27th ABCM International Congress of Mechanical Engineering
December 04-08, 2023, Florianópolis-SC, Brazil

COB-2023-2068
DEVELOPMENT OF PARAMETERIZED MAPPING: A STARTING POINT FOR THE COUPLING BETWEEN THE STRUCTURAL SOLVER GIRAFFE AND THE FLUID DYNAMICS SOLVER OPENFOAM

Rodolfo Puraca
Bruno Souza Carmo
Department of Mechanical Engineering, Escola Politécnica - University of São Paulo - Brazil
rodolfo.puraca@usp.br, bruno.carmo@usp.br
Alfredo Gay Neto
Department of Structural and Geotechnical Engineering, Escola Politécnica - University of São Paulo - Brazil
alfredo.gay@usp.br

Abstract. *Fluid-structure interaction is a challenge in engineering as it involves designing structures and machines capable of withstanding the associated forces during operation. To address this, it is necessary to develop models that couple fluid dynamics with solid dynamics. One important aspect of this coupling is the transfer of loads and displacements between the fluid and solid domains, which may differ in detail and topology. The purpose of this paper is to provide a comprehensive description and substantiation of a mesh mapping technique that enables information transfer between the Giraffe structural solver and the OpenFOAM fluid dynamics solver. This technique utilizes a mesh-less mapping approach based on weighted least-squares minimization, employing radial basis functions as weights and polynomial functions as basis functions. To demonstrate the effectiveness of this approach, a test case involving an elastic strip with simple geometry and imposed load is presented. The test case focuses on observing the transfer of load and displacement between the fluid interface mesh and the structural beam mesh. The results of the test case confirm the successful achievement of the objective, showcasing the accurate transfer of beam displacements and the forces present in the fluid interface mesh.*

Keywords: *Aeroelasticity, Parameterized Mapping, CFD, FEM.*

1. INTRODUCTION

Aeroelasticity is a critical phenomenon that arises when there is an interaction between a structure and the surrounding fluid under specific flow conditions. Understanding and accounting for aeroelastic effects are essential in the design and development of various engineering structures, including aircraft, wind turbines, suspension bridges, and other systems subjected to significant aerodynamic forces and structural motion.

To computationally simulate aeroelasticity, it is necessary to solve the differential equations governing fluid dynamics and structural mechanics in a coupled manner. Achieving such a coupling typically involves integrating two distinct solvers: one specialized for fluid flow (Computational Fluid Dynamics (CFD) solver) and another for structural analysis (Computational Solid Dynamics (CSD) solver) (Yuan *et al.*, 2021). However, the coupling process is nontrivial and requires the seamless exchange of information between the external surfaces of the structure and the fluid flow domain. This exchange involves transferring forces and displacements through the mapping of these boundaries, as shown in Rendall and Allen (2008).

The OpenFOAM is an open-source library initially developed by Jasak (1996) for numerical simulation of fluid flows. It provides a flexible and extensible platform for solving complex CFD problems across a wide range of applications. Giraffe is the acronym for “Generic Interface Readily Accessible for Finite Elements”. It is a software based on the finite element method and is developed by researchers at Escola Politécnica, University of São Paulo. Implemented using the C++ programming language, Giraffe emphasizes modularity and aims to create a software structure that can be expanded and to serve as a computational platform for non-linear finite element models (Gay Neto, 2018).

The focus of this paper is the development of a parameterized mapping procedure that facilitates the coupling between the CSD solver Giraffe and the CFD solver OpenFOAM. The proposed mapping technique serves as a vital bridge between the fluid and structural domains, enabling the exchange of crucial information required for accurate aeroelastic simulations. A force distribution will be applied at the interface of the representative fluid mesh, where these loads will be mapped to the beam, causing it to deform and transferring this displacement to the fluid interface mesh. It’s important to note that the force distribution will not result from a fluid-structure coupling but will serve as a representative parameter for testing the mapping between different topological meshes.

To achieve this objective, the paper will present the mathematical formulation and implementation details of the parameterized mapping technique. The results will demonstrate the effectiveness of the proposed mapping procedure in capturing and transferring the deformation and the forces between different mesh topologies.

2. THEORY

In this section, we will present the problem that aeroelasticity imposes on an Arbitrary Lagrangian Eulerian (ALE) formulation and the parameterized mapping solution, as well as the use of Radial Basis Functions (RBF) to support the parameterization.

2.1 Aeroelasticity Problem

When employing the ALE method for discretizing space and time for the fluid, a moving or dynamic mesh approach can be used. In this approach, the fluid mesh nodes at the fluid-structure interaction interface are treated as Lagrangian nodes, being moved by structural displacements. The continuity and the momentum equations for an incompressible fluid submitted to moving or deforming grids are, as showed by Duarte *et al.* (2004),

$$\nabla.\mathbf{U} = 0, \tag{1}$$

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla.\left[\left(\mathbf{U} - \mathbf{U}_g\right)\mathbf{U}\right] = -\nabla p + \nabla.\left[\nu\nabla\mathbf{U} + \left(\nabla\mathbf{U}\right)^T\right], \tag{2}$$

where $\mathbf{U}$ is the flow field velocity, $\mathbf{U}_g$ is the moving mesh velocity, p is the pressure field and $\nu = \mu/\rho$ is the kinematic viscosity. Part of the field that is moving at the velocity $\mathbf{U}_g$ is in the intersection Γ, where the fluid-structure interaction happens, and their velocities and displacements are called $\dot{\mathbf{x}}_f$ and $\mathbf{x}_f$ respectively.

As stated in Chopra (2012), the general discretized non-linear structural part of the problem is described by

$$\mathbf{M}\ddot{\mathbf{x}}_s + \mathbf{f}_S + \mathbf{f}_D = \mathbf{F}(t), \tag{3}$$

where $\mathbf{M}$ is the mass matrix, $\mathbf{f}_S$ is the elastic (or inelastic) resisting force, $\mathbf{f}_D$ is the damping force, $\ddot{x}_s$ is the solid acceleration on degrees of freedom and $\mathbf{F}(t)$ is the aerodynamic force.

To complete the model, as showed in Gatzhammer (2014), the coupling conditions are stated on the intersection Γ by

$$\left.\begin{aligned} \sigma_s\,\mathbf{n} &= \int_\Gamma \left(-p\,\mathbf{n} + \sigma_f\,\mathbf{n}\right)\,\mathrm{dA} \\ \mathbf{x}_s &= \mathbf{x}_f \\ \dot{\mathbf{x}}_s &= \dot{\mathbf{x}}_f \end{aligned}\right\}\quad \text{on } \Gamma, \tag{4}$$

being σ_s and σ_f the structure stress tensor and the fluid viscous stress tensor, respectively, and **n** the unit vector normal to Γ.

2.2 Parameterized Mapping

The coupling equations shown in Eq. (4) work for a continuum problem, but in CFD and FEM solvers, this problem is solved discretely, and thus a consistent expression must be developed to conserve the coupling properties. There is an energy balance that must be respected in the system, where the energy released or absorbed by the structure, with the exception of the energy dissipated by damping, must be absorbed or released by the fluid.

2.2.1 Forces - Virtual Work

A method of obtaining these coupling expressions in a discrete model and independent of the topology of the fluid mesh and structure is using variational principles such as Virtual Work, which will weakly couple these conditions, as shown in Quaranta *et al.* (2005). Considering $\delta\mathbf{x}_f$ and $\delta\mathbf{x}_s$ as virtual displacements for the fluid and solid field, respectively, and that the displacement in Γ reflects the same displacement in the structure and in the fluid, the trace of these two fields must be equal to,

$$\mathrm{Tr}\left(\delta\mathbf{x}_f\right)|_\Gamma = \mathrm{Tr}\left(\delta\mathbf{x}_s\right)|_\Gamma, \tag{5}$$

and the relationship between the virtual displacements is,

$$\left(\delta\mathbf{x}_f\right)_i = \sum_{j=1}^{j_s} h_{ij}\left(\delta\mathbf{x}_s\right)_j, \tag{6}$$

where $\left(\delta\mathbf{x}_f\right)_i$ and $\left(\delta\mathbf{x}_s\right)_j$ are the discrete virtual displacements of $\delta\mathbf{x}_f$ and $\delta\mathbf{x}_s$ respectively, h_{ij} are the coefficients of the interpolation matrix H, i is the interface node number of the fluid mesh and j is the interface node number of the structural mesh. Using basis functions to approximate the space of the aerodynamic field discretization, Eq. (6) takes the form,

$$\delta \mathbf{x}_f = \sum_{i=1}^{i_f} N_i \sum_{j=1}^{j_s} h_{ij} \left(\delta \mathbf{x}_s\right)_j , \tag{7}$$

where N_i are the basis functions, i_f and i_s are the total number of interface nodes that belongs to the fluid and structural meshes. The virtual work of the aerodynamic load is,

$$\begin{aligned} \delta W_f &= \int_{\Gamma_f} \left(-p\,\mathbf{n} + \sigma_f\,\mathbf{n}\right) . \delta \mathbf{x}_f \text{ dA} \\ \delta W_f &= \int_{\Gamma_f} \left(-p\,\mathbf{n} + \sigma_f\,\mathbf{n}\right) . \sum_{i=1}^{i_f} N_i \sum_{j=1}^{j_s} h_{ij} \left(\delta \mathbf{x}_s\right)_j \text{ dA} \quad \text{where} \quad \mathbf{F}_i = \int_{\Gamma_f} \left(-p\,\mathbf{n} + \sigma_f\,\mathbf{n}\right) N_i \text{ dA} \\ \delta W_f &= \sum_{i=1}^{i_f} \mathbf{F}_i \, . \sum_{j=1}^{j_s} h_{ij} \left(\delta \mathbf{x}_s\right)_j , \end{aligned} \tag{8}$$

and the virtual work in the solid part is,

$$\delta W_s = \sum_{j=1}^{j_s} \mathbf{f}_j \, . \left(\delta \mathbf{x}_s\right)_j . \tag{9}$$

Equaling the virtual work of Eq. (8) and Eq. (9), we find the expression that relate the forces of the fluid and solid interface meshes,

$$\mathbf{f}_j = \sum_{i=1}^{i_f} \mathbf{F}_i \; h_{ij}. \tag{10}$$

At this point, it is found that the forces between the mesh of the fluid and the structure are directly related by the multiplication of the interpolation matrix H, showing that the conservation of energy that the problem of fluid-structure interaction requires occurs. However, in addition to energy conservation, it is necessary to conserve the velocity transmitted by the structure to the fluid. In this way, conservation of momentum is achieved by correctly calculating the coefficients h_{ij}.

2.2.2 Displacement - Weighted Least-Squares

The correct calculation of the coefficients can be achieved based on the idea of the strategy of minimizing the difference in the trace of the displacement field of the fluid and the structure, as shown in Eq. (5), multiplied by a weight function that correlates the two meshes, using the Weighted Least-Squares technique, i.e.

$$\text{Min } J(\mathbf{x}) = \int_{\Gamma} \phi \left(\text{Tr}\left(\delta \mathbf{x}_f\right)|_{\Gamma} - \text{Tr}\left(\delta \mathbf{x}_s\right)|_{\Gamma} \right)^2 \text{ dA}, \tag{11}$$

where ϕ is the weight function. Based on this idea, mesh-less methods can be used, as they make it possible to calculate the coefficients of the matrix H without having a mesh with connectivity between the points (Quaranta *et al.*, 2005). Instead, the meshes are seen as clouds of points, and the matrix H correlates these clouds, which allows passing displacement and velocity information between meshes with different details and topologies, which is useful in aeroelastic problems.

The basis of the mesh-less method using weighted least-squares is to reconstruct the function f from its values $f\left(\mathbf{x_{s1}}\right)$, $f\left(\mathbf{x_{s2}}\right)$, ..., $f\left(\mathbf{x_{s}}_N\right)$, where $\mathbf{x_s}_N$ are the points belonging to the interface of the structure and N is the number of points with the smallest Euclidean distance from the point $\mathbf{x_s}$ to the point under study on the interface of the fluid. First it is necessary to use a function $\hat{f}$ that builds a local approximation of f as the sum of functions with local basis $p_i(\mathbf{x_d})$, that is,

$$\hat{f} = \sum_{i=1}^{m} p_i(\mathbf{x_d}) a_i(\mathbf{x_f}) \equiv \mathbf{p}^T(\mathbf{x_d}) a(\mathbf{x_f}), \tag{12}$$

where m is the number of basis functions, $a_i(x_f)$ the coefficients and $\mathbf{x_d} = \mathbf{x_s} - \mathbf{x_f}$ the euclidean distance between the points of the fluid interface and structural interface meshes. The basis function $\mathbf{p}^T(\mathbf{x_d})$ can be either linear,

$$\mathbf{p}^T(\mathbf{x_d}) = \left(1, x_d, y_d, z_d\right)^T , \tag{13}$$

or quadratic polynomial,

$$\mathbf{p}^T(\mathbf{x_d}) = \left(1, x_d, y_d, z_d, {x_d}^2, x_d y_d, x_d z_d, {y_d}^2, y_d z_d, {z_d}^2\right)^T, \tag{14}$$

and are constructed using the coordinates of the points of the fluid interface mesh.

The coefficients a_i are calculated using the Weighted Least-Squares, which is a minimization procedure equivalent to Eq. (11), written in the form

$$\begin{aligned} \text{Min } J(\mathbf{x_f}) &= \int_\Omega \phi\left(\mathbf{x_f} - \mathbf{x_s}\right)\left(\hat{f} - f\left(\mathbf{x_s}\right)\right)^2 \, \mathrm{d}\Omega\left(\mathbf{x_s}\right), \\ \hat{f} &= \sum_{i=1}^{m} p_i(\mathbf{x_s}) a_i(\mathbf{x_f}). \end{aligned} \tag{15}$$

These equations work to find the coefficients a_i if the topologies of the structural and fluid interface are the same, then Ω is equated with Γ, and the minimization integration is done. For more general cases, where interface topologies differ, this problem is seen in its discrete form, and the integrals of Eq. (15) are transformed into summations (Quaranta *et al.*, 2005), resulting in a matrix minimization,

$$\text{Min } J(\mathbf{x_f}) = \left(\mathbf{P}a(\mathbf{x_f}) - \mathbf{f}\right)^T \Phi \left(\mathbf{P}a(\mathbf{x_f}) - \mathbf{f}\right), \tag{16}$$

where $\mathbf{P}$ is the matrix of $p_m\left(\mathbf{x_d}_N\right)$ of basis polynomial functions,

$$\mathbf{P} = \begin{bmatrix} p_1\left(\mathbf{x_d}_1\right) & p_2\left(\mathbf{x_d}_1\right) & \cdots & p_m\left(\mathbf{x_d}_1\right) \\ p_1\left(\mathbf{x_d}_2\right) & p_2\left(\mathbf{x_d}_2\right) & \cdots & p_m\left(\mathbf{x_d}_2\right) \\ \vdots & \vdots & \ddots & \vdots \\ p_1\left(\mathbf{x_d}_N\right) & p_2\left(\mathbf{x_d}_N\right) & \cdots & p_m\left(\mathbf{x_d}_N\right) \end{bmatrix}, \tag{17}$$

$\mathbf{f}$ is the vector of $\mathbf{f}_N$ that we want to approximate,

$$\mathbf{f} = \left[\mathbf{f}\left(\mathbf{x_f}_1\right) \quad \mathbf{f}\left(\mathbf{x_f}_2\right) \quad \cdots \quad \mathbf{f}\left(\mathbf{x_f}_N\right)\right], \tag{18}$$

and Φ is the weight function of the normalized distances r_N between N fluid interface points closer to the analyzed fluid mesh interface point,

$$\Phi = \begin{bmatrix} \phi\left(r_1\right) & 0 & \cdots & 0 \\ 0 & \phi\left(r_2\right) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \phi\left(r_N\right) \end{bmatrix}. \tag{19}$$

Performing the minimization of Eq. (16) and reorganizing the equation to isolate $a\left(\mathbf{x_s}\right)$, we find the coefficients that can be used in the interpolation matrix H,

$$\begin{aligned} \frac{\partial J}{\partial a\left(\mathbf{x_f}\right)} &= 0 = \left(\mathbf{P}a(\mathbf{x_f} - \mathbf{f}\right)^T \left(\Phi + \Phi^T\right) \frac{\partial\left(\mathbf{P}a\left(\mathbf{x_f}\right)\mathbf{f}\right)}{\partial a\left(\mathbf{x_f}\right)} \\ \frac{\partial J}{\partial a\left(\mathbf{x_f}\right)} &= \left(\mathbf{P}a(\mathbf{x_f} - \mathbf{f}\right)^T 2\Phi\,\mathbf{P} = 0 \\ a\left(\mathbf{x_f}\right) &= \left(\Phi\mathbf{P}^T\mathbf{P}\right)^{-1}\left(\Phi\mathbf{P}^T\mathbf{f}\right), \end{aligned} \tag{20}$$

which can be written as

$$a\left(\mathbf{x_f}\right) = A^{-1}\, b\, f, \tag{21}$$

where $A = \Phi\mathbf{P}^T\mathbf{P}$ and $b = \Phi\mathbf{P}^T$.

2.3 Radial Basis Function (RBF)

For the calculation of the ϕ weight functions, radial basis functions (RBFs) are used, which are equations whose values depend only on the normalized Euclidean input distance between the points belonging to the fluid interface mesh in relation to the points of the structural mesh. There are different types of RBFs, such as Gaussian, multiquadratic, and thin plate spline, among others, that do not have the characteristic of being supported compactly. The class of compactly

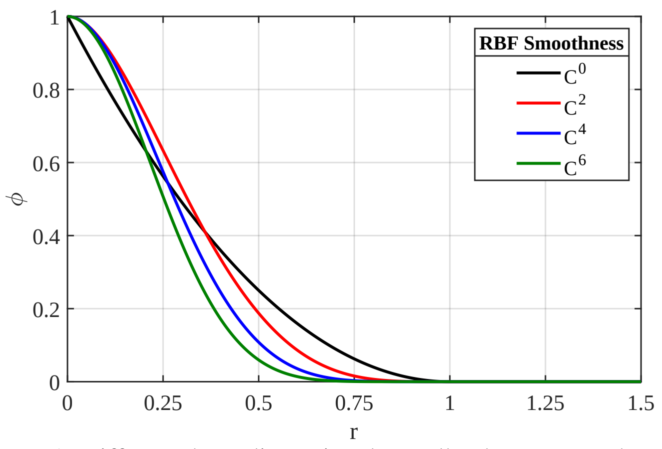


Figure 1: Different three-dimensional Wendland RBF smoothness.

supported RBFs has the functionality of presenting values only within a range of distances, which makes it possible for them to generate sparse matrices, which is computationally beneficial. In Wendland (1995), a series of compactly supported smooth RBFs was created, which received the name Wendland RBFs and have the following properties: they are monotonically decreasing functions as a function of the Euclidean distance r, having radial symmetry, are functions with compact support, i.e.,

$$\phi\left(r\right)=\begin{cases}>0, & \text{if } r<1\\ =0, & \text{if } r\geq 1\end{cases}; \tag{22}$$

and have normality properties,

$$\int_{\Omega}\phi(r)\ \mathrm{d}\Omega=1. \tag{23}$$

The Wendland compact RBF for three-dimensional cases are

$$\phi^0(r)=(1-r)^2 \qquad C^0 \tag{24a}$$
$$\phi^2(r)=(1-r)^4(4r+1) \qquad C^2 \tag{24b}$$
$$\phi^4(r)=(1-r)^6\left(\frac{35}{3}r^2+\frac{18}{3}r+1\right) \qquad C^4 \tag{24c}$$
$$\phi^6(r)=(1-r)^8\left(32r^3+25r^2+8r+1\right) \qquad C^6 \tag{24d}$$

C^n is the smoothness parameter of the curve. Graphs of the Wendland RBFs are shown in Fig. 1,

The normalized distances are calculated by taking the Euclidean distance, d, between the point of the structural mesh and the point in the fluid mesh interface, for each N closest point of the structural mesh interface to the mapped fluid point,

$$d_N=\sqrt{\left(x_{fN}-x_{sN}\right)^2+\left(y_{fN}-y_{sN}\right)^2+\left(z_{fN}-z_{sN}\right)^2}, \tag{25}$$

and normalizing it with the parameter r_{max} (Masarati *et al.*, 2014),

$$r_{max}=\begin{cases}\frac{d_N-d_{N-1}}{2}, & \text{if } d_N-d_{N-1}>100 \text{ eps}\\ 1.1d_N, & \text{else}\end{cases}, \tag{26}$$

where eps is the relative spacing between any two adjacent numbers in the machine's floating point system. So the RBF weight used in the least-square problem is

$$r_N=d_N/r_{max}. \tag{27}$$

3. METHODOLOGY

This section demonstrates the implementation of the mapping described in the previous section for constructing the interpolation matrix H. Additionally, it presents the implementation for calculating force and mesh displacements. A test case is also included at the end of the section. The case involves a strip of elastic structure with simple geometry and a small number of elements to be possible to show the details of the mapping without losing the understanding due to a large amount of data and points.

3.1 Mapping Implementation

For the implementation to build the matrix H, first, it is necessary to have five inputs: the mesh coordinates of the fluid interface; the mesh coordinates of the structure; definition of the number of points that will be used in the assembly of the matrix, which in this case would be the N points of the fluid interface mesh closest to the structure mesh point; the order of the polynomial basis functions, which can be linear or quadratic; and the degree of smoothness of the RBF.

The first step involves calculating the distances between the fluid interface mesh coordinates and the structure mesh coordinates. From this calculation, N structural interface mesh points closest to each fluid mesh point are selected. Due to the potentially large number of points between the meshes, this process can be computationally expensive. However, there are algorithms, such as the kd-tree algorithm mentioned in Wendland (2004), that efficiently handle this task with reduced computational requirements. The kd-tree algorithm organizes the data in a tree structure, subdividing the data points into smaller, manageable portions, thereby minimizing the number of data points that need to be searched. Consequently, after processing the data using this algorithm, a vector of size N will be obtained for each node of the fluid's mesh, providing the coordinates and distances of the N closest points.

Next, a loop is required for each node of the fluid mesh. Within this loop, the polynomial basis function is calculated for the N points from the structure's interface mesh that are in close proximity to the current node. The choice of Eq. (13) or (14) depends on the order of the selected polynomial. This calculation contributes to the construction of the matrix shown in Eq. (17), which has dimensions $N \times m$. Here, m represents the number of terms in the polynomial basis function.

Continuing within the loop, the RBF is computed for the N points using Eq. (24a) to (24d), depending on the desired level of smoothness. Subsequently, the diagonal matrix illustrated in Eq. (19) is assembled with dimensions $N \times N$. This matrix incorporates the RBF calculations.

Finally inside the loop, Eq. (21) is employed to determine the vector $a\left(\mathbf{x_f}\right)$, which has length N. This calculation involves utilizing the resultant matrices from A and b. To conserve memory, the vector f is employed to store only the first line of $a\left(\mathbf{x_f}\right)$ that is a $m \times N$ matrix. Consequently, only the first N points of $a\left(\mathbf{x_f}\right)$ are saved, storing the index of the fluid interface mesh analyzed point and the N most significant index points of the structure interface mesh, to construct the sparse interpolation matrix H.

To address the rotation mapping problem, offset beams are introduced alongside the main beam. These additional beams are positioned at a certain offset distance, which can be either constant or variable, from the structural beam. The purpose of these offset beams is to aid in the construction of the interpolation matrix H.

During the construction of the interpolation matrix, when selecting the N points from the structural mesh that are closest to the fluid interface mesh, the points from these offset beams are also taken into account. This inclusion allows for the calculation of rotations using the theory of rigid body motion and the rotation matrices associated with the nodes of the main beam.

The algorithm used to calculate the movement of the mesh based on the movement of the beam is below.

```
void Dynamic::DisplacementMapping()
{
    Matrix R(3,3);
    Matrix m_x(mapping->nCols_FEM,1);
    Matrix m_xp(mapping->nCols_FEM,1);
    int c = 0;
    for (int i = 0; i < number_nodes; i++){
        R = nodes[i]->Q; //Rotation Matrix
        for (int j = 0; j < mapping->n_Offsets; j++){
            Matrix f(3,1);
            Matrix offset(3,1);
            offset(0,0) = offsetNodes[i].nOffset[j].offset[0]; //Offset initial relative X position
            offset(1,0) = offsetNodes[i].nOffset[j].offset[1]; //Offset initial relative Y position
            offset(2,0) = offsetNodes[i].nOffset[j].offset[2]; //Offset initial relative Z position

            f = R * offset; //Rotation parameter

            Matrix XCurr(3,1);
            XCurr(0,0) = nodes[i]->coordinates[0]; //Node current X position
            XCurr(1,0) = nodes[i]->coordinates[1]; //Node current Y position
            XCurr(2,0) = nodes[i]->coordinates[2]; //Node current Z position

            Matrix x(3,1);
            x = XCurr + f; //Main beam node and offset position

            Matrix WCurr(3,1);
            WCurr(0,0) = nodes[i]->vel[3]; //Node X angular velocity
            WCurr(1,0) = nodes[i]->vel[4]; //Node Y angular velocity
            WCurr(2,0) = nodes[i]->vel[5]; //Node Z angular velocity

            Matrix Wcross(3,1);
            Wcross = cross(WCurr,f);

            Matrix v(3,1);
            v = XCurr + Wcross; //Main beam node and offset velocity

```

```
                m_x(c,0) = x(0,0);
                m_x(c+1,0) = x(1,0);
                m_x(c+2,0) = x(2,0);

                m_xp(c,0) = v(0,0);
                m_xp(c+1,0) = v(1,0);
                m_xp(c+2,0) = v(2,0);

                c = c + 3;
           }
      }
      m_q = sparseMatrixH * m_x; //Fluid interface mesh points position
      m_qp = sparseMatrixH * m_xp; //Fluid interface mesh points velocity
}
```

The program takes the rotation matrix, position, and angular velocity of the beam nodes as inputs. These inputs enable the calculation of the rotation parameter f, as well as the position, x, and updated velocity, v, of both the main beam nodes and the offset beams. Subsequently, the nodes are organized into vectors m_p and m_{xp}, allowing for multiplication with the interpolation matrix H, represented by the variable **sparseMatrixH**. This process yields the variables m_q and m_{qp}, which represent the vectors corresponding to the points belonging to the fluid interface mesh.

The algorithm for transferring loads to the beam is presented next.

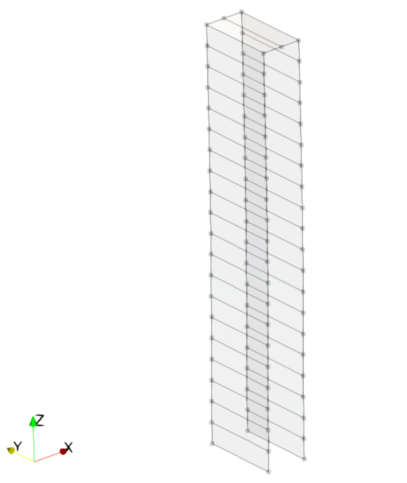
```
void Dynamic::LoadsMapping()
{
     Matrix m_f(mapping->nCols_FEM,1);
     m_f = sparseMatrixH_t * m_p; //Calculating the main and offset bean forces

    //Resenting forces and moments
     for (auto& node : Nodes) {
          for (int i = 0; i < 3; i++) {
               node.F[i] = 0.0;
               node.M[i] = 0.0;
          }
     }
     int c=0;
     int i=0;
     for (auto& node : Nodes) {
          Matrix R(3,3);
          R = *nodes[i]->Q; //Rotation Matrix
          for (int j = 0; j < mapping->n_Offsets; j++){
               for (int k = 0; k < 3; k++){
                    //Summing the node forces
                    offsetNodes[i].nOffset[j].Force[k] = m_f(c,0);
                    Nodes[i].F[k] += offsetNodes[i].nOffset[j].Force[k];
                    c = c + 1;
               }
               Matrix R(3,3);
               R = *nodes[i]->Q; //Rotation Matrix
               Matrix offset(3,1);
               offset(0,0) = offsetNodes[i].nOffset[j].offset[0]; //Offset initial relative X position
               offset(1,0) = offsetNodes[i].nOffset[j].offset[1]; //Offset initial relative Y position
               offset(2,0) = offsetNodes[i].nOffset[j].offset[2]; //Offset initial relative Z position
               Matrix f(3,1);
               f = R * offset; //Rotation parameter
               Matrix offset_F(3,1);
               offset_F(0,0) = offsetNodes[i].nOffset[j].Force[0];
               offset_F(1,0) = offsetNodes[i].nOffset[j].Force[1];
               offset_F(2,0) = offsetNodes[i].nOffset[j].Force[2];
               Matrix Cross_Nodes_F(3,1);
               Cross_Nodes_F = cross(offset_F,f); //Calculating moments
               for (int k = 0; k < 3; k++){
                    Nodes[i].M[k] += Cross_Nodes_F(k,0); //Summing the node moments
               }
          }
          i=i+1;
     }
}
```

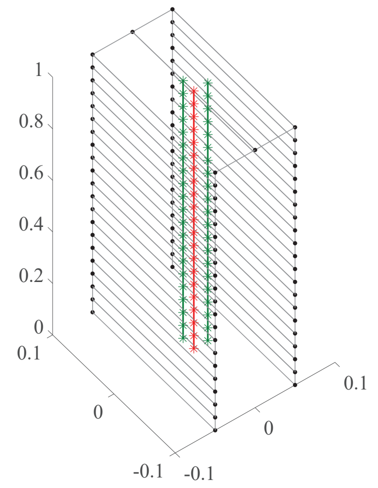
Initially, the forces at the points of the fluid interface mesh are multiplied by the inverse of the matrix H. This multiplication yields the forces acting on the main beam and offsets points. Subsequently, the contributions of the forces in the offset beams are summed together with those of the main beam, resulting in a consolidated force vector for each node. The same procedure is applied to calculate the moments, now taking into account the offset arms. This distribution of forces and moments is then transferred to each beam node to be used in subsequent solver calculations.

3.2 Example case

The example case involves an elastic strip with a simple geometry and a limited number of points in the fluid interface mesh and the structural beam. Figure 2 illustrates this case, with (a) displaying the interface surface of the fluid and (b) demonstrating the arrangement of the mapping objects, where the black dots represent the nodes and the gray lines depict the mesh cells of the fluid interface. A red line with asterisk markers represents the main beam, while green lines and green asterisk markers represent the offset beams and their corresponding nodes.



(a) Fluid mesh interface. (in scale)



(b) Structural and fluid mesh interface schematization. (no scale)

Figure 2: Elastic strip example case. In (a) is showed the representative fluid interface mesh with the dimensions in scale and in (b) is showed the schematization of the fluid interface mesh and the beam structural mesh, showing in red the main beam and in green the offset beams.

The main beam consists of 10 elements, each having 3 nodes, resulting in a total of 21 points. The first node, located at the base, has a fixed contour condition, restricting the 6 degrees of freedom. The offset beams follow the same distribution as the main beam, mirroring its movements. When combined, the main beam and the offset beams yield a total of 63 points. Each point possesses X, Y, and Z components, resulting in a total of 189 degrees of freedom, which corresponds to the number of columns in the interpolation matrix H.

Regarding the fluid interface mesh, it consists of 86 points. Taking into account the coordinate of each point, a total of 258 degrees of freedom are considered. This number represents the variable m mentioned in the previously shown equations and corresponds to the number of rows in the interpolation matrix H. For the mapping process, $N = 12$ closest points were used to assemble the weighted least-squares. Quadratic functions were employed as the basis polynomial functions, and the chosen degree of smoothness for the RBFs was C^4.

The elastic strip material has a modulus of elasticity of 3×10^6 Pa, a Poisson's ratio of 0.3, and a density of 3000 kg/m³. Rayleigh damping coefficients (Zienkiewicz *et al.*, 2013) are set to $\alpha = 0$ and $\beta = 0.001$. The strip has a height of 1 m and a rectangular cross-section measuring 0.1×0.2 m. The simulation was dynamic, spanning a total time of 3.5 s, with a time-step of 0.1 s, and the solver utilizes Newmark coefficients, with $\beta = 0.3$ and $\gamma = 0.6$. Time-constant load distribution is applied to the nodes of the elastic strip's fluid surface mesh at the beginning of the simulation. This distribution is then transferred to the beam nodes using the H interpolation matrix and integrated into them. Similarly, the displacement of the beam is transferred to the fluid surface mesh through the matrix, causing the nodes of the fluid mesh to undergo displacement.

4. RESULTS

In this section, we present the results obtained from the example case of the elastic strip. Figure 3 illustrates the format of the H interpolation matrix, where green represents non-zero values, while white represents zero values. It can be observed that the matrix has a significant number of zero values. Out of the $48,762$ terms in the matrix, $45,666$ (93.65%) are zero, while only $3,096$ (6.35%) have non-zero values. This sparsity of the matrix provides a computational advantage where the non-zero values correspond to the smallest distances between the points of the fluid interface mesh and the beam nodes, being the sparse matrix format determined by the distribution of the fluid interface mesh points.

At time zero, Fig. 4 depicts the initial displacement of the mesh and the distribution of forces applied to the fluid interface mesh, which remain constant over time. It shows the direction of the forces, and their magnitude is indicated by color. In 3.5 s, the mesh has undergone displacement due to the transfer of forces to the beam.

Figures 5(a) and 5(b) displays the distribution of forces and moments, respectively, that have been integrated and distributed among the beam nodes. The details on the beam, including their magnitude and direction, remain constant throughout the simulation. However, the moments vary over time as they are influenced by the rotation matrix of the beam, which changes due to the displacement of the beam, as shown in lines 34 and 41 of the algorithm `LoadsMapping`. Initially, the beam experiences moments close to zero, with values of approximately $-0.4\,\mathrm{N}\cdot\mathrm{m}$ near the root and $0.4\,\mathrm{N}\cdot\mathrm{m}$ near the tip. At $3.5\,\mathrm{s}$, the moments in the y direction at the nodes in the middle of the beam change direction and reach values close to $0.1\,\mathrm{N}\cdot\mathrm{m}$.

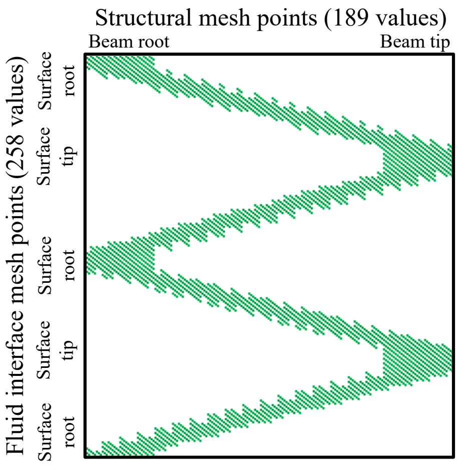


Figure 3: Configuration of the interpolation matrix H.

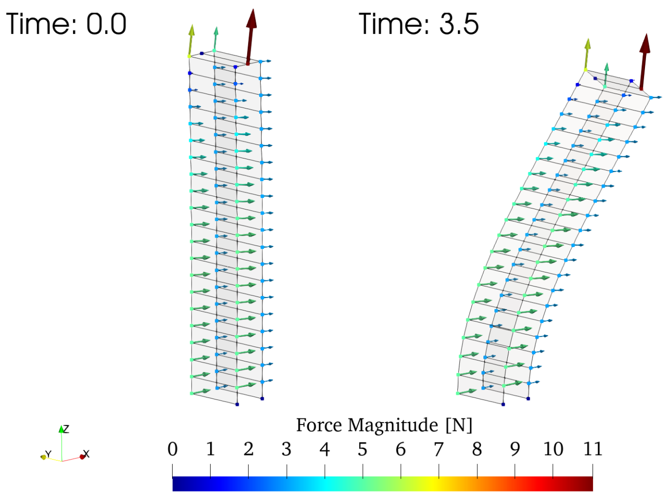
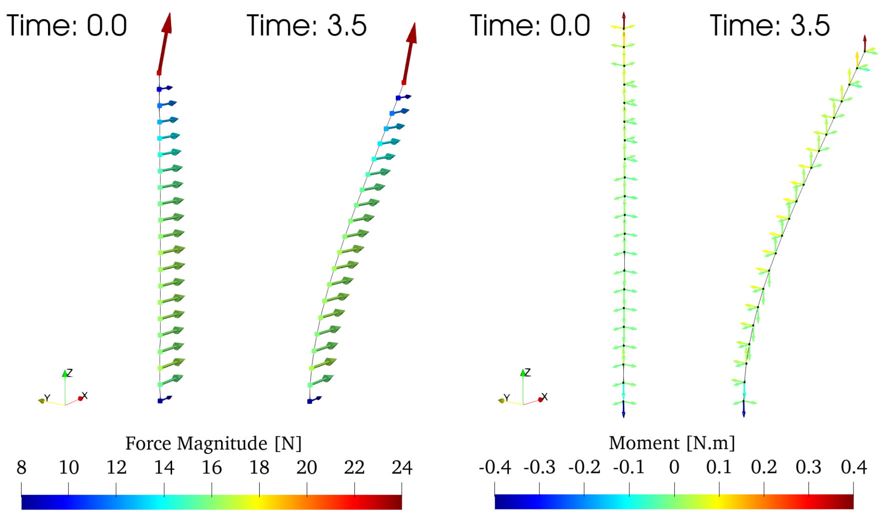


Figure 4: Applied forces in the fluid interface mesh.



(a) Applied forces in the structural mesh.

(b) Applied moments in the structural mesh.

Figure 5: Applied forces and moments in the structural mesh.

5. CONCLUSION

Calculating the interpolation matrix H for meshes with different topologies was possible. This enabled the transfer of forces from the fluid interface mesh to the beam model, accurately calculating the forces and moments. Additionally, the developed algorithm facilitated the transfer of movement and rotation of the mesh nodes from the structure to the fluid interface mesh. As described in the literature, the utilization of offset beams proved instrumental in resolving the challenge of transferring rotation from the beam to the fluid interface mesh. With this solver component in place, the groundwork has been laid for developing aeroelastic coupling between the Giraffe and OpenFOAM.

6. ACKNOWLEDGEMENTS

This work was developed as part of the R&D project conducted by Petrobras, and the University of São Paulo entitled "Research and Development on Deep Water Floating Offshore Wind Turbines" (agreement Petrobras #5900.0112605.19.9). The authors wish to thank Petrobras for funding this project and the Brazilian National Petroleum Agency (Agência Nacional do Petróleo, ANP) for providing the regulatory framework under which this funding takes place. B. S. Carmo thanks the Brazilian National Council for Scientific and Technological Development (CNPq) for the financial support in the form of a productivity grant, number 314221/2021-2.

7. REFERENCES

Chopra, A., 2012. *Dynamics of Structures: Theory and Applications to Earthquake Engineering*. Civil Engineering and Engineering Mechanics Series. Prentice Hall. ISBN 9780132858038. URL `https://books.google.com.br/books?id=3cctkgEACAAJ`.

Duarte, F., Gormaz, R. and Natesan, S., 2004. "Arbitrary lagrangian–eulerian method for navier–stokes equations with moving boundaries". *Computer Methods in Applied Mechanics and Engineering*, Vol. 193, No. 45, pp. 4819–4836. ISSN 0045-7825. doi:https://doi.org/10.1016/j.cma.2004.05.003. URL `https://www.sciencedirect.com/science/article/pii/S0045782504002476`.

Gatzhammer, B., 2014. *Efficient and Flexible Partitioned Simulation of Fluid-Structure Interactions*. Ph.D. thesis, Technischen Universitat München.

Gay Neto, A., 2018. "Modelagem computacional do contato pontual entre corpos: uma visão integrada". Thesis (Associate Professor) - University of São Paulo.

Jasak, H., 1996. *Error analysis and estimation for the finite volume method with applications to fluid flows*. Ph.D. thesis, Imperial College London (University of London).

Masarati, P., Morandini, M. and Mantegazza, P., 2014. "An Efficient Formulation for General-Purpose Multibody/Multiphysics Analysis". *Journal of Computational and Nonlinear Dynamics*, Vol. 9, No. 4, p. 041001. ISSN 1555-1415. doi:10.1115/1.4025628. URL `https://doi.org/10.1115/1.4025628`.

Quaranta, G., Masarati, P. and Mantegazza, P., 2005. "A conservative mesh-free approach for fluid structure problems". *Int. Conf. on Computational Methods for Coupled Problems in Science and Engineering, Barcelona*.

Rendall, T. and Allen, C., 2008. "Unified fluid-structure interpolation and mesh motion using radial basis functions". *International Journal for Numerical Methods in Engineering*, Vol. 74, No. 10, pp. 1519–1559. ISSN 0029-5981. doi:10.1002/nme.2219. Publisher: Wiley Interscience.

Wendland, H., 1995. "Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree". *Advances in Computational Mathematics*, Vol. 4, pp. 389–396.

Wendland, H., 2004. *Scattered Data Approximation*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press. doi:10.1017/CBO9780511617539.

Yuan, W., Sandhu, R. and Poirel, D., 2021. "Fully coupled aeroelastic analyses of wing flutter towards application to complex aircraft configurations". *Journal of Aerospace Engineering*. doi:10.1061/(asce)as.1943-5525.0001232.

Zienkiewicz, O., Taylor, R. and Zhu, J., 2013. "Chapter 12 - the time dimension: Semi-discretization of field and dynamic problems". In O. Zienkiewicz, R. Taylor and J. Zhu, eds., *The Finite Element Method: its Basis and Fundamentals (Seventh Edition)*, Butterworth-Heinemann, Oxford, pp. 379–405. Seventh edition edition. ISBN 978-1-85617-633-0. doi:https://doi.org/10.1016/B978-1-85617-633-0.00012-5. URL `https://www.sciencedirect.com/science/article/pii/B9781856176330000125`.

8. RESPONSIBILITY NOTICE