TÉCNICO
LISBOA

FE Simulation Applied to the Vibration of Hydraulic Pipes

Eduardo Machado da Conceição Rodrigues Pereira

Thesis to obtain the Master of Science Degree in

Mechanical Engineering

Supervisors: Prof. Miguel António Lopes de Matos Neves
Dr. Olavo Mecias da Silva Junior

Examination Committee

Chairperson: Prof. Paulo Rui Alves Fernandes
Supervisor: Prof. Miguel António Lopes de Matos Neves
Member of the Committee: Prof. Hugo Filipe Diniz Policarpo

December 2021

To my family for their unconditional support,

Acknowledgments

First and foremost, I would like to thank Professor Miguel Matos Neves from Instituto Supeior Técnico for having accepted me to carry out this thesis and for all unconditional assistance provided since the beginning

I would also like to thank Dr. Olavo M. Silva, researcher from Federal University of Santa Catarina, for the help in crucial stages of this thesis with the knowledge and experience in the acoustics and vibration areas. I also thank Dr. Olavo for the industrial example you kindly provided to test and to verify the work preformed.

It is also important to recognize the contribution of the different professors and colleagues from whom I have learned in the past few years and thus helped me reach my goals.

Finally, I would like to thank my family and friends for all the love and encouragement throughout this process.

Resumo

A vibração estrutural em um sistema de tubagens hidráulico ou de transporte de gás pode ser fortemente afetada devido à excitação de alta pressão do fluido dentro da tubagem. Compreender o comportamento da vibração é o primeiro passo para controlar as vibrações nas tubagens com tecnologias comuns de controle de vibração para garantir a segurança do sistema de tubagens e maquinaria associada. As tecnologias comuns de controle de vibração demonstraram ser eficazes em estruturas típicas como estruturas aeroespaciais e estruturas de transporte de gás. Tanto a vibração acústica quanto a vibração induzida por fluxo de fluido podem ser uma causa de falha e os dois tipos de vibração usam métodos diferentes para realizar a análise de vibração.

A análise de vibração estrutural de dutos tridimensionais transportando fluido é modelada pelo método dos elementos finitos com base nas teorias de viga de Euler-Bernoulli e Timoshenko. O método da matriz de transferência usando ondas de pressão acústica unidimensional é utilizado para representar a interação acústica-estrutura numa tubagem como um sistema acoplado. Esta interação fluido-estrutura em sistemas de tubagem é modelada pela teoria do golpe de ariete estendida para o fluido e teoria de vigas para a estrutura da tubagem.

Um código-fonte foi desenvolvido em MATLAB para prever o comportamento dinâmico de sistemas de tubos simples e complexos tridimensionais, sujeitos a cargas externas harmônicas e acústicas. A partir dos campos de deslocamento e pressão dentro da tubagem, a localização dos apoios pode ser otimizada para reduzir as amplitudes de vibração.

Palavras-chave: Método dos elementos finitos, Método da matrix de transferência, Interação fluido-estrutura, Acústica, Teoria de vigas de Euler-Bernoulli, Teoria de vigas de Timoshenko

Abstract

The structural vibration in a hydraulic or gas pipeline system can be strongly affected due to multi-source excitation of high fluid pressure fluctuation inside the pipes. Understanding the vibration behavior is the first step to controlling the vibrations in the pipeline with common vibration control technologies to ensure the safety of the pipe system and machinery. The common vibration control technologies have been demonstrated to be effective in typical structures as aerospace structures and gas pipeline structures. Both acoustic induced vibration and flow induced vibration can be a cause of failure and both types of vibration use different methods to perform the vibration analysis.

The structural vibration analysis of three-dimensional pipelines conveying fluid is modeled by the finite element method (FEM) based on Euler-Bernoulli and Timoshenko beam theories and the transfer matrix method (TMM) used for the one-dimensional acoustic pressure waves solution and to represent the acoustic-structure interaction in a pipeline as a weak or one-way coupled system. This fluid-structure interaction in compliant piping systems is modeled by extended water hammer theory for the fluid and beam theory for the pipe structure.

A source code was developed in MATLAB to predict the dynamic behavior of three-dimensional complex and simple pipe systems subjected to harmonic and acoustic external loads or resonant frequencies of complex pipe networks conveying steady fluid flow. From the displacement and pressure fields optimal layout technique of pipeline using clamps can be archived to reduce vibration amplitudes.

Keywords: Finite element method, Transfer matrix method, Fluid-structure interaction, Acoustics, Euler-Bernoulli beam theory, Timoshenko beam theory

Contents

List of Tables

List of Figures

Nomenclature

ν	Poisson coefficient
ρ	Material density
ρ_f	Fluid density
A	Pipe area
A_i	Internal pipe area
c_0	Fluid speed of sound corrected
c_f	Fluid speed of sound
D_e	Pipe external diameter
D_i	Pipe internal diameter
E	Young's modulus
I_x	Torsional constant
I_y	Second moment of area around y axis
I_z	Second moment of area around z axis
K_f	Fluid bulk modulus
p	Fluid pressure
q	Fluid volume velocity
$\mathbf{C}$	Global damping matrix
$\mathbf{C}^e$	Element damping matrix
$\mathbf{K}$	Global stiffness matrix
$\mathbf{K}^e$	Element stiffness matrix
$\mathbf{M}$	Global mass matrix
$\mathbf{M}^e$	Element mass matrix
$\mathbf{T}$	Transformation matrix

Chapter 1

Introduction

1.1 Motivation

Pipelines are essential structural systems to transport all types of fluids, liquids and gases, and can range from the very simple ones to a more complex ones. They can be simple as a single pipe conveying fluid from one reservoir to another and more elaborate as a complex transport of natural oil or gas in a major metropolitan area, containing several kinds of pumps, valves, branches and supporting parts. Pipelines can also be made with small diameters pipes (6 mm) like in the hydraulic pipelines in airplanes and cars, and larger diameters (0.5 m) for long distance gas transport, for example. One of the major causes of failures or downtime is associated with vibration of these structures and controlling these vibrations is the challenging work to ensure a normal operation of the pipeline or the flight safety of an aircraft. Excessive vibration, that usually involves the lateral vibration of the pipeline and the shell wall radial vibrations. At low frequencies, pipe vibration occurs laterally, like a beam, and at higher frequencies, the pipe shell wall starts to vibrate radially across its cross-section.

Piping configuration, number and type of supports, span length, or material affect vibration levels and can be modified to make the system acceptable. To understand what can be changed it is important to know all the principle mechanisms and excitation sources that excites the structure.

1.2 Topic Overview

The pipelines are subjected to various types of vibration in their lifetime. Pipelines can transport gas for supplying energy by combustion process and for other processes that involve high pressure gas transportation through the pipes and other related components. Another practical use of pipelines is in aircraft hydraulic systems. This is a typical high-pressure and high-speed system that includes supporting parts, such as brackets or clamps, where, due to multi-source excitation of high fluid pressure fluctuations can cause unwanted vibrations [1]. In aeronautical applications, space constraint is rigorous and given the large number of pipelines (Figure 1.1) collision between adjacent pipes also leads to the potential damage. The displacements due to vibrations may result in material fatigue damage to the pipeline structure and damage to pipeline supports. These vibrations can be lateral pipeline vibration due to low frequency or high frequency piping shell wall vibrations that can be caused by excitation of circumferential radial frequency modes [2] (Figure 1.2). Failure modes of hydraulic pipeline system includes excessive vibration and fatigue. This occurs due to pipeline resonance. That is, when the fluid pulsation frequency or external excitation are similar to the pipeline natural frequencies. During this resonance, damage may occur due to high vibration amplitudes, and consequently stress, and then a crack may open and propagate.

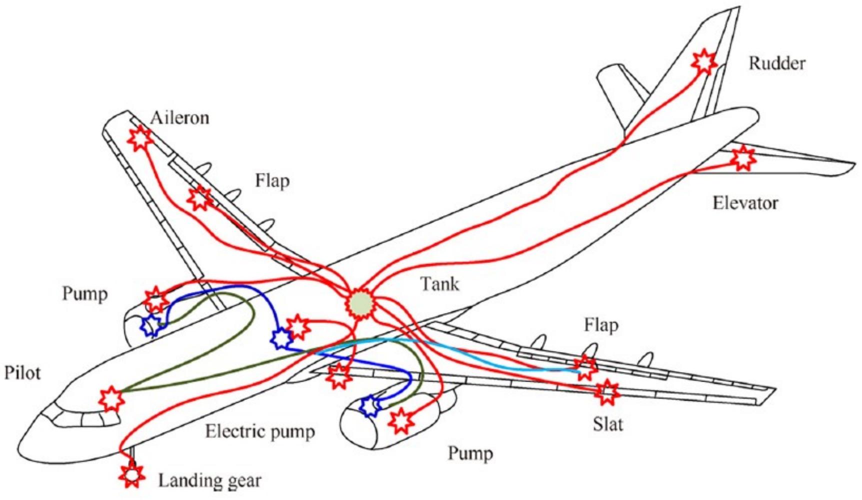


Figure 1.1: Distribution of aircraft hydraulic pipeline system [1]

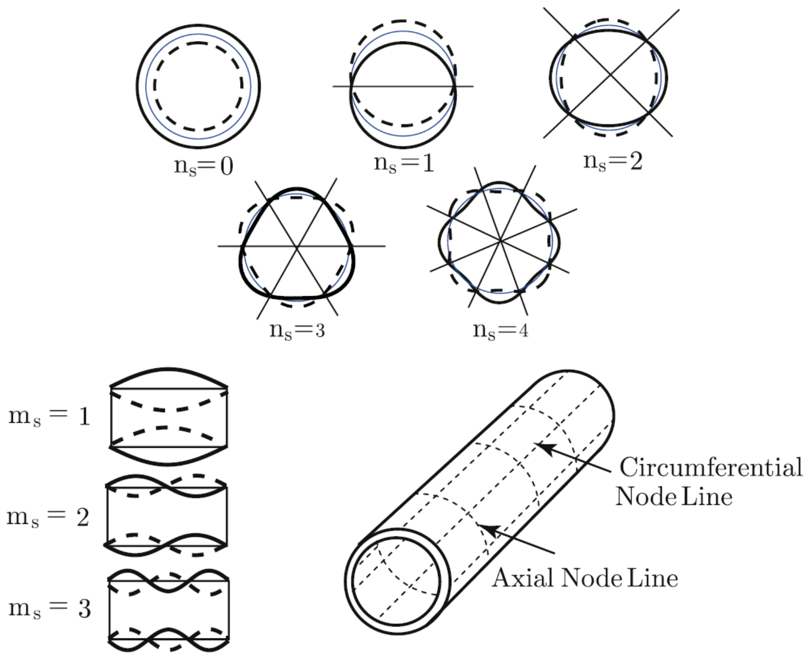


Figure 1.2: Axial and circumferential nodal patterns for simply supported cylinder without axial constraint [2]

The main sources of vibration in a pipeline system is the acoustic-induced vibration (AIV) and flow-induced vibration (FIV). Both phenomenon cause identical damage but their generation mechanism and mitigation strategies are different [3]. Flow-induced vibration is the result of turbulence in the fluid, which occurs due to flow discontinuities, that is, in changes in direction or velocity, such as bends, tees, partially closed valve and fittings that causes the pipe to displace longitudinally and transversely. Thus, this is phenomenon is more prominent for liquids that gases because the momentum flux on the pipe is higher in liquids than gases because of the higher density. FIV risk of fatigue is increased as seen before, at high velocities, high density fluid conditions and due to long spans or inadequate pipe supports or constrains.

Acoustic-induced vibration happens when vibrations are induced by compressor gas pulsation or pump pressure ripples in hydraulic systems. This is a vibration mechanism where unsteady fluid flow is more present in gases since their compressibility ratio is much higher than liquids. This pressure pulsations and velocity oscillations are generated by compressors or positive displacement pumps, which contains many harmonic components of the rotational speed. The phenomenon related to this unsteady fluid motion is known as Fluid-structure interaction (FSI), and this interaction is manifested in the vibration of the pipeline and perturbations in fluid pressure. A sudden opening or closing of valve, the change in flow direction and mechanical excitation can induce the coupling vibration of pipeline and moving fluid. The three more common coupling mechanisms include Poisson coupling, friction coupling and junction coupling [4, 5]. Poisson coupling is caused by the internal fluid pressure that is translated into axial stress by the Poisson coefficient. The stress waves generated by the transient pressure can travel faster than the fluid wave in the pipeline. The friction between fluid and pipeline is known as friction coupling and junction coupling refers to changes in momentum at some location due to increase in flow velocity, in valves and tees, or change in flow direction, as in bends and tees.

This problem is modeled by a combination of the finite element method for structural field and transfer matrix method [6] for the acoustic field considering a plane wave assumption. Both use the same mesh and can be used for complex geometries.

1.3 Objectives and Deliverables

The main objective of this thesis is to develop an efficient program in MATLAB capable of solving simple pipeline vibration analysis using finite element analysis. It needs to be capable of solving the structural free vibration analysis, where only the structure or the structure plus internal steady flow are used. This analysis calculated and displays the natural frequencies of vibration and the mode shapes for each natural frequency. An forced vibration analysis is also needed to calculate the structure response to an harmonic excitation to locate points of resonance and anti-resonance. Then, the program should be able to analyse an acoustic gas pulsation problem, enabling the module to do a time harmonic analysis of the acoustic fluid. With this analysis is now possible to couple the acoustic results with the structural harmonic analysis to determine how the structure responds to a given acoustic situation.

The first task is to import or help the program user to define the pipeline. The pipeline is displayed as a series of nodes and elements that form the network, so there is no three-dimensional display of the cross-section, just the pipeline axis. Line style network display approach was used because of the time-consuming process that is to create and display 3D solids in MATLAB and also because it is not the main objective of this work.

Both modal and harmonic structural analysis are going to be developed using two different beam theories to describe the structural part of the system: Euler-Bernoulli and Timoshenko beam theory, considering only linear behavior from the structure and material. This will be perform so that the differences between the models are visible and to provide additional information for the analysis.

The acoustic time harmonic analysis enables to determine the pressure field inside the pipeline for a given frequency and for different flow velocity sources. This pressure and velocity frequency dependency is useful to model excitation sources as compressors, pumps and different piping elements. With the pressure field solved, the forces acting on the pipe walls by the acoustic pressure waves are calculated and now a structural forced vibration analysis can be performed with these harmonic forces to predict the dynamic response of structural systems subjected to harmonic acoustic loads.

After the development of the program, this will be tested in a variety of ways to ensure that the results are in accordance with analytical results, laboratory experiments, other authors models and commercial finite element software. In this work the main comparison of results are performed with results in the literature and with an open source code *OpenPulse*. *OpenPulse* [7] is a open source software for pulsation analysis of pipeline systems written in Python for numerical modelling of low-frequency acoustically induced vibration in gas pipeline systems. The use of *OpenPulse* is due to the fact that there is no other open source software capable of performing this type of analysis.

1.4 Thesis Outline

This document is formed by five chapters. In this first introductory chapter, the theme of this work is presented as well as the objectives and motivation. Also treats some simplifying hypotheses and ends up with a brief description of the results obtained.

In chapter 2 some fundamental theory that supports the work developed is presented. Starts with the fundamental structural vibration analysis. Next, the finite element method (FEM) is introduced to compute the element matrices and to make free vibration and forced vibration analyses. Modeling of internal fluid flow forces and equations are shown next and for the last topic, the analysis of acoustic pressure waves is done with the help of the relation between pressure and velocity in a compressible fluid flow, the fluid-structure interaction formulates the forces in that interaction and the consequences of that interaction in the acoustic model that describes the planar pulsating waves.

Chapter 3 includes all the aspects related to the development of the computational code. It starts with the description and development of the algorithm to create the mesh used to define the pipeline that is used in both structural and acoustic analyses. The geometry transformations relating to the elements in that mesh, the global matrices assembly and plot of the results is also present in this chapter. Then it is described the effects of constant internal steady fluid flow in a pipeline. It concludes with the acoustic analysis, with the different possible boundary conditions in this systems and the creation of the force vector derived from the weak interaction between the fluid and the structure. The relevant computational code functions are present in appendix A.

Chapter 4 is dedicated to the presentation of results obtained with the developed MATLAB code, with, first, verifying it with simple analytical and experimental results and then with more complex examples, comparing it with commercially developed computer software. This validation is done step-by-step, that is, the structural component is analysed first and then the effects of constant fluid flow and pulsating pressure waves is analysed after. After this, the comparisons between results and models used are performed. *OpenPulse* is used to verify the developed MATLAB code in the acoustic section.

Chapter 5 presents all the major conclusions obtained from the work developed, discusses assumptions made during the process and presents some suggestions for future work based on the code developed and what was archived.

Chapter 2

Background

2.1 Structural

2.1.1 Beam Theory

In this section, the formulation of stiffness and mass matrices are described for three-dimensional frame elements. This can be archived by the principle of minimum potential energy or the principle of virtual displacements and using either Lagrange equations or Hamilton's principle to derive the dynamic equations of the structure [8]. To do this, the distribution of strains and velocities within the element is written in terms of nodal coordinates.

A component of a time-dependent three dimensional displacement $d_i(x_1, x_2, x_3, t)$, $(i = 1, ..., 3)$ in a solid continuum can be expressed in terms of the displacements of a set of nodal displacements, $u_n(t)$, $(n = 1, ..., N)$ and a corresponding set of shape functions ϕ_{in}, each relating coordinate nodal displacement $u_n(t)$ to internal displacement $d_i(\mathbf{x}, t)$.

$$d_i(x_1, x_2, x_3, t) = \sum_{n=1}^{N} \phi_{in}(x_1, x_2, x_3)\, u_n(t) \tag{2.1}$$

$$\mathbf{d}(\mathbf{x}, t) = \mathbf{\Phi}(\mathbf{x})\, \mathbf{u}(t) \tag{2.2}$$

Equation (2.2) can be rewritten to account for virtual displacements which can make use of the same set of shape functions

$$\delta\mathbf{d}(\mathbf{x}, t) = \mathbf{\Phi}(\mathbf{x})\, \delta\mathbf{u}(t) \tag{2.3}$$

Axial strain ϵ_{ii} and shear strain γ_{ij} are derived from partial derivatives of the displacement field so the strain vector can be expressed in terms of the nodal displacement vector. This is a linear relation through a matrix $\mathbf{B}$ which contains derivatives of the shape functions [8].

$$\boldsymbol{\epsilon}(\mathbf{x}, t) = \mathbf{B}(\mathbf{x})\, \mathbf{u}(t) \tag{2.4}$$

$$\delta\boldsymbol{\epsilon}(\mathbf{x}, t) = \mathbf{B}(\mathbf{x})\, \delta\mathbf{u}(t) \tag{2.5}$$

The principle of virtual displacements, or virtual work, states that the equilibrium of a body requires that for any small virtual displacement imposed on the body in its state of equilibrium, the total internal virtual work is equal to the total external virtual work. The work of a set of external forces $\mathbf{f}$ acting on the body with nodal virtual displacements $\delta\mathbf{u}$ is

$$\delta W_{ext} = \mathbf{f}^T \, \delta\mathbf{u} \tag{2.6}$$

The inertial force on an accelerating volume element with mass $\rho \, dV$ is $(\rho \, dV) \, \ddot{\mathbf{d}}(\mathbf{x}, t)$, with $\ddot{\mathbf{d}}(\mathbf{x}, t)$ as the second time derivative of **d** in expression (2.2). The internal virtual work,δW_{int} , of these forces moving through collocated virtual displacements throughout an elastic solid is

$$\delta W_{int} = \int_V \ddot{\mathbf{d}}^T \, \rho \, \delta\mathbf{d} \, dV \tag{2.7}$$

and substituting the shape function relations (2.2) and (2.3) into equation (2.7) gives

$$\delta W_{int} = \int_V \ddot{\mathbf{u}}^T \, \mathbf{\Phi}^T \, \rho \, \mathbf{\Phi} \, \delta\mathbf{u} \, dV \tag{2.8}$$

The distribution of internal elastic stresses $\boldsymbol{\sigma}(\mathbf{x}, t)$ collocated with distributed internal virtual strains $\delta\boldsymbol{\epsilon}(\mathbf{x}, t)$ is

$$\delta W_{ela} = \int_V \boldsymbol{\sigma}^T \, \delta\boldsymbol{\epsilon} \, dV \tag{2.9}$$

where substituting the stress-strain relation from Hooke's law, $\boldsymbol{\sigma} = \mathbf{D}\boldsymbol{\epsilon}$, in equation (2.9), and substituting the strain-displacement relations (2.4) and (2.5) gives

$$\delta W_{ela} = \int_V \mathbf{u}^T \, \mathbf{B}^T \, \mathbf{D} \, \mathbf{B} \, \delta\mathbf{u} \, dV \tag{2.10}$$

where $\mathbf{D}$ is the material stiffness matrix that is equal to the inverse of the compliance matrix.

Using equations (2.6), (2.8), (2.10) and using the to principal of virtual work to equate internal virtual word to external virtual work

$$\delta W_{int} + \delta W_{ela} = \delta W_{ext} \tag{2.11}$$

$$\ddot{\mathbf{u}}^T \int_V \mathbf{\Phi}^T \, \rho \, \mathbf{\Phi} \, dV \, \delta\mathbf{u} + \mathbf{u}^T \int_V \mathbf{B}^T \, \mathbf{D} \, \mathbf{B} \, dV \, \delta\mathbf{u} = \mathbf{f}^T \, \delta\mathbf{u} \tag{2.12}$$

and eliminating the $\delta\mathbf{u}$ term from each term and transposing both sides, equation (2.12) becomes

$$\left[\int_V \mathbf{\Phi}^T \, \rho \, \mathbf{\Phi} \, dV \right] \ddot{\mathbf{u}} + \left[\int_V \mathbf{B}^T \, \mathbf{D} \, \mathbf{B} \, dV \right] \mathbf{u} = \mathbf{f} \tag{2.13}$$

in which the first term matrix is the element mass matrix and the second term matrix is the element stiffness matrix given respectively by equations (2.14) and (2.15).

$$\mathbf{K}^e = \int_V \mathbf{B}^T \, \mathbf{D} \, \mathbf{B} \, dV \tag{2.14}$$

$$\mathbf{M}^e = \int_V \mathbf{\Phi}^T \rho \, \mathbf{\Phi} \, dV \tag{2.15}$$

Given equation (2.13), and using the global mass and stiffness matrices, the undamped dynamic equation of motion of the structure is given by

$$\mathbf{M}\ddot{\mathbf{u}}(t) + \mathbf{K}\mathbf{u}(t) = \mathbf{f}(t) \tag{2.16}$$

Damping in vibrating structures can arise from multiple phenomena. For example, within structural systems from internal friction forces present in the material. The damping matrix $\mathbf{C}$ can be derived with the internal virtual work of real viscous stresses moving through virtual strains. Given a material viscous damping matrix, a structural element damping matrix can be determined through an equation similar to equation (2.14) but substituting $\mathbf{D}$ with the material viscous damping matrix. A simpler approach is to use the Rayleigh damping matrix [8] that is proportional to system's mass and stiffness matrices, and it is given by

$$\mathbf{C} = \alpha\mathbf{M} + \beta\mathbf{K} \tag{2.17}$$

where α is the mass-proportional coefficient and β is the stiffness-proportional coefficient. Now, the damped dynamic equation of motion can be rewritten was

$$\mathbf{M}\ddot{\mathbf{u}}(t) + \mathbf{C}\dot{\mathbf{u}}(t) + \mathbf{K}\mathbf{u}(t) = \mathbf{f}(t) \tag{2.18}$$

In this work, to model a pipeline, a three-dimensional beam element is used, and each element contain two nodes and every node has six degrees of freedom (Figure 2.1): three translational displacements and three rotational degrees of freedom.

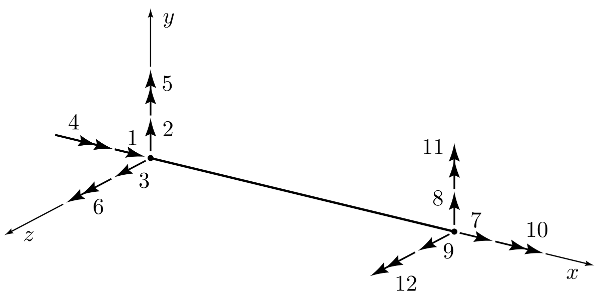


Figure 2.1: Three-dimensional element with two nodes and six degrees-of-freedom at each node

The displacements of a point in a cross section are described by the translation components, w_x, w_y, w_z, of the neutral line and the rotations, θ_x, θ_y, θ_z of the cross section. This nodal displacements and rotations are given in figure 2.1 by the numbers 1 through 6 for the first node and 7 to 12 for the second node. It is considered that w_i are small compared by beam length and the rotation components are all small so that $\sin\theta \approx \tan\theta \approx \theta$. The displacement of any point, d_i is determined by [9]

$$d_x(x,y,z,t) = w_x(x,t) + z\theta_y(x,t) + y\theta_z(x,t) \tag{2.19}$$

$$d_y(x,y,z,t) = w_y(x,t) - z\theta_x(x,t) \tag{2.20}$$

$$d_z(x,y,z,t) = w_z(x,t) + y\theta_x(x,t) \tag{2.21}$$

2.1.2 Euler-Bernoulli Beam

Euler-Bernoulli beam kinematics [9, 10] assumes that a cross-section remains orthogonal to the deformed beam axis, as in figure (2.2) If the rotation is equal to the slope of the beam, then

$$\theta_y = -\frac{dw_z}{dx}, \quad \theta_z = \frac{dw_y}{dx} \tag{2.22}$$

where θ_y is the cross-section rotation angle around the y axis and θ_z is the angle of rotation around the z axis.

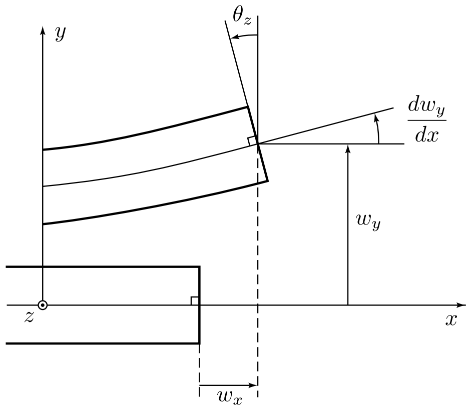


Figure 2.2: Deformation in Euler-Bernoulli beam theory

The internal displacements $d_x(\mathbf{x},t)$, $d_y(\mathbf{x},t)$ and $d_z(\mathbf{x},t)$ are expressed in terms of a set of shape functions, ψ_{xn}, ψ_{yn}, ψ_{zn}, the end displacements ($u_1(t)$, $u_2(t)$, $u_3(t)$, $u_7(t)$, $u_8(t)$, $u_9(t)$) and the end rotations ($u_6(t)$, $u_7(t)$, $u_6(t)$, $u_{10}(t)$, $u_{11}(t)$, $u_{12}(t)$). The shape functions satisfy the differential equation describing static bending of a Euler-Bernoulli beam with a specified unit displacement at each nodal co-ordinate, produced by forces and moments applied at each degree-of-freedom. With this kind of loading, internal bending moments $M(x)$ vary linearly along the neutral axis and the transverse displacements of the neutral axis are cubic polynomials. For bending degrees of freedom, this element introduces C^1 continuous Hermite shape functions [11].

$$\phi_{x1} = 1 - \frac{x}{L} \tag{2.23a}$$

$$\phi_{x7} = \frac{x}{L} \tag{2.23b}$$

$$\phi_{y2} = 1 - 3\left(\frac{x}{L}\right)^2 + 2\left(\frac{x}{L}\right)^3 \tag{2.23c}$$

$$\phi_{y6} = \left[\left(\frac{x}{L}\right) - 2\left(\frac{x}{L}\right)^2 + \left(\frac{x}{L}\right)^3\right] L \tag{2.23d}$$

$$\phi_{y8} = 3\left(\frac{x}{L}\right)^2 - 2\left(\frac{x}{L}\right)^3 \tag{2.23e}$$

$$\phi_{y12} = \left[-\left(\frac{x}{L}\right)^2 + \left(\frac{x}{L}\right)^3\right] L \tag{2.23f}$$

$$\phi_{z3} = 1 - 3\left(\frac{x}{L}\right)^2 + 2\left(\frac{x}{L}\right)^3 \tag{2.23g}$$

$$\phi_{z5} = -\left[\left(\frac{x}{L}\right) - 2\left(\frac{x}{L}\right)^2 + \left(\frac{x}{L}\right)^3\right] L \tag{2.23h}$$

$$\phi_{z9} = 3\left(\frac{x}{L}\right)^2 - 2\left(\frac{x}{L}\right)^3 \tag{2.23i}$$

$$\phi_{z11} = \left[\left(\frac{x}{L}\right)^2 - \left(\frac{x}{L}\right)^3\right] L \tag{2.23j}$$

$$\phi_{y4} = -\left(1 - \frac{x}{L}\right) z \tag{2.23k}$$

$$\phi_{z4} = \left(1 - \frac{x}{L}\right) y \tag{2.23l}$$

$$\phi_{y10} = -\left(\frac{x}{L}\right) z \tag{2.23m}$$

$$\phi_{z10} = \left(\frac{x}{L}\right) y \tag{2.23n}$$

The shape functions (2.23a) to (2.23n) can be separated into four groups, each of which can be considered to be independent from the others. Axial displacements, bending in the xy plane, bending in the xz plane and torsional displacements. Torsional displacements can also be treated considering an rotation angle about the x axis with the same displacement model as axial displacements (2.23a) and (2.23b).

The shape function matrix, $\mathbf{\Phi}$ is

$$\mathbf{\Phi}(x,y,z) = \begin{bmatrix} \phi_{x1} & 0 & 0 & 0 & 0 & 0 & \phi_{x7} & 0 & 0 & 0 & 0 & 0 \\ 0 & \phi_{y2} & 0 & \phi_{y4} & 0 & \phi_{y6} & 0 & \phi_{y8} & 0 & \phi_{y10} & 0 & \phi_{y12} \\ 0 & 0 & \phi_{z3} & \phi_{z4} & \phi_{z5} & 0 & 0 & 0 & \phi_{z9} & \phi_{z10} & \phi_{z11} & 0 \end{bmatrix} \tag{2.24}$$

Now the stiffness matrix and mass matrix can be computed using equations (2.14) and (2.15).

In expression (2.25) the sub-matrices $\mathbf{K}_{22}^e$ and $\mathbf{M}_{22}^e$ are equal to $\mathbf{K}_{11}^e$ and $\mathbf{M}_{11}^e$, respectively, except for the sign of the off-diagonal entries.

$$\mathbf{K}^e = \begin{bmatrix} \mathbf{K}_{11}^e & \mathbf{K}_{12}^e \\ \mathbf{K}_{21}^e & \mathbf{K}_{22}^e \end{bmatrix}, \quad \mathbf{M}^e = \begin{bmatrix} \mathbf{M}_{11}^e & \mathbf{M}_{12}^e \\ \mathbf{M}_{21}^e & \mathbf{M}_{22}^e \end{bmatrix} \tag{2.25}$$

$$\mathbf{K}_{11}^e = \begin{bmatrix} \frac{EA}{L} & 0 & 0 & 0 & 0 & 0 \\ & \frac{12EI_z}{L^3} & 0 & 0 & 0 & \frac{6EI_z}{L^2} \\ & & \frac{12EI_y}{L^3} & 0 & -\frac{6EI_y}{L^2} & 0 \\ & & & \frac{GI_x}{L} & 0 & 0 \\ & Sym & & & \frac{4EI_y}{L} & 0 \\ & & & & & \frac{4EI_z}{L} \end{bmatrix} \tag{2.26}$$

$$\mathbf{K}_{12}^e = {\mathbf{K}_{21}^e}^T = \begin{bmatrix} -\frac{EA}{L} & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{12EI_z}{L^3} & 0 & 0 & 0 & \frac{6EI_z}{L^2} \\ 0 & 0 & -\frac{12EI_y}{L^3} & 0 & -\frac{6EI_y}{L^2} & 0 \\ 0 & 0 & 0 & -\frac{GI_x}{L} & 0 & 0 \\ 0 & 0 & \frac{6EI_y}{L^2} & 0 & \frac{2EI_y}{L} & 0 \\ 0 & -\frac{6EI_z}{L^2} & 0 & 0 & 0 & \frac{2EI_z}{L} \end{bmatrix} \tag{2.27}$$

$$\mathbf{M}_{11}^e = \frac{\rho AL}{420} \begin{bmatrix} 140 & 0 & 0 & 0 & 0 & 0 \\ & 156 & 0 & 0 & 0 & 22L \\ & & 156 & 0 & -22L & 0 \\ & & & 140r_x^2 & 0 & 0 \\ & Sym & & & 4L^2 & 0 \\ & & & & & 4L^2 \end{bmatrix} \tag{2.28}$$

$$\mathbf{M}_{12}^e = {\mathbf{M}_{21}^e}^T = \frac{\rho AL}{420} \begin{bmatrix} 70 & 0 & 0 & 0 & 0 & 0 \\ 0 & 54 & 0 & 0 & 0 & -13L \\ 0 & 0 & 54 & 0 & 13L & 0 \\ 0 & 0 & 0 & 70r_x & 0 & 0 \\ 0 & 0 & -13L & 0 & -3L^2 & 0 \\ 0 & 13L & 0 & 0 & 0 & -3L^2 \end{bmatrix} \tag{2.29}$$

where r_x is the radius of gyration of the cross section around the x axis, $r_x = (I_x/A)^{1/2}$. A is the area of the cross section and I_x is the torsional constant.

2.1.3 Timoshenko Beam

Timoshenko beam theory differs from Euler-Bernoulli theory because it accounts for shear deformation [10]. Timoshenko beam elements are more adequate for beams where $L/h < 10$, since shear deformation has an influence in the solution. Timoshenko beam elements require C^0 continuity for the deflection and rotation fields in contrast with C^1 continuity of the Euler-Bernoulli beam elements which means that shape functions are simpler. However, they suffer from a defect called shear locking [12].

Shear locking is a phenomenon in finite element analysis of beams when elements used to analyse thick beams are used to analyse slender beams. This is characterized as the inability of the element to yield zero shear strains as it becomes progressively slender or thinner, thus the solution obtained was an error associated with that term. Two possible solutions to avoid shear locking are: 1) using approximation functions for w_i with a higher degree than the approximation functions for θ_i; 2) using function with the same degree for w_i and θ_i but adopting reduced integration rule.

The rotation of plane section orthogonal to beam axis is presented in figure (2.3) and it is given by [9]

$$\theta_y = -\frac{dw_z}{dx} + \gamma_{xz}, \quad \theta_z = \frac{dw_y}{dx} - \gamma_{xy} \tag{2.30}$$

where dw/dx is the slope of the beam axis and γ is the rotation due to the distortion of the cross-section.

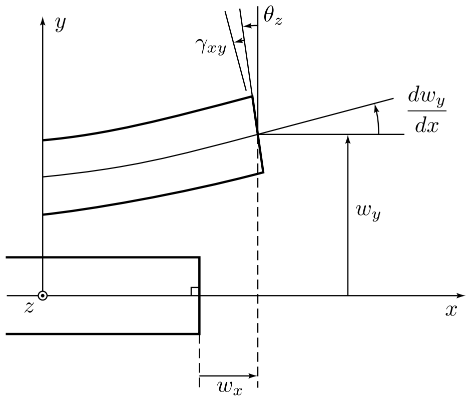


Figure 2.3: Deformation in Timoshenko beam theory

Two methods to determine both stiffness and mass matrices are presented. The first method is similar to the method presented in section 2.1.2 with some modifications on the shape functions to account for shear deformation. The second method uses Gauss integration to evaluate the integrals in the problems weak formulation.

In the first method the transverse deformation of a beam with shear and bending strains may be separated into a portion related to bending deformation and a portion related to shear deformation. Since shear deformation only has an effect on bending degrees of freedom, the shape functions for axial and torsional deformations remain the same as (2.23a), (2.23b) and (2.23k) to (2.23n). The new shape functions are given in [13, 14] and the stiffness matrix is given by

$$\mathbf{K}^e_{11} = \begin{bmatrix} \frac{EA}{L} & 0 & 0 & 0 & 0 & 0 \\ & \frac{12EI_z}{(1+\alpha_y)L^3} & 0 & 0 & 0 & \frac{6EI_z}{(1+\alpha_y)L^2} \\ & & \frac{12EI_y}{(1+\alpha_z)L^3} & 0 & -\frac{6EI_y}{(1+\alpha_z)L^2} & 0 \\ & & & \frac{GI_x}{L} & 0 & 0 \\ & Sym & & & \frac{EI_y(4+\alpha_z)}{(1+\alpha_z)L} & 0 \\ & & & & & \frac{EI_z(4+\alpha_y)}{(1+\alpha_y)L} \end{bmatrix} \tag{2.31}$$

$$\mathbf{K}_{12}^e = \mathbf{K}_{21}^{e\,T} = \begin{bmatrix} -\frac{EA}{L} & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{12EI_z}{(1+\alpha_y)L^3} & 0 & 0 & 0 & \frac{6EI_z}{(1+\alpha_y)L^2} \\ 0 & 0 & -\frac{12EI_y}{(1+\alpha_z)L^3} & 0 & -\frac{6EI_y}{(1+\alpha_z)L^2} & 0 \\ 0 & 0 & 0 & -\frac{GI_x}{L} & 0 & 0 \\ 0 & 0 & \frac{6EI_y}{(1+\alpha_z)L^2} & 0 & \frac{EI_y(2-\alpha_z)}{(1+\alpha_z)L} & 0 \\ 0 & -\frac{6EI_z}{(1+\alpha_y)L^2} & 0 & 0 & 0 & \frac{EI_z(2-\alpha_y)}{(1+\alpha_y)L} \end{bmatrix} \tag{2.32}$$

where

$$\alpha_y = \frac{12EI_z}{k_y GAL^2}, \quad \alpha_z = \frac{12EI_y}{k_z GAL^2} \tag{2.33}$$

is the ratio of bending stiffness to shear stiffness. If shear stiffness is very large, shear deformations are negligible. Note that setting $\alpha_y = \alpha_z = 0$ leads to an Euler-Bernoulli model, (2.26) and (2.27), as the influence of shear stiffness is neglected.

The shear coefficient k is a dimensionless quantity, that is introduced to account for the fact that the shear stress and strain are not uniformly over the cross section (Figure 2.4). Note that the normal stresses generated by an applied moment are linear across the section (Figure 2.5). For symmetric cross-sections, $k_y = k_z = k$, and for circular tube cross-sections the shear coefficient is given by Cowper [15] and it is given by

$$k = \frac{6(1+\nu)(1+(D_i/D_e)^2)^2}{(7+6\nu)(1+(D_i/D_e)^2)^2 + (20+12\nu)(D_i/D_e)^2} \tag{2.34}$$

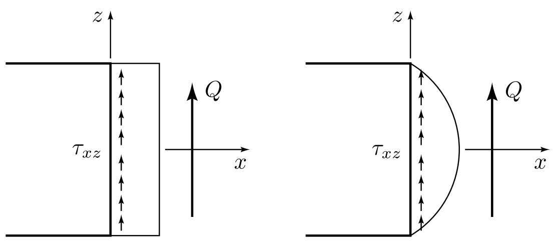


Figure 2.4: Assumed and exact distribution of tangential stresses

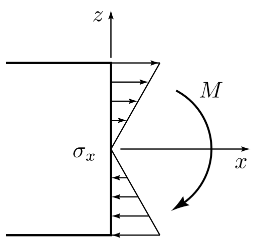


Figure 2.5: Distribution of normal stresses

For the derivation of the mass matrix there is three possibilities to consider: 1) using the same shape functions as above to account for shear deformation effects; 2) including rotatory inertia but not shear deformation effects; 3) combining both effects for a more complicated derivation, presented in Przemieniecki [12].

When including rotatory inertia to account for axial displacements outside the neutral axis, new shape functions in the x direction are derived based on a linearly variable axial beam displacements. New eight shape functions are introduced in the shape function matrix (2.24), ϕ_{x2}, ϕ_{x3}, ϕ_{x5}, ϕ_{x6}, ϕ_{x8}, ϕ_{x9}, ϕ_{x11}, ϕ_{x12}. This shape functions are given in Gavin [13] and the final mass matrix is

$$\mathbf{M}_{11}^{e} = \frac{\rho A L}{420} \begin{bmatrix} 140 & 0 & 0 & 0 & 0 & 0 \\ & 156 + \dfrac{504 I_z}{A L^2} & 0 & 0 & 0 & 22L + \dfrac{42 I_z}{AL} \\ & & 156 + \dfrac{504 I_y}{A L^2} & 0 & -22L - \dfrac{42 I_y}{AL} & 0 \\ & & & 140 r_x^2 & 0 & 0 \\ & Sym & & & 4L^2 + \dfrac{56 I_y}{A} & 0 \\ & & & & & 4L^2 + \dfrac{56 I_z}{A} \end{bmatrix} \tag{2.35}$$

$$\mathbf{M}_{12}^{e} = {\mathbf{M}_{21}^{e}}^{T} = \frac{\rho A L}{420} \begin{bmatrix} 70 & 0 & 0 & 0 & 0 & 0 \\ 0 & 54 - \dfrac{42 I_z}{AL} & 0 & 0 & 0 & -13L - \dfrac{42 I_z}{AL} \\ 0 & 0 & 54 - \dfrac{42 I_y}{AL} & 0 & 13L + \dfrac{42 I_y}{AL} & 0 \\ 0 & 0 & 0 & 70 r_x & 0 & 0 \\ 0 & 0 & -13L - \dfrac{42 I_y}{AL} & 0 & -3L^2 - \dfrac{28 I_y}{A} & 0 \\ 0 & 13L + \dfrac{42 I_z}{AL} & 0 & 0 & 0 & -3L^2 - \dfrac{28 I_y}{A} \end{bmatrix} \tag{2.36}$$

The second method to determine stiffness and mass matrices using the Timoshenko beam theory is presented in Hughes [16]. In opposition to the other methods, here the interpolation of displacements is independent for both displacements, $\mathbf{w}$, and rotations, $\boldsymbol{\theta}$,

$$\mathbf{w} = \mathbf{N}\,\mathbf{w}^{e}, \quad \boldsymbol{\theta} = \mathbf{N}\,\boldsymbol{\theta}^{e} \tag{2.37}$$

where the shape function are defined, for both types of displacements, as

$$\mathbf{N} = \left[\tfrac{1}{2}(1-\xi) \quad \tfrac{1}{2}(1+\xi) \right] \tag{2.38}$$

in natural coordinates $\xi \in [-1, 1]$. Now, the integrals presented in the stiffness and mass matrices in Hughes [16] can be computed by Gauss quadrature.

2.1.4 Free Vibration Analysis

The oscillatory motion when there are no external forces applied is important to find the dynamic response of the elastic structure. This motion is a characteristic property of the structure, and it depends on how mass and stiffness is distributed in the structure. Assuming that there is no damping in the structure, the oscillatory motion will continue indefinitely with the same amplitudes.

By assuming the external force vector $\mathbf{f}$ to be zero and the displacements to be harmonic, and with $\mathbf{U}$ being the amplitude of displacements and $\mathbf{F}$ the amplitude of forces,

$$\mathbf{u}(t) = \mathbf{U}e^{i\omega t} \tag{2.39}$$

$$\mathbf{f}(t) = \mathbf{F}e^{i\omega t} \tag{2.40}$$

the equation (2.18) gives

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{K}\mathbf{u} = 0 \tag{2.41}$$

$$[\mathbf{K} - \omega^2\mathbf{M}]\mathbf{U} = \mathbf{0} \tag{2.42}$$

since $\ddot{\mathbf{u}} = -\omega^2\mathbf{U}e^{i\omega t}$. ω denotes the natural frequency of vibration and $\mathbf{U}$ the eigenmodes of displacements. For this equation to have a non-zero solution, the matrix $[\mathbf{K} - \omega^2\mathbf{M}]$ has to be singular so that

$$|\mathbf{K} - \omega^2\mathbf{M}| = 0 \tag{2.43}$$

The solutions of this equation are its natural frequencies. In equation (2.42), ω^2 is the eigenvalue and $\mathbf{U}$ the eigenvector for this eigenvalue problem. By solving this eigenvalue problem, we find the natural frequencies and the correspondent mode shapes.

One method of calculating natural frequencies with the damping term is presented and derived by Meirovitch [17]. It states that the eigenvalue solution shall be executed using the characteristics matrix, equal to,

$$\mathbf{\Omega} = \begin{bmatrix} -\mathbf{M}^{-1}\mathbf{K} & -\mathbf{M}^{-1}\mathbf{C} \\ \mathbf{I} & 0 \end{bmatrix} \tag{2.44}$$

where the imaginary part of the solution represents natural frequencies of vibration and the real part represents the rate of decay of the free vibration.

2.1.5 Forced Vibration Analysis

Harmonic analysis is used to predict the steady state dynamic response of a structure subjected to an harmonic loading. Given an damped system equation (2.18) and using (2.39) and (2.40), the equation is presented as

$$[-\omega^2\mathbf{M} + i\omega\mathbf{C} + \mathbf{K}]\mathbf{U}e^{i\omega t} = \mathbf{F}e^{i\omega t} \tag{2.45}$$

$$[-\omega^2\mathbf{M} + i\omega\mathbf{C} + \mathbf{K}]\mathbf{U} = \mathbf{F} \tag{2.46}$$

Since the amplitude force vector $\mathbf{F}$ is given, the displacements amplitude $\mathbf{U}$ can be calculated for each given known excitation frequency ω by

$$\mathbf{U} = [-\omega^2\mathbf{M} + i\omega\mathbf{C} + \mathbf{K}]^{-1}\mathbf{F} \tag{2.47}$$

If the calculation is done for multiple frequencies a frequency response graph can be created with the magnitude of the displacements for each frequency.

2.2 Internal Fluid Flow

Considering a pipe with fluid flowing through it at pressure p and at a constant velocity v through the internal cross-section of area A_i. Due to the lateral vibration of the pipeline, the deflected pipe, the fluid is accelerated because of the changing curvature. Forces and moments acting on the fluid and pipe element are shown in figure 2.6.

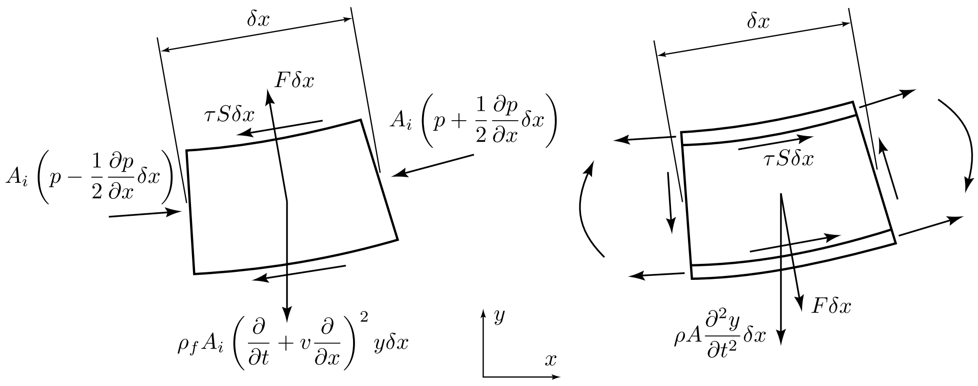

δx
$F\delta x$
$\tau S\delta x$
$A_i\left(p+\frac{1}{2}\frac{\partial p}{\partial x}\delta x\right)$
$A_i\left(p-\frac{1}{2}\frac{\partial p}{\partial x}\delta x\right)$
$\rho_f A_i\left(\frac{\partial}{\partial t}+v\frac{\partial}{\partial x}\right)^2 y\delta x$
y
x
δx
$\tau S\delta x$
$F\delta x$
$\rho A\frac{\partial^2 y}{\partial t^2}\delta x$

Figure 2.6: Fluid and pipe forces

Given the forces and moments the derivation of the equation of motion for a free vibration of a fluid convening pipe is given by equation (2.48). Derivation of this equation is presented in Blevins [18] and it is done by a balance of forces in both fluid and pipe element considering small deflections and if gravity, internal damping, externally imposed tension and pressurization effects are either absent or neglected.

$$EI\frac{\partial^4 y}{\partial x^4}+2\rho_f A_i v\frac{\partial^2 y}{\partial x\partial t}+\rho_f A_i v^2\frac{\partial^2 y}{\partial x^2}+(\rho A+\rho_f A_i)\frac{\partial^2 y}{\partial t^2}=0 \tag{2.48}$$

where v is the flow velocity of the fluid and $(\rho A + \rho_f A_i)$ is the total mass per unit length of the pipe. In equation (2.48) the first term is the force component acting on the pipe as a result of bending of the pipe. The second term is the force component acting on the pipe as a result of flow around deflected pipe curvature. The third therm is the force required to rotate the fluid element, also known as Coriolis force. The last term represents the force component acting on the pipe as a result of the inertia of the pipe and the fluid flowing through it.

With this equation four different matrices can be derived. This derivation is done in [19–21] in different method but only considering two degrees of freedom per node or in two-dimensional beam element. The two degrees of freedom are deflection in the y-direction and the rotation in x-y plane. The first term of equation (2.48) produces the element stiffness matrix for pipe as beam element. The second is the stiffness matrix for the force that conforms fluid to the pipe, the third is the damping matrix for the Coriolis force and the last is the mass matrix for the pipe with fluid. This planar matrices can be converted into a three-dimensional system by rewriting them in the form of the x-z plane. The difference between planes is that a positive rotation in the x-y plane makes the pipe deform in the positive y-direction and a positive rotation in the x-z plane makes the pipe deform in the negative z-direction.

Given the transformation between planar and spacial dimensions the stiffness matrix for the fluid that conforms fluid to the pipe is

$$\mathbf{K}^e_{f11} = \frac{\rho_f A_i v^2}{30L} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 36 & 0 & 0 & 0 & 3L \\ 0 & 0 & 36 & 0 & -3L & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -3L & 0 & 4L^2 & 0 \\ 0 & 3L & 0 & 0 & 0 & 4L^2 \end{bmatrix} \tag{2.49}$$

$$\mathbf{K}^e_{f12} = {\mathbf{K}^e_{f21}}^T = \frac{\rho_f A_i v^2}{30L} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -36 & 0 & 0 & 0 & 3L \\ 0 & 0 & -36 & 0 & -3L & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3L & 0 & -L^2 & 0 \\ 0 & -3L & 0 & 0 & 0 & -L^2 \end{bmatrix} \tag{2.50}$$

The stiffness matrix $\mathbf{K}_f$ tends to weaken the overall stiffness of the pipe system so the global stiffness matrix used is calculated by

$$\mathbf{K}^e = \mathbf{K}^e_p - \mathbf{K}^e_f \tag{2.51}$$

where $\mathbf{K}_p$ is the pipe structural stiffness matrix presented in section 2.1.

Mass and damping matrices matrix given by the following matrices. Note that both matrices need to be summed to the structural mass matrix and to the structural damping if material damping is considered.

This analysis was modeled with Euler-Bernoulli beam theory. Chu and Lin [22] developed the stiffness, mass and damping matrices using the Timoshenko beam model with both effects of shearing deformation and rotary inertia considered with the effect of moving fluid treated as external distributed forces on the pipe.

$$\mathbf{M}^e_{f11} = \frac{\rho_f A_i L}{420} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 156 & 0 & 0 & 0 & 22L \\ 0 & 0 & 156 & 0 & -22L & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -22L & 0 & 4L^2 & 0 \\ 0 & 22L & 0 & 0 & 0 & 4L^2 \end{bmatrix} \tag{2.52}$$

$$\mathbf{M}^e_{f12} = {\mathbf{M}^e_{f21}}^T = \frac{\rho_f A_i L}{420} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 54 & 0 & 0 & 0 & -13L \\ 0 & 0 & 54 & 0 & 13L & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -13L & 0 & -3L^2 & 0 \\ 0 & 13L & 0 & 0 & 0 & -3L^2 \end{bmatrix} \tag{2.53}$$

$$\mathbf{C}^e_{f11} = \frac{\rho_f A_i v}{30} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -30 & 0 & 0 & 0 & -6L \\ 0 & 0 & -30 & 0 & 6L & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -6L & 0 & 0 & 0 \\ 0 & 6L & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.54}$$

$$\mathbf{C}^e_{f12} = {\mathbf{C}^e_{f21}}^T = \frac{\rho_f A_i v}{30} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -30 & 0 & 0 & 0 & 6L \\ 0 & 0 & -30 & 0 & -6L & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 6L & 0 & L^2 & 0 \\ 0 & -6L & 0 & 0 & 0 & L^2 \end{bmatrix} \tag{2.55}$$

2.3 Acoustics

2.3.1 Introduction

Acoustics is a branch of physics that study the propagation of waves in fluids (gases or liquids). In fluids, sound propagates primarily as a pressure wave and as part of fluid dynamics, the fluid motion can be described by using the laws of mass conservation, momentum and energy conservation applied a fluid element. These equations can be written in integral or differential form. Both forms are derived e.g. in White [23].

Mass conservation equation is differential form is given by

$$\frac{\partial \rho_f}{\partial t} + \boldsymbol{\nabla} \cdot (\rho_f \mathbf{v}) = 0 \tag{2.56}$$

where ρ_f is the fluid density and $\mathbf{v}$ is the flow velocity in a given position. The momentum conservation law is:

$$\rho_f \left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \boldsymbol{\nabla} \mathbf{v} \right) = -\boldsymbol{\nabla} p + \mathbf{f} + \boldsymbol{\nabla} \cdot \boldsymbol{\tau} \tag{2.57}$$

where $\boldsymbol{\tau}$ is the viscous stress tensor, p is the pressure and $\mathbf{f}$ is an external force density, like the gravitational force.

Assuming that the flow is frictionless ($\boldsymbol{\nabla} \cdot \boldsymbol{\tau} = 0$), no external forces are applied and limit the analysis to acoustic perturbations (ρ'_f, p', $\mathbf{v}'$) at a stagnant uniform fluid (ρ_{f0}, p_0) as

$$p(\mathbf{x}, t) = p_0 + p'(\mathbf{x}, t) \tag{2.58}$$

$$\rho_f(\mathbf{x}, t) = \rho_{f0} + \rho'_f(\mathbf{x}, t) \tag{2.59}$$

$$\mathbf{v}(\mathbf{x}, t) = \mathbf{v}'(\mathbf{x}, t) \tag{2.60}$$

equations (2.56) and (2.57) can be written as

$$\frac{\partial \rho'}{\partial t} + \rho_0 \boldsymbol{\nabla} \cdot \mathbf{v}' = 0 \tag{2.61}$$

$$\rho_0 \frac{\partial \mathbf{v}'}{\partial t} + \boldsymbol{\nabla} p' = 0 \tag{2.62}$$

By subtracting the time derivative of the mass conservation law (2.61) from the divergence of the momentum conservation law (2.62) it gives

$$\frac{\partial^2 \rho'}{\partial t^2} - \boldsymbol{\nabla}^2 p' = 0 \tag{2.63}$$

and using the relation of the fluid speed of sound, $p' = c_f^2 \rho'$, to eliminate ρ', and obtain the non-dissipative wave equation [24, 25],

$$\frac{\partial^2 p'}{\partial t^2} - c_f^2 \boldsymbol{\nabla}^2 p' = 0 \tag{2.64}$$

With this relation between pressure and velocity, the main goal of this acoustic analysis is to obtain the acoustic pressure, at all the points in the pipe system, for a certain frequency range.

2.3.2 Transfer Matrix Method

When modeling acoustics of pipe systems, two methods can be applied: the transfer matrix method and the stiffness matrix method [26]. The transfer matrix is a adequate technique for dealing with long sections of pipes with changes in cross-sectional area, as the system matrix is found simply by multiplying individual transfer matrices of the components. The stiffness matrix method is applied most of the time to structural analysis because it can easily deal with branched and pipeline networks. However, the larger the number of components, the larger the size of the stiffness matrix.

Considering a straight uniform pipe element shown in figure 2.7. The fluid properties remain uniform inside the tube. Pressure and volume velocity or flow rate in the inlet and outlet are denoted as p_1, q_1 and p_2, q_2. The acoustic impedance if the element is represented by Z_f. The first objective is to find a matrix equation that expresses the volume velocity and pressure at any point inside the tube element at the wave number, $k = \omega/c_f$, in terms of their values at the inlet. The tube element can be represented as a linear system with to inputs and two outputs as

$$\begin{bmatrix} p_2 \\ q_2 \end{bmatrix} = \mathbf{T} \begin{bmatrix} p_1 \\ q_1 \end{bmatrix} \tag{2.65}$$

where $\mathbf{T}$ is the transfer matrix for the uniform tube element.

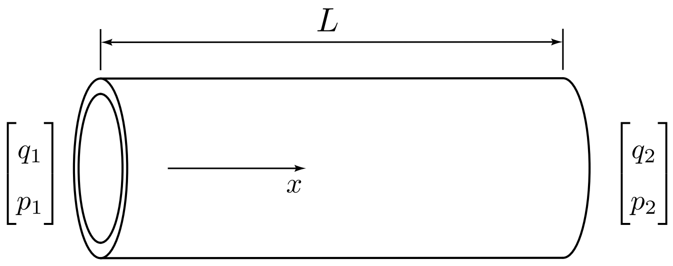

L
$\begin{bmatrix} q_1 \\ p_1 \end{bmatrix}$
x
$\begin{bmatrix} q_2 \\ p_2 \end{bmatrix}$

Figure 2.7: Uniform tube element

Considering the assumption of planar waves which means that the fluid properties are only defined by the x coordinate, which needs to be much larger than the radial dimension for this assumption to be valid. The one-dimensional form of equation (2.64) is

$$\frac{\partial^2 p'}{\partial t^2} - c_f^2 \frac{\partial^2 p'}{\partial x^2} = 0 \tag{2.66}$$

and the one-dimensional solution can be written is terms of travelling waves for the pressure as

$$p(x,t) = (Ae^{-ikx} + Be^{ikx})e^{i\omega t} \tag{2.67}$$

With the aid of the one-dimensional momentum conservation equation from equation (2.62),

$$\frac{\partial u}{dt} = -\frac{1}{\rho}\frac{\partial p}{\partial x} \tag{2.68}$$

we can write the volume velocity, q, with $q = u\,A_i$

$$q(x,t) = \frac{A_i}{\rho c_f}(Ae^{-ikx} - Be^{ikx})e^{i\omega t} \tag{2.69}$$

In equations (2.67) and (2.69) the constants A and B that appear in the general solutions are determined from the boundary conditions of the problem. But evaluating those equation at both ends of the tube element the constants are eliminated and the transfer matrix is obtained .

$$\begin{bmatrix} p_2 \\ q_2 \end{bmatrix} = \begin{bmatrix} \cos(kx) & -i\,Z_f\sin(kx) \\ -i/Z_f\sin(kx) & \cos(kx) \end{bmatrix} \begin{bmatrix} p_1 \\ q_1 \end{bmatrix} \tag{2.70}$$

where $Z_f = \rho_f\,c_f/A_i$ is the acoustic impedance. Fluid speed of sound should be replaced with c_0 that denotes the corrected fluid speed of sound because c_f is affected by the mechanical compliance of the pipe wall. The equation for c_0 is given by [27],

$$c_0 = c_f\left(1 + \left(\frac{D_i K_f}{Et}\right)\right)^{-1/2} \tag{2.71}$$

where D_i is the internal diameter, t is the pipe wall thickness and K_f the fluid bulk modulus calculated as $K_f = c_f^2\rho_f$.

2.3.3 Stiffness Matrix Method

The stiffness matrix relates, in contrast with the transfer matrix, the same type of variables in each nodal vector which is more suitable with the finite element method. This stiffness matrix, $\mathbf{S}$ relates the nodal volume velocities and the nodal pressures as

$$\begin{bmatrix} q_1 \\ q_2 \end{bmatrix} = \mathbf{S}\begin{bmatrix} p_1 \\ p_2 \end{bmatrix} \tag{2.72}$$

A simple algebraic manipulation will reveal that the stiffness matrix can be expressed in terms of the transfer matrix elements. That manipulation is given as follows [6].

$$\mathbf{S} = \begin{bmatrix} -T_{12}^{-1}T_{12} & T_{12}^{-1} \\ T_{21} - T_{22}T_{12}^{-1}T_{11} & T_{22}T_{12}^{-1} \end{bmatrix} \tag{2.73}$$

Now, with the definition of transfer matrix in equation (2.72) and the manipulation in expression (2.73) the stiffness matrix is

$$\begin{bmatrix} q_1 \\ q_2 \end{bmatrix} = \begin{bmatrix} -i\cot(kx)/Z_f & i/Z_f\sin(kx) \\ i/Z_f\sin(kx) & -i\cot(kx)/Z_f \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} \tag{2.74}$$

The matrix $\mathbf{S}$ is also known as the element mobility matrix. It relates amplitudes of pressure, p, and amplitudes of flow rate, q. This matrix is the structural analog for the force-displacement stiffness matrix hence, to find the pressure field across a pipe network, the same solver can be used. Note that $\mathbf{S}$ depends on the fluid wave-number, k, so this matrix is frequency dependent. This means that for each frequency analysed a different $\mathbf{S}$ needs to be calculated.

2.3.4 Fluid-Structure Interaction

Fluid-structure interaction is a dynamic phenomenon that causes a compliant system to move when pressure waves exert forces on the structure. This interaction is always caused by dynamic forces which act on fluid and pipe and can be divided into two main groups [4, 5]: distributed forces and local forces. Forces that act along the pipe are called distributed forces and it is caused by the fluid pressure what causes the pipe to develop axial stresses in the walls. This is refereed as Poisson coupling in connection with the Poisson coefficient, ν, that transform radial and hoop stresses into axial stress by the generalized Hook's law. Forces caused by fluid friction in the pipe walls are also distributed forces and are called friction coupling. Local forces act at specific points in a pipe system such as unrestrained valves, bends and tees and is generally more dominant compared with the other coupling mechanisms.

First, considering only internal pressure in the pipe system the radial and circumferential stresses are derived by Ugural and Fenster [28], Boresi et al. [29] and can be expresses as

$$\sigma_{rr}(r,\omega) = \frac{p(\omega)D_i^2}{D_e^2 - D_i^2} - \frac{D_i^2 D_e^2 p(\omega)}{4r^2(D_e^2 - D_i^2)} \tag{2.75}$$

$$\sigma_{\theta\theta}(r,\omega) = \frac{p(\omega)D_i^2}{D_e^2 - D_i^2} + \frac{D_i^2 D_e^2 p(\omega)}{4r^2(D_e^2 - D_i^2)} \tag{2.76}$$

where D_i is the internal diameter, D_e the external diameter and $p(\omega)$ in internal pressure, frequency dependent. Their sum eliminates the second term in both equations and eliminates the radial dependency so this sum is constant through the thickness of the pipe wall.

$$\sigma_{rr} + \sigma_{\theta\theta} = \frac{2p(\omega)D_i^2}{D_e^2 - D_i^2} \tag{2.77}$$

Given the stress-strain relations, the axial strain, ϵ_{xx}, generated by the internal pressure on a open tube element is

$$\epsilon_{xx} = \frac{1}{E}[\sigma_{zz} - \nu(\sigma_{rr} + \sigma_{\theta\theta})] \tag{2.78}$$

with $\sigma_{zz} = 0$ for an open ended pipe element.

The most important interaction mechanism is junction coupling. In most of the times, a pipe network consists of straight sections of pipe connected by elbows, tees and diameter changes. The local forces on this sections of pipe can be calculated with help from the Reynolds transport theory applied for momentum conservation. From fluid mechanics, e.g. White [23], this conservation takes the form

$$\sum \mathbf{F} = \int_V \left(\frac{\partial}{\partial t}\rho_f \mathbf{v}\right) dV + \int_A \rho_f \mathbf{V}(\mathbf{V}\cdot\mathbf{n})dA \tag{2.79}$$

where the term $\sum \mathbf{F}$ is the vector sum of all forces acting on the system. It includes surface forces as pressure and body forces like external forces and gravity. The first term in the right-hand side is the time change of the linear momentum of the contents inside the control volume. The second term is the net flow rate of linear momentum out of the control surface by mass flow.

As indicated before the force vector can be divided into pressure forces and external forces. If an external pressure force is acting on the control surface and since the normal unit vector $\mathbf{n}$ is defined as outward, the pressure force is

$$\mathbf{F}_p = -\int_A p\mathbf{n}dA \tag{2.80}$$

Now, the external forces are given by

$$\sum \mathbf{F} = \int_A p\mathbf{n}dA - \int_V \left(\frac{\partial}{\partial t}\rho_f \mathbf{v}\right) dV + \int_A \rho_f |\mathbf{V}| \mathbf{V}\mathbf{n}dA \tag{2.81}$$

Given the theory presented some simplifications and assumptions were performed. Firstly, the viscosity of the fluid is assumed negligible and thus eliminating the friction coupling between the fluid and the pipe wall. In equation (2.81) the last term of the right-hand side is zero because the forces due to change in momentum are neglected. This is a good approximation if the fluid density or the fluid velocity is small compared to the fluid pressure. The second term is also neglected because no changes within the control volume are considered. Therefore only fluid pressure forces at each end of the control surface are considered, and they are considered uniform across the inlet and outlet areas. For the equilibrium to be archived pipe internal forces need to compensate the vector sum of the pressure forces (Figure 2.8).

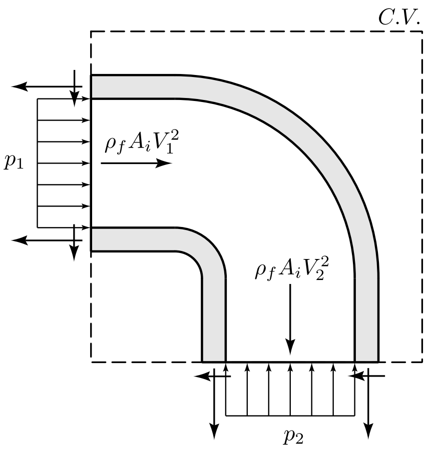


Figure 2.8: Elbow control volume

This forces due to pressure at each end of the pipe and along the inside of the pipeline will be analysed in the implementation section of this document.

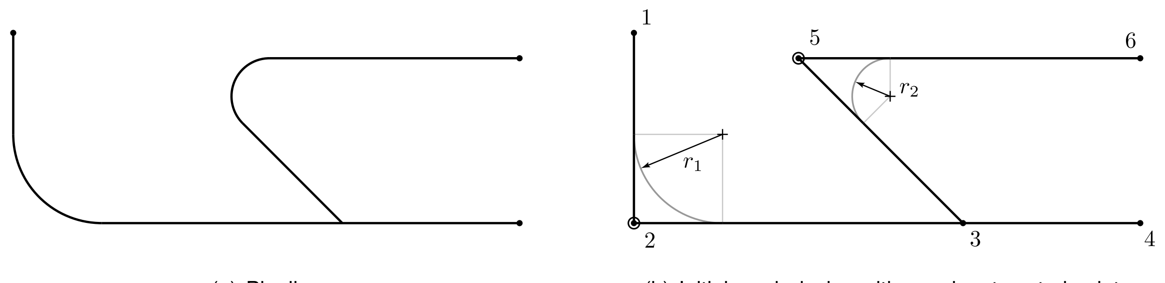
Chapter 3

Implementation

3.1 Mesh

3.1.1 Mesh Import

The program developed uses a text file, created by the user, to import the mesh. This file contains all the relevant information about node locations, element connectivity, element refinement, corner locations and corner radius. However, it is only necessary to provide an initial scheme of the final mesh, containing only straight lines. For example, to replicate the two-dimensional pipeline in figure 3.1a, the user needs to provide the coordinates for the six nodes in figure 3.1b and their connectivity. These are the minimum number of points possible since the user can add nodes between segments but those points are not necessary. The nodes that define a corner (nodes 2 and 5 in the example), pipe endings (nodes 1, 4 and 6) and branches (node 3) are the only necessary nodes needed to fully define a mesh, along with radius of bends defined by nodes 2 and 5.



(a) Pipeline (b) Initial mesh design with user input control points

Figure 3.1: Pipeline and user input control points

Mesh refinement is introduced to define more points along a single straight segment or to define a corner with line segments. This refinement is done element by element and corner by corner given the number of intermediate points each of the initial elements has. For example, if the user decides that each straight initial element has three segments and each corner is made out of four segments, the final mesh will look like the mesh represented in figure 3.2 after the refinement process is completed.

The final mesh is obtained by the program after performing all the operations of node numbering,

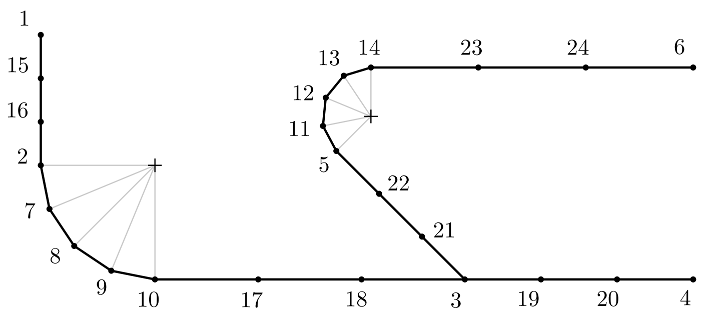


Figure 3.2: Final mesh with refinement and node numbering

node connectivity between the nodes and then calculating their coordinates. Next it will be explained all this process, starting from establishing element, given the initial user input element connectivity and refinement parameters, and then calculating node coordinates.

3.1.2 Element Connectivity

Since it is necessary to create nodes depending on mesh refinement, numbering them accordingly and then join them together to create elements, a method for this process was created. The main idea behind this process is to preserve numbering from the initial mesh, however in corners the initial node number will appear on either side of the corner arc. Refinement of corners is done first because the start and end of the corner arc needs to be connected with respective line elements. Depending on their respective refinement, nodes will be created and information regarding initial and final arc nodes will be saved for refining line elements and calculating node coordinates. The same strategy is used for line elements, however, first is necessary to determine the start and end nodes for every single element because numbering changed after modifying corners, information that it is also saved for future use in mesh process.

Following the mesh example given in section 3.1.1 a possible element connectivity array given by the user for the mesh in figure (3.1b) is

$$\begin{bmatrix} 1 & 2 & 3 & 3 & 5 \\ 2 & 3 & 4 & 5 & 6 \end{bmatrix}$$

After all corners in the mesh have been processed the array is transformed into

$$\begin{bmatrix} 1 & 3 & 3 & 2 & 7 & 8 & 9 & 10 & 5 & 11 & 12 & 13 & 14 \\ 2 & 4 & 5 & 7 & 8 & 9 & 10 & 3 & 11 & 12 & 13 & 14 & 6 \end{bmatrix}$$

and taking into account the start and end points in each arc, the process of dividing straight elements into smaller ones is accomplished and the final connectivity array is given by

$$\begin{bmatrix} 2 & 7 & 8 & 9 & 5 & 11 & 12 & 13 & 1 & 15 & 16 & 10 & 17 & 18 & 3 & 19 & 20 & 3 & 21 & 22 & 14 & 23 & 24 \\ 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 2 & 17 & 18 & 3 & 19 & 20 & 4 & 21 & 22 & 5 & 23 & 24 & 6 \end{bmatrix}$$

There are many options to define the initial connectivity matrix for the same pipeline, and with some different matrix the final connectivity matrix is also going to be different.

3.1.3 Node Coordinates

Calculating node coordinates for straight line elements is straight forward (Figure 3.3). Given the line start and end points, A and B respectively, and the number of segments the element is going to be divided into, N_s, the new node coordinates, C_i, are given by

$$C_i = A + \mathbf{a}\left(\frac{i}{N_s}\right), \quad i = 1, ..., N_s - 1 \tag{3.1}$$

with

$$\mathbf{a} = B - A$$

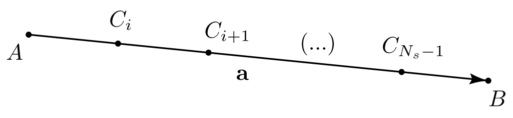


Figure 3.3: Refinement of straight elements

For every straight element this process is done as presented in listing 3.1 where the coordinates of the intermediate points are calculated.

Listing 3.1: Straight element refinement MATLAB code

```
for p_num_el = 1 : pre_number_elements
    segments = element_segments(p_num_el);
    if (segments > 1)
        % Element nodes
        node_a = line_segments(1, p_num_el);
        node_b = line_segments(2, p_num_el);

        % Vector that connects element node 1 to node 2
        v_a = node_coordinates(:, node_b) - node_coordinates(:, node_a);

        % Given the refinement number
        for li_seg = 1 : segments-1
            node_new = node_coordinates(:, node_a) + v_a * (li_seg / segments);
            node_coordinates(:, current_node) = node_new;
            current_node = current_node + 1;
        end
    end
end
```

A more difficult and time-consuming process is to determine nodal coordinates for the nodes that form corners. Given the corner node, C, both adjacent nodes, A and B, and corner radius, r, it is possible to calculate the arc start, S, end, E, and centre point, O, by first defining unit vectors (3.2) represented in figure 3.4.

$$\mathbf{a} = \frac{C - A}{\|C - A\|} \tag{3.2a}$$

$$\mathbf{b} = \frac{C - B}{\|C - B\|} \tag{3.2b}$$

$$\mathbf{c} = \frac{\mathbf{a} + \mathbf{b}}{\|\mathbf{a} + \mathbf{b}\|} \tag{3.2c}$$

and then, using basic trigonometric relations, the three defining arc points can be calculated by equations (3.3), with $\theta_{ab} = \arccos(\mathbf{a} \cdot \mathbf{b})$.

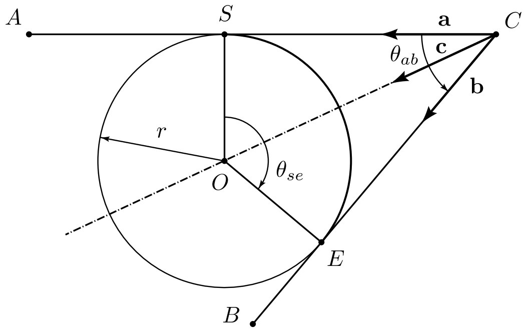


Figure 3.4: Important points and corner geometry

$$O = C + \mathbf{c}\left(\frac{r}{\sin\theta_{ab}}\right) \tag{3.3a}$$

$$S = C + \mathbf{a}\left(\frac{r}{\tan\theta_{ab}}\right) \tag{3.3b}$$

$$E = C + \mathbf{b}\left(\frac{r}{\tan\theta_{ab}}\right) \tag{3.3c}$$

With those three arc defining points, the vectors $\mathbf{s}$ and $\mathbf{e}$ can defined as indicated in figure 3.5. To calculate all the inside arc points, to form multiple elements, vector $\mathbf{s}$ must be rotated $N_s - 1$ times by an angle of $\theta_i = \theta_{se}/N_s$ where N_s is the number of elements that form the corner and $\theta_{se} = 2\pi - \theta_{ab}$. This three-dimensional vector rotation around an axis normal to the arc plane is performed using Olinde Rodrigues formula [30].

If $\mathbf{v}$ is a vector and $\mathbf{k}$ is a unit vector describing an axis of rotation about which $\mathbf{v}$ rotates by an angle θ, according to the right-hand rule (Figure 3.5), the Olinde Rodrigues formula [31] for the rotated vector $\mathbf{v}_{rot}$ is

$$\mathbf{v}_{rot} = \mathbf{v}\cos\theta + (\mathbf{k} \times \mathbf{v})\sin\theta + \mathbf{k}(\mathbf{k} \cdot \mathbf{v})(1 - \cos\theta) \tag{3.4}$$

with $\mathbf{k}$ defined as normal to the corner plane and with the right direction because of the arc start and

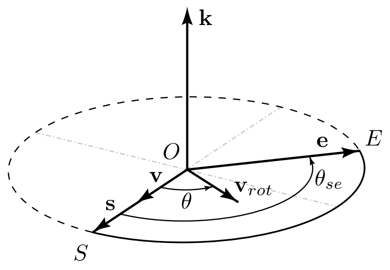


Figure 3.5: Vector rotation around axis

end points.

$$\mathbf{k} = \frac{\mathbf{s} \times \mathbf{e}}{\|\mathbf{s} \times \mathbf{e}\|} \tag{3.5}$$

Since the vector $\mathbf{v}$ we wish to rotate is in the plane formed by vectors $\mathbf{a}$ and $\mathbf{b}$ the term $\mathbf{k}(\mathbf{k} \cdot \mathbf{v})$ in equation (3.4) is zero because vectors $\mathbf{v}$ and $\mathbf{k}$ are orthogonal.

In matrix form, the transformation matrix, $\mathbf{R}$, between two vectors, $\mathbf{v}_{rot} = \mathbf{R}\mathbf{v}$, is given by

$$\mathbf{R} = \mathbf{I} + (\sin\theta)\mathbf{K} + (1 - \cos\theta)\mathbf{K}^2 \tag{3.6}$$

with

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{K} = \begin{bmatrix} 0 & -k_z & k_y \\ k_z & 0 & -k_x \\ -k_y & k_x & 0 \end{bmatrix} \tag{3.7}$$

This process code is presented in listing 3.2 where with the three points, radius and number of elements, the arc intermediate points are calculated.

Listing 3.2: Corner element refinement MATLAB code

```
for num_co = 1 : number_corners~
    % Corner properties
    node = corner_nodes(num_co);
    radius = corner_radius(num_co);
    segments = corner_segments(num_co);

    % Corner lines nodes
    node_a = corner_s_e(1, num_co);
    node_b = corner_s_e(2, num_co);

    % Unit vectors that connects center point to the start and end points
    v_a = current_n_c(:, node_a) - current_n_c(:, node);
    v_b = current_n_c(:, node_b) - current_n_c(:, node);
    v_a = v_a / norm(v_a);
    v_b = v_b / norm(v_b);

    % Angle between vectors and determination of distances to points S and E
    theta_ab = acos(dot(v_a, v_b));
    d_e = radius / tan(theta_ab / 2);
    d_i = radius / sin(theta_ab / 2);

    v_c = v_a + v_b;
    v_c = v_c / norm(v_c);

    % Arc center, start and end points
    center_p = current_n_c(:, node) + v_c * d_i;
    start_p = current_n_c(:, node) + v_a * d_e;
    end_p = current_n_c(:, node) + v_b * d_e;

    % Unit vector from center to start and end points
    v_ca = start_p - center_p;
    v_cb = end_p - center_p;
    v_ca = v_ca / norm(v_ca);
    v_cb = v_cb / norm(v_cb);

    % Normal vector to corner plane
    v_n = cross(v_ca, v_cb);
    v_n = v_n / norm(v_n);

    theta_cacb = pi - theta_ab;
    dtheta_cacb = theta_cacb / segments;
    t = 0;

    node_first = center_p + radius * v_ca;
    node_coordinates(:, node) = node_first;

    for s = 1 : segments
        t = t + dtheta_cacb;

        % Matrix notation Rodrigues formula
        I = [1, 0, 0; 0, 1, 0; 0, 0, 1];
        K = [0, -v_n(3), v_n(2); v_n(3), 0, -v_n(1); -v_n(2), v_n(1), 0];
        R = I + sin(t) * K + (1 - cos(t)) * K^2;
        v_ca_rot = R * v_ca;

        node_new = center_p + radius * v_ca_rot;
        node_coordinates(:, current_node) = node_new;

        current_node = current_node + 1;
    end
    current_n_c = node_coordinates(:, 1:current_node-1);
end
```

3.2 Structural Procedure

3.2.1 Element Transformation Matrix

The element stiffness and mass matrices presented in sections 2.1.2 and 2.1.3 are obtained in a local coordinate system. To transform the element matrices into the global coordinate system it is required a transformation of each element to account for the differences in orientation of all local coordinate systems in three-dimensional space.

Different authors use different methods to develop element transformation matrix in spatial coordinates. Liu and Quek [32], Rao [8], Meek [33] and Logan [34] use different approaches but the one that was implemented was based on Logan [34] with the implementation of Ferreira [35] and that derivation is now presented.

First it was established the sign convection used in figure 2.1. The local axis (x',y',z') is defined with x' being positive from node 1 to node 2, then y' and z' being the principal axis for which the second moment of area is minimum and maximum, respectively. Their direction is established by the right-hand rule. But since all the structural elements are pipes, the second moment of area with respect to any axis is equal to the other one. The definition of the y' local axis relative of the global coordinate system is that y' is perpendicular to z and x' ($y' = z \times x'$).

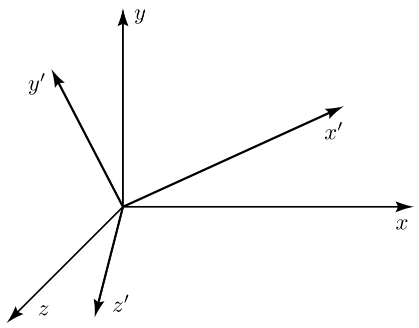


Figure 3.6: Global and local coordinate system

The transformation matrix $\mathbf{T}$ to transform local vector into global vector is given by

$$\mathbf{T} = \begin{bmatrix} \mathbf{R}_{(3\times3)} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{(3\times3)} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R}_{(3\times3)} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{R}_{(3\times3)} \end{bmatrix} \tag{3.8}$$

where $\mathbf{T}$ is a 12×12 symmetric matrix given the twelve degrees of freedom per element. The sub-matrix $\mathbf{R}$ is given as

$$\mathbf{R} = \begin{bmatrix} C_{xx'} & C_{yx'} & C_{zx'} \\ C_{xy'} & C_{yy'} & C_{zy'} \\ C_{xz'} & C_{yz'} & C_{zz'} \end{bmatrix} \tag{3.9}$$

All elements in matrix $\mathbf{R}$ denote the direction cosines. For example, $C_{xx'}$, $C_{yx'}$ and $C_{zx'}$, represent the direction cosines of the x' axis with respect to the global x, y, z axes. This is,

$$x' = (\cos\theta_{xx'})\mathbf{i} + (\cos\theta_{yx'})\mathbf{j} + (\cos\theta_{zx'})\mathbf{k} \tag{3.10}$$

It can be seen that finding the direction cosines of the x' axis is easy since

$$\cos\theta_{xx'} = \frac{x_2 - x_1}{L} = l \tag{3.11a}$$
$$\cos\theta_{yx'} = \frac{y_2 - y_1}{L} = m \tag{3.11b}$$
$$\cos\theta_{zx'} = \frac{z_2 - z_1}{L} = n \tag{3.11c}$$

where (x_1, y_1, z_1) and (x_2, y_2, z_3) are the node coordinates of an element. The element's length, L, is given by $L = [(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2]^{1/2}$.

Was opted to define the $\mathbf{y}'$ axis to be perpendicular to the $\mathbf{x}'$ and $\mathbf{z}$ axes. Therefore,

$$y' = z \times x' = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ 0 & 0 & 1 \\ l & m & n \end{vmatrix} \tag{3.12}$$

Since the x' and z axes can make an arbitrary angle, the cross product between them needs to be normalized, as follows:

$$y' = -\left(\frac{m}{D}\right)\mathbf{i} + \left(\frac{l}{D}\right)\mathbf{j} \tag{3.13}$$
$$D = (l^2 + m^2)^{1/2} \tag{3.14}$$

The remaining z' axis is simply determined by the orthogonality condition $z' = x' \times y'$, so

$$z' = x' \times y' = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ l & m & n \\ -\frac{m}{D} & \frac{l}{D} & 0 \end{vmatrix} \tag{3.15}$$

$$z' = -\left(\frac{ln}{D}\right)\mathbf{i} + \left(\frac{mn}{D}\right)\mathbf{j} + D\mathbf{k} \tag{3.16}$$

Combining equations (3.10), (3.13) and (3.16) the transformation matrix becomes

$$\mathbf{R} = \begin{bmatrix} l & m & n \\ -\frac{m}{D} & \frac{l}{D} & 0 \\ -\frac{ln}{D} & \frac{mn}{D} & D \end{bmatrix} \tag{3.17}$$

A problem arises when the local x' axis coincides with the global z axis. In this situation the local $\mathbf{y}'$

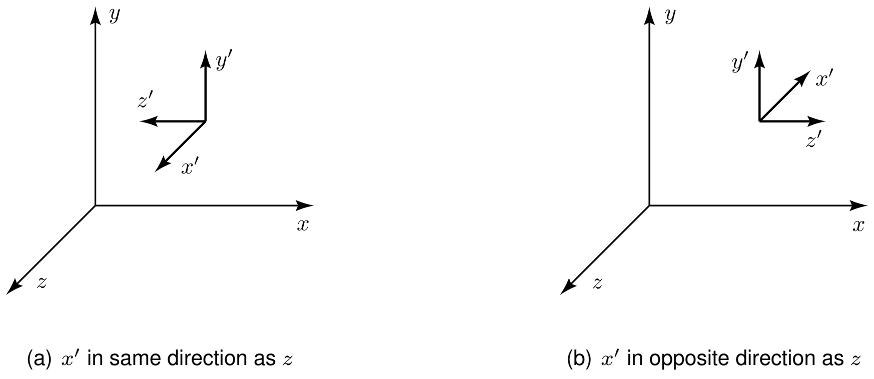
axis becomes uncertain. With a small angle between x' and z it can be seen that the direction of the y' axis is going to close to the global y axis. So, in this particular case the local y' axis is selected as the global y axis. Depending on the direction of the x' axis there will be two different transformations. If the positive x' is in the same direction as z, $\mathbf{R}$ becomes

$$\mathbf{R} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \tag{3.18}$$

For the opposite direction it becomes

$$\mathbf{R} = \begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \tag{3.19}$$

Both these matrices, (3.18) and (3.19) can be easily demonstrated with the help of figure 3.7. This the calculation of this transformation matrix needs to be perform to all element in the system and the code is presented in listing 3.3.



y
y'
z'
x'
x
z
(a) x' in same direction as z
y
y'
x'
z'
x
z
(b) x' in opposite direction as z

Figure 3.7: Special cases of transformation between local and global coordinate systems

Listing 3.3: Transformation matrix MATLAB code

```
% Element length projection
dx = xx(node2) - xx(node1);
dy = yy(node2) - yy(node1);
dz = zz(node2) - zz(node1);
L = (dx^2 + dy^2 + dz^2)^0.5;

if (dx == 0 && dy == 0)
    if (dz > 0)
        R = [0, 0, 1; 0, 1, 0; -1, 0, 0];
    else
        R = [0, 0, -1; 0, 1, 0; 1, 0, 0];
    end
else
    CXx = dx / L;
    CYx = dy / L;
    CZx = dz / L;
    D = sqrt(CXx^2 + CYx^2);
    CXy = -CYx / D;
    CYy = CXx / D;
    CZy = 0;
    CXz = -CXx * CZx / D;
    CYz = -CYx * CZx / D;
    CZz = D;
    R = [CXx, CYx, CZx; CXy, CYy, CZy; CXz, CYz, CZz];
end
% Transformation matrix
T = [...
    R, zeros(3, 9); zeros(3, 3), R, zeros(3, 6); ...
    zeros(3, 6), R, zeros(3, 3); zeros(3, 9), R];
```

3.2.2 Stiffness and Mass matrices

Both Euler-Bernoulli and Timoshenko element stiffness and mass matrices implemented in MATLAB are presented in listing 3.4 and listing 3.5, respectively.

Listing 3.4: Euler-Bernoulli element stiffness and mass matrices MATLAB code

```
% Local stiffness matrix
k1 = E * A / L;        k2 = 12 * E * Iz / L^3; k3 = 6 * E * Iz / L^2;
k4 = 4 * E * Iz / L;   k5 = 2 * E * Iz / L;     k6 = 12 * E * Iy / L^3;
k7 = 6 * E * Iy / L^2; k8 = 4 * E * Iy / L;     k9 = 2 * E * Iy / L; k10 = G * J / L;
a1 = [k1, 0, 0; 0, k2, 0; 0, 0, k6]; b1 = [0, 0, 0; 0, 0, k3; 0, -k7, 0];
c1 = [k10, 0, 0; 0, k8, 0; 0, 0, k4]; d1 = [-k10, 0, 0; 0, k9, 0; 0, 0, k5];
Kl = [...
    a1, b1, -a1, b1; ...
    b1', c1, (-b1)', d1; ...
    (-a1)', -b1, a1, -b1; ...
    b1', d1', (-b1)', c1];

% Local mass matrix
a2 = [140, 0, 0; 0, 156, 0; 0, 0, 156];          b2 = [0, 0, 0; 0, 0, 22*L; 0, -22*L, 0];
c2 = [70, 0, 0; 0, 54, 0; 0, 0, 54];             d2 = [0, 0, 0; 0, 0, -13*L; 0, 13*L, 0];
e2 = [140*J/A, 0, 0; 0, 4*L^2, 0; 0, 0, 4*L^2];
f2 = [70*J/A, 0, 0; 0, -3*L^2, 0; 0, 0, -3*L^2];
Ml = Rho * A * L / 420 * [...
    a2, b2, c2, d2; ...
    b2', e2, d2, f2; ...
    c2', d2', a2, -b2; ...
    d2', f2', (-b2)', e2];

% Local to global transformation
K0 = T' * Kl * T;
M0 = T' * Ml * T;
```

Listing 3.5: Timoshenko element stiffness and mass matrices MATLAB code

```
% Local stiffness matrix
Psi_y = 12 * E * Iy / (G * k * A * L^2); Psi_z = 12 * E * Iz / (G * k * A * L^2);
k1 = E * A / L; k10 = G * J / L;
k2 = 12 * E * Iz / ((1 + Psi_y) * L^3);         k3 = 6 * E * Iz / ((1 + Psi_y) * L^2);
k4 = (4 + Psi_y) * E * Iz / ((1 + Psi_y) * L); k5 = (2 - Psi_y) * E * Iz / ((1 + Psi_y) * L);
k6 = 12 * E * Iy / ((1 + Psi_z) * L^3);         k7 = 6 * E * Iy / ((1 + Psi_z) * L^2);
k8 = (4 + Psi_z) * E * Iy / ((1 + Psi_z) * L); k9 = (2 - Psi_z) * E * Iy / ((1 + Psi_z) * L);
a1 = [k1, 0, 0; 0, k2, 0; 0, 0, k6];  b1 = [0, 0, 0; 0, 0, k3; 0, -k7, 0];
c1 = [k10, 0, 0; 0, k8, 0; 0, 0, k4]; d1 = [-k10, 0, 0; 0, k9, 0; 0, 0, k5];
Kl = [...
    a1, b1, -a1, b1; ...
    b1', c1, (-b1)', d1; ...
    (-a1)', -b1, a1, -b1; ...
    b1', d1', (-b1)', c1];
```

```
% Local mass matrix
a2 = [140, 0, 0; 0, 156+504*Iz/A/L^2, 0; 0, 0, 156+504*Iy/A/L^2];
b2 = [0, 0, 0; 0, 0, 22*L+42*Iz/A/L; 0, -22*L-42*Iy/A/L, 0];
c2 = [70, 0, 0; 0, 54-504*Iz/A/L^2, 0; 0, 0, 54-504*Iy/A/L^2];
d2 = [0, 0, 0; 0, 0, -13*L-42*Iz/A/L; 0, 13*L+42*Iy/A/L, 0];
e2 = [140*J/A, 0, 0; 0, 4*L^2+56*Iy/A, 0; 0, 0, 4*L^2+56*Iz/A];
f2 = [70*J/A, 0, 0; 0, -3*L^2-28*Iy/A, 0; 0, 0, -3*L^2-28*Iz/A];
Ml = Rho * A * L / 420 * [...
    a2, b2, c2, d2; ...
    b2', e2, d2, f2; ...
    c2', d2', a2, -b2; ...
    d2', f2', (-b2)', e2];

% Local to global transformation
K0 = T' * Kl * T;
M0 = T' * Ml * T;
```

3.2.3 Matrix Assembly

The assembly is a procedure in which the elementary matrices and the element connectivity matrix are used to obtain the global matrices. This process depend on mesh properties such as number of nodes, number of elements and number of degrees of freedom per node. In the structural part, the three dimensional beam element has two nodes and each node has six degrees of freedom. Therefore, elementary matrices have 12×12 dimension. So, the dimensions of the global matrices are $6 \times number\_nodes$.

Since global matrix dimensions are exclusive dependent on mesh data, it can be initialized with the exact dimensions before the calculation of local elementary matrices. In MATLAB, this is done with the function *zeros(n)* that return a n-by-n matrix of zeros.

The assembly process is done by modifying the global matrix entries that correspond to the local element degrees of freedom. First, given the two node numbers that form an element, the degrees of freedom of node j must to be calculated from

$$DOF_i = 6(j-1) + i, \quad i = 1, ..., 6 \tag{3.20}$$

Listing 3.6: Element degrees-of-freedom by node numbers MATLAB code

```
% Element nodes
node1 = element_nodes(1, num_el);
node2 = element_nodes(2, num_el);
% Element degrees of freedom
element_dof = [...
    6*node1-5, 6*node1-4, 6*node1-3, 6*node1-2, 6*node1-1, 6*node1, ...
    6*node2-5, 6*node2-4, 6*node2-3, 6*node2-2, 6*node2-1, 6*node2];
```

Now, an array, $element\_dof$ (listing 3.6), can be constructed with the twelve degrees of freedom of the current element and the assembly process is given by listing 3.7 and it is done for every element in the mesh.

Listing 3.7: Assembly process MATLAB code

```
KK(element_dof, element_dof) = KK(element_dof, element_dof) + K0;
```

```
MM(element_dof, element_dof) = MM(element_dof, element_dof) + M0;
```

For the acoustic stiffness matrix this process is easier because each node has only one degree of freedom, so, this degree of freedom has the same number as the node.

3.2.4 Adding Spring

The program allows the user to add a linear spring in any node and at any off the three global directions. This is done after the structural stiffness matrix is completed and it is perform adding the spring stiffness in N/m to the global stiffness matrix in the correct location. Given the node transnational degrees of freedom, given by equation (3.20) for only the first three terms, that calculation is performed by

$$\mathbf{K}(DOF_1, DOF_2, DOF_3) = \mathbf{K}(DOF_1, DOF_2, DOF_3) + \begin{bmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & k_z \end{bmatrix} \tag{3.21}$$

where k_x, k_y and k_z are the spring stiffness in the three global directions (same directions as DOF_1, DOF_2 and DOF_3) as presented in listing 3.8.

Listing 3.8: Adding spring MATLAB code

```
for num_sn = 1 : number_spring_nodes
    % Input dialog for the user
    ans2 = inputdlg({'Node:', 'kx␣[N/m]', 'ky␣[N/m]', 'kz␣[N/m]'}, 'Add␣Spring');
    spring_nodes(num_sn) = str2double(ans2{1});
    node = spring_nodes(num_sn);
    node_dof = [6*node-5, 6*node-4, 6*node-3];
    kkx = str2double(ans{2});
    kky = str2double(ans{3});
    kkz = str2double(ans{4});
    KK(node_dof, node_dof) = KK(node_dof, node_dof) + diag([kkx, kky, kkz]);
end
```

3.2.5 Free Vibration Analysis

When performing the modal analysis, the eigenvalue problem (2.42) is solved by a MATLAB in-built function, then the vibration frequencies and vector of mode shapes are sorted from the lowest frequency to the highest as presented in listing 3.9. This type of analysis processes the non-zero displacements boundary conditions as fixed boundary conditions, so the matrices used only take into account free degrees of freedom.

Listing 3.9: Structural modal analysis MATLAB code

```
% Using only free degrees of freedom from both stiffness and mass matrices
K = KK(free_dofs, free_dofs);
M = MM(free_dofs, free_dofs);

[vec, val] = eig(K, M);

[natural_freq, ind] = sort(real(sqrt(diag(val))));

freq_modes = zeros(number_dofs, number_free);
freq_modes(free_dofs, :) = vec(:, ind);
```

3.2.6 Forced Vibration Analysis

In this analysis equation (2.46) is solved for the harmonic displacements for a given array of frequencies. For each of the frequencies, the dynamic matrix is calculated and with the known vector of forces, the displacements are calculated.

Listing 3.10: Structural harmonic analysis MATLAB code

```
j = sqrt(-1);

for num_fq = 1 : number_freq
    freq_hz = harm_freq(num_fq);
    freq_rad = freq_hz * 2 * pi;
    dynamic_matrix = KK - freq_rad^2 * MM + j * freq_rad * CC;
    d = dynamic_matrix(free_dofs, free_dofs) \ FF(free_dofs);
    harm_displ(free_dofs, num_fq) = (real(d).^2 + imag(d).^2).^0.5;
end
```

3.2.7 Internal Fluid Flow

The program developed offers the possibility to analyse a pipeline with the effects of with velocity fluid flow. This analysis only changes the stiffness and mass matrices of the structural pipe element, already calculated. Since the size of this matrices is $number\_dofs \times number\_dofs$ where $number\_dofs$ is the global number of degrees of freedom of the system, the process of defining the new stiffness and mass matrices of the system is simply an arithmetic calculation expressed in equation (2.51). In a more complex pipe network this particular implementation explained in section 2.2 has its drawbacks. First the effects of gravity and pressure are not developed in this implementation. Second the program still only allows one value for the fluid flow velocity. This is a very rough approximation given the fact that for pipelines with side branches, tees, at some given angles, the velocities change from pipe to pipe because of mass conservation inside the whole system.

The calculation of the stiffness, mass and damping element matrices and their respective transformation and assembly process is described in listing 3.11 and the modification of the global structural plus fluid matrices is presented in listing 3.12.

Listing 3.11: Local stiffness, mass and damping internal fluid matrices MATLAB code

```
% Local stiffness matrix
a1 = [0, 0, 0; 0, 36, 0; 0, 0, 36];        b1 = [0, 0, 0; 0, 0, 3*L; 0, -3*L, 0];
c1 = [0, 0, 0; 0, 4*L^2, 0; 0, 0, 4*L^2]; d1 = [0, 0, 0; 0, -L^2, 0; 0, 0, -L^2];
Kl = Rhof * Ai * v^2 / (30 * L) * [...
    a1, b1, -a1, b1; ...
    b1', c1, b1, d1; ...
    (-a1)', b1', a1, -b1; ...
    b1', d1', (-b1)', c1];

% Local mass matrix
a2 = [0, 0, 0; 0, 156, 0; 0, 0, 156];      b2 = [0, 0, 0; 0, 0, 22*L; 0, -22*L, 0];
c2 = [0, 0, 0; 0, 54, 0; 0, 0, 54];        d2 = [0, 0, 0; 0, 0, -13*L; 0, 13*L, 0];
e2 = [0, 0, 0; 0, 4*L^2, 0; 0, 0, 4*L^2]; f2 = [0, 0, 0; 0, -3*L^2, 0; 0, 0, -3*L^2];
Ml = Rhof * Ai * L / 420 * [...
    a2, b2, c2, d2; ...
    b2', e2, d2, f2; ...
    c2', d2', a2, -b2; ...
    d2', f2', (-b2)', e2];
```

```
% Local damping matrix
a1 = [0, 0, 0; 0, -30, 0; 0, 0, -30]; b1 = [0, 0, 0; 0, 0, -6*L; 0, 6*L, 0];
c1 = [0, 0, 0; 0, 0, 0; 0, 0, 0];     d1 = [0, 0, 0; 0, L^2, 0; 0, 0, L^2];
Cl = Rhof * Ai * v / 30 * [...
    a1, b1, a1, -b1; ...
    b1, c1, -b1, d1; ...
    -a1, -b1, -a1, b1; ...
    -b1, -d1, b1, c1];

% Local to global transformation
K0 = T' * Kl * T;
M0 = T' * Ml * T;
C0 = T' * Cl * T;
% Assembly
KKf(element_dof, element_dof) = KKf(element_dof, element_dof) + K0;
MMf(element_dof, element_dof) = MMf(element_dof, element_dof) + M0;
CCf(element_dof, element_dof) = CCf(element_dof, element_dof) + C0;
```

Listing 3.12: New global stiffness, mass and damping matrices MATLAB code

```
Structural.stiffness_matrix = KK - KKf;
Structural.mass_matrix = MM + MMf;
Structural.damping_matrix = CC + CCf;
```

3.2.8 Plot Mode Shapes

The eigenvectors extracted from the structural modal analysis can be used to plot the mode shapes of the pipeline structure. The nodal displacements and rotations are given in global coordinates, so the first step is to, element by element, transform global coordinates to local coordinates. In section 3.2.1 it was defined how the local coordinates are presented: x' is given as the unit vector passing through both nodes; y' is defined as the cross product between x' and z; and z' is defined normally as $x' \times y'$. If the x' axis is parallel to z, the definition of y' is not possible, so in this special case the local axis are defined automatically as in figure (3.7). Using the transfer matrix, $\mathbf{T}$, the local coordinates nodal displacements, $\mathbf{u}_{local}$, are given by $\mathbf{u}_{local} = \mathbf{T}\,\mathbf{u}_{global}$, where $\mathbf{u}_{global}$ contains the twelve nodal displacements and rotation of an element.

Now, using shape functions defined in equations (2.23a) to (2.23j), an element can be represented by multiple points along the element, and since this representation is done in MATLAB using lines its not possible to represent graphically the torsional deformation of the structure.

Listing 3.13: Calculation of internal displacements for ploting mode shapes MATLAB code

```
    for num_pt = 1 : number_points
        current_point = node_coordinates(:, node1) + ux * L * unit_length(num_pt);
        % Axial
        phi1 = 1 - unit_length(num_pt);
        phi4 = unit_length(num_pt);
        % Transverse
        phi2 = 1 - 3 * unit_length(num_pt)^2 + 2 * unit_length(num_pt)^3;
        phi5 = 3 * unit_length(num_pt)^2 - 2 * unit_length(num_pt)^3;
        % Rotation
        phi3 = (unit_length(num_pt) - 2 * unit_length(num_pt)^2 + unit_length(num_pt)^3) * L;
        phi6 = (-unit_length(num_pt)^2 + unit_length(num_pt)^3) * L;
        rx = phi1 * f1 + phi4 * f7;
        ry = phi2 * f2 + phi3 * f6 + phi5 * f8 + phi6 * f12;
        rz = phi2 * f3 - phi3 * f5 + phi5 * f9 - phi6 * f11;
        x(:, num_pt, num_el) = current_point + rx * ux + ry * uy + rz * uz;
    end
```

3.3 Acoustic Procedure

3.3.1 Stiffness Matrix

To construct the frequency dependent stiffness matrix the fluid properties, pipe cross-section and element length are necessary. Since the stiffness matrix is dependent on the analysis frequency, the variable charged of storing that information is a three-dimensional array with a depth corresponding to the number of frequencies analysed.

Listing 3.14: Acoustic stiffness matrix MATLAB code

```
j = sqrt(-1);

Zf = rho_f * c_o / A_i;
kL = frequencies * 2 * pi / c_o * L;

for num_fq = 1 : number_freq
    c1 = -j * cos(kL(num_fq)) / (sin(kL(num_fq)) * Zf);
    c2 = j / (Zf * sin(kL(num_fq)));
    % Local stiffness matrix
    K0 = [c1, c2; c2, c1];
    % Assembly
    KKa(element_dof, element_dof, num_fq) = KKa(element_dof, element_dof, num_fq) + K0;
end
```

3.3.2 Pressure Field

The acoustic system of equation defined in section 2.3.3 is given by

$$\mathbf{q} = \mathbf{S}\,\mathbf{p} \tag{3.22}$$

Each of one-dimensional acoustic element has two nodes and each node has 1 degree-of-freedom. In this global system, the global pressure vector $\mathbf{p}$ is presented as the unknown variable but first boundary conditions must be applied. They can be classified as nodal volume velocity, nodal pressure or a nodal acoustic impedance [6]. Several method exist to solve this system with prescribed, non zero value, nodal pressures. The method used in this program is described in Rao [8]. To understand this method the global system is divided into

$$\begin{bmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} \\ \mathbf{S}_{21} & \mathbf{S}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{q}_1 \\ \mathbf{q}_2 \end{bmatrix} \tag{3.23}$$

where $\mathbf{p}_2$ is assumed to be the vector of specified nodal pressure degree-of-freedom and $\mathbf{p}_1$ the vector of free pressure nodes. Then $\mathbf{q}_1$ is be the vector of known volume velocities and $\mathbf{q}_2$ will be the vector of unknown volume velocities. Equation (3.23) can be written as

$$\mathbf{S}_{11}\,\mathbf{p}_1 + \mathbf{S}_{12}\,\mathbf{p}_2 = \mathbf{q}_1 \tag{3.24}$$

Hence equation (3.24) can be solved to obtain (Listing 3.15)

$$\mathbf{p}_1 = \mathbf{S}_{11}^{-1}(\mathbf{q}_1 - \mathbf{S}_{12}\,\mathbf{p}_2) \tag{3.25}$$

and the unknown vector $\mathbf{q}_1$ can now be found.

Listing 3.15: Acoustic harmonic analysis MATLAB code

```
free_dofs = Acoustic.free_dofs;
prescribed_values = Acoustic.prescribed_values;
volume_velocity_vector = Acoustic.volume_velocity_vector;
KKa = Acoustic.mobility_matrix;
harm_freq = Acoustic.frequencies;

number_freq = length(harm_freq);
harm_press = zeros(number_dofs, number_freq);

for num_fq = 1 : number_freq
    harm_press(free_dofs, num_fq) = KKa(free_dofs, free_dofs, num_fq) \ ...
        (volume_velocity_vector(free_dofs, num_fq) - ...
        KKa(free_dofs, prescribed_dofs, num_fq) * prescribed_values(num_fq);
end
```

3.3.3 Fluid-Structure Interaction

In section 2.3.4 the identification of the forces involved in the fluid-structure interaction was performed. From that, the only forces at it will be taken into account are the Poisson coupling force and the junction coupling force. Since, from the acoustic harmonic analysis the nodal pressures are calculated some approximations have to be perform to calculate the forces originated from those pressures. Given that in expression (2.77), the Poisson coupling force assumes a constant pressure along a pipe element, the average pressure at an element needs to be calculated to calculate the nodal forces that arise from the coupling mechanism. This average pressure is also used to calculate the forces acting on the nodes. The force acting on the nodes for given element is,

$$\mathbf{F} = [-F_x, 0, 0, 0, 0, 0, F_x, 0, 0, 0, 0, 0]' \tag{3.26}$$

where F_x is given sum of local pressure forces and the distributed Poisson coupling force, that is

$$F_x = p_{avg}A_i + A(-\nu(\sigma_{rr} + \sigma_{\theta\theta})) \tag{3.27}$$

Listing 3.16 presents the forces acting on both nodes of a given element that will be inserted into a new global force vector that is summed into an already existing external force vector (Listing 3.17).

Listing 3.16: Poisson and junction coupling forces MATLAB code

```
% Nodal pressures and average pressure
press_1 = harm_press(node1, num_fq);
press_2 = harm_press(node2, num_fq);
press_avg = (press_1 + press_2) / 2;

% Poisson and junction
```

```
stress_axial = (press_avg * Di^2) / (De^2 - Di^2);
f_p = - 2 * Nu * A * stress_axial;
f_j = press_avg * A_i;
F = f_j + f_p;
```

Listing 3.17: Coupling forces MATLAB code

```
ux = (node_coordinates(:, node2) - node_coordinates(:, node1)) / L;
F_global_node1 = -F * [ux(1); ux(2); ux(3)];
F_global_node2 = F * [ux(1); ux(2); ux(3)];
F_fluid(node1_dof) = F_fluid(node1_dof) + F_global_node1;
F_fluid(node2_dof) = F_fluid(node2_dof) + F_global_node2;
```

This approach is possible because it accurately represents the forces acting on pipelines when changes in velocity (speed or direction) occurs. For example, figure 3.8 from [36] shows different common pipeline sections and the forces acting on them given fluid the pressures. The elbow pipe section is analysed in more depth because it is the only section that doesn't use a straight pipe element. In figure 3.9 an $90°$ elbow is represented with the pressure forces acting on the control surface. The sum of the forces acting on that control volume has the outward bend direction, and if p_1 and p_2 are equal, the net force makes an $45°$ angle with the positive x axis.

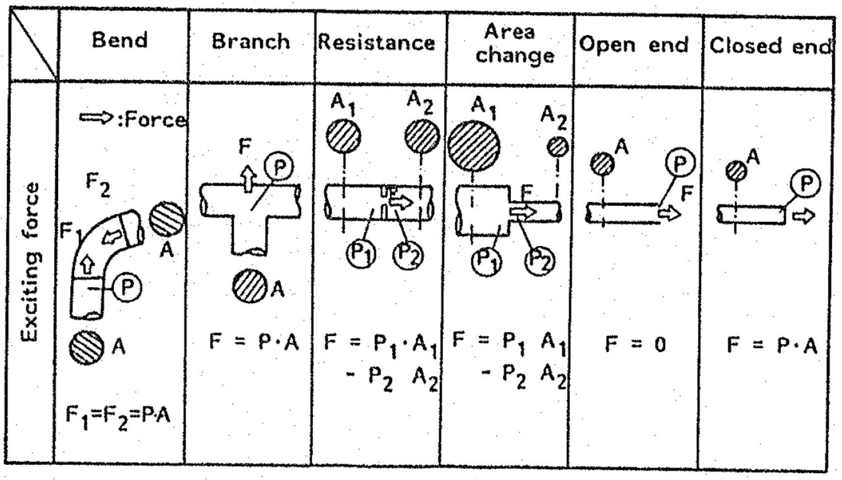


Figure 3.8: Piping sections excited by pressure pulsation [36]

If a sharp $90°$ bend, without any curvature (Figure 3.10), is modeled, the output force calculated has the same direction as described before. If that same bend is modeled with a non-zero bend radius and three elements defining that bend (Figure 3.11), the forces at each of the four nodes at compose the bend are calculated, and it is possible to verify that, the sum of the forces in the same control volume is the same, that is,

$$\mathbf{F}_1 + \mathbf{F}_2 + \mathbf{F}_3 + \mathbf{F}_4 = \mathbf{F} \tag{3.28}$$

This ensures that the pressure forces originated at bends and branches are accurately represented in the program developed. [37] also uses an one-dimensional fluid analysis to obtain the forces on the piping system and apply these forces, originated from elbows, tees and tapers, to the structural model. As seen in figure 3.12 the direction of the forces originated by the change in momentum are directed to

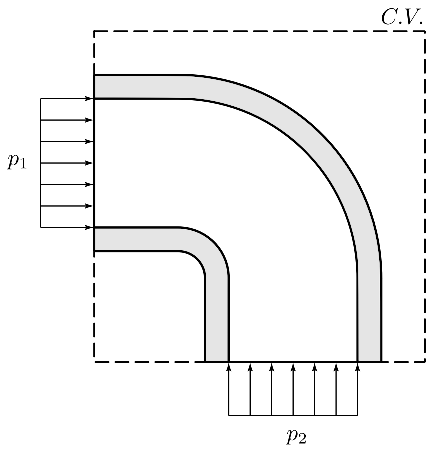


Figure 3.9: Pressure forces acting on the control surface of an elbow

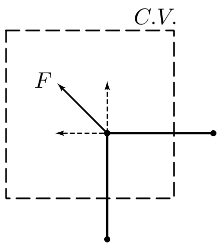


Figure 3.10: Resultant pressure forces acting on elbow defined by multiple elements

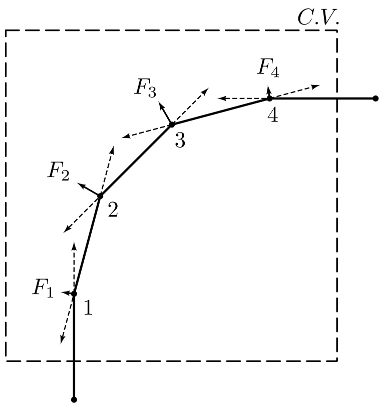


Figure 3.11: Second mode of an *L* structure

the outside of the elbow.

Since the pressure perturbation is harmonic (Equation (2.67)), this force vector is also harmonic and is used to perform the structural-acoustic coupled harmonic analysis that needs to be summed up to the existing external applied global force vector. This coupled analysis is very similar to the one presented in listing 3.10, one differs in the fact that the updated force vector is frequency dependent, so for each of the acoustic pressure fields frequencies a force vector is calculated and then used in this coupled analysis for the same frequency range (Listing 3.18).

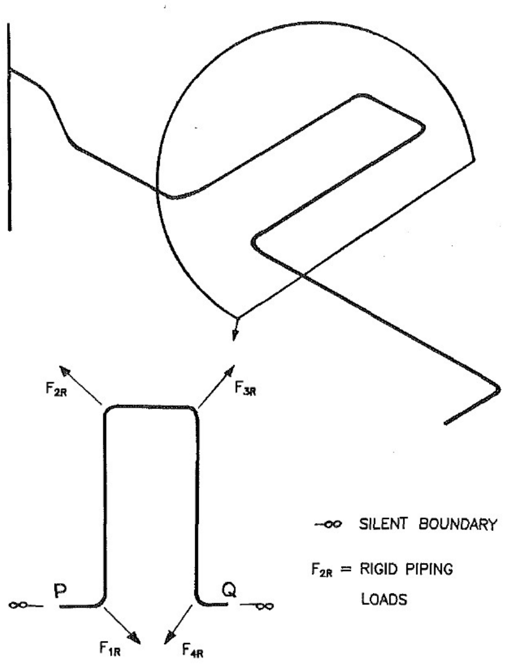


Figure 3.12: Forces due to pressure in piping system on elbows [37]

Listing 3.18: Coupled harmonic analysis MATLAB code

```
j = sqrt(-1);

for num_fq = 1 : number_freq
    freq_hz = harm_freq(num_fq);
    freq_rad = freq_hz * 2 * pi;
    dynamic_matrix = KK - freq_rad^2 * MM + j * freq_rad * CC;
    d = dynamic_matrix(free_dofs, free_dofs) \ FF(free_dofs, num_fq);
    harm_displ(free_dofs, num_fq) = (real(d).^2 + imag(d).^2).^0.5;
end
```

Chapter 4

Results

4.1 Structural Results

The initial steps to verify the computed code is to compare pipe structural element with analytical result and to compare the different beam theories used and the benefits of each other depending on the application.

Analytical calculations, present in the literature, only account for simple two dimensional beams considering the Euler-Bernoulli theory. These results predict both natural vibrations frequencies and vibration modes for beams with different boundary conditions and for different modes of vibration. Given that these results are two-dimensional and the implementation is done in three dimensions, only the x-y plane (in-plane) deformations will be considered. Analytical natural frequencies and vibration modes for the first four modes of the transverse vibration of a beam are presented in figure 8.15 from Rao [38] for common boundary conditions.

The first case study is done considering a pinned-pinned beam. In this scenario the pinned nodes, apart from the three zero displacement degrees of freedom, the axial rotation also needs to be fixed to zero (Figure 4.1) because it can create an axial rotation rigid body motion that results in natural frequencies of zero value.

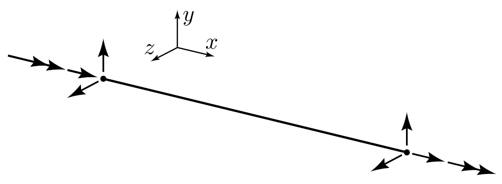


Figure 4.1: Fixed degrees of freedom for pinned-pinned pipe

In all this results the pipe material and geometry parameters are indicated in table 4.1

The differences between Euler-Bernoulli and Timoshenko beam models described in sections 2.1.2 and 2.1.3 are now analysed. The comparison between these models and analytical result based on Euler-Bernoulli is also presented. For the three first modes of vibrations of a pinned-pinned pipe, the

Table 4.1: Pipe material and geometry

Parameters	Values
Length	3.048 m
Modulus of elasticity	68.9 GPa
Poisson coefficient	0.3
External diameter	0.0254 m
Internal diameter	0.0221 m
Density	2699 kg/m^3

natural frequencies for each mode is calculated for an increasing number of elements in the FEM model. For the first mode, figure 4.2, the theoretical value based in Euler-Bernoulli model is represented with a grey margin representing a plus and minus percentage of that value.

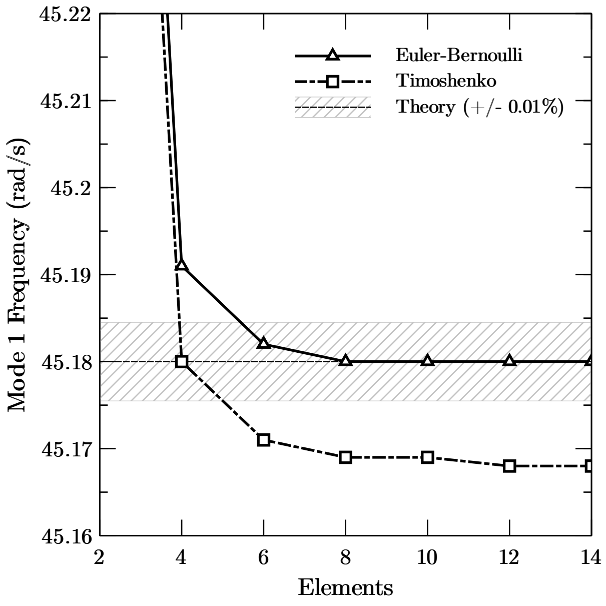


Figure 4.2: Mode 1 natural frequencies for pinned-pinned pipe

The Timoshenko model gives smaller natural frequency than the Euler-Bernoulli as predicted given the reduced stiffness of the model. With 6 elements the Euler-Bernoulli model is inside that analytical margin for the first mode. For the second and third modes, respectively given in figures 4.3 and 4.4, the differences between FEM models is more significant, seen by the grey margins. Also, it takes more elements to archive the same error margins, 12 elements for the second mode and 16 for the third.

An overview of these graphs is present in table 4.2 for up to 20 elements.

Table 4.2: Natural frequencies for different models and elements

Mode	Theory	Euler-Bernoulli			Timoshenko		
		8	12	20	8	12	20
1	45.180	45.180	45.180	45.180	45.169	45.168	45.168
2	180.718	180.765	180.728	180.720	180.592	180.551	180.541
3	406.616	407.139	406.722	406.630	406.293	405.844	405.735

Now, the default length of the pipe is reduced without modifying the cross-section geometry to study

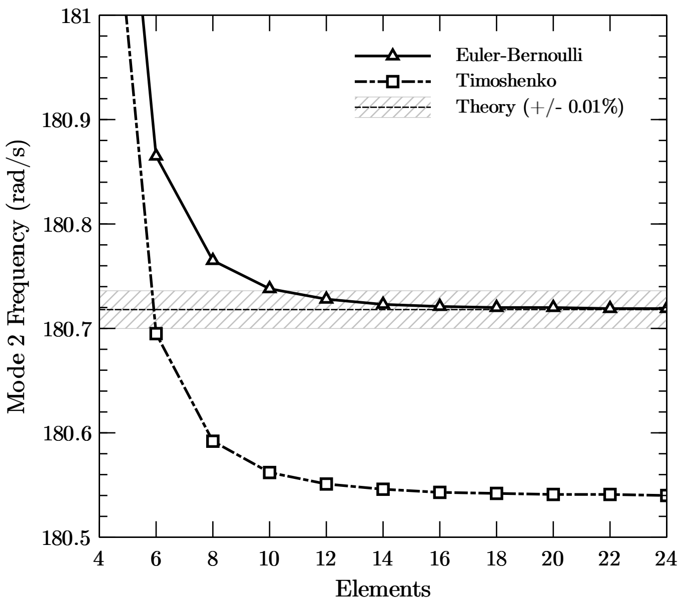


Figure 4.3: Mode 2 natural frequencies for pinned-pinned pipe

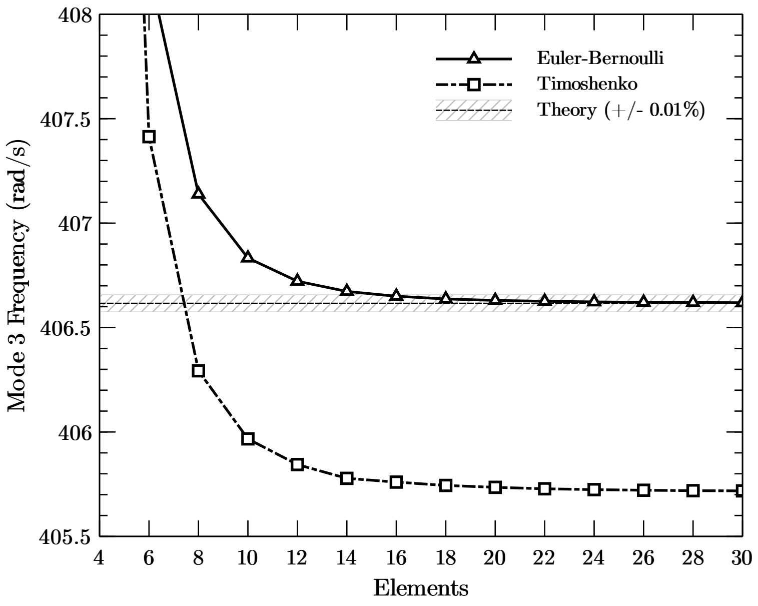


Figure 4.4: Mode 3 natural frequencies for pinned-pinned pipe

the influence of the pipe thickness-to-length ratio in both Euler-Bernoulli and Timoshenko beam theories. Instead of thickness it has used the gyration radius to better describe the section geometry. Figure 4.5 shows the ratio of frequencies between FEM models for the first three vibration frequencies for an decreasing pipe length.

These results show that the Timoshenko theory results are very similar to the Euler-Bernoulli results when r_x/L is small, however, the results show that the difference between theories tends to grow larger for a thicker beam. Remember that r_x is the radius on gyration of the cross-section around the x acxis. This difference can also be seen in table 4.3, where the non-dimensionalized frequency was used with a 20 number of elements.

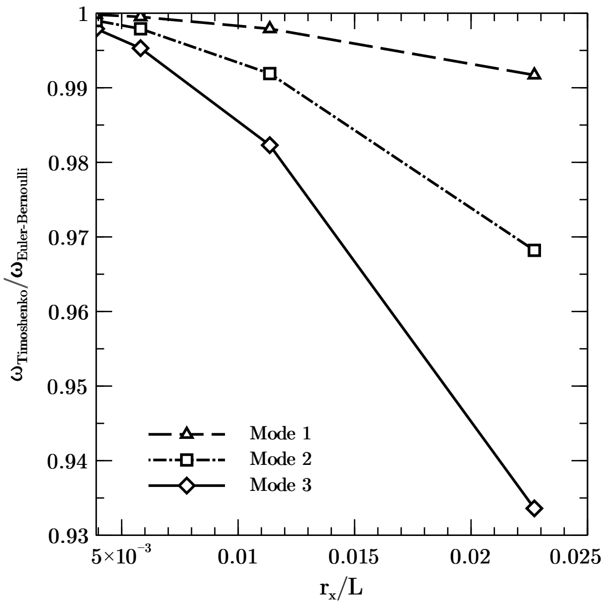


Figure 4.5: Length influence in FEM structural models with 20 elements

Table 4.3: Non-dimensional natural frequency ($\lambda^2 = \omega L^2\sqrt{\rho A/EI}$) for a Timoshenko pinned-pinned beam

Mode	Theory	Length (m)					
		3.048	2.548	2.048	1.548	1.048	0.548
1	3.1416	3.1412	3.1410	3.1407	3.1401	3.1383	3.1296
2	6.2832	6.2801	6.2788	6.2763	6.2712	6.2572	6.1901
3	9.4248	9.4146	9.4101	9.4020	9.3849	9.3385	9.1233

As the length decreases the computed natural frequencies tend to show some quantitative differences from the Euler-Bernoulli results. Similar results are provided by [35, 39] where the author used a different approach to develop Timoshenko model.

To verify mode shapes, three simple beam configurations will be used. Pinned-pinned, clamped-pinned and clamped-free. For this verification, Euler-Bernoulli theory has used and with 5 elements and each plotted with 10 intermediate points. For the pinned-pinned condition the first three modes of vibration are represented in figure 4.1

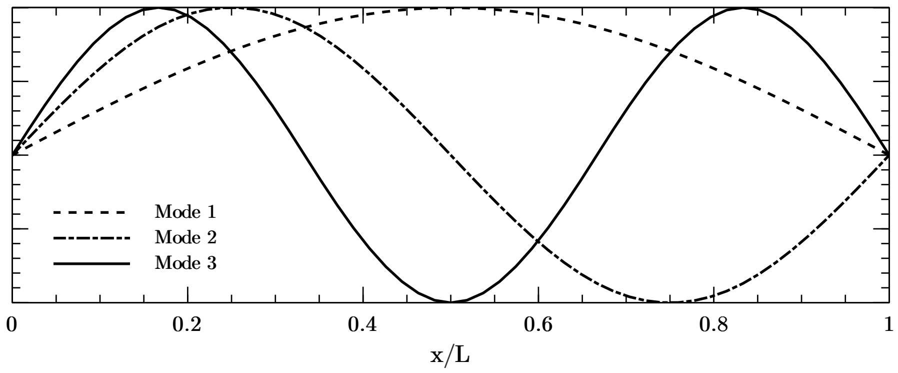


Figure 4.6: First 3 modes of vibration for pinned-pinned pipe

For the clamped-pinned and clamped-free configurations the fixed degrees of freedom and vibration modes are represented in figures 4.7, 4.8, 4.9, 4.9.

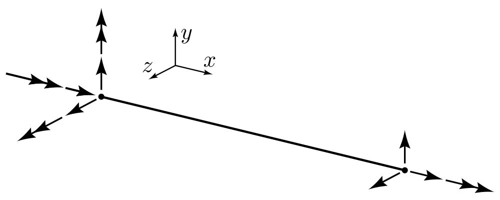


Figure 4.7: Fixed degrees of freedom for clamped-pinned pipe

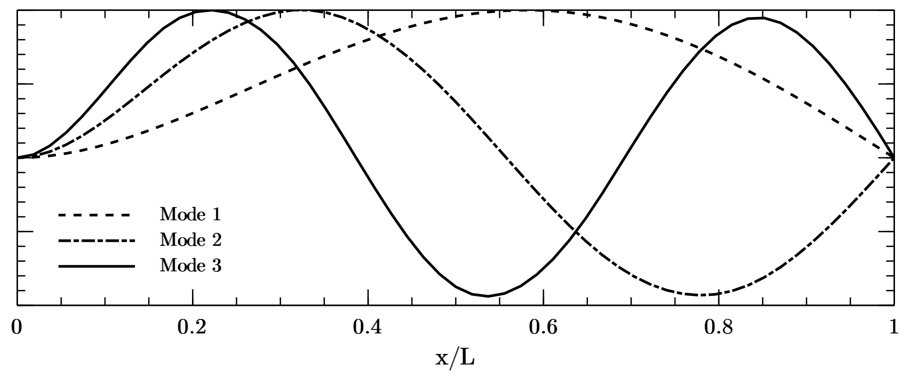


Figure 4.8: First 3 modes of vibration for clamped-pinned pipe

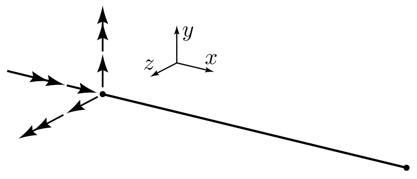


Figure 4.9: Fixed degrees of freedom for clamped-free pipe

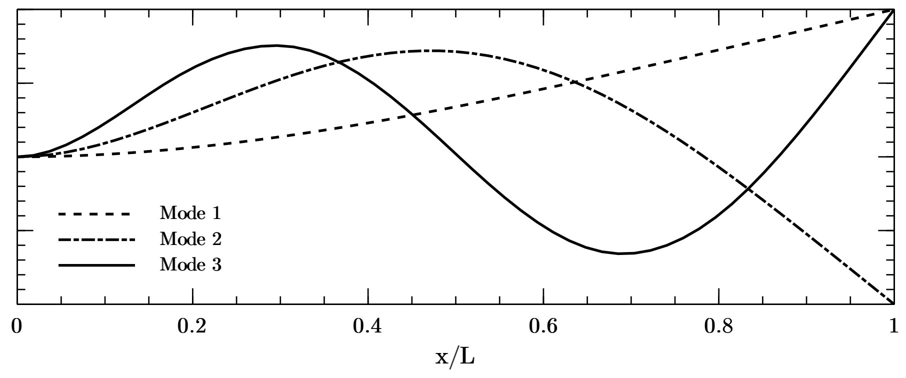


Figure 4.10: First 3 modes of vibration for clamped-free pipe

Given a more complex pipe system, like the one displayed in figure 4.11 with properties in table 4.4, *OpenPulse* program will be used to analyse the differences between models. A pipe with a small length to diameter ratio was used to showcase the differences between the two beam theories developed in this thesis and the timoshenko beam theory present in *OpenPulse* based on [16]. The planar pipe system is clamped on one side and free on the other.

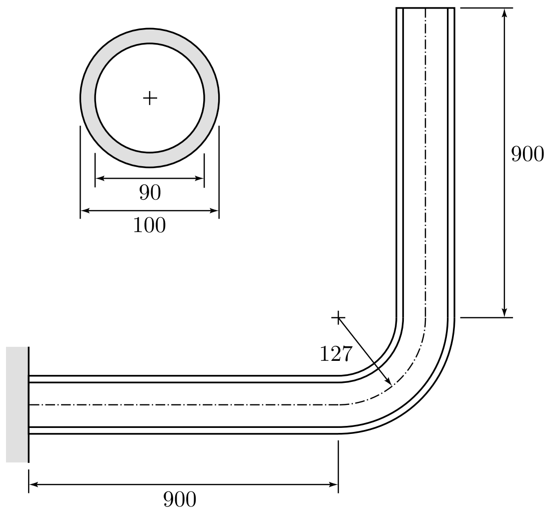


Figure 4.11: Clamped-free *L* pipe geometry and cross-section

Table 4.4: Pipe material and geometry for the clamped-free pipe

Parameters	Values
Pipes length	0.9 m
Elbow radius	0.127 m
Modulus of elasticity	210 GPa
Poisson coefficient	0.3
External diameter	0.1 m
Internal diameter	0.09 m
Density of pipe	7800 kg/m^3

The first six natural frequencies are displayed in table 4.5. Each of the straight pipe segments was divided into 20 elements and the $90°$ elbow divided into 10 elements, which equal to a total of 50 elements. In *OpenPulse* [7] a mesh with node spacing of 0.01 m was used which translated to around 200 nodes.

Table 4.5: Natural frequencies in Hz for the first six modes of vibration for the clamped-free *L* pipe

Mode	Model Euler-Bernoulli	Model Timoshenko	*OpenPulse*
1 *(out-of-plane)*	29.604	29.441	29.437
2 *(in-plane)*	31.412	31.269	31.271
3 *(out-of-plane)*	84.920	83.799	83.880
4 *(in-plane)*	89.049	86.872	86.788
5 *(out-of-plane)*	400.465	376.364	377.495
6 *(in-plane)*	410.176	385.869	387.464

It is clear that the use of the Timoshenko model is preferable over Euler-Bernoulli theory given the pipe geometry and with the use of 25% the elements used in *OpenPulse* the Timoshenko model only presents an deviation of 0.41% against *OpenPulse* in the 6th mode of vibration. Figures 4.13 and 4.14 contain the correspondent mode shapes for the out-of-plane and in-plane frequency modes, drawn with 10 elements in each segment and 5 in the elbow.

Another method to compare the two models developed with *OpenPulse* an forced vibration analysis was performed. For this, a unit force was applied to the free end of *L* shape pipe. This force acts parallel to the pipe plane and acts orthogonal to free end pipe. Figure 4.12 shows structural response for a frequency from 0 to $100\ Hz$. From this graph is possible to identify the resonant frequencies, corresponding to in-plane natural frequencies of modes 2 and 4 from table 4.5. A anti-resonant frequency is located at around $62\ Hz$. Differences between models from table 4.5 translate to this graph where it is possible to see the larger stiffness from the Euler-Bernoulli theory and how close the developed Timoshenko model is to the *OpenPulse* model.

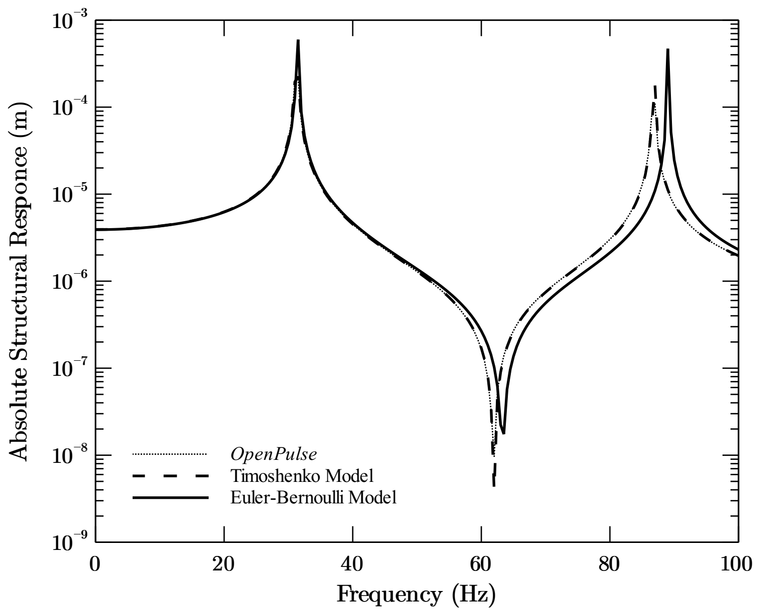
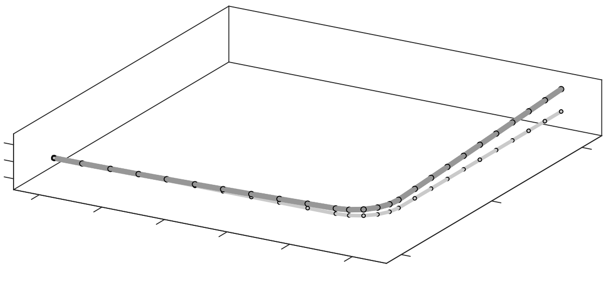
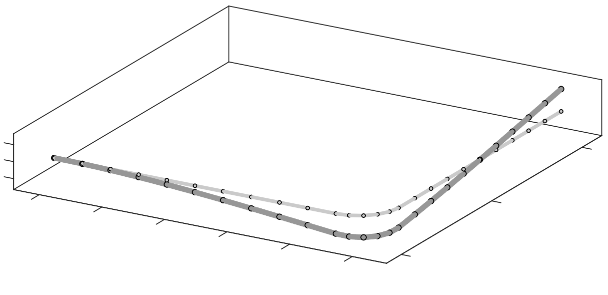


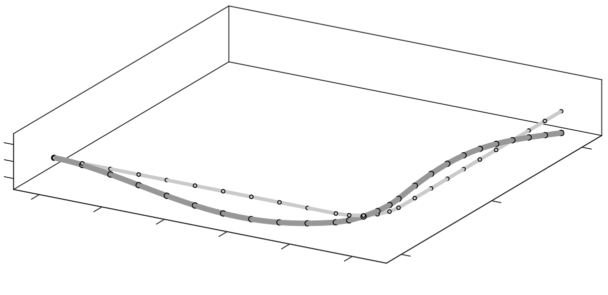
Figure 4.12: Structural response (in-plane) of free end with a transverse unit applied force



(a) Mode 1 (Out-of-plane)

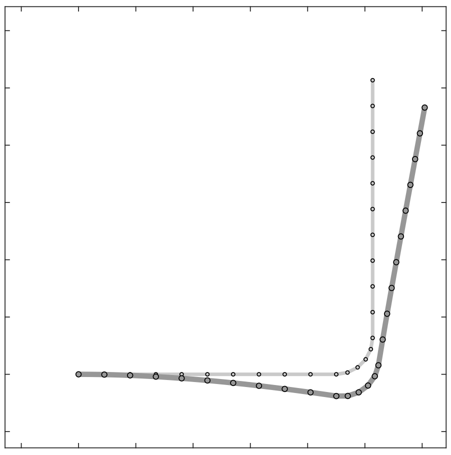


(b) Mode 3 (Out-of-plane)

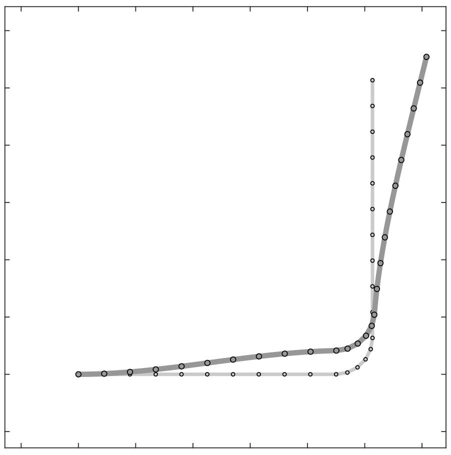


(c) Mode 5 (Out-of-plane)

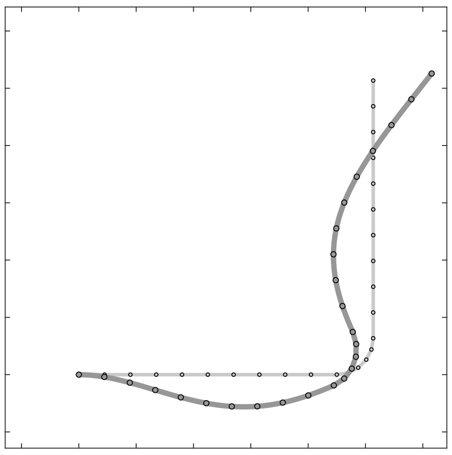
Figure 4.13: Out-of-plane modes of *L* shape clamped-free pipe system



(a) Mode 2 (In-plane)



(b) Mode 4 (In-plane)



(c) Mode 6 (In-plane)

Figure 4.14: In-plane modes of *L* shape clamped-free pipe system

4.2 Steady Internal Flow Results

Now, the results of the vibration of a pipe with fluid flow are presented. First, it is shown the results comparing it to the experiment performed by Dodds and Runyan [40]. In that experiment, the objective was to study the effect of high-velocity fluid flow on the bending vibrations and stability of a simple supported pipe. To investigate free vibrations, the pipe transporting different fluid velocities, a small disturbance to the pipe was necessary. The procedure was done using the material, fluid and geometry present in table 4.6.

Table 4.6: Pipe material, geometry and fluid properties used in [40]

Parameters	Values
Length	3.048 m
Modulus of elasticity	68.9 GPa
Poisson coefficient	0.3
External diameter	0.0254 m
Internal diameter	0.0012 m
Density of pipe	2699 kg/m^3
Density of water	1000 kg/m^3

For the fundamental mode of vibration, the experimental data for two identical pipes are shown in table 4.7. The frequency results from the internal flow model using the Euler-Bernoulli theory with 20 number of elements are also present for a given flow velocity.

Table 4.7: Natural frequencies for different models and elements for pinned-pinned pipe conveying fluid

Pipe	Flow velocity (m/s)	Experiment (rad/s)	FEM model (rad/s)	Error %
1	0	29.59	30.78	4.02
	13.10	26.0996	29.15	11.69
	23.485	24.1116	25.169	4.39
	29.722	18.8	21.084	12.15
2	6.59	29.9052	30.376	1.57
	13.973	27.2072	28.919	6.29
	21.433	26.19	26.19	0
	29.6826	21.0615	21.116	0.26

A graphical interpretation of the results is present in figure 4.15, where it can be seen that the model follows the experimental results with some precision. As predicted, the system becomes unstable which results in permanent deformation of the pipe [40]. The velocity for which the system becomes unstable, or $\omega = 0$, is called critical flow velocity.

Now, this fluid flow model is compared with a model proposed by Piet-Lahanier and Ohayon [41]. This proposed method considers a slender fluid-structure system, consisting of an elastic pipe with a compressible, viscous fluid. A finite element computer program was developed that makes possible to represent a pipe as a frequency dependent elasto-acoustic element. For this comparison, a cantilever pipe conveying fluid is used. This system is excited by a unit transverse force at the free. The response of the pipe free end in the direction of the force is presented in figure 4.16 for flow velocities of $V = 0\,m/s$

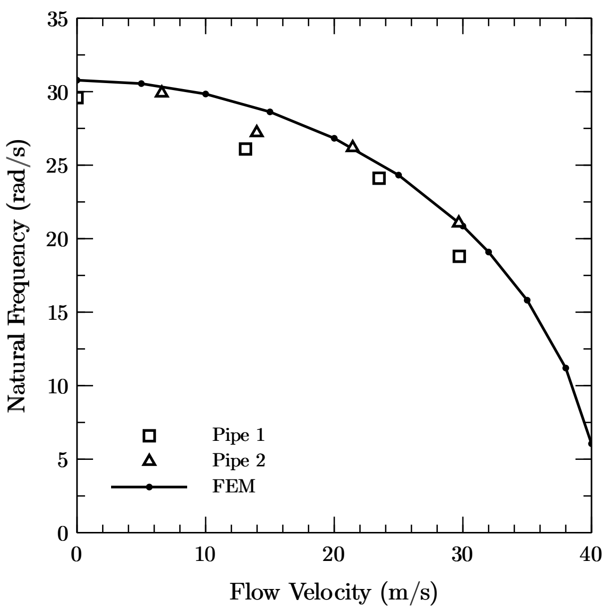


Figure 4.15: Lowest natural frequency of the simply supported straight pipe conveying fluid

and $V = 50\,m/s$.

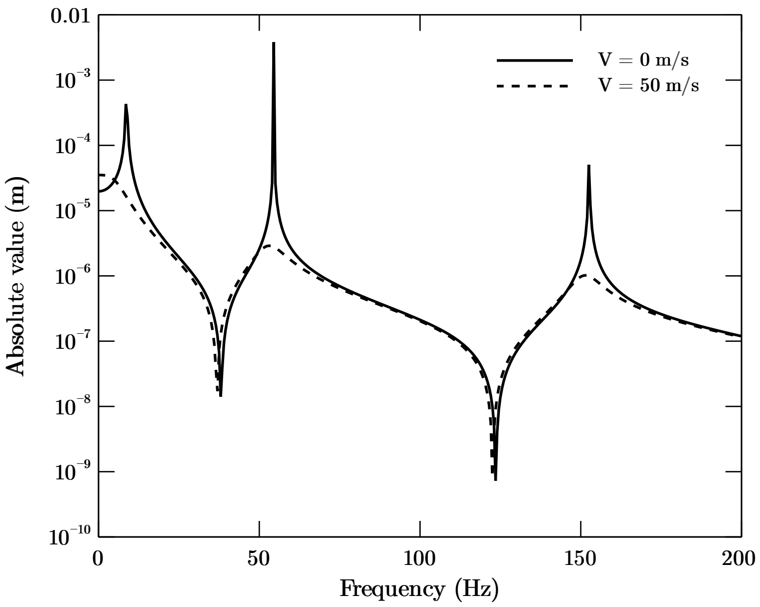


Figure 4.16: Transverse displacement of the free end for two fluid velocities

When $V = 0\,m/s$, and not considering material damping, the system presents sharp resonances corresponding to the theoretical natural frequencies. When the fluid is flowing, the response is smoother because of the damping forces originated by the fluid flow. This response is close to the response obtain by [41] model, only differing on the response for low frequencies when $V = 50\,m/s$. This difference might be caused by the authors model since the static displacement ($f = 0Hz$) only involves the stiffness matrix and force vector for the displacement calculation, and, since with the increase in flow velocity through the pipe, the overall stiffness decreases and the static deflection is increases. So the higher the flow velocity, the higher the first data point in figures 4.16 and 4.17 should be.

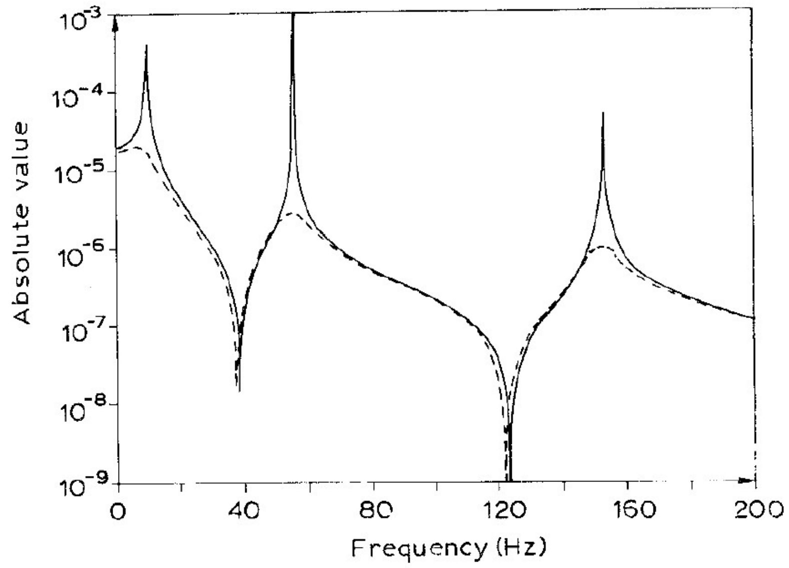


Figure 4.17: Transverse displacement response of the free end given by [41]

In this thesis, the same method for solving straight pipes is used in circular pipes but with a larger number of element to approximate the curvature. However many authors studied and developed methods for analysing the dynamics of curved pipes transporting fluid. Kisra, Paidoussis, and Van [42], Chen [43] and Zhang, Ouyang, Zhao, and Ding [44] developed different methods of dealing with this problem. In this section a comparison will be made to ensure a good structural approximation plus fluid flow approximation to this other methods. A semi-circle clamped-clamped pipe (Figure 4.18) is used because it is present in all author results. The radius and material properties are listed in table 4.8.

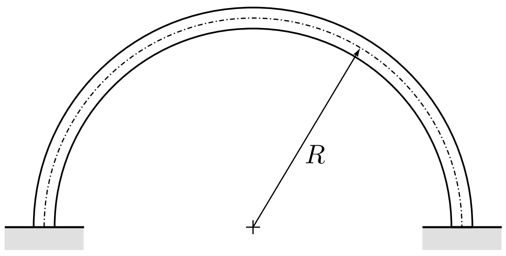


Figure 4.18: Semi-circle clamped-clamped pipe conveying fluid

For flow velocity of zero and using Euler-Bernoulli model the first 6 natural frequencies are presented in table 4.9.

Dimensionless natural frequencies and dimensional flow velocity are calculated by

$$\omega^* = \omega R^2 \sqrt{\frac{m_p + m_f}{EI}} \tag{4.1}$$

$$V^* = VR\sqrt{\frac{m_f}{EI}} \tag{4.2}$$

Table 4.8: Pipe material, geometry and fluid for the clamped-clamped semi-circle pipe

Parameters	Values
Radius	0.7 m
Modulus of elasticity	200 GPa
Poisson coefficient	0.3
External diameter	0.1 m
Internal diameter	0.094 m
Density of pipe	7900 kg/m^3
Density of water	1000 kg/m^3

Table 4.9: Dimensionless natural frequencies for different models and elements for $V^* = 0$ in the semi-circle pipe

	In-plane			Out-of-plane		
	ω_1^*	ω_2^*	ω_3^*	ω_1^*	ω_2^*	ω_3^*
Model	4.808	9.474	18.114	1.812	5.205	10.927
Ref [44]	4.4054	9.6531	17.9456	1.8464	5.2800	11.0460
Ref [42]	4.39	9.64	17.95	1.83	5.28	11.10

where R is the semi-circle radius, m_p is pipe mass per unit length and m_f is fluid mass per unit length.

If natural frequencies are calculated for an increasing flow velocity a graph of dimensionless natural frequencies versus dimensionless flow velocity can be constructed. From figure 4.19, one can see that the lowest three natural frequencies of both the in-plane and out-of-plane motions decreases as the velocity increases, and the pipe may lose stability by buckling at critical velocities, $V^* = 0$.

Comparing with figure 4.20 from [44] it can be seen that the critical flow velocities have an error associated because a completely different method has used to predict the same model.

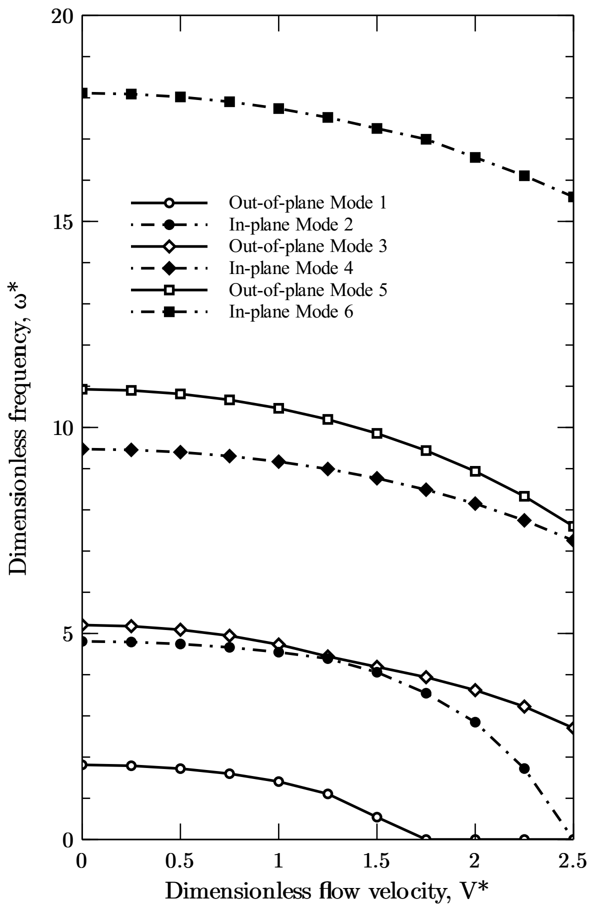


Figure 4.19: Dimensionless natural frequencies versus dimensionless fluid velocity for a semi-circle fluid conveying pipe under clamped-clamped boundary conditions

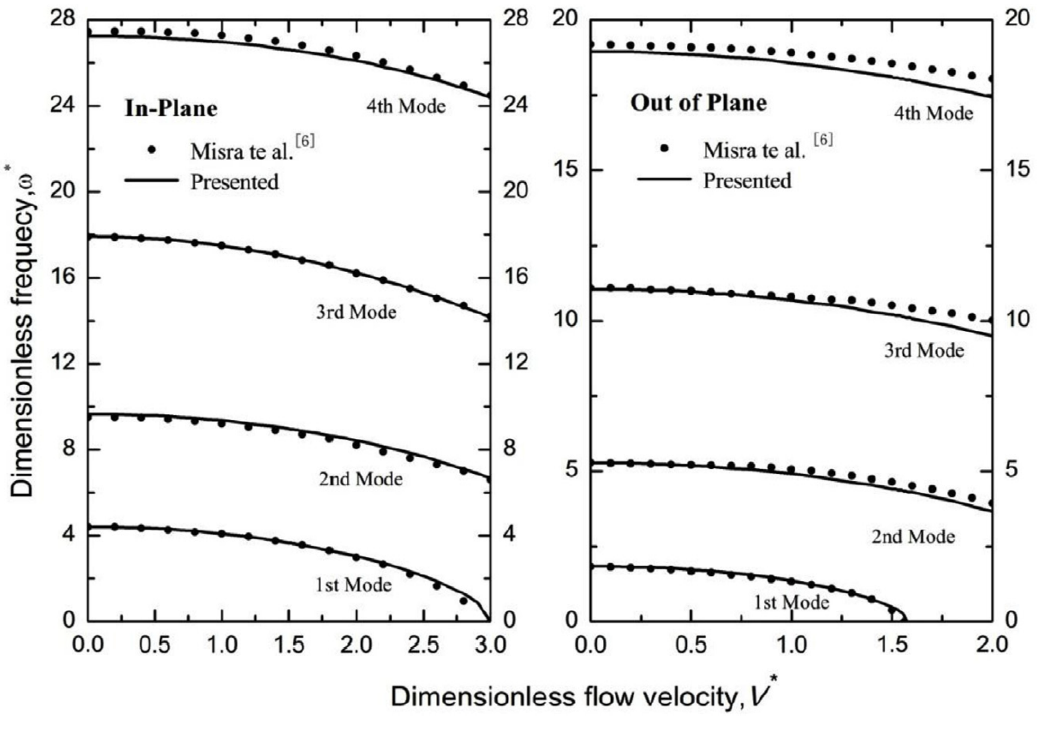


Figure 4.20: Dimensionless natural frequencies versus dimensionless fluid velocity for a semi-circle fluid conveying pipe under clamped-clamped boundary conditions from [44]

4.3 Acoustic Results

In order to verify the acoustic model, and with the lack of literature results, *OpenPulse* [7] software was used to, first verify the code developed and second to check the differences in implementation. The *L* pipe system presented in figure 4.11 from section 4.1 is used for acoustic and coupled verification. The pipe properties, material and fluid are presented in figure 4.10.

Initial check is performed with pressure boundary conditions at each end of the pipe system (Table 4.11). Figure 4.21 shows the pressure response at a node in the middle of the pipe (middle of $90°$ elbow) and it can be seen that for low frequencies, and since both pressure boundary conditions have the same numeric value, the pressure is around 5 Pa.

Table 4.10: Pipe material, geometry and fluid properties

Parameters	Values
Pipes length	0.9 m
Elbow radius	0.127 m
Modulus of elasticity	210 GPa
Poisson coefficient	0.3
External diameter	0.1 m
Internal diameter	0.09 m
Density of pipe	7800 kg/m^3
Density of fluid	1.1614 kg/m^3
Fluid speed of sound	347.21 m/s

Table 4.11: Acoustic boundary conditions for the *L* pipe

Node	Pressure
Clamped Node	5 Pa
Free Node	5 Pa

If an volume velocity value is set at the free end (table 4.12), instead of pressure boundary condition, the pressure response in the center of the elbow is presented in figure 4.22 and at the free node, pressure response is shown in figure 4.23.

Table 4.12: Acoustic boundary conditions for the *L* pipe

Node	Pressure/Volume Velocity
Clamped Node	5 Pa
Free Node	5 m^3/s

With pressure fields calculated for each of the analysis frequencies, is now possible to calculate the structural response of the pipe network with a force vector calculated based on the pressure field, introduced in section 3.3.3. Figure 4.24 presents the transverse in-plane response of the free node given the prescribed values shown in table 4.12.

All graphs shown in this section present an high degree of similarity between the acoustic model developed and *OpenPulse* software. In figure 4.24 a difference in displacements is visible and can be explained by some factors. In section 4.1 a comparison between the developed Timoshenko model and

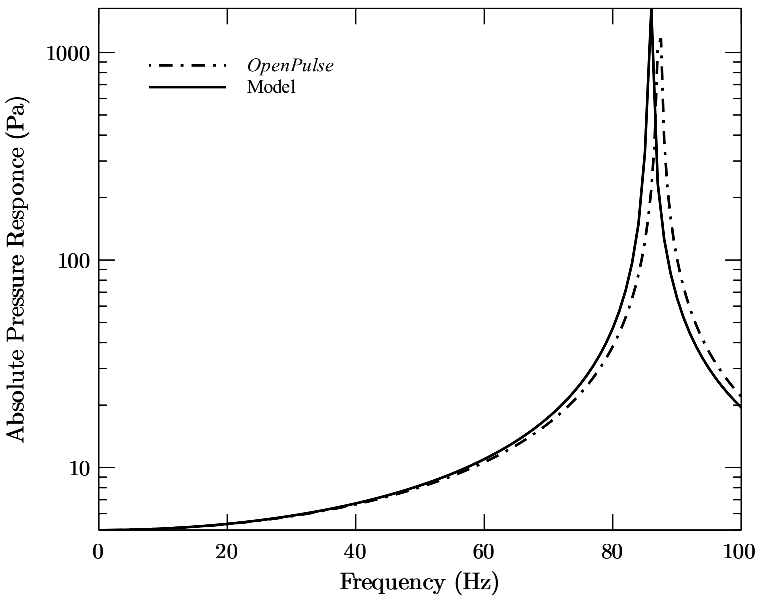


Figure 4.21: Acoustic pressure response in the center of the elbow with pressure boundary conditions at each end

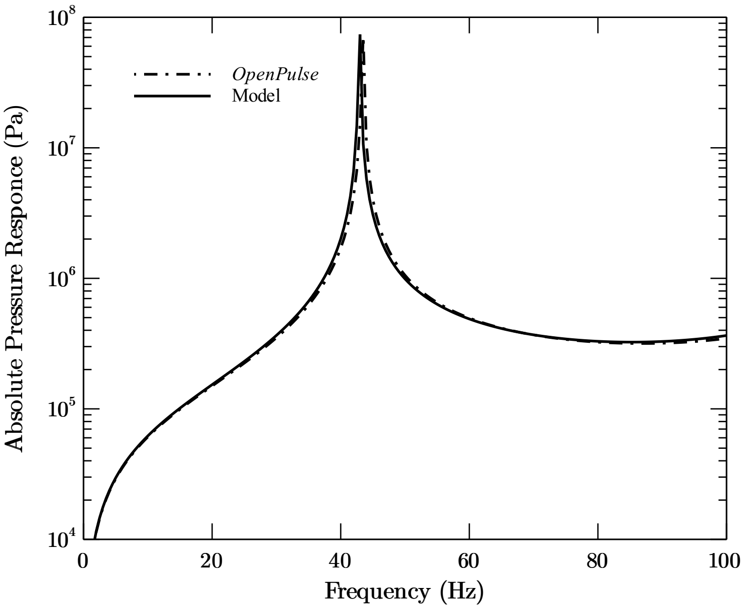


Figure 4.22: Acoustic pressure response in the center of the elbow with pressure boundary conditions and volume velocity at each end

OpenPulse was performed. From this comparison it was concluded that the structural response is very similar between them, so the structural stiffness and mass matrices can't explain this discrepancy in results. The pressure responses, figures 4.22 and 4.23, show that there is no significant pressure field difference between models so one of the explanations possibles has to do with the formation of the force vector or the formulation of fluid mass matrix. Since the lines in graph 4.24 are shifted only in the vertical direction the most probable cause is the formulation of the force vector, because in the model developed, for each frequency, the harmonic displacement is lower than the *OpenPulse* model and using the same frequency ranges and discritezation.

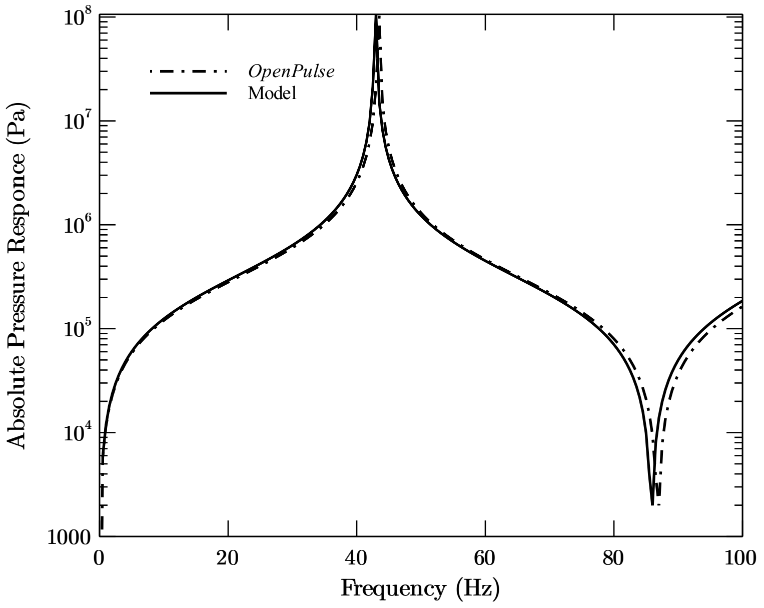


Figure 4.23: Acoustic pressure response in the free end with pressure boundary conditions and volume velocity at each end

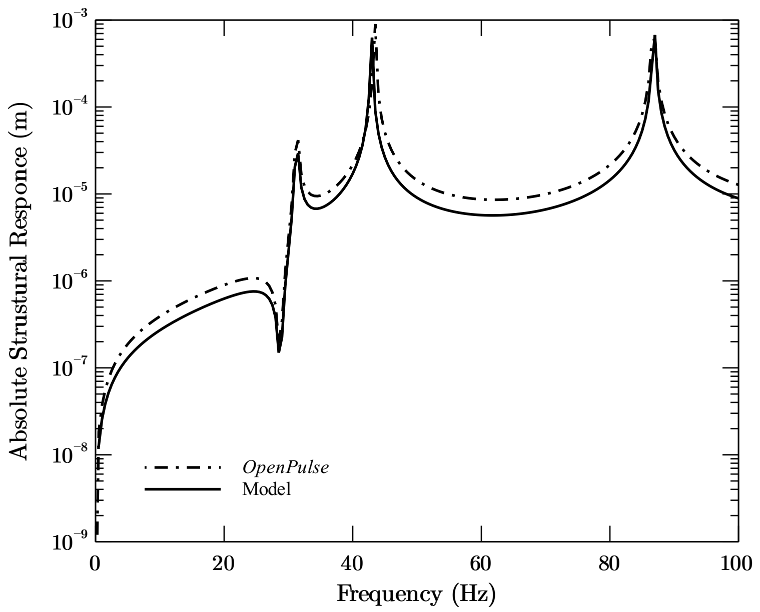


Figure 4.24: Structural response of the free end given the pressure field

4.4 *OpenPulse* Industrial Example

An industrial example is analysed in this section, kindly provided by Olavo M. Silva from the team of developers of *OpenPulse* [7] software. Figure 4.25 shows a pipe system, a structural frame and a beam supporting the large chamber on the left side of the image. The ends of the frame and supporting beam are clamped to ground, and the pipe is connected to the frame beams through elastic links.

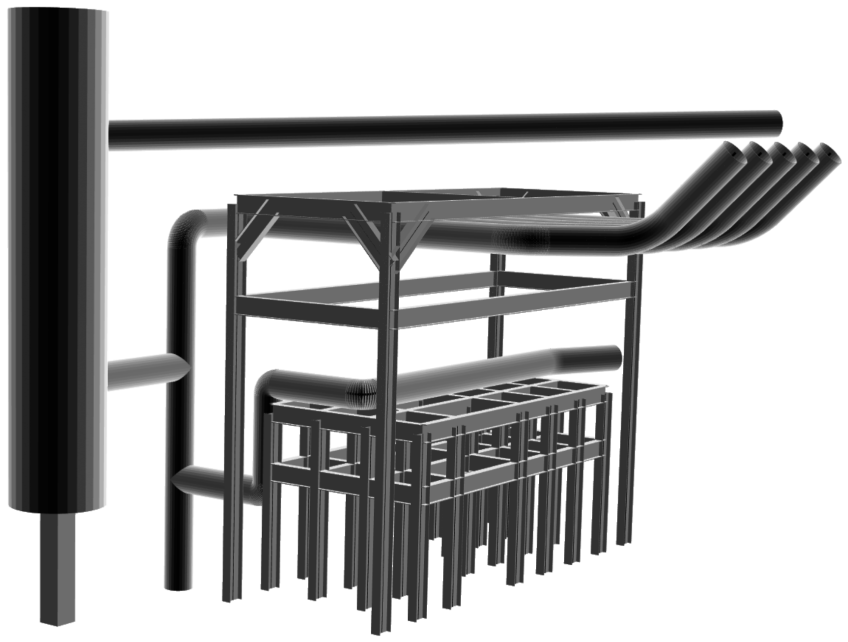


Figure 4.25: Industrial example with pipe system and structural frame

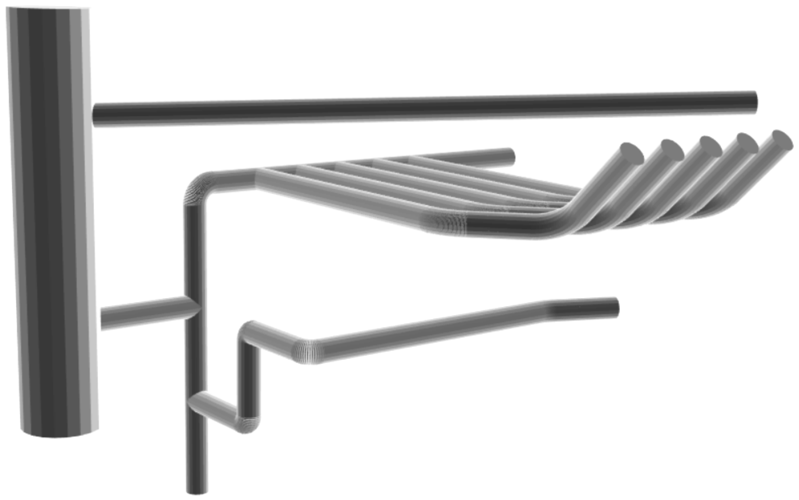


Figure 4.26: Industrial example with pipe system

Figure 4.26 only shows the pipe system. A compressor excitation is placed in pipe at the lowest node in the figure (node with lowest z coordinate). All the other pipe endings are connected to other pipe network sections. The compressor excitation profile is displayed in figure 4.27. This compressor excitation was a range from 0 Hz to 250 Hz with unitary increments. This means that both acoustic harmonic analysis and coupled analysis need to perform with that range and increment.

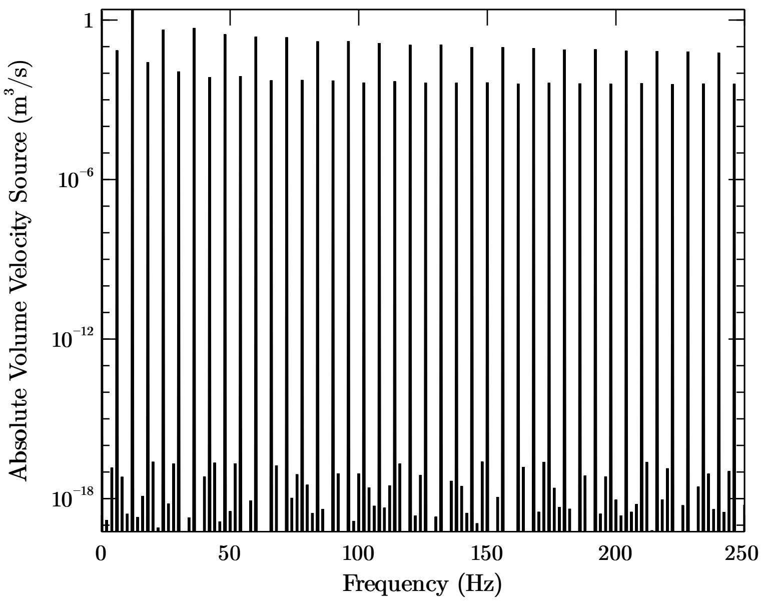


Figure 4.27: Compressor volume velocity source

The first step to replicate this network, using the method in section 3.1, is to extract, from the CAD model, all relevant pipe system points and radius of corners. Node coordinates and corner radius are presented in table 4.13 and 4.14, respectively. The node numbering is performed by the user as well as the node connectivity matrix displayed in table 4.15. Along with this connectivity matrix, each of the 34 elements and 15 corners, need to be refined. This is also done by the user, in which he chooses the number of nodes each of the elements and corners has.

Table 4.13: Pipe network input nodes coordinates

Node	x (m)	y (m)	z (m)	Node	x (m)	y (m)	z (m)
1	-1.25	2.75	0.25	19	1.75	-1.75	3.75
2	-1.25	2.75	1.75	20	1.50	2.75	3.00
3	-1.25	2.75	3.75	21	1.50	-0.25	3.00
4	-1.25	2.75	4.75	22	2.25	-1.00	3.00
5	4.25	-1.00	4.00	23	2.25	-1.75	3.75
6	0.25	2.75	-0.25	24	2.00	2.75	3.00
7	0.25	2.75	0.75	25	2.00	-0.25	3.00
8	0.25	2.75	1.75	26	2.75	-1.00	3.00
9	0.25	2.75	3.00	27	2.75	-1.75	3.75
10	4.50	2.75	3.00	28	2.50	2.75	3.00
11	0.25	1.75	0.75	29	2.50	-0.25	3.00
12	0.25	1.75	1.70	30	3.25	-1.00	3.00
13	0.25	0.75	1.70	31	3.25	-1.75	3.75
14	3.25	0.75	1.70	32	3.00	2.75	3.00
15	4.00	0.45	1.70	33	3.00	-0.25	3.00
16	1.00	2.75	3.00	34	3.75	-1.00	3.00
17	1.00	-0.25	3.00	35	3.75	-1.75	3.75
18	1.75	-1.00	3.00				

Table 4.14: Pipe network corner radius

Node	r (m)	Node	r (m)	Node	r (m)
9	0.2	17	0.5	26	0.5
11	1.7	18	0.5	29	0.5
12	1.7	21	0.5	30	0.5
13	1.7	22	0.5	33	0.5
14	1.7	25	0.5	34	0.5

Table 4.15: Pipe network element connectivity

Element	N1	N2	Element	N1	N2
1	1	2	18	28	32
2	2	3	29	32	10
3	3	4	20	16	17
4	3	5	21	17	18
5	2	8	22	18	19
6	6	7	23	20	21
7	7	8	24	21	22
8	7	11	25	22	23
9	11	12	26	24	25
10	12	13	27	25	26
11	13	14	28	26	27
12	14	15	29	28	29
13	8	9	30	29	30
14	9	16	31	30	31
15	16	20	32	32	33
16	20	24	33	33	34
17	24	28	34	34	35

Given nodal coordinates and connectivity the mesh, without refinement, is presented in MATLAB as in figure 4.28 and with every element represented with 3 elements and each corner with 5 elements, the final mesh is displayed in figure 4.29. Pipe cross-section geometry, material and fluid properties are displayed in table 4.16 and structural boundary conditions in table 4.17. Since that in the real model the chamber is supported by a clamped beam, an approximation was performed and it was set that the bottom of the chamber (Node 1) is clamped. Another approximation that was carried out is the connection between pipe tubes and structural frame (not modeled in MATLAB). It was assumed that some points along pipe tubes, that are close to the frame, are fixed. This means that there is a rigid connection between pipe and frame at some specified points.

As seen before, node 6 (Figure 4.28) was a compressor type excitation with a given volume velocity. Nodes 5, 15, 10, 19, 23, 27, 31, 35 have an imposed acoustic impedance that simulates a pipe continuity.

Performing an harmonic acoustic analysis the absolute pressure response at node 6 is given by figure 4.30. This result is very similar to the one computed with *OpenPulse* and with a mesh composed with 251 nodes compared with more than 6000 nodes used in *OpenPulse*. The same mesh is used for structural and acoustic analysis because it facilitates the transmission of data between models.

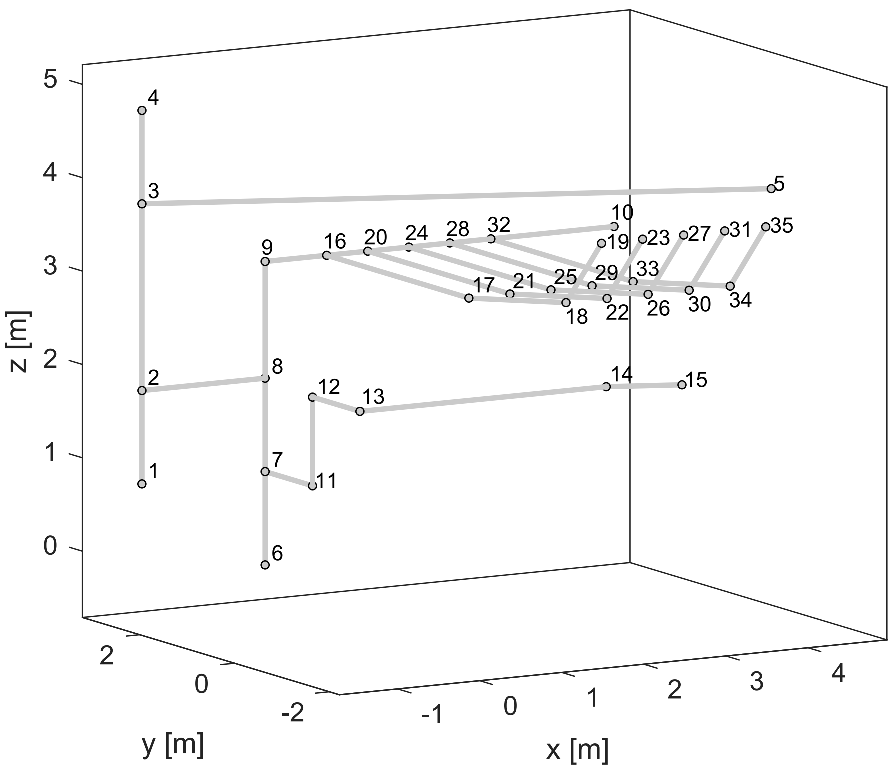


Figure 4.28: Pipe network in MATLAB with important nodes

Table 4.16: Pipe material, geometry and fluid properties

Parameters	Values
Modulus of elasticity	210 GPa
Poisson coefficient	0.3
Density of pipe	7860 kg/m^3
External diameter	0.254 m
Internal diameter	0.244 m
Density of fluid	1.1614 kg/m^3
Fluid speed of sound	347.21 m/s

Table 4.17: Structural boundary condition degrees of freedom without connection between pipe and structural frame

Node	ux	uy	uz	rx	ry	rz
1	0	0	0	0	0	0
5	-	-	0	-	-	-
6	-	-	0	-	-	-
19	-	-	0	-	-	-
23	-	-	0	-	-	-
27	-	-	0	-	-	-
31	-	-	0	-	-	-
35	-	-	0	-	-	-

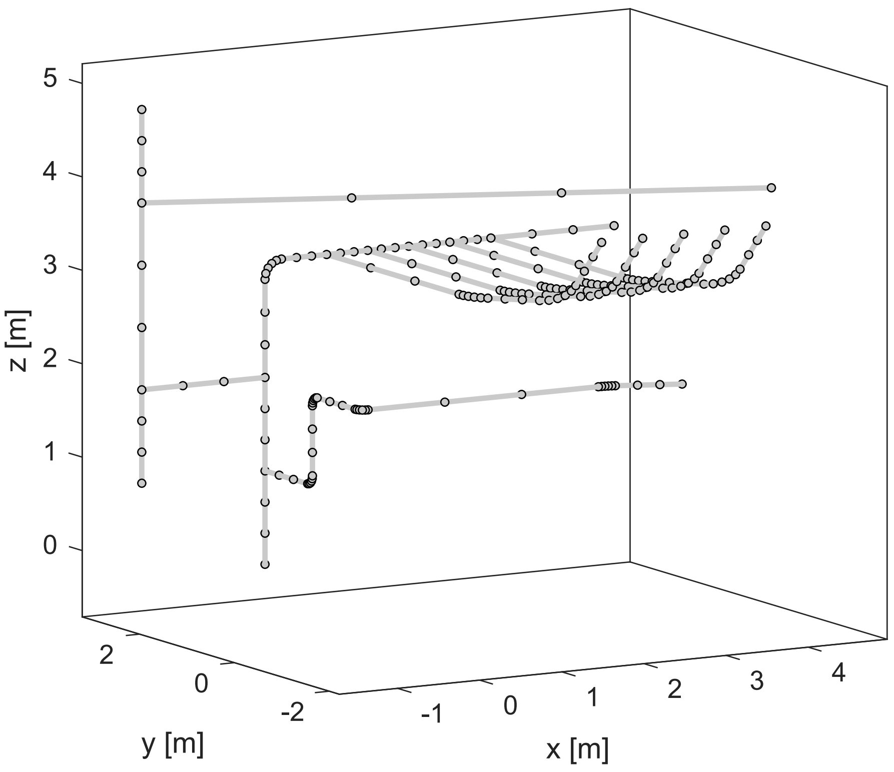


Figure 4.29: Example pipe network in MATLAB with refinement

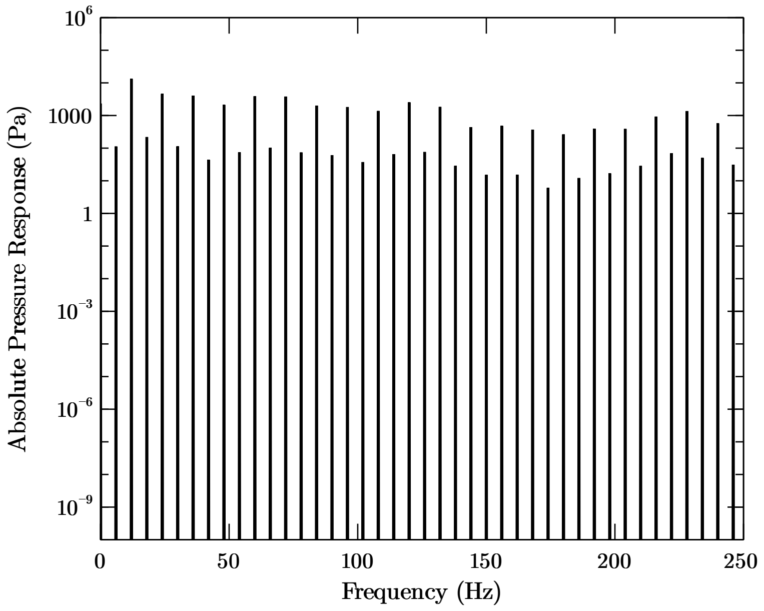


Figure 4.30: Compressor node absolute pressure response

Chapter 5

Conclusions

Considering Euler-Bernoulli and Timoshenko developed models compared to the analytical solution for simple beams with common boundary conditions it can be seen that the number of element necessary of archive a precise result increases with the increase of frequency mode number, for both models. The distinction between models stiffness is very clear an it can be seen that with the increase of mode number the difference between models increases. As the pipe thickness-to-length ratio reduces differences between models are not significant. However as ratio increases the differences in models and their application are evident. In more complex pipe designs, model validation was perform with the use of *OpenPulse* where it was concluded that the developed models results, both natural frequencies and modes of vibration, are very close to the ones in *OpenPulse*.

Different models to characterize a pipe system with steady internal flow exist. Many of those models approach to the same problem differ but all the models analysed in this thesis are very close to the one developed, for straight pipes and curved pipes. A comparison between the model and a laboratory experiment [40] proved that the model follows the experimental data with high precision.

For the acoustic results a comparison between the developed model and *OpenPulse* was performed. The pressure field was calculated for different pressure and volume velocity boundary conditions. The results show an high precision in pressure at different nodes. Given the calculated pressure field the coupled force vector has calculated and the structural response presents an error compared with *OpenPulse*. This error is only associated with the absolute response for a given frequency. This means a difference in the calculation of force vector between the models exist, but the model developed is capable of being used to calculate pressure fields and structural response of a pipeline excited by a compressor.

The pressure field calculated for an complex industrial example is a good evidence of the model precision given an compressor flow rate source. This example also provided a good opportunity to check that the mesh processing algorithm developed is capable of model very complex pipe network.

5.1 Future Work

When performing a structural analysis with steady internal flow some problems arise from the model developed. Since the user can only enter one value for the fluid velocity, in more complex pipe networks, internal velocity will change based on the amount of branches or connections, also assuming no change in cross-section area along the network. One method to solve this issue is to define the velocity for a given length section. In a branched *T* element the user must be able to define three different velocities with mass conservation in mind. This is not recommended to do automatically because depending on the *T* angle between tubes and pressures at each of the segments, the calculation of the exit velocity for two of the pipes is very time consuming only with an input velocity.

Pressure effects can also be implemented and with this an transient analysis should also be performed, because since we are dealing in steady flow, the fluid-structure interaction forces are constant. Those interactive forces can now contain the change in fluid momentum, not implemented in the acoustic coupled analysis, because of the already known velocities across system. Given this information, friction coupling mechanism can also be modeled with ease, first because the direction of that force is coincident with velocity vector and because this force can be easily computed with fluid and pipe surface properties.

In acoustics developed code, some improvements can be performed. First different pipe elements can be implemented, such as side branches, a large volume pipe and a resistance that decreases pressure across it. This can be performed by changing the global mobility matrix based on the different elements local mobility matrices. The same improvement given for the fluid-structure interaction detailed above can be implemented for this analysis. The major problem is the need to solve the volume velocity field for the entire system to be able to modify the coupled force vector to account for change in fluid momentum.

References

[1] P. Gao, T. Yu, Y. Zhang, J. Wang, and J. Zhai. Vibration analysis and control technologies of hydraulic pipeline system in aircraft: A review. *Chinese Jornal of Aeronautics*, 34(4):83–114, 2021.

[2] J. C. Wachel, S. J. Morton, and K. E. Atkins. Piping vibration analysis. *Turbomachinery and Pump Symposia*, pages 119–134, 1990.

[3] M. Jaouhari, F. Self, and Y. Liu. *Differentiating between Acoustic and Flow Induced Vibrations*. Bechtel Virtual Technology, November 2018.

[4] A. S. Tijsseling. Fluid-structure interaction in liquid-filled piping systems: A review. *Jornal of Fluids and Structures*, 10:109–146, 1996.

[5] C. S. W. Lavooij and A. S. Tijsseling. Fluid-structure interaction in liquid-filled piping systems. *Jornal of Fluids and Structures*, 5:573–595, 1991.

[6] A. Craggs. The application of the transfer matrix and matrix condensation method with finite elements to duct acoustics. *Journal of Sound and Vibration*, 132(2):394–402, 1989.

[7] O. M. Silva, D. M. Tuozzo, J. G. Vargas, L. V. Kulakauskas, A. F. Fernandes, J. L. Souza, A. P. Rocha, A. Lenzi, R. Timbo, C. O. Mendonca, and A. T. Brandao. Numerical modelling of low-frequency acoustically induced vibration in gas pipeline systems. Technical report, Federal University of Santa Catarina, Multidisciplinary Optimization Group, MOPT/LVA, Campus Trindade, Florianópolis, Brazil. Software Open Source GITHUB: https://github.com/open-pulse/OpenPulse#readme.

[8] S. S. Rao. *The Finite Element Method in Engineering*. Butterworth-Heinemann, 6^{th} edition, 2018. ISBN:978-0-12-811768-2.

[9] L. Andersen and S. R. K. Nielsen. Elastic beams in three dimensions. Aalborg University - DCE Lacture Notes No. 23, August 2008.

[10] E. Oñate. *Structural Analysis with the Finite Element Method. Linear Statics*, volume 2. CIMNE, 1^{st} edition, 2013. ISBN:978-1-4020-8742-4.

[11] C. E. Augarde. Generation of shape functions for straight beam elements. *Computers and Structures*, 68:555–560, 1998.

[12] J. S. Przemieniecki. *Theory of Matrix Structural Analysis*. Dover Publications, 1968. ISBN-13:9780486649481.

[13] H. P. Gavin. Structural element stiffness, mass, and damping matrices. Department of Civil and Environmental Engineering, Duke University, 09 2020.

[14] A. Bazoune and Y. A. Khulief. Shape functions of three-dimensional timoshenko beam element. *Journal of Sound and Vibration*, 259(2):473–480, 2003.

[15] G. R. Cowper. The shear coefficient in timoshenko's beam theory. *Journal of Applied Mechanics*, 33(2):335–340, 06 1966. doi:10.1080/10618560701678647.

[16] T. J. R. Hughes. *The finite element method: Linear Static and Dynamic Finite Element Analysis*. Prentice-Hall, 1987. ISBN:0-13-317025-X.

[17] L. Meirovitch. *Computer Methods in Structural Dynamics*. Sijthoff Noordhoff International Publishers, 1^{st} edition, 1980. ISBN:90-286-0580-0.

[18] R. D. Blevins. *Flow-Induced Vibration*. Krieger Publishing Company, 2^{nd} edition, 2001. ISBN:1-57524-183-8.

[19] M. Dangal and S. K. Ghimire. Modeling and analysis of flow induced vibration in pipes using finite element approach. 11 2020.

[20] I. Grant. Flow induced vibrations in pipes, a finite element approach. Master's thesis, Cleveland State University, 05 2010.

[21] J. N. Durrani. Dynamics of pipelines with finite element model. Master's thesis, University of Calgary, August 2001.

[22] C.-L. Chu and Y.-H. Lin. Finite element analysis of fluid-conveying timoshenko pipes. *Shock and Vibration*, 2(3):247–255, January 1995.

[23] F. M. White. *Fluid Mechanics*. McGraw-Hill Education, 8^{th} edition, 2016. ISBN:978-9-38-596549-4.

[24] M. Abom. *An Introduction of Flow Acoustics*. KTH Royal Institute of Technology, 2010. ISSN:1651-7660.

[25] S. W. Rienstra and A. Hirschberg. An introduction to acoustics. Eindhoven University of Technology, February 2021.

[26] S. Kaneko, T. Nakamura, and F. Inada. *Flow-Induced Vibrations*. Elsevier, 2^{nd} edition, 2014. ISBN:978-0-08-098347-9.

[27] T. C. Lin and G. W. Morgan. Wave propagation through fluid contained in a cylindrical, elastic shell. *The jornal of the acoustical society of america*, 28(6):1165–1176, November 1956.

[28] A. C. Ugural and S. K. Fenster. *Advanced Strength and Applied Elasticity*. Prentice Hall, 4^{th} edition, 1995. ISBN:0-13-047392-8.

[29] A. R. Boresi, R. J. Schimdt, and O. M. Sidebottom. *Advanced Mechanics of Materials*. John Wiley & Sons, 5^{th} edition, 1993. ISBN:0-471-55157-0.

[30] O. Rodrigues. Des lois géométriques qui régissent les déplacements d'un système solide dans l'espace, et de la variation des coordonnées provenant de ces déplacements considérés indépendants des causes qui peuvent les produire. *Journal de Mathématiques Pures et Appliquées*, 5:380–440, 1840.

[31] K. K. Liang. Efficient conversion from rotating matrix to rotation axis and angle by extending rodrigues's formula, October 2018.

[32] G. R. Liu and S. S. Quek. *The Finite Element Method: A Practical Course*. Butterworth-Heinemann, 1^{st} edition, 2003. ISBN:0-7506-5866-5.

[33] J. L. Meek. *Computer Methods in Structural Analysis*. E & FN SPON, 1^{st} edition, 1991. ISBN:0-419-15440-X.

[34] D. L. Logan. *A First Course in the Finite Element Method*. Cencage Learning, 6^{th} edition, 2016. ISBN-13:978-1-305-63734-4.

[35] A. J. M. Ferreira. *MATLAB Codes for Finite Element Analysis - Solids and Structures*, volume 157 of *Solid mechanics and its applications*. Springer, 2009. ISBN:978-1-4020-9199-5.

[36] M. Tanaka and K. Fujita. Vibration of piping system by pulsation of containing fluid. (in Japanese).

[37] T. Belytschko, M. Karabin, and J. I. Lin. Fluid-structure interaction in waterhammer response of flexible piping. *Journal of Pressure Vessel Technology*, 180:249–255, 1986.

[38] S. S. Rao. *Mechanical Vibrations*. Pearson, 6^{th} edition, 2017. ISBN-10:1-292-17860-4.

[39] L. Lee and W. W. Schultz. Eigenvalue analysis of timoshenko beams and axisymmetric mindlin plates by the pseudospectral method. *Journal of Sound and Vibration*, 269, 2004. doi:10.1016/S0022-460X(03)00047-6.

[40] H. L. Dodds and H. L. Runyan. Effect of high-velocity fluid flow on the bending vibrations and static divergence of a simply supported pipe. *NASA TN D-2870*, June 1965.

[41] N. Piet-Lahanier and R. Ohayon. Finite element analysis of a slender fluid-structure system. *Journal of Fluids and Structures*, 4:631–645, 1990. doi:0889-9746/90/060063.

[42] A. K. Kisra, M. P. Paidoussis, and K. S. Van. On the dynamics of curved pipes transporting fluid. part 1: Inextensible theory. *Journal of Fluids and Structures*, 2:221–244, 1988.

[43] S. S. Chen. Out-of-plane vibration and stability of curved tubes conveying fluid. *Journal of Applied Mechanics*, 40:362–368, 1973.

[44] T. Zhang, H. Ouyang, C. Zhao, and Y. J. Ding. Vibration analysis of a complex fluid-conveying piping system with general boundary conditions using the receptance method. *International Journal of Pressure Vessels and Piping*, 166:84–93, 2018.

Appendix A

MATLAB Code

In this appendix, some relevant parts of the code written is presented. All the mesh, material, fluid, structural and acoustic boundary conditions are imported into the program via text documents written by the user and each one of the files must be inserted individually. The variables are stored in structures and at the beginning of each functions all the required information is taken from this structures and at the end all the relevant data is stored in the same or different structures.

A.1 Mesh Processing

This function is responsible to determine all the mesh parameters: number of nodes, number of elements, nodes coordinates and element connectivity. First, using the initial mesh design provided by the user, it calculates the total number of nodes and elements and constructs the connectivity matrix based on the refinement number of each element. Finally, the nodes coordinates are calculated, starting from the straight line segments and then the corner nodes if they exist.

Listing A.1: *process_mesh* function MATLAB code

```
function [Mesh] = process_mesh(PreMesh, Mesh)

pre_number_nodes = PreMesh.number_nodes;
pre_number_elements = PreMesh.number_elements;
number_corners = PreMesh.number_corners;
pre_node_coordinates = PreMesh.node_coordinates;
pre_element_nodes = PreMesh.element_nodes;
element_segments = PreMesh.element_segments;
corner_nodes = PreMesh.corner_nodes;
corner_radius = PreMesh.corner_radius;
corner_segments = PreMesh.corner_segments;

element_nodes = pre_element_nodes;
number_nodes = pre_number_nodes;
number_elements = pre_number_elements;

% ---------------------------------------------------------------------
% Form elements
current_node = pre_number_nodes + 1;

corner_s_e = zeros(2, number_corners);
arc_s_e = zeros(2, number_corners);
```

```

for num_co = 1 : number_corners
    cor_node = corner_nodes(num_co);
    cor_seg = corner_segments(num_co);

    c_element_nodes = zeros(2, cor_seg);
    arc_s_e(1, num_co) = cor_node;

    pos = 1;
    for num_el = 1 : number_elements
        node_a = element_nodes(1, num_el);
        node_b = element_nodes(2, num_el);
        if (node_a == cor_node)
            corner_s_e(pos, num_co) = node_b;
            pos = pos + 1;
        elseif (node_b == cor_node)
            corner_s_e(pos, num_co) = node_a;
            pos = pos + 1;
        end
        if (pos == 3)
            element_nodes(:, num_el) = [];
            break;
        end
    end
    for co_sg = 1 : cor_seg
        if (co_sg == 1)
            c_element_nodes(:, co_sg) = [cor_node; current_node];
        else
            c_element_nodes(:, co_sg) = [current_node; current_node+1];
            current_node = current_node + 1;
        end
    end
    arc_s_e(2, num_co) = current_node;
    % end element
    e_element = [current_node; corner_s_e(2, num_co)];
    c_e_nodes = [element_nodes, c_element_nodes, e_element];
    element_nodes = c_e_nodes;

    number_elements = number_elements + cor_seg;
    number_nodes = number_nodes + cor_seg;
    current_node = current_node + 1;
end

line_segments = zeros(2, pre_number_elements);

for p_num_el = 1 : pre_number_elements
    line_seg = element_segments(p_num_el);
    node_a = pre_element_nodes(1, p_num_el);
    node_b = pre_element_nodes(2, p_num_el);
    line_segments(:, p_num_el) = [node_a; node_b];

    number_c_e = 0;
    if (line_seg > 1)
        corner_sum = zeros(1, 2);
        for num_co = 1 : number_corners
            if (node_a == corner_nodes(num_co) || node_b == corner_nodes(num_co))
                number_c_e = number_c_e + 1;
                corner_sum(number_c_e) = corner_nodes(num_co);
            end
        end
        if (number_c_e == 1)
            ind = find(corner_nodes == corner_sum(1));
            if (node_a == corner_sum(1))
                position = find(corner_s_e(:, ind) == node_b);
                pos = position;
                node_a = arc_s_e(pos, ind);
            elseif (node_b == corner_sum(1))
```

```
                position = find(corner_s_e(:, ind) == node_a);
                pos = position;
                node_b = arc_s_e(pos, ind);
            end
        elseif (number_c_e == 2)
            ind2 = [find(corner_nodes == corner_sum(1)); find(corner_nodes == corner_sum(2))];
            opti = [...
                arc_s_e(1, ind2(1)), arc_s_e(1, ind2(2)); ...
                arc_s_e(1, ind2(1)), arc_s_e(2, ind2(2)); ...
                arc_s_e(2, ind2(1)), arc_s_e(1, ind2(2)); ...
                arc_s_e(2, ind2(1)), arc_s_e(2, ind2(2))];
            for opn = 1 : 4
                for num_el = 1 : number_elements
                    if (opti(opn, 1) == element_nodes(1, num_el) && ...
                        opti(opn, 2) == element_nodes(2, num_el))
                        node_a = opti(opn, 1);
                        node_b = opti(opn, 2);
                    elseif (opti(opn, 1) == element_nodes(2, num_el) && ...
                        opti(opn, 2) == element_nodes(1, num_el))
                        node_a = opti(opn, 2);
                        node_b = opti(opn, 1);
                    end
                end
            end
        end
        line_segments(:, p_num_el) = [node_a; node_b];
        for num_el = 1 : number_elements
            new_a = element_nodes(1, num_el);
            new_b = element_nodes(2, num_el);
            if (node_a == new_a) && (node_b == new_b)
                element_nodes(:, num_el) = [];
                break;
            end
        end
        line_element_nodes = zeros(2, line_seg);
        for l_seg = 1 : line_seg
            if (l_seg == 1)
                line_element_nodes(:, l_seg) = [node_a; current_node];
            elseif (l_seg == line_seg)
                line_element_nodes(:, l_seg) = [current_node; node_b];
            else
                line_element_nodes(:, l_seg) = [current_node; current_node+1];
                current_node = current_node + 1;
            end
        end
        current_e_n = [element_nodes, line_element_nodes];
        element_nodes = current_e_n;

        number_elements = number_elements + line_seg - 1;
        number_nodes = number_nodes + line_seg - 1;
        current_node = current_node + 1;
    end
end

% -------------------------------------------------------------------
% Form coordinates

node_coordinates = zeros(3, number_nodes);
node_coordinates(:, 1:pre_number_nodes) = pre_node_coordinates;

current_n_c = pre_node_coordinates;
current_node = pre_number_nodes + 1;

% Corners coordinates
for num_co = 1 : number_corners
    node = corner_nodes(num_co);
    radius = corner_radius(num_co);
    segments = corner_segments(num_co);

    node_a = corner_s_e(1, num_co);
    node_b = corner_s_e(2, num_co);

    v_a = current_n_c(:, node_a) - current_n_c(:, node);
    v_b = current_n_c(:, node_b) - current_n_c(:, node);
    v_a = v_a / norm(v_a);
    v_b = v_b / norm(v_b);

    theta_ab = acos(dot(v_a, v_b));
    d_e = radius / tan(theta_ab / 2);
    d_i = radius / sin(theta_ab / 2);

    v_c = v_a + v_b;
    v_c = v_c / norm(v_c);

    center_p = current_n_c(:, node) + v_c * d_i;
    start_p = current_n_c(:, node) + v_a * d_e;
    end_p = current_n_c(:, node) + v_b * d_e;

    v_ca = start_p - center_p;
    v_cb = end_p - center_p;
    v_ca = v_ca / norm(v_ca);
    v_cb = v_cb / norm(v_cb);

    v_n = cross(v_ca, v_cb);
    v_n = v_n / norm(v_n);

    theta_cacb = pi - theta_ab;
    dtheta_cacb = theta_cacb / segments;
    t = 0;

    node_first = center_p + radius * v_ca;
    node_coordinates(:, node) = node_first;

    for s = 1 : segments
        t = t + dtheta_cacb;

        % Matrix notation
        I = [1, 0, 0; 0, 1, 0; 0, 0, 1];
        K = [0, -v_n(3), v_n(2); v_n(3), 0, -v_n(1); -v_n(2), v_n(1), 0];
        R = I + sin(t) * K + (1 - cos(t)) * K^2;
        v_ca_rot = R * v_ca;

        node_new = center_p + radius * v_ca_rot;
        node_coordinates(:, current_node) = node_new;

        current_node = current_node + 1;
    end
    current_n_c = node_coordinates(:, 1:current_node-1);
end

% Line coordinates
for p_num_el = 1 : pre_number_elements
    segments = element_segments(p_num_el);
    if (segments > 1)
        node_a = line_segments(1, p_num_el);
        node_b = line_segments(2, p_num_el);

        v_a = node_coordinates(:, node_b) - node_coordinates(:, node_a);
        L = norm(v_a);
        v_a = v_a / L;

        for li_seg = 1 : segments-1
            node_new = node_coordinates(:, node_a) + v_a * (li_seg / segments * L);
            node_coordinates(:, current_node) = node_new;
            current_node = current_node + 1;
        end
    end
end

Mesh.number_nodes = number_nodes;
Mesh.number_elements = number_elements;
Mesh.node_coordinates = node_coordinates;
Mesh.element_nodes = element_nodes;
```

A.2 Structural Boundary Conditions

The processing of the structural displacement boundary conditions is done by this function. It identifies the number and value of the degrees of freedom provided by the user and determines the free degrees of freedom.

Listing A.2: *process_struct_dof* function MATLAB code

```
function [Structural] = process_struct_dof(Mesh, PreStructural, Structural)

number_nodes = Mesh.number_nodes;
number_presc_nodes = PreStructural.number_presc_nodes;
node_number = PreStructural.node_number;
number_dofs_node = PreStructural.number_dofs_node;
dofs_values = PreStructural.dofs_values;

number_dofs = 6 * number_nodes;
dofs = 1 : number_dofs;
number_prescribed = sum(number_dofs_node);
number_free = number_dofs - number_prescribed;

prescribed_dofs = zeros(1, number_prescribed);
prescribed_values = zeros(1, number_prescribed);
p_n = 1;
for num_pn = 1 : number_presc_nodes
    node = node_number(num_pn);
    for num_dn = 1 : number_dofs_node(num_pn)
        dof_node = dofs_values(1, num_dn, num_pn);
        dof_value = dofs_values(2, num_dn, num_pn);
        prescribed_dofs(p_n) = 6 * node - 6 + dof_node;
        prescribed_values(p_n) = dof_value;
        p_n = p_n + 1;
    end
end
free_dofs = 1 : number_dofs;
free_dofs(prescribed_dofs) = [];

Structural.number_dofs = number_dofs;
Structural.number_prescribed = number_prescribed;
Structural.number_free = number_free;
Structural.dofs = dofs;
Structural.prescribed_dofs = prescribed_dofs;
Structural.free_dofs = free_dofs;
Structural.prescribed_values = prescribed_values;
```

A.3 Structural Matrices

The three-dimensional Euler-Bernoulli and Timoshenko beam theory matrices are constructed in this section.

Listing A.3: *stiffness_mass_matrix_eb* function MATLAB code

```
function [Structural] = stiffness_mass_matrix_eb(Mesh, Material, Section, Structural)

number_elements = Mesh.number_elements;
node_coordinates = Mesh.node_coordinates;
element_nodes = Mesh.element_nodes;
E = Material.young_modulus;
G = Material.shear_modulus;
Nu = Material.poisson_coefficient;
Rho = Material.density;
A = Section.area;
Iy = Section.second_moment;
Iz = Section.second_moment;
J = Section.polar_moment;
number_dofs = Structural.number_dofs;
Alpha = Material.mass_coefficient;
Beta = Material.stiffness_coefficient;

KK = zeros(number_dofs, number_dofs);
MM = zeros(number_dofs, number_dofs);
CC = zeros(number_dofs, number_dofs);

xx = node_coordinates(1, :);
yy = node_coordinates(2, :);
zz = node_coordinates(3, :);

for num_el = 1 : number_elements
    node1 = element_nodes(1, num_el);
    node2 = element_nodes(2, num_el);
    element_dof = [...
        6*node1-5, 6*node1-4, 6*node1-3, 6*node1-2, 6*node1-1, 6*node1, ...
        6*node2-5, 6*node2-4, 6*node2-3, 6*node2-2, 6*node2-1, 6*node2];
    dx = xx(node2) - xx(node1);
    dy = yy(node2) - yy(node1);
    dz = zz(node2) - zz(node1);
    L = (dx^2 + dy^2 + dz^2)^0.5;

    if (dx == 0 && dy == 0)
        if (dz > 0)
            Lambda = [0, 0, 1; 0, 1, 0; -1, 0, 0];
        else
            Lambda = [0, 0, -1; 0, 1, 0; 1, 0, 0];
        end
    else
        CXx = dx / L;
        CYx = dy / L;
        CZx = dz / L;
        D = sqrt(CXx^2 + CYx^2);
        CXy = -CYx / D;
        CYy = CXx / D;
        CZy = 0;
        CXz = -CXx * CZx / D;
        CYz = -CYx * CZx / D;
        CZz = D;
        Lambda = [CXx, CYx, CZx; CXy, CYy, CZy; CXz, CYz, CZz];
    end
    T = [...
        Lambda, zeros(3, 9); zeros(3, 3), Lambda, zeros(3, 6); ...
        zeros(3, 6), Lambda, zeros(3, 3); zeros(3, 9), Lambda];

    % Local stiffness matrix
    k1 = E * A / L;
    k2 = 12 * E * Iz / L^3;
    k3 = 6 * E * Iz / L^2;
    k4 = 4 * E * Iz / L;
    k5 = 2 * E * Iz / L;
    k6 = 12 * E * Iy / L^3;
    k7 = 6 * E * Iy / L^2;
    k8 = 4 * E * Iy / L;
    k9 = 2 * E * Iy / L;
    k10 = G * J / L;
    a1 = [k1, 0, 0; 0, k2, 0; 0, 0, k6] ;
    b1 = [0, 0, 0; 0, 0, k3; 0, -k7, 0];
    c1 = [k10, 0, 0; 0, k8, 0; 0, 0, k4];
    d1 = [-k10, 0, 0; 0, k9, 0; 0, 0, k5];
    Kl = [...
        a1, b1, -a1, b1; ...
        b1', c1, (-b1)', d1; ...
        (-a1)', -b1, a1, -b1; ...
        b1', d1', (-b1)', c1];

    % Local mass matrix
    a2 = [140, 0, 0; 0, 156, 0; 0, 0, 156];
    b2 = [0, 0, 0; 0, 0, 22*L; 0, -22*L, 0];
    c2 = [70, 0, 0; 0, 54, 0; 0, 0, 54];
    d2 = [0, 0, 0; 0, 0, -13*L; 0, 13*L, 0];
    e2 = [140*J/A, 0, 0; 0, 4*L^2, 0; 0, 0, 4*L^2];
    f2 = [70*J/A, 0, 0; 0, -3*L^2, 0; 0, 0, -3*L^2];
    Ml = Rho * A * L / 420 * [...
        a2, b2, c2, d2; ...
        b2', e2, d2, f2; ...
        c2', d2', a2, -b2; ...
        d2', f2', (-b2)', e2];

    % Local to global transformation
    K0 = T' * Kl * T;
    M0 = T' * Ml * T;
    % Assembly
    KK(element_dof, element_dof) = KK(element_dof, element_dof) + K0;
    MM(element_dof, element_dof) = MM(element_dof, element_dof) + M0;
end

CC = Alpha * MM + Beta * KK;

Structural.stiffness_matrix = KK;
Structural.mass_matrix = MM;
Structural.damping_matrix = CC;
```

Listing A.4: *stiffness_mass_matrix_t* function MATLAB code

```
function [Structural] = stiffness_mass_matrix_t1(Mesh, Material, Section, Structural)

number_elements = Mesh.number_elements;
node_coordinates = Mesh.node_coordinates;
element_nodes = Mesh.element_nodes;
E = Material.young_modulus;
G = Material.shear_modulus;
Nu = Material.poisson_coefficient;
Rho = Material.density;
A = Section.area;
Iy = Section.second_moment;
Iz = Section.second_moment;
J = Section.polar_moment;
k = Section.shear_coefficient;
number_dofs = Structural.number_dofs;
Alpha = Material.mass_coefficient;
```

```matlab
Beta = Material.stiffness_coefficient;

KK = zeros(number_dofs);
MM = zeros(number_dofs);
CC = zeros(number_dofs);

xx = node_coordinates(1, :);
yy = node_coordinates(2, :);
zz = node_coordinates(3, :);

for num_el = 1 : number_elements
    node1 = element_nodes(1, num_el);
    node2 = element_nodes(2, num_el);
    element_dof = [...
        6*node1-5, 6*node1-4, 6*node1-3, 6*node1-2, 6*node1-1, 6*node1, ...
        6*node2-5, 6*node2-4, 6*node2-3, 6*node2-2, 6*node2-1, 6*node2];
    dx = xx(node2) - xx(node1);
    dy = yy(node2) - yy(node1);
    dz = zz(node2) - zz(node1);
    L = (dx^2 + dy^2 + dz^2)^0.5;

    if (dx == 0 && dy == 0)
        if (dz > 0)
            Lambda = [0, 0, 1; 0, 1, 0; -1, 0, 0];
        else
            Lambda = [0, 0, -1; 0, 1, 0; 1, 0, 0];
        end
    else
        CXx = dx / L;
        CYx = dy / L;
        CZx = dz / L;
        D = sqrt(CXx^2 + CYx^2);
        CXy = -CYx / D;
        CYy = CXx / D;
        CZy = 0;
        CXz = -CXx * CZx / D;
        CYz = -CYx * CZx / D;
        CZz = D;
        Lambda = [CXx, CYx, CZx; CXy, CYy, CZy; CXz, CYz, CZz];
    end
    T = [...
        Lambda, zeros(3, 9); zeros(3, 3), Lambda, zeros(3, 6); ...
        zeros(3, 6), Lambda, zeros(3, 3); zeros(3, 9), Lambda];

    % Local stiffness matrix
    Psi_y = 12 * E * Iy / (G * k * A * L^2);
    Psi_z = 12 * E * Iz / (G * k * A * L^2);
    k1 = E * A / L;
    k2 = 12 * E * Iz / ((1 + Psi_y) * L^3);
    k3 = 6 * E * Iz / ((1 + Psi_y) * L^2);
    k4 = (4 + Psi_y) * E * Iz / ((1 + Psi_y) * L);
    k5 = (2 - Psi_y) * E * Iz / ((1 + Psi_y) * L);
    k6 = 12 * E * Iy / ((1 + Psi_z) * L^3);
    k7 = 6 * E * Iy / ((1 + Psi_z) * L^2);
    k8 = (4 + Psi_z) * E * Iy / ((1 + Psi_z) * L);
    k9 = (2 - Psi_z) * E * Iy / ((1 + Psi_z) * L);
    k10 = G * J / L;
    a1 = [k1, 0, 0; 0, k2, 0; 0, 0, k6];
    b1 = [0, 0, 0; 0, 0, k3; 0, -k7, 0];
    c1 = [k10, 0, 0; 0, k8, 0; 0, 0, k4];
    d1 = [-k10, 0, 0; 0, k9, 0; 0, 0, k5];
    Kl = [...
        a1, b1, -a1, b1; ...
        b1', c1, (-b1)', d1; ...
        (-a1)', -b1, a1, -b1; ...
        b1', d1', (-b1)', c1];

    % Local mass matrix
    a2 = [140, 0, 0; 0, 156+504*Iz/A/L^2, 0; 0, 0, 156+504*Iy/A/L^2];
    b2 = [0, 0, 0; 0, 0, 22*L+42*Iz/A/L; 0, -22*L-42*Iy/A/L, 0];
    c2 = [70, 0, 0; 0, 54-504*Iz/A/L^2, 0; 0, 0, 54-504*Iy/A/L^2];
    d2 = [0, 0, 0; 0, 0, -13*L-42*Iz/A/L; 0, 13*L+42*Iy/A/L, 0];
    e2 = [140*J/A, 0, 0; 0, 4*L^2+56*Iy/A, 0; 0, 0, 4*L^2+56*Iz/A];
    f2 = [70*J/A, 0, 0; 0, -3*L^2-28*Iy/A, 0; 0, 0, -3*L^2-28*Iz/A];
    Ml = Rho * A * L / 420 * [...
        a2, b2, c2, d2; ...
        b2', e2, d2, f2; ...
        c2', d2', a2, -b2; ...
        d2', f2', (-b2)', e2];

    % Local to global transformation
    K0 = T' * Kl * T;
    M0 = T' * Ml * T;
    % Assembly
    KK(element_dof, element_dof) = KK(element_dof, element_dof) + K0;
    MM(element_dof, element_dof) = MM(element_dof, element_dof) + M0;
end

CC = Alpha * MM + Beta * KK;

Structural.stiffness_matrix = KK;
Structural.mass_matrix = MM;
Structural.damping_matrix = CC;
```

A.4 Structural Free Vibration Analysis

Given stiffness and mass structural matrices a modal or free vibration analysis can be performed with the following function only needing the free degrees of freedom of the system.

Listing A.5: *struct_modal_analysis* function MATLAB code

```
function [SolutionStructural] = struct_modal_analysis(Structural, SolutionStructural)

number_dofs = Structural.number_dofs;
number_free = Structural.number_free;
free_dofs = Structural.free_dofs;
KK = Structural.stiffness_matrix;
MM = Structural.mass_matrix;

K = KK(free_dofs, free_dofs);
M = MM(free_dofs, free_dofs);

[vec, val] = eig(K, M);

[natural_freq, ind] = sort(real(sqrt(diag(val))));

freq_modes = zeros(number_dofs, number_free);
freq_modes(free_dofs, :) = vec(:, ind);

SolutionStructural.modal.natural_frequencies = natural_freq;
SolutionStructural.modal.modes = freq_modes;
```

A.5 Structural Forced Vibration Analysis

The structural harmonic or forced vibration analysis is perform with the code below. Given the force vector and structural matrices, absolute displacement field is calculated for the frequency array provided by the user.

Listing A.6: *struct_harmonic_analysis* function MATLAB code

```matlab
function [SolutionStructural] = struct_harmonic_analysis(Structural, SolutionStructural)

number_dofs = Structural.number_dofs;
number_prescribed = Structural.number_prescribed;
number_free = Structural.number_free;
dofs = Structural.dofs;
prescribed_dofs = Structural.prescribed_dofs;
free_dofs = Structural.free_dofs;
prescribed_values = Structural.prescribed_values;
FF = Structural.force_vector;
KK = Structural.stiffness_matrix;
MM = Structural.mass_matrix;
CC = Structural.damping_matrix;
harm_freq = SolutionStructural.harmonic.frequencies;

number_freq = length(harm_freq);
harm_displ = zeros(number_dofs, number_freq);

j = sqrt(-1);

for num_fq = 1 : number_freq
    freq_hz = harm_freq(num_fq);
    freq_rad = freq_hz * 2 * pi;
    dynamic_matrix = KK - freq_rad^2 * MM + j * freq_rad * CC;
    d = dynamic_matrix(free_dofs, free_dofs) \ FF(free_dofs);
    harm_displ(free_dofs, num_fq) = (real(d).^2 + imag(d).^2).^0.5;
end

SolutionStructural.harmonic.displacements = harm_displ;
```

A.6 Acoustic Stiffness Matrix

Acoustic one-dimensional stiffness matrix, also known as mobility matrix is calculated for each frequency of analysis (Line 13).

Listing A.7: *stiffness_matrix* function MATLAB code

```matlab
1   function [Acoustic] = stiffness_matrix(Mesh, Section, Fluid, Acoustic)
2
3   number_elements = Mesh.number_elements;
4   node_coordinates = Mesh.node_coordinates;
5   element_nodes = Mesh.element_nodes;
6   A_i = Section.internal_area;
7   rho_f = Fluid.density;
8   c_o = Fluid.speed_sound_corrected;
9   number_dofs = Acoustic.number_dofs;
10  frequencies = Acoustic.frequencies;
11  number_freq = Acoustic.number_frequencies;
12
13  KKa = zeros(number_dofs, number_dofs, number_freq);
14
15  xx = node_coordinates(1, :);
```

```matlab
yy = node_coordinates(2, :);
zz = node_coordinates(3, :);

for num_el = 1 : number_elements
    node1 = element_nodes(1, num_el);
    node2 = element_nodes(2, num_el);
    element_dof = [node1, node2];
    dx = xx(node2) - xx(node1);
    dy = yy(node2) - yy(node1);
    dz = zz(node2) - zz(node1);
    L = (dx^2 + dy^2 + dz^2)^0.5;

    j = sqrt(-1);

    Zf = rho_f * c_o / A_i;
    kL = frequencies * 2 * pi / c_o * L;

    for num_fq = 1 : number_freq
        c1 = -j * cos(kL(num_fq)) / (sin(kL(num_fq)) * Zf);
        c2 = j / (Zf * sin(kL(num_fq)));
        % Local mobility matrix
        K0 = [c1, c2; c2, c1];
        % Assembly
        KKa(element_dof, element_dof, num_fq) = KKa(element_dof, element_dof, num_fq) + K0;
    end
end

Acoustic.mobility_matrix = KKa;
```

A.7 Acoustic Harmonic Analysis

Using the mobility matrix presented above, and with the prescribed pressures, volume velocities and acoustic impedance the pressure field is given by the code below.

Listing A.8: *acoust_harmonic_analysis* function MATLAB code

```matlab
function [SolutionAcoustic] = acoust_harmonic_analysis(Acoustic, SolutionAcoustic)

number_dofs = Acoustic.number_dofs;
free_dofs = Acoustic.free_dofs;
prescribed_dofs = Acoustic.prescribed_dofs;
prescribed_values = Acoustic.prescribed_values;
volume_velocity_vector = Acoustic.volume_velocity_vector;
harm_freq = Acoustic.frequencies;
KKa = Acoustic.mobility_matrix;

number_freq = length(harm_freq);
harm_press = zeros(number_dofs, number_freq);

for num_fq = 1 : number_freq
    harm_press(free_dofs, num_fq) = KKa(free_dofs, free_dofs, num_fq) \ ...
        (volume_velocity_vector(free_dofs) - KKa(free_dofs, prescribed_dofs, num_fq) * ...
        prescribed_values');
    harm_press(prescribed_dofs, num_fq) = prescribed_values;
end

SolutionAcoustic.harmonic.frequencies = harm_freq;
SolutionAcoustic.harmonic.pressures = harm_press;
```

A.8 Coupled Force Vector

With pressure field defined, the fluid-structure interactive forces resulting from that pressure field are calculated, and are summed to an existing external applied force vector.

Listing A.9: *process_coupled_forces* function MATLAB code

```matlab
function [Coupled] = process_coupled_forces(...
    SolutionAcoustic, Structural, Section, Material, Coupled, Mesh)

F_struct = Structural.force_vector;
number_dofs = Structural.number_dofs;
harm_press = SolutionAcoustic.harmonic.pressures;
harm_freq = SolutionAcoustic.harmonic.frequencies;
Di = Section.internal_diameter;
De = Section.external_diameter;
A = Section.area;
A_i = Section.internal_area;
number_elements = Mesh.number_elements;
node_coordinates = Mesh.node_coordinates;
element_nodes = Mesh.element_nodes;
Nu = Material.poisson_coefficient;

number_freq = length(harm_freq);
force_vector = zeros(number_dofs, number_freq);

xx = node_coordinates(1, :);
yy = node_coordinates(2, :);
zz = node_coordinates(3, :);

for num_fq = 1 : number_freq
    F_fluid = zeros(number_dofs, 1);
    for num_el = 1 : number_elements
        node1 = element_nodes(1, num_el);
        node2 = element_nodes(2, num_el);
        node1_dof = [6*node1-5, 6*node1-4, 6*node1-3];
        node2_dof = [6*node2-5, 6*node2-4, 6*node2-3];
        dx = xx(node2) - xx(node1);
        dy = yy(node2) - yy(node1);
        dz = zz(node2) - zz(node1);
        L = (dx^2 + dy^2 + dz^2)^0.5;

        % nodal pressure
        press_1 = harm_press(node1, num_fq);
        press_2 = harm_press(node2, num_fq);
        press_avg = (press_1 + press_2) / 2;

        % Poisson and junction
        stress_axial = (press_avg * Di^2) / (De^2 - Di^2);
        f_p = - 2 * Nu * A * stress_axial;
        f_j = press_avg * A_i;
        F = f_j + f_p;

        ux = (node_coordinates(:, node2) - node_coordinates(:, node1)) / L;
        F_global_node1 = -F * [ux(1); ux(2); ux(3)];
        F_global_node2 = F * [ux(1); ux(2); ux(3)];
        F_fluid(node1_dof) = F_fluid(node1_dof) + F_global_node1;
        F_fluid(node2_dof) = F_fluid(node2_dof) + F_global_node2;
    end
    force_vector(:, num_fq) = F_struct + F_fluid;
end

Coupled.force_vector = force_vector;
Coupled.frequencies = harm_freq;
```

A.9 Coupled Harmonic Analysis

Harmonic displacement is now calculated given the frequency dependent force vector and structural matrices.

Listing A.10: *coupled_harmonic_analysis* function MATLAB code

```
function [SolutionCoupled] = coupled_harmonic_analysis(Structural, Coupled)

number_dofs = Structural.number_dofs;
free_dofs = Structural.free_dofs;

FF = Coupled.force_vector;
KK = Structural.stiffness_matrix;
MM = Structural.mass_matrix;
CC = Structural.damping_matrix;
harm_freq = Coupled.frequencies;

number_freq = length(harm_freq);
harm_displ = zeros(number_dofs, number_freq);

j = sqrt(-1);

for num_fq = 1 : number_freq
    freq_hz = harm_freq(num_fq);
    freq_rad = freq_hz * 2 * pi;
    dynamic_matrix = KK - freq_rad^2 * MM + j * freq_rad * CC;
    d = dynamic_matrix(free_dofs, free_dofs) \ FF(free_dofs, num_fq);
    harm_displ(free_dofs, num_fq) = (real(d).^2 + imag(d).^2).^0.5;
end

SolutionCoupled.harmonic.displacements = harm_displ;
SolutionCoupled.harmonic.frequencies = harm_freq;
```