and all the Floquet multipliers –see array *Floq*– have real parts with absolute value smaller than zero. Remember that the mode energy on the x-axis is in a decimal log scale.

With a hardening nonlinearity such as the one present in 5.4 larger oscillation amplitudes increase the fundamental frequency of vibration. That behavior is perfecty reflected by 5.5 as the frequency-energy dependence of the second main modes branch has a positive slope.

Once a NNMs branch has been computed and displayed on the NNMcont plot window the modal curve and time-series plot of each branch point can be obtained. We just click the button *Plot NNM* on the top of the GUI and then click any of the points on the FEP.



The computation of the system orbits in the phase space is carried out on a few points on the branch. After a while, two Matlab plot windows pops up. The first figure shows the modal curve of that NNM with the displacement of the first dof in the x-axis and the displacement of the seond dof in the y-axis. The second figure contains its time-series –the time evolution of the displacements of the dof in that NNM during its minimal period– on top, and the amplitudes of the displacements amplitudes by each dof when the velocities specified in the phase condition are zero.

If we calculate the trajectories of a few modes on the branch, representing them on modal curves and time-series plots, it can be devised how the modes evolve when energy is increased. Naturally, the modes with lowest energy coincide with the linear normal modes. The second linear mode of the system 5.1 has natural frequency $\nu = 0.2575\ Hz$ and mode shape $(-1, 0.618)$, with the two dofs oscillating sinusoidally out of phase. It coincides with the mode depicted on figure 5.6.

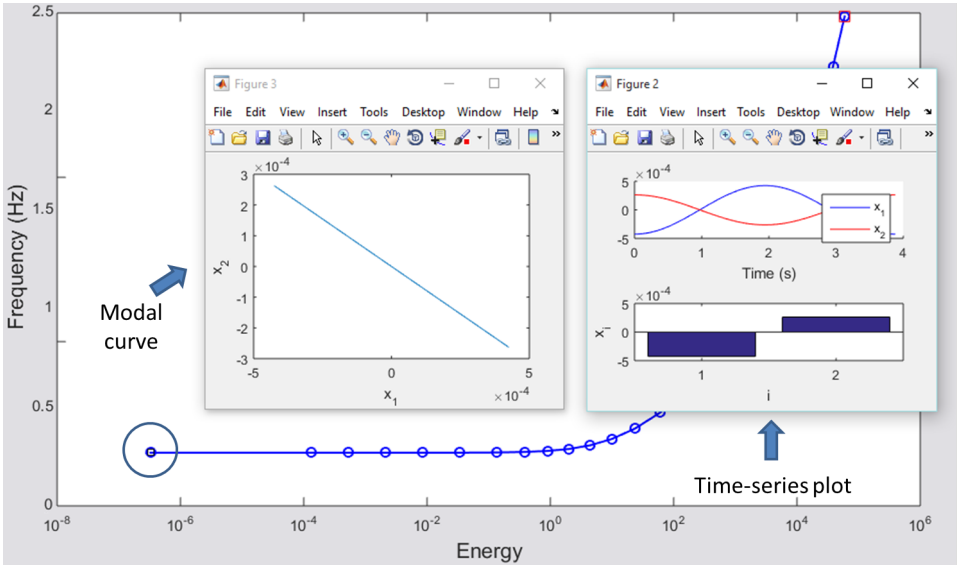


FIGURE 5.6: Modal curve, time-series plot and oscillation amplitudes of a very low energy mode of the second NNMs branch.

As is clear from figure 5.7 as we move along the branch towards higher amplitudes there is an energy transfer from the first mass to the second. In the long term the two dofs oscillate symmetrically with the same amplitudes –i.e. energies– for very large energies while their sinusoidallity is lost. The oscillations of the two masses remain out-of-phase and their fundamental frequency increases –while by definition their minimal period decreases–.

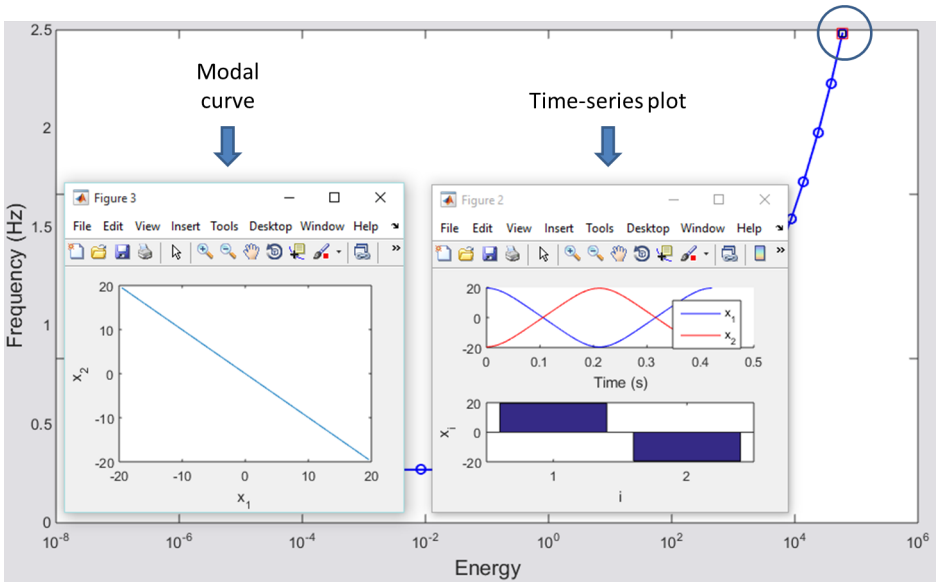
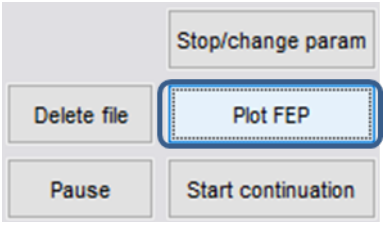


FIGURE 5.7: Modal curve, time-series plot and oscillation amplitudes of a very high energy mode of the second NNMs branch.

It may occur that after plotting a NNM some information on the FEP such as mode stability is lost. In that case we just click the button *Plot FEP*:

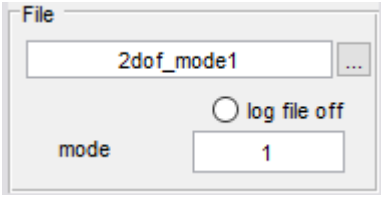


Also, all NNMs on this branch are stable. No stability changes means no bifurcations –no turning points, branching points and symmetry-breaking points–. All modes on the branch are symmetrical and there are no branches of nonlinear modes emerging from the second NNMs branch of this system.

A more detailed analysis with smaller stepsizes makes no sense for this case as there is no intricate nonlinear behavior hidden in this branch.

5.3.1.2 First mode branch

Eventually, we switch to the calculation of the first NNMs main branch. The option *Mode* is set to one and the numerical results are set to be saved in a file called *2dofs_mode1*:



Before starting any new computation from scratch or after changing a parameter value we must click the button *Stop/change param*:

Now the continuation can be started again.

Once a large bunch of points have been obtained the computation can be paused.

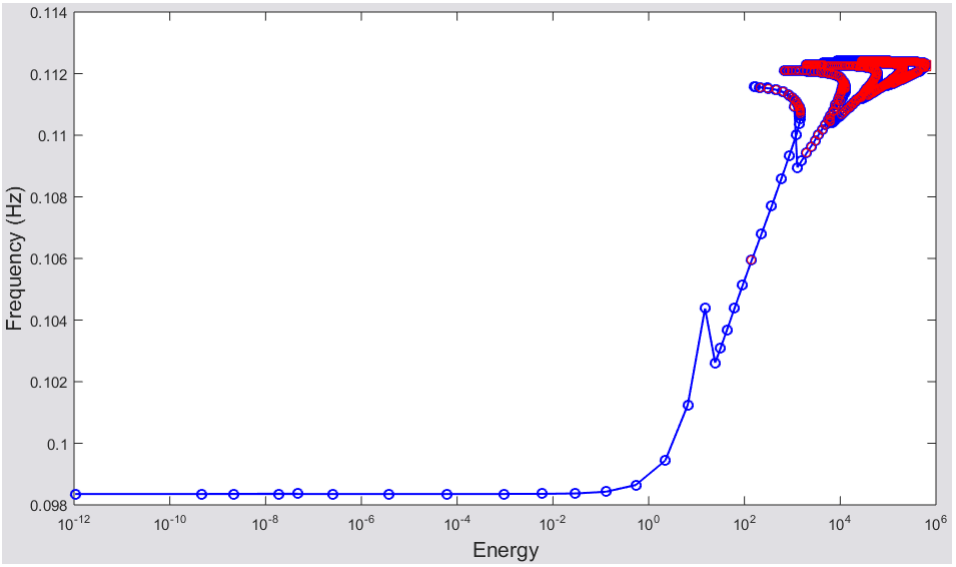


FIGURE 5.8: Second NNMs main branch on a frequency-energy plot. Unstable modes are marked with red spots.

A close-up view on the tongues of internal resonances reveals a richness of nonlinear phenomena that was not present in the second main branch, where all modes were stable with no bifurcations –that is, with no ramification, turning or symmetry-breaking points–. On this branch, more than one normal mode per each linear modes of the system exist –see 2.4.1–. Those multiple modes have dominant harmonics of order greater than one. This is due to resonances between the non-fundamental harmonic terms of the modes of the first branch and the NNMs second main branch –see 'internal resonances' in 2.5.3.2–.

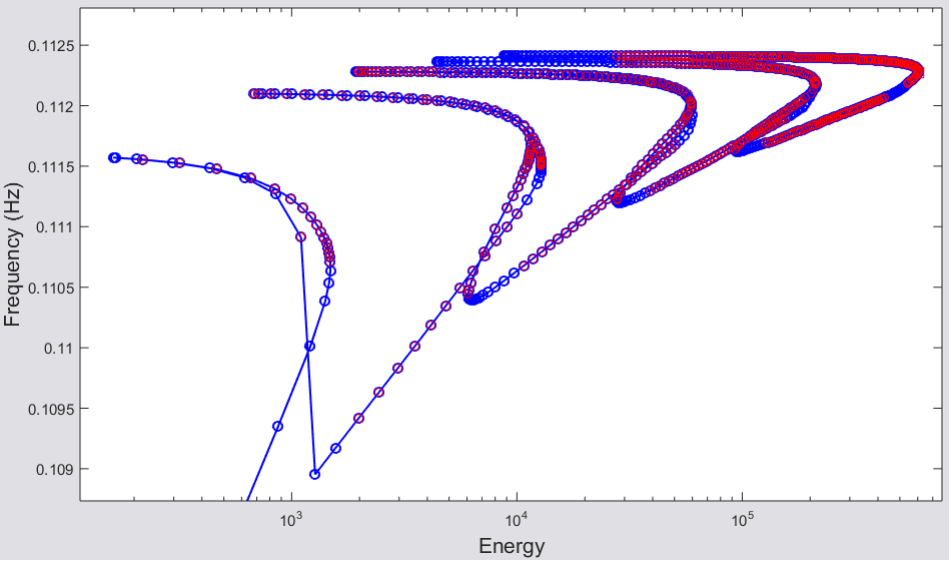


FIGURE 5.9: Detail of the internal resonances of the second NNMs main branch on the FEP.

As in any other main branch of nonlinear modes, at low energies or amplitudes the oscillations resemble the normal mode of the linearized system, in this case, the lowest-frequency linear mode. This linear normal mode has natural frequency $\nu = 0.0984\ Hz$ and mode shape $(0.618, 1)$, being this time an in-phase oscillation. the energy transfer now takes place from the second mass to the first mass along the branch, therefore their fundamental harmonics tend to equal oscillation energies for large amplitudes.

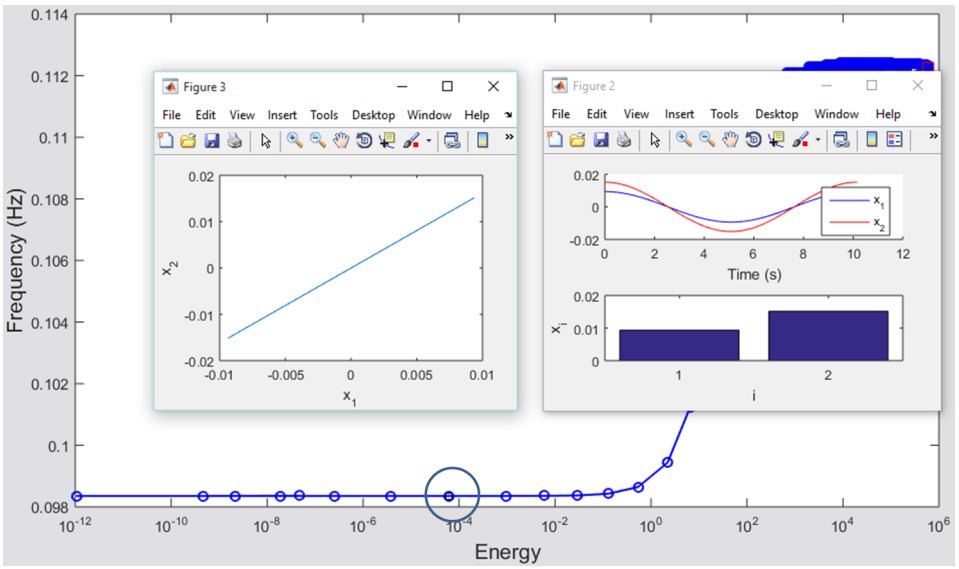


FIGURE 5.10: A low-energy mode on the second NNMs main branch.

But the periodic oscillations only become more complex as the mode energy increases:

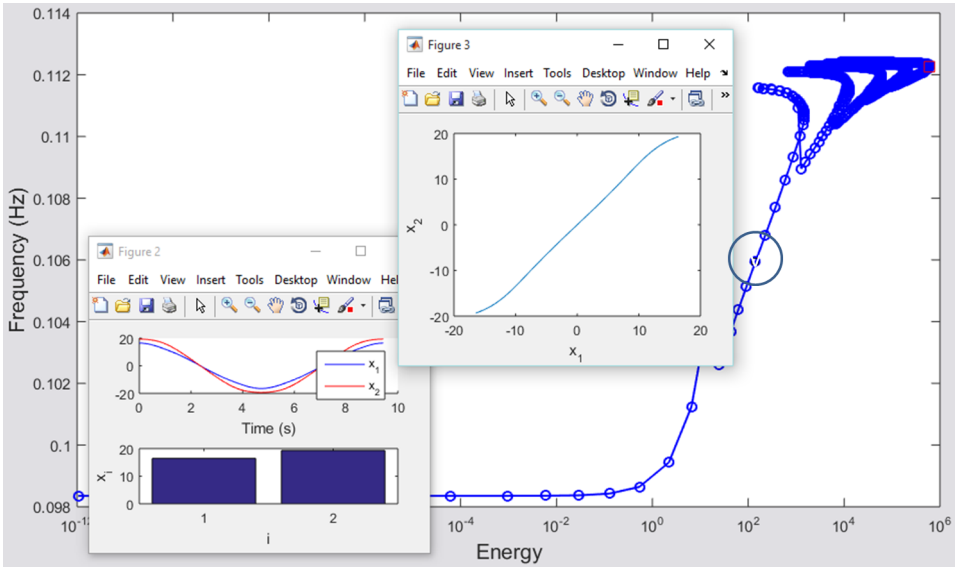


FIGURE 5.11: A medium-energy mode on the second NNMs main branch. This is an unstable point standing between two bifurcation –stability change– points candidates for being branching points.

Interation between NNMs trigger the so-called 'tongues of internal resonance', all these tongues begin with a 1:n resonance, where n is the order of the now dominant harmonic in the first dof. The second dof still oscilates with dominant fundamental frequency, but as we move forward on each branch, this second dof begins to oscillate with the same n-th-otder harmonic frequency, and now the tongue has a n:n resonance, with the two masses oscillating out-of-phase. The resulting modes are symmetrical and resonances are of odd-order on the main branch.

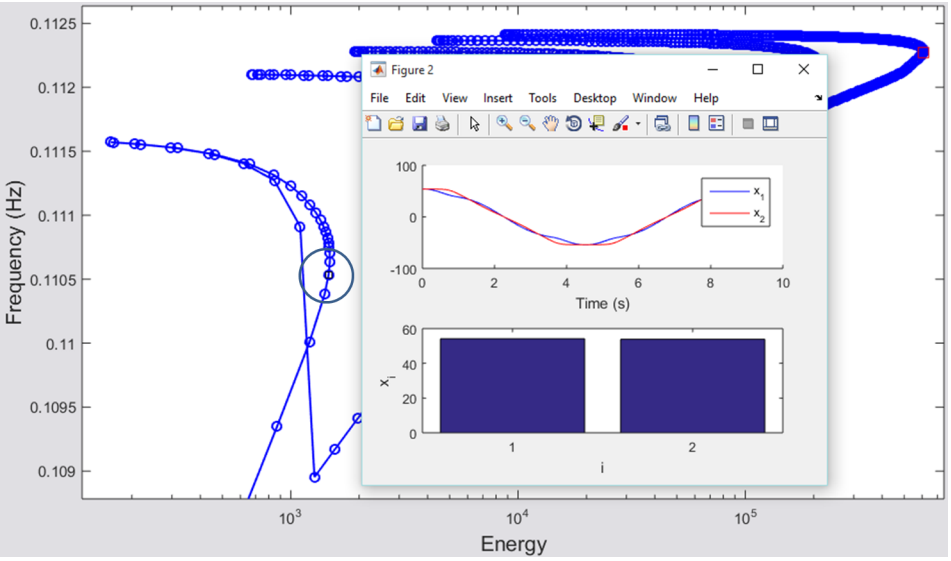


FIGURE 5.12: A 1:5 internal resonance on the second NNMs main branch, at the beginning of one of its 'tongues'.

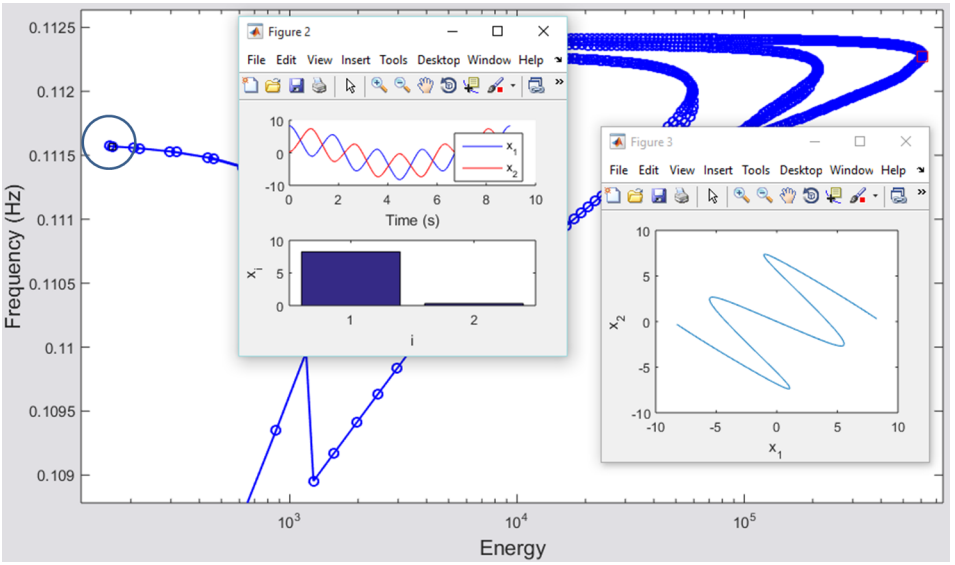


FIGURE 5.13: A 5:5 internal resonance on the tip of a tongue of the second NNMs main branch.

When the tips of the tongues are reached, the dominant frequency on both dofs is the n-th harmonic. After this turning point the n-th internal resonance fizzles out very smoothly.

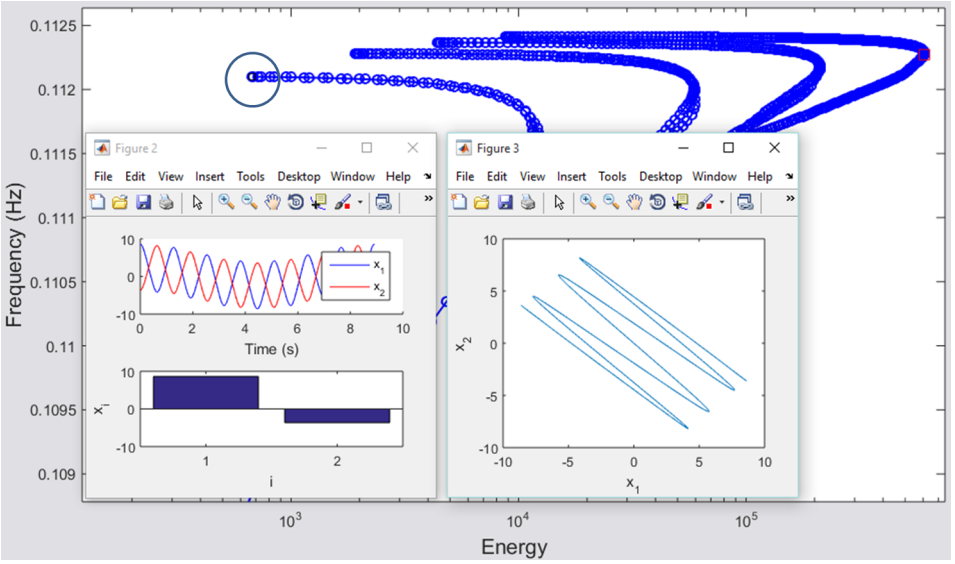


FIGURE 5.14: A 7:7 internal resonance on the tip of the following tongue of the second NNMs main branch.

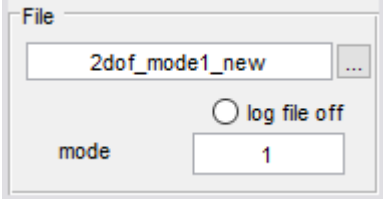
As seen in the FEPs shown above, there are changes of stability along the curve. Those changes mean the presence of bifurcation points of any class of those previously cited.

A third-order internal resonance was missing in the branch computation. A new analysis must be carried out with smaller stepsize values to increase the level of detail of the NNMs branch calculated, targeting the position of branching points or other bifurcation points and expecting to locate and compute those missing modes with internal resonance 1:3 and 3:3.

5.3.2 New parameter values for the first main branch

This time, a maximum stepsize *hmax* of two is set:

The results of this new computation are saved in a new data file named *2dofs_mode1_new*:



With this new analysis it is possible to locate more precisely the stability change point, and more importantly, to compute a section of the curve that came unnoticed in the former calculations due to stepsize values that were too large:

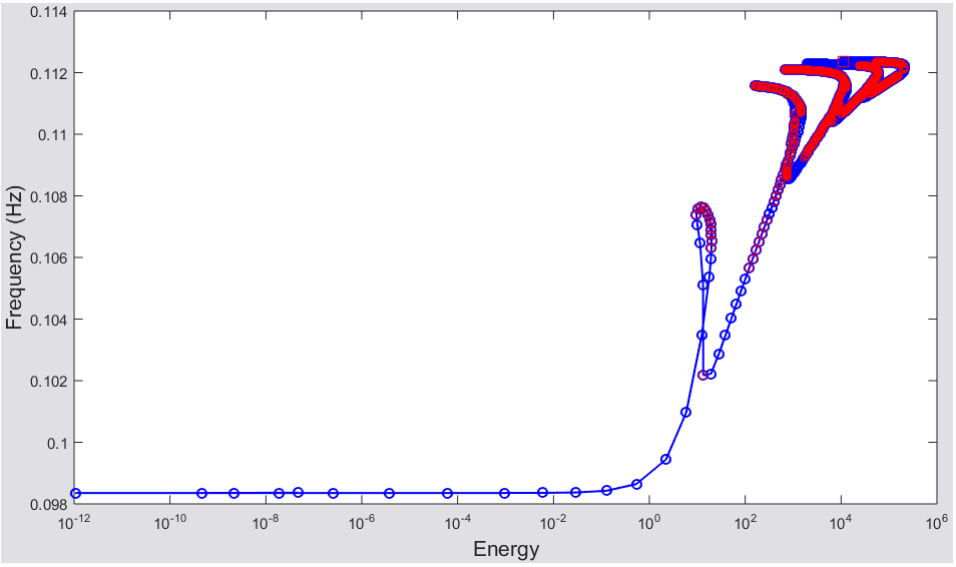


FIGURE 5.15: The first main NNMs branch computed with smaller stepsizes, resulting in a much more detailed discretization.

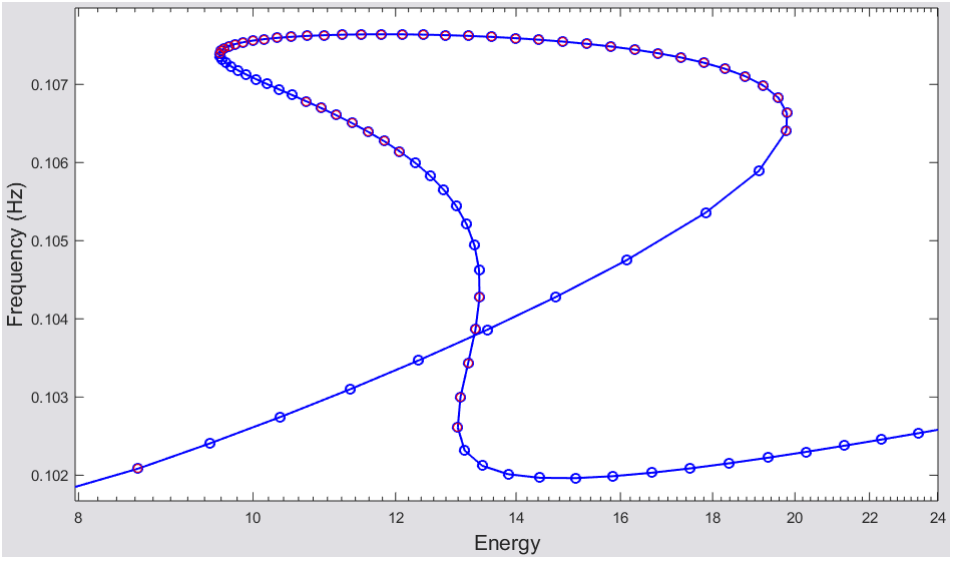


FIGURE 5.16: The new medium-energy tongue on this NNMs branch. Computed with $hmax = 0.25$.

It turns out that the modes on this tongue presents 1:3 and 3:3 internal resonance:

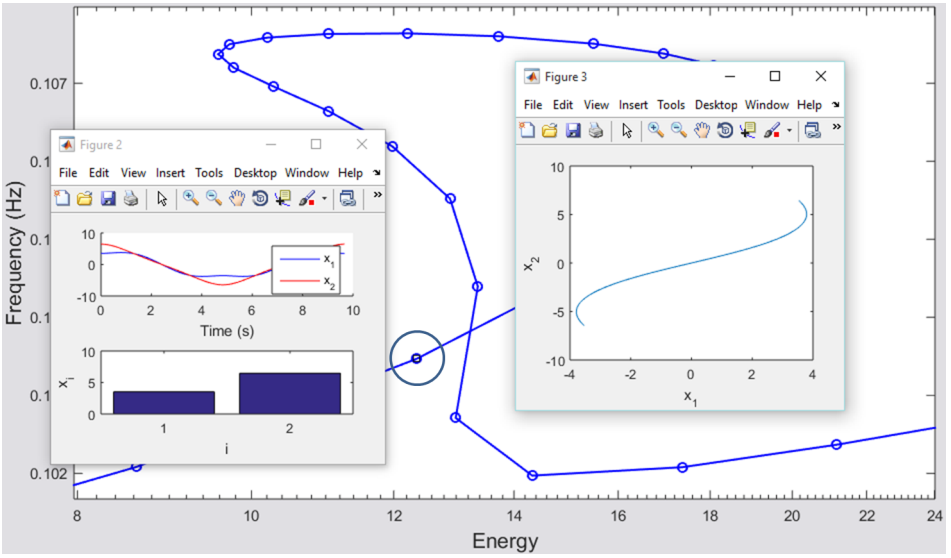


FIGURE 5.17: A 1:3 internal resonance on the main branch, before the first tongue.

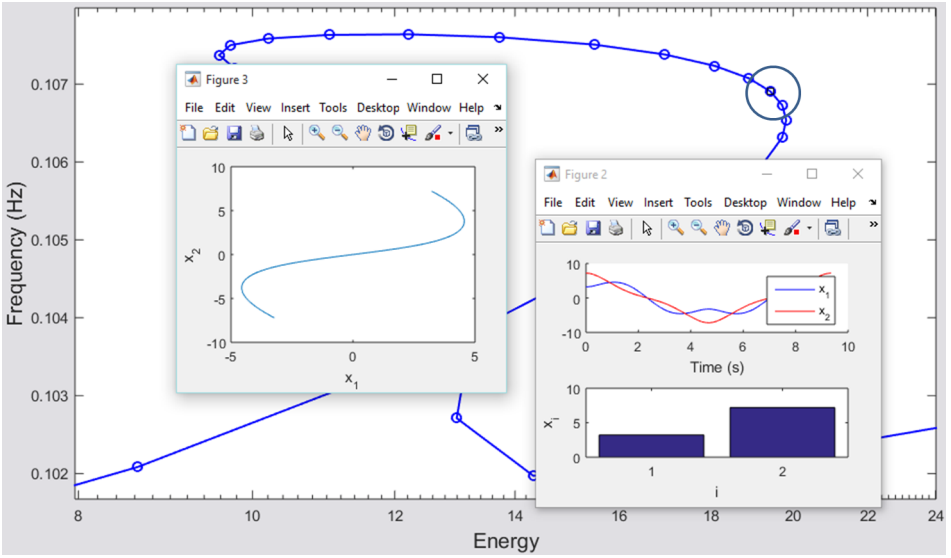


FIGURE 5.18: Emergence of a 3:3 internal resonance on the branch tongue.

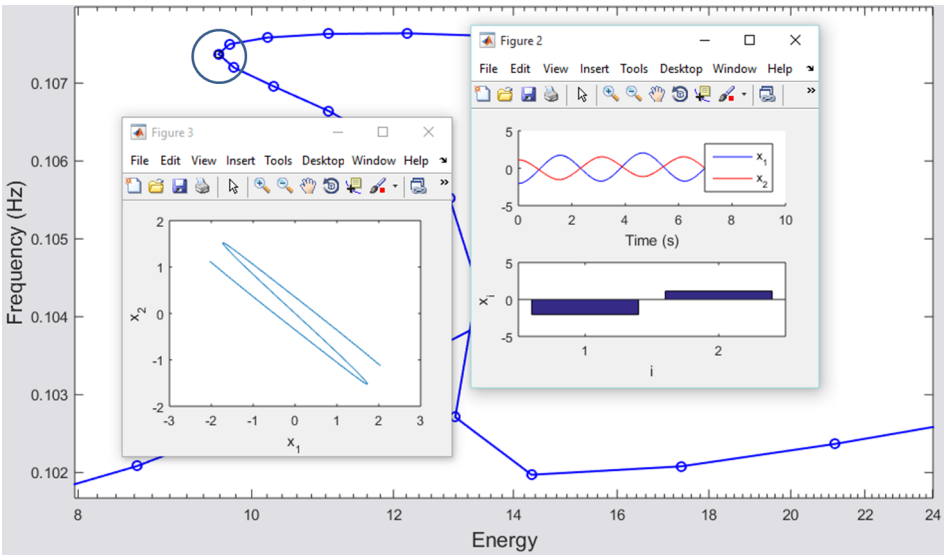


FIGURE 5.19: A 3:3 internal resonance on the tip of the tongue, where the third harmonic is clearly dominant, with the two masses oscillating out-of-phase.

The 3:3 resonance is also symmetrical and out-of-phase, as the third-order harmonics of the two masses oscillate with a phase lag of π radians.

All the resonances on the main branch are odd-order. Even-order internal resonances are ramifications of the main branch. Those ramifications can emerge from bifurcation points on the main branch. With more resolution the bifurcation points can be targeted.

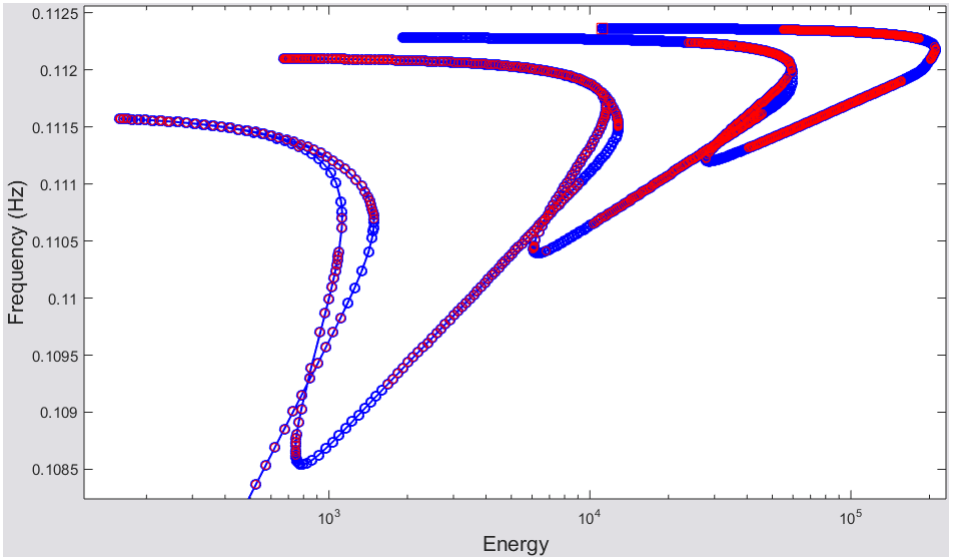


FIGURE 5.20: A close-up view of the 5:5, 7:7 and higher-order resonances with the new stepsizes.

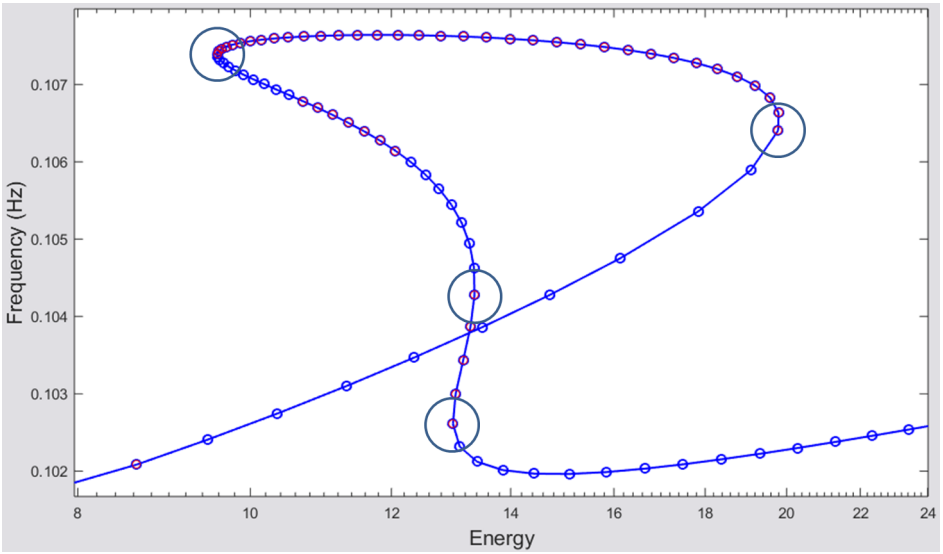


FIGURE 5.21: The turning points –or 'folds'– on the first tongue of resonances 3:1 and 3:3.

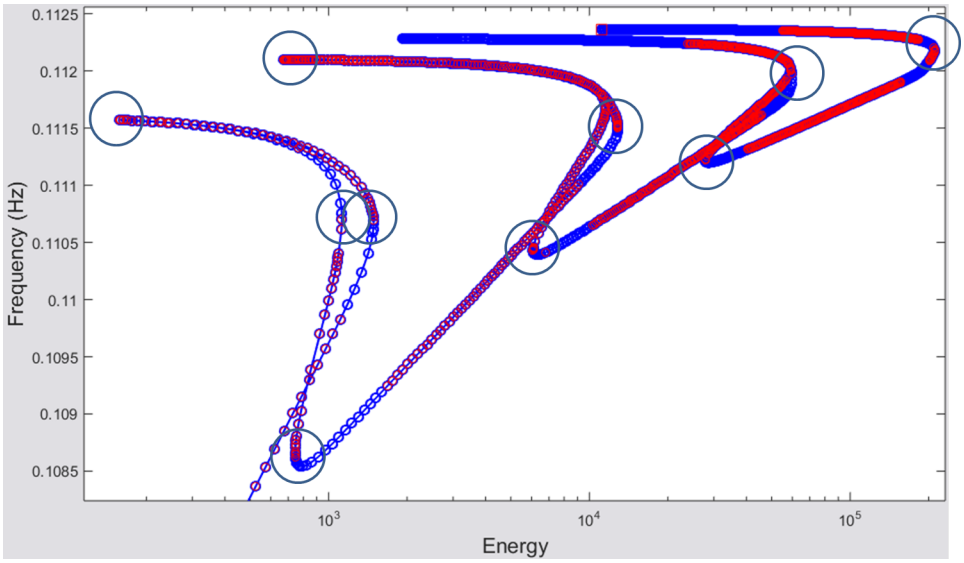


FIGURE 5.22: The turning points –or 'folds'– on the 5:5, 7:7 and higher-order resonance tongues.

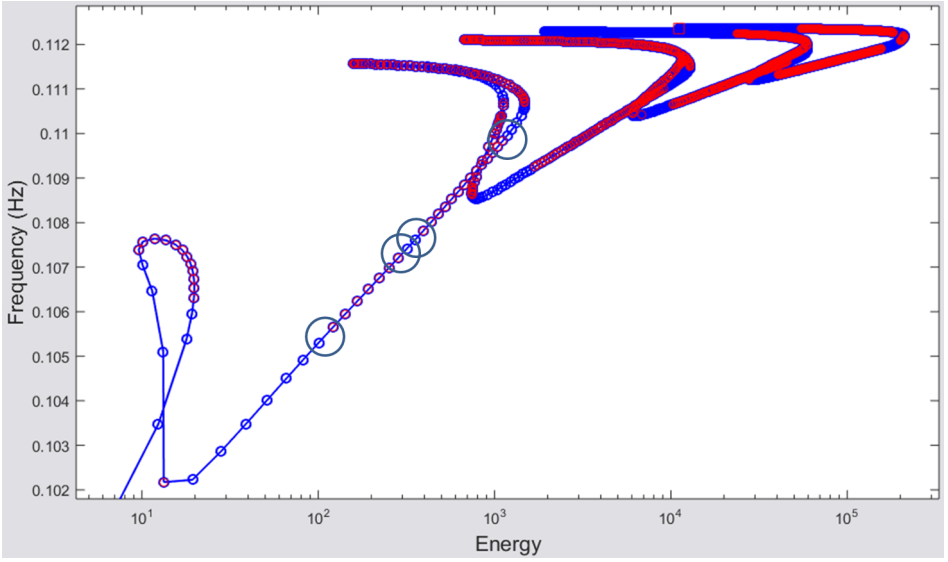


FIGURE 5.23: The sections of the main branch where candidates for branching points are located are circled –first close-up–.

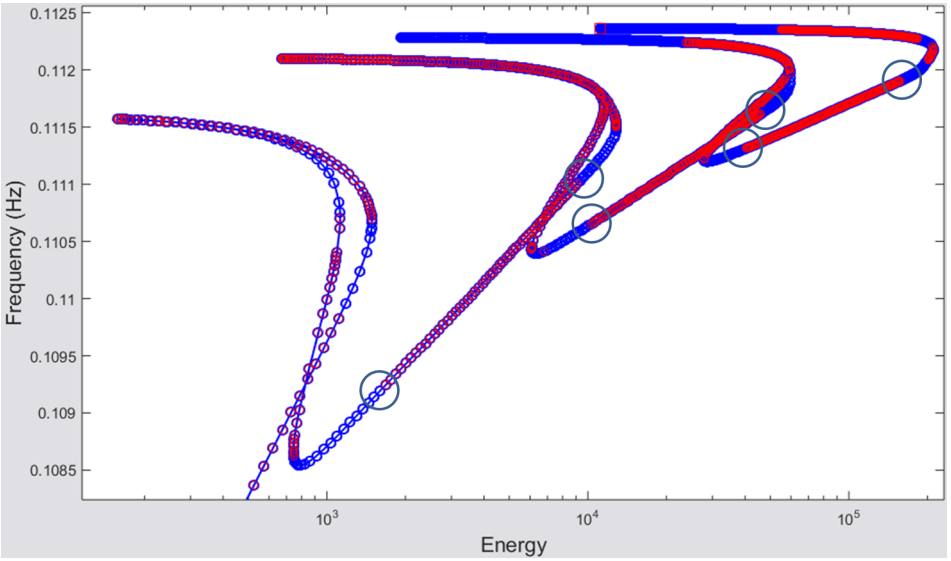
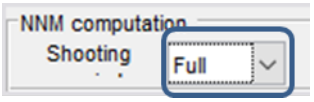


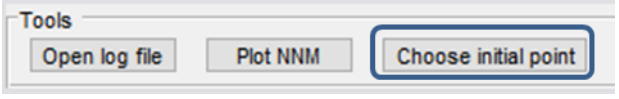
FIGURE 5.24: The sections of the main branch where candidates for branching points are located are circled –second close-up–.

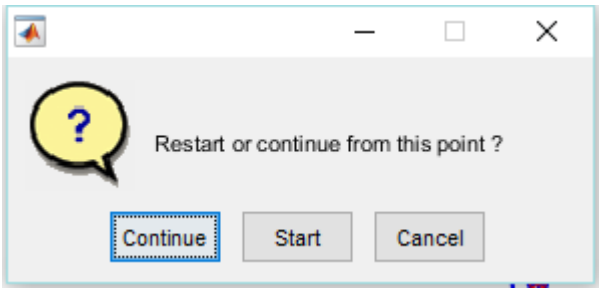
5.3.2.1 Attempting branch switching

The ramifications we presend to compute are branches of unsymmetrical NNMs, therefore the *Shooting function* must be set to 'Full':

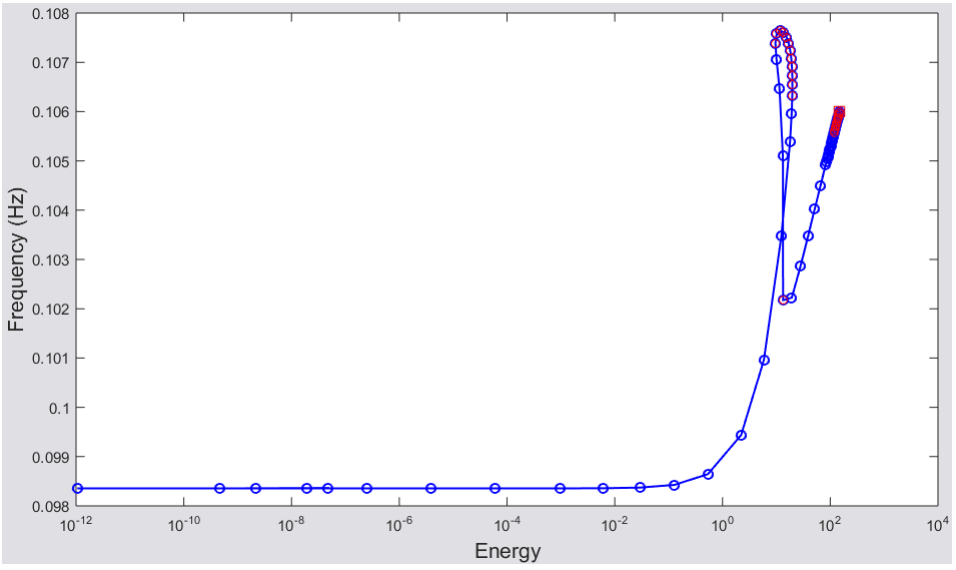


The method I have followed to attempt to force branching at the proximity of a candidate for branching point was this: First, I changed some arbitrary parameters such as the maximum stepsize and the corrector precision parameter *Prec* in a trial-error process. Then, I restarted the branch computation from one of its points, namely, from points close to the possible branching points. To do so, I clicked on the *Choose initial point* button and then selected the option 'continue' on a box that popped up:

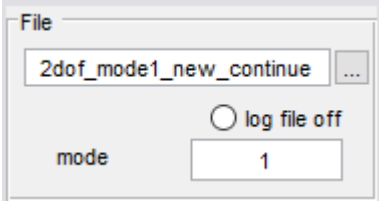




Right after selecting the point on the FEP, the continuation algorithm starts right from that point, keeping the previous points intact:



The numerical results are saved in a file automatically renamed at the beginning of the computation with a *_continue* label:



Unfortunately, it was impossible to me to force any branching from any point of the curve.

Chapter 6

CONCLUSIONS

6.1 About the analysis of the 2dofs oscillator

As seen in figures 5.6 and 5.7 there is an energy transfer along the second mode branch from the first mass to the second mass, while the fundamental frequency increases. This mode localization should slightly atenuate the motion of the first mass –the main mass– as the oscillation energy increases. In this mode of oscillation the second mass acts as a 'nonlinear energy sink' or *NES*: it absorbes a fraction of the energy of the main mass, which in the jargon of nonlinear vibration control –see for example [2]– are called 'linear oscillator' or *LO*. It is then obvious how advantageous is to add a nonlinear vibration absorber to the so-called 'linear oscillator' if the main mass or LO operates in normal conditions in a wide range of forcing frequencies including both the 'tuning' frequency and the second natural frequency, or even greater forcing frequencies. Remember that with the addition of a vibration absorber into the oscillator whose vibration one seeks to mitigate there are many parameters that can be varied to optimize the dynamical behavior –stiffness and damping coefficients, masses, nonlinear laws and so forth–. The aim is always to minimize vibrations in the interval of frequencies in which our machinery, mechanism or structure operates.

The dynamics of the first mode branch is, nevertheless, vastly different. First of all, energy transfer takes place from the second to the first mass, increasing the vibration energy of the main mass with respect to the second mass –the vibration absorber or NES– for large energies. This is an undesired effect for any NES application, whether it is vibration mitigation or energy harvesting. Secondly, it comes clear from the comparison of figures 5.15 and 5.5, and much sooner, from the single-harmonic modal branches or backbones shown on 2.8b and 2.9b that energy increase leads to much greater variations of the fundamental frequency in the second mode rather than the first. Consequently, as the mode energy increases the harmonics of the first branch, modes with order greater than one 'resonate' with the modes of the second branch. First it is the third harmonic which resonates with the NNMs of the second branch –5.17 and 5.19–, then the fifth, the seventh, and the infinity of odd-order harmonics that follow. Even resonances are branches that ramificate from bifurcation points of the main branch, but with the present algorithm they could not be computed. These internal resonances cause both masses to have dominant response frequencies multiples of the forcing frequency.

Those harmonics were completely neglected in the analytical calculation of the forced response and modal branches carried out in chapter 2 –figures 2.6, 2.7, 2.8 and 2.9– for simplicity. However, with the numerical continuation algorithm they come to light inside the modal backbones. With that technique it is possible to estimate very accurately the possible resonances that the systems can exhibit under forced response.

6.2 About the continuation algorithm

The opportunities that a numerical algorithm such as the sequential continuation method presented in chapter 4 offer over the semi-analytical harmonic balance method used in chapter 1 were proved clearly along this text. The continuation method with the shooting function has the capability to find very accurate solutions –subjected to computational accuracy and convergence factors– with any number of harmonics, direcly in the phase space of any system.

However, a few hardships have to be overcome. The most significant of them has to do with the numerical precision in the calculation of the trajectories and in the computation of the Jacobians. On the other hand, numerical precision compromises execution time, hence a compromise has to be found between the two.

The harmonic balance method is only feasible if a few harmonics of the Fourier series periodic solution are taken in consideration. Substitution of terms, rearrangement and the Galerkin integration are very dull to be handmade. Symbolic mathematics packages such as Maple are of huge help for that purpose and that particular environment has been used for this Project. The remaining nonlinear equations, which have an independent parameter –the pulsation, by convenience–, have to be solved in many complex cases with numerical methods. This is why it is a semi-analytical method.

Harmonic balance still remains as the prefered technique for the computation of the forced response curves for forced and damped systems.

6.3 About the NNMcont package

The NNMcont package has proved to be a very powerful and comprehensive tool for the computation of NNMs branches of nonlinear mechanical systems. The modelization of the nonlinearities takes advantage of the object-oriented programming paradigm, which is a feature of the Matlab language. This allows to treat each nonlinear element in the model as, literally, an object with the atributes characterizing it. However, it was unpleasant to find that some of the input options do not seem to work at all, or at least that their function in the program remains unclear. Unfortunately, this package was, in my personal opinion, underdocumented for use, but I hope this text serves to fill that gap. Another important drawback is the lack of control on the computation that it offers due to protected source codes. It can occur that certain model could not be analized because there is no numerical convergence, but the souce of the problem would remain hidden from view. An absolute control on the algorithm proves essential for the robustness of the software.

For models with a large number of degrees of freedom modal curves and time-series plots are not an ideal graphic representation of the NNMs computed. An alternative graphical output can be seen in the *BladedDisk* tutorial included in the *Examples* subfolder of the NNMcont directory. Here the displacement amplitudes of each dof when the velocities established as null phase condition are zero are plotted with black vertical bars –see *paramdisp* m-file–. For the visualization of the trajectories in each NNMs I recommend to display the Fourier discrete transform of the NNMs oscillations in each i-th dof, $x_i(t)$. That clearly singles out the amplitude of each harmonic term contained in the oscillations, and of course, the possible internal resonances that could arise.

A pending task for future releases should be to offer the posibility to compute of NNMs branches ramificating from bifurcation points. That takes the implementation of a branching algorithm as a part of the numerical continuation algorithm already implemented.

6.4 About the relevance of nonlinear modal analysis

During the course of the last decades, Mechanical Engineering has found great applications of Nonlinear Theory in its quest for improved performance and to solve the problems that may arise when more and more complex designs defy the linear behaviors and assumptions. In the field of Structural Dynamics, Nonlinear modal analysis has emerged as a set of methods that extend those from the classical linear modal analysis for the dynamical characterization of mechanical systems contain nonlinearities. It has two main branches: experimental and computational analysis, and the experience shows that the best approach to the dynamical analysis of the systems is a combination of both.

With techniques of experimental and computational modal analysis one can produce and display the free and forced response curves of the mechanical system under consideration, drawing the corresponding FRDs. Namely, the analysis of the NNMs of the system puts the stress on the free response of the system, but as described along this text they have strong implications in the forced response of the damped system. For that reason, they prove to be such a powerful tool, and their identification though either experimental or

computational means is worth considering when the linear models and hypotesis do not comply with the system behavior and properties.

For all those reasons, Nonlinear modal analysis is a field that is attracting a lot of attention from researchers, with a large body of academic literature written around it and that year after year yields new developments and discoveries. Practising engineers are also beginning to take notice of its power and take Nonlinear Theory as a tool for the design of new products and to face new engineering challenges.

6.5 Academic and professional competences put in practise

During the execution of this Project I have proved the following competences:

1. Ability to search bibliographic references, making a rigorous selection of the most useful material, skimming off the most important theory and work around the topics of interest.

2. Ability to apply the know-how and experience collected during years of college studies. This Project has required the application of matters such as Algebra, Vibration Theory, Numerical Methods, Differential Equations, Dynamical Systems, Real Analysis, Differential Geometry of Curves, Engineering Simulation and Computer Programming among others.

3. Proficency to elaborate mathematical models of theorical systems that synthesize real systems and to understand models elaborated by others in their publications.

4. Mathematical ability to understand and apply a range of applied methods of analytical or numerical calculation and computation.

5. Computational and coding skills. Competence to use computational tools to perform numerical calculation and to elaborate figures and graphs. In this Project I have

written some source code in Matlab and Maple languages before translating some of the Matlab code into C in MEX files for increased performance. Maple and Matlab are very well-known environments for symbolic and numerical computation, very popular in science and engineering.

6. Ability to arrange the topics to be covered in this text in the most rigorous, concise and clear way possible along the different chapters and setions.

7. Capacity to learn and use new tools such us the LaTeX markup text format: a widely used language for Academia writing, being specially popular among mathematicians because of the power and capabilities it provides for mathematical writing.

8. Communication and language abilities for the comprehension of the bibliography and references and the writing of this text.

9. Personal organization, perseverance, tenacity, time scheduling.

Chapter 7

FUTURE EXTENSIONS

I purpose the three following paths to extend this Project in future works:

- First and foremost, the MMNcont package is be fully complete without an implementation of a branching algorithm in a future version.
- The application of the harmonic balance method for the computation of the FRDs in 2.3 yields a series of nonlinear equations with the form $\mathbf{G}(A_1, ..., B_1, ..., \omega) = \mathbf{0}$, where ω is the independent parameter. I chose in this Project to solve the equations numerically for discrete values of ω with a Newton-Raphson algorithm, but an alternative arises when viewing the former equation as a curve in the space –whether if it is the forced response or a backbone of NNMs– with paramerer ω. That alternative is to apply a continuation algorithm to compute numerically those curves as it was done in this text with the NNMs.
- The harmonic balance method has been applied supposing a solution with one single harmonic: the fundamental harmonic. Future works could apply the method to the same models with solutions with more than one harmonic, preferably the fundamental harmonic plus odd-order harmonics –third, fifth, etc–

Chapter 8

PLANNING AND BUDGET

In this chapter the Project planning will be displayed, as well as an estimated monetary cost, with regards to the number of hours invested by the student –myself– and the advisor and the computational tools used.

8.1 Project schedule

A project schedule was one of the first tasks to be carried out as soon as the aims and scope of this Final Project had been decided at its beginning early in the month of July 2015.

The Project has been split into these five phases: *bibliographic investigation, computational coding and calculations, text writing, creation of slideshows* and *printing, recording and publication of deliverables*. Each of them has been hierarchically divided in activities and tasks, respectively. Its start date is 3 July 2015, and its deadline was scheduled on 11 March 2016.

Even though this schedule was created as the Project started it does not mean that it has remained static along its completion. The deadlines of some tasks have been redefined,

while some task have been discarded, and others were included as the circumstances required it and the Project scope changed.

For instance, first I had decided to create my own codes and programs in Matlab to implement the continuation algorithm. That was until I took notice of the Matlab package 'NNMcont', deciding instead to include the yet-existing program as a tool for the Project as it could do the job. However, in the end I decided to include my continuation program as a part of the Project deliverables, applying it to a practical case –the snapthrough absorber– and then including the codes in appendix C.

The definitive schedule of this Project can be summed up as shown in the following list:

1. **Bibliographic investigation** Academic literature and material, monographs and research papers on the topic of research were collected and skim-read, and the most relevant and practical material has been read and studied in more detail.

    - 1.1. Bibliographic investigation:
    Must start on 03/07/2015.
    Finish no later than 30/10/2015.
    Successors start-finish: 3.2.2., 4.1..
    Completion time: 150h.

2. **Computational coding and calculations** This phase gatheres all the mathematical and computational tasks and implementations of this Project.

    - 2.1. Search for models to study:
    Start no later than 10/08/2015.
    Finish no later than 21/08/2015.
    Successors start-finish: 2.2.1., 2.2.2..
    Successors start-start: 2.2.2..
    Completion time: 5h.

    2.2. Analytical problem-solving Execution of the analytical calculations required in chapter 2, plus notes taken from the academic texts prior to the computational implementations of the techniques described in chapters 2 and 3.

- 2.2.1. Establishment of the problems to solve on paper:
  Start no later than 21/08/2015.
  Finish no later than 04/09/2015.
  Predecessor start-finish: 2.1..
  Successors start-finish: 2.3.1., 2.4.1..
  Completion time: 6h.
- 2.2.2. Preliminar calculations in Maple:
  Start no later than 21/08/2015.
  Finish no later than 04/09/2015.
  Predecessor start-finish: 2.1..
  Predecessor start-start: 2.2.1..
  Successor start-finish: 2.3.1..
  Completion time: 8h.

2.3. Computational implementation in Matlab Codes for the FRDs generation in chapter 2 by the method of harmonic balance and for an original implementation of the continuation algorithm in Matlab shown in appendix C.

- 2.3.1. Matlab coding:
  Start no later than 07/09/2015.
  Finish no later than 25/09/2015.
  Predecessors start-finish: 2.2.1., 2.2.2..
  Successor start-finish: 2.3.2..
  Completion time: 20h.
- 2.3.2. Matlab code testing:
  Start no later than 28/09/2015.
  Finish no later than 16/10/2015.
  Predecessor start-finish: 2.3.1..
  Successors start-finish: 2.3.3., 2.5.1..
  Completion time: 30h.
- 2.3.3. Matlab runs and results saving:
  Start no later than 19/10/2015.
  Finish no later than 21/10/2015.

Predecessors start-finish: 2.3.2., 4.1.
Completion time: 2h.

2.4. Tests in NNMcont Modal analysis of the 2dof oscillator with the continuation algorithm in Matlab package 'NNMcont' as described in chapters 4 and 5.

- 2.4.1. NNMcont learning:
  Start no later than 19/10/2015.
  Finish no later than 21/10/2015.
  Predecessor start-finish: 2.2.1..
  Successor start-finish: 2.4.2..
  Completion time: 4h.
- 2.4.2. Computational calculations in NNMcont:
  Start no later than 22/10/2015.
  Finish no later than 27/10/2015.
  Predecessor start-finish: 2.4.1..
  Successor start-finish: 4.1..
  Completion time: 5h.

2.5. Snapthrough absorber analysis by continuation in Matlab Analysis of a snaptrough vibration absorber with my own continuation codes, described in appendix C.

- 2.5.1. Snapthrough preliminar analytical calculations in Maple:
  Start no later than 21/12/2015.
  Finish no later than 08/01/2016.
  Predecessor start-finish: 2.3.2..
  Successor start-finish: 2.5.2..
  Completion time: 8h.
- 2.5.2. Snapthrough Matlab calculations:
  Start no later than 11/01/2016.
  Finish no later than 31/01/2016.
  Predecessor start-finish: 2.5.1..

Successors start-finish: 2.3.3., 4.1..
Completion time: 25h.

3. Text writing All tasks of this phase aim for the writing of the Project text.

3.1. Familiarization with the LaTeX language The LaTeX hypertext format is the language in which this text has been composed. It required a learning and aquaintance period in which all the necessary tools were set up for the writing process.

- 3.1.1. LaTeX learning:
  Start no later than 12/10/2015.
  Finish no later than 30/10/2015.
  Successor start-finish: 3.2.2..
  Completion time: 15h.
- 3.1.2. Search for a LaTeX template:
  Start no later than 19/10/2015.
  Finish no later than 21/10/2015.
  Successor start-finish: 3.1.3..
  Completion time: 2h.
- 3.1.3. Preliminar adaptations of the LaTeX template:
  Start no later than 22/10/2015.
  Finish no later than 30/10/2015.
  Predecessor start-finish: 3.1.2..
  Successor start-finish: 3.2.2..
  Completion time: 5h.

3.2. Text writing This activity comprised the whole composition of this text.

- 3.2.1. Organization of the text structure and its scope:
  Start no later than 12/10/2015.
  Finish no later than 30/10/2015.
  Successor start-finish: 3.2.2..
  Completion time: 12h.

- 3.2.2. Text writing:
  Start no later than 02/11/2015.
  Finish no later than 19/02/2016.
  Predecessors start-finish: 1.1., 3.1.1., 3.2.1., 3.1.3..
  Successor start-finish: 5.1..
  Successor finish-finish: 4.2..
  Completion time: 200h.

4. **Creation of slideshows** Slideshows summarize the whole Project and allow for future expositions. The Project defence is a compulsory requirement and demands a slideshow presentation.

   - 4.1. Long slideshow for conferences and speeches:
     Start no later than 02/02/2016.
     Finish no later than 12/02/2016.
     Predecessors start-finish: 1.1., 2.3.3., 2.4.2., 2.5.2..
     Successor start-finish: 4.2..
     Completion time: 25h.
   - 4.2. Short slideshow for the Project defence:
     Start no later than 15/02/2016.
     Finish no later than 19/02/2016.
     Predecessor start-finish: 4.1..
     Predecessor finish-finish: 3.2.2..
     Successor start-finish: 5.1..
     Completion time: 8h.

5. **Printing, recording and publication of deliverables** Copies of this text must also be printed and delivered to the Project defence tribunal along with the codes and programs made. Text, slideshows and codes will also be made available for download from an open source site.

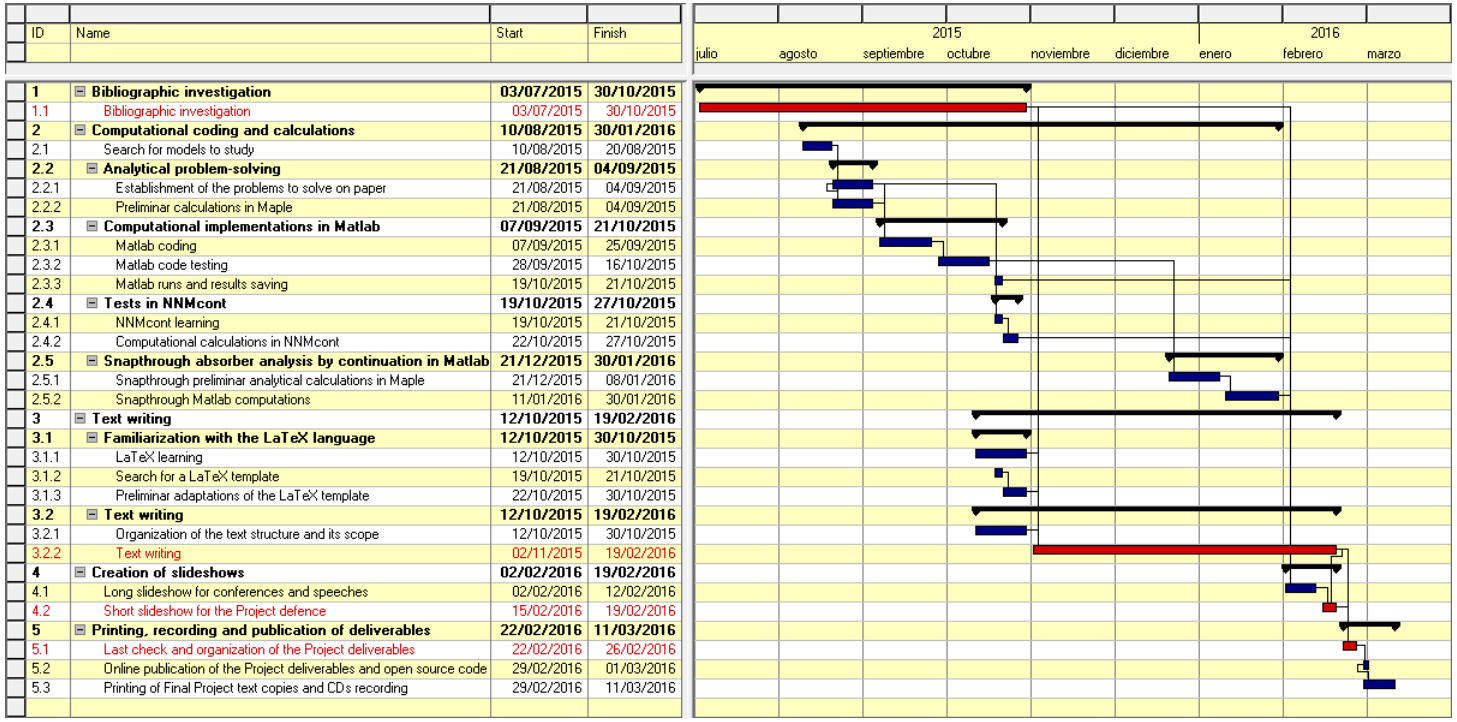
   - 5.1. Last check and organization of the Project deliverables:
     Start no later than 22/02/2016.
     Finish no later than 26/02/2016.

Predecessors start-finish: 3.2.2., 4.2..
Successor start-finish: 5.5..
Completion time: 10h.

- 5.2. Online publication of the Proyect deliverables and open source codes:
Start no later than 22/02/2016.
Finish no later than 26/02/2016.
Predecessor start-finish: 5.1..
Successor start-start: 5.3..
Completion time: 2h.
- 5.3. Printing of Final Project text copies and CDs recording:
Start no later than 22/02/2016.
Finish no later than 11/03/2016.
Predecessor start-start: 5.2..
Completion time: 4h.

In total, the time invested in this Project was 546 hours in 236 working days, out of which 536 hours required computational assistance.

This schedule was built with a free Project Management package called 'Open Workbench'.



ID	Name	Start	Finish
1	**Bibliographic investigation**	**03/07/2015**	**30/10/2015**
1.1	Bibliographic investigation	03/07/2015	30/10/2015
2	**Computational coding and calculations**	**10/08/2015**	**30/01/2016**
2.1	Search for models to study	10/08/2015	20/08/2015
2.2	**Analytical problem-solving**	**21/08/2015**	**04/09/2015**
2.2.1	Establishment of the problems to solve on paper	21/08/2015	04/09/2015
2.2.2	Preliminar calculations in Maple	21/08/2015	04/09/2015
2.3	**Computational implementations in Matlab**	**07/09/2015**	**21/10/2015**
2.3.1	Matlab coding	07/09/2015	25/09/2015
2.3.2	Matlab code testing	28/09/2015	16/10/2015
2.3.3	Matlab runs and results saving	19/10/2015	21/10/2015
2.4	**Tests in NNMcont**	**19/10/2015**	**27/10/2015**
2.4.1	NNMcont learning	19/10/2015	21/10/2015
2.4.2	Computational calculations in NNMcont	22/10/2015	27/10/2015
2.5	**Snapthrough absorber analysis by continuation in Matlab**	**21/12/2015**	**30/01/2016**
2.5.1	Snapthrough preliminar analytical calculations in Maple	21/12/2015	08/01/2016
2.5.2	Snapthrough Matlab computations	11/01/2016	30/01/2016
3	**Text writing**	**12/10/2015**	**19/02/2016**
3.1	**Familiarization with the LaTeX language**	**12/10/2015**	**30/10/2015**
3.1.1	LaTeX learning	12/10/2015	30/10/2015
3.1.2	Search for a LaTeX template	19/10/2015	21/10/2015
3.1.3	Preliminar adaptations of the LaTeX template	22/10/2015	30/10/2015
3.2	**Text writing**	**12/10/2015**	**19/02/2016**
3.2.1	Organization of the text structure and its scope	12/10/2015	30/10/2015
3.2.2	Text writing	02/11/2015	19/02/2016
4	**Creation of slideshows**	**02/02/2016**	**19/02/2016**
4.1	Long slideshow for conferences and speeches	02/02/2016	12/02/2016
4.2	Short slideshow for the Project defence	15/02/2016	19/02/2016
5	**Printing, recording and publication of deliverables**	**22/02/2016**	**11/03/2016**
5.1	Last check and organization of the Project deliverables	22/02/2016	26/02/2016
5.2	Online publication of the Project deliverables and open source code	29/02/2016	01/03/2016
5.3	Printing of Final Project text copies and CDs recording	29/02/2016	11/03/2016

FIGURE 8.1: Planning and Gantt chart. Tasks in red are those early start and finish dates match the late start and finish dates, therefore determining a critical path that in this particular case does not extend through the whole project timespan. Connected –preceding and successing– tasks are linked with black lines.

8.2 Project total estimated cost

The total cost of this project can be estimated as the sum of these four components:

$$C_{TOT} = C_{energy} + C_{advisor} + C_{student} + C_{amortization}$$

The energy cost quantifies the cost of the electrical energy demanded by my PC during the execution of this Project.

As stated above, the total number of hours invested with the need of computational assistance was 536. The power comsumption of my computer, a small HP Pavilion laptop, peaks 45W, and the average electric energy household price in Spain has averaged 0,1275 euro. With this information, an estimate of the energy cost can be calculated:

$$C_{energy} = computer\ use\ time \times computer\ power\ consumption$$
$$\times\ electric\ energy\ price = 536\ h \times \frac{45\ W}{1000\ W/kW} \times 0,1275\ €/kWh = 3,08\ €$$

The advisor cost considers the number of meetings –12– taking place between July 2015 and mid February 2016, plus three more estimated meetings until 11 March 2016; and the personal, solitary hours that my advisor dedicated to track my Project, whose number is estimated in 20. Each meeting has lasted an average of one hour and the advisor hourly wage is rounded up to 25 euro –college lecturer and researcher–, hence:

$$C_{advisor} = (meeting\ hours + advisor\ work\ hours) \times advisor\ hourly\ wage$$
$$= (1\ h/meeting \times 15\ meetings + 20\ h) \times 25\ €/h = 875\ €$$

For the calculation of the student cost I have assessed my own working time as 9 euro per hour:

$$C_{student} = (meeting\ hours + student\ work\ hours) \times student\ hourly\ wage$$
$$= (1\ h/meeting \times 15\ meetings + 546\ h) \times 9\ €/h = 5049\ €$$

Finally, the amortization cost quantifies the devaluation of my personal computer along the realization of the Project. Its purchase price was 399 euro, and its lifespan is estimated as 4 years with an usage of 8 hours a day. One out of four years is a leap year –has an addictional day–:

$$C_{amortization} = computer\ amortization\ cost$$
$$= computer\ price \times \frac{computer\ use\ hours}{computer\ lifespan}$$
$$= 399\ € \times \frac{536\ h}{(4\ years \times 365\ days/year + 1\ additional\ day\ in\ leap\ year) \times 8\ h/day} = 18,30\ €$$

The total cost can be compounded as the sum of each subcost:

$$C_{TOT} = 3,08\ € + 875\ € + 5049\ € + 18,30\ € = 5945,38\ €$$

As a final note, the Matlab environment was used with an academic license own by the Technical University of Madrid, and the remaining packages needed –NNMcont, Open Workforce– are free software. Therefore there is no software cost.

Appendix A

Matlab m-files

This appendix contains the m-files written for this Project. Some of them are isolated m-functions that NNMcont require, while others are part of a global Matlab package. In either case, the purpose and structure of each file is described, along with instructions for any future use.

All codes have been written, tested and executed in the Matlab 2015a release, with Windows 10 as the operating system under a x64, 64-bit architecture with 4GB of RAM.

As a remainder for all the appendices, all these codes –Matlab and Maple codes– are included in the CD-R deliveed with each copy of the text. They are available too for free download from GitHub in this page:

https://github.com/asilvaber/Final-Project-Alejandro-Silva/

All codes included are open source. They can be freely used and modified by any interested person for any purpose. But for courtesy, I would feel appreciated if I was contacted before anyone were to modify or to use these codes for a purpose different from the original one, just to let me know. My email address is asilvaber@gmail.com

A.1 Codes for the plotting of the FRDs

The Matlab codes shown in this section allow us to solve the algebraic equations derived with the harmonic balance method in 2.2.2, 2.3.1.1 and 2.3.1.2. The aim is to depict the frequency response diagrams and modal curves of the SDOF and 2dofs linear and nonlinear oscillators described in 2.1.1 and 2.1.2. The graphical outputs were shown and described in 2.3.2.

Some of these files –the m-functions with the Newton-Raphson loops to solve the algebraic equations– have been converted to C-MEX files with a Matlab application called *coder*. There is the option of choosing to use either the Matlab original function or the MEX function. The numerical results remain unchanged, but the execution times can be more than ten times faster with the MEX files.

A.1.1 FRC1dofmain.m

This is the executable m-file that returns the graphical outputs with the frequency responses –both the oscillation amplitude and the phase delay– and modal curves of the single degree of freedom, linear and nonlinear –Duffing nonlinearity– oscillator. Examples of outputs for the default parameter values set in the parameter definitions of this m-file are 2.6 and 2.7.

```
1  %-------------------------------------------------------------------------
2  % THIS IS THE EXECUTABLE M-FILE FOR THE COMPUTATION OF THE FRD PLOTS OF THE
3  % SINGLE DEGREE OF FREEDOM NONLINEAR OSCILLATOR
4
5  % WRITTEN BY ALEJANDRO SILVA, asilvaber@gmail.com
6
7  % OCTOBER 2015, MADRID, SPAIN
8  %-------------------------------------------------------------------------
9
10 clear
11 clc
12
```

```
13 % Convergence factor for the Newton-Raphson loops, and numerical precision
14 % factor for the plotting
15 tol = 1e-6;
16
17 % Mass of the oscillator
18 m = 1;
19 % Damping coefficient
20 %c = tol;
21 c = 0.25;
22 % Stiffness coefficient
23 k = 1;
24
25 % Nonlinear stiffness coefficient
26 %beta = 0;
27 %beta = 0.1;
28 beta = 1;
29
30 % Forcing amplitude
31 fc = 1;
32
33 % Plotting options
34 % Frequency intervals, minimum and maximum frequency
35 fMin = 0;
36 fMax = 1/2;
37 % Number of discrete values taken from the frequency plotting interval for
38 % the FRD plotting
39 numPulsSamples = 200;
40 % Number of points taken as initial points for the Newton-Raphson loops per
41 % frequency value
42 numAmplSamples = 200;
43 % Plot y-axis limit
44 figVertLim = 5;
45 % Maximum coefficient value taken as initial value for the Newton-Raphson
46 % loops
47 aMax = 4;
48
49 % Maximum number of iteration for the N-R loops
50 maxIter = 1000;
```

```
51
52 % Natural pulsation of the linearized oscillator
53 natPuls = sqrt(eigs(k,m));
54
55 % Computation of the frequency response curves and modal backbones with
56 % Matlab functions
57
58 % tic
59 % [AmplitudeFR,AamplitudeFR,BamplitudeFR,PhaseAngleFR,W] =
       NewtonRaphsonFR1dof(tol,m,c,k,beta,fc,fMin,fMax,numAmplSamples,
       numPulsSamples,aMax,maxIter);
60 % [AmplitudeNMB] = NewtonRaphsonNMB1dof(tol,m,k,beta,aMax,maxIter,W);
61 % toc
62
63 % Computation of the frequency response curves and modal backbones with
64 % C MEX-files created with the Matlab assistant 'coder'.
65
66 tic
67 [AmplitudeFR,AamplitudeFR,BamplitudeFR,PhaseAngleFR,W] =
       NewtonRaphsonFR1dof_mex(tol,m,c,k,beta,fc,fMin,fMax,numAmplSamples,
       numPulsSamples,aMax,maxIter);
68 [AmplitudeNMB] = NewtonRaphsonNMB1dof_mex(tol,m,k,beta,aMax,maxIter,W);
69 toc
70
71 % Call to the m-file that displays the curves on the FRD and the phase
72 % angle plot
73 figures1dof
74
75 % Instructions to understand the graphical output in chapter two of the
76 % project text
```

A.1.2 figures1dof.m

This file commands the production of the Matlab graphical outputs for the SDOF model given the numerical results of the Newton-Raphson loops.

```
1
```

```
% This m-file commands the plotting of the frequency response curves,
% modal backbones and phase angle of the forced response of the 1dof
% system from the numeric data contained in the arrays and matrices

% Plotting of the modal backbones from the data contained in the matrix
% AmplitudeNMB on the FRD
figure
for i=1:length(W)
    if abs(AmplitudeNMB(i))>tol && abs(AmplitudeNMB(i))<=figVertLim, plot(W(i)/(2*pi),abs(AmplitudeNMB(i)),'bo'); hold on; end
end

% Plotting of the frequency response curves from the data contained in the
% matrix AmplitudeFR on the same FRD
for i=1:length(W)
    for j=1:numAmplSamples
        if j==1,
            if abs(AmplitudeFR(j,i))>tol && abs(AmplitudeFR(j,i))<=figVertLim, plot(W(i)/(2*pi),AmplitudeFR(j,i),'r.'); hold on; end
        elseif j~=1,
            if abs(AmplitudeFR(j,i))>tol && abs(AmplitudeFR(j,i))<=figVertLim && abs(AmplitudeFR(j,i)-AmplitudeFR(j-1,i))>tol, plot(W(i)/(2*pi),AmplitudeFR(j,i),'r.'); hold on; end
        end
    end
end
for i=1:length(natPuls),
    if natPuls(i)/(2*pi)<=fMax && natPuls(i)/(2*pi)>=fMin, plot([natPuls(i)/(2*pi),natPuls(i)/(2*pi)],[0,figVertLim],'k-'); hold on; end
end
title(['M = ' num2str(m) ', K = ' num2str(k) ', C = ' num2str(c) ', \beta = ' num2str(beta) ', fc = ' num2str(fc)])
xlabel('frequency (Hz)')
ylabel('oscillation amplitude (m)')

% Plotting of the phase angle curves of the frequency response from the
% data contained in the matrix PhaseangleFR on a new diagram
figure
```

```
for i=1:length(W)
    for j=1:numAmplSamples
        if j==1,
            if abs(PhaseAngleFR(j,i))>tol, plot(W(i)/(2*pi),PhaseAngleFR(j,i),'r.'); hold on; end
        elseif j~=1,
            if abs(PhaseAngleFR(j,i))>tol && abs(PhaseAngleFR(j,i)-PhaseAngleFR(j-1,i))>tol, plot(W(i)/(2*pi),PhaseAngleFR(j,i),'r.'); hold on; end
        end
    end
end
for i=1:length(natPuls),
    if natPuls(i)/(2*pi)<=fMax && natPuls(i)/(2*pi)>=fMin, plot([natPuls(i)/(2*pi),natPuls(i)/(2*pi)],[0,-pi],'k-'); hold on; end
end
title(['M = ' num2str(m) ', K = ' num2str(k) ', C = ' num2str(c) ', \beta = ' num2str(beta) ', fc = ' num2str(fc)])
xlabel('frequency (Hz)')
ylabel('phase angle (rad)')
```

A.1.3 NewtonRaphsonFR1dof.m

This function contains the loops that compute the solutions of the algebraic equations for the frequency response of the SDOF model –2.3.1.2–. This is one of the functions converted to MEX-files.

```
function [AmplitudeFR,AamplitudeFR,BamplitudeFR,PhaseAngleFR,W] = NewtonRaphsonFR1dof(tol,m,c,k,beta,fc,fMin,fMax,numAmplSamples,numPulsSamples,aMax,maxIter)

% This function calculates the output for the frequency response and phase
% angle curves of the 1dof oscillator

% Array with the pulsation values taken to compute the curves
W = fMin*2*pi:((fMax-fMin)/(numPulsSamples-1))*2*pi:fMax*2*pi;
% Matrix with the oscillation amplitudes computed with N-R
```

```
AmplitudeFR = zeros(numAmplSamples,length(W));
% Matrix with the A coefficients of the responses, calculated with N-R
AamplitudeFR = zeros(numAmplSamples,length(W));
% Matrix with the B coefficients of the responses, calculated with N-R
BamplitudeFR = zeros(numAmplSamples,length(W));
% Matrix with the phase angles of the responses
PhaseAngleFR = zeros(numAmplSamples,length(W));

% For each pulsation and initial amplitude values N-R is applied to solve
% the equation G(A,B,w)=0
for i=1:length(W)
    for j=1:numAmplSamples
        % n: number of iterations
        n = 1;
        % The initial value of the amplitude for the N-R loop depends on
        % aMax and on the j-th iteration index
        amplitude = [j*aMax/numAmplSamples; j*aMax/numAmplSamples];
        % Initial evaluation of the equation G(amplitude,W(i))
        F = FReq1dof(m,c,k,beta,fc,W(i),amplitude(1),amplitude(2));
        % NEWTON-RAPHSON LOOP
        while norm(F)>tol && n<=maxIter
            F = FReq1dof(m,c,k,beta,fc,W(i),amplitude(1),amplitude(2));
            J = FRjac1dof(m,c,k,beta,W(i),amplitude(1),amplitude(2));
            amplitude = amplitude - J\F;
            n = n+1;
        end

        % The solutions computed in the array amplitude are assigned to the
        % output matrices AamplitudeFR, BamplitudeFR, AmplitudeFR and
        % PhaseAngleFR
        if n<=maxIter,
            AamplitudeFR(j,i) = amplitude(1);
            BamplitudeFR(j,i) = amplitude(2);
            % AmplitudeFR contains the mass oscillation amplitudes,
            % which are calculated from the harmonic response coefficients,
            % applying Pythagoras
            AmplitudeFR(j,i) = sqrt(amplitude(2)^2+amplitude(1)^2);
            PhaseAngleFR(j,i) = atan(abs(amplitude(2)/amplitude(1)));
```

```
47              if amplitude(1)>0, PhaseAngleFR(j,i) = -PhaseAngleFR(j,i);
48              elseif amplitude(1)<0, PhaseAngleFR(j,i) = PhaseAngleFR(j,i) -
    pi;
49              end
50          end
51      end
52  end
53
54  end
```

A.1.4 FReq1dof.m

This m-file returns the value of the function $\mathbf{G}(A, B, \omega)$, where $\mathbf{G}(A, B, \omega) = \mathbf{0}$ is the system of equations shown in 2.40 for the frequency response of the SDOF system. It is called by the Newton-Raphson loop in *NewtonRaphsonFR1dof.m*.

```
1  function F = FReq1dof(m,c,k,beta,fc,w,A,B)
2
3  % This function returns the value of the 1dof algebraic system
4  % G(A,B,w), represented here by double array F, for the forced response
5  % curve
6
7  F = [4*k*A-4*m*A*w^2-4*c*B*w+3*beta*A*B^2-4*fc+3*beta*A^3;
8       3*beta*A^2*B-4*w^2*m*B+4*k*B+3*beta*B^3+4*c*A*w];
9
10 end
```

A.1.5 FRjac1dof.m

This m-function returns the jacobian matrix of the frequency response equations, $\mathbf{G}(A, B, \omega) = \mathbf{0}$, of the SDOF model, as defined in 2.42, for its use in the Newton-Raphson algorithm of *NewtonRaphsonFR1dof.m*.

```
1  function J = FRjac1dof(m,c,k,beta,w,A,B)
2
```

```
% This function returns the jacobian matrix J of the 1dof algebraic system
% for the forced response curve

J = [4*k-4*m*w^2+3*beta*B^2+9*beta*A^2, -4*c*w+6*beta*A*B;
     6*beta*A*B+4*c*w, 3*beta*A^2-4*m*w^2+4*k+9*beta*B^2];

end
```

A.1.6 NewtonRaphsonNMB1dof.m

This function contains the loops that compute the solutions of the algebraic equation for the modal curve or 'backbone' of the SDOF model. This is another of the functions converted to MEX-files.

```
function [AmplitudeNMB] = NewtonRaphsonNMB1dof(tol,m,k,beta,aMax,maxIter,W)

% This function calculates the output for the modal backbone curve of the
% 1dof oscillator

% Array with the oscillation amplitudes of the mass computed with N-R
AmplitudeNMB = zeros(1,length(W));

% For each pulsation and value N-R is applied to solve the equation
% G(amplitude,w)=0
for i=1:length(W)
    % n: number of iterations
    n = 1;
    % The initial value of the amplitude for the N-R loop is aMax
    amplitude = aMax;
    % Initial evaluation of the equation G(amplitude,W(i))
    F = NMBeq1dof(m,k,beta,W(i),amplitude);
    % NEWTON-RAPHSON LOOP
    while norm(F)>tol && n<=maxIter
        F = NMBeq1dof(m,k,beta,W(i),amplitude);
        J = NMBjac1dof(m,k,beta,W(i),amplitude);
        amplitude = amplitude - J\F;
        n = n+1;
```

```
24      end
25
26      % The solution computed, amplitude, is assigned to the output matrix:
27      % AmplitudeNMB
28      if n<=maxIter,
29          AmplitudeNMB(i) = amplitude;
30      end
31  end
32
33  end
```

A.1.7 NMBeq1dof.m

This m-file returns the value of the function $\mathbf{G}(A, \omega)$, where $\mathbf{G}(A, \omega) = 0$ is the equation depicted in 2.47 for the modal curve of the SDOF system. It is called by the Newton-Raphson loop in *NewtonRaphsonNMB1dof.m*.

```
1  function F = NMBeq1dof(m,k,beta,w,amplitude)
2
3  % This function returns the value of the 1dof algebraic equation
4  % G(amplitude,w), represented here by double scalar F, for the modal
5  % backbone curve
6
7  F = 4*k*amplitude-4*m*amplitude*w^2+3*beta*amplitude^3;
8
9  end
```

A.1.8 NMBjac1dof.m

This m-function returns the derivative of the modal curve equation, $\mathbf{G}(A, \omega) = 0$, with respect to A, of the SDOF model, for its use in the Newton-Raphson algorithm of *NewtonRaphsonNMB1dof.m*.

```
1  function J = NMBjac1dof(m,k,beta,w,amplitude)
2
```

```
% This function returns the derivative J of the 1dof algebraic equation
% for the modal backbone curve

J = 4*k-4*m*w^2+9*beta*amplitude^2;

end
```

A.1.9 FRC2dofmain.m

This is the executable m-file returning the graphical outputs with the frequency responses and modal curves of the two degrees of freedom linear and nonlinear oscillator. Examples of outputs for the default parameter values set in the parameter definitions of this m-file are 2.8 and 2.9.

```
%-------------------------------------------------------------------------
% THIS IS THE EXECUTABLE M-FILE FOR THE COMPUTATION OF THE FRD PLOTS OF THE
% TWO DEGREES OF FREEDOM NONLINEAR OSCILLATOR

% WRITTEN BY ALEJANDRO SILVA, asilvaber@gmail.com

% OCTOBER 2015, MADRID, SPAIN
%-------------------------------------------------------------------------

clear
clc

% Convergence factor for the Newton-Raphson loops, and numerical precision
% factor for the plotting
tol = 1e-6;

% Masses of the oscillator
m1 = 1;
m2 = 1;
% Damping coefficients
%c1 = tol;
c1 = 0.25;
```

```
23 c2 = tol;
24 % Stiffness coefficient
25 k1 = 1;
26 k2 = 1;
27
28 % Nonlinear stiffness coefficient
29 %beta = 0;
30 beta = 0.1;
31 %beta = 1;
32
33 % Forcing amplitude coefficients
34 fc11 = 1;
35 fc12 = 0;
36 fc21 = 0;
37 fc22 = 0;
38
39 % Plotting options
40 % Frequency intervals, minimum and maximum frequency
41 fMin = 0;
42 fMax = 1/2;
43 % Number of discrete values taken from the frequency plotting interval for
44 % the FRD plotting
45 numPulsSamples = 200;
46 % Number of points taken as initial points for the Newton-Raphson loops per
47 % frequency value
48 numAmplSamples = 200;
49 % Plot y-axis limit
50 figVertLim = 10;
51 % Maximum coefficient value taken as initial value for the Newton-Raphson
52 % loops
53 aMax = 3;
54
55 % Maximum number of iteration for the N-R loops
56 maxIter = 1000;
57
58 % Natural pulsation of the linearized oscillator
59 natPuls = sqrt(eigs([k1+k2 -k2; -k2 k2],[m1 0; 0 m2]));
60
```

```
% Computation of the frequency response curves and modal backbones with
% Matlab functions

% tic
% [Amplitude1FR, Amplitude2FR, A1amplitudeFR, A2amplitudeFR, B1amplitudeFR,
    B2amplitudeFR,W] = NewtonRaphsonFR2dof(tol,m1,m2,c1,c2,k1,k2,beta,fc11,
    fc12,fc21,fc22,fMin,fMax,numAmplSamples,numPulsSamples,aMax,maxIter);
% [Amplitude1NMB, Amplitude2NMB] = NewtonRaphsonNMB2dof(tol,m1,m2,k1,k2,beta
    ,aMax,maxIter,W);
% toc

% Computation of the frequency response curves and modal backbones with
% C MEX-files created with the Matlab assistant 'coder'.

tic
[Amplitude1FR, Amplitude2FR, A1amplitudeFR, A2amplitudeFR, B1amplitudeFR,
    B2amplitudeFR,W] = NewtonRaphsonFR2dof_mex(tol,m1,m2,c1,c2,k1,k2,beta,
    fc11,fc12,fc21,fc22,fMin,fMax,numAmplSamples,numPulsSamples,aMax,maxIter
    );
[Amplitude1NMB, Amplitude2NMB] = NewtonRaphsonNMB2dof_mex(tol,m1,m2,k1,k2,
    beta,aMax,maxIter,W);
toc

% Call to the m-file that displays the curves on the FRDs
figures2dof

% Instructions to understand the graphical output in chapter two of the
% project text
```

A.1.10 figures2dof.m

This file commands the production of the Matlab graphical outputs for the 2dofs model given the numerical results of the Newton-Raphson loops.

```

% This m-file commands the plotting of the frequency response curves,
% modal backbones and phase angle of the forced response of the 2dofs
```

```
% system from the numeric data contained in the arrays and matrices

% Plotting of the modal backbones of the oscillation of the first mass
% from the data contained in the array Amplitude1NMB and on the FRD
figure
for i=1:length(W)
    if abs(Amplitude1NMB(i))>tol && abs(Amplitude1NMB(i))<=figVertLim,
        if Amplitude1NMB(i)*Amplitude2NMB(i)<0, plot(W(i)/(2*pi),abs(Amplitude1NMB(i)),'bx'); hold on;
        elseif Amplitude1NMB(i)*Amplitude2NMB(i)>0, plot(W(i)/(2*pi),abs(Amplitude1NMB(i)),'bo'); hold on; end
    end
end

% Plotting of the frequency response curves of the oscillation of the
% first mass from the data contained in the matrices A1amplitudeFR,
% A2amplitudeFR, B1amplitudeFR, B2amplitudeFR and Amplitude1FR on
% the same FRD
for i=1:length(W)
    for j=1:numAmplSamples
        if j==1,
            if abs(Amplitude1FR(j,i))>tol && abs(Amplitude1FR(j,i))<=figVertLim,
                if A1amplitudeFR(j,i)>0 && B1amplitudeFR(j,i)<=0, plot(W(i)/(2*pi),Amplitude1FR(j,i),'r.'); hold on;
                elseif A1amplitudeFR(j,i)<=0 && B1amplitudeFR(j,i)<0, plot(W(i)/(2*pi),Amplitude1FR(j,i),'m.'); hold on;
                elseif A1amplitudeFR(j,i)<0 && B1amplitudeFR(j,i)>=0, plot(W(i)/(2*pi)/(2*pi),Amplitude1FR(j,i),'g.'); hold on;
                elseif A1amplitudeFR(j,i)>=0 && B1amplitudeFR(j,i)>0, plot(W(i)/(2*pi)/(2*pi),Amplitude1FR(j,i),'c.'); hold on;
                end
            end
        elseif j~=1,
            if abs(Amplitude1FR(j,i))>tol && abs(Amplitude1FR(j,i))<=figVertLim && abs(Amplitude1FR(j,i)-Amplitude1FR(j-1,i))>tol,
                if A1amplitudeFR(j,i)>0 && B1amplitudeFR(j,i)<=0, plot(W(i)/(2*pi),Amplitude1FR(j,i),'r.'); hold on;
```

```
33                  elseif A1amplitudeFR(j,i)<=0 && B1amplitudeFR(j,i)<0, plot(W(i)/(2*pi),Amplitude1FR(j,i),'m.'); hold on;
34                  elseif A1amplitudeFR(j,i)<0 && B1amplitudeFR(j,i)>=0, plot(W(i)/(2*pi),Amplitude1FR(j,i),'g.'); hold on;
35                  elseif A1amplitudeFR(j,i)>=0 && B1amplitudeFR(j,i)>0, plot(W(i)/(2*pi),Amplitude1FR(j,i),'c.'); hold on;
36                  end
37              end
38          end
39      end
40  end
41  for i=1:length(natPuls),
42      if natPuls(i)/(2*pi)<=fMax && natPuls(i)/(2*pi)>=fMin, plot([natPuls(i)/(2*pi),natPuls(i)/(2*pi)],[0,figVertLim],'k-'); hold on; end
43  end
44  title(['M1 = ' num2str(m1) ', K1 = ' num2str(k1) ', C1 = ' num2str(c1) ', M2 = ' num2str(m2) ', K2 = ' num2str(k2) ', C2 = ' num2str(c2) ', \beta = ' num2str(beta) ', fc11 = ' num2str(fc11) ', fc12 = ' num2str(fc12) ', fc21 = ' num2str(fc21) ', fc22 = ' num2str(fc22) '])
45  xlabel('frequency dof 1 (Hz)')
46  ylabel('oscillation amplitude dof 1 (m)')
47
48  % Plotting of the modal backbones of the oscillation of the second mass
49  % from the data contained in the array Amplitude2NMB on a new FRD
50  figure
51  for i=1:length(W)
52      if abs(Amplitude2NMB(i))>tol && abs(Amplitude2NMB(i))<=figVertLim,
53          if Amplitude1NMB(i)*Amplitude2NMB(i)>0, plot(W(i)/(2*pi),abs(Amplitude2NMB(i)),'bo'); hold on;
54          elseif Amplitude1NMB(i)*Amplitude2NMB(i)<0, plot(W(i)/(2*pi),abs(Amplitude2NMB(i)),'bx'); hold on; end
55      end
56  end
57
58  % Plotting of the frequency response curves of the oscillation of the
59  % second mass from the data contained in the matrices A1amplitudeFR,
60  % A2amplitudeFR, B1amplitudeFR, B2amplitudeFR and Amplitude2FR on
61  % the same FRD
```

```
62 for i=1:length(W)
63     for j=1:numAmplSamples
64         if j==1,
65             if abs(Amplitude2FR(j,i))>tol && abs(Amplitude2FR(j,i))<=
    figVertLim,
66                 if A2amplitudeFR(j,i)>0 && B2amplitudeFR(j,i)<=0, plot(W(i)
    /(2*pi),Amplitude2FR(j,i),'r.'); hold on;
67                 elseif A2amplitudeFR(j,i)<=0 && B2amplitudeFR(j,i)<0, plot(
    W(i)/(2*pi),Amplitude2FR(j,i),'m.'); hold on;
68                 elseif A2amplitudeFR(j,i)<0 && B2amplitudeFR(j,i)>=0, plot(
    W(i)/(2*pi),Amplitude2FR(j,i),'g.'); hold on;
69                 elseif A2amplitudeFR(j,i)>=0 && B2amplitudeFR(j,i)>0, plot(
    W(i)/(2*pi),Amplitude2FR(j,i),'c.'); hold on;
70                 end
71             end
72         elseif j~=1,
73             if abs(Amplitude2FR(j,i))>tol && abs(Amplitude2FR(j,i))<=
    figVertLim && abs(Amplitude2FR(j,i)-Amplitude2FR(j-1,i))>tol,
74                 if A2amplitudeFR(j,i)>0 && B2amplitudeFR(j,i)<=0, plot(W(i)
    /(2*pi),Amplitude2FR(j,i),'r.'); hold on;
75                 elseif A2amplitudeFR(j,i)<=0 && B2amplitudeFR(j,i)<0, plot(
    W(i)/(2*pi),Amplitude2FR(j,i),'m.'); hold on;
76                 elseif A2amplitudeFR(j,i)<0 && B2amplitudeFR(j,i)>=0, plot(
    W(i)/(2*pi),Amplitude2FR(j,i),'g.'); hold on;
77                 elseif A2amplitudeFR(j,i)>=0 && B2amplitudeFR(j,i)>0, plot(
    W(i)/(2*pi),Amplitude2FR(j,i),'c.'); hold on;
78                 end
79             end
80         end
81     end
82 end
83 for i=1:length(natPuls),
84     if natPuls(i)/(2*pi)<=fMax && natPuls(i)/(2*pi)>=fMin, plot([natPuls(i)
    /(2*pi),natPuls(i)/(2*pi)],[0,figVertLim],'k-'); hold on; end
85 end
```

```
86 title(['M1 = ' num2str(m1) ', K1 = ' num2str(k1) ', C1 = ' num2str(c1) ',
       M2 = ' num2str(m2) ', K2 = ' num2str(k2) ', C2 = ' num2str(c2) ', \beta
       = ' num2str(beta) ', fc11 = ' num2str(fc11) ', fc12 = ' num2str(fc12) ',
        fc21 = ' num2str(fc21) ', fc22 = ' num2str(fc22) '])
87 xlabel('frequency dof 2 (Hz)')
88 ylabel('oscillation amplitude dof 2 (m)')
```

A.1.11 NewtonRaphsonFR2dof.m

This function contains the loops that compute the solutions of the algebraic equations for the frequency response of the 2dofs model –2.40–. Another function that has been converted to MEX-files.

```
1  function [Amplitude1FR,Amplitude2FR,A1amplitudeFR,A2amplitudeFR,
       B1amplitudeFR,B2amplitudeFR,W] = NewtonRaphsonFR2dof(tol,m1,m2,c1,c2,k1,
       k2,beta,fc11,fc12,fc21,fc22,fMin,fMax,numAmplSamples,numPulsSamples,aMax
       ,maxIter)
2
3  % This function calculates the output for the frequency response of the
4  % 2dof oscillator
5
6  % Array with the pulsation values taken to compute the curves
7  W = fMin*2*pi:((fMax-fMin)/(numPulsSamples-1))*2*pi:fMax*2*pi;
8  % Matrix with the oscillation amplitudes of the first mass computed with
9  % N-R
10 Amplitude1FR = zeros(numAmplSamples,length(W));
11 % Matrix with the oscillation amplitudes of the second mass computed with
12 % N-R
13 Amplitude2FR = zeros(numAmplSamples,length(W));
14 % Matrix with the A1 coefficients of the responses, calculated with N-R
15 A1amplitudeFR = zeros(numAmplSamples,length(W));
16 % Matrix with the B1 coefficients of the responses, calculated with N-R
17 B1amplitudeFR = zeros(numAmplSamples,length(W));
18 % Matrix with the A2 coefficients of the responses, calculated with N-R
19 A2amplitudeFR = zeros(numAmplSamples,length(W));
20 % Matrix with the B2 coefficients of the responses, calculated with N-R
21 B2amplitudeFR = zeros(numAmplSamples,length(W));
```

```
22
23 % For each pulsation and initial amplitude values N-R is applied to solve
24 % the equation G(A1,B1,A2,B2,w)=0
25 for i=1:length(W)
26     for j=1:numAmplSamples
27         % n: number of iterations
28         n = 1;
29         % The initial values of the amplitudes for the N-R loop depend on
30         % aMax and on the j-th iteration index
31         amplitude1 = [j*aMax/numAmplSamples; -j*aMax/numAmplSamples];
32         amplitude2 = [-j*aMax/numAmplSamples; j*aMax/numAmplSamples];
33         % Initial evaluation of the equation G(amplitude,W(i))
34         F = FReq2dof(m1,m2,c1,c2,k1,k2,beta,fc11,fc12,fc21,fc22,W(i),
    amplitude1(1),amplitude1(2),amplitude2(1),amplitude2(2));
35         % NEWTON-RAPHSON LOOP
36         while norm(F)>tol && n<=maxIter
37             F = FReq2dof(m1,m2,c1,c2,k1,k2,beta,fc11,fc12,fc21,fc22,W(i),
    amplitude1(1),amplitude1(2),amplitude2(1),amplitude2(2));
38             J = FRjac2dof(m1,m2,c1,c2,k1,k2,beta,W(i),amplitude1(1),
    amplitude1(2),amplitude2(1),amplitude2(2));
39             delta = -J\F;
40             amplitude1 = amplitude1 + delta(1:2);
41             amplitude2 = amplitude2 + delta(3:4);
42             n = n+1;
43         end
44
45         % The solutions computed in the arrays amplitude1 and amplitude2
46         % are assigned to the output matrices A1amplitudeFR, A2amplitudeFR,
47         % B1amplitudeFR, B2amplitudeFR, Amplitude1FR and Amplitude2FR
48         if n<=maxIter,
49             A1amplitudeFR(j,i) = amplitude1(1);
50             A2amplitudeFR(j,i) = amplitude2(1);
51             B1amplitudeFR(j,i) = amplitude1(2);
52             B2amplitudeFR(j,i) = amplitude2(2);
53             Amplitude1FR(j,i) = sqrt(amplitude1(2)^2+amplitude1(1)^2);
54             Amplitude2FR(j,i) = sqrt(amplitude2(2)^2+amplitude2(1)^2);
55         end
56     end
```

```
57 end
58
59 end
```

A.1.12 FReq2dof.m

This m-file returns the value of the function $\mathbf{G}(A_1, A_2, B_1, B_2, \omega)$, where $\mathbf{G}(A_1, A_2, B_1, B_2, \omega) = \mathbf{0}$ is the system of equations shown in 2.51 for the frequency response of the 2dofs system. It is called by the Newton-Raphson loop in *NewtonRaphsonFR2dof.m*.

```
1  function F = FReq2dof(m1,m2,c1,c2,k1,k2,beta,fc11,fc12,fc21,fc22,w,A1,B1,A2
       ,B2)
2
3  % This function returns the value of the 2dofs algebraic system
4  % G(A1,B1,A2,B2,w), represented here by double array F, for the forced
       response
5  % curve
6
7  F = [-3*beta*A2*B1^2-4*fc11-3*beta*A2*B2^2+9*beta*A2^2*A1-6*beta*B2*A1*B1
       +6*beta*A2*B2*B1-4*c1*B1*w-4*c2*B1*w-4*m1*A1*w^2+4*c2*B2*w-4*k2*A2-3*
       beta*A2^3+3*beta*A1^3+4*k2*A1+4*k1*A1+3*beta*A1*B1^2-9*beta*A2*A1^2+3*
       beta*B2^2*A1;
8        -6*beta*A2*A1*B1+6*beta*A2*B2*A1+4*c1*A1*w+4*c2*A1*w-4*m1*B1*w^2-4*c2*
       A2*w+9*beta*B2^2*B1-9*beta*B2*B1^2-3*beta*A2^2*B2+3*beta*A2^2*B1-3*beta*
       B2*A1^2+3*beta*A1^2*B1+4*k2*B1+4*k1*B1+3*beta*B1^3-3*beta*B2^3-4*k2*B2
       -4*fc12;
9        -3*beta*A2*B1^2-3*beta*A2*B2^2+9*beta*A2^2*A1-6*beta*B2*A1*B1+6*beta*
       A2*B2*B1-4*c2*B1*w+4*m2*A2*w^2+4*c2*B2*w-4*k2*A2-3*beta*A2^3+3*beta*A1
       ^3+4*fc21+4*k2*A1+3*beta*A1*B1^2-9*beta*A2*A1^2+3*beta*B2^2*A1;
10       3*beta*B1^3-6*beta*A2*A1*B1+6*beta*A2*B2*A1+4*fc22-3*beta*B2^3-4*k2*B2
       +4*k2*B1-3*beta*A2^2*B2+9*beta*B2^2*B1-9*beta*B2*B1^2-4*c2*A2*w+4*m2*B2*
       w^2+4*c2*A1*w-3*beta*B2*A1^2+3*beta*A1^2*B1+3*beta*A2^2*B1];
11
12 end
```

A.1.13 FRjac2dof.m

This m-function returns the jacobian matrix of the frequency response equations, $\mathbf{G}(A_1, A_2, B_1, B_2, \omega) = \mathbf{0}$, of the 2dofs model, for its use in the Newton-Raphson algorithm of *NewtonRaphsonFR2dof.m*.

```
function J = FRjac2dof(m1,m2,c1,c2,k1,k2,beta,w,A1,B1,A2,B2)

% This function returns the jacobian matrix J of the 2dofs algebraic system
% for the forced response curve

J = [4*k1+9*beta*A1^2+3*beta*B1^2+3*beta*B2^2-6*beta*B2*B1-18*beta*A2*A1+9*
    beta*A2^2+4*k2-4*w^2*m1, 6*beta*A1*B1-6*beta*A2*B1-6*beta*B2*A1+6*beta*
    A2*B2-4*w*c2-4*w*c1, -9*beta*A2^2-4*k2-3*beta*B1^2-3*beta*B2^2+6*beta*B2
    *B1-9*beta*A1^2+18*beta*A2*A1, -6*beta*A2*B2+6*beta*B2*A1-6*beta*A1*B1
    +6*beta*A2*B1+4*w*c2;
      4*w*c1+4*w*c2-6*beta*B2*A1+6*beta*A1*B1-6*beta*A2*B1+6*beta*A2*B2, -4*
    w^2*m1+9*beta*B2^2-18*beta*B2*B1+3*beta*A2^2+3*beta*A1^2+9*beta*B1^2+4*
    k1+4*k2-6*beta*A2*A1, -4*w*c2-6*beta*A2*B2+6*beta*A2*B1-6*beta*A1*B1+6*
    beta*B2*A1, 18*beta*B2*B1-9*beta*B1^2-3*beta*A2^2-3*beta*A1^2-4*k2-9*
    beta*B2^2+6*beta*A2*A1;
      9*beta*A1^2+3*beta*B1^2+3*beta*B2^2-6*beta*B2*B1-18*beta*A2*A1+9*beta*
    A2^2+4*k2, 6*beta*A1*B1-6*beta*A2*B1-6*beta*B2*A1+6*beta*A2*B2-4*w*c2,
    -9*beta*A2^2-4*k2-3*beta*B2^2-3*beta*B1^2+6*beta*B2*B1-9*beta*A1^2+18*
    beta*A2*A1+4*w^2*m2, -6*beta*A2*B2+6*beta*B2*A1-6*beta*A1*B1+6*beta*A2*
    B1+4*w*c2;
      -6*beta*B2*A1+6*beta*A1*B1+4*w*c2-6*beta*A2*B1+6*beta*A2*B2, -18*beta*
    B2*B1+3*beta*A2^2+3*beta*A1^2+9*beta*B1^2+4*k2+9*beta*B2^2-6*beta*A2*A1,
     -6*beta*A2*B2+6*beta*A2*B1-4*w*c2-6*beta*A1*B1+6*beta*B2*A1, -9*beta*B1
    ^2-3*beta*A2^2-3*beta*A1^2-4*k2-9*beta*B2^2+18*beta*B2*B1+4*w^2*m2+6*
    beta*A2*A1];

end
```

A.1.14 NewtonRaphsonNMB2dof.m

This function contains the loops that compute the solutions of the algebraic equations for the modal curves or 'backbones' of the 2dofs model. This function has also been converted to MEX-files.

```
function [Amplitude1NMB,Amplitude2NMB] = NewtonRaphsonNMB2dof(tol,m1,m2,k1,
    k2,beta,aMax,maxIter,W)

% This function calculates the output for the modal backbone curves of the
% 2dof oscillator

% Arrays with the oscillation amplitudes of the first and second mass
% computed with N-R
Amplitude1NMB = zeros(1,length(W));
Amplitude2NMB = zeros(1,length(W));

% For each pulsation value N-R is applied to solve the equation
% G(amplitude1,amplitude2,w)=0
for i=1:length(W)
    % n: number of iterations
    n = 1;
    % The initial amplitude array for the N-R loop is [aMax; -aMax]
    amplitude = [aMax; -aMax];
    % Initial evaluation of the equation G(amplitude(1),amplitude(2),W(i))
    F = NMBeq2dof(m1,m2,k1,k2,beta,W(i),amplitude(1),amplitude(2));
    % NEWTON-RAPHSON LOOP
    while norm(F)>tol && n<=maxIter
        F = NMBeq2dof(m1,m2,k1,k2,beta,W(i),amplitude(1),amplitude(2));
        J = NMBjac2dof(m1,m2,k1,k2,beta,W(i),amplitude(1),amplitude(2));
        amplitude = amplitude - J\F;
        n = n+1;
    end

    % The solutions computed in the array amplitude are assigned to the
    % output matrices Amplitude1NMB and Amplitude2NMB
    if n<=maxIter,
        Amplitude1NMB(i) = amplitude(1);
```

```
32            Amplitude2NMB(i) = amplitude(2);
33        end
34    end
35
36    end
```

A.1.15 NMBeq2dof.m

This m-file returns the value of the function $\mathbf{G}(A_1, A_2, \omega)$, where $\mathbf{G}(A_1, A_2, \omega) = \mathbf{0}$ is the system of equation depicted in 5.1 for the modal curve of the 2dofs system. It is called by the Newton-Raphson loop in *NewtonRaphsonNMB2dof.m*.

```
1  function F = NMBeq2dof(m1,m2,k1,k2,beta,w,amplitude1,amplitude2)
2
3  % This function returns the value of the 2dofs algebraic system
4  % G(amplitude1,amplitude2,w), represented here by double scalar F, for the
5  % modal backbone curve
6
7  F = [-3*beta*amplitude2^3-4*k2*amplitude2+4*k1*amplitude1+3*beta*amplitude1
       ^3-9*beta*amplitude2*amplitude1^2+9*beta*amplitude2^2*amplitude1+4*k2*
       amplitude1-4*m1*amplitude1*w^2;
8        -3*beta*amplitude2^3-4*k2*amplitude2+3*beta*amplitude1^3-9*beta*
       amplitude2*amplitude1^2+9*beta*amplitude2^2*amplitude1+4*k2*amplitude1
       +4*m2*amplitude2*w^2];
9
10 end
```

A.1.16 NMBjac2dof.m

This m-function returns the jacobian matrix of the modal curve equations, $\mathbf{G}(A_1, A_2, \omega) = \mathbf{0}$, of the 2dofs model, for its use in the Newton-Raphson algorithm of *NewtonRaphsonNMB2dof.m*.

```
1  function J = NMBjac2dof(m1,m2,k1,k2,beta,w,amplitude1,amplitude2)
2
```

```
% This function returns the jacobian matrix J of the 2dofs algebraic system
% for the modal backbone curve

J = [4*k1+9*beta*amplitude1^2-18*beta*amplitude2*amplitude1+9*beta*
    amplitude2^2+4*k2-4*w^2*m1, -9*beta*amplitude2^2-4*k2+18*beta*amplitude2
    *amplitude1-9*beta*amplitude1^2;
      4*k2+9*beta*amplitude1^2+9*beta*amplitude2^2-18*beta*amplitude1*
    amplitude2, -9*beta*amplitude2^2-4*k2+18*beta*amplitude2*amplitude1-9*
    beta*amplitude1^2+4*w^2*m2];

end
```

A.2 Codes for the NNMs computation in NNMcont

These are the Matlab functions that return the parameters of the model to analyse in NNMcont with the method described in chapter 4. The model represents the 2dofs non-linear oscillator described in 2.1.2. The results of the computation are found in chapter 5.

A.2.1 defsys.m

This is the *defsys.m* m-function file depicted and described in 4.4.

```
function sys = defsys()

% Linear system:
%-------------------

m1 = 1;
m2 = 1;
k1 = 1;
k2 = 1;

%beta = 0;
```

```
12 beta = 0.1;
13 %beta = 1;
14
15 sys.Klin=[k1+k2 -k2;-k2 k2];
16 sys.Mlin=diag([m1 m2]);
17
18 % Nonlinearities:
19 %----------------
20
21 % nonlinear law object:
22 coeffnl=beta;expnl=3;
23 nl_law=NL_LAW_POLY(coeffnl,expnl); % polynomial law
24
25 % nonlinear object:
26 pos1=1;pos2=2;
27 nl_spring=NL_SPRING(pos1,pos2,nl_law); % nonlinear spring
28
29 sys.nl(1)=nl_spring;
30
31 % Optional fields:
32 %-----------------
33
34 sys.norm=0.5*1e-3;
35
36 %sys.invMl=inv(sys.Ml);
37
38 %sys.IGcont=[];
39 %sys.IGcont.x0=[2.14367957541056;4.23248916538581;0;0];
40 %sys.IGcont.w=1.23465320905258;
41
42 %sys.vitfix=[2];
43
44 %sys.TFS=300;
45 %sys.NumMeth='NEWMARK';
```

A.2.2 paramdisp.m

This is the second function for the problem definition in NNMcont, called *paramdisp.m*. It is shown in 4.4 too. It determines the format of some of the graphical outputs of the computation.

```
% User Display for "NNM Plot" button:
% Inputs:
% - x is the response (state vector) of the system over one time period
% x(1:end/2,:), displacements (time evolution);
% x(end/2+1:end,:), velocities (time evolution);
% - t, time vector.

figure
% Motion in configuration space
plot(x(1,:),x(2,:))
xlabel('x_1')
ylabel('x_2')
```

Appendix B

Maple sheets

In this Project I have made use of a computational algebra system for symbolic calculation called 'Maple' for assistance in some analytic calculations. Namely, those calculations were the application of the harmonic balance method to all the mechanical models considered in this Project. Those calculations involved very intensive analytic manipulation, the calculation of jacobian matrices and the integration of certain expresions as described in 2.2.2, 2.3.1.1 and 2.3.1.2.

The Maple version that has been used is Maple 14.

B.1 HarmonicBalance.mw

All those calculations are collected in a single Maple sheet called *HarmonicBalance.mw* and shown here. Comments and anotations were added to the file to describe the purpose of each command and the outputs produced.

With this Maple sheet we obtain the systems of nonlinear algebraic equations -and their jacobians- that determine the forced response curves and modal backbones of a mechanical system, with parameter ω, by applying the harmonic balance method.

Single degree of freedom case:

Equation of the one degree of freedom oscillator:

$$eq := M\cdot\frac{d^2}{dt^2}x(t) + C\cdot\frac{d}{dt}x(t) + K\cdot x(t) + \beta\cdot x(t)^3 - F\cdot\sin(\omega\cdot t)$$

$$M\left(\frac{d^2}{dt^2}x(t)\right) + C\left(\frac{d}{dt}x(t)\right) + K\,x(t) + \beta\,x(t)^3 - F\sin(\omega\,t) \quad (1$$

We substitute a solution with one single fundamental harmonic, resulting in the residuals of the equations:

$$subs(x(t) = A\cdot\sin(\omega\cdot t) + B\cdot\cos(\omega\cdot t), eq)$$

$$M\left(\frac{\partial^2}{\partial t^2}\left(A\sin(\omega\,t) + B\cos(\omega\,t)\right)\right) + C\left(\frac{\partial}{\partial t}\left(A\sin(\omega\,t) + B\cos(\omega\,t)\right)\right) + K\left(A\sin(\omega\,t)\right. \quad (2$$
$$\left. + B\cos(\omega\,t)\right) + \beta\left(A\sin(\omega\,t) + B\cos(\omega\,t)\right)^3 - F\sin(\omega\,t)$$

$$eq := simplify(\%)$$

$$-MA\sin(\omega\,t)\,\omega^2 - MB\cos(\omega\,t)\,\omega^2 + CA\cos(\omega\,t)\,\omega - CB\sin(\omega\,t)\,\omega + KA\sin(\omega\,t) \quad (3$$
$$+ KB\cos(\omega\,t) + \beta A^3\sin(\omega\,t) - \beta A^3\sin(\omega\,t)\cos(\omega\,t)^2 + 3\,\beta A^2 B\cos(\omega\,t)$$
$$- 3\,\beta A^2 B\cos(\omega\,t)^3 + 3\,\beta A\sin(\omega\,t)\,B^2\cos(\omega\,t)^2 + \beta B^3\cos(\omega\,t)^3 - F\sin(\omega\,t)$$

We apply the Galerkin method to ortogonalise the residual and to obtain the algebraic equations for the coefficients of the solution:

Their derivatives with respect to the residuals will determine the jacobian matrix of the algebraic system.

$$a := \int_0^{\frac{2\cdot Pi}{\omega}} eq\cdot\sin(\omega\cdot t)\,dt$$

$$\frac{1}{4}\,\frac{\pi\left(3\,\beta A B^2 - 4F + 4KA - 4MA\,\omega^2 - 4CB\,\omega + 3\,\beta A^3\right)}{\omega} \quad (4$$

$$\frac{\partial}{\partial A}\, a$$

$$\frac{1}{4}\,\frac{\pi\left(3\,\beta\,B^2+4\,K-4\,\omega^2\,M+9\,\beta\,A^2\right)}{\omega} \tag{5}$$

$$\frac{\partial}{\partial B}\, a$$

$$\frac{1}{4}\,\frac{\pi\left(6\,\beta\,A\,B-4\,\omega\,C\right)}{\omega} \tag{6}$$

Setting B, the forcing and the damping coefficients to zero we obtain the equation of the modal backbone and its derivative.

$$subs(B=0,\, C=0,\, F=0\,,\, a)$$

$$\frac{1}{4}\,\frac{\pi\left(4\,K\,A-4\,M\,A\,\omega^2+3\,\beta\,A^3\right)}{\omega} \tag{7}$$

$$subs\left(B=0,\, C=0,\, F=0,\, \frac{\partial}{\partial A}\, a\right)$$

$$\frac{1}{4}\,\frac{\pi\left(4\,K-4\,\omega^2\,M+9\,\beta\,A^2\right)}{\omega} \tag{8}$$

$$a := \int_0^{\frac{2\cdot \mathrm{Pi}}{\omega}} eq\cdot\cos(\omega\cdot t)\ \mathrm{d}t$$

$$\frac{1}{4}\,\frac{\pi\left(4\,K\,B-4\,M\,B\,\omega^2+4\,\omega\,C\,A+3\,\beta\,B^3+3\,\beta\,A^2\,B\right)}{\omega} \tag{9}$$

$$\frac{\partial}{\partial A}\, a$$

$$\frac{1}{4}\,\frac{\pi\left(4\,\omega\,C+6\,\beta\,A\,B\right)}{\omega} \tag{10}$$

$$\frac{\partial}{\partial B}\, a$$

$$\frac{1}{4}\,\frac{\pi\left(4\,K-4\,\omega^2\,M+9\,\beta\,B^2+3\,\beta\,A^2\right)}{\omega} \tag{11}$$

Two degrees of freedom case:

Equations of the 2dofs oscillator:

$$eq1 := M1\cdot\frac{d^2}{dt^2}x1(t) + C1\cdot\frac{d}{dt}x1(t) - C2\cdot\frac{d}{dt}(x2(t) - x1(t)) + K1\cdot x1(t) - K2\cdot(x2(t) - x1(t)) - \beta\cdot(x2(t) - x1(t))^3 - F11\cdot\sin(\omega\cdot t) - F12\cdot\cos(\omega\cdot t)$$

$$M1\left(\frac{d^2}{dt^2}x1(t)\right) + C1\left(\frac{d}{dt}x1(t)\right) - C2\left(\frac{d}{dt}x2(t) - \left(\frac{d}{dt}x1(t)\right)\right) + K1\,x1(t) - K2\,(x2(t) - x1(t)) - \beta\,(x2(t) - x1(t))^3 - F11\sin(\omega\,t) - F12\cos(\omega\,t) \quad (12)$$

$$eq2 := M2\cdot\frac{d^2}{dt^2}x2(t) + C2\cdot\frac{d}{dt}(x2(t) - x1(t)) + K2\cdot(x2(t) - x1(t)) + \beta\cdot(x2(t) - x1(t))^3 - F2\cdot\sin(\omega\cdot t) - F22\cdot\cos(\omega\cdot t)$$

$$M2\left(\frac{d^2}{dt^2}x2(t)\right) + C2\left(\frac{d}{dt}x2(t) - \left(\frac{d}{dt}x1(t)\right)\right) + K2\,(x2(t) - x1(t)) + \beta\,(x2(t) - x1(t))^3 - F21\sin(\omega\,t) - F22\cos(\omega\,t) \quad (13)$$

We substitute the single-harmonic solutions for the displacement of each degree of freedom, obtainin the residuals:

$$eq1 := subs(x1(t) = A1\cdot\sin(\omega\cdot t) + B1\cdot\cos(\omega\cdot t), eq1)$$

$$M1\left(\frac{\partial^2}{\partial t^2}(A1\sin(\omega\,t) + B1\cos(\omega\,t))\right) + C1\left(\frac{\partial}{\partial t}(A1\sin(\omega\,t) + B1\cos(\omega\,t))\right) - C2\left(\frac{d}{dt}x2(t) - \left(\frac{\partial}{\partial t}(A1\sin(\omega\,t) + B1\cos(\omega\,t))\right)\right) + K1\,(A1\sin(\omega\,t) + B1\cos(\omega\,t)) - K2\,(x2(t) - A1\sin(\omega\,t) - B1\cos(\omega\,t)) - \beta\,(x2(t) - A1\sin(\omega\,t) - B1\cos(\omega\,t))^3 - F11\sin(\omega\,t) - F12\cos(\omega\,t) \quad (14)$$

$$eq1 := subs(x2(t) = A2\cdot\sin(\omega\cdot t) + B2\cdot\cos(\omega\cdot t), eq1)$$

$$M1\left(-A1\sin(\omega\,t)\,\omega^2 - B1\cos(\omega\,t)\,\omega^2\right) + C1\,(A1\cos(\omega\,t)\,\omega - B1\sin(\omega\,t)\,\omega) - C2\left(\frac{\partial}{\partial t}(A2\sin(\omega\,t) + B2\cos(\omega\,t)) - A1\cos(\omega\,t)\,\omega + B1\sin(\omega\,t)\,\omega\right) + K1\,(A1\sin(\omega\,t) + B1\cos(\omega\,t)) - K2\,(A2\sin(\omega\,t) + B2\cos(\omega\,t) - A1\sin(\omega\,t) - B1\cos(\omega\,t)) - \beta\,(A2\sin(\omega\,t) + B2\cos(\omega\,t) - A1\sin(\omega\,t) - B1\cos(\omega\,t))^3 - F11\sin(\omega\,t) - F12\cos(\omega\,t) \quad (15)$$

$$eq2 := subs(x1(t) = A1\cdot\sin(\omega\cdot t) + B1\cdot\cos(\omega\cdot t), eq2)$$

$$M2\left(\frac{d^2}{dt^2}x2(t)\right) + C2\left(\frac{d}{dt}x2(t) - \left(\frac{\partial}{\partial t}(A1\sin(\omega\,t) + B1\cos(\omega\,t))\right)\right) + K2\,(x2(t) - A1\sin(\omega\,t) - B1\cos(\omega\,t)) + \beta\,(x2(t) - A1\sin(\omega\,t) - B1\cos(\omega\,t))^3 - F21\sin(\omega\,t) - F22\cos(\omega\,t) \quad (16)$$

$$eq2 := subs(x2(t) = A2\cdot\sin(\omega\cdot t) + B2\cdot\cos(\omega\cdot t), eq2)$$

$$M2\left(\frac{\partial^2}{\partial t^2}\left(A2\sin(\omega t) + B2\cos(\omega t)\right)\right) + C2\left(\frac{\partial}{\partial t}\left(A2\sin(\omega t) + B2\cos(\omega t)\right)\right. \tag{17}$$
$$\left. - A1\cos(\omega t)\,\omega + B1\sin(\omega t)\,\omega\right) + K2\left(A2\sin(\omega t) + B2\cos(\omega t) - A1\sin(\omega t)\right.$$
$$\left. - B1\cos(\omega t)\right) + \beta\left(A2\sin(\omega t) + B2\cos(\omega t) - A1\sin(\omega t) - B1\cos(\omega t)\right)^3$$
$$- F21\sin(\omega t) - F22\cos(\omega t)$$

$$eq1 := simplify(eq1)$$

$$6\,\beta\,A2\sin(\omega t)\,B2\cos(\omega t)^2\,B1 - 6\,\beta\,B2\cos(\omega t)^2\,A1\sin(\omega t)\,B1 \tag{18}$$
$$- 3\,\beta\,A2\sin(\omega t)\,B2^2\cos(\omega t)^2 - 3\,\beta\,A2\sin(\omega t)\,B1^2\cos(\omega t)^2$$
$$+ 3\,\beta\,B2^2\cos(\omega t)^2\,A1\sin(\omega t) + 3\,\beta\,A1\sin(\omega t)\,B1^2\cos(\omega t)^2 - 6\,\beta\,A2\,A1\,B1\cos(\omega t)$$
$$+ 6\,\beta\,A2\,B2\cos(\omega t)\,A1 - 3\,\beta\,A2^2\,A1\sin(\omega t)\cos(\omega t)^2 + 3\,\beta\,A2\,A1^2\sin(\omega t)\cos(\omega t)^2$$
$$+ 6\,\beta\,A2\,A1\,B1\cos(\omega t)^3 - 6\,\beta\,A2\,B2\cos(\omega t)^3\,A1 + 3\,\beta\,A2^2\,A1\sin(\omega t)$$
$$- 3\,\beta\,A2\,A1^2\sin(\omega t) + K1\,A1\sin(\omega t) + K1\,B1\cos(\omega t) - K2\,A2\sin(\omega t)$$
$$- K2\,B2\cos(\omega t) + K2\,A1\sin(\omega t) + K2\,B1\cos(\omega t) - \beta\,B2^3\cos(\omega t)^3$$
$$+ \beta\,B1^3\cos(\omega t)^3 - \beta\,A2^3\sin(\omega t) + 3\,\beta\,A2^2\,B2\cos(\omega t)^3 - 3\,\beta\,A2^2\,B1\cos(\omega t)^3$$
$$+ 3\,\beta\,B2\cos(\omega t)^3\,A1^2 - 3\,\beta\,A1^2\,B1\cos(\omega t)^3 - \beta\,A1^3\sin(\omega t)\cos(\omega t)^2$$
$$+ \beta\,A2^3\sin(\omega t)\cos(\omega t)^2 + \beta\,A1^3\sin(\omega t) - F11\sin(\omega t) - F12\cos(\omega t)$$
$$+ 3\,\beta\,A1^2\,B1\cos(\omega t) - M1\,A1\sin(\omega t)\,\omega^2 - M1\,B1\cos(\omega t)\,\omega^2 + C1\,A1\cos(\omega t)\,\omega$$
$$- C1\,B1\sin(\omega t)\,\omega - C2\,A2\cos(\omega t)\,\omega + C2\,B2\sin(\omega t)\,\omega + C2\,A1\cos(\omega t)\,\omega$$
$$- C2\,B1\sin(\omega t)\,\omega + 3\,\beta\,B2^2\cos(\omega t)^3\,B1 - 3\,\beta\,B2\cos(\omega t)^3\,B1^2 - 3\,\beta\,B2\cos(\omega t)\,A1^2$$
$$+ 3\,\beta\,A2^2\,B1\cos(\omega t) - 3\,\beta\,A2^2\,B2\cos(\omega t)$$

$$eq2 := simplify(eq2)$$

$$-6\,\beta\,A2\sin(\omega t)\,B2\cos(\omega t)^2\,B1 + 6\,\beta\,B2\cos(\omega t)^2\,A1\sin(\omega t)\,B1 \tag{19}$$
$$+ 3\,\beta\,A2\sin(\omega t)\,B2^2\cos(\omega t)^2 + 3\,\beta\,A2\sin(\omega t)\,B1^2\cos(\omega t)^2$$
$$- 3\,\beta\,B2^2\cos(\omega t)^2\,A1\sin(\omega t) - 3\,\beta\,A1\sin(\omega t)\,B1^2\cos(\omega t)^2 + 6\,\beta\,A2\,A1\,B1\cos(\omega t)$$
$$- 6\,\beta\,A2\,B2\cos(\omega t)\,A1 + 3\,\beta\,A2^2\,A1\sin(\omega t)\cos(\omega t)^2 - 3\,\beta\,A2\,A1^2\sin(\omega t)\cos(\omega t)^2$$
$$- 6\,\beta\,A2\,A1\,B1\cos(\omega t)^3 + 6\,\beta\,A2\,B2\cos(\omega t)^3\,A1 - 3\,\beta\,A2^2\,A1\sin(\omega t)$$
$$+ 3\,\beta\,A2\,A1^2\sin(\omega t) + K2\,A2\sin(\omega t) + K2\,B2\cos(\omega t) - K2\,A1\sin(\omega t)$$
$$- K2\,B1\cos(\omega t) + \beta\,B2^3\cos(\omega t)^3 - \beta\,B1^3\cos(\omega t)^3 + \beta\,A2^3\sin(\omega t)$$
$$- 3\,\beta\,A2^2\,B2\cos(\omega t)^3 + 3\,\beta\,A2^2\,B1\cos(\omega t)^3 - 3\,\beta\,B2\cos(\omega t)^3\,A1^2$$

$$+3\,\beta\,A1^2\,B1\cos(\omega\,t)^3+\beta\,A1^3\sin(\omega\,t)\cos(\omega\,t)^2-\beta\,A2^3\sin(\omega\,t)\cos(\omega\,t)^2$$
$$-M2\,A2\sin(\omega\,t)\,\omega^2-M2\,B2\cos(\omega\,t)\,\omega^2-\beta\,A1^3\sin(\omega\,t)-3\,\beta\,A1^2\,B1\cos(\omega\,t)$$
$$+C2\,A2\cos(\omega\,t)\,\omega-C2\,B2\sin(\omega\,t)\,\omega-C2\,A1\cos(\omega\,t)\,\omega+C2\,B1\sin(\omega\,t)\,\omega$$
$$-3\,\beta\,B2^2\cos(\omega\,t)^3\,B1+3\,\beta\,B2\cos(\omega\,t)^3\,B1^2+3\,\beta\,B2\cos(\omega\,t)\,A1^2-F21\sin(\omega\,t)$$
$$-F22\cos(\omega\,t)-3\,\beta\,A2^2\,B1\cos(\omega\,t)+3\,\beta\,A2^2\,B2\cos(\omega\,t)$$

Again, applying Galerkin on each residual:

$$a:=\int_0^{\frac{2\cdot Pi}{\omega}} eq1\cdot\sin(\omega\cdot t)\,dt$$

$$\frac{1}{4}\,\frac{1}{\omega}\left(\pi\left(3\,\beta\,A1^3-3\,\beta\,A2^3+4\,K2\,A1+4\,K1\,A1-4\,K2\,A2-4\,F11+6\,\beta\,A2\,B2\,B1\right.\right. \tag{20}$$
$$-6\,\beta\,B2\,A1\,B1+3\,\beta\,B2^2\,A1+3\,\beta\,A1\,B1^2-3\,\beta\,A2\,B2^2-3\,\beta\,A2\,B1^2+9\,\beta\,A2^2\,A1$$
$$\left.\left.-9\,\beta\,A2\,A1^2-4\,M1\,A1\,\omega^2-4\,C1\,B1\,\omega-4\,C2\,B1\,\omega+4\,C2\,B2\,\omega\right)\right)$$

The derivatives of each nonlinear algebraic equation with respect to the coefficients determine the jacobian:

$$\frac{\partial}{\partial A1}\,a$$
$$\frac{1}{4}\,\frac{1}{\omega}\left(\pi\left(9\,\beta\,A1^2+4\,K2+4\,K1-6\,\beta\,B2\,B1+3\,\beta\,B2^2+3\,\beta\,B1^2+9\,\beta\,A2^2-18\,\beta\,A2\,A1\right.\right. \tag{21}$$
$$\left.\left.-4\,M1\,\omega^2\right)\right)$$

$$\frac{\partial}{\partial B1}\,a$$
$$\frac{1}{4}\,\frac{\pi\left(6\,\beta\,A2\,B2-6\,\beta\,B2\,A1+6\,\beta\,A1\,B1-6\,\beta\,A2\,B1-4\,C1\,\omega-4\,C2\,\omega\right)}{\omega} \tag{22}$$

$$\frac{\partial}{\partial A2}\,a$$
$$\frac{1}{4}\,\frac{\pi\left(-9\,\beta\,A2^2-4\,K2+6\,\beta\,B2\,B1-3\,\beta\,B2^2-3\,\beta\,B1^2+18\,\beta\,A2\,A1-9\,\beta\,A1^2\right)}{\omega} \tag{23}$$

$$\frac{\partial}{\partial B2}\,a$$
$$\frac{1}{4}\,\frac{\pi\left(6\,\beta\,A2\,B1-6\,\beta\,A1\,B1+6\,\beta\,B2\,A1-6\,\beta\,A2\,B2+4\,C2\,\omega\right)}{\omega} \tag{24}$$

Equally to the SDOF case, setting the B coefficients, dampings and forcings to zero results in the equations for the modal backbones:

$$subs(B1=0, B2=0, C1=0, C2=0, F11=0, F12=0, F21=0, F22=0, a)$$

$$\frac{1}{4}\,\frac{1}{\omega}\left(\pi\left(3\,\beta\,A1^3-3\,\beta\,A2^3+4\,K2\,A1+4\,K1\,A1-4\,K2\,A2+9\,\beta\,A2^2\,A1-9\,\beta\,A2\,A1^2-4\,M1\,A1\,\omega^2\right)\right) \tag{25}$$

$$subs\left(B1=0, B2=0, C1=0, C2=0, F11=0, F12=0, F21=0, F22=0, \frac{\partial}{\partial A1}\,a\right)$$

$$\frac{1}{4}\,\frac{\pi\left(9\,\beta\,A1^2+4\,K2+4\,K1+9\,\beta\,A2^2-18\,\beta\,A2\,A1-4\,M1\,\omega^2\right)}{\omega}$$

$$subs\left(B1=0, B2=0, C1=0, C2=0, F11=0, F12=0, F21=0, F22=0, \frac{\partial}{\partial A2}\,a\right)$$

$$\frac{1}{4}\,\frac{\pi\left(-9\,\beta\,A2^2-4\,K2+18\,\beta\,A2\,A1-9\,\beta\,A1^2\right)}{\omega} \tag{27}$$

$$a:=\int_0^{\frac{2\cdot Pi}{\omega}} eq1\cdot\cos(\omega\cdot t)\,dt$$

$$\frac{1}{4}\,\frac{1}{\omega}\left(\pi\left(3\,\beta\,B1^3-4\,K2\,B2-3\,\beta\,B2^3+4\,K2\,B1+4\,K1\,B1+4\,C2\,A1\,\omega-4\,M1\,B1\,\omega^2-4\,C2\,A2\,\omega+4\,C1\,A1\,\omega+3\,\beta\,A2^2\,B1-3\,\beta\,A2^2\,B2-3\,\beta\,B2\,A1^2+3\,\beta\,A1^2\,B1+9\,\beta\,B2^2\,B1-9\,\beta\,B2\,B1^2+6\,\beta\,A2\,B2\,A1-6\,\beta\,A2\,A1\,B1-4\,F12\right)\right) \tag{28}$$

$$\frac{\partial}{\partial A1}\,a$$

$$\frac{1}{4}\,\frac{\pi\left(4\,C2\,\omega+4\,C1\,\omega-6\,\beta\,B2\,A1+6\,\beta\,A1\,B1+6\,\beta\,A2\,B2-6\,\beta\,A2\,B1\right)}{\omega} \tag{30}$$

$$\frac{\partial}{\partial B1}\,a$$

$$\frac{1}{4}\,\frac{1}{\omega}\left(\pi\left(9\,\beta\,B1^2+4\,K2+4\,K1-4\,M1\,\omega^2+3\,\beta\,A2^2+3\,\beta\,A1^2+9\,\beta\,B2^2-18\,\beta\,B2\,B1-6\,\beta\,A2\,A1\right)\right) \tag{31}$$

$$\frac{\partial}{\partial A2}\,a$$

$$\frac{1}{4}\,\frac{\pi\left(-6\,\beta\,A1\,B1+6\,\beta\,B2\,A1-6\,\beta\,A2\,B2+6\,\beta\,A2\,B1-4\,C2\,\omega\right)}{\omega} \tag{32}$$

$$\frac{\partial}{\partial B2}\, a$$

$$\frac{1}{4}\,\frac{\pi\left(-4\,K2-9\,\beta\,B2^2-3\,\beta\,A2^2-3\,\beta\,A1^2+18\,\beta\,B2\,B1-9\,\beta\,B1^2+6\,\beta\,A2\,A1\right)}{\omega} \tag{33}$$

$$a := \int_0^{\frac{2\cdot Pi}{\omega}} eq2\cdot\sin(\omega\cdot t)\,\mathrm{d}t$$

$$-\frac{1}{4}\,\frac{1}{\omega}\left(\pi\left(4\,F21-3\,\beta\,A2^3+3\,\beta\,A1^3-4\,K2\,A2+4\,K2\,A1-4\,C2\,B1\,\omega+4\,\omega^2\,M2\,A2\right.\right. \tag{34}$$
$$+4\,C2\,B2\,\omega+3\,\beta\,B2^2\,A1+3\,\beta\,A1\,B1^2-3\,\beta\,A2\,B2^2-3\,\beta\,A2\,B1^2+9\,\beta\,A2^2\,A1$$
$$\left.\left.-9\,\beta\,A2\,A1^2+6\,\beta\,A2\,B2\,B1-6\,\beta\,B2\,A1\,B1\right)\right)$$

$$\frac{\partial}{\partial A1}\, a$$

$$-\frac{1}{4}\,\frac{\pi\left(9\,\beta\,A1^2+4\,K2+3\,\beta\,B2^2+3\,\beta\,B1^2+9\,\beta\,A2^2-18\,\beta\,A2\,A1-6\,\beta\,B2\,B1\right)}{\omega} \tag{36}$$

$$\frac{\partial}{\partial B1}\, a$$

$$-\frac{1}{4}\,\frac{\pi\left(-4\,C2\,\omega+6\,\beta\,A1\,B1-6\,\beta\,A2\,B1+6\,\beta\,A2\,B2-6\,\beta\,B2\,A1\right)}{\omega} \tag{37}$$

$$\frac{\partial}{\partial A2}\, a$$

$$-\frac{1}{4}\,\frac{\pi\left(-9\,\beta\,A2^2-4\,K2+4\,\omega^2\,M2-3\,\beta\,B2^2-3\,\beta\,B1^2+18\,\beta\,A2\,A1-9\,\beta\,A1^2+6\,\beta\,B2\,B1\right)}{\omega} \tag{38}$$

$$\frac{\partial}{\partial B2}\, a$$

$$-\frac{1}{4}\,\frac{\pi\left(6\,\beta\,A2\,B1-6\,\beta\,A1\,B1+6\,\beta\,B2\,A1-6\,\beta\,A2\,B2+4\,C2\,\omega\right)}{\omega} \tag{39}$$

$$subs(B1=0,\,B2=0,\,C1=0,\,C2=0,\,F11=0,\,F12=0,\,F21=0,\,F22=0,\,a)$$

$$-\frac{1}{4}\,\frac{\pi\left(-3\,\beta\,A2^3+3\,\beta\,A1^3-4\,K2\,A2+4\,K2\,A1+4\,\omega^2\,M2\,A2+9\,\beta\,A2^2\,A1-9\,\beta\,A2\,A1^2\right)}{\omega} \tag{40}$$

$$subs\left(B1=0,\,B2=0,\,C1=0,\,C2=0,\,F11=0,\,F12=0,\,F21=0,\,F22=0,\,\frac{\partial}{\partial A1}\, a\right)$$

$$-\frac{1}{4}\,\frac{\pi\left(9\,\beta\,A1^2+4\,K2+9\,\beta\,A2^2-18\,\beta\,A2\,A1\right)}{\omega}$$

$$subs\left(B1=0,\,B2=0,\,C1=0,\,C2=0,\,F11=0,\,F12=0\,,\,F21=0\,,\,F22=0,\,\frac{\partial}{\partial A2}\,a\right)$$

$$-\frac{1}{4}\,\frac{\pi\left(-9\,\beta\,A2^2-4\,K2+4\,\omega^2\,M2+18\,\beta\,A2\,A1-9\,\beta\,A1^2\right)}{\omega} \tag{42}$$

$$a:=\int_0^{\frac{2\cdot Pi}{\omega}} eq2\cdot\cos\left(\omega\cdot t\right)\,dt$$

$$-\frac{1}{4}\,\frac{1}{\omega}\left(\pi\left(-6\,\beta\,A2\,A1\,B1+6\,\beta\,A2\,B2\,A1-3\,\beta\,A2^2\,B2+3\,\beta\,A1^2\,B1-3\,\beta\,B2\,A1^2\right.\right. \tag{43}$$
$$\left.\left.+9\,\beta\,B2^2\,B1+3\,\beta\,A2^2\,B1-9\,\beta\,B2\,B1^2+4\,\omega^2\,B2\,M2-3\,\beta\,B2^3-4\,K2\,B2+4\,K2\,B1\right.\right.$$
$$\left.\left.+3\,\beta\,B1^3+4\,C2\,A1\,\omega-4\,C2\,A2\,\omega+4\,F22\right)\right)$$

$$\frac{\partial}{\partial A1}\,a$$

$$-\frac{1}{4}\,\frac{\pi\left(-6\,\beta\,A2\,B1+6\,\beta\,A2\,B2+6\,\beta\,A1\,B1-6\,\beta\,B2\,A1+4\,C2\,\omega\right)}{\omega} \tag{45}$$

$$\frac{\partial}{\partial B1}\,a$$

$$-\frac{1}{4}\,\frac{\pi\left(-6\,\beta\,A2\,A1+3\,\beta\,A1^2+9\,\beta\,B2^2+3\,\beta\,A2^2-18\,\beta\,B2\,B1+4\,K2+9\,\beta\,B1^2\right)}{\omega} \tag{46}$$

$$\frac{\partial}{\partial A2}\,a$$

$$-\frac{1}{4}\,\frac{\pi\left(-6\,\beta\,A1\,B1+6\,\beta\,B2\,A1-6\,\beta\,A2\,B2+6\,\beta\,A2\,B1-4\,C2\,\omega\right)}{\omega} \tag{47}$$

$$\frac{\partial}{\partial B2}\,a$$

$$-\frac{1}{4}\,\frac{\pi\left(6\,\beta\,A2\,A1-3\,\beta\,A2^2-3\,\beta\,A1^2+18\,\beta\,B2\,B1-9\,\beta\,B1^2+4\,\omega^2\,M2-9\,\beta\,B2^2-4\,K2\right)}{\omega} \tag{48}$$

Appendix C

Extra: a practical NNMs analysis of a snapthrough vibration absorber with my own Matlab codes

During the first stages of this Project, and before knowing of the existence of the NNMcont package, I worked on my own Matlab codes to apply the algorithm described in chapter 3. Later on, I got a chance to apply those codes and algorithms to a simple but practical example previously covered in the Academia.

In the end, this work has been discarded from the Project itself as it did not have nothing new to offer to it in contrast to the rest of the sections. However, it was a difficult archievement with a lot of value for the amount of time I invested to apply those algorithms and make them work, and because of the programming and analytical skills that I prove with it.

Again, I used Maple to assist myself through the analytical calculations and manipulations, but for reasons of space I abstain from publishing the sheets I used in this text.

C.1 The snapthrough vibration absorber

Out of the different vibration absortion devices that exist, this section deals with an absorber with a snapthrough geometry, conceived to atenuate longitudinal motion. The model of this mechanism is portrayed in C.1:

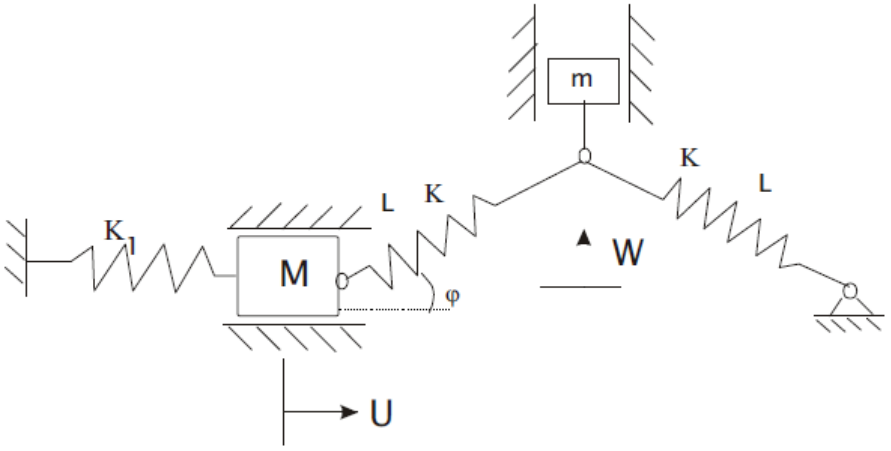


FIGURE C.1: Model of a snapthrough vibration absorber. This figure has been taken from [37].

The reason why this model has been chosen was that Mikhlin and Avramov had previously studied this model by applying analytical techniques –energy methods–. In [38] forced vibration is analized, while in [37] they apply the theory of nonlinear normal modes to study the free vibration regime of the system. In this appendix I describe as briefly as I can the steps I followed to compute numerically the NNMs that emanate from the stable equilibrium points of this system.

C.1.1 Model and equations

The equations of motion of the model C.1 are:

$$\begin{aligned} MU'' + K_1 U + K\left(U - L\cos\phi + \frac{L}{\sqrt{1+\frac{W^2}{(L\cos\phi - U)^2}}}\right) = 0 \\ mW'' + KW\left(2 - \frac{L}{\sqrt{(L\cos\phi - U)^2 + W^2}} - \frac{L}{\sqrt{(L\cos\phi)^2 + W^2}}\right) = 0 \end{aligned} \tag{C.1}$$

where U and W are the general coordinates of the two masses –the main mass and the absorbing mass, with masses M and m, respectively–, and hence the two degrees of freedom. L is the length of the beams or springs attatched to the absorbing mass. ϕ is the angle of those beams in their stable equilibrium position. K is the stiffness coefficient of the attatchment of the main mass to the ground, and K_1 is the stiffness of the springs or beams that conform the absorbing mechanism.

These equations coincide with the equations considered and studied in the paper [37].

This system has three equilibrium points or fixed points. the point $(\frac{KL(\cos\phi - 1)}{K_1 + K}, 0)$ is an unstable point –a 'source' in the dynamical system–, while $(0, L\sin\phi)$ and $(0, -L\sin\phi)$ are 'centers' –and therefore, stable points–. Due to the model symmetry this system is symmetric in dynamical behavior with respect to the degree of freedom W in the null position or origin –the unstable equilibrium point–.

The periodic oscillations occur around any of the two centers. Due to symmetry, dynamical behavior is equal in both, thus the modal analysis can be conducted in any of them.

To simplify the analysis, the coordinates origin $(0, 0)$ is translated to any of the two centers. This is done through the coordinate change $W = W_1 + L\sin\phi$, with the new degree of freedom W_1. The state $W_1 = 0$ conforms to the first of the two centers.

The new equations are after the coordinate change as follows:

$$\begin{aligned} MU'' + K_1 U + K\left(U - L\cos\phi + \frac{L}{\sqrt{1+\frac{(W_1+L\sin\phi)^2}{(L\cos\phi-U)^2}}}\right) = 0 \\ mW_1'' + 2K(W_1 + L\sin\phi) - \frac{KL(W_1+L\sin\phi)}{\sqrt{(L\cos\phi-U)^2+(W_1+Lsin\phi)^2}} \\ -\frac{KL(W_1+L\sin\phi)}{\sqrt{(L\cos\phi)^2+(W_1+L\sin\phi)^2}} = 0 \end{aligned} \tag{C.2}$$

The numerical computation of the NNMs branches stems from the linear normal modes of the system linearized with respect to the equilibrium point. The SODE for the linearized mechanical system has the well-known form:

$$[\mathbf{M}]\mathbf{x}'' + [\mathbf{K}]\mathbf{x} = \mathbf{0} \tag{C.3}$$

with mass and stiffness matrices $[\mathbf{M}]$ and $[\mathbf{K}]$, and state vector $\mathbf{x} = (U, W_1)^T$.

The linearized snapthrough absorber takes into consideration the hypotesis of small displacements of the degrees of freedom. In other words, the angle ϕ is considered to remain unchanged during the motion with respect to its value in the equilibrium position $(0, 0)$. With this in mind, it is not difficult to derive the mass and stiffness matrices of the linearized absorber:

$$[\mathbf{M}] = \begin{pmatrix} M, & 0 \\ 0, & m \end{pmatrix} \tag{C.4}$$

$$[\mathbf{K}] = \begin{pmatrix} K_1 + K\cos^2\phi, & -K\sin\phi\cos\phi \\ -K\sin\phi\cos\phi, & 2K\sin^2\phi \end{pmatrix} \tag{C.5}$$

The continuation algorithm described in this text requires to write the system of equations C.2 with the following structure:

$$[\mathbf{M}]\mathbf{x}'' + [\mathbf{K}]\mathbf{x} + \mathbf{f_{nl}}(\mathbf{x}) = \mathbf{0} \tag{C.6}$$

where $[\mathbf{K}]$ and $[\mathbf{M}]$ are the linear stiffness and mass matrices shown above, and $\mathbf{f_{nl}}(\mathbf{x})$ is the nonlinear stiffness component of C.2.

Rearranging terms of C.2, the nonlinear stiffness vector of C.2 has these components:

$$\begin{aligned} f_{nl,1} &= KU - KL\cos\phi + \frac{KL}{\sqrt{1 + \frac{W_1^2 + 2W_1L\sin\phi + L^2\sin^2\phi}{(L\cos\phi - U)^2}}} - K\cos^2\phi \\ &\quad + KW_1\sin\phi\cos\phi \\ f_{nl,2} &= 2KW_1 - \frac{KW_1L + KL^2\sin\phi}{\sqrt{L^2 - 2UL\cos\phi + U^2 + W_1^2 + 2W_1L\sin\phi}} \\ &\quad - \frac{KW_1L + KL^2\sin\phi}{\sqrt{L^2 + W_1^2 + 2W_1L\sin\phi}} - 2KW_1\sin^2\phi + KU\sin\phi\cos\phi \end{aligned} \tag{C.7}$$

In order to represent the computed NNMs on frequency-energy plots it is required an expression for the mechanical energy of a state $\mathbf{z} = (\mathbf{x}', \mathbf{x})$ of C.2 –Remember energy calculation in 3.11–.

The total energy of a state $\mathbf{z}$ is the sum of three components:

$$E = T + L_L + L_{NL} \tag{C.8}$$

where T is the kinetic energy –3.29–, L_L is the linear potential energy –3.30– and L_{NL} is the nonlinear potential energy –3.31– related to the nonlinear stiffness component of the system.

The nonlinear potential energy is the most difficult to calculate. Here I have used an analytical, heuristic approach to it, taking the procedure that I briefly sketched in the end of section 3.11.

The nonlinear potential energy results from the integration of any of the two components of the nonlinear force in C.7. Integrating $f_{nl,2}$ in the domain of W_1:

$$L_{NL} = \int f_{nl,2}(U, W_1)\, dW_1 = F_{nl,2}(U, W_1) + g(U) \tag{C.9}$$

where $g(U)$ must be chosen so that the derivative of the energy L_{NL} with respect to U returns the nonlinear force component $f_{nl,1}$:

$$\frac{\partial L_{NL}}{\partial U} = \frac{\partial F_{nl,2}}{\partial U} + \frac{\partial g}{\partial U} = f_{nl,1} \tag{C.10}$$

An exact solution of $g(U)$ is very difficult to find, but it can be approached by a polynomial of n-th order with coefficients $A_1, A_2 ... A_n$:

$$g(U) \approx A_0 + A_1 U + A_2 U^2 + ... + A_n U^n + \mathcal{O}(U^{n+1}) \tag{C.11}$$

Truncating to the first two terms, it is supposed a linear expression of $g(U)$:

$$g(U) \approx A_0 + A_1 U \tag{C.12}$$

Coefficients A_0 and A_1 must be chosen so that the approximation is as accurate as possible in certain points of the phase space (U, W_1). We define a residual R as the accurateness of the polynomial approximation of $g(U)$ to its real solution, defining the real solution of $g(U)$ as the one that satisfies the condition C.10:

$$R(U, W_1, A_0, A_1) = \frac{\partial F_{nl,2}}{\partial U} + \frac{\partial g}{\partial U} - f_{nl,1} = 0 \tag{C.13}$$

Substituting the approximation of $g(U)$ in C.9:

$$L_{NL} = \int f_{nl,2}(U, W_1)\, dW_1 = F_{nl,2}(U, W_1) + A_0 + A_1 U \tag{C.14}$$

By forcing L_{NL} to be zero in the origin $(0, 0)$, the coefficient A_0 can be cleared:

$$L_{NL}(0,0) = \int f_{nl,2}(0,0)\, dW_1 = F_{nl,2}(0,0) + A_0 = 0 \rightarrow A_0 \tag{C.15}$$

The residual R can be forced to be zero in the origin $(0,0)$, and the coefficient A_1 is cleared:

$$R(0,0,A_0,A_1) = \frac{\partial F_{nl,2}}{\partial U}(0,0) + \frac{\partial g}{\partial U}(0,0) - f_{nl,1}(0,0) = 0 \rightarrow A_1 \tag{C.16}$$

where $\frac{\partial g}{\partial U}(0,0) = A_1$.

The resulting expressions for L_{NL} and the two coefficients A_0 and A_1 are:

$$\begin{aligned} L_{NL} = K(&2W_1 L\sin\phi + W_1^2\cos^2\phi + W_1 U\sin\phi\cos\phi \\ &-L\sqrt{L^2 - 2L\cos\phi U + U^2 + W_1^2 + 2W_1 L\sin\phi} - L\sqrt{L^2 + W_1^1 + 2W_1 L\sin\phi}) \\ &+A_0 + A_1 U \end{aligned} \tag{C.17}$$

$$A_0 = 2KL^2\ sign(L)$$

$$A_1 = KL\ sign(L)\cos\phi(-1 - \ sign(L) + \ sign(\cos\phi\ sign(L)))$$

Unlike the algorithm described in chapter 3, the phase condition affects the displacements $\mathbf{x}$ –instead of the velocities of displacement, $\mathbf{x}'$– of the degrees of freedom. As a phase condition, the displacement of one of the two dofs must be set to zero. If the chosen displacement is U, it is ensured that the value of the nonlinear potential energy L_{NL} is exact –indeed, $g(U=0) = A_0$–.

C.2 Matlab codes for the NNMs computation

This is a Matlab package that was originally developed by myself with the aim to compute numerically the NNMs of any model whose equations were introduced in the corresponding m-files. Here those algorithms are applied to the snapthrough vibration absorber, but due to the modularity of this package any new model can be added and analized with the same codes.

This package is contained among the remaining source codes of this Project, all included in the adjoint CD-R and available for free download from the Internet. Namely, the codes are in a folder called 'Shooting analysis with my continuation algorithms'.

The subfolder 'main' contains the executable m-files and functions for the numerical calculations and the graphical representation of the FEPs, modal curves and time-series plots. The subfolder 'continuer' holds modules of Matlab functions for the computation of the NNMs via continuation algorithms, taking the continuation parameters and the initial LNMs as inputs and returning the numerical results. Finally, the subfolder 'models' contains whatever models are introduced for their analysis. In this case, one model –the snapthrough absorber– has been included.

Before executing any codes, the subfolders 'main' and 'continuer' must be added to the Matlab search path, and the subfolder of 'models' holding the system to be analized –in our application, 'snapthrough'– must be set as the current Matlab path.

In the upcoming sections, each Matlab file and function is described. Unfortunately, for reasons of time and space, I do not detail how the codes and algorithms work. I simply explain the purpose of each file and function and how to use the package.

C.2.1 'main' folder

C.2.1.1 contmain1.m

Files *contmain1.m* and *contmain2.m* are the main executables of the package. Their purpose is to obtain the numerical outputs of the computation of the NNMs of a certain model given a set of control parameters as an input. Those numerical results are stored in a mat-file.

Control parameters are first defined. Then, the m-function *initpoint* is called to calculate the low-energy linear normal mode which are the initial point of the NNMs branch computation. A call to the function *curvecomputation* starts the continuation algorithm, returning its numerical results. The meaning of each control parameter and output variable is defined in the file comments.

On the other hand, it is given the option to start the continuation from another NNM from the previous computation or a mat-file. As explained in the file comments just uncomment the corresponding section of the code given. During its execution the numerical data is loaded from a given mat-file, and the 'pt-th' branch point –where *pt* is the index of the NNM considered– is taken as the initial point of the new process. By taking another NNM as initial point one can continue a previous computation or trying to force branching.

contmain1.m computes the first NNMs branch of the system, taking the first linear mode as the initial point –$NumMod = 1$–, while *contmain2.m* computes the second branch –$NumMod = 2$–.

```
%-------------------------------------------------------------------------------
% THIS IS THE EXECUTABLE M-FILE FOR THE COMPUTATION OF THE NNMS BRANCHES
% FROM A MODEL CONTAINED IN THE 'MODELS' FILE

% WRITTEN BY ALEJANDRO SILVA, asilvaber@gmail.com

% DECEMBER 2015, MADRID, SPAIN
%-------------------------------------------------------------------------------

%-------------------------------------------------------------------------------
% THE PARAMETERS ON THIS FILE ARE SET FOR THE COMPUTATION OF THE FIRST MODE
% NNMS BRANCH OF THE SNAPTHROUGH ABSORBER MODEL
%-------------------------------------------------------------------------------

clear
clc

% INPUT PARAMETERS FOR THE COMPUTATION OF THE FIRST MODE OF THE SNAPTHROUGH
% ABSORBER

% Relative differentiation interval 'h' for the calculations of the
% jacobians with finite differences
RelDifferInt = 1e-7;
% Initial frequency stepsize for the predictor phase
InitFreqStep = -1e-6;
% Maximum absolute value of the frequency stepsize for the predictor phase
MaxFreqStep = 5e-5;
% Minimum absolute value of the frequency stepsize for the predictor phase
MinFreqStep = 1e-99;
% Maximum the number of NNMs to be obtained (first stopping condition)
MaxPtsCurve = 60;
% Maximum energy of the NNMs computed (second stopping condition)
MaxEnergy = 1e7;
% Maximum frequency of the NNMs computed (third stopping condition)
MaxFreq = 1;
% Maximum number of calculations of a jacobian matrix during the
% Newton-Raphson loop in the corrector phase
NumJacCalc = 5;
```

```
39 % Relative error tolerance of the solution obtained from the Newton-Raphson
40 % loop in the corrector phase (convergence condition)
41 ConvErrTol = 1e-6;
42 % Optimal ('average') number of iterations of the Newton-Raphson loop in
43 % the corrector phase
44 AverageIter = 10;
45 % Maximum number of iterations of the Newton-Raphson loop in the corrector
46 % phase
47 MaxIter = 15;
48 % Symmetry condition of the NNMs computed (1 = 'yes', 0 = 'no')
49 Symmetry = 0;
50 % Runge-Kutta relative tolerances for the shooting function and jacobians
51 % calculation, respectively
52 ShootRKRelTol = 1e-10;
53 JacobRKRelTol = 1e-12;
54 % Runge-Kutta absolute tolerances for the shooting function and jacobians
55 % calculation, respectively
56 ShootRKAbsTol = 1e-12;
57 JacobRKAbsTol = 1e-15;
58 % Degrees of freedom whose displacements are to be set to zero as a phase
59 % condition (indeces from 1 to the total number of dofs)
60 PhaseCondCoor = [ 1 ];
61
62 % Mat-file name where numerical results are save
63 matFile = 'NNModes_1.mat';
64
65 % The initial point is the NumMod-th linear normal mode of the linearized
66 % system, with y0 = (x0, v0) as modal shape and freq0 as natural frequency
67 NumMod = 1;
68 [y0,freq0] = initpoint(NumMod,1e-5);
69
70 % IF THE INITIAL POINT IS TAKEN FROM A CURVE CONTAINED IN A MAT-FILE
71 % UNCOMMENT THE FOLLOWING PARAMETER DEFINITIONS
72
73 % 'pt' is the index of the NNM taken from the mat-file as the initial point
74 % of a new branch
75
76 % load 'NNModes_1.mat'
```

```
77 % pt = 34;
78 % y0 = y11(pt,:);
79 % freq0 = freq11(pt);
80 % RelDifferInt = 1e-7;
81 % InitFreqStep = FreqStep(pt);
82 % MaxFreqStep = 4e-7;
83 % MinFreqStep = 1e-99;
84 % MaxPtsCurve = 35;
85 % MaxEnergy = 1e7;
86 % MaxFreq = 1;
87 % NumJacCalc = 5;
88 % ConvErrTol = 1e-6;
89 % AverageIter = 10;
90 % MaxIter = 15;
91 % Symmetry = 0;
92 % ShootRKRelTol = 1e-10;
93 % JacobRKRelTol = 1e-12;
94 % ShootRKAbsTol = 1e-12;
95 % JacobRKAbsTol = 1e-15;
96 % PhaseCondCoor = [ 1 ];
97 % matFile = 'NNModes_1_p.mat';
98
99
100 % CALL TO FUNCTION 'CURVECOMPUTATION' FOR THE NNMS BRANCH COMPUTATION
101
102 [y11,freq11,Energy,FloqMult,y01,freq01,TangVect,FreqStep,NumIter,ConvRelErr
        ] = curvecomputation(y0,freq0,RelDifferInt,InitFreqStep,MaxFreqStep,
        MinFreqStep,MaxPtsCurve,MaxEnergy,MaxFreq,NumJacCalc,ConvErrTol,
        AverageIter,MaxIter,Symmetry,ShootRKRelTol,JacobRKRelTol,ShootRKAbsTol,
        JacobRKAbsTol,PhaseCondCoor);
103
104 % OUTPUTS
105
106 % y11: state vector with displacement 'x' and velocity 'v', (x, v), of each
107 % NNM calculated by the algorithm, under a certain phase condition
108 % freq11: frequency of each NNMs
109 % Energy: total energy of each NNM
110 % FloqMult: Floquet multipliers calculated for each NNM
```

```
% y01: state vector of a new point given by each predictor phase
% freq01: frequency of a new point given by each predictor phase
% TangVect: Tangent vector calculated in each predictor phase
% FreqStep: Frequency stepsize of each predictor phase
% NumIter: Number of iterations of each corrector phase
% ConvRelErr: relative Newton-Raphson error archieve during each corrector
% phase

% RESULTS ARE SAVED

save(matFile)
```

C.2.1.2 contmain2.m

```
%-------------------------------------------------------------------------
% THIS IS THE EXECUTABLE M-FILE FOR THE COMPUTATION OF THE NNMS BRANCHES
% FROM A MODEL CONTAINED IN THE 'MODELS' FILE

% WRITTEN BY ALEJANDRO SILVA, asilvaber@gmail.com

% DECEMBER 2015, MADRID, SPAIN
%-------------------------------------------------------------------------

%-------------------------------------------------------------------------
% THE PARAMETERS ON THIS FILE ARE SET FOR THE COMPUTATION OF THE FIRST MODE
% NNMS BRANCH OF THE SNAPTHROUGH ABSORBER MODEL
%-------------------------------------------------------------------------

clear
clc

% INPUT PARAMETERS FOR THE COMPUTATION OF THE SECOND MODE OF THE
% SNAPTHROUGH ABSORBER

% Relative differentiation interval 'h' for the calculations of the
% jacobians with finite differences
RelDifferInt = 1e-7;
```

```
% Initial frequency stepsize for the predictor phase
InitFreqStep = -1e-7;
% Maximum frequency stepsize for the predictor phase
MaxFreqStep = 1e-7;
% Minimum frequency stepsize for the predictor phase
MinFreqStep = 1e-99;
% Maximum number of NNMs to be obtained (first stopping condition)
MaxPtsCurve = 10;
% Maximum energy of the NNMs computed (second stopping condition)
MaxEnergy = 1e7;
% Maximum frequency of the NNMs computed (third stopping condition)
MaxFreq = 1;
% Maximum number of calculations of a jacobian matrix during the
% Newton-Raphson loop in the corrector phase
NumJacCalc = 5;
% Relative error tolerance of the solution obtained from the Newton-Raphson
% loop in the corrector phase (convergence condition)
ConvErrTol = 1e-5;
% Optimal ('average') number of iterations of the Newton-Raphson loop in
% the corrector phase
AverageIter = 10;
% Maximum number of iterations of the Newton-Raphson loop in the corrector
% phase
MaxIter = 15;
% Symmetry condition of the NNMs computed (1 = 'yes', 0 = 'no')
Symmetry = 0;
% Runge-Kutta relative tolerances for the shooting function and jacobians
% calculation, respectively
ShootRKRelTol = 1e-10;
JacobRKRelTol = 1e-12;
% Runge-Kutta absolute tolerances for the shooting function and jacobians
% calculation, respectively
ShootRKAbsTol = 1e-12;
JacobRKAbsTol = 1e-15;
% Degrees of freedom whose displacements are to be set to zero as a phase
% condition (indeces from 1 to the total number of dofs)
PhaseCondCoor = [ 1 ];

```

```
62 % Mat-file name where numerical results are save
63 matFile = 'NNModes_2.mat';
64
65 % The initial point is the NumMod-th linear normal mode of the linearized
66 % system, with y0 = (x0, v0) as modal shape and freq0 as natural frequency
67 NumMod = 2;
68 [y0,freq0] = initpoint(NumMod,1e-5);
69
70 % IF THE INITIAL POINT IS TAKEN FROM A CURVE CONTAINED IN A MAT-FILE
71 % UNCOMMENT THE FOLLOWING PARAMETER DEFINITIONS
72
73 % 'pt' is the index of the NNM taken from the mat-file as the initial point
74 % of a new branch
75
76 % load 'NNMmodes_2.mat'
77 % pt = 6;
78 % y0 = y11(pt,:);
79 % freq0 = freq11(pt);
80 % InitFreqStep = FreqStep(pt);
81 % MaxFreqStep = 1e-5;
82 % MinFreqStep = 1e-99;
83 % MaxPtsCurve = 5;
84 % MaxEnergy = 1e7;
85 % MaxFreq = 1;
86 % NumJacCalc = 5;
87 % ConvErrTol = 1e-5;
88 % AverageIter = 10;
89 % MaxIter = 15;
90 % Symmetry = 0;
91 % ShootRKRelTol = 1e-10;
92 % JacobRKRelTol = 1e-12;
93 % ShootRKAbsTol = 1e-12;
94 % JacobRKAbsTol = 1e-15;
95 % PhaseCondCoor = [ 1 ];
96 % matFile = 'NNModes_2_p.mat';
97
98
99 % CALL TO FUNCTION 'CURVECOMPUTATION' FOR THE NNMS BRANCH COMPUTATION
```

```
100
101 [y11,freq11,Energy,FloqMult,y01,freq01,TangVect,FreqStep,NumIter,ConvRelErr
    ] = curvecomputation(y0,freq0,RelDifferInt,InitFreqStep,MaxFreqStep,
    MinFreqStep,MaxPtsCurve,MaxEnergy,MaxFreq,NumJacCalc,ConvErrTol,
    AverageIter,MaxIter,Symmetry,ShootRKRelTol,JacobRKRelTol,ShootRKAbsTol,
    JacobRKAbsTol,PhaseCondCoor);
102
103 % OUTPUTS
104
105 % y11: state vector with displacement 'x' and velocity 'v', (x, v), of each
106 % NNM calculated by the algorithm, under a certain phase condition
107 % freq11: frequency of each NNMs
108 % Energy: total energy of each NNM
109 % FloqMult: Floquet multipliers calculated for each NNM
110 % y01: state vector of a new point given by each predictor phase
111 % freq01: frequency of a new point given by each predictor phase
112 % TangVect: Tangent vector calculated in each predictor phase
113 % FreqStep: Frequency stepsize of each predictor phase
114 % NumIter: Number of iterations of each corrector phase
115 % ConvRelErr: relative Newton-Raphson error archieve during each corrector
116 % phase
117
118 % RESULTS ARE SAVED
119
120 save(matFile)
```

C.2.1.3 fep.m

This is another executable m-file that takes the numerical results of the continuation algorithm to produce the FEP plots.

```
1 %-------------------------------------------------------------------------
2 % THIS EXECUTABLE M-FILE PRODUCES THE FREQUENCY-ENERGY PLOTS FROM THE
3 % OUTPUTS OF THE NNMS BRANCHES COMPUTATION
4
5 % WRITTEN BY ALEJANDRO SILVA, asilvaber@gmail.com
6
7 % DECEMBER 2015, MADRID, SPAIN
```

```
8  %-------------------------------------------------------------------------
9
10 figure
11 labels = cellstr(num2str([1:length(freq11)]'));
12 stableModes = [];
13 unstableModes = [];
14 allStable = 1;
15 for i=1:length(freq11)
16     if (abs(real(FloqMult(i,1)))<=1 && abs(real(FloqMult(i,2)))<=1)
17         stableModes = [stableModes, i];
18     else
19         unstableModes = [unstableModes, i];
20     end
21 end
22 if ~isempty(unstableModes), allStable = 0; end
23 plot(log10(Energy(stableModes,1)),freq11(stableModes),'bo')
24 hold on
25 plot(log10(Energy(unstableModes,1)),freq11(unstableModes),'ro')
26 text(log10(Energy(:,1)),freq11,labels,'VerticalAlignment','bottom','
    HorizontalAlignment','left')
27 grid on
28 title('Frequency-energy plot')
29 if allStable==0
30     legend('stable mode','unstable mode')
31 end
32 xlabel('log10(Energy)')
33 ylabel('Frequency (Hz)')
```

C.2.1.4 trajectories.m

This m-function returns a set of figures containing the modal curves and time-series plots of one of the NNMs contained among the solutions of the numerical computation –arrays $freq11$ and $y11$–, namely, the one with index $numPt$ in the solution arrays. This function can be executed directly in the Matlab command line with those input variables.

This function can be modified to suit to other models where different graphical outputs should be produced for each NNM.

```
%-------------------------------------------------------------------------
% THIS FUNCTION PRODUCES THE TIME-SERIES PLOTS AND MODAL CURVES WITH THE
% TRAJECTORIES OF EACH OF THE NNMS PREVIOUSLY COMPUTED

% WRITTEN BY ALEJANDRO SILVA, asilvaber@gmail.com

% DECEMBER 2015, MADRID, SPAIN
%-------------------------------------------------------------------------

function trajectories(freq11,y11,numPt)

% INPUTS

% y11: state vector with displacement 'x' and velocity 'v', (x, v), of each
% NNM calculated by the algorithm, under a certain phase condition
% freq11: frequency of each NNMs
% numPt: index of the NNM in 'y11' and 'freq11' whose trajectory will be
% computed

ModelPar = modelpar();

Klin = stiffmat(ModelPar);
Mlin = massmat(ModelPar);

T  = 1/freq11(numPt);
z0 = y11(numPt,:);

% [t,Y] = ode45(@eqs,[0 T],z0);
[t,Y] = ode15s(@eqs,[0 T],z0);

figure
plot(Y(:,1),Y(:,2));
hold on
plot(Y(1,1),Y(1,2),'bo');
hold on
plot(Y(length(t),1),Y(length(t),2),'bo');
```

```
grid on
title('Phase portrait')
xlabel('Main mass displacement u')
ylabel('Damping mass displacement w1')
figure
plot(Y(:,1),Y(:,3));
hold on
plot(Y(1,1),Y(1,3),'bo');
hold on
plot(Y(length(t),1),Y(length(t),3),'bo');
grid on
title('Phase portrait')
xlabel('Main mass displacement u')
ylabel('Main mass velocity u')
figure
plot(Y(:,2),Y(:,4));
hold on
plot(Y(1,2),Y(1,4),'bo');
hold on
plot(Y(length(t),2),Y(length(t),4),'bo');
grid on
title('Phase portrait')
xlabel('Damping mass displacement w1')
ylabel('Damping mass velocity w1')
figure
plot(t,Y(:,1));
grid on
title('Main mass time-series plot')
xlabel('time')
ylabel('Main mass displacement u')
figure
plot(t,Y(:,2));
grid on
title('Damping mass time-series plot')
xlabel('time')
ylabel('Damping mass displacement w1')
```

```
75 function dydt = eqs(~,y)
76
77 dydt(1:length(y)/2,1)     = y(length(y)/2+1:length(y),1);
78 dydt(length(y)/2+1:length(y),1) = Klin*y(1:length(y)/2,1) + nonlinear(y,
       ModelPar);
79 dydt(length(y)/2+1:length(y),1) = -Mlin\dydt(length(y)/2+1:length(y),1);
80
81 end
82
83 end
```

C.2.2 'models' folder, 'snapthrough' subfolder

If a new model was to be defined in this package, then create a new subfolder in the 'models' directory and copy and paste the four m-functions it must contain: *massmat.m*, *stiffmat.m*, *nonlinear.m* and *nlenergy.m*. Then, modify those functions to host the desired model.

C.2.2.1 modelpar.m

In this function the parameters of the model to analize are defined in an array called $ModelPar$.

A parameter that must be always present in any model in the $ModelPar$ array is the number of degrees of freedom of the system, $ndof$. It is the first element of the array.

```
1 %-------------------------------------------------------------------------
2 % THIS M-FILE CONTAINS THE PARAMETER VALUES OF THE SNAPTHROUGH ABSORBER
3 % MODEL
4
5 % WRITTEN BY ALEJANDRO SILVA, asilvaber@gmail.com
6
7 % DECEMBER 2015, MADRID, SPAIN
8 %-------------------------------------------------------------------------
9
```

```
function ModelPar = modelpar()

% Number of degrees of freedom of the model
ndof = 2;

% Main mass (of mass on first dof)
M    = 100;
% Absorber mass (of mass on second dof)
m    = 1;
% Stiffness of the attachments of the main mass to the ground
k1   = 100;
% Stiffness of the absorber springs/beams
k    = 1;
% Length of the absorber springs/beams under no deformation
L    = 1;
% Angle of the absorber springs/beams with respect to the direction of the
% first dof under no deformation
phi = 0.15;

ModelPar = [ndof,M,m,k1,k,L,phi];

end
```

C.2.2.2 massmat.m

This function returns the mass matrix of the model, [**M**]. The array *ModelPar* with the model parameters is the input variable of the function.

```
%-------------------------------------------------------------------------
% THIS IS THE EXECUTABLE M-FILE FOR THE COMPUTATION OF THE FRD PLOTS OF THE
% SINGLE DEGREE OF FREEDOM NONLINEAR OSCILLATOR

% WRITTEN BY ALEJANDRO SILVA, asilvaber@gmail.com

% DECEMBER 2015, MADRID, SPAIN
%-------------------------------------------------------------------------

```

```
10 function M = massmat(ModelPar)
11
12 M     = ModelPar(2);
13 m     = ModelPar(3);
14
15 M = diag([M m]);
16
17 end
```

C.2.2.3 stiffmat.m

This function returns the linear stiffness matrix of the model, [**K**].

```
1 %-------------------------------------------------------------------------
2 % THIS FUNCTION RETURNS THE LINEAR STIFFNESS MATRIX OF THE SNAPTHROUGH
3 % ABSORBER MODEL
4
5 % WRITTEN BY ALEJANDRO SILVA, asilvaber@gmail.com
6
7 % DECEMBER 2015, MADRID, SPAIN
8 %-------------------------------------------------------------------------
9
10 function K = stiffmat(ModelPar)
11
12 k1   = ModelPar(4);
13 k    = ModelPar(5);
14 phi  = ModelPar(7);
15
16 K = [k*cos(phi)^2+k1 -k*sin(phi)*cos(phi); -k*sin(phi)*cos(phi) 2*k*sin(phi
       )^2];
17
18 end
```

C.2.2.4 nonlinear.m

This function returns the nonlinear stiffness function, $\mathbf{f_{nl}}(\mathbf{x})$.

```
%-------------------------------------------------------------------------
% THIS FUNCTION RETURNS THE NONLINEAR STIFFNESS FUNCTION OF THE SNAPTHROUGH
% ABSORBER MODEL

% WRITTEN BY ALEJANDRO SILVA, asilvaber@gmail.com

% DECEMBER 2015, MADRID, SPAIN
%-------------------------------------------------------------------------

function Fnl = nonlinear(Y,ModelPar)

k   = ModelPar(5);
L   = ModelPar(6);
phi = ModelPar(7);

u  = Y(1);
w1 = Y(2);

Fnl = [k*u-k*L*cos(phi)+k*L/sqrt(1+w1^2/(L*cos(phi)-u)^2+2*w1*L*sin(phi)/(L
    *cos(phi)-u)^2+L^2*sin(phi)^2/(L*cos(phi)-u)^2)-k*cos(phi)^2*u+k*sin(phi
    )*cos(phi)*w1;
        2*k*w1-k*w1*L/sqrt(L^2-2*L*cos(phi)*u+u^2+w1^2+2*w1*L*sin(phi))-k*w1
    *L/sqrt(L^2+w1^2+2*w1*L*sin(phi))+2*k*L*sin(phi)-k*L^2*sin(phi)/sqrt(L
    ^2-2*L*cos(phi)*u+u^2+w1^2+2*w1*L*sin(phi))-k*L^2*sin(phi)/sqrt(L^2+w1
    ^2+2*w1*L*sin(phi))-2*k*sin(phi)^2*w1+k*sin(phi)*cos(phi)*u];

end
```

C.2.2.5 nlenergy.m

This function returns the nonlinear potential energy, L_{NL}, associated to the nonlinear stiffness component of the system.

```
%-------------------------------------------------------------------------
% THIS FUNCTION RETURNS THE NONLINEAR STIFFNESS ENERGY OF THE SNAPTHROUGH
% ABSORBER AT POSITION (Y(1), Y(2))

```

```
% WRITTEN BY ALEJANDRO SILVA, asilvaber@gmail.com

% DECEMBER 2015, MADRID, SPAIN
%--------------------------------------------------------------------------

function Lnl = nlenergy(Y,ModelPar)

k   = ModelPar(5);
L   = ModelPar(6);
phi = ModelPar(7);

u  = Y(1);
w1 = Y(2);

A0 = 2*L^2*sign(L);
A1 = L*cos(phi)*sign(cos(phi)) -L*sign(L)*cos(phi) -L*cos(phi);

Lnl = k * (-L*sqrt(L^2-2*L*cos(phi)*u+u^2+w1^2+2*w1*L*sin(phi)) -L*sqrt(L
    ^2+w1^2+2*w1*L*sin(phi)) +2*w1*L*sin(phi) +w1^2*cos(phi)^2 +sin(phi)*cos
    (phi)*w1*u +A0 +A1*u);

end
```

C.2.3 'continuer' folder

C.2.3.1 initpoint.m

A call to this function from the *contmain* files returns the initial state vector $-y0, freq-$ for the computation of a NNMs branch in case that the initial point is one of the linear normal modes of the system.

Its input parameters are the number of the mode whose branch is to be obtained, *mode*; and a shrinking coefficient to ensure the initial point is a low-energy state of the system $-factor-$.

```
%-------------------------------------------------------------------------
% THIS FUNCTION RETURNS AN INITIAL POINT FOR THE NNMS COMPUTATION FROM THE
% LINEAR MODES OF THE LINEARIZED MODEL AROUND THE EQUILIBRIUM POINT

% WRITTEN BY ALEJANDRO SILVA, asilvaber@gmail.com

% DECEMBER 2015, MADRID, SPAIN
%-------------------------------------------------------------------------

function [y0,freq] = initpoint(mode,factor)

disp(['Calculating initial point (normal mode ',num2str(mode),')...'])

ModelPar = modelpar();

ndof = ModelPar(1);

M = massmat(ModelPar);
K = stiffmat(ModelPar);

[Normalmodes,Omegasq] = eig(K,M,'chol','vector');
Natfreqs = sqrt(Omegasq)/(2*pi);

y0 = factor*[  zeros(1,ndof)  Normalmodes(:,mode)'  ];
freq = Natfreqs(mode);

disp(['INITIAL POINT (NORMAL MODE ',num2str(mode),') CALCULATED'])

end
```

C.2.3.2 curvecomputation.m

This is the core module of this package. It contains all the functions for tha execution of the continuation algorithm. Its master function is *curvecomputation*: called from the *contmain* files, it returns the numerical results of the computations.

For reasons of time and space I have left the file uncommented. Anyone interested in a detailed insight on how this module works can contact me.

```
%-------------------------------------------------------------------------
% THIS M-FILE CONTAINS ALL THE FUNCTIONS THAT CALCULATE THE NNMS BRANCHES
% WITH THE CONTINUATION METHOD GIVEN AN INITIAL SOLUTION AND THE CONTROL
% PARAMETERS SPECIFIED IN THE EXECUTABLE

% 'CURVECOMPUTATION' IS THE FUNCTION CALLED FROM THE MAIN EXECUTABLE FILE
% WHICH WILL RETURN THE ARRAYS WITH THE RESULTS OF THE COMPUTATIONS PLUS
% SOME BY-PRODUCTS OF THE ALGORITHM

% WRITTEN BY ALEJANDRO SILVA, asilvaber@gmail.com

% DECEMBER 2015, MADRID, SPAIN
%-------------------------------------------------------------------------

function [y11,freq11,Energy,FloqMult,y01,freq01,TangVect,FreqStep,NumIter,
    ConvRelErr] = curvecomputation(y0,freq0,RelDifferInt,InitFreqStep,
    MaxFreqStep,MinFreqStep,MaxPtsCurve,MaxEnergy,MaxFreq,NumJacCalc,
    ConvErrTol,AverageIter,MaxIter,Symmetry,ShootRKRelTol,JacobRKRelTol,
    ShootRKAbsTol,JacobRKAbsTol,PhaseCondCoor)

y11(1,:) = y0;
freq11(1) = freq0;
Energy = energy(y0);
FreqStep(1) = InitFreqStep;

for i=1:MaxPtsCurve
    if i>1
        InitTangVect = [];
        if HalvedFreqStep==0
            FreqStep(i) = (AverageIter/NumIter(i-1,:)) * FreqStep(i-1);
            if (FreqStep(i)>0 && FreqStep(i)>MaxFreqStep), FreqStep(i) =
    MaxFreqStep; end
            if (FreqStep(i)>0 && FreqStep(i)<MinFreqStep), FreqStep(i) =
    MinFreqStep; end
            if (FreqStep(i)<0 && FreqStep(i)<-MaxFreqStep), FreqStep(i) = -
    MaxFreqStep; end
```

```
30            if (FreqStep(i)<0 && FreqStep(i)>-MinFreqStep), FreqStep(i) = -
    MinFreqStep; end
31        else
32            FreqStep(i) = FreqStep(i-1);
33        end
34    end
35    disp(['COMPUTING POINT ',num2str(i)])
36    y00 = y11(i,:);
37    freq00 = freq11(i);
38    ConvergenceOK = 0;
39    HalvedFreqStep = 0;
40    MinStepReached = 0;
41    [TangVect(i,:),Fi] = predictor(y00,freq00,RelDifferInt,Symmetry,
    JacobRKRelTol,JacobRKAbsTol,PhaseCondCoor);
42    FloqMult(i,:) = floquet(Fi);
43    if i>1, FreqStep(i) = stepdircontr(TangVect(i,:),TangVect(i-1,:),
    FreqStep(i)); end
44    while ConvergenceOK==0
45        disp(['Frequency stepsize: ',num2str(FreqStep(i))])
46        [y01(i,:),freq01(i)] = predictorpoint(y00,freq00,TangVect(i,:),
    FreqStep(i));
47        [y11(i+1,:),freq11(i+1),NumIter(i,:),ConvergenceOK,ConvRelErr(i,:)]
    = corrector(y01(i,:),freq01(i),TangVect(i,:),RelDifferInt,MaxIter,
    NumJacCalc,ConvErrTol,Symmetry,ShootRKRelTol,JacobRKRelTol,ShootRKAbsTol
    ,JacobRKAbsTol,PhaseCondCoor);
48        if ConvergenceOK==0
49            disp('The stepsize is halved before repeating corrector phase')
50            FreqStep(i) = FreqStep(i)/2;
51            HalvedFreqStep = 1;
52            if abs(FreqStep(i))<MinFreqStep
53                disp('CONVERGENCE ABOVE THE MINIMUM STEPSIDE CANNOT BE
    REACHED')
54                disp('COMPUTATION IS TERMINATED')
55                MinStepReached = 1;
56                break;
57            end
58        end
59    end
```

```
60      if MinStepReached, break; end
61      if i==MaxPtsCurve
62          [~, Fi] = jacobian(y11(i+1,:),freq11(i+1),RelDifferInt,Symmetry,
        JacobRKRelTol,JacobRKAbsTol,PhaseCondCoor);
63          FloqMult(i+1,:) = floquet(Fi);
64      end
65      Energy(i+1,:) = energy(y11(i+1,:));
66      disp(['Energy at point ',num2str(i),': ',num2str(Energy(i+1))])
67      disp(['Frequency at point ',num2str(i),': ',num2str(freq11(i+1))])
68      if MaxEnergy<Energy(i+1)
69          disp('THE ENERGY LIMIT HAS BEEN REACHED')
70          [~, Fi] = jacobian(y11(i+1,:),freq11(i+1),RelDifferInt,Symmetry,
        JacobRKRelTol,JacobRKAbsTol,PhaseCondCoor);
71          FloqMult(i+1,:) = floquet(Fi);
72          break;
73      elseif MaxFreq<freq11(i+1)
74          disp('THE FREQUENCY LIMIT HAS BEEN REACHED')
75          [~, Fi] = jacobian(y11(i+1,:),freq11(i+1),RelDifferInt,Symmetry,
        JacobRKRelTol,JacobRKAbsTol,PhaseCondCoor);
76          FloqMult(i+1,:) = floquet(Fi);
77          break;
78      end
79  end
80
81  disp('END OF CURVE COMPUTATION')
82
83  end
84
85  function [TangVect,Fi] = predictor(y00,freq00,RelDifferInt,Symmetry,
        JacobRKRelTol,JacobRKAbsTol,PhaseCondCoor)
86
87  disp('START OF PREDICTOR PHASE')
88
89  ndof = length(y00)/2;
90
91  disp('Calculating jacobian...')
92  [dH, Fi] = jacobian(y00,freq00,RelDifferInt,Symmetry,JacobRKRelTol,
        JacobRKAbsTol,PhaseCondCoor);
```

```
93  dF = [dH; zeros(1,2*ndof) 1];
94  disp('Jacobian calculation terminated')
95
96  b = [zeros(1,2*ndof+length(PhaseCondCoor)) 1]';
97
98  disp('Calculating tangent vector...')
99  TangVect = dF\b;
100
101 disp('END OF PREDICTOR PHASE')
102
103 if TangVect(length(TangVect))<0.9
104     disp('WARNING: The frequency component of the tangent vector is
        significatively smaller than one. Jacobian matrix could be botched')
105 end
106
107 end
108
109 function [y11,freq11,NumIter,ConvergenceOK,ConvRelErr] = corrector(y01,
        freq01,TangVect,RelDifferInt,MaxIter,NumJacCalc,ConvErrTol,Symmetry,
        ShootRKRelTol,JacobRKRelTol,ShootRKAbsTol,JacobRKAbsTol,PhaseCondCoor)
110
111 disp('START OF CORRECTOR PHASE')
112 ndof = length(y01)/2;
113
114 i = 1;
115 y11 = y01;
116 freq11 = freq01;
117 ConvRelErrPres = 1e99;
118 while 1
119     disp(['Iteration ',num2str(i)])
120     disp('Calculating value of shooting function in present point...')
121     H01 = shooting(y11,1/freq11,Symmetry,ShootRKRelTol,ShootRKAbsTol,
        PhaseCondCoor);
122     ConvRelErrPast = ConvRelErrPres;
123     ConvRelErrPres = norm(H01) / norm(y11);
124     disp(['Present convergence error: ',num2str(ConvRelErrPres)])
125     if ConvRelErrPres<ConvErrTol,
126         disp('THE CONVERGENCE LIMIT HAS BEEN REACHED');
```

```
127         disp('END OF CORRECTOR PHASE')
128         ConvergenceOK = 1;
129         ConvRelErr = ConvRelErrPres;
130         break
131     end
132     if i==MaxIter
133         disp('THE ITERATION NUMBER LIMIT HAS BEEN REACHED');
134         ConvergenceOK = 0;
135         ConvRelErr = ConvRelErrPres;
136         break
137     end
138     if ConvRelErrPres>ConvRelErrPast,
139         disp('CONVERGENCE BEYOND TOLERANCE IS NOT POSSIBLE');
140         y11 = y01;
141         freq11 = freq01;
142         ConvergenceOK = 0;
143         ConvRelErr = ConvRelErrPast;
144         break
145     end
146     y01 = y11;
147     freq01 = freq11;
148     if i<NumJacCalc,
149         disp('Calculating jacobian...');
150         [dH,~] = jacobian(y01,freq01,RelDifferInt,Symmetry,JacobRKRelTol,
    JacobRKAbsTol,PhaseCondCoor);
151         dF = [dH; TangVect];
152         delta = -dF\[H01 0]';
153         disp('Jacobian has been successfully computed')
154     else disp(['Jacobian frozen after iteration ',num2str(NumJacCalc)]);
155     end
156     disp('Obtaining next point on Newton-Raphson iteration...')
157     y11 = y01 + delta(1:2*ndof,1)';
158     freq11 = freq01 + delta(2*ndof+1,1);
159     i = i+1;
160 end
161
162 NumIter = i;
163
```

```
164 end
165
166 function H = shooting(y0,T,Symmetry,ShootRKRelTol,ShootRKAbsTol,
        PhaseCondCoor)
167
168 ModelPar = modelpar();
169
170 ndof = ModelPar(1);
171
172 M = massmat(ModelPar);
173
174 if Symmetry==0
175     tspan = [   0, T   ];
176 else
177     tspan = [   0, T/2   ];
178 end
179
180 MM = [   eye(ndof,ndof)   zeros(ndof,ndof)
181              zeros(ndof,ndof)   M   ];
182
183 options = odeset('mass',MM,'RelTol',ShootRKRelTol,'AbsTol',ShootRKAbsTol);
184 [t,Y] = ode15s(@model,tspan,y0,options,ModelPar);
185
186 if Symmetry==0
187     S = Y(length(t),:) - y0;
188 else
189     S = Y(length(t),:) + y0;
190 end
191
192 H = S;
193 for i=1:length(PhaseCondCoor)
194     H = [H y0(PhaseCondCoor(i))];
195 end
196
197 end
198
199 function FreqStepNew = stepdircontr(TVectNow,TVectPrev,FreqStep)
200
```

```
201 if TVectNow*TVectPrev'<0
202     disp('A fold has been detected, the stepsize will change sign')
203     FreqStepNew = -FreqStep;
204 else
205     FreqStepNew = FreqStep;
206 end
207
208 end
209
210 function [y01,freq01] = predictorpoint(y00,freq00,TangVect,FreqStep)
211
212 ndof = length(y00) / 2;
213
214 disp('Obtaining next point on linearized curve...')
215 y01 = y00 + FreqStep*TangVect(1:2*ndof);
216 freq01 = freq00 + FreqStep*TangVect(2*ndof+1);
217
218 end
219
220 function dy = model(~,y,ModelPar)
221
222 ndof = ModelPar(1);
223
224 K   = stiffmat(ModelPar);
225 Fnl = nonlinear(y,ModelPar);
226
227 x = y(1:ndof);
228 v = y(ndof+1:2*ndof);
229
230 dx = v;
231 dv = -(K*x+Fnl);
232
233 dy = [dx; dv];
234
235 end
236
237 function [dH,Fi] = jacobian(y0,freq,RelDifferInt,Symmetry,JacobRKRelTol,
      JacobRKAbsTol,PhaseCondCoor)
```

```
238
239 tic
240
241 ModelPar = modelpar();
242
243 ndof = ModelPar(1);
244 T = 1/freq;
245
246 dy = zeros(2*ndof,2*ndof);
247 dS = zeros(2*ndof,2*ndof+1);
248
249 M = massmat(ModelPar);
250
251 if Symmetry==0
252     tspan = [   0, T   ];
253 else
254     tspan = [   0, T/2   ];
255 end
256
257 dy0 = zeros(1,8*ndof^2+2*ndof);
258
259 for i=1:2*ndof+1
260     if i==2*ndof+1
261         dy0(length(dy0)-2*ndof+1:length(dy0)) = y0;
262         break;
263     end
264     dy0(4*ndof*(i-1)+1:4*ndof*i-2*ndof) = y0;
265     dy0(4*ndof*(i-1)+i) = dy0(4*ndof*(i-1)+i) + RelDifferInt*norm(y0);
266     dy0(4*ndof*i-2*ndof+1:4*i*ndof) = y0;
267     dy0(4*ndof*i-2*ndof+i) = dy0(4*ndof*i-2*ndof+i) - RelDifferInt*norm(y0)
    ;
268 end
269
270 options = odeset('RelTol',JacobRKRelTol,'AbsTol',JacobRKAbsTol);
271 [Ty,DY] = ode15s(@modeldS,tspan,dy0,options,ModelPar,M);
272
273 for i=1:2*ndof
```

```
274     dy(:,i) = (1/(2*RelDifferInt*norm(y0))) * (DY(length(Ty),4*ndof*(i-1)
        +1:4*i*ndof-2*ndof)' - DY(length(Ty),4*i*ndof-2*ndof+1:4*i*ndof)');
275 end
276
277 Fi = dy(1:2*ndof,1:2*ndof);
278
279 if Symmetry==0
280     dS(:,1:2*ndof) = dy(:,1:2*ndof) - eye(2*ndof);
281     dSdf = model(Ty,DY(length(Ty),length(dy0)-2*ndof+1:length(dy0))',
        ModelPar) * (-1/(freq^2));
282 else
283     dS(:,1:2*ndof) = dy(:,1:2*ndof) + eye(2*ndof);
284     dSdf = model(Ty,DY(length(Ty),length(dy0)-2*ndof+1:length(dy0))',
        ModelPar) * (-1/(2*freq^2));
285 end
286 dSdf(ndof+1:2*ndof) = M \ dSdf(ndof+1:2*ndof);
287
288 dS(:,2*ndof+1) = dSdf;
289
290 dH = dS;
291 UnitMat = eye(2*ndof+1);
292 for i=1:length(PhaseCondCoor)
293     dH = [dH; UnitMat(PhaseCondCoor(i),:)];
294 end
295
296 disp(['Time lapsed for jacobian computation: ',num2str(toc)]);
297
298 end
299
300 function ddy = modeldS(~,dy,ModelPar,M)
301
302 ndof = ModelPar(1);
303
304 ddy = zeros(8*ndof^2+2*ndof,1);
305
306 for i=1:4*ndof+1
307     Fnl = nonlinear(dy(2*ndof*(i-1)+1:2*i*ndof),ModelPar);
308     K = stiffmat(ModelPar);
```

```
309       x = dy(2*ndof*(i-1)+1:2*i*ndof-ndof);
310       v = dy(2*i*ndof-ndof+1:2*i*ndof);
311       ddx = v;
312       ddv = -M \ (K*x+Fnl);
313       ddy(2*ndof*(i-1)+1:2*i*ndof) = [ddx; ddv];
314   end
315
316   end
317
318   function FloqMult = floquet(Fi)
319
320   disp('Calculating Floquet multipliers...')
321
322   FloqMult = eig(Fi,'vector')';
323
324   disp('FLOQUET MULTIPLIERS CALCULATED')
325
326   end
327
328   function Energy = energy(y)
329
330     ModelPar = modelpar();
331
332     ndof = ModelPar(1);
333
334     x = y(1:ndof);
335     v = y(ndof+1:2*ndof);
336
337    M = massmat(ModelPar);
338    K = stiffmat(ModelPar);
339     Energy = 0.5*v*M*v' + 0.5*x*K*x' + nlenergy(x,ModelPar);
340
341   end
```

C.3 Outputs

In this section I display the graphical results of the analysis of the snapthrough absorber with the Matlab codes I have presented. It is not the purpose of this Project to present a deep insight into the dynamics of this system. So I simply show a few plots describing very briefly the dynamical behavior of the snapthrough absorber given its modal analysis via NNMs.

The results conform to the following values of the model parameters of the absorber, defined in the *modelpar.m* file: $M = 100$, $m = 1$, $K_1 = 100$, $K = 1$, $L = 1$, $\phi = 0.15$.

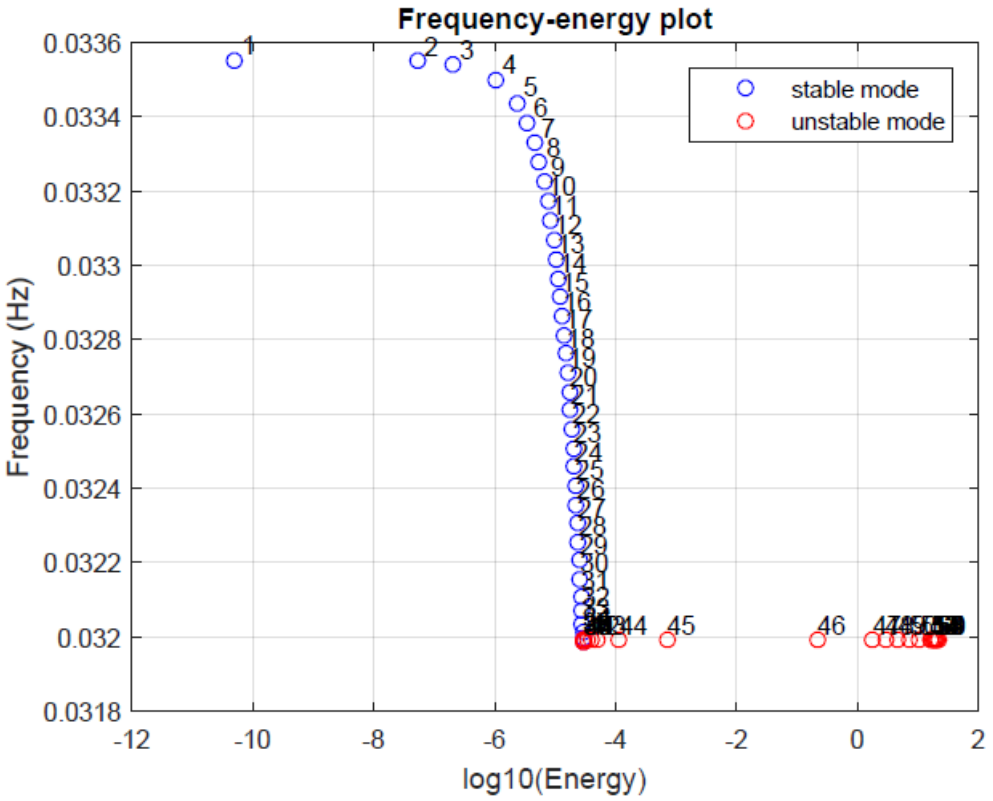


FIGURE C.2: Frequency-energy plot of the in-phase first mode branch of the snapthrough absorber. For the model parameters set in *modelpar* there is a negative frequency-energy dependence, though the decrease in frequency for larger energies is very slight. The unstable branch section is a 5:1 internal resonance as it can be seen in the following modal curves. This mode displays much greater oscillations in the mass absorber –degree of freedom two– than in the main mass –degree of freedom one–, and despite the model nonlinearity this proporcionality is mantained for larger energies. This makes this mode useful for vibration control and atenuation.

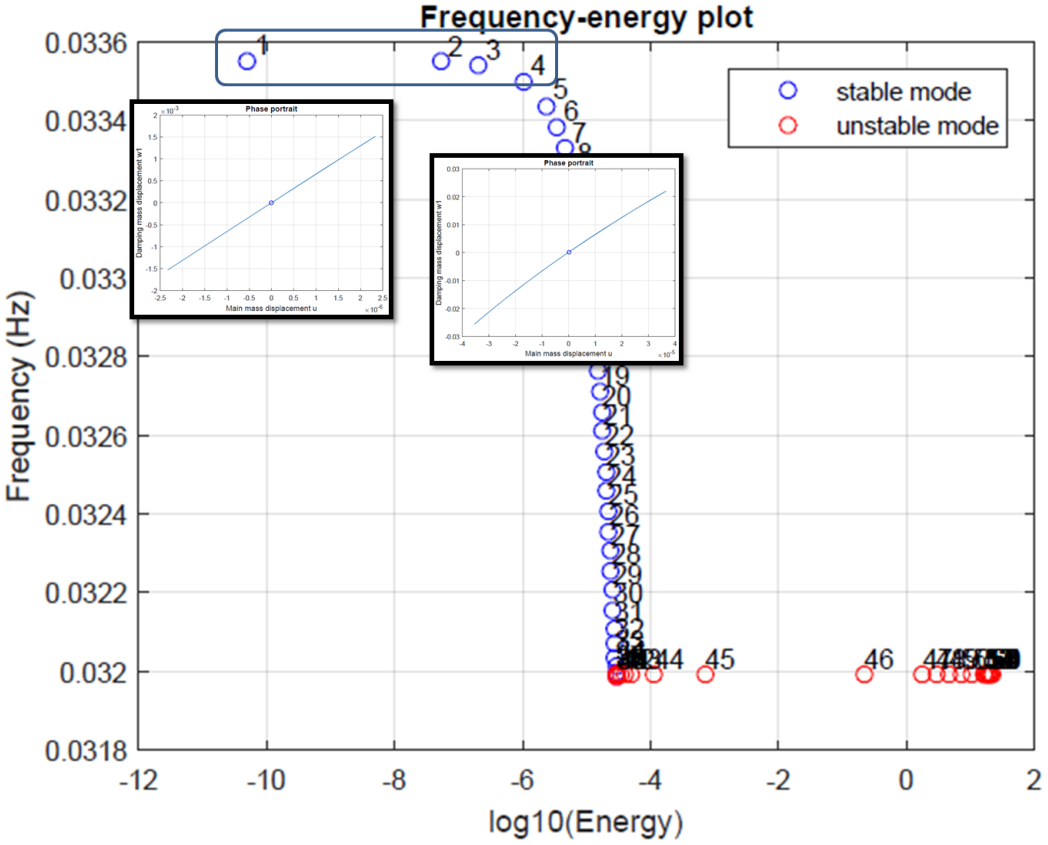


FIGURE C.3: Modal curves of the low-energy modes of the first NNMs branch. As the oscillation energy increases sinusoidallity and symmetry are lost.

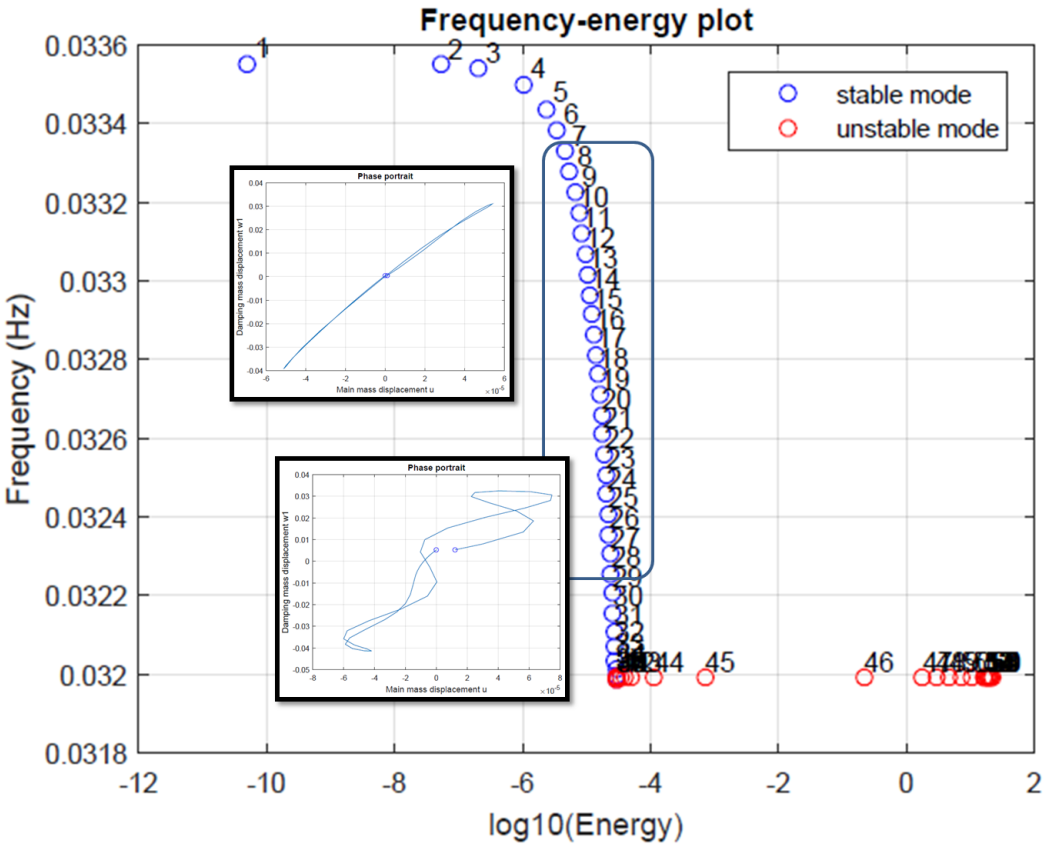


FIGURE C.4: These modal curves show the onset of a 5:1 internal resonance.

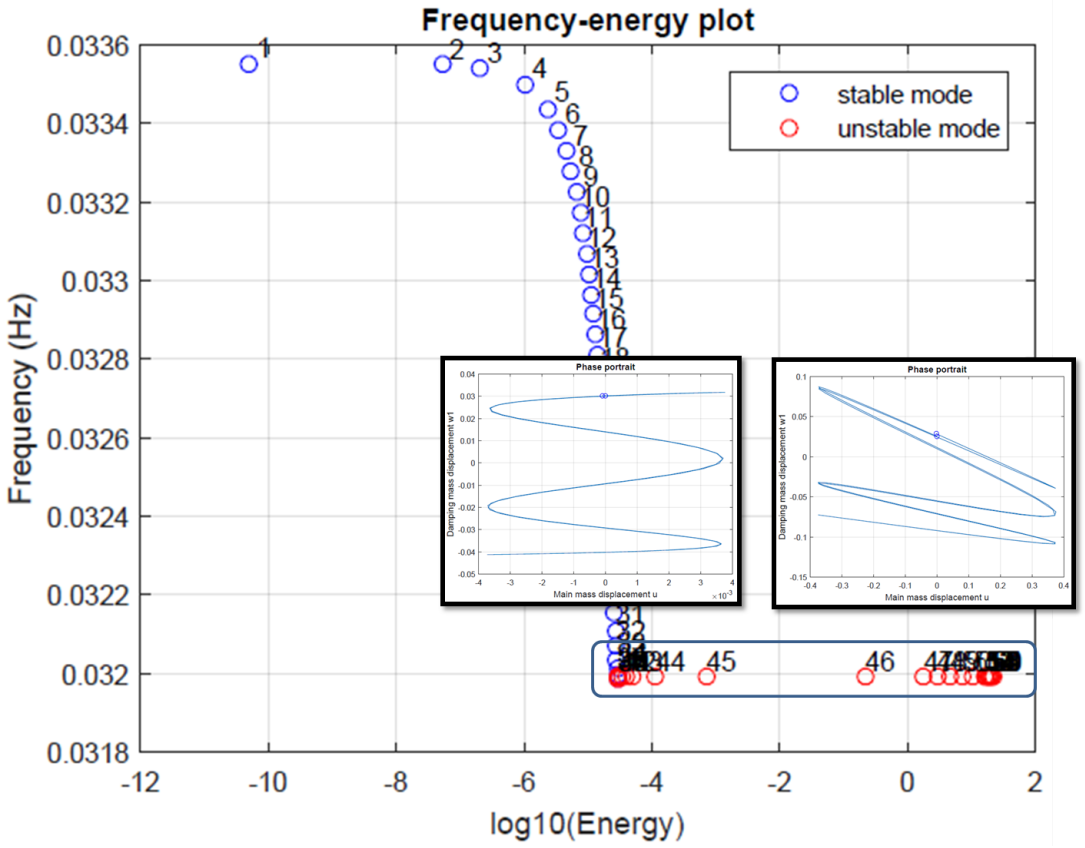


FIGURE C.5: At some point of the branch a stability change is detected. The frequency of the unstable branch remains energy-invariant. The point of stability change is suspicious of being a branching point. However, branching has been attempted without success. The oscillations remain periodic until they reach the unstable equilibrium point $W_1 = 0$. When that point is surpassed the computation results become faulty.

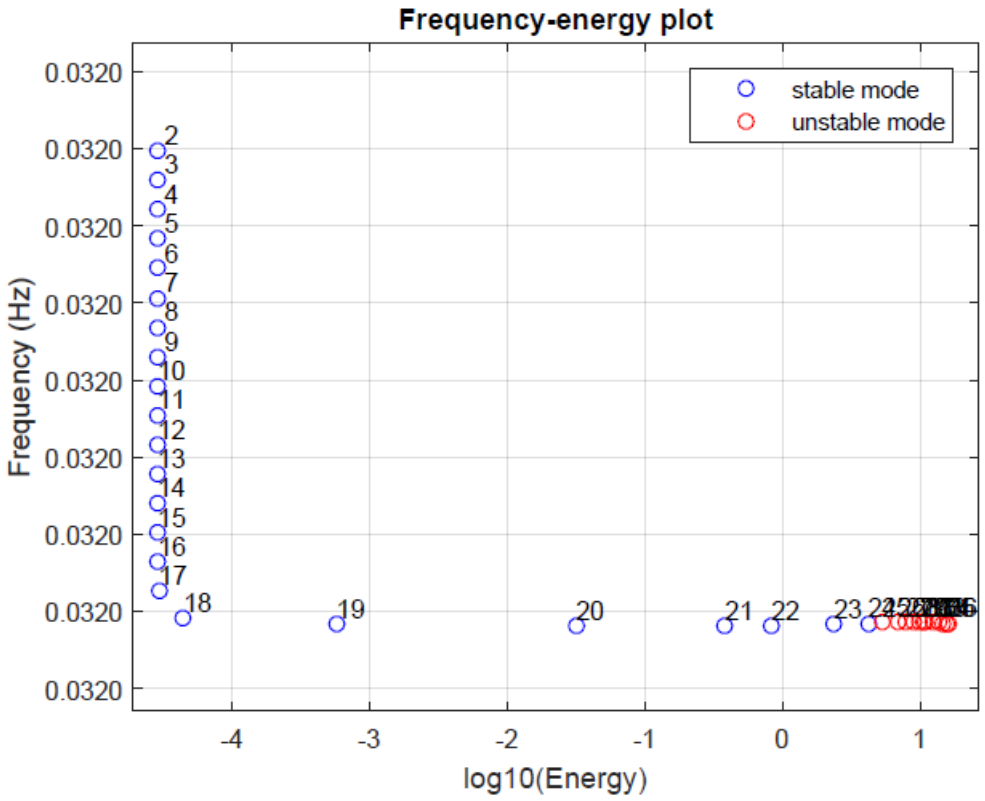


FIGURE C.6: However, if a more detailed computation of the branch is conducted around the suspected stability change we find out that there is no stability change in that point and that all the periodic trajectories are stable. There is only a stability change where the trajectories approach the unstable point –$W_1 = 0$–, making the results defective, probably for the unexistence of continuity in the NNMs branch.

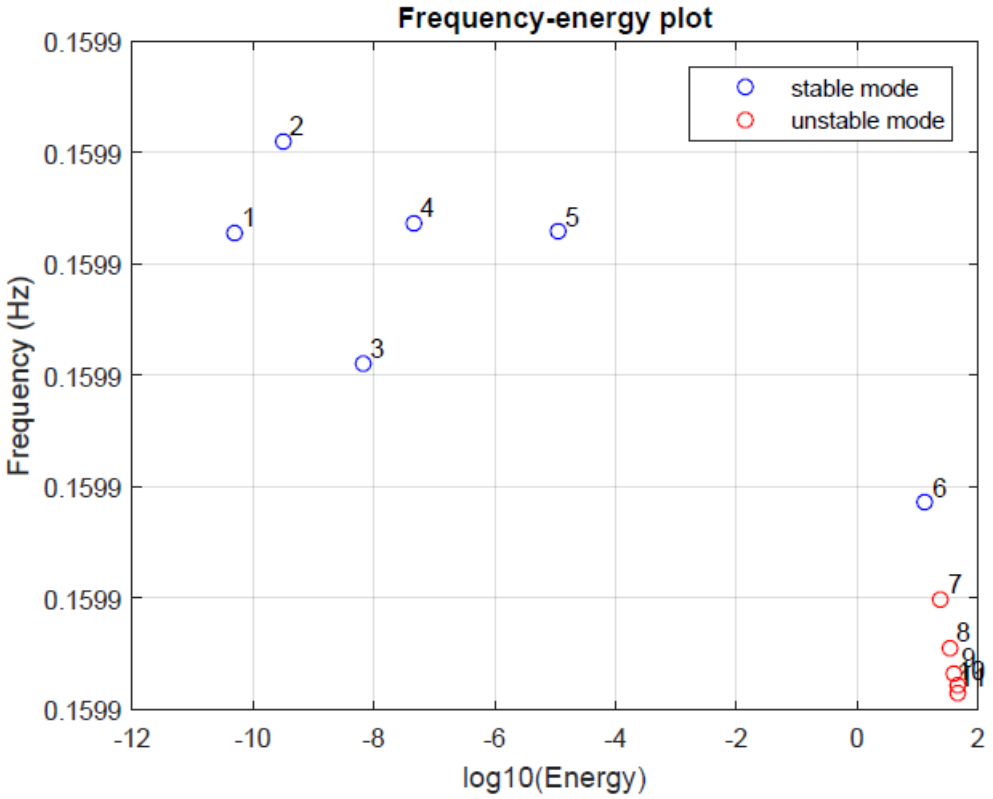


FIGURE C.7: Frequency-energy plot of the out-of-phase second mode branch of the snapthrough absorber. There is no frequency-energy dependence in this mode. The differences between the frequencies of the modes on display are due to numerical unaccuracies. Unlike the first mode, this mode is useless for vibration atenuation, because the second mass only absorbes a tiny amount of the total energy of oscillation of the whole system.

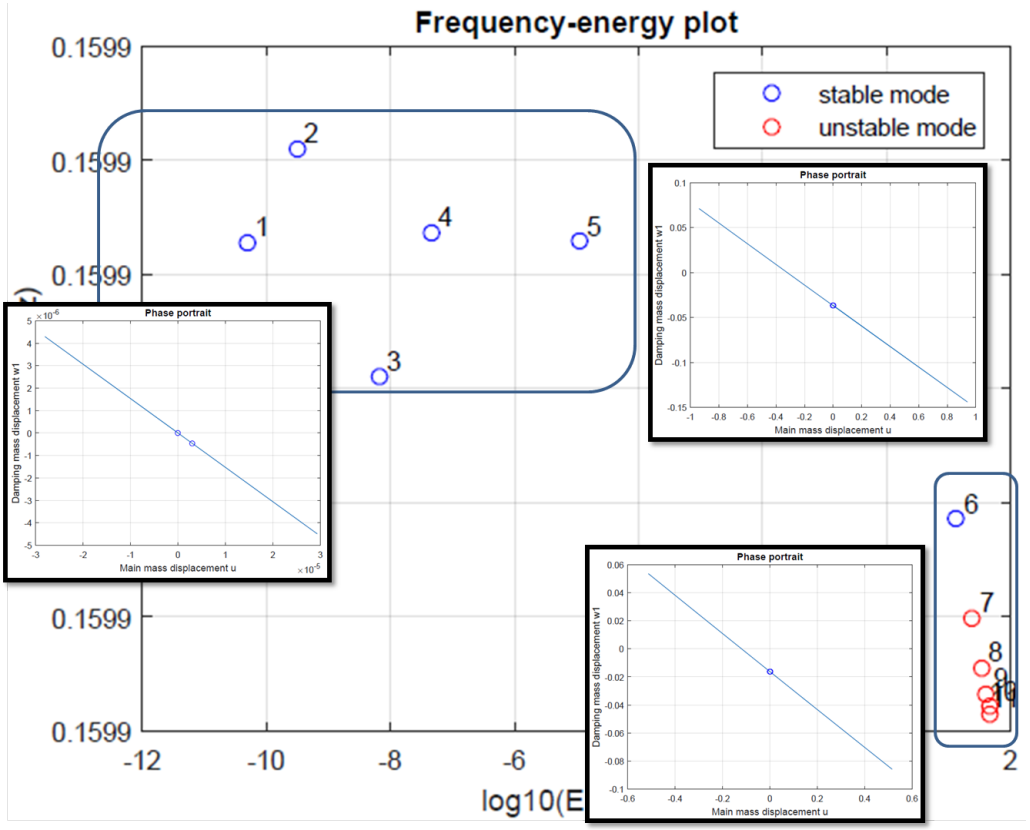


FIGURE C.8: On this second branch the displacement orbits remain straight on the phase space at any level of energy until the displacements of the first mass come close to one –$U = 1$–. At this energy level, Both stability and periodicity of the numerical solutions are lost, probably for the lack of numerical accuracy or for the unexistence of periodical solutions for higher energies –as in the case of the first modal branch–.

Bibliography

[1] M. Z. Kolovsky. *Nonlinear Dynamics of Active and Passive Systems of Protection*. Springer, 1st edition, 1999. ISBN 978-3-540-49413-9.

[2] Alexander F. Vakakis et al. *Nonlinear Targeted Energy Transfer in Mechanical and Structural Systems, volumes I and II*, volume 156 of *Solid Mechanics and its Applications*. Springer, 1st edition, 2008. ISBN 978-1-4020-9130-8.

[3] David Wagg and Simon Neild. *Nonlinear Vibration with Control for Flexible and Adaptive Structures*, volume 170 of *Solid Mechanics and its Applications*. Springer, 1st edition, 2010. ISBN 978-90-481-2837-2.

[4] *Advances in Energy Harvesting Methods*, volume 170 of *Solid Mechanics and its Applications*. Springer, 1st edition, 2013. ISBN 978-1-4614-5705-3.

[5] B.R. Mace R.Ramlan, M.J.Brennan and I. Kovacic. Potential benefits of a non-linear stiffness in an energy harvesting device. *Springer Journal of Nonlinear Dynamics*, 2009.

[6] Francesco Cottone. Introduction to vibration energy harvesting. slideshow, August 2011. URL http://www.nipslab.org/files/file/nips%20summer%20school%202011/Cottone%20Introduction%20to%20vibration%20harvesting.pdf.

[7] Y. Mikhlin and Sergei Mitrokhin. Nonlinear normal vibration modes and their applications in some applied problems. In *ENOC 2008 Electronic Library*. ENOC 2008, 2008. URL http://www.researchgate.net/profile/Yuri_Mikhlin/publication/237772562_NONLINEAR_NORMAL_VIBRATION_MODES_AND_THEIR_

APPLICATIONS_IN_SOME_APPLIED_PROBLEMS/links/0deec52d6c0cf19f04000000.pdf.

[8] *Modal Analysis*. Elvesier, 1st edition, 2001. ISBN 0-7506-5079-6.

[9] J. J. Stoker. *Nonlinear Vibrations in Mechanical and Electrical Systems*. Interscience, 4th printing edition, 1961.

[10] Ali Hasan Nayfeh. *Perturbation Methods*. Wiley, 2nd edition, 2004. ISBN 978-0-471-39917-9.

[11] Ali Hasan Nayfeh and Dean T. Mook. *Nonlinear Oscillations*. Wiley, 2nd edition, 1995. ISBN 0-471-12142-8.

[12] Alexander F. Vakakis et al. *Normal Modes and Localization in Nonlinear Systems*. Wiley, 1st edition, 1996. ISBN 0-471-13319-1.

[13] Gaetan Kerschen, editor. *Modal Analysis of Nonlinear Mechanical Systems*, volume 555 of *CIDM International center for Mechanical Sciences*. Springer, 1st edition, 2014. ISBN 978-3-7091-1791-0.

[14] C. Pierre D. Jiang and S.W. Shaw. Nonlinear normal modes for vibratory systems under harmonic excitation. *Journal of Sound and Vibration*, January 2005.

[15] S.W. Shaw and C. Pierre. Normal modes for non-linear vibratory systems. *Journal of Sound and Vibration*, 1(164), February 2009.

[16] Anindya Chatterjee. A brief introduction to nonlinear vibrations. lecture notes, February 2009. URL http://home.iitk.ac.in/~anindya/NLVnotes.pdf.

[17] J.C. Golinval G. Kerschen, M. Peeters and A.F. Vakakis. Nonlinear normal modes, part i: A useful framework for the structural dynamicist. 2008. URL http://www.ltas-vis.ulg.ac.be/cmsms/uploads/File/NnmsPartI_Revised.pdf.

[18] J.C. Golinval G. Kerschen, M. Peeters and A.F. Vakakis. Nonlinear normal modes, part i: An attempt to demystify them. In *SEM Proceedings Online Library*. IMAC XXVI, 2008. URL http://sem-proceedings.com/26i/sem.

org-IMAC-XXVI-Conf-s19p01-Nonlinear-Normal-Modes-Part-I-An-Attempt-Demystify-Them
pdf.

[19] G. Sérandour G. Kerschen M. Peeters, R. Vigié and J.C. Golinvar. Computation of nonlinear normal modes, part i: Numerical continuation in matlab, 2008. URL http://lib.physcon.ru/file?id=4aee6c206b34.

[20] G. Sérandour G. Kerschen M. Peeters, R. Viguié and J.C. Golinval. Nonlinear normal modes, part ii: Towards a practical computation using numerical continuation techniques. 2008. URL http://www.ltas-vis.ulg.ac.be/cmsms/uploads/File/NNMsPartII_Revised.pdf.

[21] G. Sérandour G. Kerschen M. Peeters, R. Viguié and J.C. Golinval. Nonlinear normal modes, part ii: Practical computation using numerical continuation techniques. In *SEM Proceedings Online Library*. IMAC XXVI, 2008. URL http://sem-proceedings.com/26i/sem.org-IMAC-XXVI-Conf-s19p02-Nonlinear-Normal-Modes-Part-II-Practical-Computation-Us
pdf.

[22] M. Peeters G. Kerschen and J.C. Golinval. Nonlinear modal analysis of a full-scale aircraft. *Journal of Aircraft*, 2013. URL http://www.ltas-vis.ulg.ac.be/cmsms/uploads/File/60_aiaa_press.pdf.

[23] L. Renson J.P.Nöel and G. Kerschen. Dynamics of a strongly nonlinear spacecraft structure, part i: Experimental identification. 2014. URL http://orbi.ulg.ac.be/bitstream/2268/165352/1/Noel%20-%20Dynamics%20of%20a%20strongly%20nonlinear%20spacecraft%20structure%20-%20Part%20I%20Experimental%20identification.pdf.

[24] G. Sérandour G. Kerschen M. Peeters, R. Viguié and J.C. Golinval. Dynamics of a strongly nonlinear spacecraft structure, part ii: Modal analysis. 2014. URL http://orbi.ulg.ac.be/bitstream/2268/165351/1/Renson%20-%20Dynamics%20of%20a%20strongly%20nonlinear%20spacecraft%20structure%20-%20Part%20II%20Modal%20analysis.pdf.

[25] J.P. Noël L. Renson and G. Kerschen. Complex dynamics of a nonlinear aerospace structure: Numerical continuation and normal modes. *Journal of Nonlinear Dynamics*, 2015.

[26] G. Kerschen M. Peeters, L. Rend¡son. Nnm computation in matlab. slideshow, October 2011.

[27] S. Graham Kelly. *Mechanical Vibrations Theory and Applications*. Cengage Learning, 2012. ISBN 978-1-4390-6212-9.

[28] *Computatonal Galerkin Methods*. Series in Computational Physics. Springer, 1st edition, 1984. ISBN 978-3-642-85949-6.

[29] R.M. Rosenberg. *Advances in Applied Mechanics*, (9).

[30] S.W. Shaw and C. Pierre. *Journal of Sound and Vibration*, (150):170–173.

[31] Régis Viguié. *Tuning Methodology of Nonlinear Vibration Absorbers Coupled to Nonlinear Mechanical Systems*. PhD thesis, University of Liège, September 2010. URL http://bictel.ulg.ac.be/ETD-db/collection/available/ULgetd-10282010-124920/unrestricted/phD_Thesis_Regis_Viguie.pdf.

[32] Tom J. Kázmierski. *Energy Harvesting Systems: Principles, Modeling and Applications*. Springer, 2011. ISBN 978-1-4419-7565-2.

[33] Yaowen Yang Lihua Tang and Chee Kiong Soh. Broadband vibration energy harvesting techniques. In Niell Elvin and Alper Erturk, editors, *Advances in Energy Harvesting Methods*, volume 170 of *Solid Mechanics and its Applications*, pages 17–61. Springer, 1st edition, 2013.

[34] G. Kerschen G. Sérandour, M. Peeters and J.C. Golinvar. Computation of nonlinear normal modes, part ii: Numerical continuation in auto, 2008. URL http://lib.physcon.ru/file?id=88c986c2495f.

[35] Eugene L. Allgower and Kurt Georg. *Introduction to Numerical Continuation Methods*, volume 45 of *Classics in Applied Mathematics*. Siam, 1st edition, 2003. ISBN 0-89871-544-X.

[36] Rüdiger Segdel. *Introduction to Numerical Continuation Methods*, volume 5 of *Interdisciplinary Applied Mathematics*. Springer, 3rd edition, 2010. ISBN 978-1-4419-1739-3.

[37] Y. V. Mikhlin K. V. Avramov. Snapthrough truss as a vibration absorber. *Journal of Sound and Vibration*, 2004.

[38] Y. V. Mikhlin K. V. Avramov. Snapthrough truss as a absorber of forced oscillations. *Journal of Sound and Vibration*, 2005. URL http://users.kpi.kharkov.ua/ndynamics/index_files/Papers%5CAvramov&MikhlinJournalSoundVibration%28in%20press%29.pdf.