SSRG International Journal of Electrical and Electronics Engineering
ISSN: 2348-8379/ https://doi.org/10.14445/23488379/IJEEE-V12I5P124

Volume 12 Issue 5, 287-331, May 2025

Original Article

Computational Simulation of the Stainless-Steel Straight-Tube Coriolis Sensor for Mass Flow Measurement

Javier Pablo Montesinos Quispe[1], Carlos Enrique Villanueva Portal[1], German Alberto Echaiz Espinoza[2], Luis Fernando Gutierrez Belizario[1], Carmelo Mayta Ojeda[3], Fernando Enrique Echaiz Espinoza[4]

[1]*School of Electronic Engineering, National University of San Agustín de Arequipa, Arequipa, Peru.*
[2]*Department of Electronic Engineering, National University of San Agustín de Arequipa, Arequipa, Peru.*
[3]*Department of Physics, National University of San Agustín de Arequipa, Arequipa, Peru.*
[4]*Institute of Mathematics, Federal University of Alagoas, Maceio, Brazil.*

[2]*Corresponding Author : gechaiz@unsa.edu.pe*

Received: 20 March 2025 Revised: 25 April 2025 Accepted: 06 May 2025 Published: 31 May 2025

Abstract *- This paper presents a computational simulation of a stainless steel straight-tube Coriolis sensor, focusing on the design and validation of a mathematical model for mass flow measurement. The Finite Element Method (FEM), implemented in GNU Octave, is used to solve the differential equations governing the tube's vibrational behavior. The simulation determines the angular resonance frequency (723.6 rad/s) and evaluates mass flow for fluid velocities ranging from 0.5 m/s to 25 m/s. The results are validated against reference data from existing literature, demonstrating strong agreement and confirming the accuracy of the proposed model. These findings reinforce the potential of Coriolis sensors for precise and efficient mass flow measurement in industrial applications such as chemical processing and fluid transport monitoring.*

Keywords *- Angular resonance, Computational simulation, Coriolis sensor, Finite element method, Mass flow measurement.*

1. Introduction

1.1. Problem Statement and Research Justification

Numerous industrial applications, including fluid conveyance, product distribution, and process control, rely on accurate mass flow measurement. Coriolis effect-based flowmeters are distinguished by their exceptional reliability and precision among many available approaches. The substantial development expenses, particularly during experimental prototyping, physical testing, and the procurement of commercial sensors, restrict implementation across numerous domains. The Finite Element Method (FEM) is not the sole technique accessible. Current research typically focuses on empirical results or simulations utilizing proprietary software without providing a comprehensive exposition of the equations governing the system's dynamic behavior or employing an open and reproducible methodology. This study addresses this requirement by offering a computational model utilizing the FEM technique, only employing free software such as Python and GNU Octave to simulate the behavior of a straight-tube Coriolis flowmeter. Two primary contributions arise from two fundamental advancements: (1) Python software designed to explicitly visualize the formulation of the equations pertinent to the model and (2) an Octave script that calculates the resonance frequency of the tube, a crucial parameter for optimal sensor design. This approach, in contrast to previous studies, offers open-source code and a flowchart-supported transparent mathematical foundation, enabling other academics and engineers to reproduce and enhance it.

1.2. Importance of the Coriolis Effect in Industrial Processes

The importance of the Coriolis effect pertains to industrial processes. Impact of the Coriolis effect on industrial methods. The implementation of sensors based on the Coriolis effect has transformed safety in industrial facilities, process optimization, and quality control. These tools provide precise measurement of mass flow rate and provide real-time data on additional fluid parameters, including Its determination mostly relies on the characteristics of density and temperature. Concentration in multiphase combinations of various phases. Derive the volumetric flow rate from the volumetric flow rate. Established volumetric flow rate Established volumetric flow rate Established volumetric flow rate Derived volumetric flow rate. The specific density values (such as API degrees, Brix, Plato, Baumé, and Balling) are essential in industries like oil refining, food production, pharmaceuticals, and chemical processes, where precision and real-time responsiveness are crucial.

1.3. A Benefit of Regulating the Coriolis Effect in Flow Measurement

Enumerate the benefits of mitigating the Coriolis effect in flow measurement.

- Advantages of regulating the Coriolis effect in flow measurement.
- Coriolis flow meters uniquely provide direct mass flow measurement, irrespective of fluid characteristics or flow profile.
- The primary advantages are mass flow, density, viscosity, and temperature, all integral to multivariable measurement.
- Standard errors of ±0.1% attain a maximum of ±0.15% in high-precision models [12].
- Independence of flow profile: Straight input or output parts are unnecessary.
- A compact installation is ideal for scenarios with spatial constraints.
- Robust immunity to disruptions in processes and external vibrations.

These advantages elucidate the position of Coriolis sensors as some of the most sophisticated technology in process metrology.

1.4. Fundamental Concept of Coriolis Flowmeters

The controlled oscillation of one or more conduits, through which the fluid flows constitutes the foundation of the operational idea. A quantifiable torsional deviation arises from the interplay between the tube's trajectory and the fluid's inertia. The protocol is as follows: External actuator tube excitation methodology The fluid Coriolis force during transit causes a phase shift. The procedure entails ascertaining the phase shift with meticulously chosen sensors. Calculate the mass flow rate using the time delay as a basis. The density is measured in relation to the natural frequency of oscillation. Integrated thermal sensors facilitate thermal correction.

1.5. Originality and the Contribution of Academic Research

This study is distinctive since it does not depend on private simulation tools or experimental data but rather provides a wholly numerical and accessible framework for modelling Coriolis sensors. This study proposes a free software simulation environment mostly utilising Python and Octave, in contrast to previous research that focused on empirical validation or specialized physical models.

The complete procedure for mathematically obtaining the governing equations is executed. Code that is reproducible and meticulously documented for both scholarly and commercial use. A design methodology based on the tube's resonance frequency facilitates its adaption to various materials. This simulation framework now studies Coriolis sensors via the Finite Element Method (FEM) utilizing free tools, marking the inaugural accessible and open computational investigation.

1.6. Inherent Limitations of the Proposed Model

Although it substantially aids in streamlining the preliminary design of Coriolis sensors, the model has several intrinsic limits. The model primarily focuses on the mechanical design of the tube, enabling it to evaluate various materials, such as PVC, steel, copper, or stainless steel, solely based on Young's modulus, thickness, and elongation, without accounting for thermal expansion or variations in fluid properties. The model excludes both the thermal expansion of the tube and the variation in fluid characteristics. The model presumes a consistent flow, excluding the dynamics of turbulent regimes. There is a lack of adequate physical validation due to the absence of laboratory electrical tests or prototypes. The paradigm remains pertinent in computational and mathematical fields. These constraints delineate the contemporary parameters of the model without undermining its significance. The results gained here are anticipated to serve as a foundation for future developments that integrate thermal elements, advanced fluid dynamics, physical validation, and electronic sensor design.

1.7. Mathematical Formulation of the Coriolis Effect

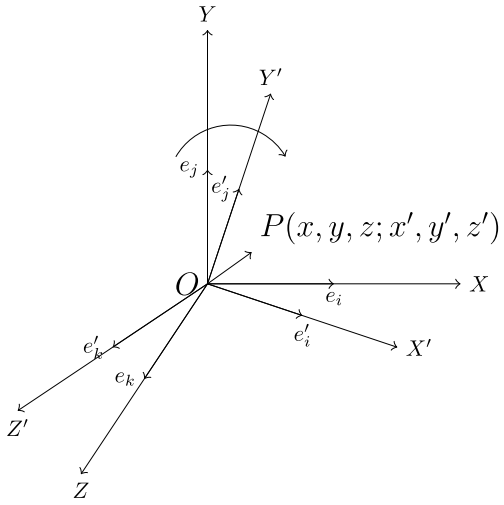


Fig. 1 Fixed and rotated axis system

Considering two observers O and O', where one reference frame rotates relative to the other without relative translation. Both have a common origin, so their position vectors are written as:

For the observer in the fixed reference frame [4]:

$$\vec{r}(t) = x\overrightarrow{e_i} + y\overrightarrow{e_j} + z\overrightarrow{e_k} \quad (1)$$

For the observer in the rotating system:

$$\vec{r}(t) = x'\overrightarrow{e'_i} + y'\overrightarrow{e'_j} + z'\overrightarrow{e'_k} \quad (2)$$

The instantaneous velocity in the stationary system is:

$$\frac{d\vec{r}(t)}{dt} = \frac{dx}{dt}\vec{e_i} + \frac{dy}{dt}\vec{e_j} + \frac{dz}{dt}\vec{e_k} \quad (3)$$

In the rotating system, considering the variation of the unit vectors:

$$\frac{d\vec{r}(t)}{dt} = \frac{dx'}{dt}\vec{e_i'} + x'\frac{d\vec{e_i'}}{dt} + \frac{dy'}{dt}\vec{e_j'} + y'\frac{d\vec{e_j'}}{dt} + \frac{dz'}{dt}\vec{e_k'} + z'\frac{d\vec{e_k'}}{dt} \quad (4)$$

Re-grouping terms:

$$\frac{d\vec{r}(t)}{dt} = \left(\frac{dx'}{dt}\vec{e_i'} + \frac{dy'}{dt}\vec{e_j'} + \frac{dz'}{dt}\vec{e_k'}\right) + \left(x'\frac{d\vec{e_i'}}{dt} + y'\frac{d\vec{e_j'}}{dt} + z'\frac{d\vec{e_k'}}{dt}\right) \quad (5)$$

It is established that:

$$\left.\frac{d\vec{r}}{dt}\right|_{fij} \neq \left.\frac{d\vec{r}}{dt}\right|_{gir} \quad (6)$$

The ratio of velocities between systems is:

$$\left.\frac{d\vec{r}}{dt}\right|_{fij} = \left.\frac{d\vec{r}}{dt}\right|_{gir} + \vec{\Omega} \times \vec{r} \quad (7)$$

Where:
- $\left.\frac{d\vec{r}}{dt}\right|_{fij}$: Absolute velocity in the fixed system.
- $\left.\frac{d\vec{r}}{dt}\right|_{gir}$: Relative velocity in the rotating system.
- $\vec{\Omega}$: Angular velocity vector of the rotating system.
- $\vec{\Omega} \times \vec{r}$: Drag speed due to rotation.

Differential operator:

$$\left.\frac{d}{dt}\right|_{fij} = \left.\frac{d}{dt}\right|_{gir} + \vec{\Omega} \quad (8)$$

Applying to acceleration:

$$\left.\frac{d\vec{v}}{dt}\right|_{fij} = \left.\frac{d\vec{v}}{dt}\right|_{gir} + \vec{\Omega} \times \vec{v} \quad (9)$$

Substituting and developing:

$$\left.\frac{d\vec{v}}{dt}\right|_{fij} = \frac{d}{dt}\left(\vec{v'} + \vec{\Omega} \times \vec{r}\right)|_{gir} + \vec{\Omega} \times \left(\vec{v'} + \vec{\Omega} \times \vec{r}\right) \quad (10)$$

$$= \vec{a'} + \frac{d\vec{\Omega}}{dt} \times \vec{r'} + 2\vec{\Omega} \times \vec{v'} + \vec{\Omega} \times \left(\vec{\Omega} \times \vec{r}\right) \quad (11)$$

Final acceleration ratio:

$$\vec{a} = \vec{a'} + \frac{d\vec{\Omega}}{dt} \times \vec{r'} + 2\vec{\Omega} \times \vec{v'} + \vec{\Omega} \times \left(\vec{\Omega} \times \vec{r}\right) \quad (12)$$

Coriolis acceleration:

$$\vec{a_c} = -2\vec{\Omega} \times \vec{v} \quad (13)$$

Matrix expression:

$$\vec{a_c} = -2\begin{bmatrix} \vec{e_i} & \vec{e_j} & \vec{e_k} \\ 0 & 0 & \Omega \\ v_x & v_y & v_z \end{bmatrix} \quad (14)$$

Coriolis force:

$$\vec{F_c} = m\vec{a_c} = -2m\vec{\Omega} \times \vec{v} = 2\vec{p} \times \vec{\Omega} \quad (15)$$

2. State of the Art

2.1. A Simple Parametric Design Model for Straight-Tube Coriolis Flowmeters. [1]

2.1.1. Objective

Develop a simple parametric model to predict the sensitivity and natural frequency of straight-tube Coriolis flowmeters while minimizing reliance on costly numerical simulations.

2.1.2. Methodology

A one-dimensional (1D) numerical simulation, based on the finite difference method, is used to derive a parametric model characterized by three non-dimensional parameters: bending stiffness (Σ), proximity to the buckling limit (R), and sensor separation distance (χ). The model is experimentally validated using 11 data sets.

2.1.3. Result and Contributions

The model predicts sensitivity with an error margin of 2–5% and enables the estimation of the natural frequency, providing designers with a quick and intuitive tool for optimizing sensor performance.

2.1.4. Limitations

The 1D approach may not fully capture complex three-dimensional dynamics, and validation was performed under a limited range of conditions and materials.

2.2. Coriolis Flowmeters-A Review of Developments Over the Past 20 Years. [2]

2.2.1. Objective

Conduct a comprehensive review of advancements in Coriolis-based flow measurement technology over the past

two decades, covering theoretical models, analytical methods, signal processing techniques, and industrial applications.

2.2.2. Methodology

A systematic literature review integrating studies, patents, and industrial developments to classify and analyze trends and challenges in Coriolis flowmeter technology.

2.2.3. Result and Contributions

Key developments include the expanded application of Coriolis meters, advancements in numerical modeling techniques (e.g., finite element method, FEM, and computational fluid dynamics, CFD), and the emergence of new signal processing methods to enhance accuracy and stability.

2.3. Fluid-Solid Interaction Simulation Methodology for Coriolis Flowmeter Operation Analysis. [3]

2.3.1. Objective

Develop and validate a fluid-solid interaction simulation methodology for Coriolis flowmeters, identifying the most suitable turbulence models and evaluating configuration simplifications.

2.3.2. Methodology

A coupled approach using finite volume simulation for the fluid and finite element modeling for the solid structure. Three turbulence models (RSM, SST, and SST-CC) are compared. The study also examines the impact of modeling simplifications, such as equivalent tube length, sensor mass inclusion, and structural reinforcements. Validation is performed by comparing simulation results with experimental data, ensuring an error margin of less than 5%.

2.3.3. Result and Contributions

- The Reynolds Stress Model (RSM) is the most accurate, particularly in high-speed flows.
- Omitting reinforcements and using an equivalent length approximation introduces significant errors.
- Experimental validation confirms an error of less than %, proving the reliability of the proposed methodology for future Coriolis flowmeter studies.

2.3.4. Limitations

Sensitivity to material property variations is observed. Temperature effects are not considered. The computational cost is high due to the complexity of fluid-solid interaction modeling.

2.4. Analysis

The evolution of mass flow measurement using Coriolis sensors has followed three fundamental research directions:

Development of simplified models for rapid design and performance prediction.

Comprehensive reviews synthesizing advancements in modeling and signal processing techniques.

Experimental validation to assess the accuracy and practical feasibility of theoretical models.

Serving as a useful tool for the first design of sensors without depending on computationally intensive simulations, C.L. Ford's study [1] presented a parametric model that offers a quick and simple approach to estimating sensitivity and natural frequency. Its one-dimensional form, however, limits the model's. Conversely, the current work uses a finite element method (FEM) applied in GNU Octave to enable a complete resolution of the governing differential equations, including the Coriolis, inertial, and elastic components.

This comprehensive formulation enables the depiction of the mass flow response and vibrational dynamics of the sensor with greater accuracy and depth. Furthermore, by utilizing open-source tools and ensuring all computational processes are repeatable, the proposed framework guarantees higher accuracy, flexibility, and transparency, unlike simplified models or commercial black-box software. These elements together help to increase the accuracy of the simulation and show the benefits of this approach above the ones already mentioned in the literature.

In contrast, the review by Tao Wang & Roger Baker [2] provides a broader perspective on the development of Coriolis flowmeters, highlighting the increasing reliance on advanced modeling techniques such as finite element methods (FEM) and computational fluid dynamics (CFD). While these techniques enhance accuracy, the review also identifies challenges such as zero stability and multiphase flow measurement, which remain key areas for innovation.

Evgeniia Shavrina [3] takes a step further by incorporating fluid-solid interaction modeling to evaluate the impact of turbulence models and design simplifications. This study demonstrates that neglecting structural reinforcements or assuming equivalent tube lengths introduces significant errors, reinforcing the importance of high-fidelity simulations. However, it also highlights the high computational cost associated with such detailed modeling.

2.4.1. Identified Gaps in the Literature

Notwithstanding these advancements, there exists a deficiency of:

Lack of computer simulations: No previous work has fully simulated Coriolis sensors for mass flow measurement, especially for a mathematically complex model. Now, published works rely on either simplified parametric models or empirical validation.

Lack of open-source alternatives: For FEM-based studies, past studies rely on proprietary or commercial software. By means of free and open-source software (GNU Octave), our

approach enhances the accessibility of the methodology for next projects and commercial uses.

Lack of mathematical derivations: None of the earlier studies have completely developed the controlling equations, even if numerical and empirical techniques are used. Our work ensures a strong theoretical basis by a complete mathematical treatment of the Coriolis effect in straight-tube sensors.

Restricted transparency in methodology: Unlike most previous studies with just final results, our work presents complete flowchart diagrams showing the sequential simulation process. To ensure total repeatability and encourage more specialised development, we also freely provide all source code in GNU Octave.

2.4.2. Positioning of the Present Study

Building on these previous contributions, the present research integrates a finite element method (FEM) approach to simulate Coriolis sensor vibrations and compute critical parameters such as angular resonance frequency and mass flow rate. By employing fully developed mathematical equations and free computational tools, this study provides an alternative to traditional commercial software while maintaining a high level of accuracy.

Furthermore, by providing all source code in GNU Octave along with detailed flowchart diagrams, we ensure full transparency and reproducibility, setting a new standard for open-source computational research in Coriolis mass flow measurement. This marks the first computational simulation in the field where mathematical derivations, open-source software, and fully accessible code are combined into a single comprehensive framework. Future research should focus on integrating temperature effects, material variability, and real-world validation, ensuring further improvements in accuracy and reliability.

3. Resources and Methods

3.1. Computational Resources

In this study, a computational simulation of the straight stainless steel tube Coriolis sensor was performed. The resources used were as follows:

3.1.1. Software

- GNU Octave: Programming environment used to implement the Finite Element Method (FEM) and solve the differential equations describing the tube's vibration.
- Custom Algorithms: Scripts were developed in Octave for calculating resonance frequencies, phase shift, and mass flow rate.

3.1.2. Hardware

- Computer: A system with an Intel Core i5 8th processor and 16 GB of RAM was used to run the simulations.

3.1.3. Input Data

- Tube Properties: Theoretical data for a stainless-steel tube were used, including its Young's modulus $E = 1.93 \times 10^{11}\ Pa$, density ($\rho_t = 7900$ kg/m³), outer diameter ($D_{ext} = 0.0127$ m), inner diameter ($D_{int} = 9.53 \times 10^{-3}$ m), and length ($L = 0.75$ m).
- Fluid Properties: Water was considered as the working fluid, with a density ($rho_f = 1000$ kg/m³) and a velocity range from 0.5 m/s to 25 m/s.

3.1.4. Simulation Parameters

- Number of Finite Elements: The tube was discretized into 52 finite elements to ensure the accuracy of the results.
- Boundary Conditions: Fixed boundary conditions were applied at the tube's ends to simulate its attachment in a real system.

3.2. Methods

The study was based on the computational simulation of the Coriolis sensor using the Finite Element Method (FEM). The complete details of the simulation methodology are described in the Simulation Methodology section.

3.2.1. Formulation of Forces Acting on a Vibrating Straight Tube

It is proposed to simulate the design of a straight tube sensor to measure mass flow by deriving the differential equations based on the Coriolis Effect that describe its motion. These equations, related to the vibration of a fixed tube at both ends, are solved by means of standardized mathematical expressions in [5-7].

$$EI\frac{\partial^4 y}{\partial x^4} + M_f V^2 \frac{\partial^2 y}{\partial x^2} + 2M_f V \frac{\partial^2 y}{\partial t\,\partial x} + \left(M_t + M_f\right)\frac{\partial^2 y}{\partial t^2} = 0 \tag{16}$$

With boundary conditions imposed at both ends fixed:

$$y(0,t) = \frac{\partial y}{\partial x}\Big|_{x=0} = y(L,t) = \frac{\partial y(L,t)}{\partial x} = 0 \tag{17}$$

The Euler-Bernoulli model is applied to the analysis of vibrations in circular beams fixed at both ends, considering deformations and slopes generated by the motion.

This defines a four-degree-of-freedom system for the vibrating tube-fluid, with key factors being the transverse elongation of the tube and the bending stiffness coefficient derived from Young's modulus and moment of area.

Where:

E: Young's modulus of straight pipe, $[N/m^2\ ]$.

I: Second moment of the cross-sectional area of the circular tube $[m^4]$.

x: Position along the length of the tube, related to the load per unit length.

t: Time variable.
V: Velocity of the fluid inside the tube.
$y = y(x,t)$: Instantaneous elongation of oscillation in the direction transverse to the fluid flow relative to the horizontal axis of symmetry.
EI: Coefficient of bending stiffness of the tube. It comes from the product of Young's modulus and the second moment of area.
V: Velocity of the fluid inside the tube.
M_f: Mass per unit length of the fluid.
M_t: Mass per unit length of the tube.

Equation (16) is in the steady state; each of the terms is expressed at the level of differential equations and is denoted:

- $EI\frac{\partial^4 y}{\partial x^4}$: Elastic force stored by the flexural rigidity of the straight tube; this is the internal force that resists the deformation of the tube due to its rigidity; when the tube is bent, its rigidity generates a force that tries to return it to its original shape. It is like a spring that opposes to be stretched or compressed; the direction of this force is opposite to the direction that the tube is bent.
- $M_f V^2 \frac{\partial^2 y}{\partial x^2}$: Centripetal force that originates due to the motion of the fluid is the force that appears when something moves inside the tube (such as a fluid) at a velocity (V). It is an outward force due to motion.
- $2M_f V \frac{\partial^2 y}{\partial t\, \partial x}$: Coriolis force or Coriolis effect that is evident on the fluid.
- $\left(M_t + M_f\right)\frac{\partial^2 y}{\partial t^2}$: Inertial force of fluid mass and the mass of the oscillating tube section is the resistance of the mass of the tube and the mass of the fluid to change its state of motion (acceleration or deceleration) if the tube is vibrating or moving, its mass generates a force that opposes these changes.

From equation (18), it is obtained that an externally applied force establishes the dynamic equation where each of the terms of the first member is a differential equation as expressed below, which is a standard differential equation that applies to the vibration theory of beams, proposed by Euler-Bernoulli. [8-9], which are very traditional in practical applications:

$$EI\frac{\partial^4 y}{\partial x^4} + M_f V^2 \frac{\partial^2 y}{\partial x^2} + 2M_f V \frac{\partial^2 y}{\partial t\, \partial x} + \left(M_t + M_f\right)\frac{\partial^2 y}{\partial t^2} = F_0 \sin(\omega t) \tag{18}$$

It is considered a tube of length (L), homogeneous linear density, and a uniform circular cross-section along its length. The Figure (2) presents a schematic of the tube-fluid system, highlighting the following aspects:

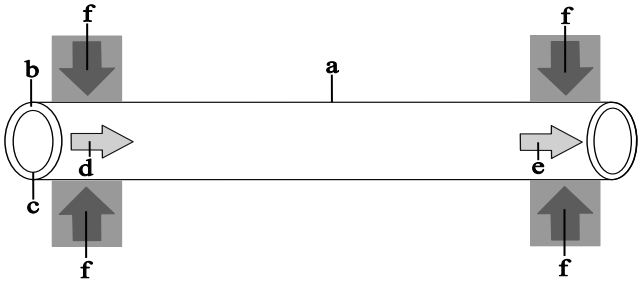


Fig. 2 General schematic of the Coriolis sensor straight tube: (a)Tube Length, (b) Thickness, (c) Tube inner diameter, (d) Fluid inlet velocities, (e) Fluid outlet velocities, and (f) Fixed initial and final ends.

In the absence of external loads, the axis of symmetry of the tube is a straight line joining the centroids of all cross-sections. When a vertical load is applied to the tube, it undergoes distortion, and the resulting deflection curve follows the shape of the tube. It is assumed that the x-axis coincides with the axis of symmetry and that the deflection $y(x,t)$ is measured positively downward. According to the theory of elasticity, the bending moment $M(x)$ at a point x along the tube is related to the load per unit length by a specific equation:

$$\frac{d^2 M(x)}{dx^2} = \omega(x) \tag{19}$$

Where $\omega(x)$ is the mass resistance of the tube and of the fluid mass to the change of its state of motion; in addition, the moment $M(x)$ is proportional to the curvature K of the elastic curve, where E and I are constants, E is the modulus of elasticity of the material used, and I is the second moment of the cross-sectional area of the circular tube. The product is called the bending stiffness. Now, according to the calculations, the curvature is given by:

$$\kappa = \frac{y''}{[1 + y'^2]^{3/2}} \tag{20}$$

For small deflections, $y' \approx 0$, therefore $[1 + y'^2]^{3/2} \approx 1$, and the equation is simplified to:

$$\kappa \approx y'' \tag{21}$$

The second derivative of this expression is:

$$\frac{d^2 M}{dx^2} = EI\frac{d^4 y}{dx^4} = \omega(x) \tag{22}$$

It is seen that the deflection $y(x)$ satisfies the fourth-order differential equation:

$$EI\frac{d^4 y}{dx^4} = \omega(x) \tag{23}$$

The conditions associated with the equation depend on how the pipe ends are located. If both ends are embedded or fixed, then the boundary conditions that are established are:

- $y(0) = U_{i1}$: It expresses the deflection at the initial end.
- $y'(0) = U_{i2}$: The deflection curve is tangent to the x-axis, i.e., the slope of the curve formed by the beam.
- $y(L) = U_{j1}$: Deflection at the end.
- $y'(L) = U_{j2}$: The deflection curve is tangent to the x-axis, the slope at the end.

4. Solution by the Finite Element Method

The solution to the equation is proposed using the finite element method, which involves discretizing the problem into nodes and elements, and then assembling or globalizing the results. A solution of the form is assumed:

$$y(x,t) = (C_1 + C_2x + C_3x^2 + C_4x^3)e^{-i\omega t} \tag{24}$$

From the development of the corresponding shape function (see Appendix A), a globalized function is obtained so that the equation can be rewritten as follows:

$$\begin{aligned} y(x) = &\left(1 - \frac{3x^2}{L^2} + \frac{2x^3}{L^3}\right)U_{i1} \\ &+ \left(x - \frac{2x^2}{L} + \frac{x^3}{L^2}\right)U_{i2} \\ &+ \left(\frac{3x^2}{L^2} - \frac{2x^3}{L^3}\right)U_{j1} \\ &+ \left(-\frac{x^2}{L} + \frac{x^3}{L^2}\right)U_{j2} \end{aligned} \tag{25}$$

Expressed in matrix form:

$$X = \begin{bmatrix} 1 - \frac{3x^2}{L^2} + \frac{2x^3}{L^3} \\ x - \frac{2x^2}{L} + \frac{x^3}{L^2} \\ \frac{3x^2}{L^2} - \frac{2x^3}{L^3} \\ -\frac{x^2}{L} + \frac{x^3}{L^2} \end{bmatrix} \tag{26}$$

Moreover, the initial conditions matrix is:

$$U = \begin{bmatrix} U_{i1} \\ U_{i2} \\ U_{j1} \\ U_{j2} \end{bmatrix} \tag{27}$$

Obtaining the matrix is essential for calculating its elements, which are defined for a segment with two nodes and one element, as per the Finite Element Method. These elements are derived by applying the differential operator on the corresponding function and minimizing errors by means of the Galerkin method, considering a permanent regime where the second member is zero.

$$y(x,t) = [X]^T[U]e^{i\omega t} \tag{28}$$

The elementary matrices $([A]), ([B]), ([C]), ([D])$ are of the same order (4×4) in each case. Considering that the matrices are derived with respect to (x) and then can be summed matrixially $([K] = [A] + [B])$. Writing most compactly, the equation (16) for the permanent regime takes the form:

$$[K_g] + i\,\omega[C_n] - \omega^2[D_n] = 0 \tag{29}$$

Where the angular frequency (ω) of oscillation appears, which is the resonance frequency of the oscillation system of the straight pipe sensor that transports the water fluid.

5. Simulation Methodology

The simulation was conducted using the Finite Element Method (FEM) to model the vibration of a straight stainless steel tube subjected to the Coriolis effect. The process is divided into three main stages:

- Definition of parameters and calculation of geometric properties.
- Assembly of global matrices.
- Calculation of resonance frequencies.

For practical purposes and to optimize simulation time, a discretization of 52 finite elements was employed. However, the number of elements can be adjusted by modifying a single variable in the script provided in the annexes.

The model implementation was carried out in the Octave environment, which provides a balance between accuracy and computational efficiency in calculating the system's dynamic responses.

5.1. Simulation Parameters

Table (1) summarizes the parameters used in the simulation of the straight-tube Coriolis sensor. These values were selected based on the properties of stainless steel and water as the working fluid.

These parameters enable the calculation of the system's geometric and mechanical properties, such as tube thickness, cross-sectional area, and second moment of area, which are essential for the simulation. Additionally, the number of finite elements ($\boldsymbol{n}$) can be modified by adjusting a single variable in the code provided in the annexes, allowing for greater flexibility in the simulation setup.

Table 1. Parameters used in the simulation

Parameter	Value	Unit
Young's Modulus (E)	1.9×10^{11}	Pa
Outer tube diameter (D_{ext})	0.0127	m
Inner tube diameter (D_{int})	9.53×10^{-3}	m
Tube density (ρ_t)	7900	kg/m3
Fluid density (ρ_f)	1000	kg/m3
Tube length (L)	0.75	m
Number of finite elements (n)	52	-
Fluid velocity (V)	5	m/s

5.2. Calculation of Geometric Properties

The geometric and mechanical properties are calculated as follows:

$$E_t = \frac{D_{\text{ext}} - D_{\text{int}}}{2} \tag{30}$$

$$R_m = \frac{D_{\text{ext}} + D_{\text{int}}}{4} \tag{31}$$

$$A_t = \frac{\pi}{4}(D_{\text{ext}}^2 - D_{\text{int}}^2) \tag{32}$$

$$M_t = \rho_t A_t \tag{33}$$

$$I_t = \frac{\pi}{64}(D_{\text{ext}}^4 - D_{\text{int}}^4) \tag{34}$$

$$A_f = \frac{\pi}{4} D_{\text{int}}^2 \tag{35}$$

$$M_f = \rho_f A_f \tag{36}$$

Where:

- (E_t): Total thickness.
- (R_m): Mean radius.
- (A_t): Cross-sectional area.
- (M_t): Cross-sectional mass.
- (I_t): Cross-sectional moment of inertia.
- (A_f): Fluid area.
- (M_f): Fluid mass.

5.3. Assembly of Global Matrices

To model the system dynamics, the global matrices are assembled from the local matrices of each finite element. When using n finite elements, the matrices $[A], [B], [C], and [D]$ have dimensions of $2n + 2$. This is because each element introduces two degrees of freedom (deflection and slope) at each node, and the global system has $n + 1$ nodes.

5.3.1. Local Matrices

The local matrices for each element are of size 4×4 *and are defined as follows [11]:*

Local stiffness matrix:

$$[A]_n = \begin{bmatrix} 12 & 6L & -12 & 6L \\ 6L & 4L^2 & -6L & 2L^2 \\ -12 & -6L & 12 & -6L \\ 6L & 2L^2 & -6L & 4L^2 \end{bmatrix} \tag{37}$$

Local centrifugal force matrix:

$$[B]_n = \begin{bmatrix} \frac{6}{5L} & \frac{1}{10} & -\frac{6}{5L} & \frac{1}{10} \\ \frac{1}{10} & \frac{2L}{15} & -\frac{1}{10} & -\frac{L}{30} \\ -\frac{6}{5L} & -\frac{1}{10} & \frac{6}{5L} & -\frac{1}{10} \\ \frac{1}{10} & -\frac{L}{30} & -\frac{1}{10} & \frac{2L}{15} \end{bmatrix} \tag{38}$$

Local Coriolis matrix:

$$[C]_n = \begin{bmatrix} -\frac{1}{2} & \frac{L}{10} & \frac{1}{2} & -\frac{L}{10} \\ -\frac{L}{10} & 0 & \frac{L}{10} & -\frac{L^2}{60} \\ -\frac{1}{2} & -\frac{L}{10} & \frac{1}{2} & \frac{L}{10} \\ \frac{L}{10} & \frac{L^2}{60} & -\frac{L}{10} & 0 \end{bmatrix} \tag{39}$$

Local mass matrix:

$$[D]_n = \begin{bmatrix} \frac{13L}{35} & \frac{11L^2}{210} & \frac{9L}{70} & -\frac{13L^2}{420} \\ \frac{11L^2}{210} & \frac{L^3}{105} & \frac{13L^2}{420} & -\frac{L^3}{140} \\ \frac{9L}{70} & \frac{13L^2}{420} & \frac{13L}{35} & -\frac{11L^2}{210} \\ -\frac{13L^2}{420} & -\frac{L^3}{140} & -\frac{11L^2}{210} & \frac{L^3}{105} \end{bmatrix} \tag{40}$$

5.3.2. Assembly of the Global Matrix

The global matrix is obtained by assembling the contributions of individual elements, considering the overlap at shared nodes. For $n = 2$, the global matrix is structured as follows:

$$[A]_n = \begin{bmatrix} [A]_{\text{element 1}} & \text{Overlap} \\ \text{Overlap} & [A]_{\text{element 2}} \end{bmatrix} \tag{41}$$

For the specific case of n = 2 elements, the global stiffness matrix $[A]_{global}$ takes the form:

$$[A]_n = \begin{bmatrix} 12 & 6L & -12 & 6L & 0 & 0 \\ 6L & 4L^2 & -6L & 2L^2 & 0 & 0 \\ -12 & -6L & 24 & 0 & -12 & 6L \\ 6L & 2L^2 & 0 & 8L^2 & -6L & 2L^2 \\ 0 & 0 & -12 & -6L & 12 & -6L \\ 0 & 0 & 6L & 2L^2 & -6L & 4L^2 \end{bmatrix} \quad (42)$$

Similarly, the matrices $[B]_n, [C]_n$, and $[D]_n$ are assembled considering the overlap at shared nodes.

5.4. Calculation of Resonance Frequencies

The resonance frequencies are obtained by solving the eigenvalue problem of the following equation:

$$[K_g] = h_1[A]_n + h_2[B]_n \quad (43)$$

$$[E_g] = h_3[C]_n \quad (44)$$

$$[M_g] = h_4[D]_n \quad (45)$$

With:

$$h_1 = \frac{EI_t}{L^3} \quad (46)$$

$$h_2 = M_f V^2 \quad (47)$$

$$h_3 = 2M_f V \quad (48)$$

$$h_4 = M_f + M_t \quad (49)$$

The eigenvalue equation is solved using the following command in GNU Octave:

```
1    [X, e] = polyeig(Kg, i*Eg, -Mg)
```

5.5. Phase Shift and Mass Flow as a Function of Fluid Velocity

The phase shift ($\Delta\phi$) and the mass flow rate ($\dot{m}$) were calculated as a function of the fluid velocity (V) using established mathematical models. The results show that the phase shift decreases as the fluid velocity increases while the mass flow rate follows a linear increasing trend.

5.5.1. Mathematical Models Used

The phase shift is calculated using the following equation [10]:

$$\Delta\phi = c\left(\frac{2f_{21}g_1M_fV}{L\theta_2(M_f+M_t)(g_1^2-g_2^2)}\right)A_1 \quad (50)$$

Where:
($\Delta\phi$): Phase shift.
($\dot{m}$): Mass flow rate.
(c): Modified amplitude.
(f_{21}): Factor related to vibration modes.
(g_1, g_2): Frequencies related to the sensors.
(M_f): Fluid mass per unit length.
(V): Fluid velocity.
(L): Tube length.
(θ_2): Factor related to the second vibration mode.
(A_1): Amplitude ratio.

The mass flow rate is calculated as [10]:

$$\dot{m} = \frac{\Delta\phi\, L\, \theta_2 \left(M_f + M_t\right)(g_1^2 - g_2^2)}{2A_1 g_1 c f_{21}}. \quad (51)$$

5.5.2. Derivation of the Second Expression

To derive the second expression, we start from the first equation and isolate the mass flow rate ($\dot{m}$). The procedure is detailed below:

Start with the phase shift equation [10]:

$$\Delta\phi = c\left(\frac{2f_{21}g_1M_fV}{L\theta_2(M_f+M_t)(g_1^2-g_2^2)}\right)A_1 \quad (52)$$

Isolate the term (M_fV):

$$M_fV = \frac{\Delta\phi\, L\, \theta_2 \left(M_f+M_t\right)(g_1^2-g_2^2)}{2A_1 g_1 c f_{21}} \quad (53)$$

Relate (M_fV) to the mass flow rate ($\dot{m}$), since:

$$\dot{m} = M_fV \quad (54)$$

Finally, obtain the mass flow rate equation:

$$\dot{m} = \frac{\Delta\phi\, L\, \theta_2 \left(M_f+M_t\right)(g_1^2-g_2^2)}{2A_1 g_1 c f_{21}} \quad (55)$$

6. Description of the Programs Used

This section describes the programs developed for the simulation and analysis of the vibration of a straight stainless steel tube subjected to the Coriolis effect.

The programs were implemented in Octave and were used to calculate the resonance frequencies, phase shift, and mass flow rate as a function of fluid velocity.

Simulation parameters employed in this work are shown in Table 1, which displays all the physical and geometrical quantities of interest for the calculations.

The values listed in Table 1 are the input parameters to the Octave programs, with which an accurate reproduction of the vibrational behavior of the tube is achieved, influenced by the Coriolis force. These attributes are the physical properties and dimensions necessary for the computational analysis.

6.1. Program 1: Calculation of Natural Frequencies

The first program aims to calculate the natural resonance frequency of the tube using the Finite Element Method (FEM).

The algorithm is described below:

Algorithm 1: Calculation of Natural Frequencies

Require: Input parameters: $E, D_{ext}, D_{int}, \rho_t, \rho_f, L, N, V$.
Ensure: Natural resonance frequency.
1: Calculate geometric properties: $E_t, R_m, M_t, I_t, A_f, M_f$.
2: Calculate coefficients: h_1, h_2, h_3, h_4.
3: Define local matrices: $[A]_n, [B]_n$: $[C]_n, [D]_n$.
4: Assemble global matrices: $[A]_{global}, [B]_{global}$, $[C]_{global}, [D]_{global}$.
5: Compute global matrices: $[K_g], [E_g], [M_g]$.
6: Solve eigenvalue problem: polyeig($[K_g], i[E_g], -[M_g]$).
7: Obtain natural frequency: frequency=real$(e(S_z, 1))$.

6.2. Program 2: Convergence of the Natural Frequency

The second program evaluates the convergence of the natural frequency as the number of finite elements increases. The algorithm is described below:

Algorithm 2 Convergence of the Natural Frequency

Require: Input parameters: $E, D_{ext}, D_{int}, \rho_t, \rho_f, L, N_{values}, V$.
Ensure: Natural resonance frequency for different values of N.
1: **for** each N in N_{values} ***do***
2: Calculate element length: L= $\frac{L_{total}}{n}$
3: Calculate coefficients: h_1, h_2, h_3, h_4.
4: Define local matrices: $[A]_n, [B]_n$: $[C]_n, [D]_n$.
5: Assemble global matrices: $[A]_{global}, [B]_{global}$, $[C]_{global}, [D]_{global}$.
6: Compute global matrices: $[K_g], [E_g], [M_g]$.
7: Solve eigenvalue problem: polyeig($[K_g], i[E_g], -[M_g]$).
8: Obtain natural frequency: frequency=real$(e(S_z, 1))$.
9: end for
10: Plot natural frequencies vs number of finite elements.

6.3. Program 3: Variation of Frequency with Tube Length

The third program analyzes the variation of the natural frequency as a function of the tube length. The algorithm is described below:

Algorithm 3 Variation of Frequency with Tube Length

Require: Input parameters: $E, D_{ext}, D_{int}, \rho_t, \rho_f, N, L_{values}, V$.
Ensure: Natural frequencies for different tube lengths.
1: **for** each L in L_{values} ***do***
2: Calculate element length: $L_{element} = \frac{L}{n}$
3: Calculate coefficients: h_1, h_2, h_3, h_4.
4: Define local matrices: $[A]_n, [B]_n$: $[C]_n, [D]_n$.
5: Assemble global matrices: $[A]_{global}, [B]_{global}$, $[C]_{global}, [D]_{global}$.
6: Compute global matrices: $[K_g], [E_g], [M_g]$.
7: Solve eigenvalue problem: polyeig($[K_g], i[E_g], -[M_g]$).
8: Obtain natural frequency: frequency=real$(e(S_z, 1))$.
9: end for
10: Plot natural frequencies vs tuve length.

6.4. Program 4: Calculation of Phase Shift and Mass Flow

The fourth program calculates the phase shift and mass flow rate as a function of fluid velocity. The algorithm is described below:

Algorithm 4 Calculation of Phase Shift and Mass Flow

Require: Input parameters: $E, D_{ext}, D_{int}, \rho_t, \rho_f, L, V_{values}$.
Ensure: Phase shift and mass flow rate for different velocities.
1: **for** each V in V_{values} ***do***
2: Calculate geometric properties: $E_t, R_m, M_t, I_t, A_f, A_t, M_f$.
3: Calculate coefficients: h_1, h_2, h_3, h_4.
4: Define local matrices: $[A]_n, [B]_n$: $[C]_n, [D]_n$.
5: Assemble global matrices: $[A]_{global}, [B]_{global}$, $[C]_{global}, [D]_{global}$.
6: Compute global matrices: $[K_g], [E_g], [M_g]$.
7: Solve eigenvalue problem: polyeig($[K_g], i[E_g], -[M_g]$).
8: Calculate phase shift: $\Delta\phi = c\left(\frac{2f_{21}g_1M_fV}{L\theta_2(M_f+M_t)(g_1^2-g_2^2)}\right)A_1$
9: Calculate mass flow rate: $\dot{m} = \frac{\Delta\phi\, L\, \theta_2\, (M_f+M_t)\,(g_1^2-g_2^2)}{2A_1g_1cf_{21}}$
10: **end for**
11: Plot phase shift vs fluid velocity.
12: Plot mass flow rate vs fluid velocity.

Table 2. Resonance angular frequency calculated using FEM

Number of Elements	Angular Frequency (rad/s)
1	878.05
2	724.4
3	724.89
4	723.91
8	723.48
16	723.61
20	723.62
24	723.67
28	723.63
32	723.63
42	723.64
52	723.64

7. Results and Discussion

7.1. Resonance Angular Frequency

The resonance angular frequency was calculated using the Finite Element Method (FEM), discretizing the tube into different numbers of elements. Table (2) shows the obtained values, where it can be observed that the angular frequency converges to 723.6 rad/s as the number of elements increases.

Figure (3) illustrates the convergence of the angular frequency as the number of elements increases. It can be noted that from approximately 16 elements onwards, the values barely vary, confirming the stability of the obtained result.

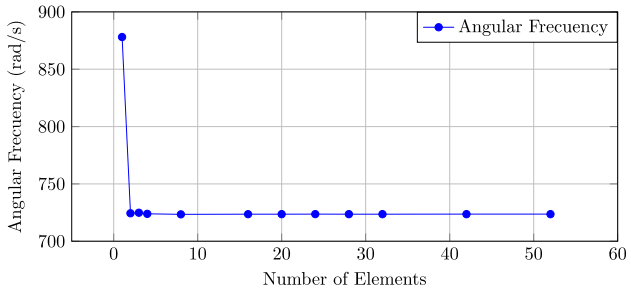


Fig. 3 Convergence of the Resonance Angular Frequency

7.2. Variation of Angular Frequency with Respect to the Length of the Straight Tube

To evaluate the effect of the tube length on the resonance angular frequency, a series of simulations were performed in which the tube lengths were varied while keeping the other parameters constant. The Finite Element Method (FEM) with 52 elements was used, and the natural frequencies were calculated for different tube lengths. The lowest natural frequency was extracted from each simulation and plotted as a function of the tube length.

The results show an inverse relationship between the tube length and the natural frequency, indicating that as the length increases, the natural frequency decreases. This behavior is consistent with the theory of vibrations in flexible structures, where an increase in length results in lower effective stiffness and, consequently, a lower resonance frequency.

Figure (4) illustrates this relationship, clearly showing the decreasing trend of the natural frequency as the tube length increases.

Table 3. Frequency obtained for different lengths of the straight tube

Frequency (Hz)	Length (m)
6512.229	0.25
3322.548	0.35
2009.915	0.45
1345.465	0.55
963.307	0.65
723.539	0.75
563.299	0.85

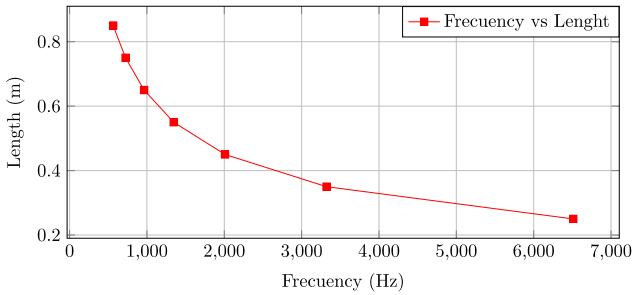


Fig. 4 Variation of Angular Frequency with Respect to the Length of the Straight Tube

7.3. Calculated Mass Flow

The mass flow rate was calculated for fluid velocities ranging from 0.5 m/s to 25 m/s. The results are shown in Table (4) and Figure (5). As observed, the mass flow rate increases linearly with the fluid velocity, which is consistent with theoretical expectations.

Table 4. Results obtained from Algorithm 4 (Mass Flow Measured at Different Velocities)

Velocity (m/s)	Mass Flow (kg/s)
0.5	0.036
5	0.285
10	0.713
15	0.999
20	1.355
25	1.783

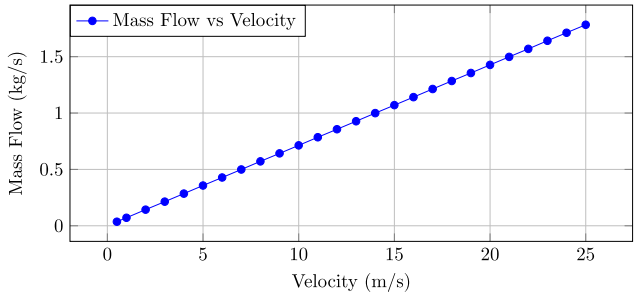


Fig. 5 Relationship between fluid velocity and mass flow

8. Conclusion

This study presented a computational simulation of a Coriolis mass flow sensor with a straight stainless-steel tube, utilizing the Finite Element Method (FEM) implemented in GNU Octave. By discretizing the tube into 52 elements, the system's resonance angular frequency was determined, converging to 723.6 rad/s as the number of elements increased. Additionally, the mass flow rate was computed for fluid velocities ranging from 0.5 m/s to 25 m/s, revealing a linear relationship between fluid velocity and mass flow rate, thereby validating the proposed model.

A key advantage of this model is its flexibility and adaptability. The computational implementation, provided in

the appendices, allows for the simulation of any other straight tube by simply adjusting the input parameters, such as tube dimensions (length, inner and outer diameters), material properties (Young's modulus, density), and fluid characteristics (density, velocity). Moreover, the developed algorithm is not restricted to 52 finite elements; rather, it supports a higher number of elements to enhance simulation accuracy, which is particularly beneficial for complex configurations or more detailed analyses.

The results confirm the feasibility of straight-tube Coriolis sensors for industrial applications, demonstrating their potential for high-precision mass flow measurement. However, it is important to note that this study is based on a theoretical and numerical model, and experimental validation is recommended for future research. Additionally, the influence of other factors, such as fluid viscosity, temperature variations, and non-ideal operating conditions, should be explored to further improve the sensor's robustness and accuracy.

Future Work

The results of the research suggest various conceivable directions for the next research.

Development and evaluation of a physical prototype should be the main focus of future studies since they will help validate the simulation results under practical operational settings, so supporting prototype development. Experimental calibration and error analysis greatly affect the dependability and accuracy of the sensor in industrial environments.

Comparative Productiveness Against Modern Technologies: Especially for sensitivity, fabrication costs, and integration simplicity, a comparison of the straight-tube Coriolis sensor to conventional U-shaped and Omega-shaped devices may provide critical fresh perspectives on their different advantages and limitations.

Tube design can be improved for increased sensitivity and less energy dissipation by the use of advanced optimization methodologies like topological optimisation and evolutionary algorithms. Lowering nonlinear effects and outside disruptions helps the integration of machine learning models to improve data processing and flow rate forecasts.

The effectiveness of the sensor with non-Newtonian fluids (e.g., slurries, polymers) and multiphase flows (e.g., gas-liquid, liquid-solid), frequently employed in industrial applications including chemical processing, food manufacture, and oil transportation, needs more research. The real-time flow measuring, maintenance forecasting help, and process management boosting potential via IoT-based monitoring systems of the Coriolis sensor will help Industry 4.0 applications.

Energy efficiency and power consumption: The sensor's relevance in distant or battery-powered applications increases by low-power operation, hence improving its feasibility for dispersed control systems. By laying a foundation for the manufacturing of more cheaply priced and efficient mass flow sensors, this work considerably advances industrial instrumentation. Future research could look at the optimum tube geometry and include complex geometrical shapes.

Funding Statement

This research has been funded by the authors' own resources.

Acknowledgments

The authors would like to thank the National University of San Agustin (UNSA) and Dr. Fernando from the Federal University of Alagoas (UFAL) for their support during the development of this research. Special thanks to colleagues who provided valuable comments and suggestions during the preparation of this manuscript.

References

[1] C.L. Ford, "A Simple Parametric Design Model for Straight-Tube Coriolis Flow Meters," *Flow Measurement and Instrumentation*, vol. 79, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[2] Tao Wang, and Roger Baker, "Coriolis Flowmeters: A Review of Developments Over the Past 20 Years, and an Assessment of the State of the Art and Likely Future Directions," *Flow Measurement and Instrumentation*, vol. 40, pp. 99-123, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[3] Evgeniia Shavrina et al., "Fluid-Solid Interaction Simulation Methodology for Coriolis Flowmeter Operation Analysis," *Sensors*, vol. 21, no. 23, pp. 1-20, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[4] Walter Hauser, *Introduction to the Principles of Mechanics*, Mexico: Regional Technical Assistance Center, 1969. [Google Scholar]

[5] Steven C. Chapra, and Raymond P. Canale, *Numerical Methods for Engineers*, 2nd ed., McGraw-Hill, 2000. [Google Scholar] [Publisher Link]

[6] Dennis Zill, Warren S. Wright, Michael R. Cullen, *Advanced Engineering Mathematics*, 4th ed., Jones and Bartlett Publishers, pp. 1-970, 2011. [Google Scholar] [Publisher Link]

[7] Clarence Raymond Wylie, and Louis C. Barrett, *Advanced Engineering Mathematics*, 3rd ed., McGraw-Hill, pp. 1-1362, 1995. [Google Scholar] [Publisher Link]

[8] G. Bobovnik, J. Kutin, and I. Bajsić, "The Effect of Flow Conditions on the Sensitivity of the Coriolis Flowmeter," *Flow Measurement and Instrumentation*, vol. 15, no. 2, pp. 69-76, 2004. [CrossRef] [Google Scholar] [Publisher Link]

[9] R. Chessewright, and Simon Shaw, "Uncertainties Associated with Finite Element Modelling of Coriolis Mass Flow Meters," *Flow Measurement and Instrumentation*, vol. 17, no. 6, pp. 335-347, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[10] R. Cheesewright, and C. Clark, "The Effect of Flow Pulsations on Coriolis Mass Flow Meters," *Journal of Fluids and Structures*, vol. 12, no. 8, pp. 1025-1039, 1998. [CrossRef] [Google Scholar] [Publisher Link]

[11] Lisandro Massera, Mauro Podoreskaa, and Mónica Romeroa, "Coriolis Mass Flowmeter: Design and Implementation," *Computational Mechanics*, vol. 26, pp. 3019-3042, 2007. [Google Scholar] [Publisher Link]

[12] *ABB Measurement & Analytics | Data Sheet, Coriolis Master FCD400 Coriolis Mass Flowmeter*. [Publisher Link]

Appendix 1: Math Models

Proposed Form Function

Whereas:

$$y(x,t) = (C_1 + C_2x + C_3x^2 + C_4x^3)e^{i\omega t}$$

When ($x = 0$) at the tail end, then:

$$C_1 = U_{i1}$$

Likewise, the derivative of the function in ($x = 0$):

$$\frac{dy(x)}{dx}|_{x=0} = C_2 = U_{i2}$$

Similarly, at the other end in ($x = L$):

$$U_{i1} + U_{i2}L + C_3L^2 + C_4L^3 = U_{j1}$$

$$\frac{dy}{dx}|_{x=L} = U_{i2} + 2C_3L + 3C_4L^2 = U_{j2}$$

A system of equations is obtained to determine the unknown constants C_3 and C_4:

$$U_{i1} + LU_{i2} + L^2C_3 + L^3C_4 = U_{j1}$$

$$U_{i2} + 2LC_3 + 3L^2C_4 = U_{j2}$$

Resolving:

$$C_3 = \frac{3}{L^2}U_{j1} - \frac{1}{L}U_{j2} - \frac{3}{L^2}U_{i1} - \frac{2}{L}U_{i2}$$

$$C_4 = \frac{2}{L^3}U_{i1} + \frac{1}{L^2}U_{i2} - \frac{2}{L^3}U_{j1} + \frac{1}{L^2}U_{j2}$$

Now substituting for $y(x)$:

$$y(x) = U_{i1} + xU_{i2} + \frac{3x^2}{L^2}U_{i1} - \frac{x^2}{L}U_{i2} - \frac{3x^2}{L^2}U_{i1} - \frac{2x^2}{L}U_{i2} + \frac{2x^3}{L^3}U_{i1} + \frac{x^3}{L^2}U_{i2} - \frac{2x^3}{L^3}U_{j1} + \frac{x^3}{L^2}U_{j2}$$

Derivatives of the elements of the matrix X

The first derivatives are:

$$\frac{dX_{i1}}{dx} = -\frac{6x}{L^2} + \frac{6x^2}{L^3}$$

$$\frac{dX_{i2}}{dx} = 1 - \frac{4x}{L} + \frac{3x^2}{L^2}$$

$$\frac{dX_{j1}}{dx} = \frac{6x}{L^2} - \frac{6x^2}{L^3}$$

$$\frac{dX_{j2}}{dx} = -\frac{2x}{L} + \frac{3x^2}{L^2}$$

The second derivatives are:

$$\frac{d^2X_{i1}}{dx^2} = -\frac{6}{L^2} + \frac{12x}{L^3}$$

$$\frac{d^2X_{i2}}{dx^2} = -\frac{4}{L} + \frac{6x}{L^2}$$

$$\frac{d^2X_{j1}}{dx^2} = \frac{6}{L^2} - \frac{12x}{L^3}$$

$$\frac{d^2X_{j2}}{dx^2} = -\frac{2}{L} + \frac{6x}{L^2}$$

Calculation of the elements of matrix A

The matrix [A] is defined as:

$$A = \left(\frac{\partial^2 X}{\partial x^2}\right)\left(\frac{\partial^2 X}{\partial x^2}\right)^T$$

Where:

$$\left(\frac{\partial^2 X}{\partial x^2}\right) = \begin{bmatrix} -\frac{6}{L^2} + \frac{12x}{L^3} \\ -\frac{4}{L} + \frac{6x}{L^2} \\ \frac{6}{L^2} - \frac{12x}{L^3} - \frac{2}{L} + \frac{6x}{L^2} \end{bmatrix}$$

$$\left(\frac{\partial^2 X}{\partial x^2}\right)^T = \left[\left(-\frac{6}{L^2} + \frac{12x}{L^3}\right)\left(-\frac{4}{L} + \frac{6x}{L^2}\right)\left(\frac{6}{L^2} - \frac{12x}{L^3}\right)\left(-\frac{2}{L} + \frac{6x}{L^2}\right)\right]$$

The elements of the matrix $[A]$ are calculated as:

$$a_{11} = \int_0^L \left(-\frac{6}{L^2} + \frac{12x}{L^3}\right)\left(-\frac{6}{L^2} + \frac{12x}{L^3}\right) dx = \frac{12}{L^3}$$

$$a_{12} = \int_0^L \left(-\frac{6}{L^2} + \frac{12x}{L^3}\right)\left(-\frac{4}{L} + \frac{6x}{L^2}\right) dx = \frac{6}{L^2}$$

$$a_{13} = \int_0^L \left(-\frac{6}{L^2} + \frac{12x}{L^3}\right)\left(\frac{6}{L^2} - \frac{12x}{L^3}\right) dx = -\frac{12}{L^3}$$

$$a_{14} = \int_0^L \left(-\frac{6}{L^2} + \frac{12x}{L^3}\right)\left(-\frac{2}{L} + \frac{6x}{L^2}\right) dx = \frac{6}{L^2}$$

$$a_{21} = \int_0^L \left(-\frac{4}{L} + \frac{6x}{L^2}\right)\left(-\frac{6}{L^2} + \frac{12x}{L^3}\right) dx = \frac{6}{L^2}$$

$$a_{22} = \int_0^L \left(-\frac{4}{L} + \frac{6x}{L^2}\right)\left(-\frac{4}{L} + \frac{6x}{L^2}\right) dx = \frac{4}{L}$$

$$a_{23} = \int_0^L \left(-\frac{4}{L} + \frac{6x}{L^2}\right)\left(\frac{6}{L^2} - \frac{12x}{L^3}\right) dx = -\frac{6}{L^2}$$

$$a_{24} = \int_0^L \left(-\frac{4}{L} + \frac{6x}{L^2}\right)\left(-\frac{2}{L} + \frac{6x}{L^2}\right) dx = \frac{2}{L}$$

$$a_{31} = \int_0^L \left(\frac{6}{L^2} - \frac{12x}{L^3}\right)\left(-\frac{6}{L^2} + \frac{12x}{L^3}\right) dx = -\frac{12}{L^3}$$

$$a_{32} = \int_0^L \left(\frac{6}{L^2} - \frac{12x}{L^3}\right)\left(-\frac{4}{L} + \frac{6x}{L^2}\right) dx = -\frac{6}{L^2}$$

$$a_{33} = \int_0^L \left(\frac{6}{L^2} - \frac{12x}{L^3}\right)\left(\frac{6}{L^2} - \frac{12x}{L^3}\right) dx = \frac{12}{L^3}$$

$$a_{34} = \int_0^L \left(\frac{6}{L^2} - \frac{12x}{L^3}\right)\left(-\frac{2}{L} + \frac{6x}{L^2}\right) dx = -\frac{6}{L^2}$$

$$a_{41} = \int_0^L \left(-\frac{2}{L} + \frac{6x}{L^2}\right)\left(-\frac{6}{L^2} + \frac{12x}{L^3}\right) dx = \frac{6}{L^2}$$

$$a_{42} = \int_0^L \left(-\frac{2}{L} + \frac{6x}{L^2}\right)\left(-\frac{4}{L} + \frac{6x}{L^2}\right) dx = \frac{2}{L}$$

$$a_{43} = \int_0^L \left(-\frac{2}{L} + \frac{6x}{L^2}\right)\left(\frac{6}{L^2} - \frac{12x}{L^3}\right) dx = -\frac{6}{L^2}$$

$$a_{44} = \int_0^L \left(-\frac{2}{L} + \frac{6x}{L^2}\right)\left(-\frac{2}{L} + \frac{6x}{L^2}\right) dx = \frac{4}{L}$$

$$A = \begin{bmatrix} \frac{12}{L^3} & \frac{6}{L^2} & -\frac{12}{L^3} & \frac{6}{L^2} \\ \frac{6}{L^2} & \frac{4}{L} & -\frac{6}{L^2} & \frac{2}{L} \\ -\frac{12}{L^3} & -\frac{6}{L^2} & \frac{12}{L^3} & -\frac{6}{L^2} \\ \frac{6}{L^2} & \frac{2}{L} & -\frac{6}{L^2} & \frac{4}{L} \end{bmatrix}$$

Calculation of the elements of matrix B

The matrix $[B]$ is defined as:

$$B = \frac{\partial X}{\partial x}\left(\frac{\partial X}{\partial x}\right)^T$$

Where:

$$\frac{\partial X}{\partial x} = \begin{matrix} -\frac{6x}{L^2} + \frac{6x^2}{L^3} \\ 1 - \frac{4x}{L} + \frac{3x^2}{L^2} \\ \frac{6x}{L^2} - \frac{6x^2}{L^3} - \frac{2x}{L} + \frac{3x^2}{L^2} \end{matrix}$$

$$\left(\frac{\partial X}{\partial x}\right)^T = \left[\left(-\frac{6x}{L^2} + \frac{6x^2}{L^3}\right)\left(1 - \frac{4x}{L} + \frac{3x^2}{L^2}\right)\left(\frac{6x}{L^2} - \frac{6x^2}{L^3}\right)\left(-\frac{2x}{L} + \frac{3x^2}{L^2}\right)\right]$$

The elements of the matrix B are calculated as:

$$b_{11} = \int_0^L \left(-\frac{6x}{L^2} + \frac{6x^2}{L^3}\right)\left(-\frac{6x}{L^2} + \frac{6x^2}{L^3}\right) dx = \frac{6}{5L}$$

$$b_{12} = \int_0^L \left(-\frac{6x}{L^2} + \frac{6x^2}{L^3}\right)\left(1 - \frac{4x}{L} + \frac{3x^2}{L^2}\right) dx = \frac{1}{10}$$

$$b_{13} = \int_0^L \left(-\frac{6x}{L^2} + \frac{6x^2}{L^3}\right)\left(\frac{6x}{L^2} - \frac{6x^2}{L^3}\right) dx = -\frac{6}{5L}$$

$$b_{14} = \int_0^L \left(-\frac{6x}{L^2} + \frac{6x^2}{L^3}\right)\left(-\frac{2x}{L} + \frac{3x^2}{L^2}\right) dx = \frac{1}{10}$$

$$b_{21} = \int_0^L \left(1 - \frac{4x}{L} + \frac{3x^2}{L^2}\right)\left(-\frac{6x}{L^2} + \frac{6x^2}{L^3}\right) dx = \frac{1}{10}$$

$$b_{22} = \int_0^L \left(1 - \frac{4x}{L} + \frac{3x^2}{L^2}\right)\left(1 - \frac{4x}{L} + \frac{3x^2}{L^2}\right) dx = \frac{2L}{15}$$

$$b_{23} = \int_0^L \left(1 - \frac{4x}{L} + \frac{3x^2}{L^2}\right)\left(\frac{6x}{L^2} - \frac{6x^2}{L^3}\right) dx = -\frac{1}{10}$$

$$b_{24} = \int_0^L \left(1 - \frac{4x}{L} + \frac{3x^2}{L^2}\right)\left(-\frac{2x}{L} + \frac{3x^2}{L^2}\right) dx = -\frac{L}{30}$$

$$b_{31} = \int_0^L \left(\frac{6x}{L^2} - \frac{6x^2}{L^3}\right)\left(-\frac{6x}{L^2} + \frac{6x^2}{L^3}\right) dx = -\frac{6}{5L}$$

$$b_{32} = \int_0^L \left(\frac{6x}{L^2} - \frac{6x^2}{L^3}\right)\left(1 - \frac{4x}{L} + \frac{3x^2}{L^2}\right) dx = -\frac{1}{10}$$

$$b_{33} = \int_0^L \left(\frac{6x}{L^2} - \frac{6x^2}{L^3}\right)\left(\frac{6x}{L^2} - \frac{6x^2}{L^3}\right) dx = \frac{6}{5L}$$

$$b_{34} = \int_0^L \left(\frac{6x}{L^2} - \frac{6x^2}{L^3}\right)\left(-\frac{2x}{L} + \frac{3x^2}{L^2}\right) dx = -\frac{1}{10}$$

$$b_{41} = \int_0^L \left(-\frac{2x}{L} + \frac{3x^2}{L^2}\right)\left(-\frac{6x}{L^2} + \frac{6x^2}{L^3}\right) dx = \frac{1}{10}$$

$$b_{42} = \int_0^L \left(-\frac{2x}{L} + \frac{3x^2}{L^2}\right)\left(1 - \frac{4x}{L} + \frac{3x^2}{L^2}\right) dx = -\frac{L}{30}$$

$$b_{43} = \int_0^L \left(-\frac{2x}{L} + \frac{3x^2}{L^2}\right)\left(\frac{6x}{L^2} - \frac{6x^2}{L^3}\right) dx = -\frac{1}{10}$$

$$b_{44} = \int_0^L \left(-\frac{2x}{L} + \frac{3x^2}{L^2}\right)\left(-\frac{2x}{L} + \frac{3x^2}{L^2}\right) dx = \frac{2L}{15}$$

$$B = \begin{bmatrix} \frac{6}{5L} & \frac{1}{10} & -\frac{6}{5L} & \frac{1}{10} \\ \frac{1}{10} & \frac{2L}{15} & -\frac{1}{10} & -\frac{L}{30} \\ -\frac{6}{5L} & -\frac{1}{10} & \frac{6}{5L} & -\frac{1}{10} \\ \frac{1}{10} & -\frac{L}{30} & -\frac{1}{10} & \frac{2L}{15} \end{bmatrix}$$

Calculation of the Elements of Matrix C

The matrix $[C]$ is defined as:

$$C = X\left(\frac{\partial X}{\partial x}\right)^T$$

Where:

$$X = \begin{matrix} 1 - \frac{3x^2}{L^2} + \frac{2x^3}{L^3} \\ x - \frac{2x^2}{L} + \frac{x^3}{L^2} \\ \frac{3x^2}{L^2} - \frac{2x^3}{L^3} \frac{x^2}{L} + \frac{x^3}{L^2} \end{matrix}$$

$$\left(\frac{\partial X}{\partial x}\right)^T = \left[\left(-\frac{6x}{L^2} + \frac{6x^2}{L^3}\right)\left(1 - \frac{4x}{L} + \frac{3x^2}{L^2}\right)\left(\frac{6x}{L^2} - \frac{6x^2}{L^3}\right)\left(-\frac{2x}{L} + \frac{3x^2}{L^2}\right)\right]$$

The elements of the matrix $[C]$ are calculated as:

$$c_{11} = \int_0^L \left(1 - \frac{3x^2}{L^2} + \frac{2x^3}{L^3}\right)\left(-\frac{6x}{L^2} + \frac{6x^2}{L^3}\right) dx = -\frac{1}{2}$$

$$c_{12} = \int_0^L \left(1 - \frac{3x^2}{L^2} + \frac{2x^3}{L^3}\right)\left(1 - \frac{4x}{L} + \frac{3x^2}{L^2}\right) dx = \frac{L}{10}$$

$$c_{13} = \int_0^L \left(1 - \frac{3x^2}{L^2} + \frac{2x^3}{L^3}\right)\left(\frac{6x}{L^2} - \frac{6x^2}{L^3}\right) dx = \frac{1}{2}$$

$$c_{14} = \int_0^L \left(1 - \frac{3x^2}{L^2} + \frac{2x^3}{L^3}\right)\left(-\frac{2x}{L} + \frac{3x^2}{L^2}\right) dx = -\frac{L}{10}$$

$$c_{21} = \int_0^L \left(x - \frac{2x^2}{L} + \frac{x^3}{L^2}\right)\left(-\frac{6x}{L^2} + \frac{6x^2}{L^3}\right) dx = -\frac{L}{10}$$

$$c_{22} = \int_0^L \left(x - \frac{2x^2}{L} + \frac{x^3}{L^2}\right)\left(1 - \frac{4x}{L} + \frac{3x^2}{L^2}\right) dx = 0$$

$$c_{23} = \int_0^L \left(x - \frac{2x^2}{L} + \frac{x^3}{L^2}\right)\left(\frac{6x}{L^2} - \frac{6x^2}{L^3}\right) dx = \frac{L}{10}$$

$$c_{24} = \int_0^L \left(x - \frac{2x^2}{L} + \frac{x^3}{L^2}\right)\left(-\frac{2x}{L} + \frac{3x^2}{L^2}\right) dx = -\frac{L^2}{60}$$

$$c_{31} = \int_0^L \left(\frac{3x^2}{L^2} - \frac{2x^3}{L^3}\right)\left(-\frac{6x}{L^2} + \frac{6x^2}{L^3}\right) dx = -\frac{1}{2}$$

$$c_{32} = \int_0^L \left(\frac{3x^2}{L^2} - \frac{2x^3}{L^3}\right)\left(1 - \frac{4x}{L} + \frac{3x^2}{L^2}\right) dx = -\frac{L}{10}$$

$$c_{33} = \int_0^L \left(\frac{3x^2}{L^2} - \frac{2x^3}{L^3}\right)\left(\frac{6x}{L^2} - \frac{6x^2}{L^3}\right) dx = \frac{1}{2}$$

$$c_{34} = \int_0^L \left(\frac{3x^2}{L^2} - \frac{2x^3}{L^3}\right)\left(-\frac{2x}{L} + \frac{3x^2}{L^2}\right) dx = \frac{L}{10}$$

$$c_{41} = \int_0^L \left(-\frac{x^2}{L} + \frac{x^3}{L^2}\right)\left(-\frac{6x}{L^2} + \frac{6x^2}{L^3}\right) dx = \frac{L}{10}$$

$$c_{42} = \int_0^L \left(-\frac{x^2}{L} + \frac{x^3}{L^2}\right)\left(1 - \frac{4x}{L} + \frac{3x^2}{L^2}\right) dx = \frac{L^2}{60}$$

$$c_{43} = \int_0^L \left(-\frac{x^2}{L} + \frac{x^3}{L^2}\right)\left(\frac{6x}{L^2} - \frac{6x^2}{L^3}\right) dx = -\frac{L}{10}$$

$$c_{44} = \int_0^L \left(-\frac{x^2}{L} + \frac{x^3}{L^2}\right)\left(-\frac{2x}{L} + \frac{3x^2}{L^2}\right) dx = 0$$

$$C = \begin{bmatrix} -\frac{1}{2} & \frac{L}{10} & \frac{1}{2} & -\frac{L}{10} \\ -\frac{L}{10} & 0 & \frac{L}{10} & -\frac{L^2}{60} \\ -\frac{1}{2} & -\frac{L}{10} & \frac{1}{2} & \frac{L}{10} \\ \frac{L}{10} & \frac{L^2}{60} & -\frac{L}{10} & 0 \end{bmatrix}$$

Calculation of the Elements of Matrix D

The matrix $[D]$ is defined as:

$$D = (X)(X)^T$$

Where:

$$X = \begin{matrix} 1 - \frac{3x^2}{L^2} + \frac{2x^3}{L^3} \\ x - \frac{2x^2}{L} + \frac{x^3}{L^2} \\ \frac{3x^2}{L^2} - \frac{2x^3}{L^3} \frac{x^2}{L} + \frac{x^3}{L^2} \end{matrix}$$

$$X^T = \left[\left(1 - \frac{3x^2}{L^2} + \frac{2x^3}{L^3}\right)\left(x - \frac{2x^2}{L} + \frac{x^3}{L^2}\right)\left(\frac{3x^2}{L^2} - \frac{2x^3}{L^3}\right)\left(-\frac{x^2}{L} + \frac{x^3}{L^2}\right)\right]$$

The elements of the matrix $[D]$ are calculated as:

$$d_{11} = \int_0^L \left(1 - \frac{3x^2}{L^2} + \frac{2x^3}{L^3}\right)^2 dx = \frac{13L}{35}$$

$$d_{12} = \int_0^L \left(1 - \frac{3x^2}{L^2} + \frac{2x^3}{L^3}\right)\left(x - \frac{2x^2}{L} + \frac{x^3}{L^2}\right) dx = \frac{11L^2}{210}$$

$$d_{13} = \int_0^L \left(1 - \frac{3x^2}{L^2} + \frac{2x^3}{L^3}\right)\left(\frac{3x^2}{L^2} - \frac{2x^3}{L^3}\right) dx = \frac{9L}{70}$$

$$d_{14} = \int_0^L \left(1 - \frac{3x^2}{L^2} + \frac{2x^3}{L^3}\right)\left(-\frac{x^2}{L} + \frac{x^3}{L^2}\right) dx = -\frac{13L^2}{420}$$

$$d_{21} = \int_0^L \left(x - \frac{2x^2}{L} + \frac{x^3}{L^2}\right)\left(1 - \frac{3x^2}{L^2} + \frac{2x^3}{L^3}\right) dx = \frac{11L^2}{210}$$

$$d_{22} = \int_0^L \left(x - \frac{2x^2}{L} + \frac{x^3}{L^2}\right)\left(x - \frac{2x^2}{L} + \frac{x^3}{L^2}\right) dx = \frac{L^3}{105}$$

$$d_{23} = \int_0^L \left(x - \frac{2x^2}{L} + \frac{x^3}{L^2}\right)\left(\frac{3x^2}{L^2} - \frac{2x^3}{L^3}\right) dx = \frac{13L^2}{420}$$

$$d_{24} = \int_0^L \left(x - \frac{2x^2}{L} + \frac{x^3}{L^2}\right)\left(-\frac{x^2}{L} + \frac{x^3}{L^2}\right) dx = -\frac{L^3}{140}$$

$$d_{31} = \int_0^L \left(\frac{3x^2}{L^2} - \frac{2x^3}{L^3}\right)\left(1 - \frac{3x^2}{L^2} + \frac{2x^3}{L^3}\right) dx = \frac{9L}{70}$$

$$d_{32} = \int_0^L \left(\frac{3x^2}{L^2} - \frac{2x^3}{L^3}\right)\left(x - \frac{2x^2}{L} + \frac{x^3}{L^2}\right) dx = \frac{13L^2}{420}$$

$$d_{33} = \int_0^L \left(\frac{3x^2}{L^2} - \frac{2x^3}{L^3}\right)\left(\frac{3x^2}{L^2} - \frac{2x^3}{L^3}\right) dx = \frac{13L}{35}$$

$$d_{34} = \int_0^L \left(\frac{3x^2}{L^2} - \frac{2x^3}{L^3}\right)\left(-\frac{x^2}{L} + \frac{x^3}{L^2}\right) dx = -\frac{11L^2}{210}$$

$$d_{41} = \int_0^L \left(-\frac{x^2}{L} + \frac{x^3}{L^2}\right)\left(1 - \frac{3x^2}{L^2} + \frac{2x^3}{L^3}\right) dx = -\frac{13L^2}{420}$$

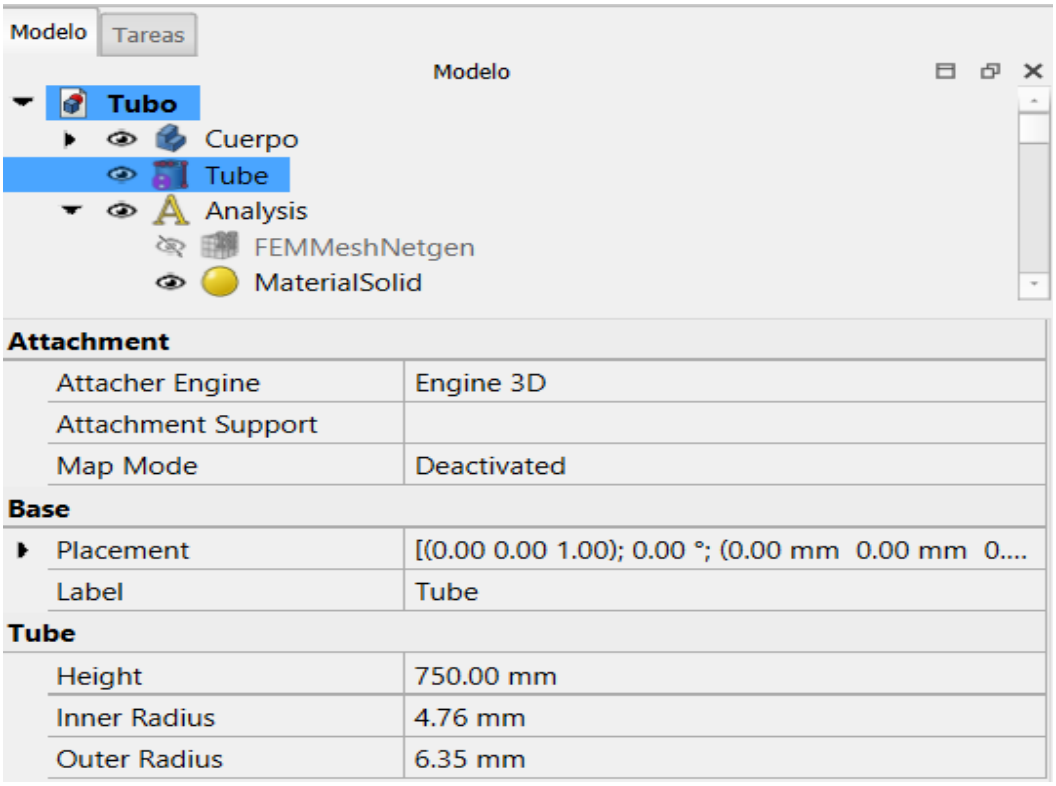
$$d_{42} = \int_0^L \left(-\frac{x^2}{L} + \frac{x^3}{L^2}\right)\left(x - \frac{2x^2}{L} + \frac{x^3}{L^2}\right) dx = -\frac{L^3}{140}$$

$$d_{43} = \int_0^L \left(-\frac{x^2}{L} + \frac{x^3}{L^2}\right)\left(\frac{3x^2}{L^2} - \frac{2x^3}{L^3}\right) dx = -\frac{11L^2}{210}$$

$$d_{44} = \int_0^L \left(-\frac{x^2}{L} + \frac{x^3}{L^2}\right)\left(-\frac{x^2}{L} + \frac{x^3}{L^2}\right) dx = \frac{L^3}{105}$$

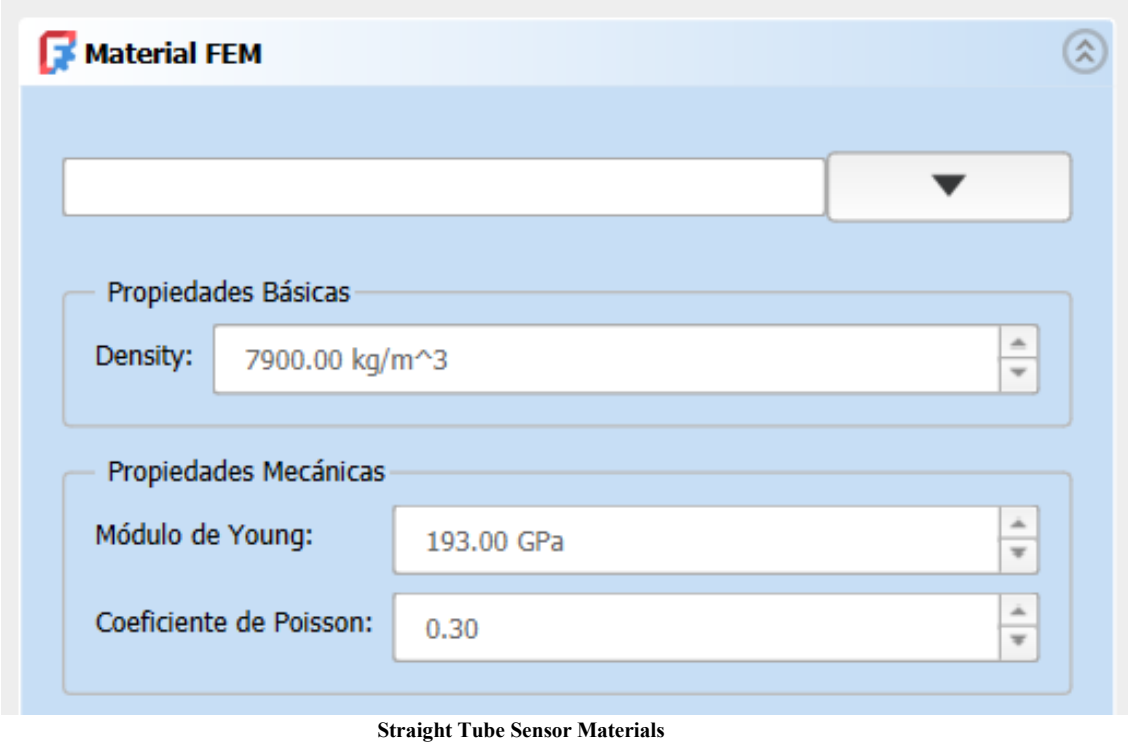
$$D = \begin{bmatrix} \frac{13L}{35} & \frac{11L^2}{210} & \frac{9L}{70} & -\frac{13L^2}{420} \\ \frac{11L^2}{210} & \frac{L^3}{105} & \frac{13L^2}{420} & -\frac{L^3}{140} \\ \frac{9L}{70} & \frac{13L^2}{420} & \frac{13L}{35} & -\frac{11L^2}{210} \\ -\frac{13L^2}{420} & -\frac{L^3}{140} & -\frac{11L^2}{210} & \frac{L^3}{105} \end{bmatrix}$$

Appendix 2: Simulation with FreeCAD

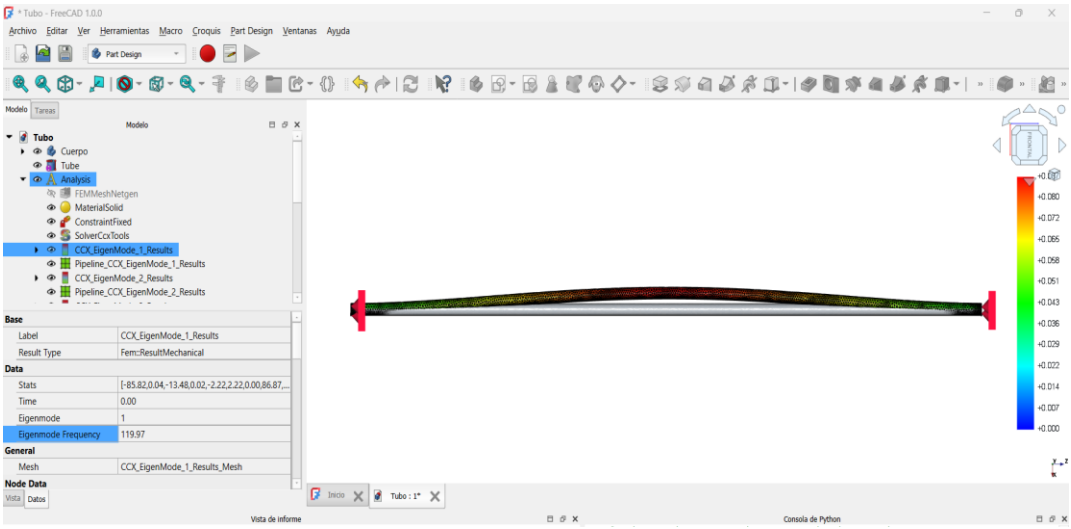


Attachment	
Attacher Engine	Engine 3D
Attachment Support	
Map Mode	Deactivated
Base	
Placement	[(0.00 0.00 1.00); 0.00 °; (0.00 mm 0.00 mm 0....
Label	Tube
Tube	
Height	750.00 mm
Inner Radius	4.76 mm
Outer Radius	6.35 mm

Straight Tube Sensor Dimensions

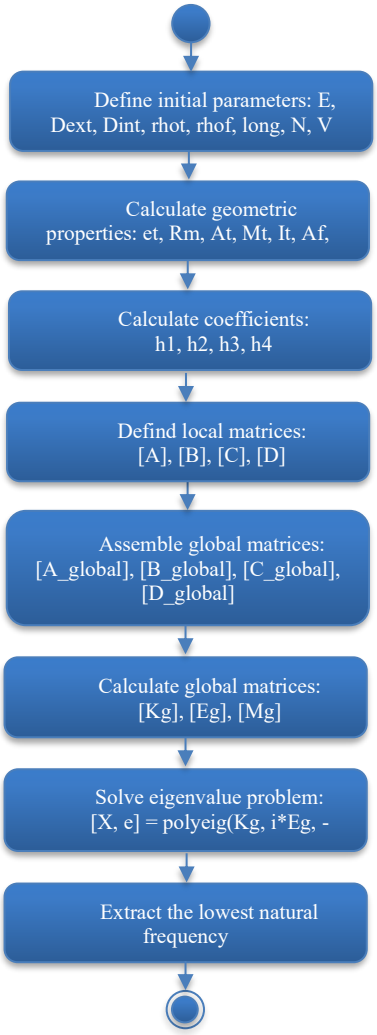


Straight Tube Sensor Materials

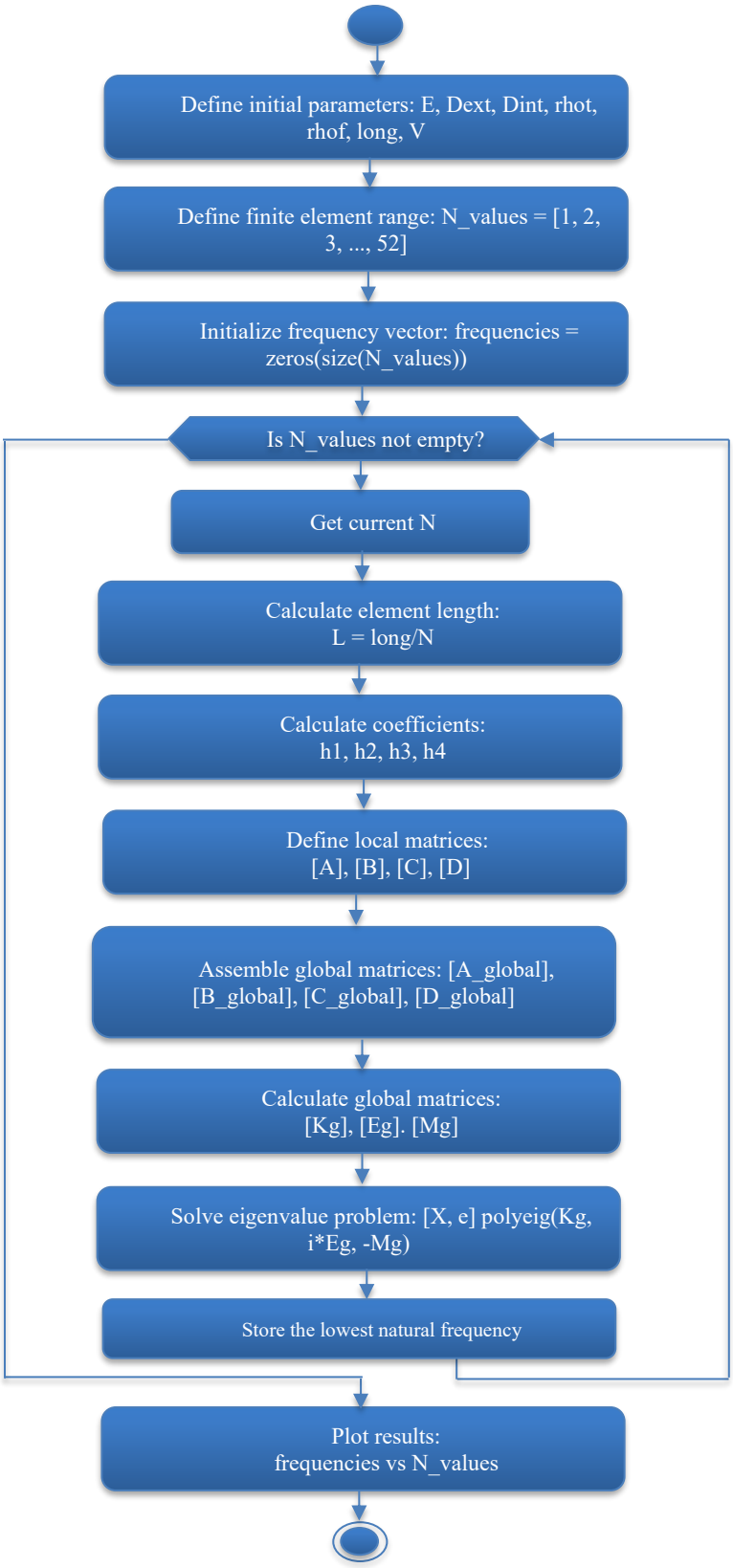


FreeCAD Simulation Results

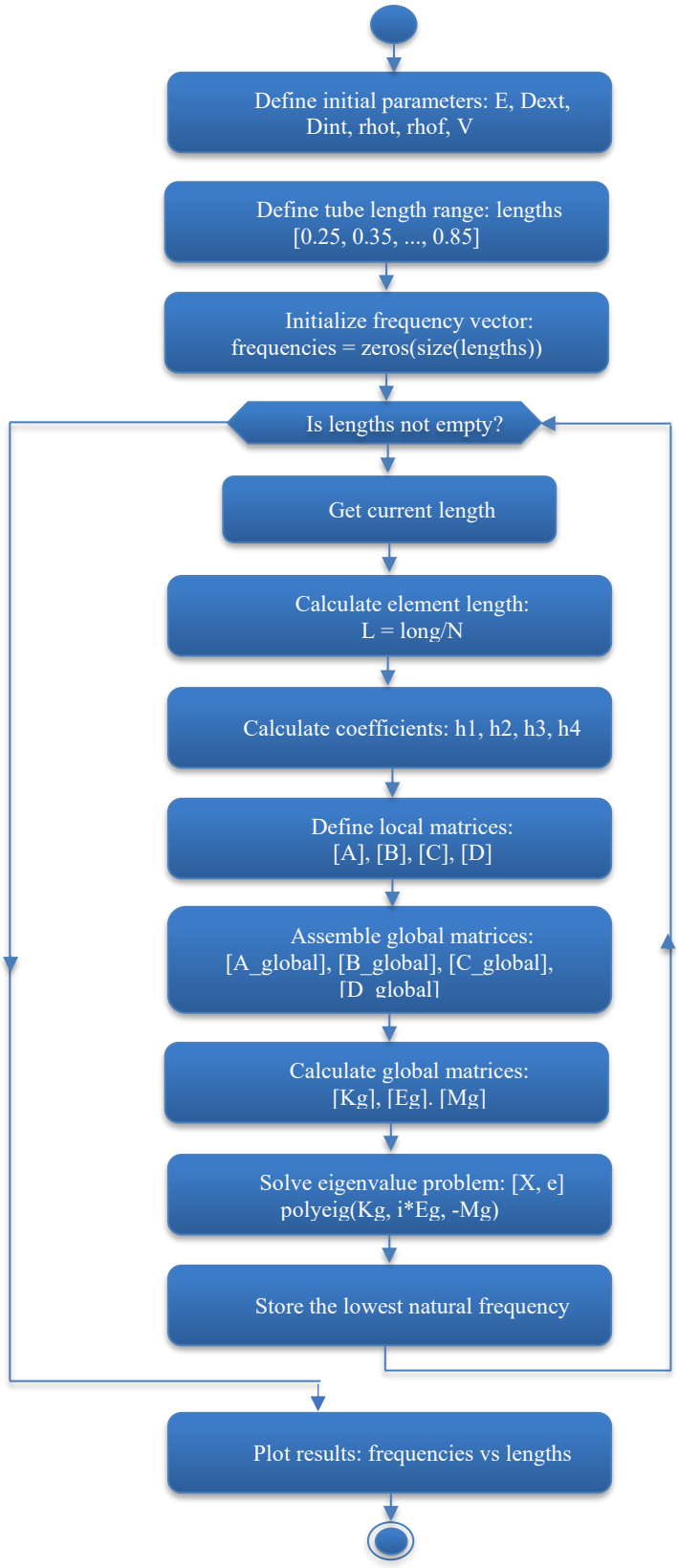
Appendix 3: Flowcharts



Flowchart of Program 1: Calculation of Natural Frequencies



Flowchart of Program 2: Convergence of the Natural Frequency



Flowchart of Program 3: Frequency Variation with Tube Length

Flowchart of Program 4: Calculation of Phase Shift and Mass Flow

Appendix 4: Simulation codes

CODE 1: Calculation of Natural Frequencies

```
clc, clear all

% II. EIGENVALUE RANGE AND MODE VALUE DEFINITION
i = 1:1:5; % ITERATION VARIABLE FOR EIGENVALUES
lan1 = 4.73; % FIRST FUNDAMENTAL MODE [21]-[23]
lan2 = 7.853; % SECOND MODE
f21 = 3.399; % FACTOR
theta1 = 1.0359; % THETA1
theta2 = 0.9984; % THETA2
xi11 = -12.74; % FACTOR11
xi22 = -45.98; % FACTOR22

% III. STAINLESS STEEL PIPE INPUT PARAMETERS
% DATA ENTRY IN FILE: STEEL P0_1
E = 1.93e11;       % YOUNG'S MODULUS [N/m^2]
Dext = 0.0127; % OUTER DIAMETER [m]
Dint = 9.53000000000000e-003; % INNER DIAMETER [m]
rhot = 7900; % TUBE DENSITY [kg/m^3]
rhof = 1000; % FLUID DENSITY [kg/m^3]
L_total = 0.75; % TUBE LENGTH [m]
N = 1; % NUMBER OF ITERATION ELEMENTS
L = L_total / N; % LENGTH OF EACH ELEMENT [m]
x = 0:0.025:L; % TUBE VARIABLE [m]
V = 10; % FLUID VELOCITY [m/s]

et = (Dext - Dint) / 2; % TUBE THICKNESS [m]
```

```
Rm = (Dext + Dint) / 2; % MEAN TUBE RADIUS [m]
At = pi * (Dext^2 - Dint^2) / 4; % CROSS-SECTIONAL AREA OF TUBE [m^2]
Mt = rhot * At; % PIPE MASS PER UNIT LENGTH [kg/m]
It = pi * (Dext^4 - Dint^4) / 64; % SECOND MOMENT OF AREA [m^4]
Af = pi * Dint^2 / 4; % FLOW AREA [m^2]
Mf = rhof * Af; % FLUID MASS PER UNIT LENGTH [kg/m]
omega = 720; % RESONANCE ANGULAR FREQUENCY [rad/s]
C = 1.5e-3; % OSCILLATION AMPLITUDE [m]

% IV. DEFINITION OF OTHER VIBRATION PARAMETERS [21]-[23]
beta1 = lan1 / L; % ARGUMENT FIRST MODE
beta2 = lan2 / L; % ARGUMENT SECOND MODE
sigma1 = (sinh(beta1 * L) - sin(beta1 * L)) / (cos(beta1 * L) - cosh(beta1 * L));
sigma2 = (sinh(beta2 * L) - sin(beta2 * L)) / (cos(beta2 * L) - cosh(beta2 * L));
omega1 = (xi11 * Mf * V^2) / (L^2 * theta1 * (Mf + Mt)); % ARGUMENT 1
omega2 = (xi22 * Mf * V^2) / (L^2 * theta2 * (Mf + Mt)); % ARGUMENT 2
gamma1 = sqrt(omega^2 - (xi11 * Mf * V^2) / (L^2 * theta1 * (Mf + Mt))); % SENSOR FREQUENCY 1
gamma2 = sqrt(omega^2 - (xi22 * Mf * V^2) / (L^2 * theta2 * (Mf + Mt))); % SENSOR FREQUENCY 2

% V. CRITICAL VALUES AND FUNCTION DEFINITION
% CHANGES ARE INTRODUCED IN THIS SECTION [21]-[23]
a = L / 4; % CRITICAL POSITION 1
b = (3 * L) / 4; % CRITICAL POSITION 2
c = L / 2; % CRITICAL POSITION 3
t = pi / (2 * omega); % TIME 1
t1 = (3 * pi) / (8 * omega); % TIME 2
w0 = sinh(beta1 * x) - sin(beta1 * x) + sigma1 * (cosh(beta1 * x) - cos(beta1 * x)); % ORIGINAL FUNCTION
```

```
54      w1 = sinh(beta2 * a) - sin(beta2 * a) + sigma2 * (cosh(beta2 * a) - cos(beta2 * a)) * cos(t);
55      w2 = sinh(beta2 * b) - sin(beta2 * b) + sigma2 * (cosh(beta2 * b) - cos(beta2 * b)) * cos(t1);
56      w3 = (cosh(beta2 * c) - cos(beta2 * c) - sigma1 * (sinh(beta2 * c) - sin(beta2 * c))) * cos(pi / 4);
57      c = C / w3; % MODIFIED AMPLITUDE
58      A1 = (w1 - w2) / (L / 4); % AMPLITUDE RATIO DEFINITION
59
60      % VI. PHASE SHIFT CALCULATION (IN RADIANS)
61      PhaseShiftTheory = 0.88 * pi / 180; % PHASE SHIFTS GENERATED FOR VELOCITIES OR FLOWS
62      PhaseShift = c * ((2 * f21 * gamma1 * Mf * V) / (L * theta2 * (Mf + Mt) * (gamma1^2 - gamma2^2))) * A1;
63      PhaseError = ((PhaseShiftTheory - PhaseShift) / PhaseShiftTheory) * 100;
64
65      % VII. MASS FLOW CALCULATION
66      z0 = Mf * V; % THEORETICAL FLOW
67      z = (PhaseShift * L * theta2 * (Mf + Mt) * (gamma1^2 - gamma2^2)) / (2 * A1 * gamma1 * c * f21);
    %CALCULATED MASS FLOW
68      % IX. GRAPHICS
69      % Define velocity range
70      velocities = [0.5, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25];
71
72      % Initialize vectors to store results
73      PhaseShift = zeros(size(velocities));
74      mass_flow = zeros(size(velocities));
75
76      % Calculate phase shift and mass flow for each velocity
77      for i = 1:length(velocities)
78          V = velocities(i); % Current velocity
79          omega1 = (xi11 * Mf * V^2) / (L^2 * theta1 * (Mf + Mt)); % ARGUMENT 1
80
```

```
81          omega2 = (xi22 * Mf * V^2) / (L^2 * theta2 * (Mf + Mt)); % ARGUMENT 2
82          gamma1 = sqrt(omega^2 - (xi11 * Mf * V^2) / (L^2 * theta1 * (Mf + Mt))); % SENSOR FREQUENCY 1
83          gamma2 = sqrt(omega^2 - (xi22 * Mf * V^2) / (L^2 * theta2 * (Mf + Mt))); % SENSOR FREQUENCY 2
84
85          % Calculate phase shift
86          PhaseShift(i) = c * ((2 * f21 * gamma1 * Mf * V) / (L * theta2 * (Mf + Mt) * (gamma1^2 - gamma2^2))) *
    A1;
87
88          % Calculate mass flow
89          mass_flow(i) = (PhaseShift(i) * L * theta2 * (Mf + Mt) * (gamma1^2 - gamma2^2)) / (2 * A1 * gamma1 * c
90  * f21);
91      end
92
93      % a) CORIOLIS FLOW VS VELOCITY PLOT
94      figure(1)
95      plot(velocities, mass_flow, 'r o')
96      title('CORIOLIS FLOW vs VELOCITY')
97      xlabel('Fluid velocity [m/s]')
98      ylabel('Mass flow [kg/s]')
99      grid on
100
101     % b) PHASE SHIFT VS VELOCITY PLOT
102     figure(2)
103     plot(velocities, PhaseShift, 'g o')
104     title('PHASE SHIFT vs VELOCITY')
105     xlabel('Fluid velocity [m/s]')
106     ylabel('Phase angle shift [rad/s]')
        grid on
```

CODE 2: Convergence of the Natural Frequency

```
clc, clear all, close all;
format longeng

function global_matrix = assemble_global_matrix(n, L, local_matrix_fn)
    dof = 2 * (n + 1); % Total degrees of freedom
    global_matrix = zeros(dof, dof); % Initialize global matrix as numeric

    for i = 1:n
        indices = 2*(i-1) + (1:4); % Global indices
        local_matrix = local_matrix_fn(L);
        global_matrix(indices, indices) = global_matrix(indices, indices) + local_matrix;
    end
end

% Initial parameters
E = 1.93e11;              % YOUNG'S MODULUS
Dext = 0.0127;            % EXTERNAL DIAMETER
Dint = 9.53e-3;           % INTERNAL DIAMETER
rhot = 7900;              % TUBE DENSITY
rhof = 1000;              % FLUID DENSITY
length_total = 0.75;      % TOTAL TUBE LENGTH
V = 5;                    % FLUID VELOCITY

% Predefined results
```

```
et = 1.58500000000000e-003;      % TUBE THICKNESS
Rm = 11.1150000000000e-003;      % MEAN RADIUS OF THE TUBE
At = 55.34630171627l1e-006;      % CROSS-SECTIONAL AREA OF THE TUBE
Mt = 437.235783558542e-003;      % TUBE MASS PER UNIT LENGTH
It = 872.087871085041e-012;      % SECOND MOMENT OF AREA OF THE TUBE
Af = 71.3305680581033e-006;      % CROSS-SECTIONAL AREA FOR THE FLUID
Mf = 71.3305680581033e-003;      % FLUID MASS PER UNIT LENGTH

% Finite element range
N_values = [1 2 3 4 8 16 20 24 28 32 42 52]; % Values of N (number of finite elements)
frequencies = zeros(size(N_values)); % Vector to store natural frequencies

% Iterate over different values of N
for idx = 1:length(N_values)
    N = N_values(idx);
    L = length_total / N;

    % Coefficients
    h1 = (E * It) / (L^3);
    h2 = Mf * V^2;
    h3 = 2 * Mf * V;
    h4 = Mf + Mt;

    % Local matrix functions
    matrixA_local = @(L) [
        12 6*L -12 6*L;
```

```
    6*L 4*L^2 -6*L 2*L^2;
    -12 -6*L 12 -6*L;
    6*L 2*L^2 -6*L 4*(L^2)];

matrixB_local = @(L) [
    6/5*L 1/10 -6/5*L 1/10;
    1/10 2*L/15 -1/10 -L/30;
    -6/5*L -1/10 6/5*L -1/10;
    1/10 -L/30 -1/10 2*L/15];

matrixC_local = @(L) [
    -1/2 L/10 1/2 -L/10;
    -L/10 0 L/10 -L^2/60;
    -1/2 -L/10 1/2 L/10;
    L/10 L^2/60 -L/10 0];

matrixD_local = @(L) [
    13*L/35 11*L^2/210 9*L/70 -13*L^2/420;
    11*L^2/210 L^3/105 13*L^2/420 -L^3/140;
    9*L/70 13*L^2/420 13*L/35 -11*L^2/210;
    -13*L^2/410 -L^3/140 -11*L^2/210 L^3/105];

% Assemble global matrices
A_global = assemble_global_matrix(N, L, matrixA_local);
B_global = assemble_global_matrix(N, L, matrixB_local);
C_global = assemble_global_matrix(N, L, matrixC_local);
```

```
77          D_global = assemble_global_matrix(N, L, matrixD_local);
78
79          % Calculate global matrices
80          Kg = h1 * A_global + h2 * B_global; % Global stiffness matrix
81          Eg = h3 * C_global;              % Global Coriolis matrix
82          Mg = h4 * D_global;              % Global mass matrix
83
84          % Calculate natural frequencies
85          [~, e] = polyeig(Kg, i*Eg, -Mg);
86          frequencies_nat = real(e);
87          frequencies_nat = frequencies_nat(frequencies_nat > 10); % Filter values greater than 10
88
89          % Select the lowest frequency
90          if ~isempty(frequencies_nat)
91              min_frequency = min(frequencies_nat);
92              frequencies(idx) = min_frequency; % Save the lowest frequency
93          end
94      end
95
96      % Display the global lowest frequency
97      if ~isempty(frequencies)
98          global_min_frequency = min(frequencies(frequencies > 0));
99          fprintf('The global lowest frequency is: %.5f Rad/s\n', global_min_frequency);
100     end
101
102     % Plot results
```

```
figure;
plot(N_values, frequencies, '-o', 'LineWidth', 1.5);
xlabel('Number of finite elements (N)');
ylabel('Natural frequency (Rad/s)');
title('Natural frequency vs Number of finite elements');
grid on;
```

CODE 3: Variation of Frequency with Tube Length

```
clc, clear all, close all;
format longeng

% Initial parameters
E = 1.93e11;                % Young's modulus
Dext = 0.0127;              % Outer diameter
Dint = 9.53e-3;             % Inner diameter
rhot = 7900;                % Tube density
rhof = 1000;                % Fluid density
V = 5;                      % Fluid velocity

% Predefined results
et = 1.58500000000000e-003;     % Tube thickness
Rm = 11.1150000000000e-003;     % Mean tube radius
At = 55.3463017162711e-006;     % Cross-sectional area of tube
Mt = 437.235783558542e-003;     % Tube mass per unit length
It = 872.087871085041e-012;     % Second moment of area of tube
Af = 71.3305680581033e-006;     % Flow area section
```

```
Mf = 71.3305680581033e-003;      % Fluid mass per unit length

% Fixed number of finite elements
N = 52; % Number of finite elements
lengths = [0.25 0.35 0.45 0.55 0.65 0.75 0.85]; % Tube lengths
frequencies = zeros(size(lengths)); % Vector to store natural frequencies

% Iterate over different lengths
for idx = 1:length(lengths)
    long = lengths(idx); % Current tube length
    L = long / N;

    % Coefficients
    h1 = (E * It) / (L^3);
    h2 = Mf * V^2;
    h3 = 2 * Mf * V;
    h4 = Mf + Mt;

    % Local matrix functions
    matrixA_local = @(L) [
        12 6*L -12 6*L;
        6*L 4*L^2 -6*L 2*L^2;
        -12 -6*L 12 -6*L;
        6*L 2*L^2 -6*L 4*(L^2)];

    matrixB_local = @(L) [
```

```
    6/5*L 1/10 -6/5*L 1/10;
    1/10 2*L/15 -1/10 -L/30;
    -6/5*L -1/10 6/5*L -1/10;
    1/10 -L/30 -1/10 2*L/15];

matrixC_local = @(L) [
    -1/2 L/10 1/2 -L/10;
    -L/10 0 L/10 -L^2/60;
    -1/2 -L/10 1/2 L/10;
    L/10 L^2/60 -L/10 0];

matrixD_local = @(L) [
    13*L/35 11*L^2/210 9*L/70 -13*L^2/420;
    11*L^2/210 L^3/105 13*L^2/420 -L^3/140;
    9*L/70 13*L^2/420 13*L/35 -11*L^2/210;
    -13*L^2/410 -L^3/140 -11*L^2/210 L^3/105];

% Function to assemble global matrix
function global_matrix = assemble_global_matrix(n, L, local_matrix_fn)
    dof = 2 * (n + 1); % Total degrees of freedom
    global_matrix = zeros(dof, dof); % Initialize global matrix

    for i = 1:n
        indices = 2*(i-1) + (1:4); % Global indices
        local_matrix = local_matrix_fn(L);
        global_matrix(indices, indices) = global_matrix(indices, indices) + local_matrix;
```

```
    end
end

% Global matrix assembly
A_global = assemble_global_matrix(N, L, matrixA_local);
B_global = assemble_global_matrix(N, L, matrixB_local);
C_global = assemble_global_matrix(N, L, matrixC_local);
D_global = assemble_global_matrix(N, L, matrixD_local);

% Global matrix calculation
Kg = h1 * A_global + h2 * B_global; % Global stiffness matrix
Eg = h3 * C_global;                 % Global Coriolis matrix
Mg = h4 * D_global;                 % Global mass matrix

% Natural frequency calculation
[~, e] = polyeig(Kg, i*Eg, -Mg);
Sz = length(e);
Sz = Sz - 5;
frequencies(idx) = real(e(Sz,1)); % Store the lowest natural frequency
end

% Plot results
figure;
plot(lengths, frequencies, '-o', 'LineWidth', 1.5);
xlabel('Tube length (m)');
ylabel('Natural frequency (Hz)');
```

```
97 title('Natural frequency vs Tube length');
98 grid on;
```

CODE 4 : Calculation of Phase Shift and Mass Flow

```
1  clc, clear all
2
3  % II. EIGENVALUE RANGE AND MODE VALUE DEFINITION
4  i = 1:1:5; % ITERATION VARIABLE FOR EIGENVALUE
5  lambda1 = 4.73; % FIRST FUNDAMENTAL MODE [21]-[23]
6  lambda2 = 7.853; % SECOND MODE
7  f21 = 3.399; % FACTOR
8  theta1 = 1.0359; % THETA1
9  theta2 = 0.9984; % THETA2
10 xi11 = -12.74; % FACTOR11
11 xi22 = -45.98; % FACTOR22
12
13 % III. STAINLESS STEEL TUBE INPUT PARAMETERS
14 % INCLUDE DATA ENTRY IN FILE: STEELP0_1
15 E = 1.93e11;       % YOUNG'S MODULUS [N/m^2]
16 Dext = 0.0127; % OUTER DIAMETER [m]
17 Dint = 9.53000000000000e-003; % INNER DIAMETER [m]
18 rhot = 7900; % TUBE DENSITY [kg/m^3]
19 rhof = 1000; % FLUID DENSITY [kg/m^3]
20 L_total = 0.75; % TOTAL TUBE LENGTH [m]
21 N = 1; % NUMBER OF ITERATION ELEMENTS
22 L = L_total / N; % LENGTH OF EACH ELEMENT [m]
```

```
x = 0:0.025:L; % TUBE VARIABLE [m]
V = 10; % FLUID VELOCITY [m/s]

et = (Dext - Dint) / 2; % TUBE THICKNESS [m]
Rm = (Dext + Dint) / 2; % MEAN TUBE RADIUS [m]
At = pi * (Dext^2 - Dint^2) / 4; % TUBE CROSS-SECTIONAL AREA [m^2]
Mt = rhot * At; % TUBE MASS PER UNIT LENGTH [kg/m]
It = pi * (Dext^4 - Dint^4) / 64; % TUBE SECOND MOMENT OF AREA [m^4]
Af = pi * Dint^2 / 4; % FLOW AREA SECTION [m^2]
Mf = rhof * Af; % FLUID MASS PER UNIT LENGTH [kg/m]
omega = 720; % RESONANCE ANGULAR FREQUENCY [rad/s]
C = 1.5e-3; % OSCILLATION AMPLITUDE [m]

% IV. DEFINITION OF OTHER VIBRATION PARAMETERS [21]-[23]
beta1 = lambda1 / L; % FIRST MODE ARGUMENTS
beta2 = lambda2 / L; % SECOND MODE ARGUMENTS
sigma1 = (sinh(beta1*L) - sin(beta1*L)) / (cos(beta1*L) - cosh(beta1*L));
sigma2 = (sinh(beta2*L) - sin(beta2*L)) / (cos(beta2*L) - cosh(beta2*L));
omega1 = (xi11 * Mf * V^2) / (L^2 * theta1 * (Mf + Mt)); % ARGUMENT 1
omega2 = (xi22 * Mf * V^2) / (L^2 * theta2 * (Mf + Mt)); % ARGUMENT 2
gamma1 = sqrt(omega^2 - (xi11 * Mf * V^2) / (L^2 * theta1 * (Mf + Mt))); % SENSOR FREQUENCY 1
gamma2 = sqrt(omega^2 - (xi22 * Mf * V^2) / (L^2 * theta2 * (Mf + Mt))); % SENSOR FREQUENCY 2

% V. CRITICAL VALUES AND FUNCTION DECLARATION
% CHANGES ARE INTRODUCED IN THIS SECTION [21]-[23]
a = L / 4; % CRITICAL POSITION 1
```

```
b = (3 * L) / 4; % CRITICAL POSITION 2
c = L / 2; % CRITICAL POSITION 3
t = pi / (2 * omega); % TIME 1
t1 = (3 * pi) / (8 * omega); % TIME 2
w0 = sinh(beta1*x) - sin(beta1*x) + sigma1*(cosh(beta1*x) - cos(beta1*x)); % ORIGINAL FUNCTION
w1 = sinh(beta2*a) - sin(beta2*a) + sigma2*(cosh(beta2*a) - cos(beta2*a)) * cos(t);
w2 = sinh(beta2*b) - sin(beta2*b) + sigma2*(cosh(beta2*b) - cos(beta2*b)) * cos(t1);
w3 = (cosh(beta2*c) - cos(beta2*c) - sigma1*(sinh(beta2*c) - sin(beta2*c))) * cos(pi/4);
c = C / w3; % MODIFIED AMPLITUDE
A1 = (w1 - w2) / (L / 4); % AMPLITUDE RATIO DEFINITION

% VI. PHASE SHIFT CALCULATION IN RADIANS
PhaseShiftTheory = 0.88 * pi / 180; % PHASE SHIFTS ARE GENERATED FOR VELOCITIES OR FLOWS
PhaseShift = c * ((2 * f21 * gamma1 * Mf * V) / (L * theta2 * (Mf + Mt) * (gamma1^2 - gamma2^2))) * A1;
PhaseError = ((PhaseShiftTheory - PhaseShift) / PhaseShiftTheory) * 100;

% VII. MASS FLOW CALCULATION
z0 = Mf * V; % THEORETICAL FLOW
z = (PhaseShift * L * theta2 * (Mf + Mt) * (gamma1^2 - gamma2^2)) / (2 * A1 * gamma1 * c * f21); % CALCULATED MASS FLOW

% IX. GRAPHS
% Define velocity range
velocities = [0.5, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25];

% Initialize vectors to store results
PhaseShift = zeros(size(velocities));
```

```
mass_flow = zeros(size(velocities));

% Calculate phase shift and mass flow for each velocity
for i = 1:length(velocities)
    V = velocities(i); % Current velocity
    omega1 = (xi11 * Mf * V^2) / (L^2 * theta1 * (Mf + Mt)); % ARGUMENT 1
    omega2 = (xi22 * Mf * V^2) / (L^2 * theta2 * (Mf + Mt)); % ARGUMENT 2
    gamma1 = sqrt(omega^2 - (xi11 * Mf * V^2) / (L^2 * theta1 * (Mf + Mt))); % SENSOR FREQUENCY 1
    gamma2 = sqrt(omega^2 - (xi22 * Mf * V^2) / (L^2 * theta2 * (Mf + Mt))); % SENSOR FREQUENCY 2

    % Calculate phase shift
    PhaseShift(i) = c * ((2 * f21 * gamma1 * Mf * V) / (L * theta2 * (Mf + Mt) * (gamma1^2 - gamma2^2))) * A1;

    % Calculate mass flow
    mass_flow(i) = (PhaseShift(i) * L * theta2 * (Mf + Mt) * (gamma1^2 - gamma2^2)) / (2 * A1 * gamma1 * c * f21);
end

% a) CORIOLIS FLOW VS VELOCITY PLOT
figure(1)
plot(velocities, mass_flow, 'ro')
title('CORIOLIS FLOW vs VELOCITY')
xlabel('Fluid velocity [m/s]')
ylabel('Mass flow [kg/s]')
grid on

% b) PHASE SHIFT VS VELOCITY PLOT
```

```
figure(2)
plot(velocities, PhaseShift, 'go')
title('PHASE SHIFT vs VELOCITY')
xlabel('Fluid velocity [m/s]')
ylabel('Phase angle shift [rad]')
grid on
```

CODE 5 : Calculation of global matrices A, B, C and D

```
import numpy as np

def local_stiffness_matrix_A(L):
    """
    Defines the local stiffness matrix [A] for a finite element.
    """
    return np.array([
        [12, 6*L, -12, 6*L],
        [6*L, 4*L**2, -6*L, 2*L**2],
        [-12, -6*L, 12, -6*L],
        [6*L, 2*L**2, -6*L, 4*L**2]
    ])

def local_centrifugal_matrix_B(L):
    """
    Defines the local centrifugal force matrix [B] for a finite element.
    """
    return np.array([
```

```
        [6/(5*L), 1/10, -6/(5*L), 1/10],
        [1/10, 2*L/15, -1/10, -L/30],
        [-6/(5*L), -1/10, 6/(5*L), -1/10],
        [1/10, -L/30, -1/10, 2*L/15]
    ])

def local_coriolis_matrix_C(L):
    """
    Defines the local Coriolis matrix [C] for a finite element.
    """
    return np.array([
        [-0.5, L/10, 0.5, -L/10],
        [-L/10, 0, L/10, -L**2/60],
        [-0.5, -L/10, 0.5, L/10],
        [L/10, L**2/60, -L/10, 0]
    ])

def local_mass_matrix_D(L):
    """
    Defines the local mass matrix [D] for a finite element.
    """
    return np.array([
        [13*L/35, 11*L**2/210, 9*L/70, -13*L**2/420],
        [11*L**2/210, L**3/105, 13*L**2/420, -L**3/140],
        [9*L/70, 13*L**2/420, 13*L/35, -11*L**2/210],
        [-13*L**2/420, -L**3/140, -11*L**2/210, L**3/105]
```

```
45    ])
46
47 def assemble_global_matrix(n, L, local_matrix_fn):
48    """
49    Assembles a global matrix for n finite elements of length L using a local matrix function.
50
51    Parameters:
52    n (int): Number of finite elements.
53    L (float): Length of each finite element.
54    local_matrix_fn (function): Function that generates the local matrix.
55
56    Returns:
57    numpy.ndarray: Global matrix of size 2(n) + 2.
58    """
59    # Size of global matrix
60    global_size = 2 * n + 2
61    global_matrix = np.zeros((global_size, global_size))
62
63    # Local matrix
64    local_matrix = local_matrix_fn(L)
65
66    # Global matrix assembly
67    for i in range(n):
68        start = 2 * i
69        end = start + 4
70        global_matrix[start:end, start:end] += local_matrix
```

```
71
72     return global_matrix
73
74 n_elements = 12  # Number of finite elements
75 element_length = 0.75   # Length of each finite element
76
77 # Global matrix [A]
78 global_matrix_A = assemble_global_matrix(n_elements, element_length, local_stiffness_matrix_A)
79 print("Global matrix [A]:")
80 print(global_matrix_A)
81
82 # Global matrix [B]
83 global_matrix_B = assemble_global_matrix(n_elements, element_length, local_centrifugal_matrix_B)
84 print("Global matrix [B]:")
85 print(global_matrix_B)
86
87 # Global matrix [C]
88 global_matrix_C = assemble_global_matrix(n_elements, element_length, local_coriolis_matrix_C)
89 print("Global matrix [C]:")
90 print(global_matrix_C)
91
92 # Global matrix [D]
93 global_matrix_D = assemble_global_matrix(n_elements, element_length, local_mass_matrix_D)
94 print("Global matrix [D]:")
95 print(global_matrix_D)
```