

Table of Contents

1. Introduction.....	2
1.1. CAE: Computer-Aided Engineering.....	2
1.2. The Finite Element Method.....	3
1.2.1. The basic features.....	3
1.2.2. Some remarks.....	4
1.3. Fluid-Structure Interaction	6
1.4. Scope of this thesis.....	6
1.5. References for Chapter1	6
2. Computational Structural Dynamics	7
2.1. Plane stress problem formulation.....	7
2.1.1. Development of weak form.....	8
2.1.2. Finite Element Model	9
2.1.3. Eigenvalue and Transient Problems	11
2.1.4. Computation of element matrices	12
2.2. Numerical examples	14
2.2.1. Thin plate structure under static load	14
2.2.2. Modal Analysis of a multihole panel	26
2.2.3. Transient Analysis of a multihole panel	33
2.3. References for Chapter2	38
3. Computational Fluid Dynamics.....	39
3.1. Navier-Stokes strong formulation	39
3.2. Variational multiscale stabilization of the weak form.....	40
3.3. Spatial Discretization	41
3.4. Time Discretization	43
3.5. Vortex shedding.....	45
3.5.1. Governing Equation	45
3.6. Numerical Example.....	46
3.6.1. Preprocessing with GiD	47

1. Introduction

1.1. CAE: Computer-Aided Engineering

CAE, or Computer-Aided Engineering, is a term used to describe the entire product engineering procedure, from design and virtual testing with complex analytical algorithms to manufacturing planning. Computer-aided engineering is prevalent in nearly every industry that uses design software to create products. CAE is the next step not only in designing a product but also in supporting the engineering process, as it enables tests and simulations of the product's physical properties to be conducted without the need for a physical prototype. Finite Element Analysis, Computational Fluid Dynamics, Thermal Analysis, Multibody Dynamics, and Optimizations are the most frequently employed simulation analysis types in the context of CAE.

Utilizing the benefits of engineering simulation, particularly when combined with the power and speed of high-performance cloud computing, it is possible to significantly reduce the cost and duration of each design iteration cycle, as well as the duration of the overall development process. The standard CAE workflow begins with the generation of an initial design, followed by the simulation of the CAD geometry. The simulation results are then evaluated and applied to the design improvement process. This procedure is repeated until all product specifications are met and virtually confirmed. In case of any weak spots or areas where the performance of the digital prototype does not meet expectations, engineers and designers can improve the CAD model and test the effects of their changes by simulating the revised design. This process expedites product development because physical prototypes are not required in the early stages of development.

Simulating with CAE methods requires no more than a few hours, whereas building a physical prototype could take days or even weeks. Since it is necessary to construct a physical prototype of a product prior to beginning serial production, simulation can help reduce the number of prototypes. When planning to integrate simulation techniques into the product development process, it is essential to understand the product's environment (forces, temperatures, etc.). Understanding these conditions is essential for properly configuring a simulation. Any simulation's predictive value is limited to the precision of its boundary conditions. In addition to predicting environmental factors, engineering simulation has historically been a complex endeavor, reserved primarily for experienced engineers and simulation specialists. Modern CAE simulation tools, attempt to overcome these limitations by enabling inexperienced users with limited understanding of physical processes and solver characteristics to generate insightful simulation results.

Even with modern computers, it is difficult to simulate complex geometries due to the high computing power required. Large corporations with an advanced IT infrastructure host and run simulations on their own servers. However, the rise of HPC (High Performance Computing) now gives smaller companies access to the same simulation tools and capabilities that were previously only available to a select few. This disruption in the market for simulation products has made it possible for all designers to simulate their products.

1.2. The Finite Element Method

The finite element method is a powerful and general numerical method that can be applied to real-world problems involving multi-physics, complex geometry, and boundary conditions. In the finite element method, a given domain is viewed as a collection of subdomains, and the governing equation is approximated over each subdomain using one of the traditional variational methods or any other suitable method. It is easier to represent a complicated function as a collection of simple polynomials, which is the primary motivation for seeking approximate solutions on a collection of subdomains. Obviously, each segment of the solution must be compatible with its neighbors in the sense that the function and, if applicable, its derivatives up to a specified order are continuous (single-valued) at the connecting points.

1.2.1. The basic features

The superiority of the finite element method over other competing techniques can be attributed to three unique characteristics. Detailed below are these characteristics.

- A domain of the problem (Ω) that is geometrically complex is represented as a mesh of subdomains with simple geometry. Subdomains are referred to as finite elements. In this instance, the term "domain" refers to the geometric region over which the equations are solved. Note that not all geometric shapes qualify as finite elements; only those that allow the derivation of approximation functions qualify as finite elements. Actually, the discretized domain is a collection of points.
- Over each finite element, algebraic relations between the values of the duality pairs (i.e., cause and effect or primary and secondary degrees of freedom) of the problem at element nodes are generated using (a) statements corresponding to the problem's governing equations and (b) an approximation method. Alternately, one may obtain the correlations using physical principles directly. In theory, any appropriate method of approximation can be used to derive the algebraic relations. The resultant set of algebraic equations between the nodal values of the duality pairs (e.g. displacements and forces) is referred to as a finite element model.
- The equations from all components, are constructed (i.e., elements are returned to their original positions on the mesh) utilizing (a) continuity of the primary variables (e.g. displacements) and (b) balance of the secondary variables (e.g., forces)

Multiple stages of engineering analysis incorporate approximations. The split of the entire domain into finite elements may not be precise (i.e., the collection of elements, where N is the number of elements, may not perfectly match the original domain), hence introducing inaccuracy into the domain (and, consequently, the boundary data) being modeled.

The second phase involves the derivation of element equations. Generally, the dependent unknowns (\mathbf{u}) of a problem are estimated utilizing the fundamental concept that any continuous function can be represented by a linear combination of known functions (ϕ_i) and indeterminate coefficients (c_i). By solving the governing equations in a weighted-integral

sense across every element, algebraic connections among the indeterminate coefficients c_i are produced. The approximation functions are frequently assumed to be polynomials and are generated using interpolation theory concepts. Consequently, these are also known as interpolation functions. Thus, approximation errors are introduced in the second stage during both the representation of the solution u and the evaluation of the integrals.

Finally, errors are introduced during the system of equations' solution. Clearly, some of the errors described above can be zero. When all errors are zero, we have the exact solution to the problem (which is not the case for the majority of two- and three-dimensional problems).

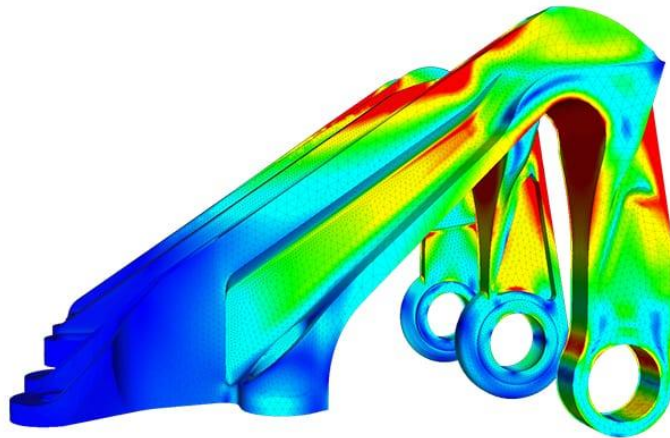


Figure 1: Finite element analysis of an aircraft's bearing bracket

1.2.2. Some remarks

In summary, the finite element method divides a given domain into subdomains called finite elements and develops an approximation of the solution over each element. The partition of an entire domain into subsections permits simple representation of the whole solution by functions defined within each element that capture local effects (such as high gradients in the solution).

Following are the three fundamental steps of the finite element method:

1. To represent both the geometry and the solution to the problem, subdivide the entire domain.
2. Seek an approximation of the solution as a linear combination of nodal values and approximation functions over each element, and establish the algebraic relations between the nodal values of the solution.
3. Assemble the elements and solve for the solution in the whole domain.

Although the above steps are the core of the finite element method, there are several features that need to be mentioned:

- Depending on the shape of the domain, its geometry can be discretized into a mesh of more than one type of element (by shape or order). For instance, one can approximate an irregular domain using a combination of rectangles and triangles. Nonetheless, the element interfaces must be compatible in that the solution must be uniquely defined along the interface.
- If more than one type of element is employed in the domain's representation, one of each type must be isolated and its equations developed.
- In engineering problems, the governing equations are differential equations. Two reasons prevent most equations from being solved over an element. First, they prohibit the exact solution. Here is where variational methods become relevant. Second, the discrete equations derived from variational methods cannot be solved independently of the remaining elements because the ensemble of elements is subject to continuity, boundary, and/or initial conditions.
- In general, the assembly of elements is based on the assumption that the solution (and possibly its derivatives in higher-order equations) is continuous at the boundaries.
- In general, the collection of finite elements is subject to initial and/or boundary conditions. After the boundary and/or initial conditions have been imposed, the discrete equations associated with the finite element mesh are solved.
- There are three sources of error in a finite element solution: (a) those resulting from the approximation of the domain, (b) those resulting from the approximation of the solution, and (c) those resulting from numerical computation (e.g., numerical integration and round-off errors in a computer). The estimation of these errors is typically not straightforward. However, they can be estimated for classes of elements and problems under specific conditions.
- The accuracy and convergence of the finite element solution are dependent on the differential equation, its weighted integral form, and the element employed. "Accuracy" refers to the difference between the exact solution and the finite element solution, whereas "convergence" refers to the accuracy as the number of mesh elements increases.
- Usually, a two-step formulation is followed for time-dependent problems. In the first step, the differential equations are approximated using the finite element method in order to produce a set of ordinary differential equations in time. In the second step, the differential equations in time are solved exactly or approximatively using variational or finite difference methods in order to obtain algebraic equations, which are then solved for the nodal values. Alternately, the finite element method may be utilized at both approximation stages.
- Today's desktop computers are more powerful than the supercomputers that were available when the finite element method was first implemented. Therefore, the analysis time is drastically reduced, assuming the mesh used to model the issue is

adequate. Even automatic mesh generation programs cannot guarantee meshes that are free of irregularly shaped elements and have a sufficient number of elements in regions with high gradients in the solution, both of which lead to a loss of accuracy or, in the case of nonlinear problems, non-convergence of solutions.

1.3. Fluid-Structure Interaction

Fluid-Structure interaction (FSI) is a multiphysics problem and a huge research field. It has major impact in analyses like aerodynamic flutter of aircrafts, simulation of cardiac and blood related systems, flows with discrete particles and many more.

The above mentioned problems are characterized by coexistence of a structure and a fluid field, which interact with each other in the time and spatial domains. This interaction is taken into account in the cases that it significantly changes the physical and engineering properties that describe each of the two fields. Consequently, there is a need for a coupled solution of the equations that describe the two fields.

There are numerous numerical methods that can be used to solve FSI problems. These

1.4. Scope of this thesis

1.5. References for Chapter1

2. Computational Structural Dynamics

This chapter is devoted to giving an introduction on plane stress theory and presenting some numerical examples from selected structural problems. These problems are solved within cane Multiphysics, a framework for numerical simulations in engineering that uses the Finite Element Method. It's input is made externally, using GiD pre-processing software [1] to describe the parameters of the problem, whereas the output files of cane can be postprocessed in Paraview [2], an open source post-processing software. This framework is capable of performing many different analyses that need to be performed in order to accurately model physical phenomena that occur in engineering problems. Such analyses are:

- Computational Structural Dynamics analyses (CSD)
- Computational Fluid Dynamics analyses (CFD)
- Computational Fluid-Structure Interaction analyses (FSI)
- Contact Mechanics analyses
- Thermal Conduction analyses
- Isogeometric analyses (IGA)

The problems solved in this chapter are carefully selected from bibliography in order for the reader to be able to compare results between cane and the results from the solution of the problem in the selected references.

2.1. Plane stress problem formulation

A state of plane stress is defined as one in which the following stress field exists:

$$\sigma_{xx} = \sigma_{yz} = \sigma_{zz} = 0 \quad (2.1)$$

$$\sigma_{xx} = \sigma_{xx}(x, y), \sigma_{xy} = \sigma_{xy}(x, y), \sigma_{yy} = \sigma_{yy}(x, y) \quad (2.2)$$

An example of a plane stress problem is provided by a thin plate under external loads applied in the xy plane (or parallel to it) that are independent of z . The top and bottom surfaces of the plate are assumed to be traction-free, and the specified boundary forces are in the xy plane so that $\mathbf{f}_z = \mathbf{0}$ and $\mathbf{u}_z = \mathbf{0}$.

The strain field associated with the stress field in is:

$$\begin{Bmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ 2\epsilon_{xy} \end{Bmatrix} = \begin{bmatrix} s_{11} & s_{12} & 0 \\ s_{12} & s_{22} & 0 \\ 0 & 0 & s_{66} \end{bmatrix} \begin{Bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{Bmatrix} \quad (2.3)$$

where s_{ij} are the elastic compliances:

$$\begin{aligned} s_{11} &= \frac{1}{E_1}, \quad s_{22} = \frac{1}{E_2}, \quad s_{33} = \frac{1}{E_3} \\ s_{12} &= -\nu_{21}s_{22} = -\nu_{12}s_{22} = -\nu_{12}s_{11}, \quad s_{66} = \frac{1}{G_{12}} \\ s_{13} &= -\nu_{31}s_{33} = -\nu_{13}s_{11}, \quad s_{23} = -\nu_{32}s_{33} = -\nu_{23}s_{22} \end{aligned} \quad (2.4)$$

where E_1 and E_2 are principal (Young's) moduli in the x and y directions, respectively, G_{12} the shear modulus in the xy plane, and ν_{12} the Poisson ratio.

2.1.1. Development of weak form

This section focuses on the development of the weak form of the plane stress equations. We employ the principle of virtual displacements as expressed by matrices relating displacements to strains, strains to stresses, and the equations of motion.

Let Ω_e be a plane elastic finite elements with volume $V_e = h_e \Omega_e$.

$$0 = \int_{V_e} (\sigma_{ij} \delta \epsilon_{ij} + \rho \ddot{u}_i \delta u_i) dV - \int_{V_e} f_i \delta u_i dV - \oint_{S_e} \hat{t}_i \delta u_i ds \quad (2.5)$$

where S_e is the surface of the volume element V_e and h_e is the thickness of the finite element Ω_e , δ denotes the variational operator σ_{ij} and ϵ_{ij} are the components of stress and strain tensors, respectively and f_i and t_i are the components of the body force and boundary stress vectors, respectively. The correspondence between the (x, y) components and (x_1, x_2) components of the stress and strain tensors is given by:

$$\begin{aligned} \sigma_{11} &= \sigma_{xx}, \sigma_{12} = \sigma_{xy}, \sigma_{22} = \sigma_{yy}, \epsilon_{11} = \epsilon_{xx}, \epsilon_{12} = \epsilon_{xy}, \epsilon_{22} = \epsilon_{yy} \\ u_1 &= u_x, u_2 = u_y, f_1 = f_x, f_2 = f_y, t_1 = t_x, t_2 = t_y \end{aligned} \quad (2.6)$$

The first term in the equation above represents the virtual strain energy stored in the body, the second term represents the kinetic energy stored in the body, the third term represents the virtual work performed by the body forces, and the fourth term represents the virtual work performed by the surface tractions. In accordance with the assumptions of plane elasticity, we assume that all quantities are independent of the z -coordinate thickness. Therefore, we have:

$$0 = \int_{\Omega_e} h_e [\sigma_{xx} \delta \varepsilon_{xx} + \sigma_{yy} \delta \varepsilon_{yy} + 2\sigma_{xy} \delta \varepsilon_{xy} + \rho(\ddot{u}_x \delta u_x + \ddot{u}_y \delta u_y)] dx dy - \int_{\Omega_e} h_e (f_x \delta u_x + f_y \delta u_y) dx dy - \oint_{\Gamma_e} h_e (\hat{t}_x \delta u_x + \hat{t}_y \delta u_y) dS \quad (2.7)$$

wherein, f_x and f_y are the body forces per unit area, while, t_x and t_y are the boundary forces per unit length. When stresses are expressed as strains and strains as displacements, the resulting equation has the form:

$$0 = \int_{\Omega_e} h_e [(\mathbf{D}\delta\mathbf{u})^T \mathbf{C}(\mathbf{D}\mathbf{u}) + \rho(\delta\mathbf{u})^T \ddot{\mathbf{u}}] dx dy - \int_{\Omega_e} (\delta\mathbf{u})^T h_e \mathbf{f} dx dy - \oint_{\Gamma_e} h_e (\delta\mathbf{u})^T \mathbf{t} dS \quad (2.8)$$

2.1.2. Finite Element Model

In this section, we develop the vector form of the finite element model of the plane elasticity equations. Examination of the weak form reveals, u_x and u_y are the primary variables that must be carried as the key nodal degrees of freedom. Moreover, only the first derivatives of, u_x and u_y are considered. The weak forms with respect to x and y are represented. In conclusion, u_x and u_y must be approximated by a Lagrange family interpolation function that is at least linear. These conditions are satisfied by the linear triangular and linear quadrilateral elements, which are the simplest elements. Despite the fact that, u_x and u_y independent of one another, they constitute the displacement vector's components. Therefore, both components must be approximated using the same interpolation type and degree. Let, u_x and u_y be approximated by the interpolations of finite elements:

$$u_x \approx \sum_{j=1}^n u_x^j \psi_j(x, y), u_y \approx \sum_{j=1}^n u_y^j \psi_j(x, y) \quad (2.9)$$

Currently, we will not restrict ψ_j to a particular element, so that the to-be-developed finite element formulation is valid for any admissible element. Using a linear triangular element ($n = 3$) as an example, there are two u_x and u_y degrees of freedom per node and six nodal displacements per element. There are eight nodal displacements per element for a linear quadrilateral element with n equal to four. Since the first derivatives of ψ_i for a triangular element are element-wise constants, all strains computed for the linear triangular element ($\varepsilon_{xx}, \varepsilon_{yy}, \varepsilon_{xy}$) are also element-wise constants. For plane elasticity problems, the linear triangular element is known as the constant-strain triangular (CST) element. The first derivatives ψ_i of an element of a quadrilateral are not constant: $\partial\psi_i/\partial\xi$ is linear in η and constant in ξ , while $\partial\psi_i/\partial\eta$ is linear in ξ and constant in η .

The finite element approximation in Eq. (2.9) can be expressed in vector form as:

$$\mathbf{u} = \begin{Bmatrix} u_x \\ u_y \end{Bmatrix} = \mathbf{\Psi}\Delta, \mathbf{w} = \delta\mathbf{u} = \begin{Bmatrix} w_1 = \delta u_x \\ w_2 = \delta u_y \end{Bmatrix} = \mathbf{\Psi}\delta\Delta \quad (2.10)$$

where Ψ is a $2 \times 2n$ matrix and Δ is a $2n \times 1$ vector of nodal degrees of freedom:

$$\begin{aligned}\Psi &= \begin{bmatrix} \psi_1 & 0 & \psi_2 & 0 & \dots & \psi_n & 0 \\ 0 & \psi_1 & 0 & \psi_2 & \dots & 0 & \psi_n \end{bmatrix} \\ \Delta &= \{u_x^1 \quad u_y^1 \quad u_x^2 \quad u_y^2 \quad \dots \quad u_x^n \quad u_y^n\}^T\end{aligned}\quad (2.11)$$

The strains are:

$$\varepsilon = \mathbf{D}\mathbf{u} = \mathbf{D}\Psi\Delta \equiv \mathbf{B}\Delta, \sigma = \mathbf{C}\mathbf{B}\Delta \quad (2.12)$$

where \mathbf{D} is:

$$\mathbf{D} = \begin{bmatrix} \partial/\partial x & 0 \\ 0 & \partial/\partial y \\ \partial/\partial y & \partial/\partial x \end{bmatrix} \quad (2.13)$$

and \mathbf{B} is a $3 \times 2n$ matrix:

$$\mathbf{B} = \mathbf{D}\Psi = \begin{bmatrix} \frac{\partial\psi_1}{\partial x} & 0 & \frac{\partial\psi_2}{\partial x} & 0 & \dots & \frac{\partial\psi_n}{\partial x} & 0 \\ 0 & \frac{\partial\psi_1}{\partial y} & 0 & \frac{\partial\psi_2}{\partial y} & \dots & 0 & \frac{\partial\psi_n}{\partial y} \\ \frac{\partial\psi_1}{\partial y} & \frac{\partial\psi_1}{\partial x} & \frac{\partial\psi_2}{\partial y} & \frac{\partial\psi_2}{\partial x} & \dots & \frac{\partial\psi_n}{\partial y} & \frac{\partial\psi_n}{\partial x} \end{bmatrix} \quad (2.14)$$

To obtain the vector form of the finite element model, we substitute the finite element expansion into the virtual work statement and obtain:

$$\begin{aligned}0 &= \int_{\Omega_e} h_e (\delta\Delta^e)^T (\mathbf{B}^T \mathbf{C} \mathbf{B} \Delta^e + \rho \Psi^T \Psi \ddot{\Delta}^e) dx dy \\ &\quad - \int_{\Omega_e} h_e (\delta\Delta^e)^T \Psi^T \mathbf{f} dx dy - \oint_{\Gamma_e} h_e (\delta\Delta^e)^T \Psi^T \mathbf{t} dS \\ &= (\delta\Delta^e)^T (\mathbf{K}^e \Delta^e + \mathbf{M}^e \ddot{\Delta}^e - \mathbf{f}^e - \mathbf{Q}^e)\end{aligned}\quad (2.15)$$

Since the above equation holds for any arbitrary variations $\delta\Delta$, it follows (from the Fundamental Lemma of variational calculus) that the coefficient of $\delta\Delta$ in the expression should be identically zero, giving the result:

$$\mathbf{M}^e \ddot{\Delta}^e + \mathbf{K}^e \Delta^e = \mathbf{f}^e + \mathbf{Q}^e \quad (2.16)$$

where,

$$\begin{aligned} \mathbf{K}^e &= \int_{\Omega_e} h_e \mathbf{B}^T \mathbf{C} \mathbf{B} dx dy, \mathbf{M}^e = \int_{\Omega_e} \rho h_e \boldsymbol{\Psi}^T \boldsymbol{\Psi} dx dy \\ \mathbf{f}^e &= \int_{\Omega_e} h_e \boldsymbol{\Psi}^T \mathbf{f} dx dy, \quad \mathbf{Q}^e = \oint_{\Gamma_e} h_e \boldsymbol{\Psi}^T \mathbf{t} dS \end{aligned} \quad (2.17)$$

The element mass matrix \mathbf{M}^e and stiffness matrix \mathbf{K}^e are of order $2n \times 2n$ and the element body-force vector \mathbf{f}^e and the vector of internal forces \mathbf{Q}^e is of order $2n \times 1$, where n is the number of nodes in a Lagrange finite element (a triangle or quadrilateral).

2.1.3. Eigenvalue and Transient Problems

For natural vibration study of plane elastic bodies, we seek a periodic solution of the form:

$$\Delta = \Delta_0 e^{-i\omega t} (i = \sqrt{-1}) \quad (2.18)$$

where ω is the frequency of natural vibration. Then the system's equation reduces to an eigenvalue problem:

$$(-\omega^2 \mathbf{M}^e + \mathbf{K}^e) \Delta_0^e = \mathbf{Q}^e \quad (2.19)$$

For transient analysis, using a time-approximation like the famous Newmark integration scheme, equations can be reduced to the following system of algebraic equations:

$$\hat{\mathbf{K}}^{s+1} \Delta^{s+1} = \hat{\mathbf{F}}^{s,s+1} \quad (2.20)$$

Where

$$\begin{aligned} \hat{\mathbf{K}}^{s+1} &= \mathbf{K}^{s+1} + a_3 \mathbf{M}^{s+1} \\ \hat{\mathbf{F}}^{s,s+1} &= \mathbf{F}^{s+1} + \mathbf{M}^{s+1} (a_3 \Delta^s + a_4 \dot{\Delta}^s + a_5 \ddot{\Delta}^s) \\ a_3 &= \frac{2}{\gamma(\Delta t)^2}, a_4 = \Delta t a_3, a_5 = \frac{1}{\gamma} - 1 \end{aligned} \quad (2.21)$$

where γ is the parameter in the (α, γ) -family of approximation.

2.1.4. Computation of element matrices

For the linear triangular (i.e., **CST**) element, the ψ_{ie} and its derivatives are given by:

$$\psi_i^e = \frac{1}{2A_e}(\alpha_i^e + \beta_i^e x + \gamma_i^e y), \quad \frac{\partial \psi_i^e}{\partial x} = \frac{\beta_i^e}{2A_e}, \quad \frac{\partial \psi_i^e}{\partial y} = \frac{\gamma_i^e}{2A_e} \quad (2.22)$$

$$\mathbf{B}^e = \frac{1}{2A_e} \begin{bmatrix} \beta_1^e & 0 & \beta_2^e & 0 & \cdots & \beta_n^e & 0 \\ 0 & \gamma_1^e & 0 & \gamma_2^e & \cdots & 0 & \gamma_n^e \\ \gamma_1^e & \beta_1^e & \gamma_2^e & \beta_2^e & \cdots & \gamma_n^e & \beta_n^e \end{bmatrix}_{(3 \times 2n)} \quad (2.23)$$

where A_e is the area of the triangular element. Since \mathbf{B}^e and \mathbf{C}^e are independent of \mathbf{x} and \mathbf{y} , the element stiffness matrix for the **CST** element is given by:

$$\mathbf{K}^e = h_e A_e (\mathbf{B}^e)^T \mathbf{C}^e \mathbf{B}^e \quad (2n \times 2n) \quad (2.24)$$

For the case in which the body force components \mathbf{f}_x and \mathbf{f}_y are element-wise constant (say, equal to \mathbf{f}_{x0}^e and \mathbf{f}_{y0}^e respectively), the load vector \mathbf{F}^e has the form:

$$\mathbf{f}^e = \int_{\Omega_e} h_e (\mathbf{\Psi}^e)^T \mathbf{f}_0^e dx dy = \frac{A_e h_e}{3} \begin{Bmatrix} f_{x0}^e \\ f_{y0}^e \\ f_{x0}^e \\ f_{y0}^e \\ f_{x0}^e \\ f_{y0}^e \end{Bmatrix}_{(6 \times 1)} \quad (2.25)$$

For a general quadrilateral element, it is difficult to manually compute the stiffness matrix coefficients. In such cases, numerical integration over the element is used. However, for a linear rectangular element with sides \mathbf{a} and \mathbf{b} , the stiffness can be obtained using the element coefficient matrices. Specifically, the submatrices are given by:

$$\mathbf{M}^{11} = \mathbf{M}^{22} = \frac{\rho h a b}{36} \begin{bmatrix} 4 & 2 & 1 & 2 \\ 2 & 4 & 2 & 1 \\ 1 & 2 & 4 & 2 \\ 2 & 1 & 2 & 4 \end{bmatrix} \quad (2.26)$$

$$\mathbf{K}^{11} = hc_{11} \frac{b}{6a} \begin{bmatrix} 2 & -2 & -1 & 1 \\ -2 & 2 & 1 & -1 \\ -1 & 1 & 2 & -2 \\ 1 & -1 & -2 & 2 \end{bmatrix} + hc_{66} \frac{a}{6b} \begin{bmatrix} 2 & 1 & -1 & -2 \\ 1 & 2 & -2 & -1 \\ -1 & -2 & 2 & 1 \\ -2 & -1 & 1 & 2 \end{bmatrix} \quad (2.27)$$

$$\mathbf{K}^{12} = \frac{h}{4} \left(c_{12} \begin{bmatrix} 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & -1 \end{bmatrix} + c_{66} \begin{bmatrix} 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 \\ -1 & 1 & 1 & -1 \\ -1 & 1 & 1 & -1 \end{bmatrix} \right) \quad (2.28)$$

$$\mathbf{K}^{11} = hc_{11} \frac{b}{6a} \begin{bmatrix} 2 & -2 & -1 & 1 \\ -2 & 2 & 1 & -1 \\ -1 & 1 & 2 & -2 \\ 1 & -1 & -2 & 2 \end{bmatrix} + hc_{66} \frac{a}{6b} \begin{bmatrix} 2 & 1 & -1 & -2 \\ 1 & 2 & -2 & -1 \\ -1 & -2 & 2 & 1 \\ -2 & -1 & 1 & 2 \end{bmatrix} \quad (2.29)$$

For a linear quadrilateral element with constant body force components (f_{x0}, f_{y0}) , the load vector is given by

$$\mathbf{f}^e = \frac{A_e h_e}{4} \begin{Bmatrix} f_{x0}^e \\ f_{y0}^e \\ f_{x0}^e \\ f_{y0}^e \\ \vdots \end{Bmatrix}_{(8 \times 1)} \quad (2.30)$$

Let \mathbf{Q}^e denote the element load vector referred to the element coordinates. Then the corresponding load vector referred to the global coordinates is given by

$$\mathbf{F}^e = \mathbf{R} \mathbf{Q}^e \quad (2.31)$$

where \mathbf{R} is the transformation matrix

$$\mathbf{R}^e = \begin{bmatrix} \cos \alpha & \sin \alpha & 0 & 0 \\ -\sin \alpha & \cos \alpha & 0 & 0 \\ 0 & 0 & \cos \alpha & \sin \alpha \\ 0 & 0 & -\sin \alpha & \cos \alpha \\ & & & \ddots \end{bmatrix}_{2n \times 2n} \quad (2.32)$$

and α is the angle between the global x -axis and the traction vector \mathbf{t} .

2.2. Numerical examples

2.2.1. Thin plate structure under static load

In this problem, a uniformly distributed load is subjected to a thin plate structure as shown in the figure below. The plate is discretized using two linear triangular elements for illustration purposes. More elements must be used in order to obtain reliable results. This problem can be found in [3].

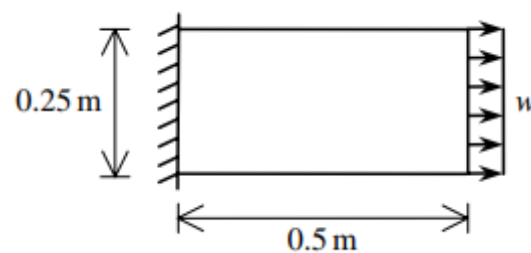


Figure 1: Thin plate structure under uniform load

The abovementioned problem will be solved using the following parameters:

- Modulus of Elasticity: $E = 210 \text{ GPa}$
- Poisson's ratio: $\nu = 0.3$
- Plate thickness: $t = 0.25 \text{ m}$
- Distributed load: $w = 3000 \text{ kN} / \text{m}^2$

Since the thickness is relatively small compared to the other dimensions of the plate, we can assume plane stress state for the static analysis.

2.2.1.1. Preprocessing with GiD

Firstly, the user must specify the MATLAB GiD problem type. Select Data -> Problem Type -> matlab.

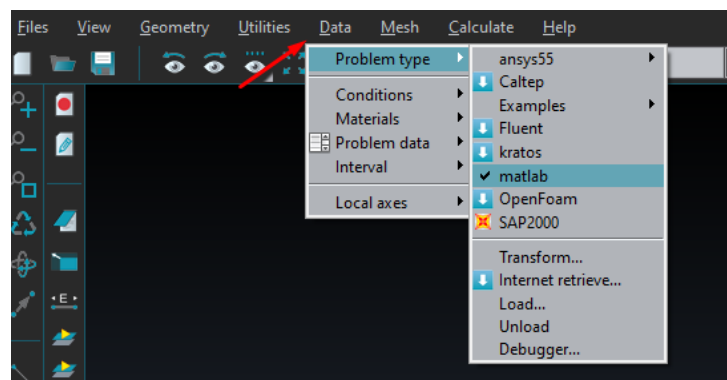


Figure 2: GiD – Select problem type

At this point the user should save the project with a name of his choice. Select Files->Save and type the file name.

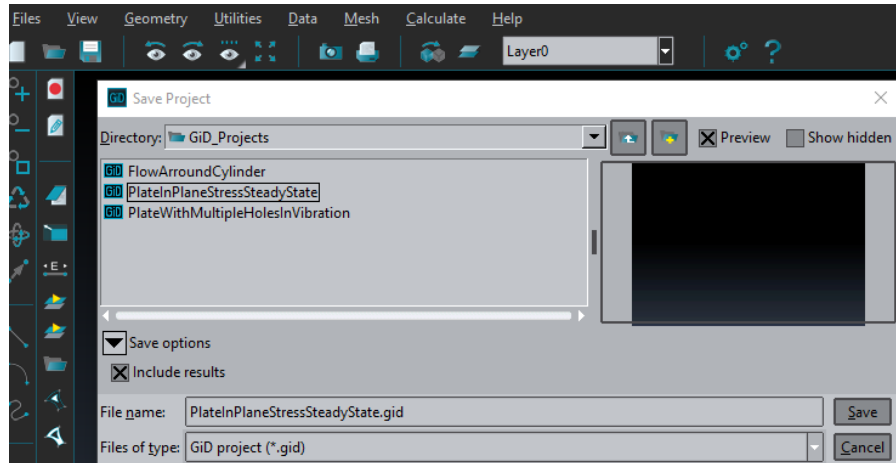


Figure 3: GiD – Save Project

Geometry Setup

To create a geometry, select create object → rectangle. The rectangle can be drawn by clicking on the drawing plane or by specifying the coordinates of its corner edges (Fig. 1). The coordinates must be typed in the format x y z. They must contain white space between each coordinate, whereas omitting coordinates are assumed by default to be zero. Enter the first point (0, 0) in the command line and confirm with esc. Now enter the second point (0.5, 0.25).

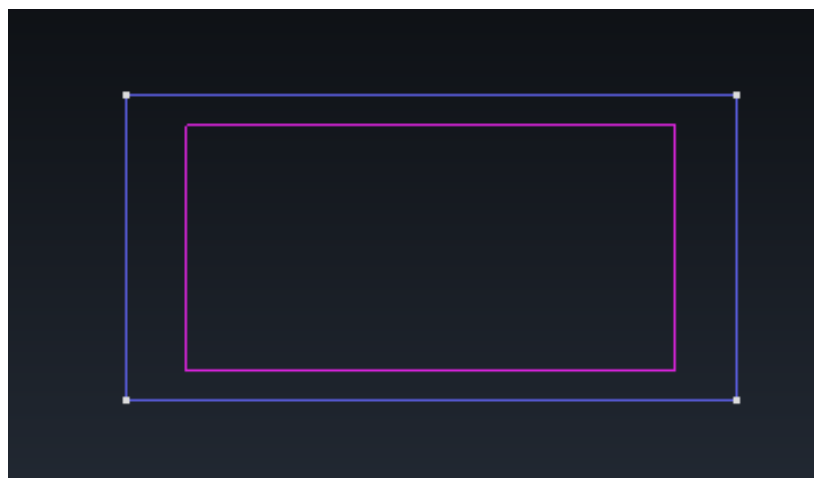


Figure 4: GiD – Unmeshed geometry

Boundary Conditions

To specify the Dirichlet boundary conditions select Data → Conditions → Constraints. Select lines (line icon) as selection type and select Structure-Dirichlet.

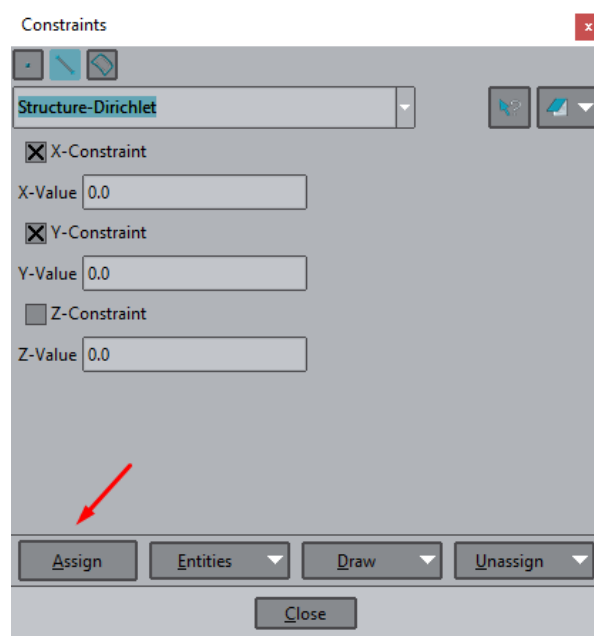


Figure 5: GiD – Specify Dirichlet Boundary Conditions

Then click on the Assign button and select the appropriate side to apply the boundary condition. Here, both translations on x and y are so we choose the value 0.

To assign the constant horizontal load select Data → Conditions → Loads

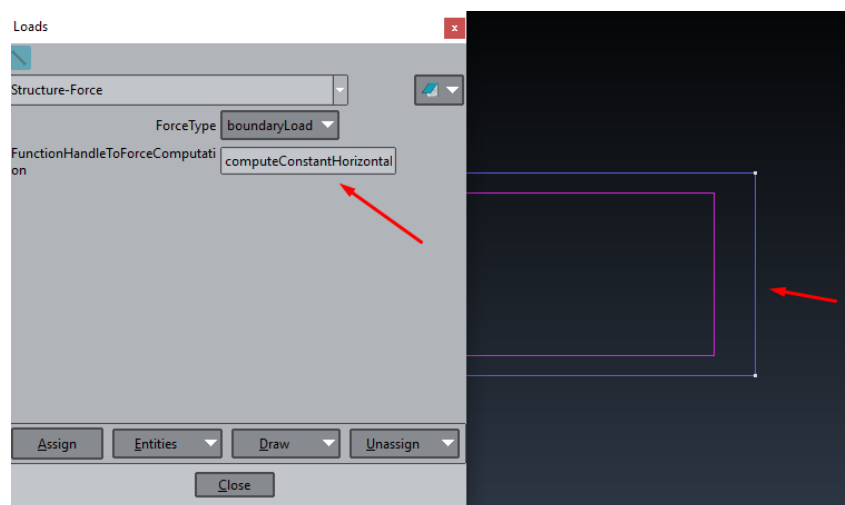
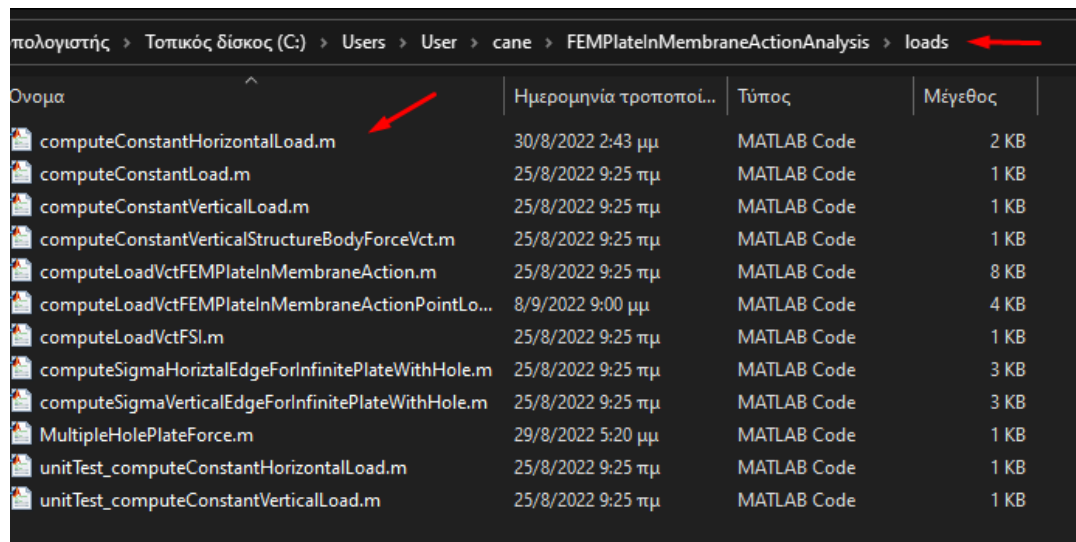


Figure 6: GiD – Choose function handle for load computation

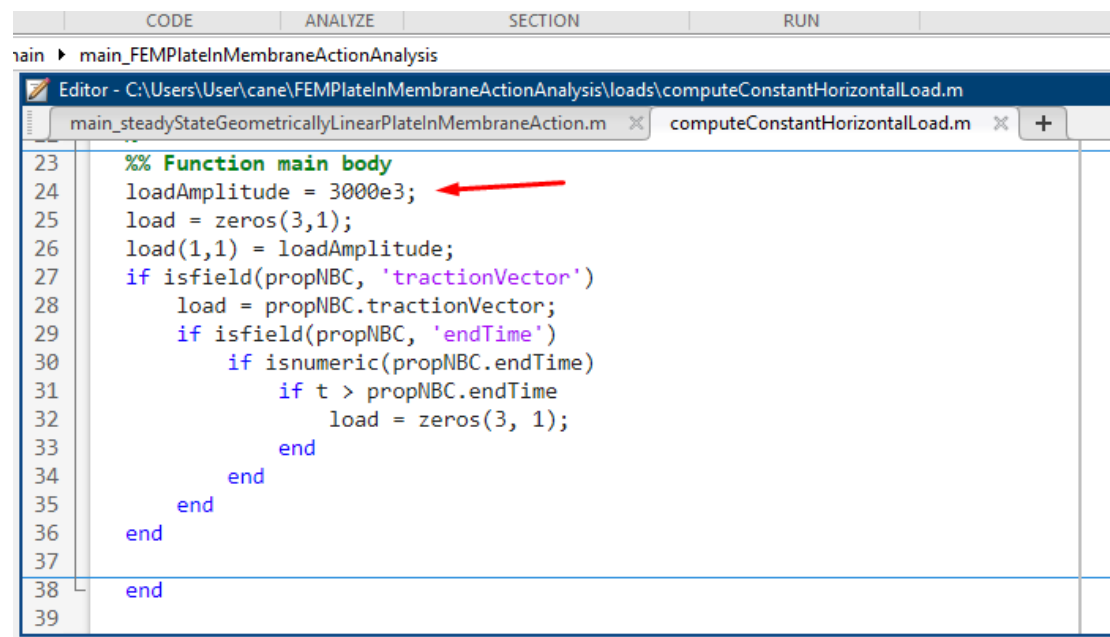
In this case, we use the function 'computeConstantHorizontalLoad', which is located in the loads directory of cane repository.



Όνομα	Ημερομηνία τροποποι...	Τύπος	Μέγεθος
computeConstantHorizontalLoad.m	30/8/2022 2:43 μμ	MATLAB Code	2 KB
computeConstantLoad.m	25/8/2022 9:25 πμ	MATLAB Code	1 KB
computeConstantVerticalLoad.m	25/8/2022 9:25 πμ	MATLAB Code	1 KB
computeConstantVerticalStructureBodyForceVct.m	25/8/2022 9:25 πμ	MATLAB Code	1 KB
computeLoadVctFEMPlateInMembraneAction.m	25/8/2022 9:25 πμ	MATLAB Code	8 KB
computeLoadVctFEMPlateInMembraneActionPointLo...	8/9/2022 9:00 μμ	MATLAB Code	4 KB
computeLoadVctFSI.m	25/8/2022 9:25 πμ	MATLAB Code	1 KB
computeSigmaHorizontalEdgeForInfinitePlateWithHole.m	25/8/2022 9:25 πμ	MATLAB Code	3 KB
computeSigmaVerticalEdgeForInfinitePlateWithHole.m	25/8/2022 9:25 πμ	MATLAB Code	3 KB
MultipleHolePlateForce.m	29/8/2022 5:20 μμ	MATLAB Code	1 KB
unitTest_computeConstantHorizontalLoad.m	25/8/2022 9:25 πμ	MATLAB Code	1 KB
unitTest_computeConstantVerticalLoad.m	25/8/2022 9:25 πμ	MATLAB Code	1 KB

Figure 7: GiD – Location of load script

Inside the script, we have to set the value for the amplitude of the load (3000 kN/m^2):



```

23  %% Function main body
24  loadAmplitude = 3000e3;
25  load = zeros(3,1);
26  load(1,1) = loadAmplitude;
27  if isfield(propNBC, 'tractionVector')
28      load = propNBC.tractionVector;
29      if isfield(propNBC, 'endTime')
30          if isnumeric(propNBC.endTime)
31              if t > propNBC.endTime
32                  load = zeros(3, 1);
33              end
34          end
35      end
36  end
37  end
38  end
39

```

Figure 8: Specify amplitude of horizontal load

Definition of the computational domain

The computational mesh needs then to be assigned to a domain, so that the nodes and the elements for the chosen domain are written out to the desirable input file. Select Data →

Conditions → Domains, choose Structure-Nodes from the drop-down menu and select the whole surface. Confirm with esc. Then, select Structure-Elements and repeat the previous step to assign the elements to the computational domain.

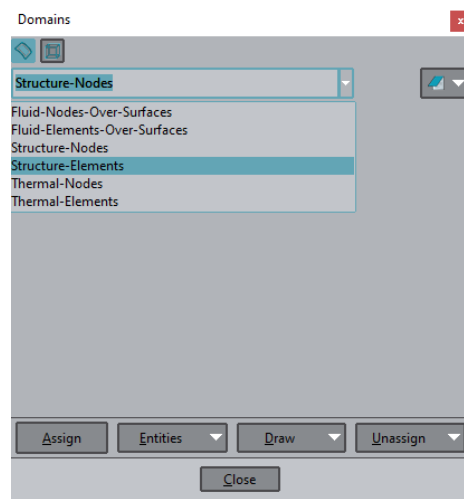


Figure 9: GiD – Assign nodes and elements to computational domain

Selection of material properties

In order to select the material properties, select Data → Materials → Solids. There one can select the default material Steel from the drop-down menu or just change the given parameters to adjust material properties. Apply the material to the geometry of the problem by selecting Assign -> Surfaces and choose the surface of defining the problem's domain. Save the changes if asked so. The user may also expand the materials selection by editing the corresponding files under the folder matlab.gid.

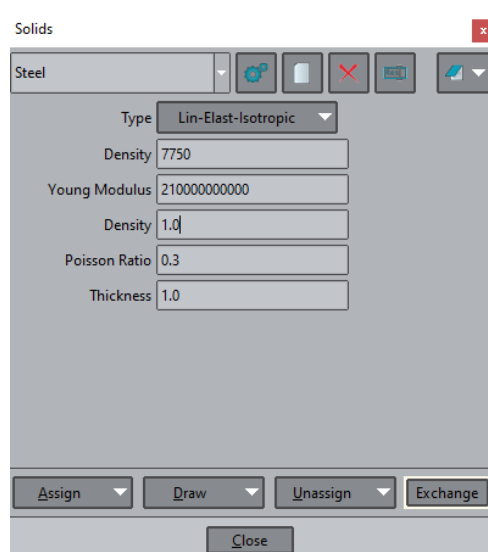


Figure 10: GiD – Material Properties selection

Generation of the computational mesh

Lastly, the finite element mesh needs to be generated. The simplest way is to go to Mesh → Generate Mesh and specify the element size.

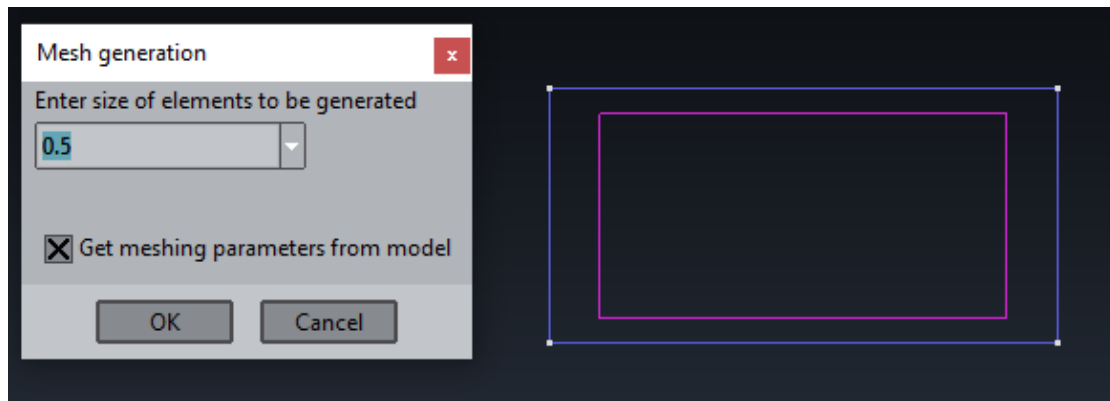


Figure 11: GiD – Mesh generation

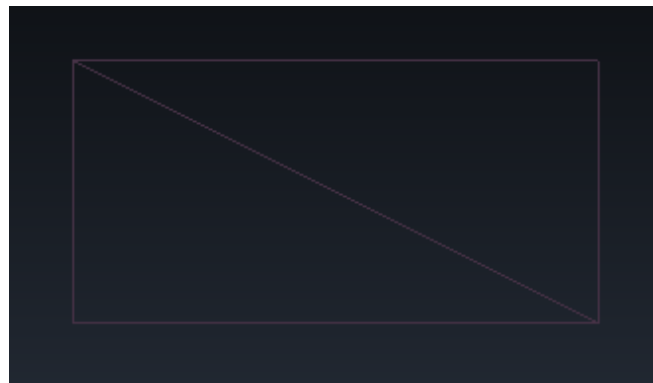


Figure 13: GiD – Generated mesh

Selection of the analysis type and solver setup

To select the analysis type and setup the solver, select Data → Problem Data → Structural Analysis. In this window the user can select plane stress or plain strain analysis type and choose between a steady state or a transient analysis. Many more settings are possible, specifically on the time integration schemes and Gauss integration, but they are not needed for this simple case in which the default options are sufficient.

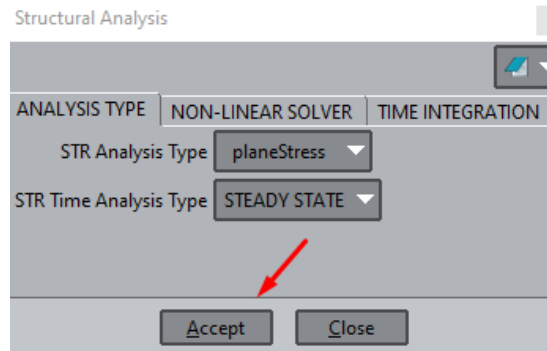


Figure 12: GiD – Analysis selection type

Generation of input file for analysis in MATLAB

After the setup is complete, select Calculation → Calculation (F5) or just select F5 to write out the input file which will be later on parsed within cane MATLAB framework. This file has the same name as our project whereas its extension is .dat. The user can also open it with any text editor to check or adjust the data. This file needs then to be placed under folder ./cane/inputGiD/FEMPlateInMembraneActionAnalysis and then a new caseName with the same name needs to be defined in the MATLAB main driver script.

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Structural Boundary Value Problem
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

STRUCTURE_ANALYSIS
  ANALYSIS_TYPE,planeStress

STRUCTURE_MATERIAL_PROPERTIES
  DENSITY,7750
  YOUNGS_MODULUS,210000000
  POISSON_RATIO,0.3
  THICKNESS,0.025

STRUCTURE_NLINEAR_SCHEME
  NLINEAR_SCHEME,NEWTON_RAPHSON
  NO_LOAD_STEPS,1
  TOLERANCE,1e-9
  MAX_ITERATIONS,100

STRUCTURE_TRANSIENT_ANALYSIS
  SOLVER STEADY_STATE
  TIME_INTEGRATION EXPLICIT_EULER
  ALPHA_BETA -0.1
  GAMMA 0.6
  START_TIME 0
  END_TIME 10
  NUMBER_OF_TIME_STEPS 100
  ADAPTIVE_TIME_STEPPING true

STRUCTURE_INTEGRATION
  DOMAIN default
  domainNoGP 1
  boundaryNoGP 1

```

Figure 13: GiD – Generated .dat file

Enumeration of nodes and degrees of freedom

The generated data file contains the structure nodes and elements enumerated. The enumeration starts from the top and ends in the bottom of each column of nodes. An illustrative example is defined below:

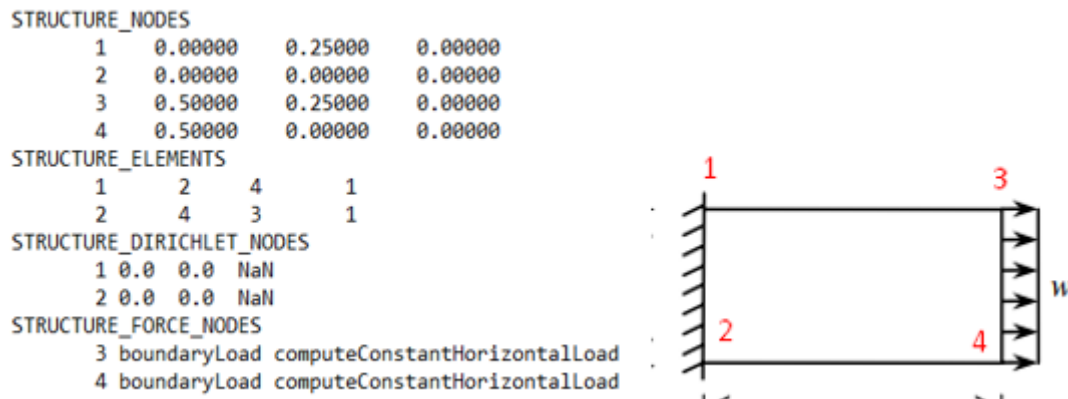


Figure 14: GiD – Enumeration of nodes

2.2.1.2. Solving with cane Multiphysics

Now, the geometry of the problem is properly defined, along with the boundary conditions and the mesh. The next step is to open the appropriate matlab script from the cane repository and specify how to handle our case problem.

From the cane main repository select main->mainFEM_PlateInMembraneActionAnalysis->main_steadyStateGeometricallyLinearPlateInMembraneAction.m. This is the matlab script which will parse the data file created from GiD.

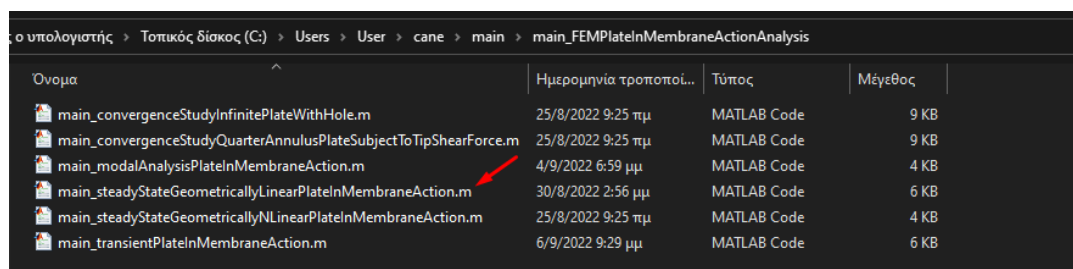


Figure 15: cane – main script location

The first thing we need to do when we open the script is to define a path for matlab in order to find the generated file from GiD. We create a new case with the name of the generated file, in our case 'PlateInPlaneStressSteadyState'.

```

Editor - C:\Users\User\cane\main\main_FEMPlateInMembraneActionAnalysis\main_steadyStateGeometricallyLine
main_steadyStateGeometricallyLinearPlateInMembraneAction.m  computeConstantHorizontalLoad.m
46      % Include performance optimized functions
47      addpath('..\..\efficientComputation/');
48
49      %% Parse data from GiD input file
50
51      % Define the path to the case
52      pathToCase = '../inputGiD/FEMPlateInMembraneActionAnalysis/';
53      % caseName = 'infinitePlateWithHoleQuadrilaterals';
54      % caseName = 'cantileverBeamPlaneStress';
55      % caseName = 'PlateWithAHolePlaneStress';
56      % caseName = 'PlateWithMultipleHolesPlaneStress';
57      % caseName = 'InfinitePlateWithAHolePlaneStress';
58      % caseName = 'unitTest_curvedPlateTipShearPlaneStress';
59      % caseName = 'gammaStructureMixedElementsPlaneStress';
60      % caseName = 'NACA2412_AoA5_CSD';
61      caseName = 'PlateInPlaneStressSteadyState';
62

```

Figure 16: cane – specify selected case

Now, everything is set and the problem is ready to be solved. Hit F5 or run for matlab to start the calculations. Matlab will automatically generate two figures. The first figure shows the initial configuration of the problem along with the forces applied and the second one shows deformation and stresses. For more post-processing options we open the output .vtk file in Paraview.

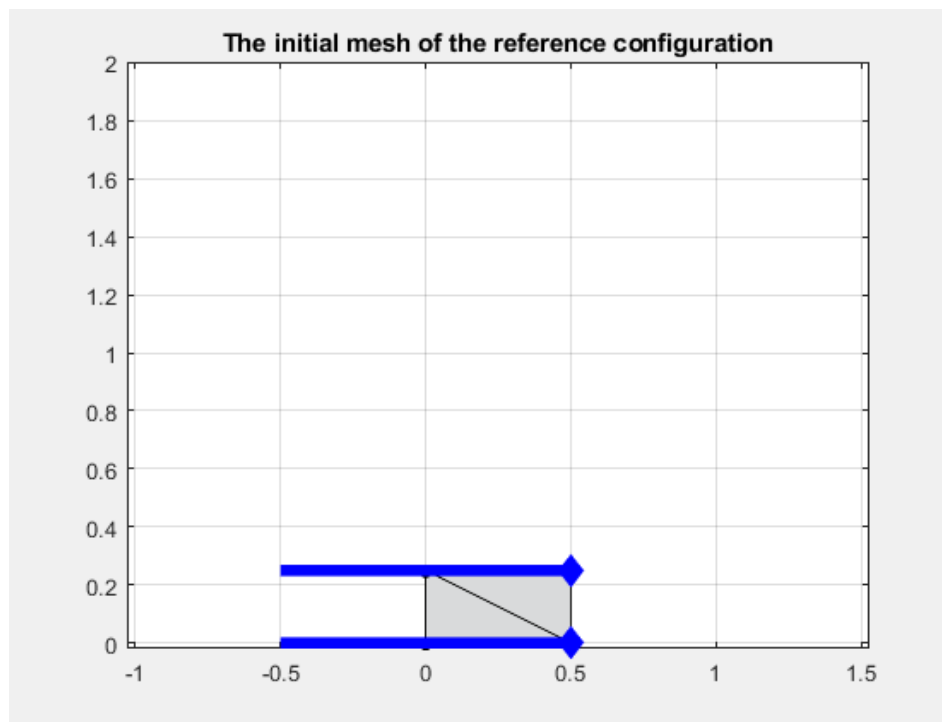


Figure 17: Initial mesh of the reference configuration

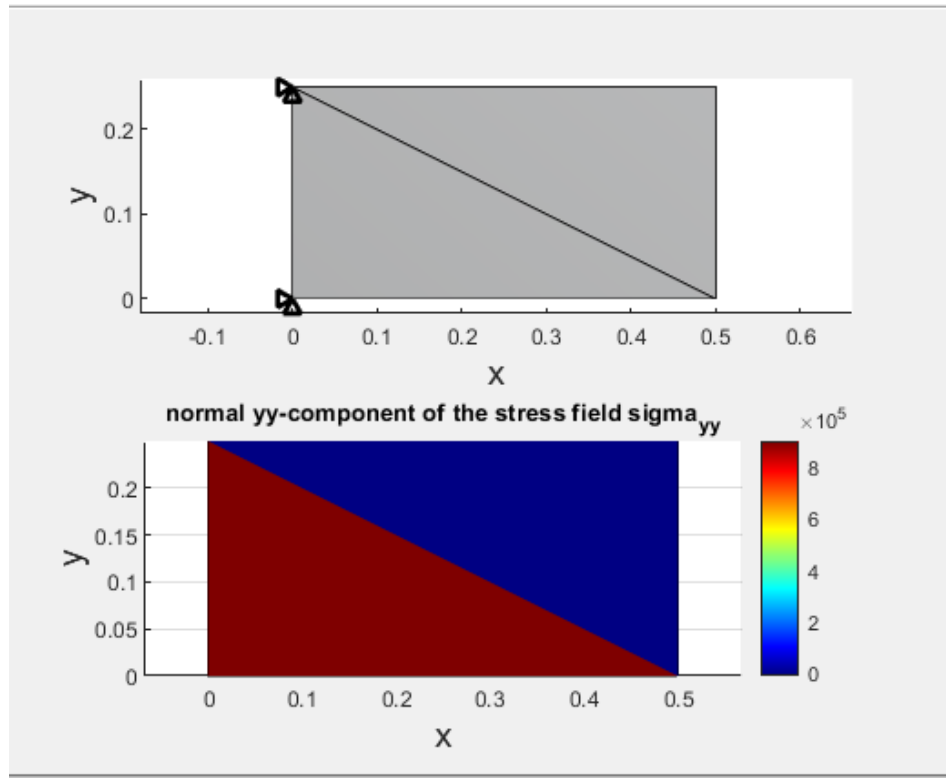


Figure 18: Deformations and stresses on the structure

We can print the results inside matlab through the command window. By typing 'dHat' we get the displacement vector and by typing F the user can get the force vector.

```
dHat =
    1.0e-05 *
         0
         0
         0
         0
         0
    0.7111
    0.1115
    0.6531
    0.0045

F =
         0
         0
         0
         0
         0
    375000
         0
    375000
         0
```

Figure 19: Displacement and force vectors calculated in cane

From a quick view we can easily validate that the Dirchlett boundary condition on the left end of the plate has been imposed, since the corresponding displacements are set to zero. Also, since the thickness of the plate is 0.25m we can equally distribute the uniform force in the right end to the two edges of the plate. The calculation is simple:

- $F_{distributed} = (3000000 \text{ N/m}^2) * (0.5 \text{ m}) * (0.25 \text{ m}) = 375000 \text{ N}$

As the figure above shows, the force vector is calculated as expected. We can further validate our results by comparing the displacement vector calculated from cane and the one found in [3]. We can easily see that the two displacement vectors are the same.

$$U = 1.0e-005 \cdot \begin{bmatrix} 0 \\ 0 \\ 0.7111 \\ 0.1115 \\ 0.6531 \\ 0.0045 \\ 0 \\ 0 \end{bmatrix}$$

Figure 20: Displacement vector from the reference solution

2.2.1.3. Results

At this point, the problem has been solved and the output files have been prepared. We can open the output files with Paraview, an open source post-processing software for better visualization of the results.

The first step is to open Paraview and then choose the appropriate VTK file. The file is located in cane repository in the path /cane/outputVTK/.

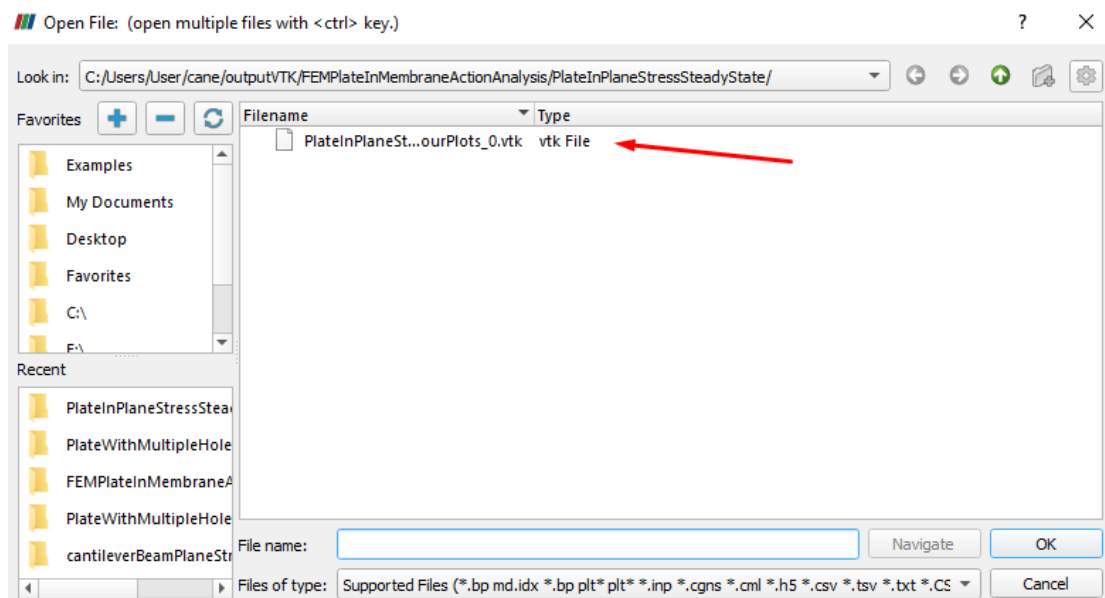


Figure 21: Paraview – Open VTK file

Then we select Apply from the Properties tab:

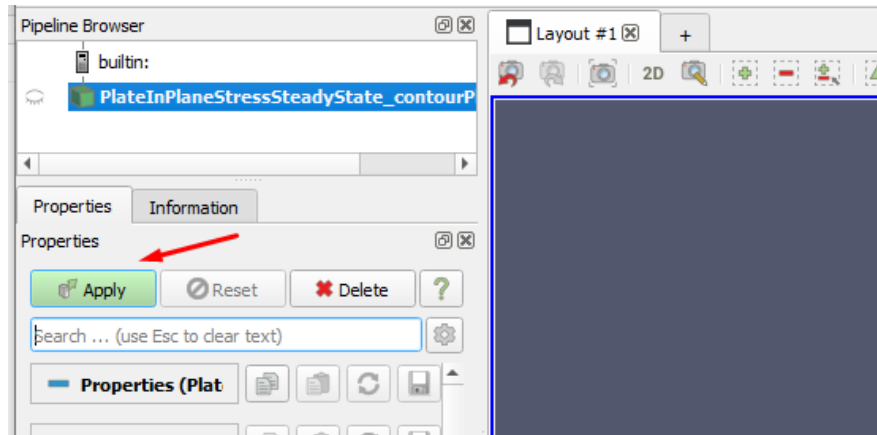


Figure 22: Paraview – Select apply button

In order to visualize the resultings displacements of structure we select the following options from Paraview's interface, as shown in the figure below:

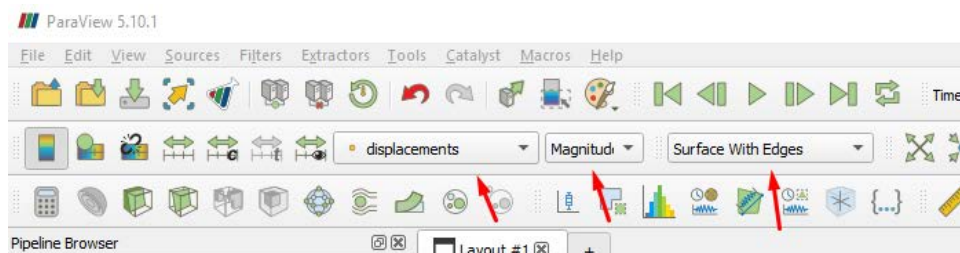


Figure 23: Paraview – Selection of quantities to show

The corresponding displacement field is shown in the figure below:

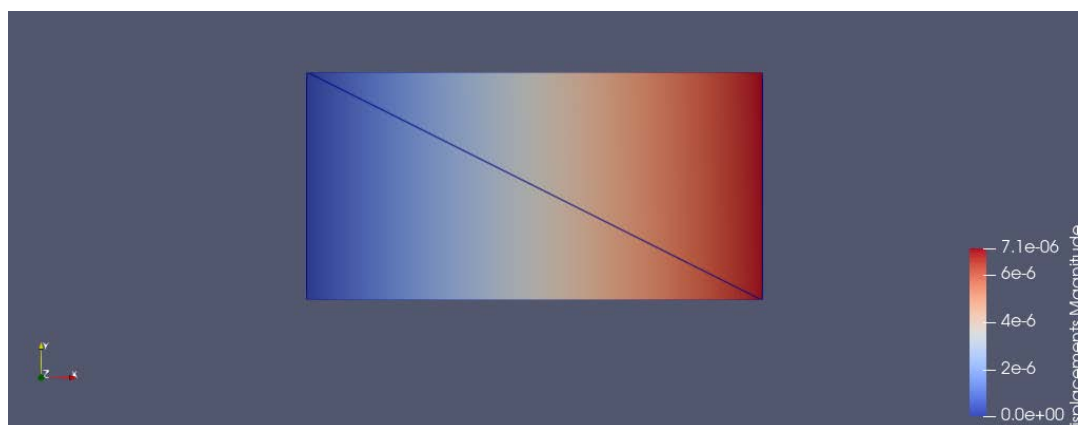


Figure 24: Paraview – Displacement field

2.2.2. Modal Analysis of a multihole panel

A panel with four circular holes is considered to demonstrate cane's modal analysis capabilities. The complete description of the problem can be found in [4]. The problem has the following parameters:

- Height: $H = 1 \text{ m}$
- Length: $L = 5 \text{ m}$
- Radius of holes: $r = 0.3 \text{ m}$
- Modulus of elasticity: $E = 206 \text{ GPa}$
- Mass density: $\rho = 7800 \text{ kg/m}^3$
- Poisson's ratio $\nu = 0.3$

The geometry of the problem is presented in the figure below:

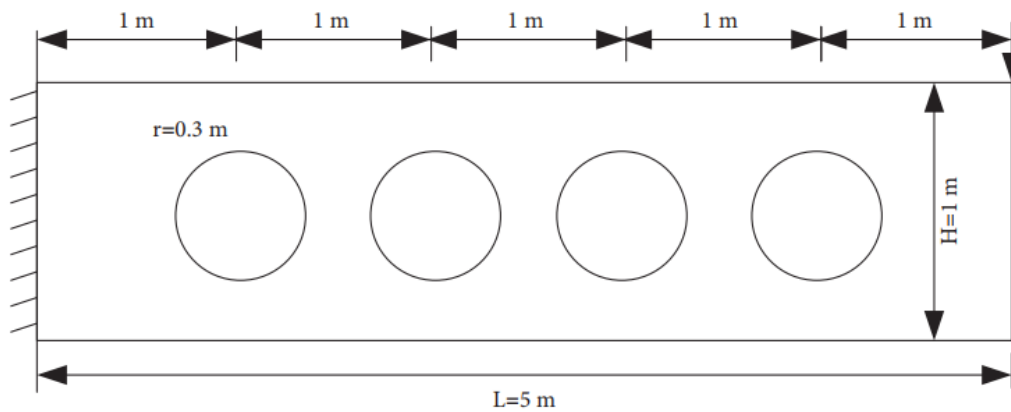


Figure 25: Multihole Panel

2.2.2.1. Preprocessing with GiD

The pre-processing stage of the problem doesn't differ so much from the previous steady state problem, so now we will focus only on the different features of GiD that we will need to use in order to properly describe the problem.

Firstly we have to create a rectangle object with two points, namely $(0,0)$ and $(5,1)$. After that, we need to create four circles. These circles will have a radius of 0.3 m and the distance between them will be set at 1 m . The normal vector of all the circles have to point in the positive Z direction.

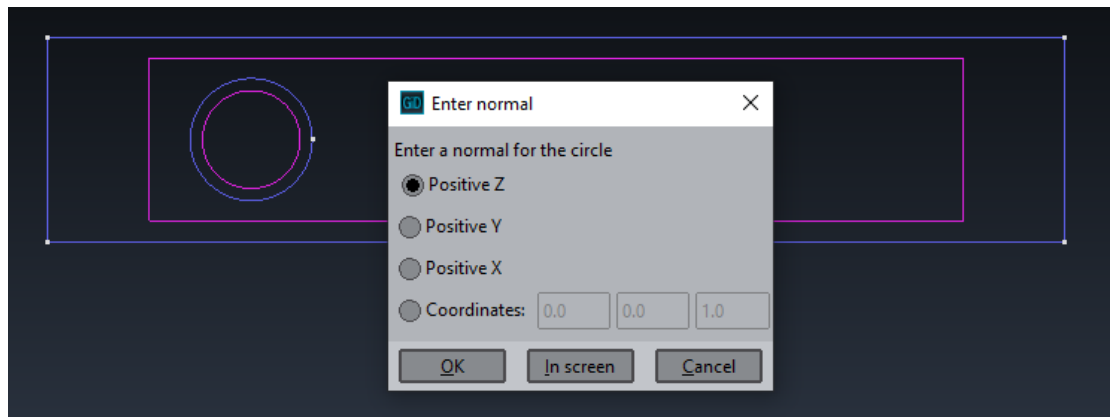


Figure 26: GiD – Circle creation

Now that the four circles have been designed, we have to pass the information that these circles represent holes. This can be done with the Boolean surface operations of GiD. The main idea is that we subtract from the rectangle surface, the four circular surfaces.

In order to do this, we select Geometry -> Edit -> Surface Boolean op. ->Subtraction

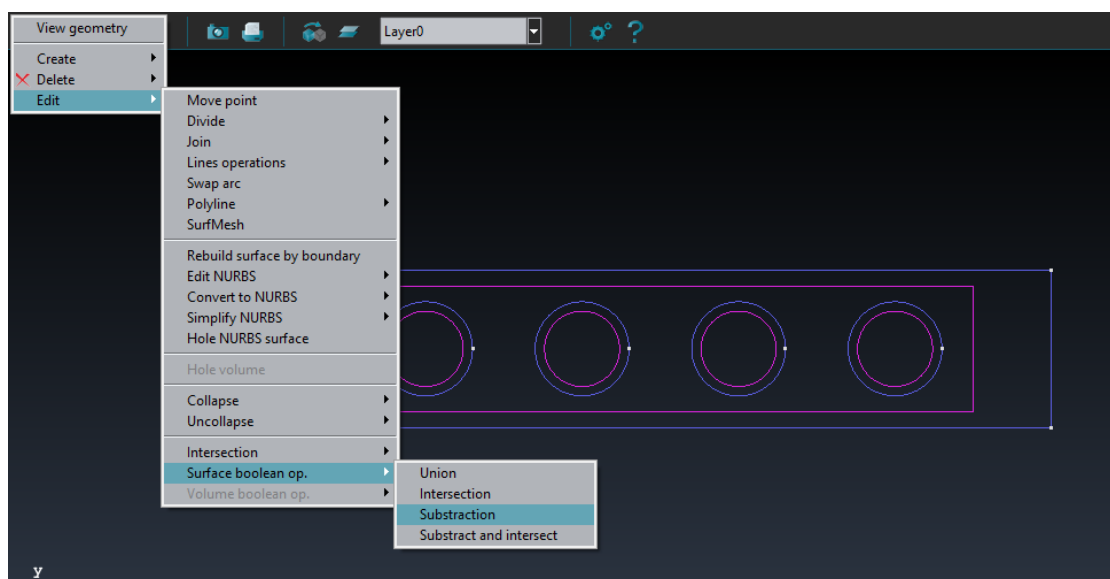


Figure 27: GiD – Surface boolean operations

At this point we have to select the surface from which we will subtract the other surfaces, namely the rectangle surface. After selection hit the ESC button. Then select the four circular surfaces and hit the ESC button again. If the operation was successful GiD will return the message 'Surfaces subtraction finished'. The resulting surface is shown below.

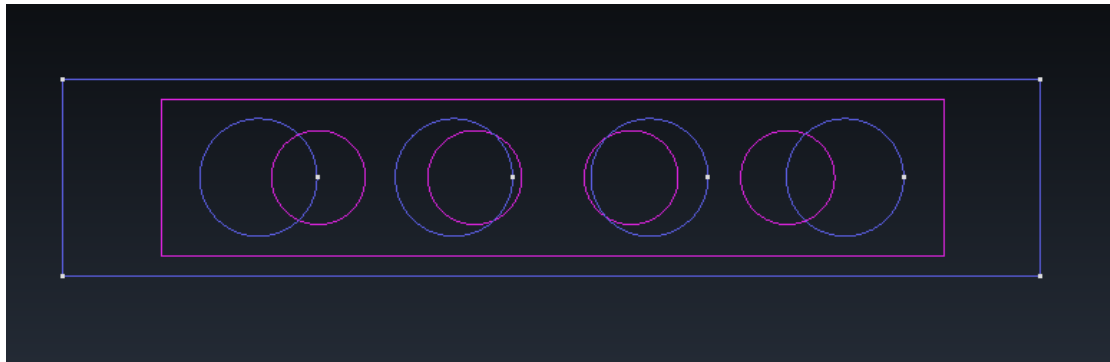


Figure 28: GiD – Subtracted surfaces

A good way to validate that our operation was successful is to see whether you can apply material property to the circular surfaces. The figure below clearly shows that the circular holes are void, and consequently the program prevents you from assigning material properties to them.

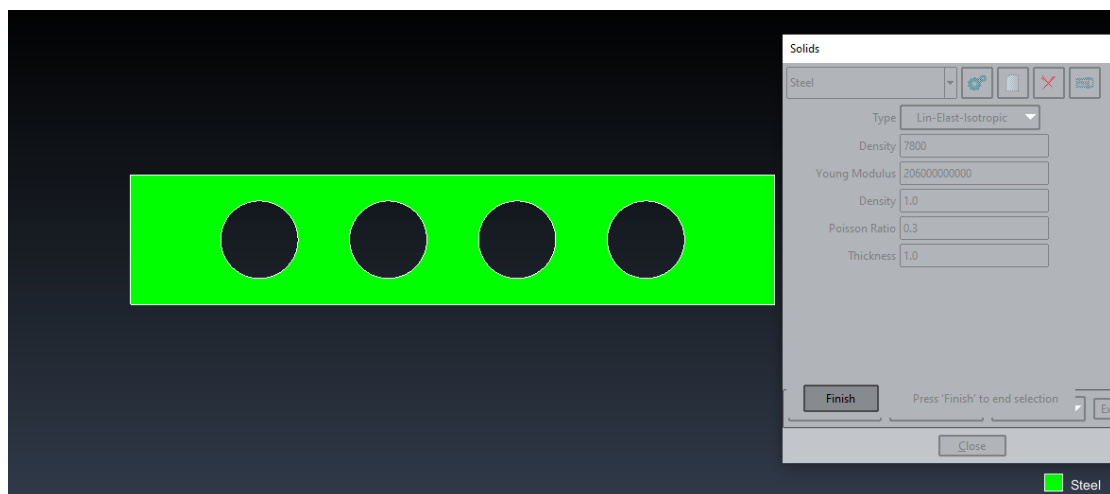


Figure 29: GiD – Show section with selected material property

For the discretization of the structure we choose to use quadrilateral plane stress finite elements. To do that, we select Mesh -> Element type -> Quadrilateral

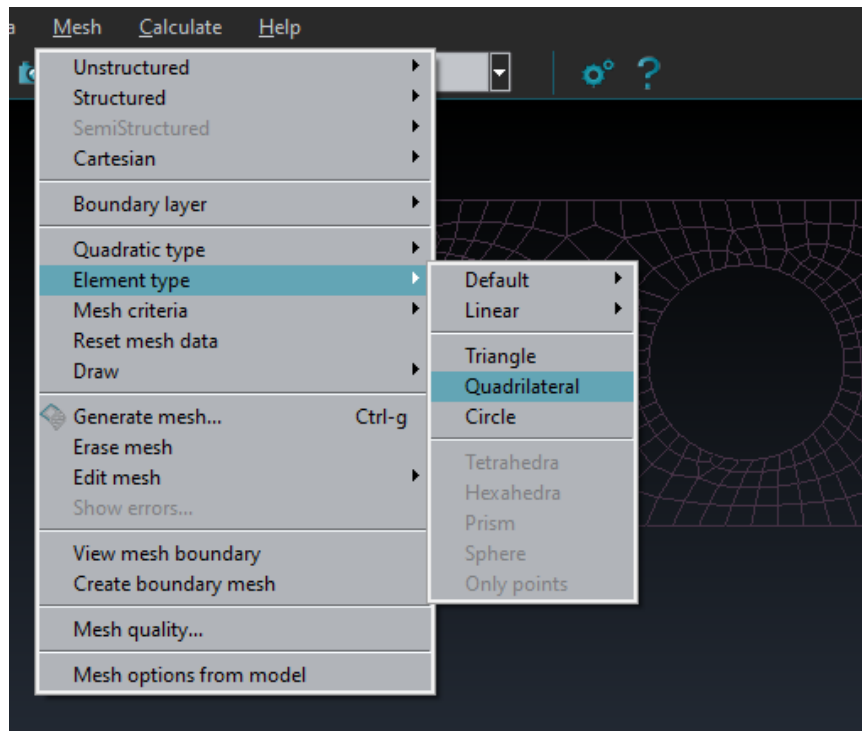


Figure 30: GiD – Quadrilateral elements selection

Finally, we generate the mesh and press F5 to produce the .dat file that will be processed by the cane matlab script for modal analysis

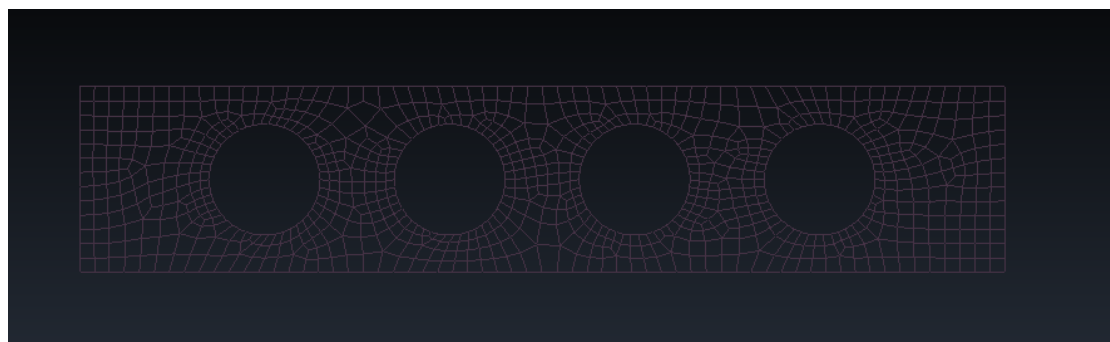


Figure 31: GiD – Generated mesh for the multihole panel

2.2.2.2. Solving with cane Multiphysics

At this point, we open the matlab script in cane repository which will perform the modal analysis.

From the cane main repository we select main_modalAnalysisPlateInMembraneAction.m. As we did in the previous example, we now have to define a new case with the name of the generated data file from GiD. The modal analysis is independent of loads, so we don't have to specify any loading condition. We can now run the analysis by pressing F5. Matlab will

automatically produce a figure that shows the first eigenmode that corresponds to the first natural eigenfrequency.

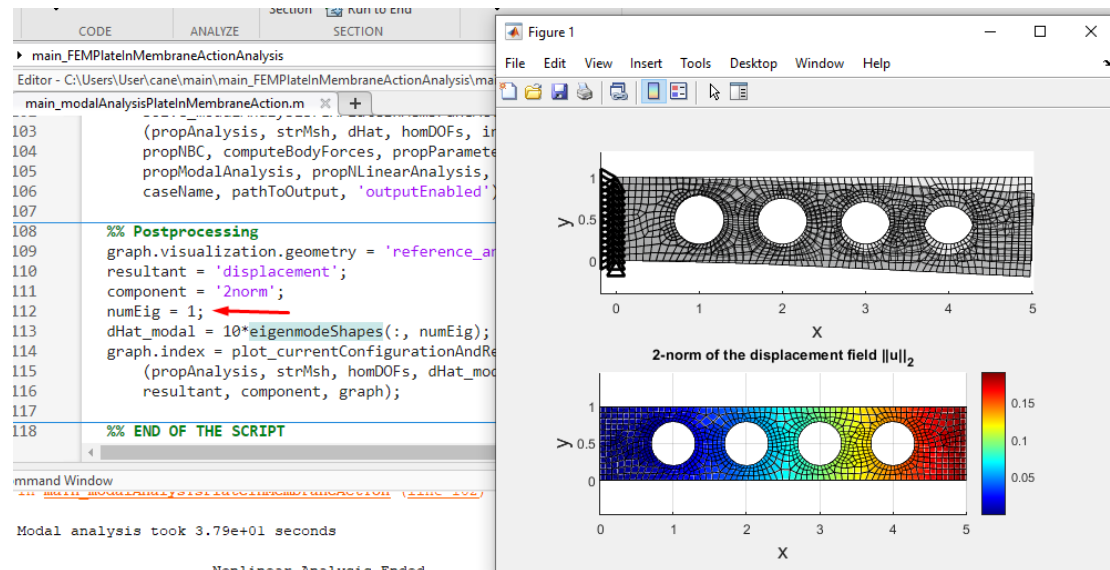


Figure 33: Displacement field and first eigenmode

We can visualize any other eigenfrequency by changing the value of numEig variable. We now set it to 2 for illustration purposes and press Run Section while inside the Postprocessing section.

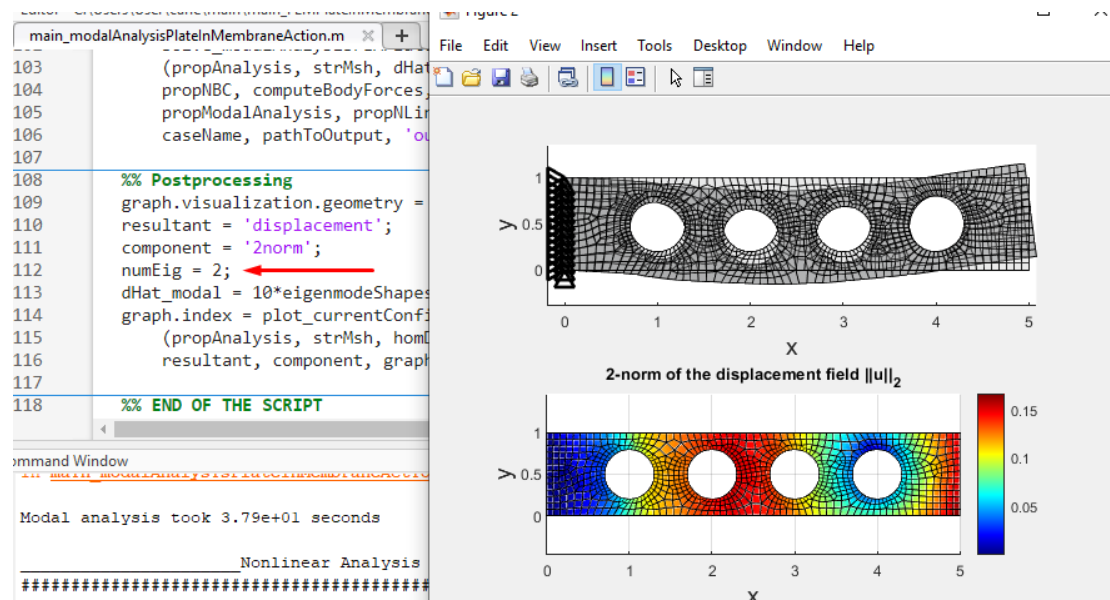


Figure 34: Displacement field and second eigenmode

The user can print the natural frequencies calculated through the command window by typing 'naturalFrequencies'. The figure below shows the first 10 natural frequencies. To validate our results, we get the values of natural frequencies as computed in [4].

```
naturalFrequencies =  
  
    31.4194  
    157.2139  
    200.9254  
    358.5547  
    550.5410  
    596.2356  
    753.8505  
    861.7080  
    916.7075  
    956.4134
```

Figure 35: First 10 natural eigenfrequencies

<i>Eigenfrequencies (Hz)</i>	<i>Mode 1</i>	<i>Mode 2</i>	<i>Mode 3</i>	<i>Mode 4</i>
<i>Reference Eigenfrequencies</i>	31.811	150.19	209.31	330.64
<i>Cane Eigenfrequencies</i>	31.42	157.21	200.92	358.55

ΒΑΛΕ ΔΙΑΓΡΑΜΜΑ ΜΕ ΙΔΙΟΣΥΧΝΟΤΗΤΕΣ ΑΠΟ ΕΠΙΛΥΣΗ ΜΕ ΠΑΡΑΠΑΝΩ ΚΟΜΒΟΥΣ ΣΥΓΚΡΙΤΙΚΑ!

The small differences in the values is due to the discretization of the structure. The reference solution uses more quadrilateral elements (1004 nodes in contrast to 2478 nodes) and thus, a more accurate solution is obtained.

2.2.2.3. Results

We can visualize the resulting eigenmodes in paraview by opening the .vtk output file. The first four eigenmodes corresponding to the first four natural eigenfrequencies are shown below.

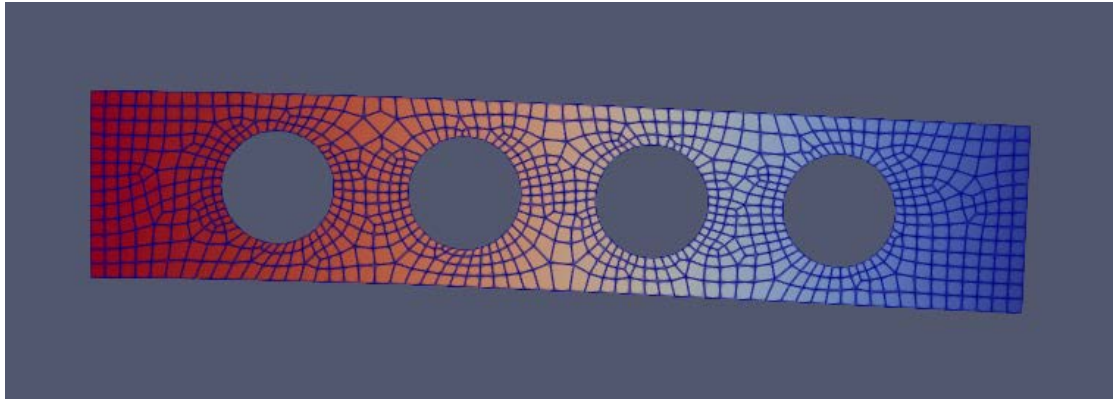


Figure 36: First eigenmode $f = 31.81$ (Hz)

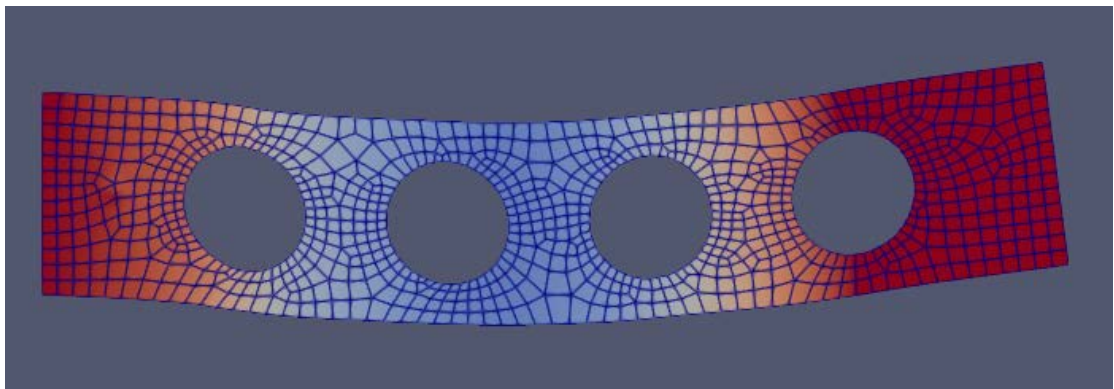


Figure 37: Second eigenmode $f = 150.19$ (Hz)

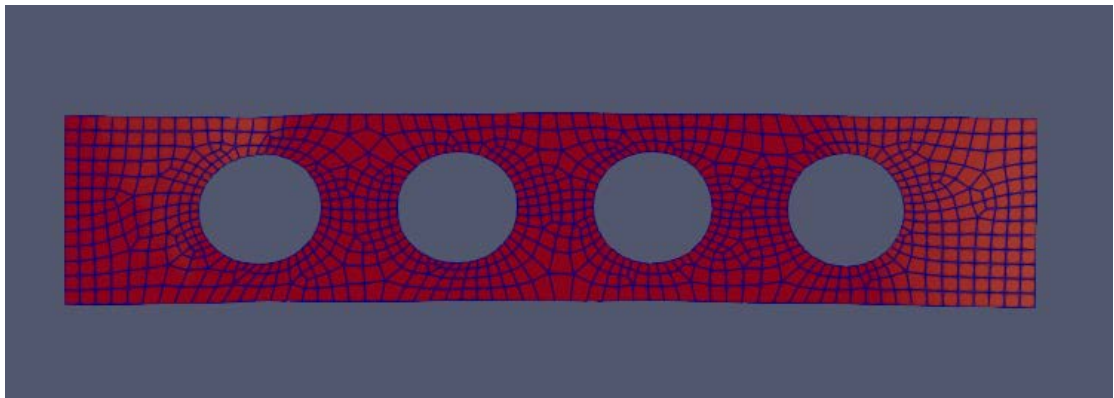


Figure 38: Third eigenmodes $f = 209.31$ (Hz)

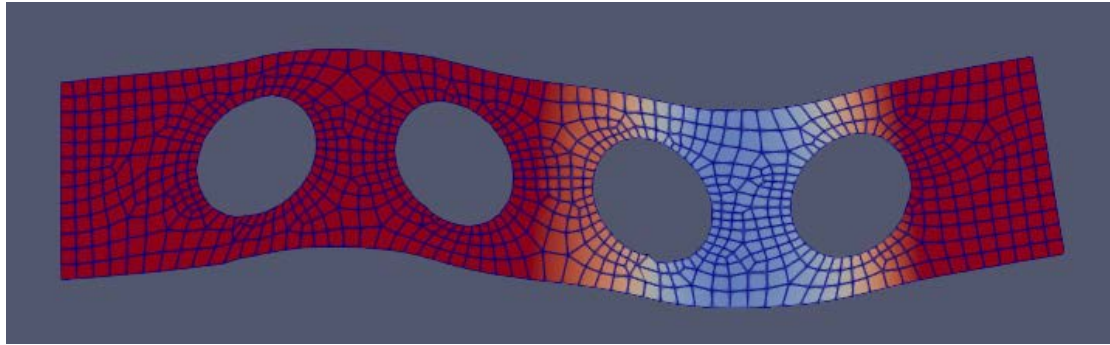


Figure 39: Fourth eigenmode $f = 330.64$ (Hz)

The user can navigate through the eigenmodes using the 'Next Frame' and 'Previous Frame' buttons:

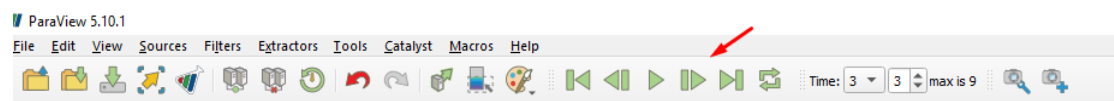


Figure 40: Navigation through eigenmodes

2.2.3. Transient Analysis of a multihole panel

The third problem [4] addressed in this chapter is about the forced vibration of the same multihole panel presented in the previous example. A time-dependent force $p(t) = 10000\sin(199.77\pi t)$ acts on the right end of the panel, as presented below. The same mesh and material properties are applied in this problem as well. We are going to monitor the transient response of the vertical displacement of the bottom right node and compare it with the reference response.

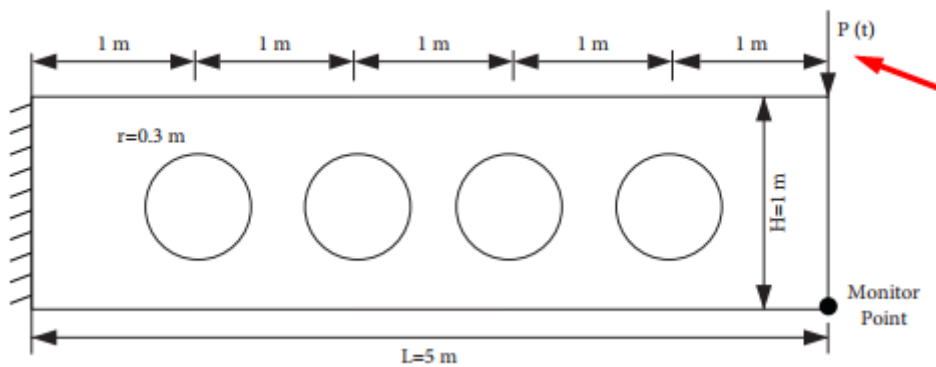


Figure 41: Nodal force applied on the multihole panel

2.2.3.1. Preprocessing with GiD

In the previous example, we didn't have to create a loading boundary condition because the modal analysis is independent of loads. In this occasion, we have to create a new condition inside GiD.

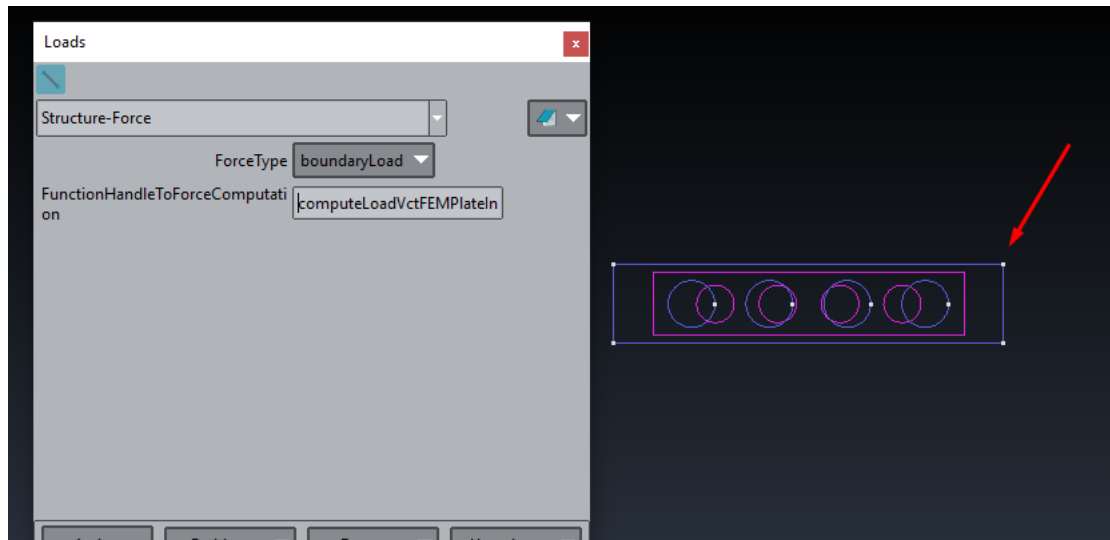


Figure 42: Function handle to compute nodal force

The cane framework doesn't have a function for calculating nodal forces on specific nodes, so there is a need to create a new matlab script. In this script, we will have to search our mesh and find the node we want by checking if there exists a node in the mesh with the exact coordinates given. In this occasion, the node we have coordinates (5,1,0) which is the top right node. The following code finds the node and applies the force amplitude in the corresponding components of the force vector. The name of the matlab script is `computeLoadVctFEMPlateInMembraneActionPointLoad` and through GiD we created a function handle that points to that script, so that when the load vector is calculated during the analysis, this function will be used.

```
%% 1.Get the force vector
p_ampl_y = -10000*sin(199.77*pi*t);
p_ampl_x = 0;

id = ismember(strMsh.nodes(:,2:4), [5 1 0], 'rows');
pos = find(id==true);
node_id = strMsh.nodes(pos,1);
F = zeros(2*numel(strMsh.nodes(:, 1)), 1);
F(2*node_id - 1) = p_ampl_x;
F(2*node_id) = p_ampl_y;
```

Figure 43: Source code used to calculate nodal force

Furthermore, we have to tell GiD that the analysis is going to be transient. Select Data-> Structural Analysis. In the window that pops up choose transient analysis.

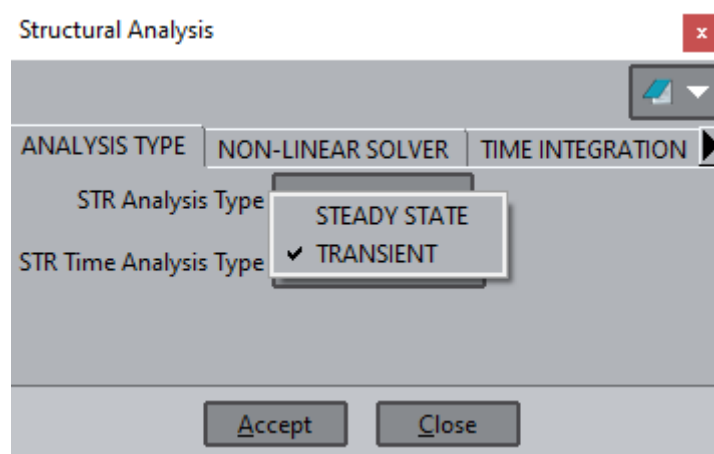


Figure 44: Transient structural analysis

Moving to the time integration tab, we set the end time of the analysis to 0.4. This step is done in order to produce relevant results to those in the solution of the problem in [4].

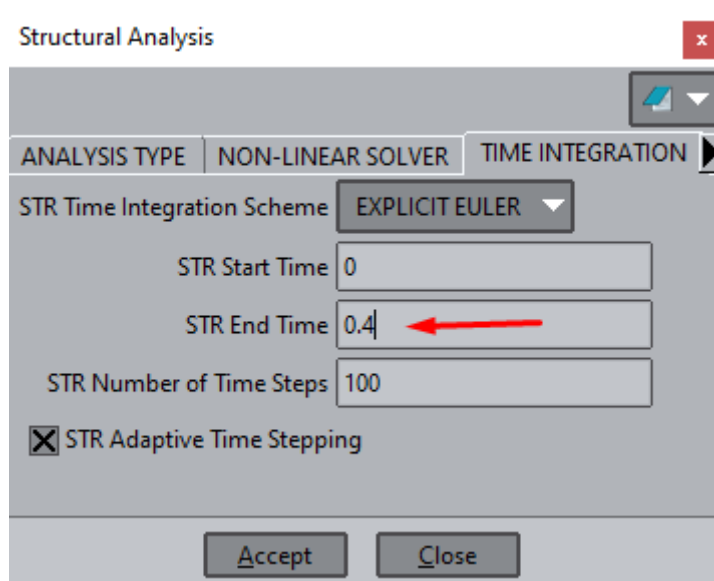


Figure 45: Set time domain

2.2.3.2. Solving with cane Multiphysics

Once again, we create a new case named after the data file produced by GiD.

```

%% Parse data from GiD input file

% Define the path to the case
pathToCase = '../inputGiD/FEMPlateInMembraneActionAnalysis/';

% caseName = 'curvedPlateTipShearPlaneStressTransient';
% caseName = 'cantileverBeamPlaneStressTransientNLinear';
% caseName = 'turek_csd';
caseName = 'PlateWithMultipleHolesInVibrationTransientAnalysis';

% Parse the data from the GiD input file
[strMsh, homDBC, inhomDBC, valuesInhomDBC, propNBC, propAnalysis, ...
 parameters, propNLinearAnalysis, propStrDynamics, propGaussInt] = .
    parse_StructuralModelFromGid(pathToCase, caseName, 'outputEnabled')

% Define traction vector
propNBC.tractionLoadVct = [0; -1e1; 0];

```

Figure 46: Case selection for transient analysis

Everything is set, so we press F5 to run the analysis.

2.2.3.3. Results

Inside Matlab, we want to select the bottom right node and plot the time history of its vertical displacement. To do that in Matlab we have to find the id of the node interest using its coordinates (5,0,0). Then, from the total displacement vector, dHat we extract the vertical displacement of the node in each timestep. To perform this calculation, the following code was used:

```

id = ismember(strMsh.nodes(:,2:4), [5 0 0], 'rows');
pos = find(id==true);
node_id = strMsh.nodes(pos,1);
dy = zeros(propStrDynamics.noTimeSteps, 1);
for ii = 1:propStrDynamics.noTimeSteps
    dy(ii, 1) = dHat(2*node_id, ii);
end
plot(propStrDynamics.T0+propStrDynamics.dt:propStrDynamics.dt:propStrDynamics.TEnd, dy);

```

Figure 47: Source code to plot displacement of node

Finally, the resulting displacements from the reference solution and cane are compared. The first diagram represents the displacement from the reference solution. We are only interested in the case when damping is applied and the behavior of the system results in a steady state solution, as shown in the figure below:

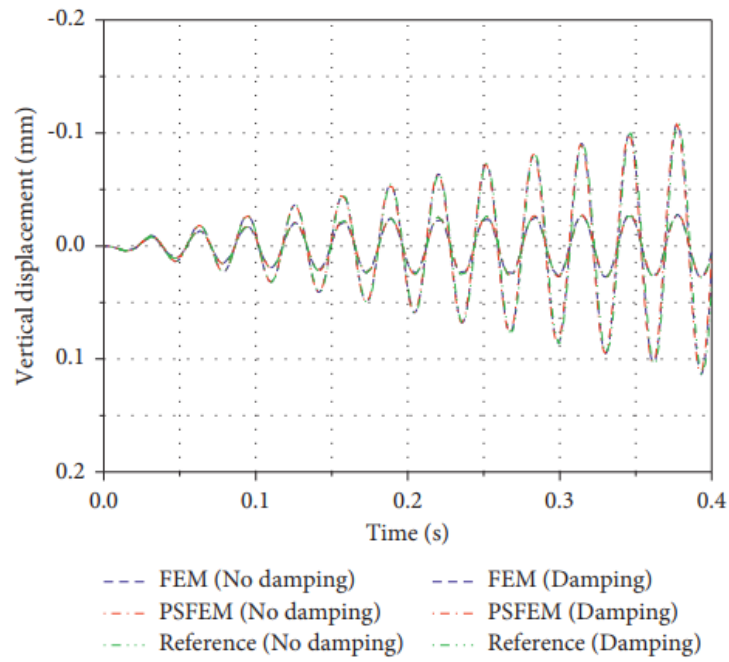
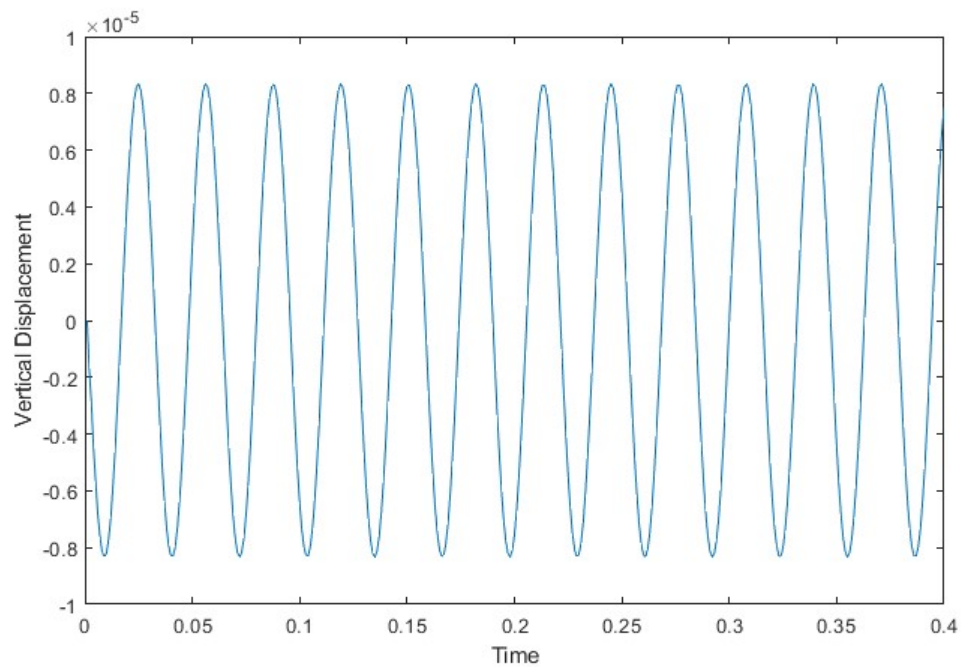


Figure 48: Time history of vertical displacement, reference solution

Next, the results from the analysis performed by cane are presented:



We can see that the two solutions present the same behavior. The resulting displacement fields are almost identical.

2.3. References for Chapter2

- [1] GiD Simulation, GiD Simulation, [Ηλεκτρονικό]. Available: <https://www.gidsimulation.com/>.
- [2] ParaView, [Ηλεκτρονικό]. Available: <https://www.paraview.org/>.
- [3] P. I.Kattan, MATLAB Guide to Finite Elements_An interactive Approach, 2008.
- [4] C. S. Y. Y. Nan Ye, «Free and Forced Vibration Analysis in Abaqus Based on the Polygonal Scaled Boundary Finite Element Method,» *Hindawi - Advances in Civil Engineering*, τόμ. 2021, p. 17, 2021.

3. Computational Fluid Dynamics

This chapter deals with the presentation of fluid dynamics theory and the numerical methods used to solve the corresponding problems. Specifically, the strong formulation of the incompressible Navier-Stokes equations is stated. Then the Variational Multiscale Stabilization (VMS) method is used to produce the weak form. Finally, classic Finite Element Method (FEM) is used for the spatial discretization and the Bossak integration scheme is used time discretization.

3.1. Navier-Stokes strong formulation

Let Ω be a domain with a piecewise continuous boundary $\Gamma = \partial\Omega$ where a Newtonian incompressible fluid flow takes place. The characteristics of the domain and flow are as stated as follows:

- Fluid density (constant) : ρ
- Kinematic viscosity (constant) : ν
- Dynamic viscosity (constant) : $\nu = \mu / \rho$
- Subjected body forces: \mathbf{f}

The primary unknown variables of the problem is the velocity field $\bar{\mathbf{u}}$ and the pressure field \bar{p} which are prefixed to some value in the beginning of the domain. An inlet (Dirichlet) boundary condition is imposed at the domain and is called the inlet boundary $\Gamma_d \subset \Gamma$.

The fluid traction vector along a cut $\gamma \subset \bar{\Omega}$ with outward normal \mathbf{n} is given by:

$$\mathbf{t} = -p\mathbf{n} + \mathbf{n} \cdot \nu \nabla^s \mathbf{u} \quad (3.1)$$

where ∇^s is the symmetric gradient operator, namely $\nabla^s = \mathbf{1}/2(\nabla + \nabla^t)$. Moreover:

- p is expressed in $\mathbf{Nm} / \mathbf{Kg}$ since it is normalized by the fluid density,
- The actual pressure is obtained by $\tilde{p} = \rho p$.
- The domain Ω is also assumed to be moving with a given velocity \mathbf{u}_d independently of the fluid flow. Then, accordingly to the Arbitrary Lagrangian-Eulerian description of motion, the inertial forces are given by means of the material derivative:

$$\mathbf{D}(\cdot) = (\cdot) + (\mathbf{u} - \mathbf{u}_d) \cdot \nabla(\cdot) \quad (3.2)$$

where $(\cdot) = \partial(\cdot) / \partial t$ stands for the time derivative.

Now that the characteristics of the problem are fully stated, the presentation of the strong formulation of the transient incompressible Navier-Stokes problem follows:

Find the velocity and pressure field $\mathbf{u} \in \mathcal{C}^2(\Omega)$ and $p \in \mathcal{C}^1(\Omega)$, respectively, such that,

$$\dot{\mathbf{u}} - \nu \Delta \mathbf{u} + (\mathbf{u} - \mathbf{u}_d) \cdot \nabla \mathbf{u} + \nabla p = \mathbf{f} \text{ in } \Omega \quad (3.3)$$

$$\nabla \cdot \mathbf{u} = 0 \text{ in } \Omega \quad (3.4)$$

$$\mathbf{u} = \bar{\mathbf{u}} \text{ on } \Gamma_d \quad (3.5)$$

$$p = \bar{p} \text{ on } \Gamma_d \quad (3.6)$$

$$\mathbf{t} = \bar{\mathbf{t}} \text{ on } \Gamma_n \quad (3.7)$$

- $\bar{\mathbf{t}}$ stands for the applied traction vector on the outlet (Neumann) boundary $\Gamma_n \subset \Gamma$
- Δ stands for the Laplace second-order operator.

3.2. Variational multiscale stabilization of the weak form

This section deals with the presentation of the variational formulation of the Navier-Stokes equations. Firstly, the following conditions have to be applied:

- Multiply equations (3.3) and (3.4) with test functions $\delta \mathbf{u}$ and δp respectively.
- Integrate over Ω .
- Perform integration by parts.
- Apply the boundary conditions.

Now the problem can be restated; Find $\mathbf{u} \in \mathcal{H}^1(\Omega)$ and $p \in \mathcal{L}^2(\Omega)$ such that,

$$\begin{aligned} \langle \delta \mathbf{u}, \dot{\mathbf{u}} \rangle_{0,\Omega} + \langle \nu \nabla \delta \mathbf{u}, \nabla \mathbf{u} \rangle_{0,\Omega} + \langle \delta \mathbf{u}, (\mathbf{u} - \mathbf{u}_d) \cdot \nabla \mathbf{u} \rangle_{0,\Omega} - \langle \delta \mathbf{u}, p \rangle_{0,\Omega} - \langle \delta p, \nabla \cdot \mathbf{u} \rangle_{0,\Omega} = \\ \langle \delta \mathbf{u}, \mathbf{f} \rangle_{0,\Omega} + \langle \delta \mathbf{u}, \mathbf{t} \rangle_{0,\Gamma_e}, \end{aligned} \quad (3.8)$$

for all $\delta \mathbf{u} \in \mathcal{H}^1(\Omega)$ and all $\delta p \in \mathcal{L}^2(\Omega)$, where $\langle \mathfrak{T}, \mathfrak{T}' \rangle_{0,\Omega} = \int_{\Omega} \mathfrak{T}_{\alpha_1 \dots \alpha_m} \mathfrak{T}'_{\alpha_1 \dots \alpha_m} d\Omega$ stands for the \mathcal{L}^2 -norm in Ω for any pair of equal order tensors $\mathfrak{T}, \mathfrak{T}' \in \mathfrak{S}^m$.

At this point, the so-called Variational Multi-Scale (VMS) stabilization with the Algebraic Sub-Grid Scales (ASGS) projection of variational formulation in equation (3.8) is employed.

Let $\tilde{\mathcal{L}}: \mathbf{C}^2(\Omega) \times C^1(\Omega) \rightarrow \mathbb{R}^3$ be the differential operator of the Navier-Stokes equations (3.3) and (3.4), namely,

$$\tilde{\mathcal{L}} \begin{bmatrix} \mathbf{u} \\ p \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{u}} - \nu \Delta \mathbf{u} + (\mathbf{u} - \mathbf{u}_d) \cdot \nabla \mathbf{u} + \nabla p \\ \nabla \cdot \mathbf{u} \end{bmatrix} \quad (3.9)$$

Then the residual form of the Navier-Stokes equations can be written as,

$$\tilde{\mathcal{R}}(\mathbf{u}, p) = \tilde{\mathcal{L}} \begin{bmatrix} \mathbf{u} \\ p \end{bmatrix} - \begin{bmatrix} \mathbf{f} \\ 0 \end{bmatrix} \quad (3.10)$$

The Picard-linearized steady-state adjoint operator of the Navier-Stokes equations $\tilde{\mathcal{L}}^*: \mathbf{C}^2(\Omega) \times C^1(\Omega) \rightarrow \mathbb{R}^3$ writes,

$$\tilde{\mathcal{L}}^* \begin{bmatrix} \delta \mathbf{u} \\ \delta p \end{bmatrix} = \begin{bmatrix} -\nu \Delta \delta \mathbf{u} - (\mathbf{u} - \mathbf{u}_d) \cdot \nabla \delta \mathbf{u} - \nabla \delta p \\ -\nabla \cdot \delta \mathbf{u} \end{bmatrix} \quad (3.11)$$

for a given velocity field \mathbf{u} of the primal solution. Given also a stabilization second order tensor $\in \mathfrak{S}^2$, the VMS-stabilized version of variational formulation in Eq. (3.8) writes,

$$\begin{aligned} & \left\langle \delta \mathbf{u}, \frac{\partial \mathbf{u}}{\partial t} \right\rangle_{0,\Omega} + \langle \nu \nabla \delta \mathbf{u}, \nabla \mathbf{u} \rangle_{0,\Omega} + \langle \delta \mathbf{u}, (\mathbf{u} - \mathbf{u}_d) \cdot \nabla \mathbf{u} \rangle_{0,\Omega} - \langle \delta \mathbf{u}, \nabla p \rangle_{0,\Omega} - \langle \delta p, \nabla \cdot \mathbf{u} \rangle_{0,\Omega} - \\ & - \left\langle \tilde{\mathcal{L}}^* \begin{bmatrix} \mathbf{u} \\ \delta p \end{bmatrix}, \boldsymbol{\tau} \cdot \tilde{\mathcal{R}}(\mathbf{u}, p) \right\rangle_{0,\Omega} = \langle \delta \mathbf{u}, \mathbf{f} \rangle_{0,\Omega} + \langle \delta \mathbf{u}, \mathbf{t} \rangle_{0,\Gamma_e}. \end{aligned} \quad (3.12)$$

The stabilization tensor $\boldsymbol{\tau} = \tau_{\alpha\beta} \mathbf{e}_1 \otimes \mathbf{e}_2$ is chosen diagonal, that is, $\tau_{12} = \tau_{21} = 0, \tau_{11} = \tau_m$ and $\tau_{22} = \tau_c$. In this way, the VMS-stabilized variational formulation in Eq. (3.12) becomes,

$$\begin{aligned} & \langle \delta \mathbf{u}, \dot{\mathbf{u}} \rangle_{0,\Omega} - \langle \Delta \delta \mathbf{u}, \nu \tau_m \dot{\mathbf{u}} \rangle_{0,\Omega} - \langle (\mathbf{u} - \mathbf{u}_d) \cdot \nabla \delta \mathbf{u}, \tau_m \dot{\mathbf{u}} \rangle_{0,\Omega} - \langle \nabla \delta p, \tau_m \dot{\mathbf{u}} \rangle_{0,\Omega} + \langle \nabla \delta \mathbf{u}, \nu \nabla \mathbf{u} \rangle_{0,\Omega} + \\ & \langle \delta \mathbf{u}, (\mathbf{u} - \mathbf{u}_d) \cdot \nabla \mathbf{u} \rangle_{0,\Omega} - \langle \delta \mathbf{u}, \nabla p \rangle_{0,\Omega} - \langle \delta p, \nabla \cdot \mathbf{u} \rangle_{0,\Omega} - \langle \Delta \delta \mathbf{u}, \nu^2 \tau_m \Delta \mathbf{u} \rangle_{0,\Omega} - \\ & \langle (\mathbf{u} - \mathbf{u}_d) \cdot \nabla \delta \mathbf{u}, \nu \tau_m \Delta \mathbf{u} \rangle_{0,\Omega} - \langle \nabla \delta p, \nu \tau_m \Delta \mathbf{u} \rangle_{0,\Omega} + \langle \Delta \delta \mathbf{u}, \nu \tau_m (\mathbf{u} - \mathbf{u}_d) \cdot \nabla \mathbf{u} \rangle_{0,\Omega} + \\ & \langle \nabla \delta p, \tau_m (\mathbf{u} - \mathbf{u}_d) \cdot \nabla \mathbf{u} \rangle_{0,\Omega} + \langle \Delta \delta \mathbf{u}, \nu \tau_m \nabla p \rangle_{0,\Omega} + \langle (\mathbf{u} - \mathbf{u}_d) \cdot \nabla \delta \mathbf{u}, \tau_m \nabla p \rangle_{0,\Omega} + \\ & \langle \nabla \delta p, \tau_m \nabla p \rangle_{0,\Omega} + \langle (\mathbf{u} - \mathbf{u}_d) \cdot \nabla \delta \mathbf{u}, \tau_m (\mathbf{u} - \mathbf{u}_d) \cdot \nabla \mathbf{u} \rangle_{0,\Omega} + \langle \nabla \cdot \delta \mathbf{u}, \tau_c \nabla \cdot \mathbf{u} \rangle_{0,\Omega} = \\ & \langle \Delta \delta \mathbf{u}, \nu \tau_m \mathbf{f} \rangle_{0,\Omega} + \langle \mathbf{u} \cdot \nabla \delta \mathbf{u}, \tau_m \mathbf{f} \rangle_{0,\Omega} + \langle \nabla \delta p, \tau_m \mathbf{f} \rangle_{0,\Omega} + \langle \delta \mathbf{u}, \mathbf{f} \rangle_{0,\Omega} + \langle \delta \mathbf{u}, \mathbf{t} \rangle_{0,\Gamma_e}. \end{aligned} \quad (3.13)$$

3.3. Spatial Discretization

For the spatial discretization of the problem, we apply the following assumptions and conditions:

- Velocity and pressure fields are discretized with the standard linear Finite Element Method (FEM) on triangles.
- The Bubnov-Galerkin FEM scheme is employed.
- The unknown velocity/pressures fields (\mathbf{u}, p) and their variation $(\delta \mathbf{u}, \delta p)$ are discretized using the same basis functions $(\boldsymbol{\varphi}_i, \phi_i)$, that is,

$$\begin{aligned}
\begin{bmatrix} u_h \\ p_h \end{bmatrix} &= \sum_{i=1}^n \begin{bmatrix} \psi_i & 0 & 0 \\ 0 & \psi_i & 0 \\ 0 & 0 & \psi_i \end{bmatrix} \begin{bmatrix} \hat{u}_{2i-1} \\ \hat{u}_{2i} \\ \hat{p}_i \end{bmatrix}, \\
\begin{bmatrix} \delta u_h \\ \delta p_h \end{bmatrix} &= \sum_{i=1}^n \begin{bmatrix} \psi_i & 0 & 0 \\ 0 & \psi_i & 0 \\ 0 & 0 & \psi_i \end{bmatrix} \begin{bmatrix} \delta \hat{u}_{2i-1} \\ \delta \hat{u}_{2i} \\ \delta \hat{p}_i \end{bmatrix},
\end{aligned} \tag{3.14}$$

where φ_i are the linear basis functions at the element's parametric space. Let,

$$\boldsymbol{\varphi}_i = \varphi_{\text{mod}_2^i} \mathbf{e}_{\text{mod}_2^{i+1}+1} \tag{3.15}$$

be the vector-valued basis function for the discretization for the discretization of the velocity field. Also, (\hat{u}_i, \hat{p}_i) and $(\delta \hat{u}_i, \delta \hat{p}_i)$ stands for the Degrees of Freedom (DOFs) of the unknown and the test fields, respectively, and $n \in \mathbb{N}$ is the number of nodes in the mesh. Subscript h indicates the smallest element length size within the computational mesh.

Substituting Eq. (3.14) in (3.13) and taking the variation with respect to DOFs $(\delta \hat{u}_i, \delta \hat{p}_i)$ the following discrete residual systems of equations $\hat{\mathbf{R}}(\mathbf{u}, \mathbf{p})$ and $\hat{\mathcal{R}}(\mathbf{u}, \mathbf{p})$, respectively, are obtained,

$$\begin{aligned}
\hat{R}_i(\mathbf{u}, \mathbf{p}) &= \frac{\partial(\mathcal{B})}{\partial \delta \hat{u}_i} = \langle \varphi_i, \dot{\mathbf{u}} \rangle_{0,\Omega} - \langle \Delta \varphi_i, \nu \tau_m \dot{\mathbf{u}} \rangle_{0,\Omega} - \langle (\mathbf{u} - \mathbf{u}_d) \cdot \nabla \varphi_i, \tau_m \dot{\mathbf{u}} \rangle_{0,\Omega} + \\
&\quad \langle \nabla \varphi_i, \nu \nabla \mathbf{u} \rangle_{0,\Omega} + \langle \varphi_i \cdot (\mathbf{u} - \mathbf{u}_d) \cdot \nabla \mathbf{u} \rangle_{0,\Omega} - \langle \varphi_i, \nabla p \rangle_{0,\Omega} - \langle \Delta \varphi_i, \nu^2 \tau_m \Delta \mathbf{u} \rangle_{D,\Omega} - \\
&\quad \langle (\mathbf{u} - \mathbf{u}_d) \cdot \nabla \varphi_i, \nu \tau_m \Delta \mathbf{u} \rangle_{0,\Omega} + \langle \Delta \varphi_i, \nu \tau_m (\mathbf{u} - \mathbf{u}_d) \cdot \nabla \mathbf{u} \rangle_{0,\Omega} + \langle \Delta \varphi_i, \nu \tau_m \nabla p \rangle_{0,\Omega} + \\
&\quad \langle (\mathbf{u} - \mathbf{u}_d) \cdot \nabla \varphi_i, \tau_m \nabla p \rangle_{0,\Omega} + \langle (\mathbf{u} - \mathbf{u}_d) \cdot \nabla \varphi_i, \tau_m (\mathbf{u} - \mathbf{u}_d) \cdot \nabla \mathbf{u} \rangle_{0,\Omega} + \\
&\quad \langle \nabla \cdot \varphi_i, \tau_c \nabla \cdot \mathbf{u} \rangle_{0,\Omega} - \langle \Delta \varphi_i, \nu \tau_m \mathbf{f} \rangle_{0,\Omega} - \langle (\mathbf{u} - \mathbf{u}_d) \cdot \nabla \varphi_i, \tau_m \mathbf{f} \rangle_{0,\Omega} - \\
&\quad \langle \varphi_i, \mathbf{f} \rangle_{0,\Omega} - \left\langle \varphi_i, \mathbf{t} \right\rangle_{0,\Gamma_c},
\end{aligned} \tag{3.16}$$

$$\begin{aligned}
\hat{\mathcal{R}}_i(\mathbf{u}, \mathbf{p}) &= \frac{\partial(\mathcal{B})}{\partial \delta \hat{p}_i} = -\langle \nabla \varphi_i, \tau_m \dot{\mathbf{u}} \rangle_{0,\Omega} - \langle \varphi_i, \nabla \cdot \mathbf{u} \rangle_{0,\Omega} - \langle \nabla \varphi_i, \nu \tau_m \Delta \mathbf{u} \rangle_{0,\Omega} + \\
&\quad \langle \nabla \varphi_i, \tau_m (\mathbf{u} - \mathbf{u}_d) \cdot \nabla \mathbf{u} \rangle_{0,\Omega} + \langle \nabla \varphi_i, \tau_m \nabla p \rangle_{0,\Omega} - \langle \nabla \varphi_i, \tau_m \mathbf{f} \rangle_{0,\Omega}
\end{aligned} \tag{3.17}$$

Applying the approximations in Eq. (3.14) into residual Eqs. (3.16) and (3.17) one obtains the following matrix-vector expressions for the dynamic residual equations,

$$\begin{bmatrix} \hat{\mathbf{R}}(\hat{\mathbf{u}}, \hat{\mathbf{p}}) \\ \hat{\mathcal{R}}(\hat{\mathbf{u}}, \hat{\mathbf{p}}) \end{bmatrix} = \begin{bmatrix} \mathbf{M}(\hat{\mathbf{u}}) & 0 \\ \mathcal{M} & 0 \end{bmatrix} \begin{bmatrix} \hat{\mathbf{u}} \\ \hat{\mathbf{p}} \end{bmatrix} + \begin{bmatrix} \mathbf{R}(\hat{\mathbf{u}}, \hat{\mathbf{p}}) \\ \mathcal{R}(\hat{\mathbf{u}}, \hat{\mathbf{p}}) \end{bmatrix} - \begin{bmatrix} \hat{\mathbf{F}}(\hat{\mathbf{u}}) \\ \hat{\mathcal{F}} \end{bmatrix} \tag{3.18}$$

where $\hat{\mathbf{u}}$ and $\hat{\mathbf{p}}$ stand for the complete vectors of velocity and pressure DOFs, that is,

$$\begin{aligned}\hat{\mathbf{u}} &= [\hat{u}_1 \quad \cdots \quad \hat{u}_{2n}] \\ \hat{\mathbf{p}} &= [\hat{p}_1 \quad \cdots \quad \hat{p}_n]\end{aligned}\tag{3.19}$$

and $\dot{\hat{\mathbf{u}}}$ and $\dot{\hat{\mathbf{p}}}$ their time derivatives. Mass matrices $(\hat{\mathbf{u}})$, steady-state residual vectors $\mathbf{R}(\hat{\mathbf{u}}, \hat{\mathbf{p}})$ and $\mathcal{R}(\hat{\mathbf{u}}, \hat{\mathbf{p}})$ and force vectors $\hat{\mathbf{F}}(\hat{\mathbf{u}}), \hat{\mathcal{F}}$ in Eq. (3.17) are defined as follows,

$$\begin{aligned}M_{ij}(\hat{\mathbf{u}}) &= \langle \varphi_i, \varphi_j \rangle_{0,\Omega} - \langle \Delta \varphi_i, \nu \tau_m \varphi_j \rangle_{0,\Omega} - \langle (\mathbf{u}_h - \mathbf{u}_d) \cdot \nabla \varphi_i, \tau_m \varphi_j \rangle_{0,\Omega}, \\ \mathcal{M}_{ij} &= -\langle \nabla \varphi_i, \tau_m \varphi_j \rangle_{0,\Omega} \\ \bar{R}_i(\hat{\mathbf{u}}, \hat{\mathbf{p}}) &= \langle \nabla \varphi_i, \nu \nabla \mathbf{u}_h \rangle_{0,\Omega} + \langle \varphi_i, (\mathbf{u}_h - \mathbf{u}_d) \cdot \nabla \mathbf{u}_h \rangle_{0,\Omega} - \langle \varphi_i, \nabla p_h \rangle_{0,\Omega} - \\ &\quad \langle \Delta \varphi_i, \nu^2 \tau_m \Delta \mathbf{u}_h \rangle_{0,\Omega} - \langle (\mathbf{u}_h - \mathbf{u}_d) \cdot \nabla \varphi_i, \nu \tau_m \Delta \mathbf{u}_h \rangle_{0,\Omega} + \\ &\quad \langle \Delta \varphi_i, \nu \tau_m (\mathbf{u}_h - \mathbf{u}_d) \cdot \nabla \mathbf{u}_h \rangle_{0,\Omega} + \langle \Delta \varphi_i, \nu \tau_m \nabla p_h \rangle_{0,\Omega} + \\ &\quad \langle (\mathbf{u}_h - \mathbf{u}_d) \cdot \nabla \varphi_i, \tau_m \nabla p_h \rangle_{0,\Omega} + \langle (\mathbf{u}_h - \mathbf{u}_d) \cdot \nabla \varphi_i, \tau_m (\mathbf{u}_h - \mathbf{u}_d) \cdot \nabla \mathbf{u}_h \rangle_{0,\Omega} + \\ &\quad \langle \nabla \cdot \varphi_i, \tau_c \nabla \cdot \mathbf{u}_h \rangle_{0,\Omega} \\ \mathcal{R}_i(\hat{\mathbf{u}}, \hat{\mathbf{p}}) &= -\langle \varphi_i, \nabla \cdot \mathbf{u}_h \rangle_{0,\Omega} - \langle \nabla \varphi_i, \nu \tau_m \Delta \mathbf{u}_h \rangle_{0,\Omega} + \langle \nabla \varphi_i, \tau_m (\mathbf{u}_h - \mathbf{u}_d) \cdot \nabla \mathbf{u}_h \rangle_{0,\Omega} + \\ &\quad \langle \nabla \varphi_i, \tau_m \nabla p_h \rangle_{0,\Omega}, \\ \hat{F}_i(\hat{\mathbf{u}}) &= \langle \Delta \varphi_i, \nu \tau_m \mathbf{f} \rangle_{0,\Omega} + \langle (\mathbf{u}_h - \mathbf{u}_d) \cdot \nabla \varphi_i, \tau_m \mathbf{f} \rangle_{0,\Omega} + \langle \varphi_i, \mathbf{f} \rangle_{0,\Omega} + \langle \varphi_i, \bar{t} \rangle_{0,\Gamma_c}, \\ \hat{\mathcal{F}}_i &= \langle \tau_m \nabla \varphi_i, \mathbf{f} \rangle_{0,\Omega},\end{aligned}\tag{3.20}$$

where, \mathbf{u}_h and p_h is the numerical approximation using the finite element method as per Eqs. (3.18).

3.4. Time Discretization

Within the so-called Bossak time integrations scheme, given two reals $\alpha_b, \gamma_b \in \mathbb{R}$ and a time discretization with time step $dt = t_{\hat{n}+1} - t_{\hat{n}}$, the following approximation of residual equations in (3.17) are made,

$$\begin{aligned}\begin{bmatrix} \hat{\mathbf{R}}(\hat{\mathbf{u}}_{\hat{n}+1}, \hat{\mathbf{p}}_{\hat{n}+1}) \\ \hat{\mathcal{R}}(\hat{\mathbf{u}}_{\hat{n}+1}, \hat{\mathbf{p}}_{\hat{n}+1}) \end{bmatrix} &= (1 - \alpha_b) \begin{bmatrix} \mathbf{M}(\hat{\mathbf{u}}_{\hat{n}+1}) & 0 \\ \mathcal{M} & 0 \end{bmatrix} \begin{bmatrix} \dot{\hat{\mathbf{u}}}_{\hat{n}+1} \\ \dot{\hat{\mathbf{p}}}_{\hat{n}+1} \end{bmatrix} + \alpha_b \begin{bmatrix} \mathbf{M}(\hat{\mathbf{u}}_{\hat{n}}) & 0 \\ \mathcal{M} & 0 \end{bmatrix} \begin{bmatrix} \dot{\hat{\mathbf{u}}}_{\hat{n}} \\ \dot{\hat{\mathbf{p}}}_{\hat{n}} \end{bmatrix} + \\ &\quad \begin{bmatrix} \mathbf{R}(\hat{\mathbf{u}}_{\hat{n}+1}, \hat{\mathbf{p}}_{\hat{n}+1}) \\ \mathcal{R}(\hat{\mathbf{u}}_{\hat{n}+1}, \hat{\mathbf{p}}_{\hat{n}+1}) \end{bmatrix} - \begin{bmatrix} \hat{\mathbf{F}}_{\hat{n}+1}(\hat{\mathbf{u}}_{\hat{n}+1}) \\ \hat{\mathcal{F}}_{\hat{n}+1} \end{bmatrix}\end{aligned}\tag{3.21}$$

$$\begin{bmatrix} \hat{\mathbf{u}}_{\hat{n}+1} \\ \hat{\mathbf{p}}_{\hat{n}+1} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{u}}_{\hat{n}} \\ \hat{\mathbf{p}}_{\hat{n}} \end{bmatrix} + dt(1 - \gamma_b) \begin{bmatrix} \dot{\hat{\mathbf{u}}}_{\hat{n}} \\ \dot{\hat{\mathbf{p}}}_{\hat{n}} \end{bmatrix} + dt\gamma_b \begin{bmatrix} \dot{\hat{\mathbf{u}}}_{\hat{n}+1} \\ \dot{\hat{\mathbf{p}}}_{\hat{n}+1} \end{bmatrix}.\tag{3.22}$$

The subscript \hat{n} indicates the time instance at which the quantities are evaluated, that is for example $\hat{\mathbf{u}}_{\hat{n}} = \hat{\mathbf{u}}(t_{\hat{n}})$. Solving Eqs. (3.20) for the time derivatives of the unknown fields at the

current time instance $t_{\hat{n}+1}$ and substituting them into (3.19) one arrives at a nonlinear residual equation which is to be solved for the pair $(\hat{\mathbf{u}}_{\hat{n}+1}, \hat{\mathbf{p}}_{\hat{n}+1})$ at the current time instance, namely,

$$\begin{aligned} \begin{bmatrix} \hat{\mathbf{R}}(\hat{\mathbf{u}}_{\hat{n}+1}, \hat{\mathbf{p}}_{\hat{n}+1}) \\ \hat{\mathcal{R}}(\hat{\mathbf{u}}_{\hat{n}+1}, \hat{\mathbf{p}}_{\hat{n}+1}) \end{bmatrix} &= \frac{1-\alpha_b}{\gamma_b} \begin{bmatrix} \mathbf{M}(\hat{\mathbf{u}}_{\hat{n}+1}) & \mathbf{0} \\ \mathcal{M} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{u}}_{\hat{n}+1} \\ \hat{\mathbf{p}}_{\hat{n}+1} \end{bmatrix} + \begin{bmatrix} \mathbf{R}(\hat{\mathbf{u}}_{\hat{n}+1}, \hat{\mathbf{p}}_{\hat{n}+1}) \\ \mathcal{R}(\hat{\mathbf{u}}_{\hat{n}+1}, \hat{\mathbf{p}}_{\hat{n}+1}) \end{bmatrix} - \\ \frac{1-\alpha_b}{dt\gamma_b} \begin{bmatrix} \mathbf{M}(\hat{\mathbf{u}}_{\hat{n}+1}) & \mathbf{0} \\ \mathcal{M} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{u}}_{\hat{n}} \\ \hat{\mathbf{p}}_{\hat{n}} \end{bmatrix} - \frac{1-\alpha_b-\gamma_b}{\gamma_b} \begin{bmatrix} \mathbf{M}(\hat{\mathbf{u}}_{\hat{n}+1}) & \mathbf{0} \\ \mathcal{M} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \dot{\hat{\mathbf{u}}}_{\hat{n}} \\ \dot{\hat{\mathbf{p}}}_{\hat{n}} \end{bmatrix} - \\ \begin{bmatrix} \hat{\mathbf{F}}_{\hat{n}+1}(\hat{\mathbf{u}}_{\hat{n}+1}) \\ \hat{\mathcal{F}}_{\hat{n}+1} \end{bmatrix}. \end{aligned} \quad (3.23)$$

As equation system in Eq. (3.21) is highly nonlinear on $(\hat{\mathbf{u}}_{\hat{n}+1}, \hat{\mathbf{p}}_{\hat{n}+1})$, it is not fully linearized but only the actual convection term in Eq. (3.12) is exactly linearized. For all other terms a Picard linearization is assumed. Let $(\hat{\mathbf{u}}_{\hat{n},\hat{l}}, \hat{\mathbf{p}}_{\hat{n},\hat{l}})$ be the solution at time step \hat{n} and nonlinear iteration \hat{l} . Within the employed quasi-Newton linearization, for all the terms which depend on $(\hat{\mathbf{u}}_{\hat{n}+1,\hat{l}}, \hat{\mathbf{p}}_{\hat{n}+1,\hat{l}})$ but the actual convection term, the Picard linearization reads,

$$\begin{bmatrix} \hat{\mathbf{u}}_{\hat{n}+1,0} \\ \hat{\mathbf{p}}_{\hat{n}+1,0} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{u}}_{\hat{n}} \\ \hat{\mathbf{p}}_{\hat{n}} \end{bmatrix} \quad (3.24)$$

$$\begin{bmatrix} \hat{\mathbf{u}}_{\hat{n}+1,\hat{l}} \\ \hat{\mathbf{p}}_{\hat{n}+1,\hat{l}} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{u}}_{\hat{n}+1,\hat{l}-1} \\ \hat{\mathbf{p}}_{\hat{n}+1,\hat{l}-1} \end{bmatrix}, \forall \hat{l} \geq 1 \quad (3.25)$$

where $(\hat{\mathbf{u}}_{\hat{n}}, \hat{\mathbf{p}}_{\hat{n}})$ stands for the converged solution at time instance $t_{\hat{n}}$ and $(\hat{\mathbf{u}}_{\hat{n}+1,0}, \hat{\mathbf{p}}_{\hat{n}+1,0})$ stands for the initial guess of the quasi-Newton iterative procedure at time instance $t_{\hat{n}+1}$. Therefore, the quasi-Newton linearization of residual equations in Eq. (3.21) leads to the following set of iterations,

$$\begin{aligned} \left(\frac{1-\alpha_b}{\gamma_b} \mathbf{M}(\hat{\mathbf{u}}_{\hat{n}+1,\hat{l}}) + \begin{bmatrix} \mathbf{K}(\hat{\mathbf{u}}_{\hat{n}+1,\hat{l}}, \hat{\mathbf{p}}_{\hat{n}+1,\hat{l}}) & \mathcal{C}(\hat{\mathbf{u}}_{\hat{n}+1,\hat{l}}, \hat{\mathbf{p}}_{\hat{n}+1,\hat{l}}) \\ \mathfrak{C}(\hat{\mathbf{u}}_{\hat{n}+1,\hat{l}}, \hat{\mathbf{p}}_{\hat{n}+1,\hat{l}}) & \mathcal{K}(\hat{\mathbf{u}}_{\hat{n}+1,\hat{l}}, \hat{\mathbf{p}}_{\hat{n}+1,\hat{l}}) \end{bmatrix} \right) \begin{bmatrix} \Delta_{\hat{n}+1,\hat{l}} \\ \Delta_{\hat{n}+1,\hat{l}} \hat{\mathbf{p}} \end{bmatrix} = \\ - \begin{bmatrix} \hat{\mathbf{R}}(\hat{\mathbf{u}}_{\hat{n}+1,\hat{l}}, \hat{\mathbf{p}}_{\hat{n}+1,\hat{l}}) \\ \hat{\mathcal{R}}(\hat{\mathbf{u}}_{\hat{n}+1,\hat{l}}, \hat{\mathbf{p}}_{\hat{n}+1,\hat{l}}) \end{bmatrix}, \end{aligned} \quad (3.26)$$

where $\Delta_{\hat{n}+1,\hat{l}} \hat{\mathbf{u}} = \hat{\mathbf{u}}_{\hat{n}+1,\hat{l}+1} - \hat{\mathbf{u}}_{\hat{n}+1,\hat{l}}$. Matrices $\mathbf{K}(\hat{\mathbf{u}}, \hat{\mathbf{p}})$, $\mathcal{C}(\hat{\mathbf{u}}, \hat{\mathbf{p}})$, $\mathfrak{C}(\hat{\mathbf{u}}, \hat{\mathbf{p}})$ and $\mathcal{K}(\hat{\mathbf{u}}, \hat{\mathbf{p}})$ are defined as follows,

$$\begin{aligned}
K_{ij}(\hat{\mathbf{u}}, \hat{\mathbf{p}}) &= \frac{\partial \bar{R}_i(\hat{\mathbf{u}}, \hat{\mathbf{p}})}{\partial \hat{u}_j} = \langle \nabla \varphi_i, \nu \nabla \varphi_j \rangle_{0,\Omega} + \langle \varphi_i, \varphi_j \cdot \nabla \mathbf{u}_h \rangle_{0,\Omega} + \\
&\quad \langle \varphi_i, (\mathbf{u}_h - \mathbf{u}_d) \cdot \nabla \varphi_j \rangle_{0,\Omega} - \langle \Delta \varphi_i, \nu^2 \tau_m \Delta \varphi_j \rangle_{0,\Omega} - \\
&\quad \langle (\mathbf{u}_h - \mathbf{u}_d) \cdot \nabla \varphi_i, \nu \tau_m \Delta \varphi_j \rangle_{0,\Omega} + \langle \Delta \varphi_i, \nu \tau_m (\mathbf{u}_h - \mathbf{u}_d) \cdot \nabla \varphi_j \rangle_{0,\Omega} + \\
&\quad \langle (\mathbf{u}_h - \mathbf{u}_d) \cdot \nabla \varphi_i, \tau_m (\mathbf{u}_h - \mathbf{u}_d) \cdot \nabla \varphi_j \rangle_{0,\Omega} + \langle \nabla \cdot \varphi_i, \tau_c \nabla \cdot \varphi_j \rangle_{0,\Omega}
\end{aligned} \tag{3.27}$$

$$\begin{aligned}
C_{ij}(\hat{\mathbf{u}}, \hat{\mathbf{p}}) &= \frac{\partial \bar{R}_i(\hat{\mathbf{u}}, \hat{\mathbf{p}})}{\partial \hat{p}_j} = \langle \varphi_i, \nabla \varphi_j \rangle_{0,\Omega} + \langle \Delta \varphi_i, \nu \tau_m \nabla \varphi_j \rangle_{0,\Omega} + \\
&\quad \langle (\mathbf{u}_h - \mathbf{u}_d) \cdot \nabla \varphi_i, \tau_m \nabla \varphi_j \rangle_{0,\Omega},
\end{aligned} \tag{3.28}$$

$$\mathfrak{C}_{ij}(\hat{\mathbf{u}}, \hat{\mathbf{p}}) = \frac{\partial \bar{R}_i(\hat{\mathbf{u}}, \hat{\mathbf{p}})}{\partial \hat{u}_j} = -\langle \varphi_i, \nabla \cdot \varphi_j \rangle_{0,\Omega} - \langle \nabla \varphi_i, \nu \tau_m \Delta \varphi_j \rangle_{0,\Omega} + \tag{3.29}$$

$$\mathcal{K}_{ij}(\hat{\mathbf{u}}, \hat{\mathbf{p}}) = \frac{\partial \bar{R}_i(\hat{\mathbf{u}}, \hat{\mathbf{p}})}{\partial \hat{p}_j} = \langle \nabla \varphi_i, \tau_m \nabla \varphi_j \rangle_{0,\Omega} \tag{3.30}$$

Unconditional stability for the Bossak time integration algorithm can be ensured by choosing $\gamma_b = 1/2 - \alpha_b$.

3.5. Vortex shedding

When a fluid like air or water flows past a bluff (rather than streamlined) body at specific velocities (which vary with the size and shape of the body), an oscillating flow phenomenon known as vortex shedding occurs. In this type of flow, a Kármán vortex street is formed when vortices form at the back of the body and periodically detach from the sides of the body. On the downstream side of the object, low-pressure vortices form intermittently as the fluid flows past. Moving toward the area of lowest pressure is the natural tendency of the object. Resonance occurs when the bluff structure begins to vibrate with harmonic oscillations driven by the energy of the flow if it is not mounted rigidly and if the frequency of vortex shedding is the same as the resonance frequency of the structure.

3.5.1. Governing Equation

The frequency at which vortex shedding takes place for an infinite cylinder is related to the Strouhal number by the following equation:

$$St = \frac{fD}{V} \quad (3.31)$$

Where St is the dimensionless Strouhal number, f is the vortex shedding frequency (s^{-1}), D is the diameter of the cylinder (m), and V is the flow velocity (ms^{-1}).

The Strouhal number depends on the Reynolds number Re but a value of 0.22 is commonly used. Over four orders of magnitude in Reynolds number, from 100 to 100000, the Strouhal number varies only between 0.18 and 0.22

3.6. Numerical Example

In this section, Cane Multiphysics is used to solve a laminar vortex-shedding flow past a circular cylinder in 2D. A benchmark Von-Karman vortices problem is chosen from bibliography in order to produce and compare results from the two solutions.

In the following figure, the computational domain of the problem is presented, along with the characteristic quantities and boundary conditions that describe it,

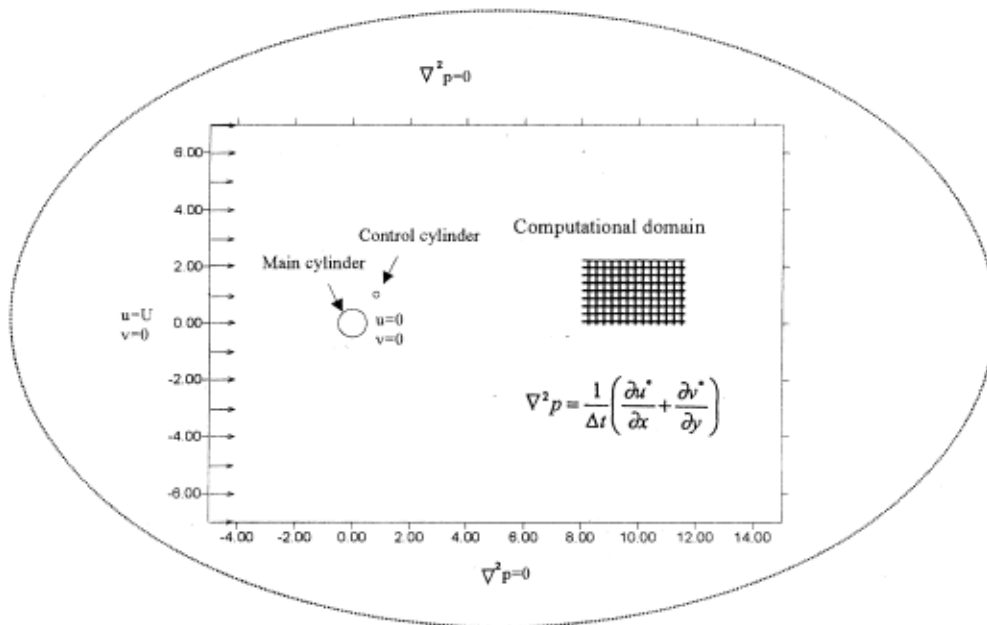


Figure 32

It is assumed, that the computational domain is limited to the finite region. As a consequence, only the inflow boundary condition, namely, $\mathbf{u} = \mathbf{1}$ and $\mathbf{v} = \mathbf{0}$, is needed as the boundary requirement of the computational domain, where u, v are the fluid velocities in

the x and y directions respectively. The boundary condition of the fixed body in flow is set as no-slip boundary, namely, $\mathbf{u} = \mathbf{0}$ and $\mathbf{v} = \mathbf{0}$.

The coefficient of drag, coefficient of lift are found from the following equations:

$$C_d = \frac{F_d}{1/2 \rho u_0^2 D}, \quad C_l = \frac{F_l}{1/2 \rho u_0^2 D} \quad (3.31)$$

$$F_d = \oint_s p_s n_y ds - \oint_x \tau_s n_x ds \quad (3.32)$$

$$F_l = \oint_s p_s n_x ds + \oint_s \tau_s n_y ds \quad (3.33)$$

where u_0 is the fluid velocity, F_d is the drag force, ρ is the mass density, D is the characteristic dimension, F is the lift force, f is the frequency of the oscillation, τ_s is the shear force acting on the body, p_s is the pressure acting on the body, and n_x and n_y are the direction cosines in the x - and y -coordinates, respectively.

The present model was tested at Reynolds number $Re = 80$ (laminar flow). The time domain is $[0, 200]$ s and a time-step of 0.0125 s is used for the iterative procedure.

Preprocessing with GiD

In this section a description of how to make a computational model of a fluid mechanics problem is presented. Basic procedures are the same as in the structural problems, so they will not be discussed.

Dirichlet boundary conditions

Firstly, we create the geometry of the domain making the necessary Boolean surface operations in order to separate the cylinder. Then, the two boundary conditions are imposed:

- 1) Inflow boundary condition.
- 2) No-slip boundary condition.

To specify the Dirichlet boundary conditions select Data → Conditions → Constraints. Select lines (line icon) as selection type and select Fluid-Dirichlet-Over-Lines. To apply the no-slip boundary conditions set the x and y component of the velocity to 0.0 on top and bottom line and on the circle.

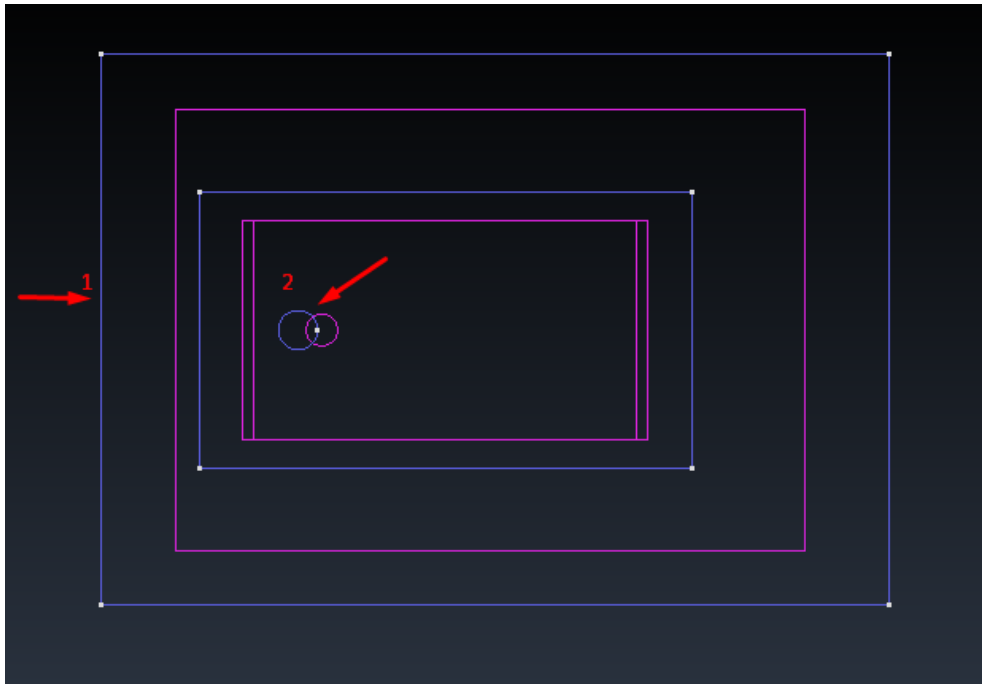


Figure 33

Selection of postprocessing boundary

In the end of the analysis, the Lift Force near the cylinder has to be calculated, in order to compare it with the results found in the reference solution. In order to do that in GiD, the user has to specify that boundary. To do that select Data->Constraints->Fluid-Post-Processing-Over-Lines and select the circular boundary

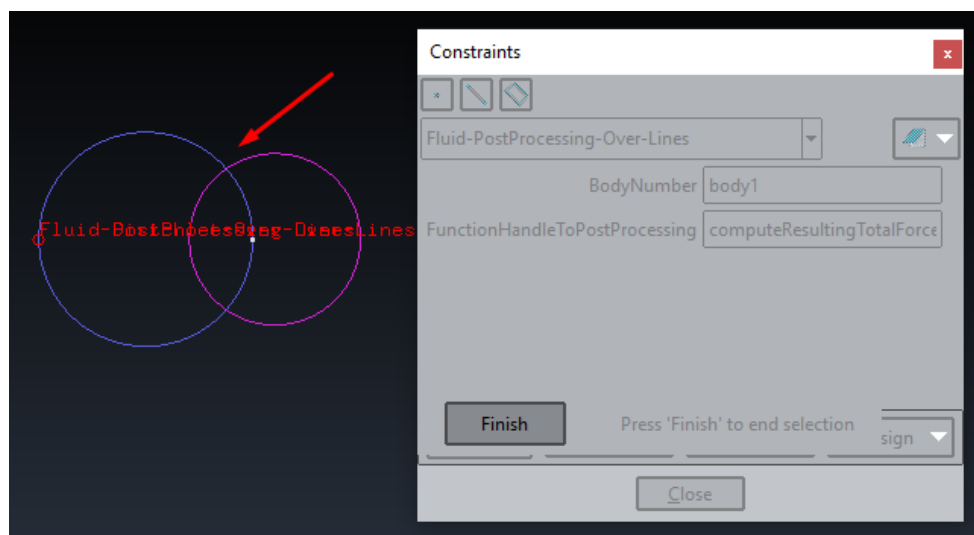
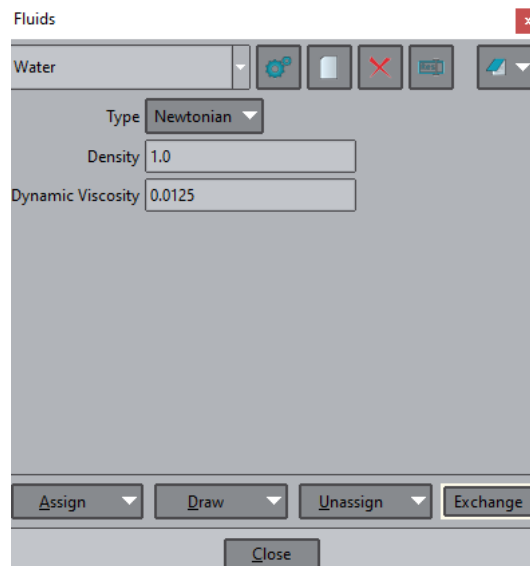


Figure 34

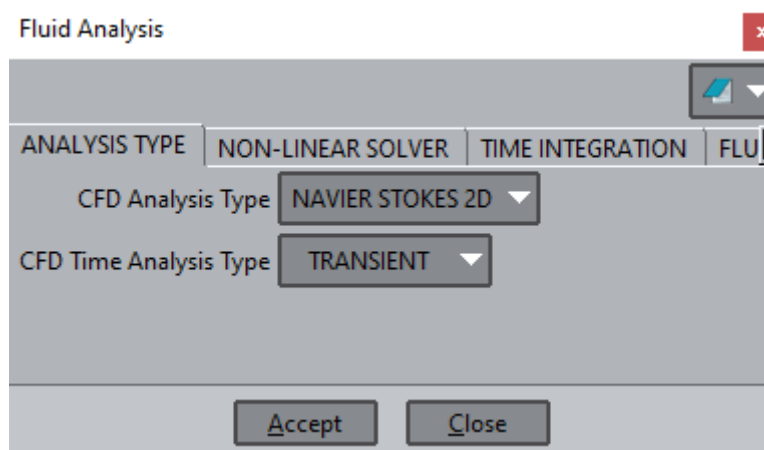
Selection of the material properties

In order to select the material properties, select Data → Materials → Fluids. There one can select the default material Water from the drop-down menu or just change the given parameters to adjust material properties. Apply the material to the geometry of the problem by selecting Assign → Surfaces and choose the surface of defining the problem's domain.

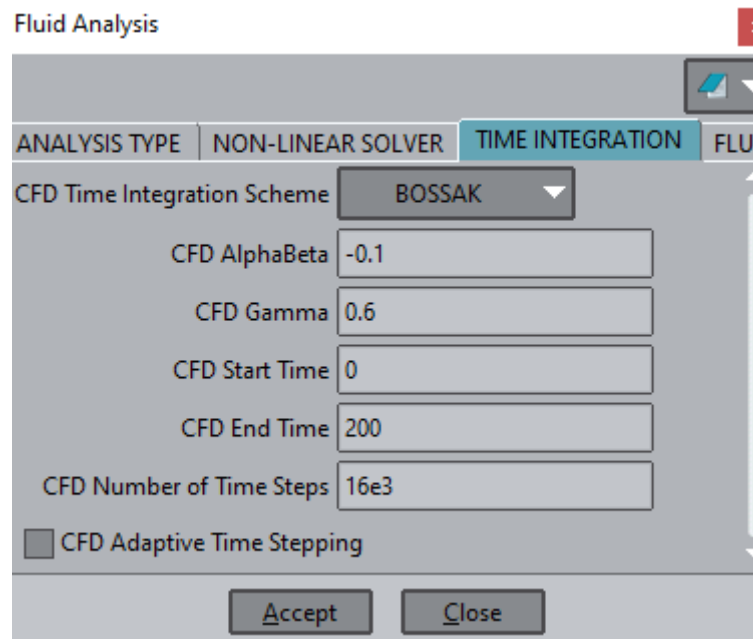


Selection of the analysis type and solver setup

To select the analysis type and setup the solver, select Data → Problem Data → Fluid Analysis. In this window the user can select 2D or 3D analysis and furthermore choose between a steady state or a transient analysis. Many more settings are possible, specifically on the time integration schemes and Gauss integration, but they are not needed for this simple case in which the default options are sufficient.



Next, the time domain of the analysis has to be specified, along with the number of time steps. In this case, a time-step of 0.0125s is chosen, so that means that the analysis will have 16000 total time-steps. A small time-step is very beneficial in such analyses because the fluid velocity and pressure might change a lot from one stage of the analysis to the next one, and a big time-step might result in great loss of information. To do that in GiD select Data->Problem Type->Fluid Analysis->Time Integration and fill the needed fields.



Generation of the computational mesh

Three different levels of discretization were used in order to model the problem.

Firstly, the user has to specify the thickness of the boundary layer. The thin layer of fluid that forms next to a bounding surface due to fluid flow is called a boundary layer in the fields of physics and fluid mechanics. Because of the fluid's interaction with the wall, a no-slip boundary condition is imposed (zero velocity at the wall). Above the surface, the flow speed increases continuously until it reaches the bulk flow speed again. The velocity boundary layer is the relatively thin layer of fluid whose speed has not yet returned to that of the bulk flow.

To create the boundary layers select Mesh -> Boundary Layer -> 2D -> Assign. Then, pick the surface onto which the boundary layer will be imposed, which is our case is the circle with no-slip condition.

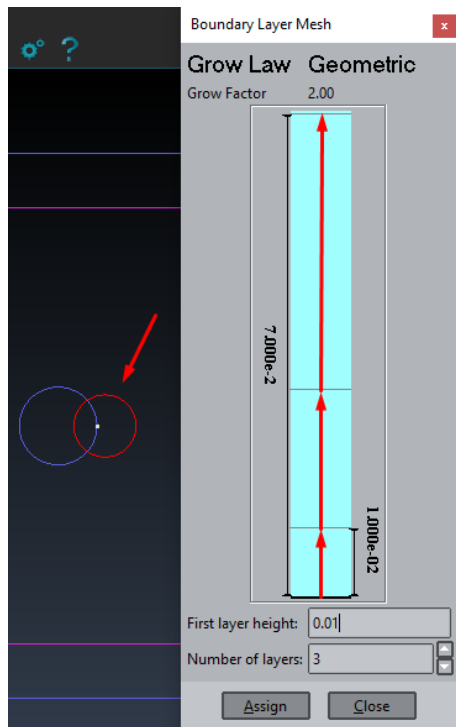


Figure 35

The length of the element used in the discretization is decreased exponentially when approaching the boundary. In this case we chose 3 layers in order to fully represent the boundary layer, as shown in the figure above.

The second level of discretization concerns the area close to the cylinder. In this area, we need a mesh dense enough, in order for the results to be accurate, because the Von-Karman vortices will take place there. In order to do that, we create a smaller surface inside the computational domain and pick a smaller element size to be used in that specific area.

To do this in GiD select Mesh -> Unstructured -> Assign sizes on surfaces. Then, the inner surface is selected and an element size of 0.3 is chosen for the discretization.

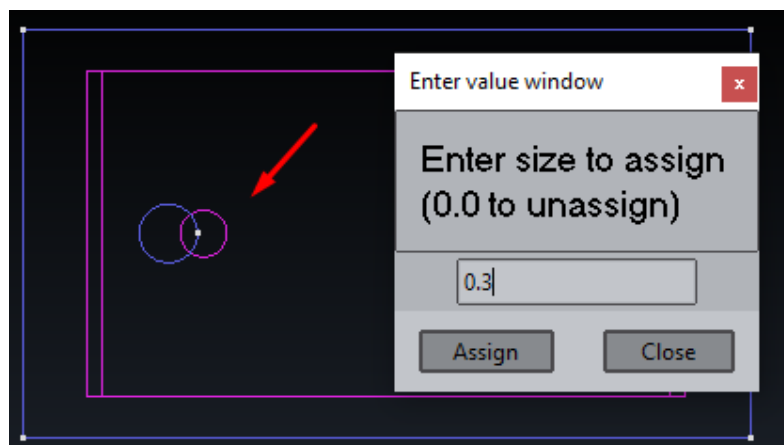


Figure 36

In the rest of the computational domain, an element size of 0.6 is used. Now everything is set up, and the mesh can be calculated. Select Calculate or press F5. The generated mesh is shown in the figure below:

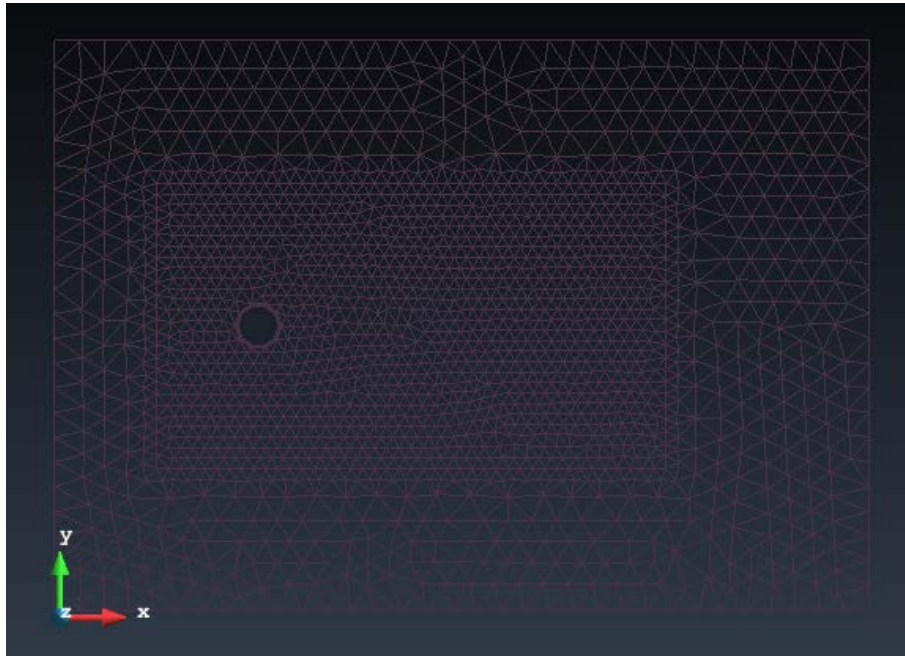


Figure 37

The different levels of discretization have been imposed correctly, also the boundary layers has been simulated sufficiently:

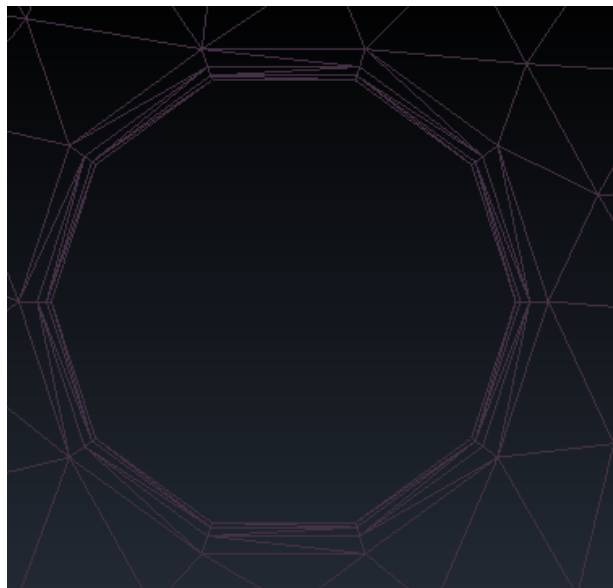


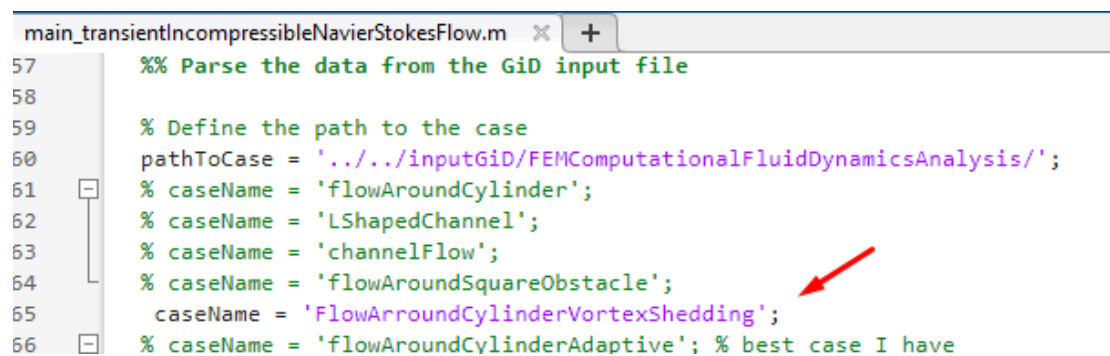
Figure 38

Set up cane for the epilysis

At this point, the geometry of the problem is properly defined, along with the boundary conditions and the mesh. The next step is to open the appropriate matlab script from the cane repository and specify how to handle our case problem.

From the cane main repository select main->mainFEM_ComputationalFluidDynamics->main_transientIncompressibleNavierStokesFlow.m. This is the matlab script which will parse the data file created from GiD.

The first thing we need to do when we open the script is to define a path for matlab in order to find the generated file from GiD, provided that the corresponding file exist in the input_GiD folder of cane repository. We create a new case with the name of the generated file, in this case 'flowAroundCylinderVortexShedding'.



```
main_transientIncompressibleNavierStokesFlow.m  x +
57  %% Parse the data from the GiD input file
58
59  % Define the path to the case
60  pathToCase = '../inputGiD/FEMComputationalFluidDynamicsAnalysis/';
61  % caseName = 'flowAroundCylinder';
62  % caseName = 'LShapedChannel';
63  % caseName = 'channelFlow';
64  % caseName = 'flowAroundSquareObstacle';
65  caseName = 'FlowAroundCylinderVortexShedding';
66  % caseName = 'flowAroundCylinderAdaptive'; % best case I have
```

Figure 39

At this point, the problem is fully set and we can run the script in order for the epilysis to take place.

Results

The time marching calculation of vortex shedding for the problem is carried out at Reynolds number of 80, as stated in the previous paragraph. The pressure and velocity fields are estimated at $t = 200s$. The figure below shows the velocity field of the reference solution:

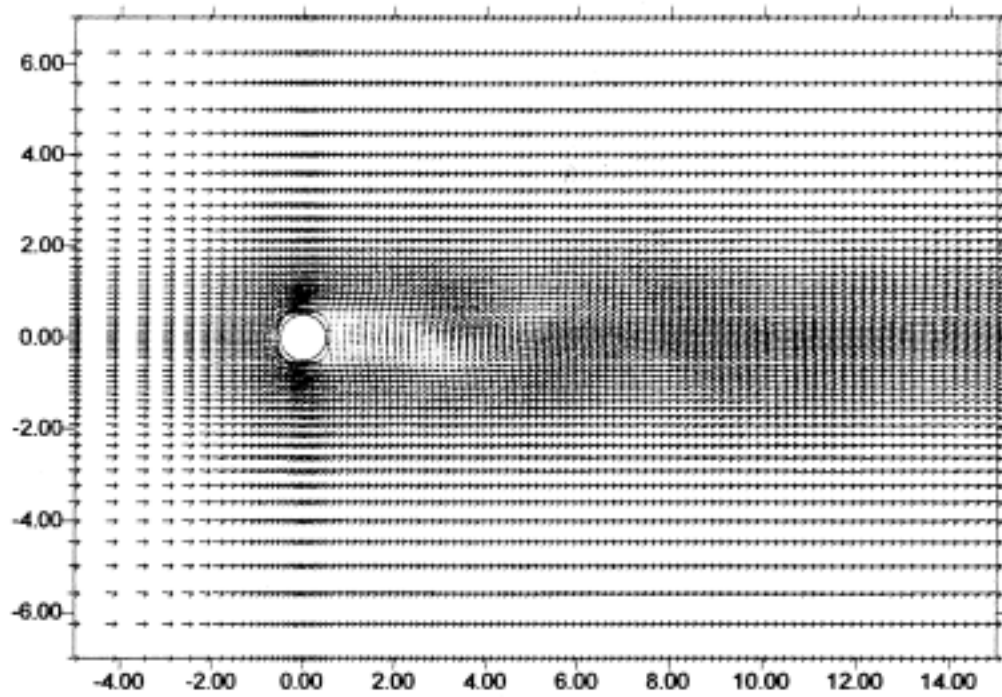


Figure 40

The velocity field at time $t = 200s$ as calculated in cane is shown in the figure below. ParaView was used for the visualization of the results.

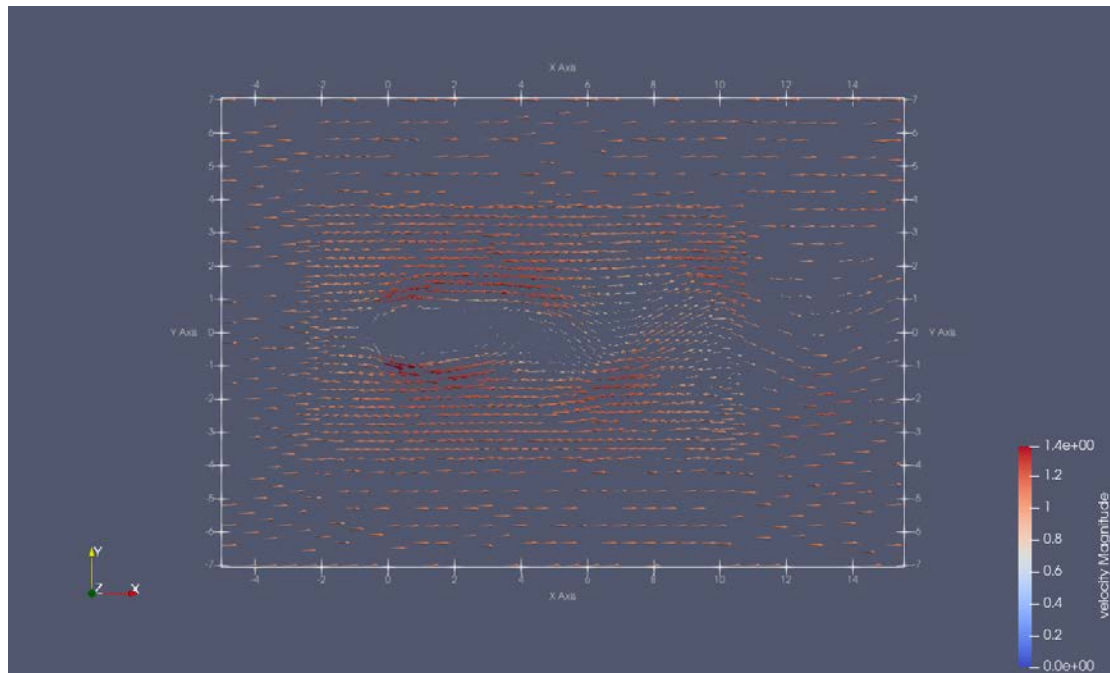


Figure 41

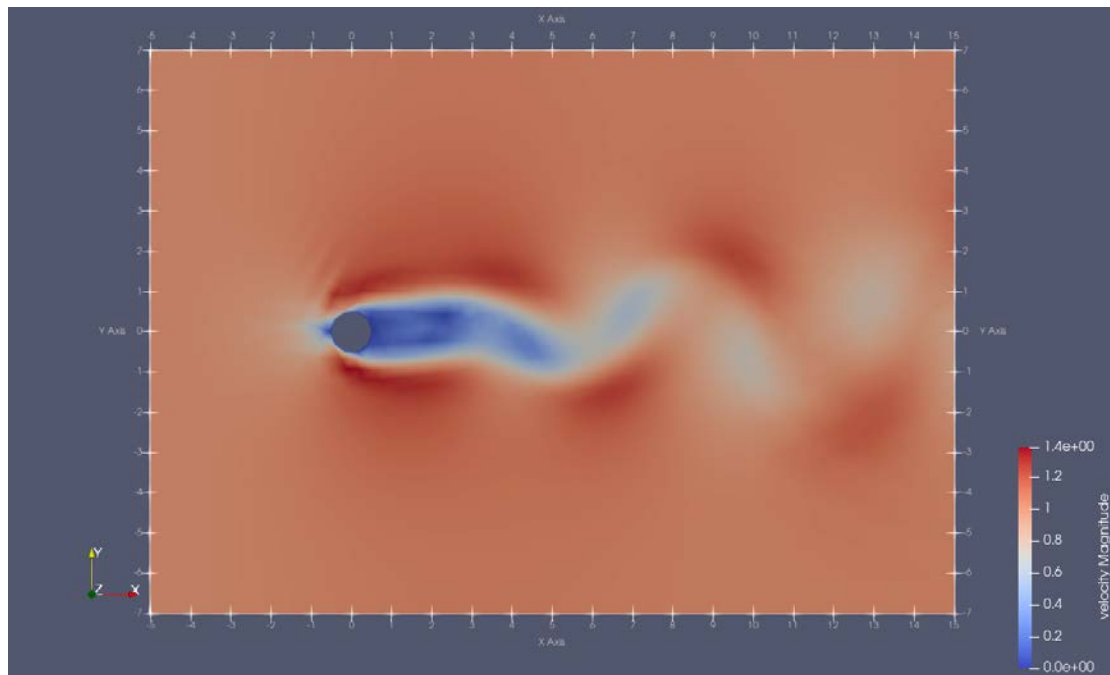


Figure 42

As the above figures depict, the numerical simulations with the present model have rendered very reasonable and satisfactory results in comparison with the reference solution.

Furthermore, the time behavior of the lift coefficient flow the flow across the circular cylinder is presented and compared between the two solutions. The following figure represents the coefficient of lift of the reference solution:

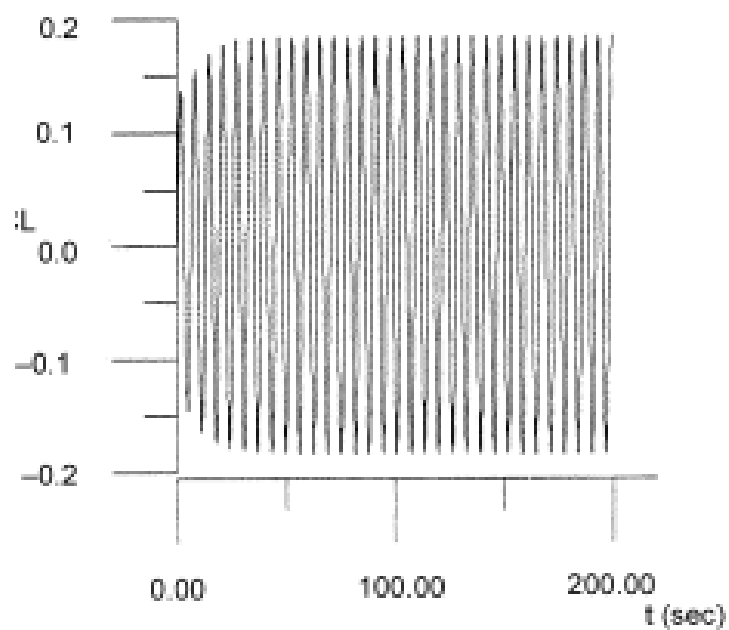


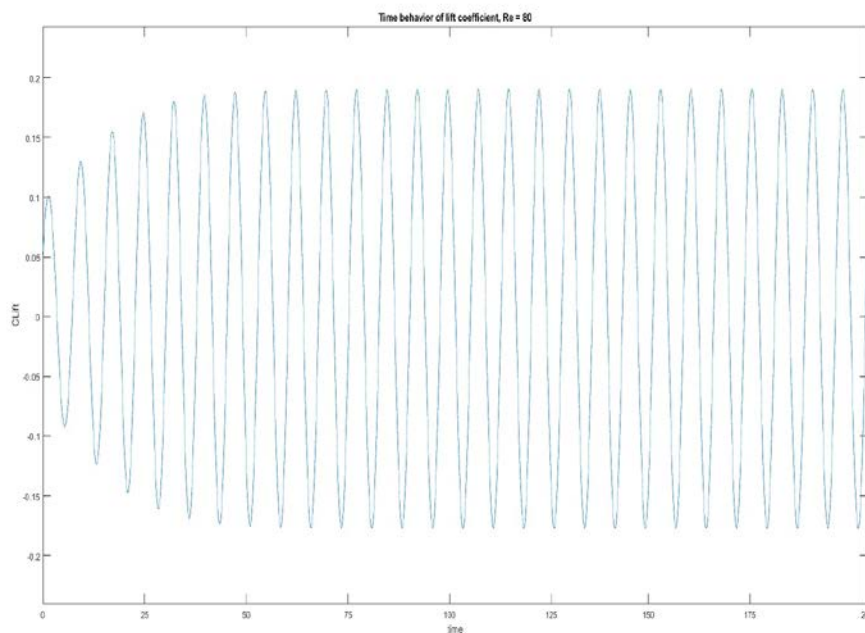
Figure 43

In order to calculate the lift coefficient through cane, the following function script has to be used:

```
%% Postprocessing

% Compute the forces acting on the domain of interest
if isstruct(propPostproc) && ~ischar(FHistory)
    forcesOnCylinder = zeros(propFldDynamics.noTimeSteps + 1, 2);
    for iTimeStep = 1:propFldDynamics.noTimeSteps + 1
        propPostproc = computePostProc ...
            (FHistory(:, iTimeStep), propAnalysis, parameters, propPostproc);
        forcesOnCylinder(iTimeStep, :) = propPostproc.valuePostProc{1}';
    end
end
```

In essence, this function used the nodes stated in GiD during pre-processing and extracted the forces acting on the cylinder into the vector 'forcesOnCylinder' which has dimensions of (numberOfTimeSteps, 2), where 2 stands for the Drag (1) and the Lift(2) force. Then after using the following equation (3.31) to calculate the coefficient, we plot the resulting vector over the time domain,



Once again, the produced results from the cane epilysis are very accurate and close to the results in the reference solution.