POLITECNICO DI MILANO

Facoltà di Ingegneria Industriale e dell'Informazione

Dipartimento di Scienze e Tecnologie Aerospaziali

Corso di Laurea Magistrale in
Ingegneria Aeronautica

A reduced-order Inverse Distance Weighting technique for the efficient mesh-motion of deformable interfaces and moving shapes in computational problems

Advisors:

Prof. Simona PEROTTO

Prof. Ing. Gianluigi ROZZA

Co-Advisor:

Dr. Francesco BALLARIN

Author:

Alessandro D'AMARIO

Matr. 800490

Anno Accademico 2014-2015

Contents

List of Figures

List of Tables

Abstract

Nowadays, despite the constant technological growth of the last twenty years, Computational Fluid Dynamics (CFD) problems represent a hard challenge for scientific computing because of their large demand on computational resources. In fact, in the context of aerodynamics optimization and design, CFD applications require to simulate many different possible realizations of a system which can become prohibitive in terms of computational and memory effort. These considerations have produced an intensive development of reduced order methods to provide high-fidelity simulations via efficient and low-dimensional models.

In this thesis we focus on an efficient and flexible technique, namely the Inverse Distance Weighting (IDW). This strategy can be used to solve mesh motion problems in a Fluid-Structure Interaction (FSI) framework.

IDW is an interpolation strategy which computes the displacement of the grid nodes starting from the movement of some data points, called *control points*. The original formulation of IDW uses as control points the nodes of the grid belonging to the interface and requires assembling a matrix $[\mathcal{N}_s \times \mathcal{N}_c]$, where $\mathcal{N}_s$ is the number of values to be interpolated and $\mathcal{N}_c$ indicates the number of control points. Since the number of interface nodes for a typical mesh motion problem can be very large (of the order of hundreds of thousands), the system to solve assumes significantly high dimensions.

In order to reduce the computational and the memory load due to these huge dimensions, we implement an ad hoc algorithm which performs a *geometrical* and a *model order* reduction of the system: the former reduction is based on an iterative procedure which selects some relevant control points among the initial ones, while the latter reduction is inspired by the Proper Orthogonal Decomposition (POD) strategy to exploit the advantages of an *Offline-Online splitting* between ROM construction and evaluation of the solution.

The implemented algorithm is validated on some examples, from benchmark cases to more complex three dimensional configurations such as a wing or a hull.

All the reduction strategies developed are implemented using an efficient *C++* object oriented code and an open-source Finite Element library (*libMesh*), while the processing of the grids is performed by the open-source visualization software *Paraview*.

Acknowledgements

Prima di tutto vorrei ringraziare i miei relatori, la Professoressa Simona Perotto e il Professor Gianluigi Rozza, per avermi aiutato con ogni mezzo a portare a termine questo percorso e per avermi fornito una tale occasione di crescita dal punto di vista personale e lavorativo. Aggiungo un ulteriore ringraziamento al Dott. Francesco Ballarin, senza il cui aiuto e sostegno, il completamento di questo lavoro non sarebbe stato possibile.

Un ringraziamento particolare va a tutto il gruppo SISSA Mathlab di Trieste che mi ha permesso di accrescere le conoscenze accademiche in diversi campi e che mi ha riservato un'invidiabile accoglienza e disponibilità.

Non riuscirò mai a ringraziare abbastanza la mia famiglia che mi ha supportato e aiutato in ogni scelta durante tutto il percorso di una vita che, oggi, mi ha portato fin qui. Spero di renderli orgogliosi nel condividere con me il raggiungimento di questo traguardo.

In particolare, vorrei riservare un ringraziamento speciale a mia sorella Ada Maria e a suo marito Matteo che sono riusciti a farmi sentire a casa anche quando ero lontano, donandomi spesso un sorriso nei momenti bui.

Infiniti ringraziamenti vanno a mia madre per la sensibilità, la pazienza e la disponibilità che la caratterizzano da sempre.

Non posso esimermi dal rivolgere un caloroso grazie agli zii e ai cugini tutti, con un riferimento particolare a mia zia Rosella, madre adorabile, sorella modello e amica fedele e saggia.

Un amorevole ringraziamento alla vera fonte della mia forza, modelli ineguagliabili di vita, i nonni.

Durante tutto il percorso della laurea magistrale il ricordo di Voi non ha mai lasciato il mio cuore e la mia mente. Per gli abbracci, per le mani tese, per le pacche sulla spalla anche quando non c'eravate più. Oggi un pensiero particolare è rivolto a voi. Grazie Zio, grazie Nonno.

Un ringraziamento pieno di amore va al mio Mister, compagno di dure battaglie, fonte inesauribile di sorrisi e saggezza. Te lo devo. Grazie papà.

Ringrazio in modo particolare Emanuela, una compagna di vita che ha saputo sempre spendere per me le parole giuste, permettendomi di capire fino in fondo il significato di tante situazioni delicate.

Veniamo a voi, cari amici. Siete tanti e rischierò di dimenticare qualcuno, non me ne vogliate, siete indiscutibilmente tutti presenti in ogni caso.

A Luca e Roberta, compagni di avventura a Trieste e pazienti ascoltatori delle mie storie strampalate, un affettuoso ringraziamento.

Mille grazie agli amici Riccardo, Roberta, Michela, Lidia, Maria Antonietta, Miky, Antonella, Guglielmo, Tiziana e Marco e a tutti gli altri che con me hanno condiviso le esperienze, belle e brutte, vissute negli ultimi anni in ogni città d'Italia.

Voglio ringraziare anche i nuovi colleghi di lavoro, Luca, Paolo, Matteo, Gianluca, Alessandro e Thomas, che negli ultimi mesi mi hanno permesso di sostituire carichi e tensioni con sorrisi e soddisfazioni.

Un ringraziamento speciale va a Luigi. Grazie per gli anni vissuti fianco a fianco, per l'affetto dimostrato in ogni occasione, per tutto quello che mi hai insegnato e per tutto quello che abbiamo condiviso, quotidianamente e straordinariamente. Se chi trova un amico trova un tesoro, chi trova un fratello cosa trova?

Dulcis in fundo, come la dolcezza che mi hai riservato ogni giorno in ogni gesto, un ringraziamento particolare va a te, Valeria. Grazie per la pazienza, la comprensione, i suggerimenti, il sostegno e gli abbracci che non hai mai smesso di donarmi anche quando non me ne accorgevo. Grazie.

Sintesi

In questo lavoro di tesi abbiamo analizzato e sviluppato una tecnica di interpolazione, nota in letteratura come **Inverse Distance Weighting** (**IDW**), utilizzata tipicamente nel contesto di interazione multifisica fluido-struttura come tecnica di parametrizzazione di forma, per risolvere problemi inerenti la **deformazione della griglia di calcolo** (mesh).

A partire da una deformazione della struttura, questa tecnica permette di studiare e produrre lo spostamento della mesh, tipicamente quella relativa al fluido, attraverso una interpolazione basata sulla media pesata delle distanze tra i nodi interni alla griglia e alcuni punti di controllo preventivamente assegnati.

L'obiettivo del lavoro è stato quello di ridurre il carico computazionale e di memoria legato all'implementazione della tecnica mediante una **riduzione di tipo geometrico sui parametri** in gioco, nel nostro caso i punti di controllo, in vista di una riduzione computazionale attraverso l'utilizzo di un **metodo di ordine ridotto**.

L'idea fondamentale nella realizzazione dell'intero processo di riduzione risiede, prima di tutto, nella riduzione dei **punti di controllo** attraverso una logica standardizzata che permetta di ottenere una distribuzione efficiente dei punti noti che, tipicamente, giacciono sull'**interfaccia** tra i due sottodomini.

L'algoritmo di riduzione elaborato può essere ulteriormente inserito all'interno di un processo che, servendosi della **Proper Orthogonal Decomposition (POD)**, ha portato alla costruzione di un metodo di ordine ridotto, permettendo, così, di risolvere rapidamente il problema e di abbatterne i costi computazionali. In questo modo, la soluzione può essere ottenuta attraverso la combinazione di un insieme di soluzioni calcolate per valori dei parametri assegnati. A valle dell'esecuzione della procedura implementata, il numero necessario di queste soluzioni, indicate come funzioni di base, risulta decisamente esiguo poichè l'energia del sistema può essere catturata in pochi *modi* che ne descrivono in modo opportuno il comportamento.

Il vantaggio fondamentale fornito dall'utilizzo di un tale metodo risiede nella possibilità di suddividere la risoluzione del problema in due fasi di calcolo ben distinte:

- *Fase Offline*: la fase più onerosa in cui lo sforzo computazionale è volto alla costruzione dello spazio ridotto, ottenuto, a partire dai valori dei parametri assegnati, mediante il calcolo delle funzioni di base.

◦ *Fase Online*: la fase meno costosa nella quale, per ogni nuovo valore dei parametri, viene calcolata molto rapidamente la soluzione attraverso una proiezione sullo spazio ridotto precedentemente costruito.

Questa metodologia risulta essere di notevole utilità per l'ottenimento della soluzione di processi in tempo reale, per esempio nel caso di problemi di ottimizzazione di forma, per i quali è necessario ottenere la soluzione del sistema al variare, anche in modo significativo, dei valori dei parametri.

In primo luogo, la tecnica IDW si serve dell'assemblaggio di una matrice all'interno della quale vengono memorizzati, pesati mediante un parametro p, i reciproci delle distanze tra i nodi interni e i punti di controllo. La tecnica permette di calcolare il valore di una qualsiasi grandezza fisica o geometrica, relativa ai problemi di interazione fluido-struttura, a partire da valori assegnati della stessa grandezza nei punti di controllo. Nella forma originale della tecnica questi ultimi risultano essere tutti i nodi appartenenti alla mesh che giacciono sull'interfaccia.

Nel nostro caso, la strategia racchiusa nella IDW viene utilizzata per ricavare la deformazione della griglia computazionale relativa al fluido, noti i valori di spostamento dei punti di controllo. Il movimento dell'interfaccia, infatti, può essere ottenuto a partire dal risultato del solutore strutturale, il quale fornisce la nuova posizione dei punti di controllo e permette di ricavare l'ampiezza dello spostamento di tutti i restanti nodi del reticolo.

È ben noto che la difficoltà principale nell'utilizzo di questa tecnica è data dall'elevato numero di nodi che costituiscono tipicamente una griglia di calcolo di interesse aeronautico e/o navale, per esempio. Non è irrealistico, infatti, considerare reticoli dell'ordine di centinaia di migliaia di nodi per studi di tipo fluidodinamico.

Parte di questi nodi, come anticipato, assume il ruolo di punti di controllo contribuendo all'assemblaggio di una matrice le cui dimensioni aumentano all'aumentare del numero di punti di controllo. Il risultato della tecnica, inoltre, prevede l'utilizzo di questa matrice in prodotti matrice-vettore che producono un carico computazionale proibitivo per simulazioni in tempo reale o di ottimizzazione di forma, nonostante il costante aumento della disponibilità di risorse computazionali a cui stiamo assistendo negli ultimi anni.

Per le ragioni di cui sopra, ai fini di migliorare le prestazioni computazionali del metodo, sono state implementate diverse procedure di riduzione che hanno prodotto risultati soddisfacenti in termini di tempi di calcolo contenendo l'errore prodotto rispetto alla formulazione standard della stessa tecnica IDW.

Le fasi di sviluppo hanno richiesto l'analisi di due aspetti fondamentali: la geometria del reticolo e la modellistica del sistema. Per quel che concerne le caratteristiche geometriche della mesh sono state proposte due procedure differenti:

- riduzione dei punti di controllo;

- applicazione in *cascata dimensionale*.

L'algoritmo di riduzione del numero dei punti di controllo è stato implementato secondo una logica di eliminazione che prevede la definizione di aree circolari, lasciando la possibilità all'utente di selezionarne il raggio R. A partire da uno qualsiasi dei punti di controllo e tracciato un cerchio di raggio R centrato in esso, il processo elide iterativamente tutti i punti che giacciono al suo interno, per poi selezionare un punto esterno e ripetere la procedura di eliminazione.

L'esigenza di conoscere lo spostamento dei nodi di bordo che non appartengono all'interfaccia e la necessità di permettere all'utente di spostare un qualsiasi nodo della mesh, hanno prodotto lo sviluppo di una procedura di esecuzione in *cascata dimensionale*.

Con il termine *cascata dimensionale* si intende un'applicazione che prevede, in primo luogo, lo studio dello spostamento dei nodi di vertice e di spigolo della griglia di calcolo, per poi calcolare la nuova posizione dei nodi appartenenti alle facce del bordo e quindi, una volta ottenuto l'intero bordo deformato, utilizzare i valori di spostamento dei punti di controllo per il calcolo della soluzione nei nodi interni.

Sebbene i risultati, frutto dello sviluppo di tipo geometrico, siano già alquanto promettenti (si arriva a considerare circa il 40% dei punti di controllo rispetto alla formulazione IDW originale), si è deciso di potenziare ulteriormente le prestazioni della tecnica annettendo una riduzione di modello. Sfruttando interamente la potenzialità della suddivisione tra fase offline e fase online, la modellistica ha ridotto ancor più i tempi di calcolo, con un errore sullo spostamento dei nodi interni (relativo alla formulazione standard) che non ha mai superato pochi punti percentuali. Il processo di riduzione è passato attraverso la costruzione di un metodo ispirato alla Proper Orthogonal Decomposition. In modo simile alla POD, infatti, nella fase offline è stata assemblata una matrice di funzioni di forma, cioè di soluzioni per differenti spostamenti dell'interfaccia, che ha permesso di ricavare uno spazio ridotto di soluzione. È mediante una proiezione su questo spazio che, nella fase online, è stato possibile ottenere la soluzione per valori arbitrari dei parametri.

A valle di queste implementazioni sono stati presentati e confrontati i risultati ottenuti per differenti applicazioni, con particolare contestualizzazione nell'ambito aeronautico e navale.

All'aumentare della complessità delle griglie di riferimento, i vantaggi della tecnica hanno prodotto effetti molto significativi sulla riduzione dei costi computazionali e della dimensione delle matrici assemblate, permettendo, così, di ampliare l'utilizzo della tecnica per problemi di ottimizzazione di forma e calcoli in tempo reale che, come avevamo anticipato, in precedenza risultavano davvero proibitivi.

Ad oggi la tecnica continua ad essere oggetto di sviluppo al fine di renderla sempre più performante nell'ambito della Fluidodinamica Computazionale (CFD) e della Aeroservoelasticità Computazionale (CA) anche, e non solo, mediante i metodi di ordine ridotto (vedi, ad esempio, POD, Proper Generalized Decomposition (PGD) e Reduced Basis Methods (RBM) per un approfondimento sui quali rimandiamo a [24]).

Chapter 1

Introduction

In this Master Thesis we want to present a work, realized in the framework of collaboration between mathLab team of SISSA (International School of Advanced Studies) in Trieste and MOX laboratory at Politecnico di Milano, about the development of an efficient and flexible technique used for mesh motion applications in Fluid-Structure Interaction (FSI) multi-physics problems.

The main idea is to exploit a well-known shape representation technique called Inverse Distance Weighting (IDW) [77], in order to find the displacement of each node starting from the position of some control points, and to reduce its computational load and memory effort in order to efficiently represent the deformation of the fluid grid following a structural sub-domain movement.

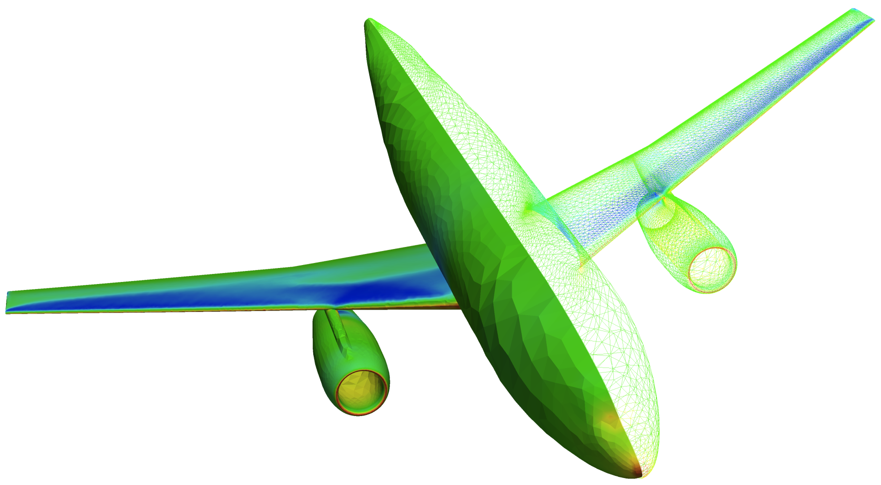


Figure 1.1: Example of an aeronautical mesh.

1.1 Motivations

In several engineering fields, the interest for FSI problems has known a significant growth in the last twenty years, particularly for biomedical, aerospace and maritime industry, but not limited to.

First of all, it is important to say that, tipically, we deal with a fluid-structure interaction problem when the effect of fluid dynamics on elastic bodies and vice-versa is of primary interest. The solution of this kind of problems is necessary to several engineering applications: aero-elasticity [75], turbomachinery design, modelling of the cardiovascular system [56], high performance boat dynamics [52, 51], etc. Fluid-Structure interaction phenomena require an unsteady computation of the dynamics of different physical domains (i.e. multiphysics) and their coupling with moving boundary conditions and meshes. This kind of simulations starts from a mesh generation to enter in a loop of computations divided in two parts: the *structural solver* and the *fluid solver*. The link between these two parts is the so-called **interface**, the only *object* that allows to transfer physical information (stresses, pressure, displacement).

Generally, any variation in one or both the two parts produces a displacement that has to be represented in the algorithm by an efficient implementation of the mesh motion computation. The algorithm for a generic FSI simulation process is represented in Figure 1.2.

We can formulate the problem as follows: denoting by $\hat{\Omega}^F \subset \mathbb{R}^3$ and $\hat{\Omega}^S \subset \mathbb{R}^3$ the initial domain configuration for the fluid and the structure, respectively, thanks to the solution of the structural sub-problem, we obtain the displacement field $\varsigma \in \mathcal{D}_S$ leading to a structural deformed domain Ω^S:

$$\varsigma : \hat{\Omega}^S \longrightarrow \Omega^S \tag{1.1}$$

with $\mathcal{D}_S := H^1(\hat{\Omega}^S)$ the space of kinematically admissible structural displacements. At this point, once we update the structural configuration, we also need to deform the fluid subdomain accordingly, before advancing the FSI simulation in time. Therefore, indicating by $\mathcal{D}_\mathcal{F} := H^1(\hat{\Omega}^F)$ the space of kinematically admissible fluid displacements, such a problem could be set as:

find the fluid displacement field $\zeta : \hat{\Omega}^F \to \Omega^F$, with $\zeta \in \mathcal{D}_\mathcal{F}$, that guarantees no compenetrations at the interfaces between fluid and structure domains, that is $\Gamma^F = \Gamma^S$, where Γ^F and Γ^S indicate the fluid and structure interfaces in the deformed configuration, respectively, and Ω^F is the new fluid configuration. (1.2)

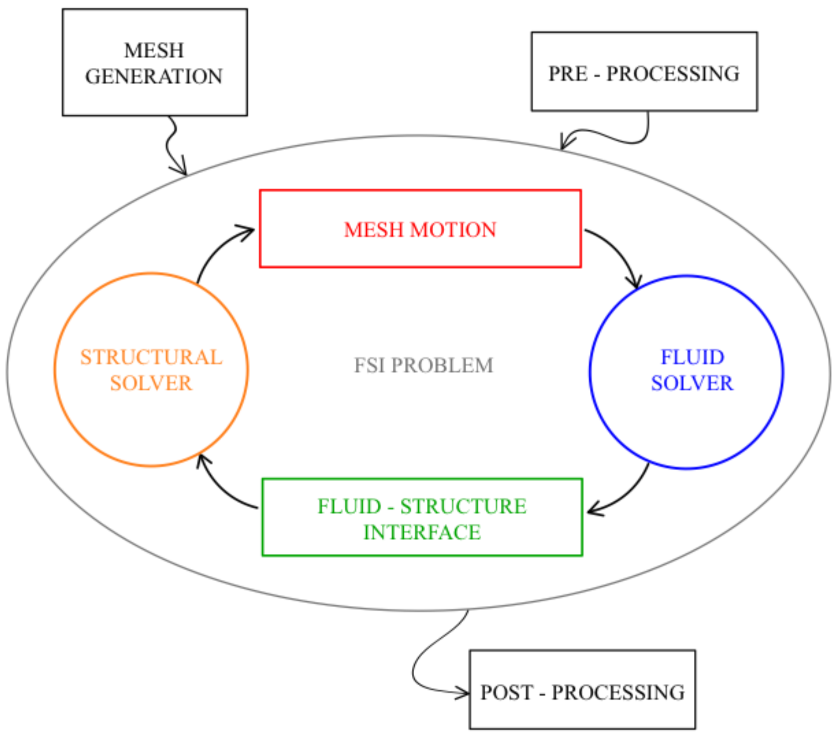


Figure 1.2: Scheme of a FSI simulation process.

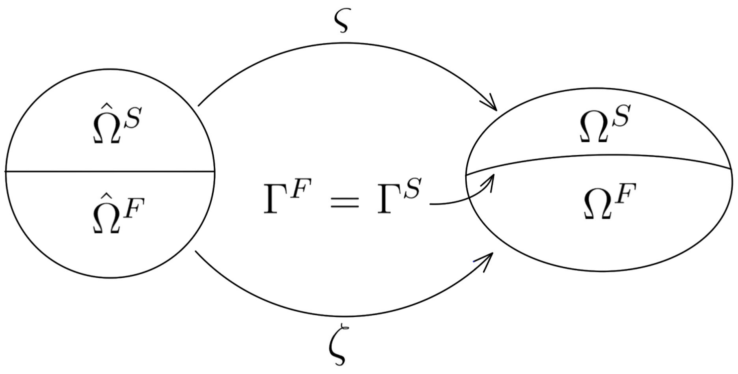


Figure 1.3: Schematic visualization of the condition to be satisfied at the fluid-structure interface.

In other words, extracted ς from the structural solver, the problem becomes: find ζ employing a **shape parametrization technique**.

Before introducing the shape parametrization techniques suited for these problems, for the sake of clarity, we summarize here some important theoretical and numerical notions inherent to discretizations and energy conservation for FSI problems.

To solve a FSI problem, we need to deal with different discretization schemes used for the two domains: fluid and structure.

The main idea relies on the **energy conservation principle** valid at the interface: the virtual work δW performed by fluid loads and by structural forces has to be preserved by the coupling scheme [12, 56]. Thus, it is possible to write:

$$\delta W = \delta \boldsymbol{\varsigma}^T \, \mathbf{f}_S = \delta \boldsymbol{\zeta}^T \, \mathbf{f}_F, \tag{1.3}$$

where $\delta \boldsymbol{\varsigma}$ and $\delta \boldsymbol{\zeta}$ indicate nodal values of virtual displacements for structure and fluid domains, while $\mathbf{f}_S$ and $\mathbf{f}_F$ represent the structural and fluid loads.

We can express a relation between the two displacements by introducing a transfer matrix $\mathbf{H}$ as follows:

$$\boldsymbol{\zeta} = \mathbf{H} \boldsymbol{\varsigma}. \tag{1.4}$$

If we assume that the virtual work is null for each arbitrary $\delta \boldsymbol{\varsigma}$, the relation written above becomes:

$$\delta \boldsymbol{\zeta} = \mathbf{H} \, \delta \boldsymbol{\varsigma}, \tag{1.5}$$

so:

$$\mathbf{f}_S = \mathbf{H}^T \, \mathbf{f}_F. \tag{1.6}$$

Thus, it is possible to obtain the so-called *conservative coupling approach*, relying on the energy conservation at the interface, when a transfer matrix $\mathbf{H}$ exists.

In order to find the coupling matrix $\mathbf{H}$, a very wide range of methods can be used, the simplest Nearest Neighbor Interpolation (NNI) strategy [79], Weighted Residual Methods (WRM) [22] and Radial Basis Functions (RBF) [12, 17], for example.

Among the strategies cited above, Radial Basis Functions are surely the most used for problems involving the data-transfer across non-matching discretization schemes.

RBF is a well-known and tested method that allows to approximate both multivariate functions and scattered data by a real-valued function whose values depend only on the distance from the origin. Thus, a radial basis function is a function ϕ that satisfies the following property:

$$\phi(\mathbf{x}) = \phi(\|\mathbf{x}\|), \tag{1.7}$$

or, alternatively, on the distance from another point $\mathbf{c}$, called center, such that:

$$\phi(\mathbf{x}, \mathbf{c}) = \phi(\|\mathbf{x} - \mathbf{c}\|), \tag{1.8}$$

where the norm considered is, usually, the Euclidean distance.

RBF strategy has several advantages: versatility due to the applicability in almost any dimension, little restrictions on the way the data are prescribed, high-accurancy of the results, wide range of applications in different fields like aero-elasticity [75], sail boat design [52, 51, 29], haemodynamics [56, 7], etc.

Another goal to achieve when we deal with FSI problems is to solve the governing equations for both fluid and structure *moving* domains under deformation.

The mesh motion part of FSI problems can be summarized in the following definition:

to update the fluid mesh configuration on the basis of the motion computed for the nodes lying on the interface Γ. (1.9)

Due to the multiphysics interaction between the fluid and the structure, in fact, during the FSI simulation the computational grids will move, resulting in geometrically deformed boundaries (see Figure 1.4). The displacement of the interface nodes is considered to be given as solution of the structural sub-problem.

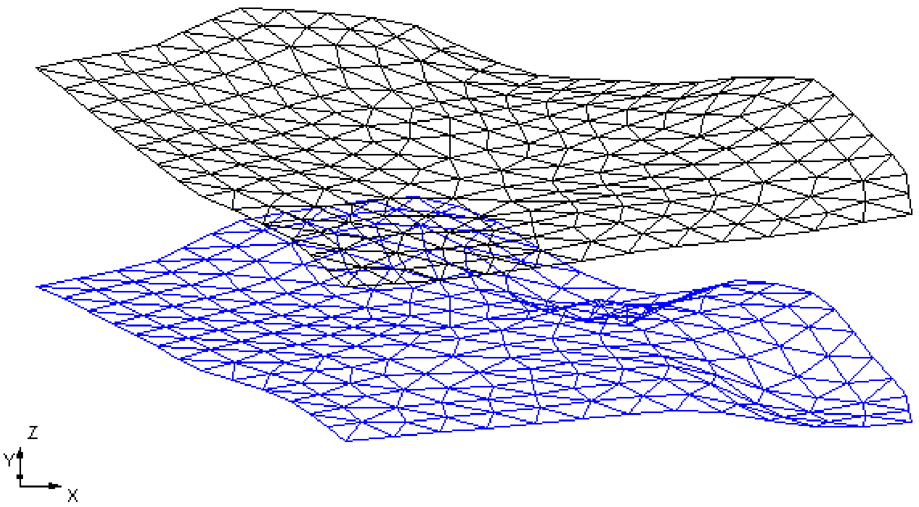


Figure 1.4: Example of mesh deformation: starting configuration (black) and final deformed mesh (blue)

In the framework of mesh motion strategies for FSI problem, it is necessary to describe a particular class of techniques that involve the solution of a partial differential equation given on the fluid-structure interface and defined on the fluid mesh region. Among these strategies are worth of note the most commonly adopted ones : Laplacian [36] and Solid Body Rotation Stress (SBRS) [30]. The former is given by the solution of the following Laplace equation:

$$\nabla \cdot (\gamma \nabla \mathbf{u}) = 0. \tag{1.10}$$

It allows to assign large displacement values close to the moving boundaries, while small ones are assigned at larger distances. The vector **u** represents the displacement of the grid nodes, while γ indicates the diffusivity coefficient, in order to control the magnitude of the nodal displacement for best performances. The diffusivity coefficient has to be inversely proportional to the Euclidean norm of the distance from the deformed boundary. So, a common choice is :

$$\gamma = \frac{1}{\|\mathbf{x}\|^{\alpha}}, \tag{1.11}$$

where α is usually chosen to be equal to two to guarantee a quadratic decrease.

SBRS, instead, uses the linear elasticity equation:

$$\nabla \cdot \boldsymbol{\sigma} = \mathbf{f}, \tag{1.12}$$

with $\boldsymbol{\sigma}$ denoting the stress tensor and **f** the body forces. Equation 1.12 can be rearranged using the elastic constitutive law and including the Lamé constants λ and μ defined as follows:

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)}, \qquad \mu = \frac{E}{2(1+\nu)}, \tag{1.13}$$

where E is the Young's modulus and ν is the Poisson's ratio, while the kinematic law between the strain and displacement is provided by:

$$\boldsymbol{\epsilon} = \frac{1}{2}(\nabla \mathbf{u} + \nabla \mathbf{u}^T + \nabla \mathbf{u}^T \cdot \nabla \mathbf{u}). \tag{1.14}$$

Particular attention has to be reserved to the term $(\nabla \mathbf{u}^T \cdot \nabla \mathbf{u})$ that allows to include rigid rotations of the mesh.

Finally, the solid body rotation stress equation becomes:

$$\nabla \cdot [\, \mu(\nabla \mathbf{u} + \nabla \mathbf{u}^T) - \lambda \nabla \cdot \mathbf{u}\,] = 0. \tag{1.15}$$

It is possible to show that the role played by the diffusivity coefficient in the Laplacian method is here played by the Lamé constants for which a tuning operation is necessary.

Also in this context, it is possible to include the RBF strategy. In fact, the RBF interpolation procedure is based on the adoption of a set of interpolation sites, or control points, whose displacement values are known, in order to get the ones related to the inner fluid mesh. A first choice of the interpolation sites can fall on all interface boundary nodes. Nevertheless, in order to reduce memory and computational costs, it is suitable to construct an automated and standardized selection process for the control point number and locations.

To achieve a reduction of the costs, first of all, we need to introduce some shape parametrization strategies used for FSI and mesh motion problems [52, 68, 31, 32, 8].

They rely on mapping a reference domain, $\Omega \subset \mathbb{R}^3$, through suitable parametric maps $T(\mathbf{x}, \boldsymbol{\mu})$, $\mathbf{x} \in \Omega$, based on a small set of control points, whose possible displacements $\boldsymbol{\mu} \in \mathbb{R}^3$ induce the shape deformations. In this way, thanks to control point displacements, it is possible to represent efficiently a family of admissible shape configurations. Thus, their goal is to reduce the complexity in the description of the geometry of solid objects, as well as their deformations, such that we can represent them through low dimensional spaces, instead of using geometrical properties themself, always compatibly with, and adaptable to, various numerical analysis issues in the FSI framework and the dynamic mesh handling.

Shape parametrization strategies like Free-Form Deformation, Radial Basis Functions and Inverse Distance Weighting, give the possibility to describe complex structural configurations, reducing significantly the complexity of the geometrical model. In this way, we may achieve a great reduction in the computational effort, within the updates of fluid or structural mesh configuration, since to impose shape deformations we have to move only few control points. Moreover, other ideas of cost reduction are applicable to shape parametrization techniques, for example the ones proposed by Forti and Romanelli [31, 68], where the global structural displacement has been sub-divided into two parts: a *small* deformative component and a *large* rigid one. Known, in fact, the possibility to correctly identify the entity of rigid translations and/or rotations as output of the structural solver, the shape parametrization technique can be reduced only to the part subject to deformation.

To apply **Free-Form Deformation** (FFD) [76, 42, 8, 7] to mesh motion problems for FSI, means not to operate directly on the geometrical properties. In fact, the evaluation of local and/or global shape deformations is obtained using a parametric map $T_{FFD}(\cdot, \boldsymbol{\mu})$ on a reference and parameter-independent domain configuration Ω, with a small set of control points playing the role of the parameters $\boldsymbol{\mu}$. In this way, the shape deformation is computable perturbing (displacing) the control points.

Radial Basis Functions (RBF) shape parametrization technique [28, 17] is still based on the use of a set of parameters, the so-called **control points**, as for FFD. Nevertheless, RBF is also interpolatory: the function values computed at the control points are exactly the ones of the function to be interpolated. This is due to the definition of a map, $\mathcal{M}(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^n$, which is a sum of basis functions ϕ, centered at each control point $\mathbf{X}_C$, that allows to recover the exact function values at least over the interpolation sites. The map introduced is defined as:

$$\mathcal{M}(\mathbf{X}) = p(\mathbf{X}) + \sum_{i=1}^{\mathcal{N}_c} \gamma_i \, \phi(\| \mathbf{X} - \mathbf{X}_{C_i} \|), \tag{1.16}$$

where $p(\mathbf{X})$ is a low-degree polynomial term, γ_i are the weights, corresponding to the a-priori selected $\mathcal{N}_c$ control points and associated with the basis functions, while $\phi(\| \mathbf{X} - \mathbf{X}_{C_i} \|)$ is a radial basis function based on the Euclidean distance between the control point position $\mathbf{X}_{C_i}$ and $\mathbf{X}$. This method allows the possibility of trasferring data across non-matching grids and facing the dynamic mesh handling.

The **Inverse Distance Weighting** (IDW) technique [77, 83, 84, 23] is an *average moving* interpolation strategy. Its main idea lies in the computation of all the interpolated values by averaging the weighted sum of the distances between data points and grid nodes. The influence of a data point on each interpolated value is inversely proportional to the distance of the former from the mesh nodes. In this way, moving away from the considered data points, they affect progressively less the computed values.

This point-to-point interpolation technique is characterized by a particular flexibility in handling *arbitrary mesh topologies*. Furthermore, it combines the ideas and the advantages proposed above in the Laplacian method (mean of diffusivity coefficient and its relation with the Euclidean distance) and with the RBF strategy (control points and interpolation concepts), remaining robust in case of large deformations.

For these reasons, we will use IDW interpolation technique for our analysis and we will try to make it more efficient in terms of computational and memory effort, by implementing an *ad hoc* algorithm of geometric and model order reduction.

In the next chapters, after a brief background summary, a detailed presentation of the IDW technique, explanations about geometric and model reduction strategies and finally some applications to different FSI cases are presented.

1.2 Focus

All the interest in FSI problems is principally justified by the reduction of the separation between rigid and elastic modes of aircrafts and ships, with a consequent requirement of an integrated environment where, from the very beginning of the design process, both static and dynamic problems have to be taken into account. The main drawback of this field of study is the expensive cost of tests and simulations. So, especially in the aerospace field, everyday needs of complex numerical simulations of physical phenomena are more and more increasing.

Thanks to the availability of more powerful computing resources and improved numerical algorithms, everyday techniques are faster and more accurate, with results unimaginable years ago. This trend allows to pursue the adoption of high-fidelity mathematical models and numerical methods such as **Computational Structural Dynamics** (CSD) for structural sub-systems and **Computational Fluid Dynamics** (CFD) for aerodynamic sub-systems (see Figure 1.5), until to the more sophisticated *Computational Aeroelasticity*.

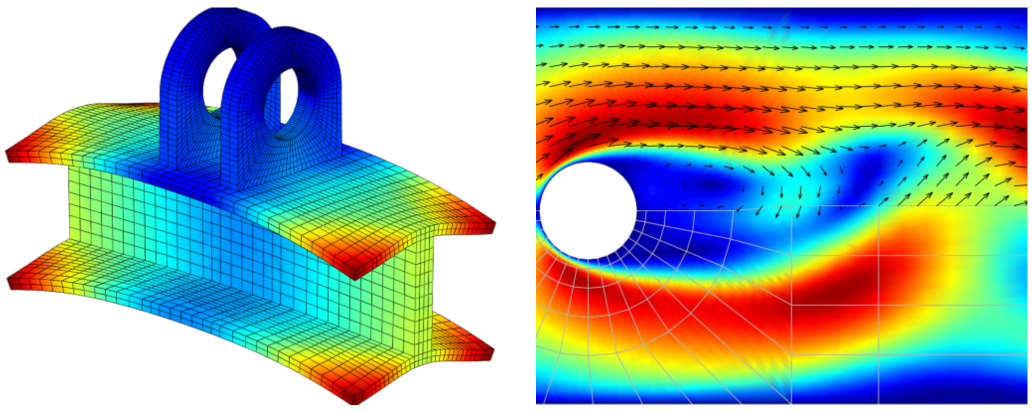


Figure 1.5: Examples of CSD (left) and CFD (right) post-processing visualizations. [1]

From almost twenty years, CFD is massive used in the *aeronautical industry* and its use is now extended from simplified cases up to the most complex configurations and geometrical shapes (see Figure 1.6).

Since the mid-1990s, efforts are underway to fully incorporate the existing body of CFD knowledge into methods and routines used in aircraft design. Thus, numerical simulation has become a principal element in the aircraft design process, because of the flexibility it provides for rapid and comparatively inexpensive evaluation of alternative designs and because it can be integrated in a simulation environment treating concurrently both multidisciplinary analysis and optimization.

Nowadays, in fact, with the increasing fidelity of mathematical modelling and numerical methods within the fruitful research field of CFD, the role of the wind tunnel is changing: it is moving from being a design tool to becoming a tool to validate the designs obtained through the use of CFD, thus producing significant savings in terms of

[1]Realized with MATLAB *(R2015b)* FEATool toolbox.

time and budget. CFD, in fact, through its three-dimensional volumetric results can significantly help to understand the flow physics at very detailed levels, information that, often, are expensive and not easy to obtain from wind tunnel tests.

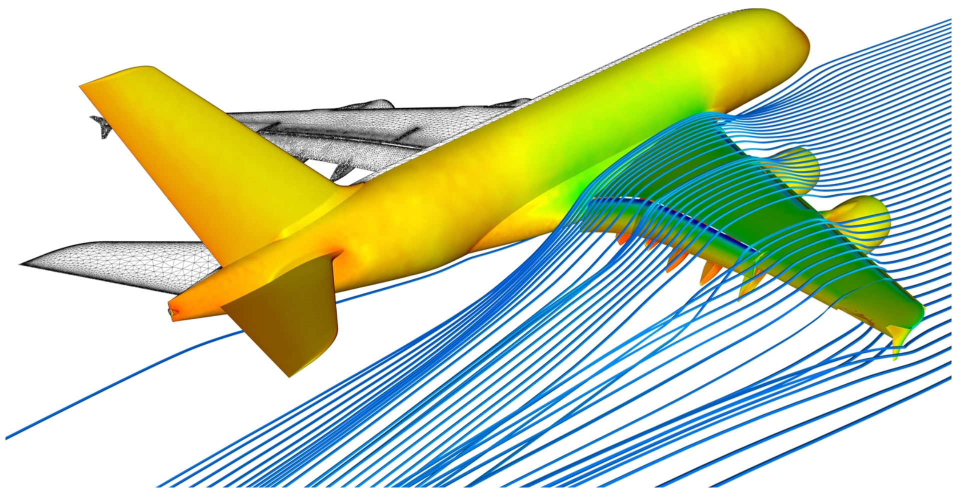


Figure 1.6: Post-processing visualization example of aeronautical CFD application. [2]

Also in the *motor-sport industry*, as *Formula 1* (Figure 1.7), CFD is used to **optimize the shape** of a given part of the car in order to reduce drag and increment the downforce or to study the heat exchange between the brake-disks and the air flowing around them.

In the *maritime industry*, instead, the most common simulations today are carried out to evaluate static and dynamic properties of a given hull shape in flat or wavy sea by simulating the wave pattern around the boat and the forces generated by the fluid (Figure 1.8). CFD and numerical simulations play a fundamental role for yacht designers since they allow to obtain accurate prediction of the kinematic and dynamic properties and a quantitative comparison between different designs. These tools become crucial for racing boats, like in the *America's Cup* sailing competitions, where a gain of a few percent of performance makes the difference between a winning design and an average one.

The desire and, at the same time, the necessity to analyze more detailed fluid dynamic phenomena increases everyday the number of cases considered and consequently the amount of CFD simulations.

[2]Available online at `www.dlr.de/DesktopDefault.aspx/tabid-4551/84_read-9166/gallery-1/gallery_read-Image.1.3497` *(03/04/2016)*.

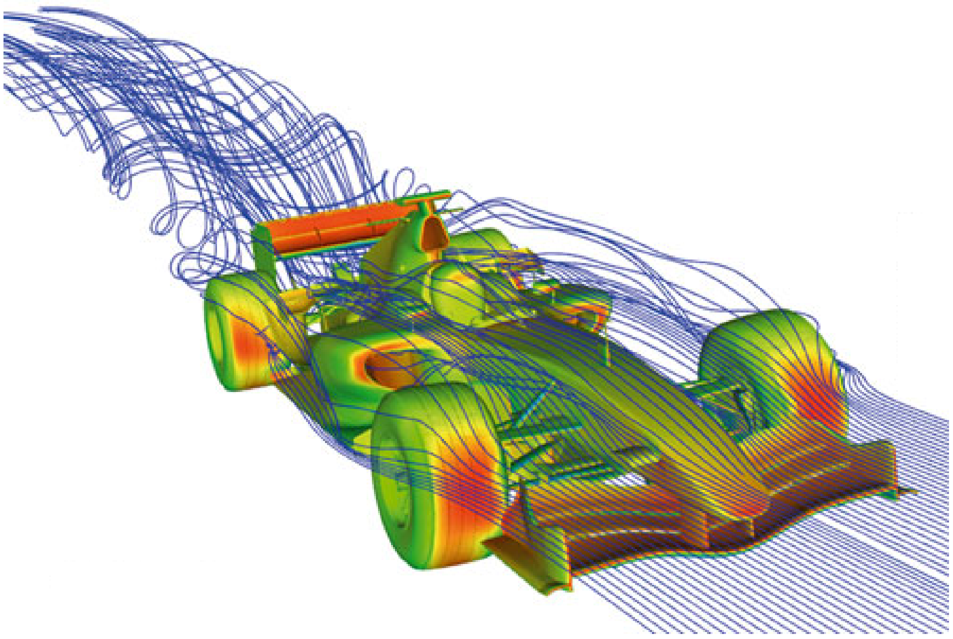


Figure 1.7: Example of fluid flow visualization around a *Formula 1* car obtained by CFD application. [3]

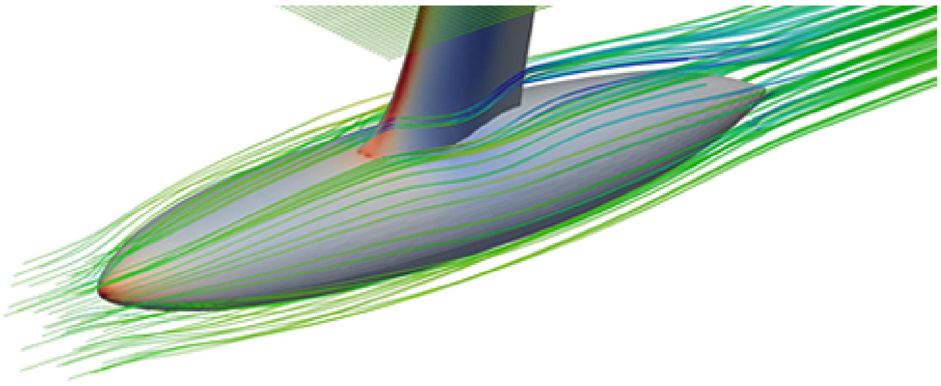


Figure 1.8: Example of CFD visualization of fluid flow around a hull. [4]

Despite the constant increase in available computer power and in improvement of numerical methods and algorithms, some problems and/or applications can still be very demanding. This effort is even more substantial whenever we are interested in repeated

[3]Available online at `www.bitmat.it/blog/news/17536` *(03/04/2016)*.

[4]Available online at `www.leadingedge.it/fluidodinamica.html` *(03/04/2016)*.

solution of the fluid equations such as in flow control or optimal design problems (*many-query* contexts), or in *real-time* flow visualization and output evaluation [71].

These problems represent a remarkable challenge to classical numerical approximations techniques and require huge computational efforts. For this reason, we need to rely on suitable **Reduced Order Methods** (ROMs) – that can reduce both the amount of CPU time and storage capacity – in order to enhance the computational efficiency.

During the last three decades, intense efforts in theoretical foundation, numerical investigation and methodological improvements of reduced order models have allowed to tackle many problems arising in fluid dynamics with different classes of methods (look at [13, 24, 37, 74] for a general overview on reduction methods).

Today, the construction of a reduced order method is crucial to obtain results with an acceptable accuracy depending on a more and more ambitious approximation degree with a reduced computational cost.

The general idea behind ROMs is the reconstruction of the approximate solution by the combination of solutions of the full-order (high-fidelity) problem for some parameter values, properly selected.

Starting from the assumption that the behaviour of a system can be well described by a small number of dominant modes, it is possible to split the numerical approximation in two stages: in the former, we solve the full-order problem for some selected parameter values to compute the *reduced space* of basis solutions (**offline stage**); then, in the latter phase, we find the solution for new instances of the parameter values performing a very inexpensive reduced-order simulation (**online stage**). The expression of the resulting reduced solution is a linear combination of the basis solutions and is computed, for instance, through a **Galerkin projection** onto this reduced space.

In this work, we describe and use a very popular variant of the *reduced basis method* (RBM) according to which the basis used to build the reduced order model are selected via the *Proper Orthogonal Decomposition* (POD). The main idea of POD method is that the solution of a problem can be obtained by a linear combination of well-chosen solutions for specific choices of the parameters (the *control points positions* in our case). In particular this technique reduce the dimensionality of a system by trasforming the original unknowns into new variables – called *POD modes* or principal components – such that the first few modes retain most of the energy of the system.

POD was introduced in several contexts such as turbolence by Lumley [14], fluid dynamics applied to optimal control by Ravindran [66] and blade optimization by Kunish et al. [44, 45] and Aubry et al. [4, 5]. Since, in the past, RBM did not fully exploit the *Offline-Online procedure*, much effort has been devoted to the efficient splitting of these two steps in the last decade.

Most of the RBM applications deal with physical or engineering parameters, such as viscosity, transport velocity, Peclet number, Biot number, Young modulus or thermal conductivity. There are, instead, just a few applications dealing with simple *geometri-*

cal parameters, such as length or thickness that characterize the problem ([63]). Hence, the application to complex geometry parametrization is quite innovative and very recent. Among these applications, the employment for shape optimization of racing car components by Günther [33], the treatment of potential flows by Rozza [69] and the coupling of isogeometric analysis and ROMs by Salmoiraghi et al. [73] achieve significant importance.

Thus, the main idea behind ROM techniques, specially for FSI problems, is to reduce as most as possible the computational effort. Shape parametrization represents an important task for ROM to guarantee efficient computation, versatility of the approach and automatic geometrical representation by:

i) reducing the complexity in the description of the shapes and their deformations,

ii) representing geometrical properties by means of low dimensional spaces (*control points*),

iii) reducing the computational effort by simplifying the deformation procedure and its management.

The goals introduced are valid also for **shape optimization** (e.g., in industrial and bioengineering applications), **shape reconstruction** (e.g., in biomedicine) and, in general, for problems with multiphysics interfaces which require multiple iterative evaluations in real-time and many query contexts.

For these reasons, the use of shape parametrization techniques like FFD, RBF and IDW has become mandatory in order to deal with more involved problems such as 3D complex applications [43, 52, 58]. The shape parametrization schemes have allowed to use reduced order methods also in the geometrical parametric space [32], not only in time and physics, and to realize optimization of complex industrial and bioengineering shapes.

In this thesis we present the advantages of the use of IDW interpolation technique in combination with ROMs, describing its characteristics and some possible improvements in order to reduce computational and memory effort, by preserving an acceptable quality of results, according to the goals sought by CFD.

1.3 Thesis Structure

The thesis has been organized as follows.

In **Chapter 1** we have described the context where the cited technique is developed. First of all, we have presented Fluid-Structure Interaction problems and cited some of the common strategies adopted to parametrize and solve them. In these procedures, in particular concerning mesh motion aspects, shape parametrization techniques play a very important role, i.e., the reduction of the system parameters. For this reason, the discussion has been moved to the real focus of this work: the shape parametrization techniques, their advantages, bottlenecks and possible developments for mesh motion problems. Furthermore, known the increasing reliability and recent massive use of Computational Fluid Dynamics and Computational Structure Dynamics for aeronautical and naval applications, we have briefly described the evolution and the new challenges of these numerical methods for which mesh motion problems represent an important load on the performances. Thus, among the requirements to operate in a CFD or a CSD framework, the most important one is the cheapness in terms of CPU time and memory effort for which, techniques like the common RBF, FFD and IDW have to be improved considerably. Looking to the conclusions of some previous works [68, 31, 32, 8] and starting from them, we have decided to improve the computational performances of Inverse Distance Weighting, a very used shape parametrization strategy, in terms of flexibility and simplicity of implementation.

In **Chapter 2** we introduce the *standard* formulation of IDW and its advantages and drawbacks. In order to use IDW technique for mesh displacement computation, the classic formulation is extended to a matrix form where the system unknowns are the displacement vectors of mesh internal nodes. Following this transformation, we present some IDW applications starting from the simplest square example up to the more complex 3D aeronautical and naval cases. As anticipated, the results in terms of CPU time and matrix dimensions suggest a radical improvement.

Since the parameters for IDW technique are represented by some known data points (*control points*) lying typically on the interface, in **Chapter 3** we realize an *ad hoc* algorithm to reduce the number of these parameters defining a corresponding smart distribution based on concentric annuli of user-defined width. Moreover, to solve problems related to the violation of interpenetration constraint and to increase the number of admissible interface displacements, we introduce a '*dimensional cascade*' procedure which provides for the IDW application in succession of steps. It starts with a 1D deformation (vertices and edges), then a 2D displacement of the boundary nodes (faces), to finally compute the 3D (internal) final mesh configuration. This new process, namely the Reduced Inverse Distance Weighting (RIDW), is validated on the same examples used for the *standard* formulation in the previous chapter. The results show that RIDW strategy allows to considerably reduce the memory storage and, almost for the most complex cases, the CPU time with respect to the ones required by the *standard* IDW technique.

In **Chapter 4**, we further reduce the computational time of the RIDW application. In order to achieve this goal, we exploit the advantages of Reduced Order Methods and their process of splitting between *Offline* and *Online* phases. Inspired by Proper Orthogonal Decomposition, in the expensive but one-time *Offline* phase, our algorithm collects a sampling of RIDW solutions for several given values of control point displacements and extract the solution from the reduced space by a Galerkin projection. In this way, in the *Online* stage, it is possible to quickly find the solution for an arbitrary assigned interface movement by a simple projection on the reduced space of the solution. Also for this new development, the validation is realized on the same examples analyzed in the previous chapters with very promising results. The CPU time decreases of two orders of magnitude for 3D complex applications.

The innovative reduction process implemented in this thesis has obtained great results in terms of memory effort reduction and solution velocity, two crucial aspects of CFD applications.

Our geometric tool of reduction has allowed to work with a very small number of parameters containing the error in the displacement computation. Furthermore, the user can decide the solution accuracy degree by the width of annuli of reduction and the type of interface displacement.

The construction of a new Reduced Order Method for RIDW strategy has considerably accelerated mesh motion solution computation so that RIDW technique can be used also in real-time and optimization problems, where it is required a repeated evaluation of the solution for different values of parameters.

This work is a first attempt to improve IDW technique. First of all, a validation of the benefits of this strategy can be furnished via a CFD example with a complete mesh motion and fluid dynamics analysis. Many additional ideas can be investigated for the realization of a smart standard procedure for the control points choice and reduction by considering, for example, physical and geometrical aspects like main stream direction or the gradient of the displacement. Furthermore, the advent of new numerical methods, e.g., Proper Generalized Decomposition (PGD), can represent elements of inspiration also for the construction of other Reduced Order Methods and to introduce performance improvements of the IDW and RIDW strategies.

Chapter 2

Standard Inverse Distance Weighting

This Chapter introduces the shape parametrization technique chosen to deal with the fluid-structure interface as well as the mesh motion problems, namely the Inverse Distance Weighting (IDW) approach.

First of all, we define a shape parametrization map to be a parametric function $T(\mathbf{x};\boldsymbol{\mu})$ from a reference domain $\Omega \subset \mathbb{R}^d$, with $d = 2, 3$ to a deformed one $\Omega_o \subset \mathbb{R}^d$, by displacing a small set of control points, the so-called parameters $\boldsymbol{\mu} \in \mathbb{R}^d$. These displacements induce the shape deformations $\Omega_o(\boldsymbol{\mu}) = T(\Omega;\boldsymbol{\mu})$. In this way, it is possible to efficiently represent a large family of admissible shape configurations [50].

A first simple solution leads us to identify the parametric map $T(\mathbf{x};\boldsymbol{\mu})$ via an interpolation method. In fact, assuming that a finite number N of couples $(\mathbf{x}_i, d_i)$ are given, where $\mathbf{x}_i$ is the position of the data point D_i and d_i is the corresponding data value, we look for an interpolation function $d = f(\mathbf{x})$ to assign a value to any location $\mathbf{x}$ in the space. This function has to interpolate the data d_i of the points $\mathbf{x}_i$, (i.e., $f(\mathbf{x}_i) = d_i$), and meet the user expectation about the phenomenon under investigation. Furthermore, the function should be accurate and, at the same time, suitable for computer applications at a reasonable computational cost, all features typical of IDW interpolation strategy.

2.1 An introduction to Inverse Distance Weighting

Several methods to solve interpolation problems have been developed in the last years for rectangular and triangular grids [12, 17, 35, 53, 79]. The current exact interpolation methods are of two kinds: global or local ones. The former use a global function to interpolate data. However, often the complexity to find a global function which interpolates exactly all the data becomes unmanageable and not viable. The latter, in contrast, are based on a suitably-defined collection of local functions which match appropriately at their extremities. Inverse Distance Weighting (IDW) technique belongs to the latter class and it is constructed so that the sub-domain for each local function is automatically defined and the function is globally continuous differentiable.

The IDW strategy has been used for the first time for two-dimensional interpolation problems in [77] and later extended to three-dimensional applications in [29]. It is based on the construction of a surface related to a weighted average of the values at the data points, and the weighting is a function of the distances to the points. Thus, today IDW has extended its applications in three-dimensional cases, representing one of the most common used techniques to solve mesh motion and FSI problems [23, 53, 84, 83].

The simplest form of IDW interpolation was proposed by Shepard [77] for the interpolation of a scalar function $u : \Omega \to \mathbb{R}$. In his original formulation, the value of u at any point $\mathbf{x}$ in the plane was a weighted average of $\mathcal{N}_c$ values at the data points $\mathbf{x}_k$. Thus, given $\mathcal{N}_c$ control points located at $\mathbf{x}_k$, with $k = 1, 2, ..., \mathcal{N}_c$, the IDW interpolation of a set of samples $u_k = u(\mathbf{x}_k)$ is given by:

$$\hat{u}(\mathbf{x}) = \sum_{k=1}^{\mathcal{N}_c} \frac{w(\mathbf{x}, \mathbf{x}_k)}{\displaystyle\sum_{j=1}^{\mathcal{N}_c} w(\mathbf{x}, \mathbf{x}_j)} u_k, \tag{2.1}$$

where $w(\mathbf{x}, \mathbf{x}_i)$ represents the weighting function:

$$w(\mathbf{x}, \mathbf{x}_i) = \parallel \mathbf{x} - \mathbf{x}_i \parallel^{-p}, \tag{2.2}$$

being $\parallel \mathbf{x} - \mathbf{x}_i \parallel \geq 0$ the Euclidean distance between $\mathbf{x}$ and the data point $\mathbf{x}_i$, and *p* an integer number.

It is clear that the terms $w(\mathbf{x}, \mathbf{x}_i)$ can be easily collected in a matrix. Consequently the IDW technique can be expressed in a *matrix formulation*:

$$\mathbf{u}(\mathbf{x}_s) = \sum_{k=1}^{\mathcal{N}_c} \frac{\boldsymbol{IDW}(s,k)}{\sum_{j=1}^{\mathcal{N}_c} \boldsymbol{IDW}(s,j)} \mathbf{u}_k \quad \forall s = 1, ..., \mathcal{N}_s \tag{2.3}$$

where $\mathcal{N}_s$ represents the number of data to be interpolated and $\boldsymbol{IDW}$ represents the $[\mathcal{N}_s \times \mathcal{N}_c]$ matrix, whose generic (s,k)-component is given by:

$$\boldsymbol{IDW}(s,k) = w(\mathbf{x}_s, \mathbf{x}_k) \tag{2.4}$$

The matrix $\boldsymbol{IDW}$ allows to save *once* and for all the weighted distances between the control points and the interpolation nodes with respect to the undeformed configuration.

Among the several benefits of IDW, the first one to be underlined is the **easiness** of implementation. The strategy, in fact, can be simply expressed by the algebraic formulation (2.3), thus requiring just to assemble the matrix $\boldsymbol{IDW}$ that collects all the distances between control and internal points and compute matrix-vector products.

Nevertheless, considering that the *standard* formulation of IDW technique suggests to select as control points **all the nodes belonging to the boundaries of the mesh**, the $\boldsymbol{IDW}$ matrix can assumes considerable dimensions, particularly for complex meshes (i.e., aeronautical and naval ones). This consideration makes the control point selection a crucial phase of the IDW technique.

Another important advantage of IDW is the possibility of dealing with N*-dimensional spaces*. In fact, in order to compute the solution, the IDW technique uses the distance between points without any restrictions about the dimensionality of the space they belong to. The interpolated value depends on a ratio between **norms** of vectors representing distances. In this way, the dimensionality of the space of the points does not influence in any way the solution.

For the same reason, IDW allows to work with **any type of grid**, resulting a winning strategy in the context of mesh motion problems.

On the other hand we have, within the original *Shepard formulation*, low performances due to high costs of the interpolation (which are of the order of $\mathcal{O}(\mathcal{N}_c)$). A too large weight is assigned also to distant nodes by considering, with reference to (2.1), $\mathcal{N}_c$ as the number of data points. When the number of data points is large, the calculation of u becomes proportionately longer. Eventually, the method becomes inefficient and impractical. Furthermore, only the distance from the data points $\mathbf{x}_i$ to $\mathbf{x}$, and not the direction, is considered. Therefore, the following two configurations of collinear points, for example, would yield identical interpolated values at P:

(a) $D_1 - - - - - - - - - D_2 - - - - - - - - - P - - - - - - - - - - - - - - - - - - - D_3$;

(b) $D_1 - - - - - - - - - - - - - - - - - - - P - - - - - - - - - D_2 - - - - - - - - - D_3$.

We expect that the value at P in the configuration (a) should be closer to the value at D_3 than in configuration (b), and, conversely, for the value at D_1, because the presence of an intermediate data point should be expected to screen the effect of the most distant point. On the other hand, the zero directional derivatives obtained at every data point D_i represent an arbitrary and undesiderable constraint on the interpolated surface and the computational error can become significant in the neighborhood of points D_i. For these reasons, the *standard* IDW technique can be modified by the introduction of some correction terms that take into account the direction and the computational error as proposed in [77].

A first observation about (2.3) and (2.4) suggests the similarity between the IDW strategy and Laplacian method. In both cases, in fact, the influence of data point displacement on the interior points decreases as the distance between them increases. Moreover, for IDW technique, the Euclidean norm is weighted by a number, *p*, that has the same meaning as α in the definition of the diffusivity coefficient in (1.11). In general, the value of p is, often but not always, equal to 2 to guarantee a quadratic decreasing dependence between control and interior points. However, it is interesting to study the effects of varying p, from a geometrical point of view, since the selected value produces considerable variations on the displacement of the interior nodes.

In particulars, the value of p plays a very important role, influencing the constraint of no-compenetration between fluid and structure. Thus, the choice of this value is fundamental to avoid that some nodes of the fluid grid, for example, locate over the interface as a result of the IDW computation. This is the case of large value of displacement for a single control point. A fine tuning process for p, as the one realized for α in (1.11) and Lamé constants in (1.13), has to be performed to prevent these phenomena.

In Figure 2.1 we show the results obtained for a simple mono-dimensional case for different values of p. We highlight the geometrical effects due to these variations on the interior node displacements computed by IDW technique.

We observe that, as the value of p increases, the curves obtained are less and less sharp. Accordingly to the role of p, this trend shows that, if we increase the value of p, we weight less and less the influence of the furthest data points. The negative power in the definition (2.2) denotes, in fact, an inversely proportional relation between the Euclidean distance $\| \mathbf{x} - \mathbf{x}_i \|$ and the weighting function $w(\mathbf{x}, \mathbf{x}_i)$.

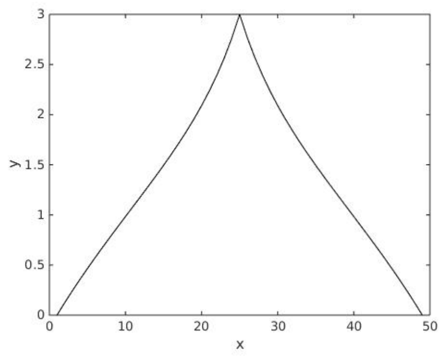
In order to further understand the role played by p, it is possible, for example, to compute the internal node displacement of a rod, imposing one of its natural vibration modes at the boundaries. The undeformed configuration of the mesh considered is provided in Figure 2.2.

By computing the internal node displacement of a three-dimensional rod, while imposing to the boundary faces the motion described by the function:

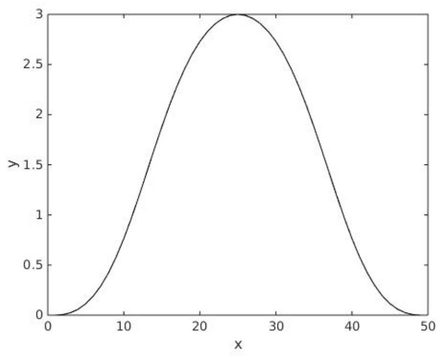
$$y(x) = \frac{\cos(x)}{2}, \tag{2.5}$$

we observe that using a value for p equal to 2 the result is the one represented in Figure 2.3, where it is evident that the internal nodes across the boundaries (in the red ellipsis), locating out of the domain. Since this configuration violates the no-compenetration constraint, we perform a tuning of the value of p, according to which the adopted value is equal to 4.

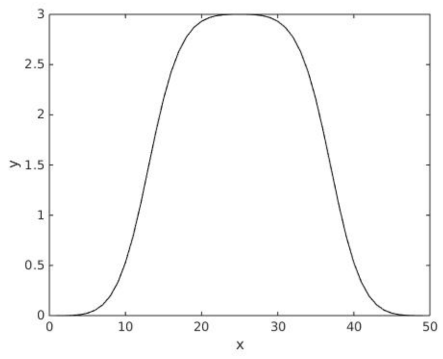
Indeed, contrary to the case for $p = 2$, by choosing $p = 4$, the motion of the nodes closer to the boundary affects more effectively the internal nodes displacement, reducing the effect of the null displacement associated with the (fixed) more distant nodes. The result for $p = 4$ is represented in Figure 2.4.



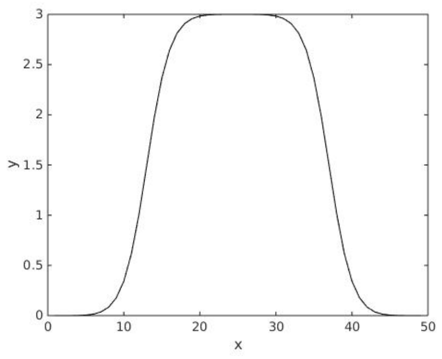
(a) $p = 1$



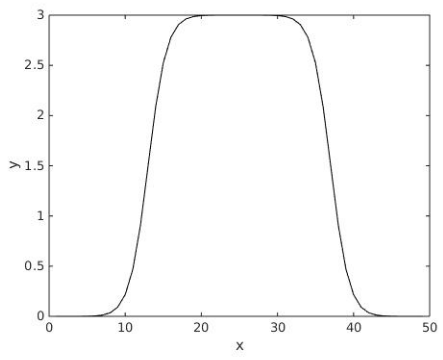
(b) $p = 2$



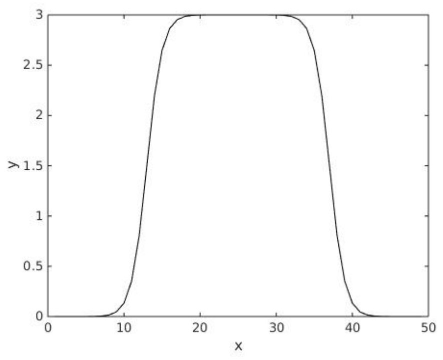
(c) $p = 3$



(d) $p = 4$



(e) $p = 5$



(f) $p = 6$

Figure 2.1: Displacements obtained for different values of p

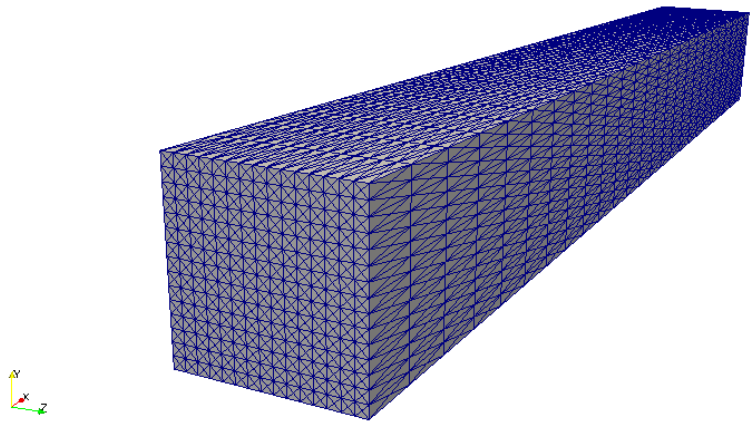


Figure 2.2: Undeformed 3D rod mesh.

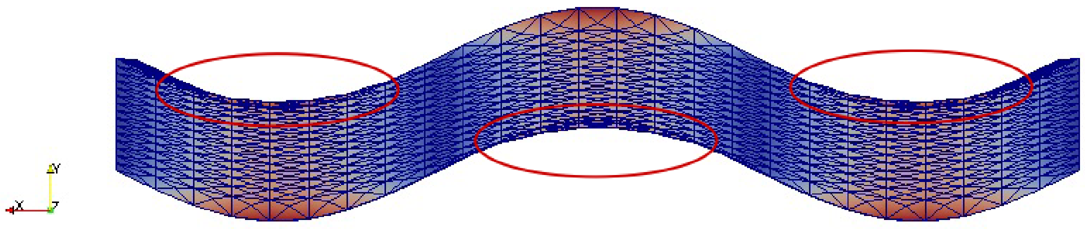


Figure 2.3: No-compenetration constraint violation for p value equal to 2.

The example just seen has driven us to choose the value for p equal to 4 for all the considered applications. This value, in fact, turnes out to be correct for all the cases analysed, avoiding the violation of the no-compenetration constraint.

As anticipated, a fundamental aspect is represented by the **control points selection**, in order to represent, in the best possible way, the deformation of the interface.

In fact, the control points have a crucial role in preserving the mesh quality. In the IDW computation, we have to consider that not only the nodes moved are important, but also the ones with null displacement. These ones, in fact, are included in the computation of the interior node displacement by the *denominator* of (2.1). Since the denominator is characterized by the sum of the distances between the node located at $\mathbf{x}$ and **all the control points** $\mathbf{x}_k$. The choice of the $\mathcal{N}_c$ control point positions, in addition

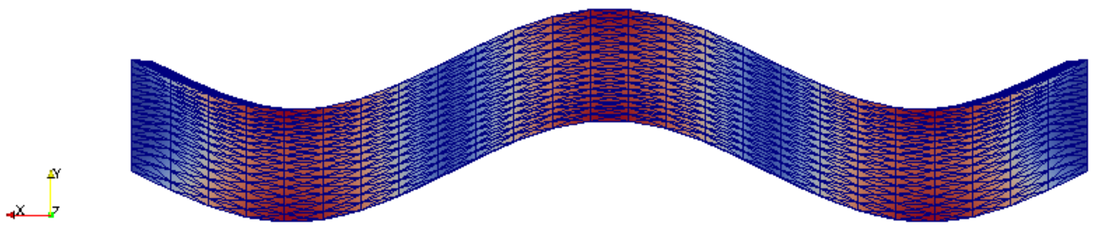


Figure 2.4: Rod mesh deformed configuration computed by IDW assuming p equal to 4.

to the optimal description of interface displacement, has to take into account the parts that are not moved in order to respect the constraints imposed by the fixed boundaries.

In this way, a good choice for the control point position allows to respect the constrain about the admissible displacements and to obtain smoother solutions.

In this chapter we comply with the common use of the IDW technique, considering as control points all the nodes belonging to the mesh boundaries. In the next chapter we will explain a reduction procedure of the number of the control points, in order to limit the memory storage and the computational effort required by the *standard* IDW formulation.

2.2 IDW for engineering applications

After introducing about the principal characteristics and potentialities of IDW, we discuss its important role in engineering fields such as **mesh motion** and **shape optimization** problems.

For **mesh motion** cases the IDW algorithm is used to obtain the displacement of mesh nodes starting from the motion of some control points via an average moving interpolation. The main idea of this strategy lies in the possibility of estimating the local trends by analyzing a moving average of nearby data points, also highly variable. The IDW technique, therefore, computes a value for each grid node by the average of the weighted sum of all the surrounding data point distances. Due to the inverse proportionality, all the data points lying progressively further from the considered grid node influence less than the closer ones. This kind of point-to-point interpolation allows to analyze arbitrary mesh topologies and remains robust in case of large deformations, leading to a very useful flexibility in the applications.

It can be estimated that, in large-scale three-dimensional applications, the computational costs of the mesh motion are dominated by the system solution step. Therefore, in order to significantly reduce the computational costs of mesh deformation it is necessary to focus on reducing the computational cost for solving the system of $\mathcal{N}_s$ equations. In this case, differently from the other techniques, IDW provides a good advantage leading to an algebraic expression for the internal point displacements as function of the boundary deformation (e.g., where the boundary can be the structural interface). In fact, for general geometries, IDW results automatically in a global parametrization that can handle translations and rotations.

In many fields of engineering applications (aero-elasticity, turbomachinery design, modelling of cardiovascular system, high performance boat dynamics, etc.) the IDW strategy plays an important role as well as in **shape optimization** applications. Several works in literature use this technique to obtain optimized mesh and shape for aerodynamic profiles [83] and catamaran geometry [29].

The main advantage of IDW technique is the ability of solving problems through a 'small' number of parameters. Although rarely, it is possible to use some parameters like camber, thickness, width, etc. to obtain and test several configurations and sizes for the same shape of the object under investigation, especially in the first phases of a project. In this way, it is possible to optimize the shape of wings, hulls and other fluid dynamic and aerodynamic objects.

In order to obtain the optimized shape in an FSI framework, the new mesh configuration will be provided, change after change, to the fluid solver for the physical variables computation as long as the optimum fluid dynamic shape is not found. The shapes achievable as a result of a deformation are of widely different kinds. An example can be the achievement of a 'segment of wing' based on lozenge profile (see the frontal face in Figure 2.6) from a rod with a square base (Figure 2.5).

We will present in the following sections several applications of the *standard* IDW to investigate the effects in terms of internal node displacements.

We have decided to use both two-dimensional and three-dimensional cases, using triangular and tetrahedral meshes, respectively. Some of these will be compared with innovative and reduced IDW implementations, described in Chapter 3, to observe the differences in terms of *CPU time* and the variations on the internal node displacements.

We have realized all the algorithms in C++ language, using the open source library *libMesh* [38] which provided some tools and functions for the mesh management, particularly for the nodes identification. Furthermore, we have decided to use *Paraview* [2], a widely used visualization software, in the post-processing phase to observe and compare the initial configurations to the ones obtained at the end of the mesh motion computed by IDW. In this way we are able to detect graphically the displacement of the internal nodes and verify the control nodes movement. As anticipated, for this kind

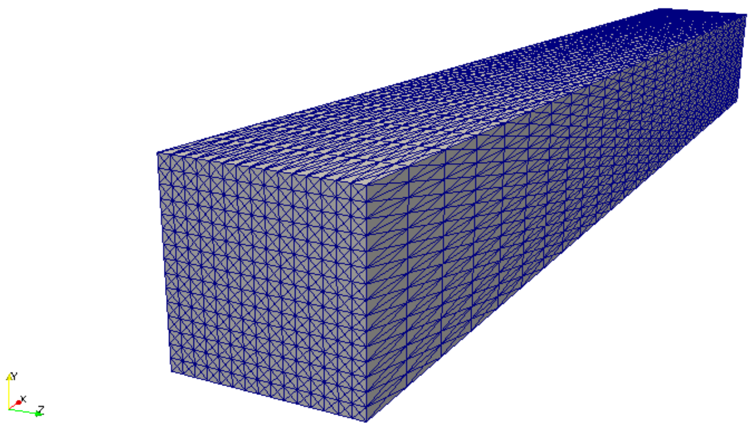


Figure 2.5: Undeformed mesh of a rod used to apply IDW technique for a shape optimization.

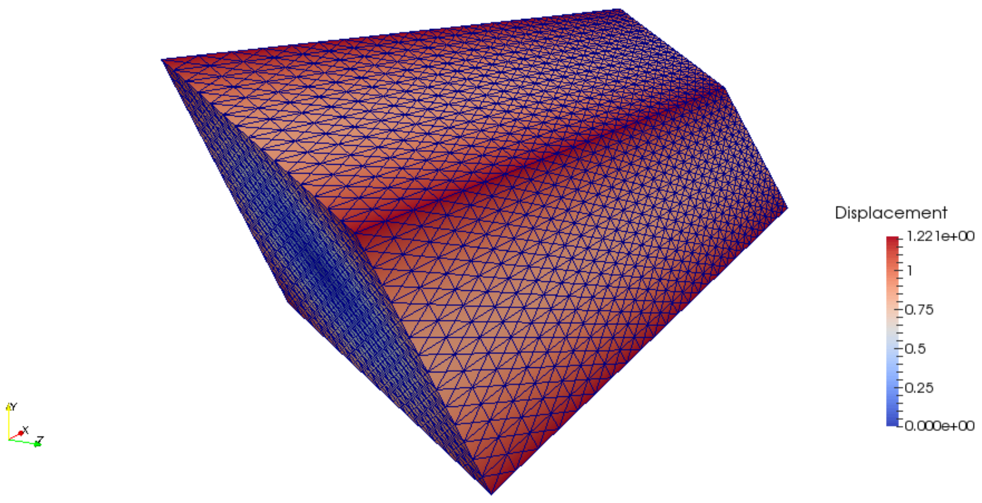


Figure 2.6: 3D wing with a lozenge profile.

of implementation we have used **all the boundary nodes** as control points. This is a choice as simple as delicate in terms of the memory load for the IDW matrix allocation that, in this case, can demand large memory space.

2.2.1 Deformation of a square

First of all we consider a very simple case: the square $(0,\pi)^2$ in *(x,y)*-plane with deformable upper boundary. We impose to the upper boundary a movement based on the following function:

$$y = \frac{\sin(x)}{2}. \tag{2.6}$$

The triangular mesh in Figure 2.7 represents the reference configuration, while Figure 2.8 shows the deformed mesh.

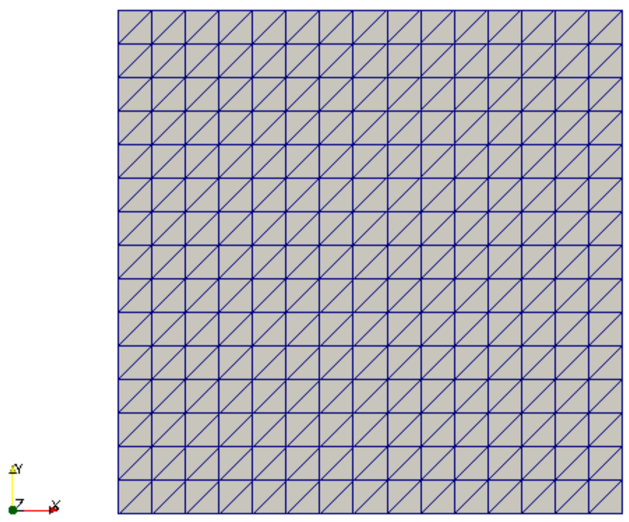


Figure 2.7: Square undeformed mesh.

The properties of the mesh and the results obtained are summarized in Table 2.1.

In Figure 2.8 we observe the representation of the internal node displacements. As expected, the internal nodes far enough from the control points are not affected by the boundary motion. We can also observe the new position of all internal nodes on the deformed configuration. In particular, the internal nodes close to the boundary have followed the movement of the boundary nodes with an amplitude of displacement decreasing as the distance increases. Particularly, in the middle of the moved boundary,

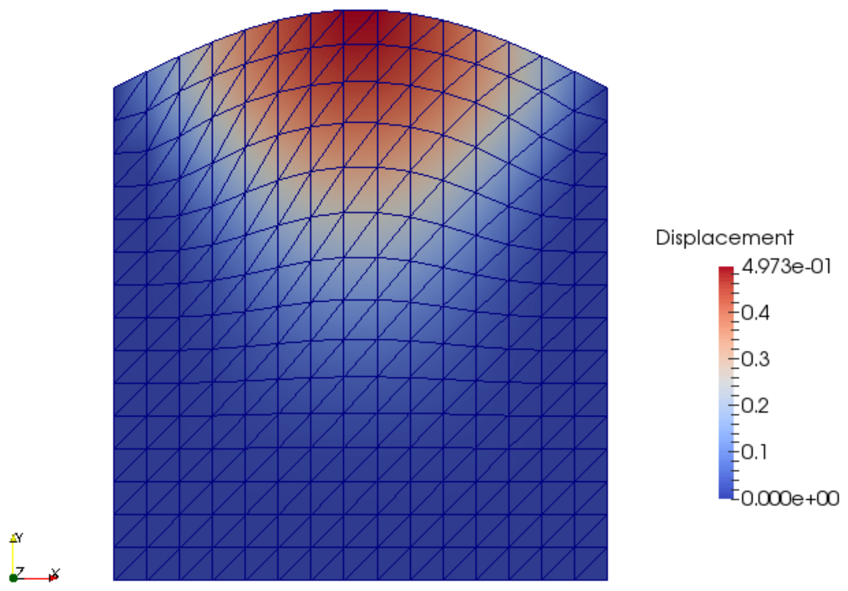


Figure 2.8: IDW computed square mesh deformation following the upper boundary motion in (2.6).

Length of square side	π m
Number of elements	450
Number of nodes	256
Number of control points (boundary nodes)	60
Number of internal nodes	196
CPU Time	0.0972 s

Table 2.1: Quantitative information related to the deformation of the square in Figure 2.7.

where the amplitude of the displacement is greater, the internal nodes have been shifted more intensely.

The results, summarized in Table 2.1, show a relative small CPU time value (0.0972 s) because of the coarse discretization (just 256 nodes).

Unfortunately, the same result is very hard to obtain for finer meshes. Geometrically, more complex shapes require finer discretizations to avoid substantial losses of domain data with the consequent demand of longer CPU time as well as the assembling of a larger IDW matrix.

2.2.2 An airfoil

As second test case, we consider an hypothetical flow domain around a 2D NACA0012 airfoil, a symmetric and very used aerodynamic profile [1]. The choice of the profile shape is completely indifferent for the IDW procedure due to the versatility of this interpolation technique. In fact, our choice has been driven only by its simplicity and wide use in aerodynamics.

NACA0012 is a symmetric NACA aerodynamic profile typically used for wings, fan, propeller and helicopter blade. Its shape is, like others 4-digit profiles, represented by the following function:

$$y(x,t,c) = 5tc\left[0.2969\sqrt{\frac{x}{c}} - 0.1260\left(\frac{x}{c}\right) - 0.3516\left(\frac{x}{c}\right)^2 + 0.2843\left(\frac{x}{c}\right)^3 - 0.1015\left(\frac{x}{c}\right)^4\right], \tag{2.7}$$

where:

- c is the chord length;
- x is the abscissa position along the chord;
- t is the maximum thickness expressed as percent fraction of the chord (0.12 in our case);
- $y(x,t,c)$ is the half of profile thickness in x.

Mirroring the curve resulting from the function in (2.7) with respect to the axis x, we obtain the geometrical shape represented in Figure 2.9.

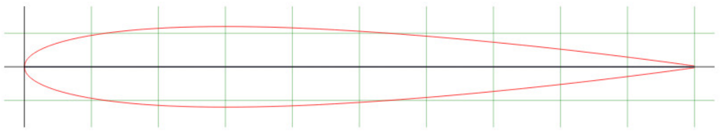


Figure 2.9: NACA0012 profile.

We use the profile in Figure 2.9 to build a rectangular flow domain characterized by a hole with the NACA profile shape. In this way, we have the possibility to investigate the displacement of the internal nodes of the fluid mesh, simulating a profile movement or a size variation by modifying the hole. As for the previous case, we provide the two

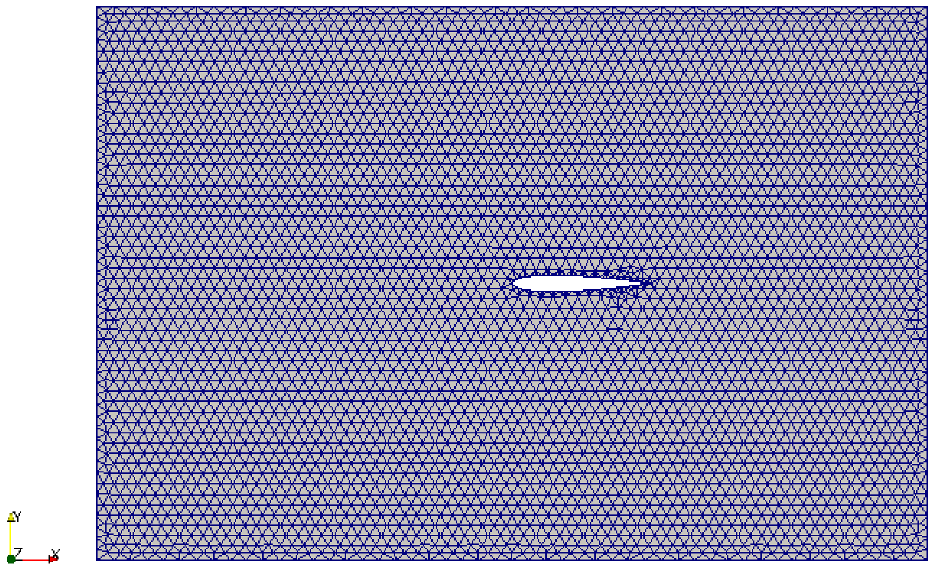


Figure 2.10: Fluid mesh around a NACA0012 profile with null angle of attack.

configurations, the undeformed one (Figure 2.10) and the deformed one yielded by the IDW application (Figure 2.11). Table 2.2 summarizes the quantitative results.

Mesh dimensions	6 m $\times$4 m
NACA0012 chord length	1.01 m
Number of elements	7798
Number of nodes	4049
Number of control points (boundary nodes)	300
Number of internal nodes	3749
CPU Time	2.0257 s

Table 2.2: Quantitative information related to the deformation of the mesh in Figure 2.10.

In particular, the deformed configuration is obtained via a rotation around axis z of NACA0012 profile of an angle equal to $\pi/5$ with respect to origin, fixed in the profile leading edge. This is a typical case of aerodynamics study, where the flow grid variations for different angle of attack of a certain profile are investigated. In order to analyze

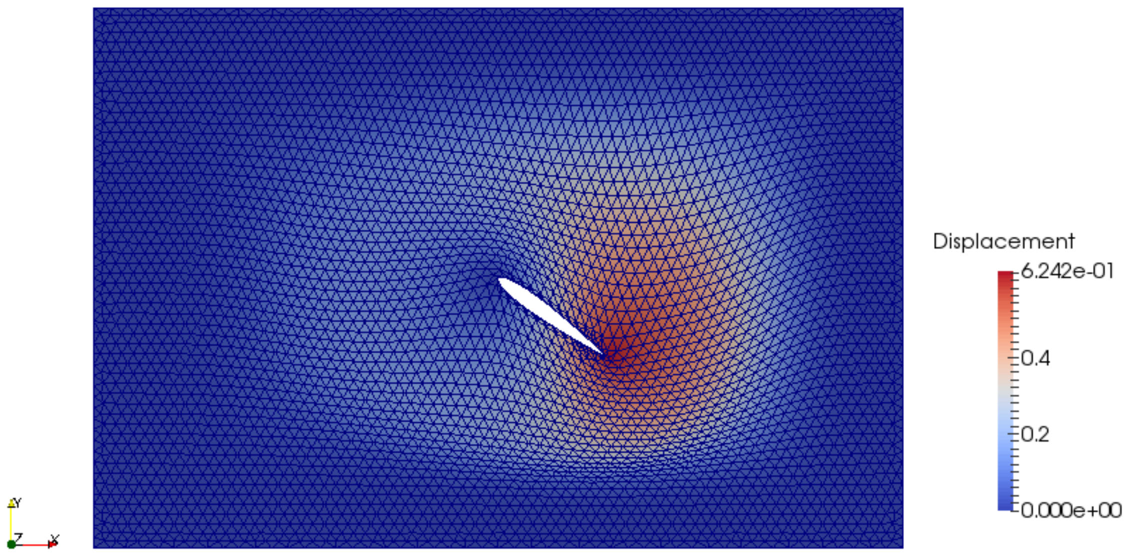


Figure 2.11: IDW computed fluid mesh deformation around a NACA0012 with angle of attack equal to $\pi/5$.

the consequence of a finer mesh on the required time, although the two grids are of completely different shape, we can appreciate the difference on the CPU time respect to the mesh in Figure 2.7. The CPU time is considerably increased due to the larger number of grid nodes (4049 versus 256). This increase underlines the direct proportionality between the number of values to be interpolated and the required computational effort.

Moreover, it is possible to observe that, also the nodes lying ahead to the profile leading edge are affected by the boundary movement.

2.2.3 A 2D mesh motion around a moving rectangular structure

The following test case mimics a study operated in the wind tunnel.

As for the applications previously described, if we want to investigate the flow grid node motion when the structure deforms, we need to create a mesh around it and reproduce its movement by considering its boundary as an interface, to observe the resulting motion of the internal nodes.

We fix a wing segment or, more simply, a cantilever beam to the wind tunnel left wall. In the 2D case, the analysis about internal node displacement is performed on a plane (two-dimensional mesh). It is possible to have a more realistic view in the next three-dimensional test case. In a 3D space, in fact, the object is reproduced in its complete shape and, consequently, the extraction of information depends on the position of the plane we want to project the deformed mesh on.

Also in this case, Figures 2.12 and 2.13 reproduce the undeformed configuration and the one resulting after the motion.

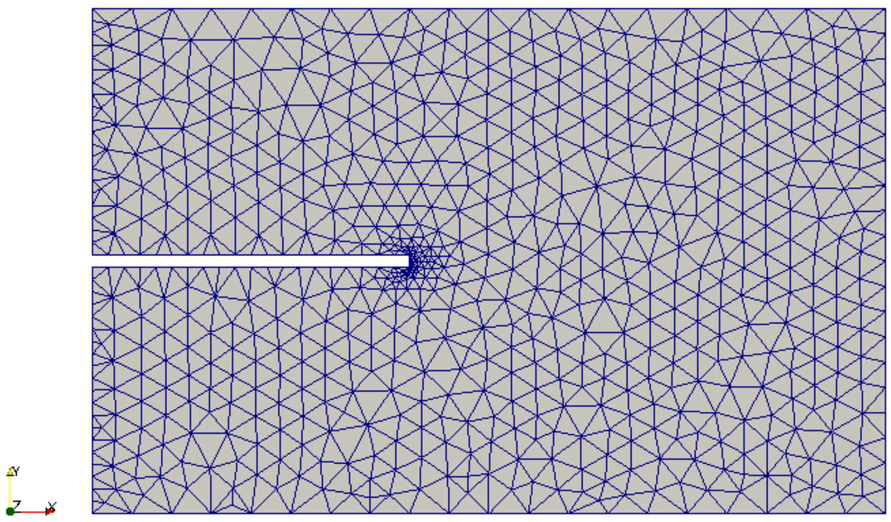


Figure 2.12: Undeformed 2D fluid mesh around a cantilever beam.

The properties of the triangular mesh used in this case are listed in Table 2.3.

Mesh dimensions	6π m $\times 16$ m
Rectangular hole dimensions	2π m $\times 1$ m
Number of elements	1360
Number of nodes	761
Number of control points (boundary nodes)	160
Number of internal nodes	601
CPU Time	0.255 s

Table 2.3: Quantitative information related to the fluid mesh motion around the cantilever beam in Figure 2.12.

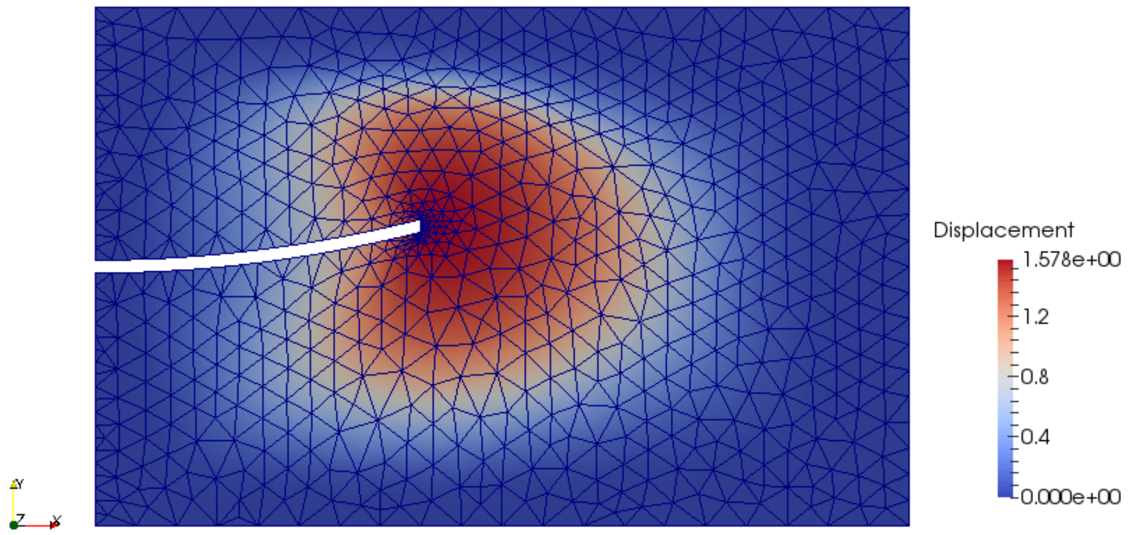


Figure 2.13: IDW computed 2D fluid mesh deformation following the motion of the beam.

Also for this application, the CPU time is quite low (0.255 s), due to the relative coarse grid composed of 761 nodes only. It is very important to see how all the internal nodes around the moved boundary are displaced alongside the direction of the motion, thus conserving a good mesh quality and avoiding stretched triangular elements. Accordingly to the discussion in Section 2.1, the nodes far enough from the displaced boundary are not affected by the motion. We remark that, since all the boundary nodes are considered as control points, the boundary nodes that are not displaced are fixed in their position due to the null displacement imposed to them in the algorithm. Anyway, they are influencing the internal node movement through the *denominator* of (2.1) where their distance from any internal points is considered with the same weight of the moved control points.

2.2.4 A 3D mesh motion around a deformed NACA0012-based wing

We are going to present a three-dimensional case, starting from the three-dimensional extension of the previously considered 2D configuration.

We have applied the same kind of displacement as in the previous section on a parallelepiped holed. The shape of the hole reproduces a wing, supposed clamped to the left at the wall of a test chamber in a wind tunnel. Imaging this scenario, we keep, also in this case, fixed the six parallelepiped external faces while displacing the top, bottom and right faces of the wing. Figures 2.14 and 2.15 show the 3D shape described above.

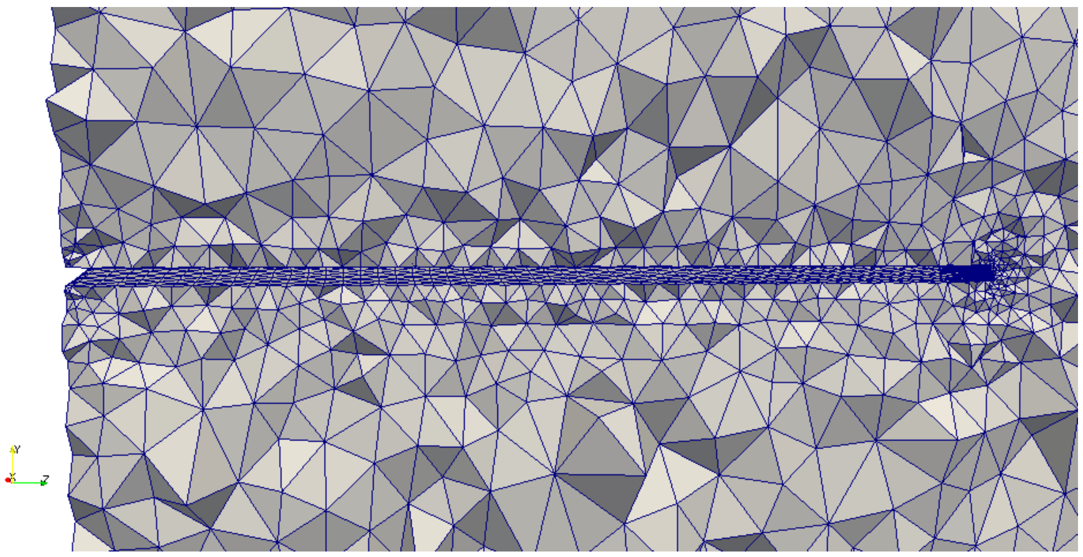


Figure 2.14: Undeformed 3D fluid mesh around a NACA0012-based wing.

In this case, the mesh is composed by tetrahedral elements and characterized by the values in Table 2.4.

Mesh dimensions	$10 \text{ m} \times 5 \text{ m} \times 4\pi \text{ m}$
Wing longitudinal dimension	2π m
NACA0012 profile chord length	1.01 m
Number of elements	169598
Number of nodes	36036
Number of control points (boundary nodes)	14126
Number of internal nodes	21910
CPU Time	446.485 s

Table 2.4: Quantitative information about the 3D fluid mesh represented in Figure 2.14.

The results in Figures 2.14 and 2.15 and listed in Table 2.4 denote the complex increase in terms of shape representation and discretization. The number of nodes is

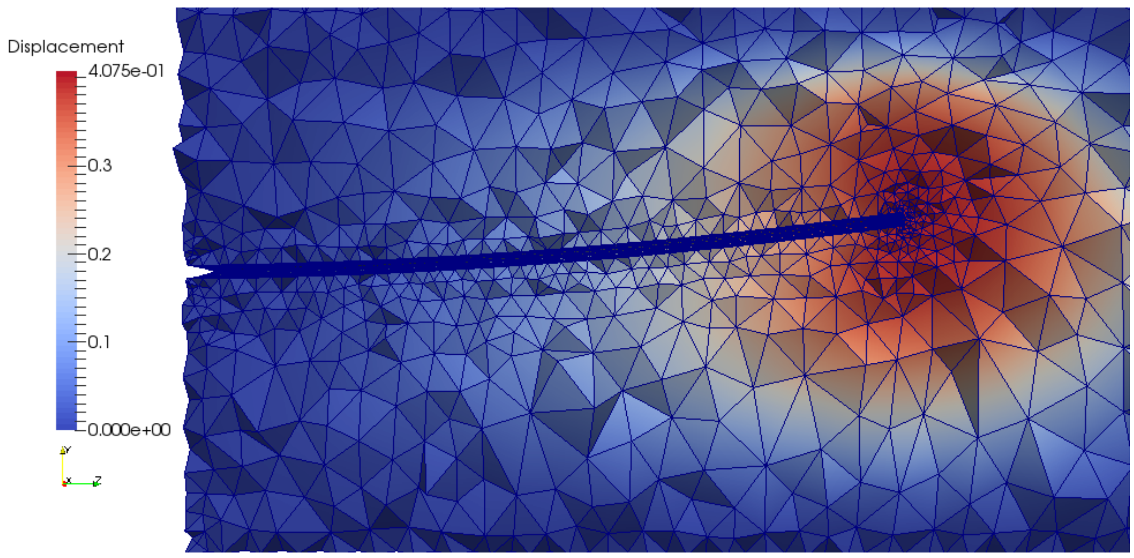


Figure 2.15: IDW computed fluid mesh deformation following the motion of the NACA0012-based wing.

clearly grown (36036 nodes) and consequently the CPU time (446 seconds), emphasizing the inadequacy of the *standard* IDW strategy for applications to *real-time* problems.

The discretization of a 3D shape and the generation of a 3D mesh grid require certainly an important computational effort with an evident effect on the number of variables that we want to limit.

2.2.5 A naval application

So far we have essentially presented typical aerodynamic configurations used for CFD study of wings. To corroborate the versatility of IDW technique, we present a test case related to a different field, the naval framework.

It is possible to apply IDW to fluid grid inherent to naval studies, as test tank applications or reproductions of hull conditions in the water. Under the assumption of ideal conditions, for the purpose of grid node displacement computation, it is not fundamental to know the physical properties of the fluid, independently by its nature (air or water). As a result of this consideration, we have operated the IDW displacement computation on the internal nodes of a flat top fluid grid around a hull with simple geometrical characteristics. The starting undeformed fluid grid and the deformed one are shown in the Figures 2.16 and 2.17, respectively.

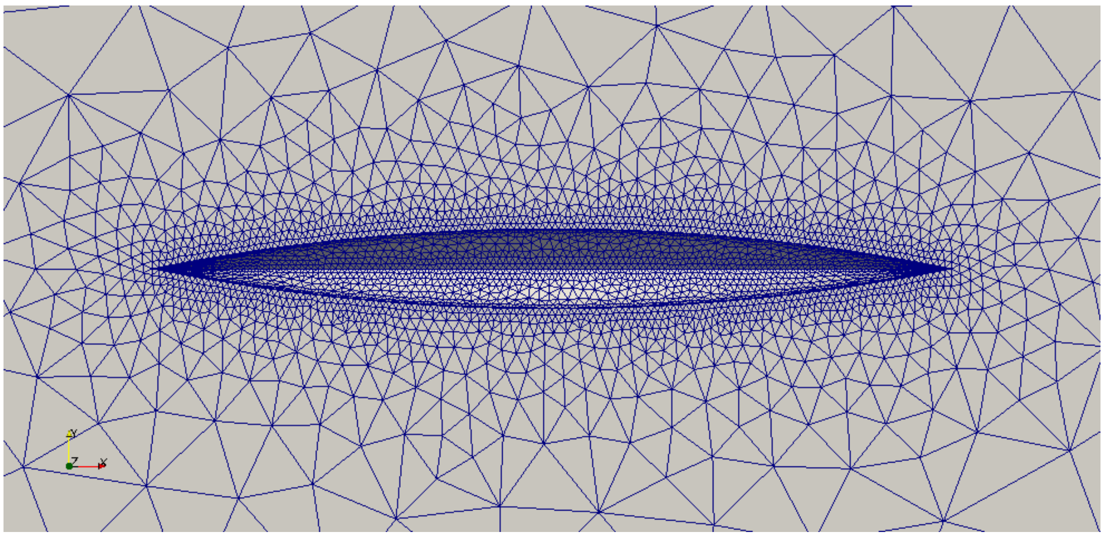


Figure 2.16: 3D undeformed fluid mesh around a hull.

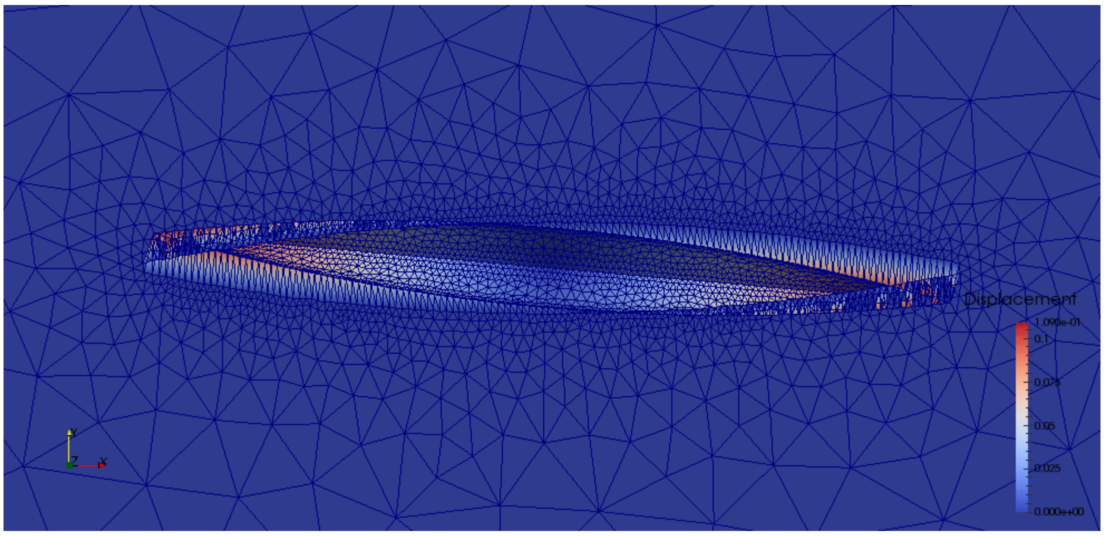


Figure 2.17: IDW computed 3D fluid mesh deformation following the 5^o rotation of the hull.

We have operated a rotation of the hull respect to the z axis of an angle equal to -5^o, thus obtaining the result in Figure 2.17 with the numerical results listed in Table 2.5.

Also for this 3D naval case, the results summarized in Table 2.5 denote an excessive CPU time (almost 20 seconds), which makes prohibitive the use of *standard* IDW technique for applications where a repeated computation of the solution (*real-time* and *shape optimization* problems, for example) is required.

Mesh dimensions	$50\text{ m} \times 25\text{ m} \times 10\text{ m}$
Hull length	2.50 m
Number of elements	30265
Number of nodes	7186
Number of control points (boundary nodes)	3864
Number of internal nodes	3322
CPU Time	19.955 s

Table 2.5: Quantitative information about the 3D fluid mesh showed in Figure 2.16.

Looking to the deformed three-dimensional mesh in Figure 2.17, we observe that just the boundary nodes belonging to the hull interface are displaced, producing a reduction of the mesh quality for the elements close to the top fluid boundary. The structural solver, in fact, will provide the displacement of nodes belonging only to the structure, ignoring the new position of nodes of the fluid boundaries neighboring (in this case the upper face of fluid mesh). Moreover, since the overlap is localized to the plane of the top face of the mesh, its visualization turn out to be difficult to understand by Figure 2.17 and impossible to extract by observing the phenomenon from different points of view.

This represents a result as unacceptable as difficult to correct analytically. In fact, it is very hard to know a priori the movement to impose to the boundary nodes belonging to the upper face of the fluid mesh.

From this consideration it follows the idea of decomposing the process of motion of the grid in three steps, computing primarily the displacement of the not moved boundary nodes as a function of the displaced ones in 1D and 2D (thus analyzing vertices, edges and faces) and then switching to the internal node displacement computation. Thus, all these steps require an IDW application, accordingly to an ad hoc algorithm that will be detailed in Section 3.2.

2.2.6 Structural mesh motion

In all the previous cases, we have applied the IDW interpolation technique to compute the displacement of the internal nodes of a fluid grid. However, nothing prevents to apply IDW to typical structural domains, always considering that no assumption is made about the physical properties of the material used to realize the structure.

Same results of those obtained for fluid grids, in fact, can be obtained for several shapes of structures: hulls, wing, etc.

For the sake of clarity, we report here the results obtained for a wing of the same shape of the one used in Figure 2.14, imposing to it a movement similar to the one applied in the previous case, via the function:

$$y(z) = 0.01(z)^2. \tag{2.8}$$

Figure 2.18 and 2.19 show the initially undeformed mesh and the corresponding deformed one.

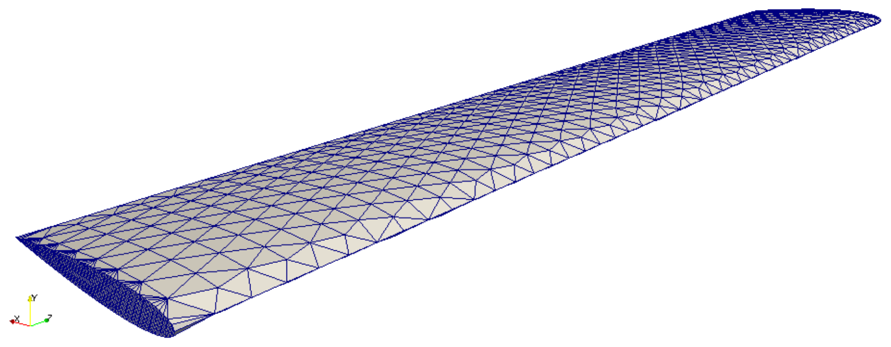


Figure 2.18: Undeformed mesh for a 3D wing based on NACA0012 profile.

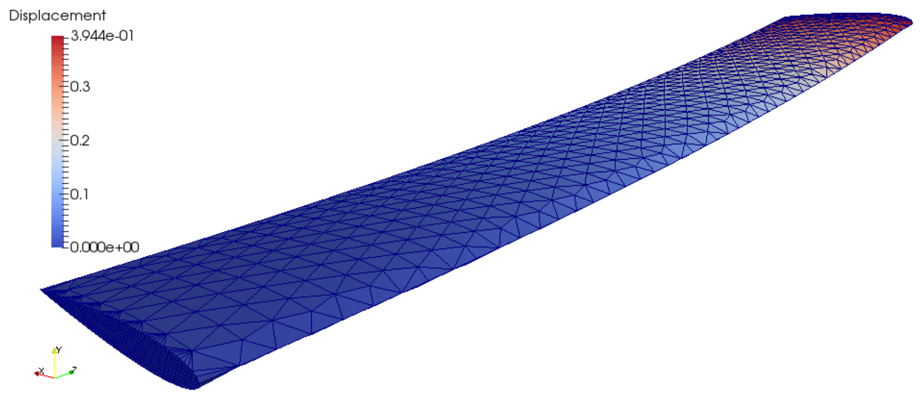


Figure 2.19: IDW computed mesh for a deformed 3D NACA0012-based wing.

2.3 IDW and possible extensions

Nowadays, a number of more advanced IDW formulations have been developed to improve on the limitations of the *standard* IDW interpolation. For example, spatially non-uniform p parameter distributions based on data density have been used in adaptive IDW [53] and through a Fourier integral representation [9]. Some applications propose also the using of *Genetic Algorithm* to optimize IDW in order to find the optimal parameter values for the IDW function [23]

We want to underline here that, since IDW is an *Extremum Conserving* (EC) interpolation method, an EC-IDW mesh deformation method can be constructed by applying IDW interpolation directly to the location of the boundary nodes and at internal mesh points. This is, however, only applicable to translations and not to rotations. For a more practical EC-IDW formulation, please refer to [21].

Another possibility to solve the problem is to separate the '*large*' translation and rotation contributions from the '*small*' deformations. It is important to remark here that the deformations classified as *large* and *small* represent the rigid and elastic components of the displacement, respectively, since the former involves a perturbation of the whole structural domain, while, the latter, generally leads to localized shape deformations.

This kind of approach derives from the idea that, to solve efficiently *mesh motion* problems, we should know the entity of both the two (rigid and deformative) displacement components. Indeed, the structure would be subject to both deformations and rigid movements due to the effect of the fluid forces acting on the structural external surfaces. Thus, next goal is to set up a methodology, in the next chapter, to identify both rigid translation and rotations, while, for the recovery of the deformative component, it will be employed the shape parametrization technique introduced here. In fact, the motion of the fluid mesh, when considering free-body movements, can be known a priori (i.e., consider the case of imposed oscillations of an airfoil, aileron deflections, etc.) or it can be the output returned by the structural sub-system. In the next chapter we will discuss this new approach by focusing on the possibility of knowing the rigid displacement of the fluid mesh by the output of the structural solver.

To summarize, we have studied the results of applying the IDW technique on several fluid and structural meshes for different boundaries and interface displacements. We have also underlined the effects on internal node displacements due to the selected value of p. We said that IDW strategy, as RBF, can use as control points **all the nodes lying on the interface and boundaries**, but we added that this kind of choice produces a very large computational cost in the extraction of the value of the interior node displacement.

We have highlighted the influence of the number of nodes on the CPU time and the memory storage by assembling the $[\mathcal{N}_s \times \mathcal{N}_c]$ $\boldsymbol{IDW}$ matrix. Thus, the goal is to obtain a strategic and reduced distribution of $\hat{\mathcal{N}_c}$ control points with $\hat{\mathcal{N}_c} \ll \mathcal{N}_c$. Finally we have noticed that the results obtained are not so satisfactory for CFD uses due to the long CPU time and the excessive memory effort. For this reason, we are going to present some important innovations and improvements in order to optimize the *standard* IDW interpolation technique and to extend its usage to CFD applications.

Chapter 3

Geometrical reduction for IDW

In this chapter we overcome some of the limitations that characterize the *standard* IDW strategy proposed in Chapter 2, within its applications to the mesh motion problems. In particular, the first issue to be faced is the choice of the control points, i.e., their number and location. As seen in Chapter 2, the distribution of the control points has a large impact on the accuracy and on the computational costs of the mesh motion procedure. We show how it is possible to standardize the procedure, leading to a reduction of the memory costs too. In order to improve the *standard* IDW approach, a wide range of strategies is presented in the literature [31, 77]. Basically, the goal in the FSI framework is to establish a procedure to automatically choose a reduced set of control points in order to represent both the initial shape configuration and its deformation during the fluid-structure interaction process. Moreover, another important issue to be taken into account when dealing with FSI problems, is related to the possibility of studying the multiphysics interaction between fluid and structure when the latter consists of a rigid body. In fact, during the simulation process the structure can be not only deformed, but, it can be subject also to rigid translations and/or rotations. Overall, to estimate the entity of such a movement can be a possible solution in order to treat correctly the mesh motion problem and to be able to update the fluid mesh according to the structure deformation and rigid behaviour.

3.1 RIDW: Reduced Inverse Distance Weighting

In the previous chapter we have observed that the number of control points plays a very important role in the mesh motion computation via IDW. In fact, as it is evident in the applications previously considered, the CPU time increases with the number of control points. However, since all boundary nodes are selected as control points, the user has not chance to act on the method in order to decrease the computational cost. Thus, in this section we propose a reduced version of the IDW technique to automate the selection of the most relevant control points. We exploit, here, the simple example of the square domain in Section 2.2.1, observing how the CPU time changes with an increasing number of nodes, namely of control points, that coincide with the boundary nodes. Obviously, diminishing the number of grid nodes, the shape representation may suffer of a not optimal discretization producing a possibly significant loss of information about movement and displacement, depending on the required accuracy. Assuming to use a mesh with 5 elements along both the directions (see Figure 3.1), for a sinusoidal motion of the upper boundary as in 2.2.1, the resulting CPU time demanded by the *standard* IDW is equal to 0.0624 s.

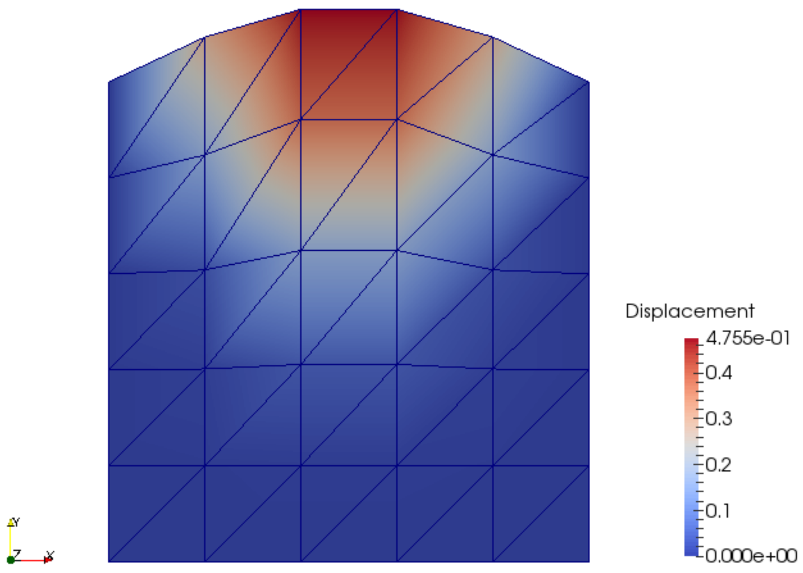


Figure 3.1: Square with upper boundary deformed dicretized by 5 elements for side.

Using the same square but with 10 (Figure 3.2) and 15 (Figure 3.3) elements for each side, the CPU time increases to 0.0814 s and 0.09723 s, respectively.

The most expensive part of the IDW technique, stringly related to the number of control points, is represented by the memory allocation for the $\boldsymbol{IDW}$ matrix. It is a $[\mathcal{N}_c \times \mathcal{N}_s]$ matrix that can be of very large dimensions when $\mathcal{N}_c$ and $\mathcal{N}_s$ increase considerably.

In agreement with the previous considerations, to cut computational and, in particular, memory costs, it is possible to set a strategy in order to reduce the number of the control points. However, a particular attention in the selection of these points

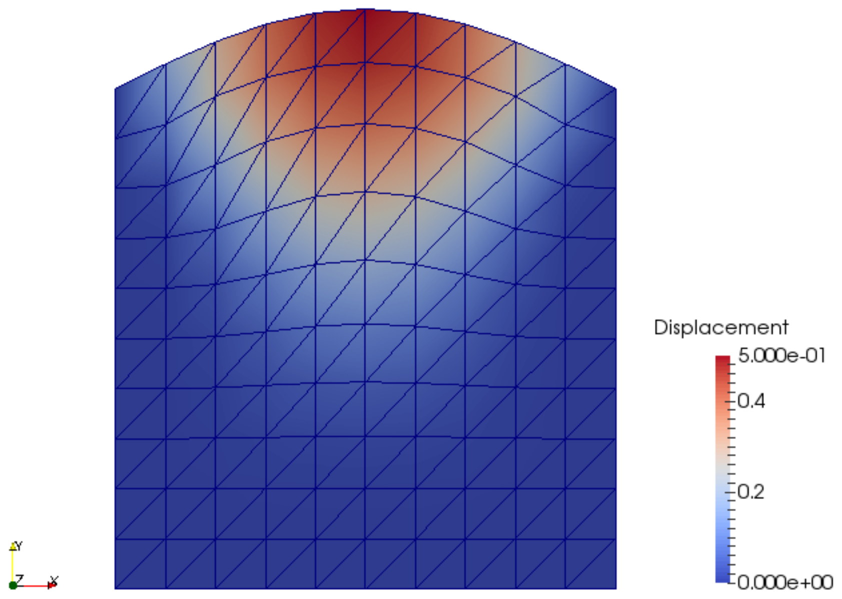


Figure 3.2: Square with upper boundary deformed dicretized by 10 elements for side.

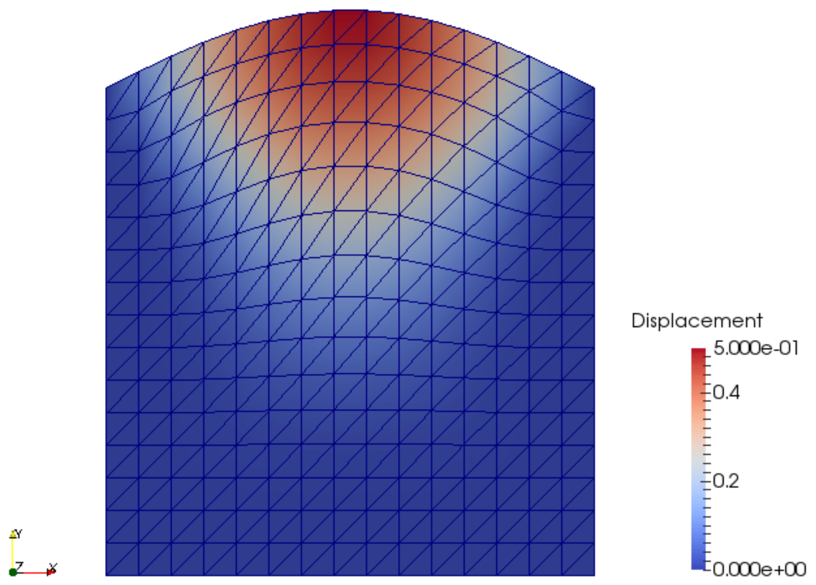


Figure 3.3: Square with upper boundary deformed dicretized by 15 elements for side.

is required to avoid an important loss of information which can produce unacceptable computational errors.

To reduce the number of the control points, we have implemented an ad hoc algorithm which implements the reduction procedure proposed by Shepard [77], and used in some subsequent works (e.g., [31]). In particular, we build a subset $\hat{\mathcal{I}_c}$ of control points, such that $\hat{\mathcal{I}_c} \subset \mathcal{I}_c$, where $\mathcal{I}_c$ denotes the original set of all the control points.

Each control point $\hat{\mathbf{x}_k} \in \hat{\mathcal{I}_c}$ satisfies the following relation:

$$\parallel \mathbf{x} - \hat{\mathbf{x}} \parallel \leq R \tag{3.1}$$

where R is the user-defined radius of a sphere of influence adopted to improve the performances of the standard method, and $\mathbf{x}$ represents the position of each point considered.

However, using this relation in case of large displacements for some control points, it is possible to overweight the displacement of the closest control points and neglect the information about the boundary limits, up to locating the internal nodes over the boundary ones. This means to violate the no-compenetration constraint, with the consequence of producing an unacceptable mesh motion similar to the one observed in the previous chapter for low values of *p* (see Figure 2.3).

Thus, to preserve the boundary information, we have decided to retain each control point movement while using just some of them for the internal mesh displacement.

The technique presented here, which we name **Reduced Inverse Distance Weighting** (RIDW), selects the control points to be considered for the internal displacement computation with the idea of reducing their number while preserving a uniform distribution of them on the grid. For this reason, we have revisited the idea proposed by Shepard by employing the same relation and by using as reference the distance between two neighboring control points. Thus, we evaluate each distance between the $\mathcal{N}_c$ control points and, starting from a randomly chosen control point, we erase from the global set of control points the ones that lie inside a circle of radius R around the selected one, in order to identify the subset $\hat{\mathcal{I}_c} \subset \mathcal{I}_c$.

The algorithm consists of the following two phases: *initialisation* and *selection*. Given a user defined radius R and an empty set $\hat{\mathcal{I}_c}$, the algorithm can be summarized via the following steps.

RIDW algorithm

Initialisation phase

1) Choose randomly the node $P \in \mathcal{I}_c$; $\hat{\mathcal{I}_c} \to \hat{\mathcal{I}_c} \cup \{P\}$;

2) Erase all the points $\tilde{P} \in \mathcal{I}_c$ belonging to the circle of radius R centered at P;

3) Define n sets α_i $(i = 1, \cdots, n)$ which collect the nodes belonging to concentric annuli of thickness $0.8R$, always centered at P;

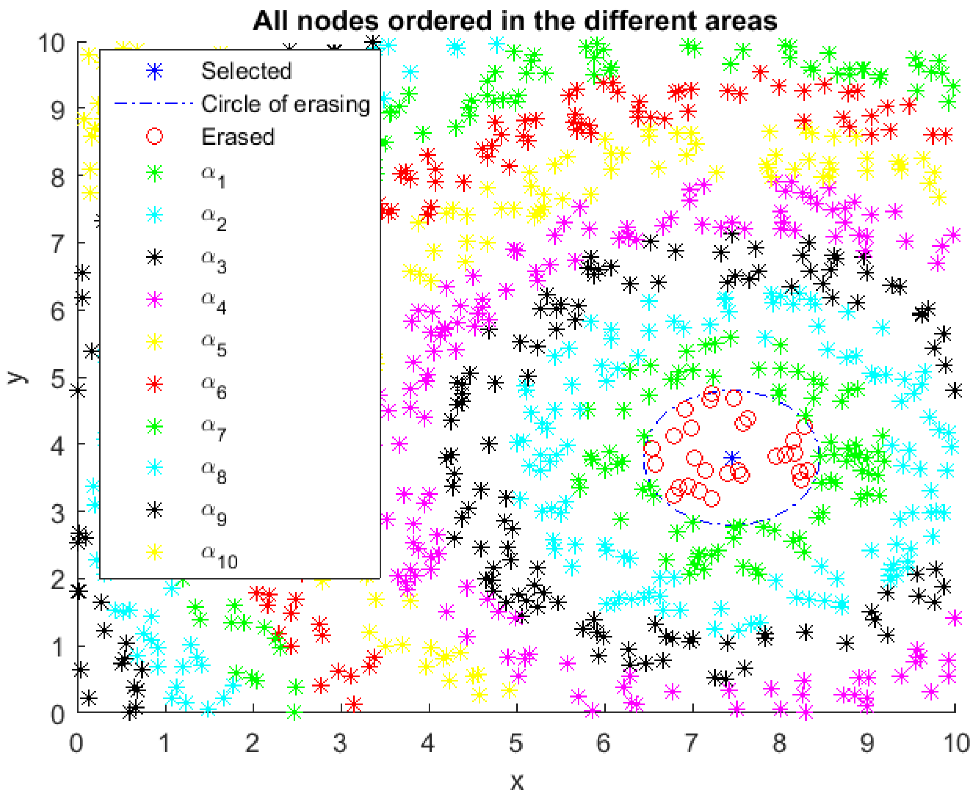


Figure 3.4: Visualization of the first phase of RIDW algorithm for 1000 points randomly distributed on the square $(0, 10)^2$.

Figure 3.4 provides an example of this first phase for $n = 10$.

Selection phase

```
1) Define the sets of nodes $\beta = \alpha_1$ and $\gamma = \emptyset$;
2) Initialise $k = 1$;
3) do
   {
        do
        {
          ∘ $\beta$ = {all points $\bar{P}$ belonging to $\alpha_k$ and to the annulus centered in $P$
            with internal radius $R_i = R$ and external radius $R_e = 1.3R$};
        ∘ $\gamma = \alpha_k \setminus \beta$;
        ∘ Choose $P \in \beta$ and $\hat{\mathcal{I}}_c \to \hat{\mathcal{I}}_c \cup \{P\}$;
        ∘ $\beta \to \beta \setminus \{$all points $\tilde{P}$ belonging to $\beta$ and to the circle of radius R
          centered in $P\}$;
        ∘ $\gamma \to \gamma \setminus \{$all points $\tilde{P}$ belonging to $\gamma$ and to the circle of radius R
          centered in $P\}$;
        ∘ $\alpha_k \to \alpha_k \setminus \{$all points $\tilde{P}$ belonging to $\alpha_k$ and to the circle of radius R
          centered in $P\}$;
        ∘ if exists $\alpha_{k+1} \to \alpha_{k+1} \setminus \{$all points $\tilde{P}$ belonging to $\alpha_{k+1}$ and to the circle
          of radius R centered in $P\}$;
        ∘ if exists $\alpha_{k+2} \to \alpha_{k+2} \setminus \{$all points $\tilde{P}$ belonging to $\alpha_{k+2}$ and to the circle
          of radius R centered in $P\}$;
      $\}$ while ($\beta \neq \emptyset \wedge \gamma \neq \emptyset$);
      $k = k + 1$;
  $\}$ while ($k \neq i + 1$);
```

In the Figures 3.5, 3.6 and 3.7 we provide a visualization of the first steps of the the RIDW second phase.

The procedure described above allows to loop just on the grid nodes that play the role of control points and which lie on some, not all, external annuli at every step. In fact, depending on the specific radius R, we can take into account only the annuli erasable control points belong to. In this way we reduce the computational effort required for a process that has to loop on all the remaining points, at every reduction step. Continuing in the selection process, we jump, step by step, around and gradually farther from the first data point selected. This method produces a control point distribution quite uniform, conserving the most of information about the motion of data points and reducing considerably their number.

For the sake of completeness, we show in Figures 3.8 and 3.9 the starting distribution of points the RIDW process is applied to and the resulting reduced set, respectively. The starting configuration is characterized by 1000 randomly distributed points on the square $(0, 10)^2$.

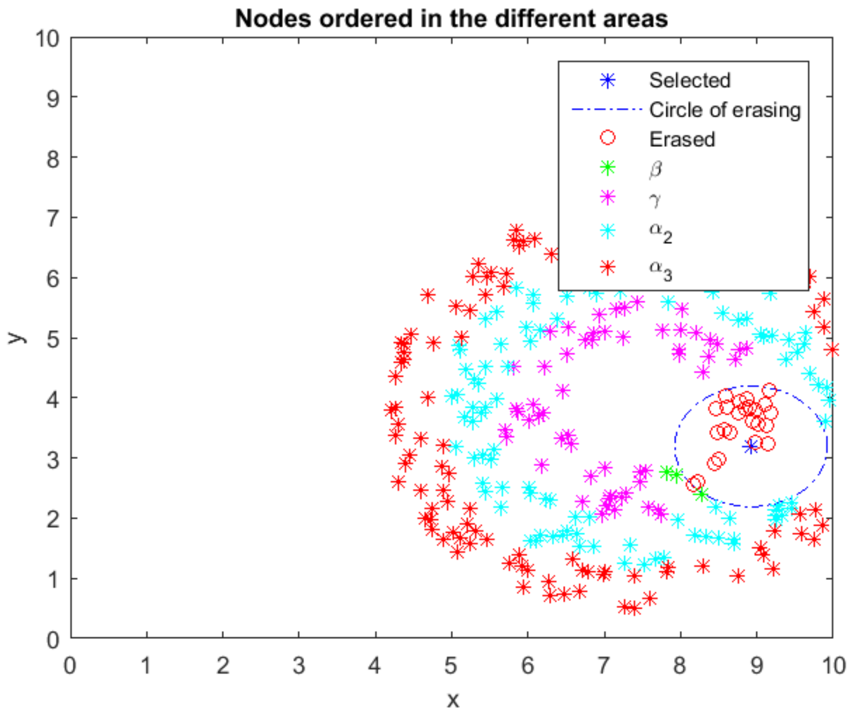


Figure 3.5: Second control point selected and its areas of erasing and reordering.

Figure 3.9 shows a considerable reduced distribution of control points almost uniform, clearly depending on the radius R selected. More interesting configurations will be provided for more complex applications in the following sections.

We underline that the deformation of the data points is totally conserved. Furthermore, at the end of the reduction process the new **RIDW matrix** has only $\hat{\mathcal{N}}_c$ columns, with $\hat{\mathcal{N}}_c$ equal to the resulting number of control points.

The most important role for the application of the RIDW algorithm is played by the **radius** R.

Moreover, it is important to consider that the discretization of each part of the boundary can be generally different from the others. As a consequence, it is advisable a diversification of the radius value along boundary so that, a priori, the user has to select as many values of R as the number of parts the boundary is divided in. For example, a cube is composed by a boundary divided into six parts (the six faces). Then it is necessary to select six values for R to execute the reduction process.

Certainly, the choice of these values not can be random. Depending on the accuracy required for the internal node displacement, the user can properly choose the values for R. The higher the value of R, the larger the expected computational error on the internal node displacement. In fact, when the value of the radius increases, the circle of erasing

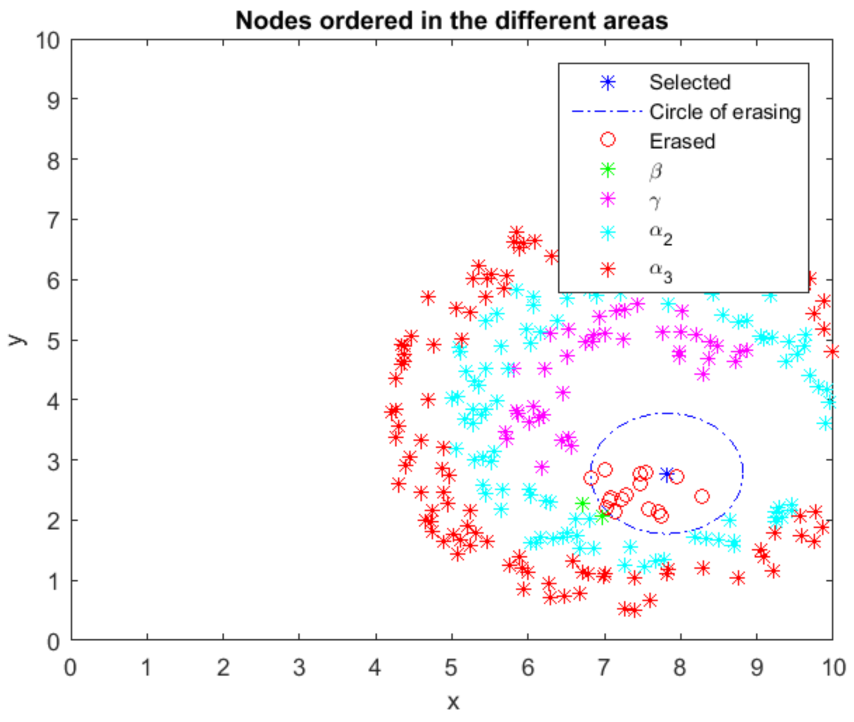


Figure 3.6: Third control point selected and its areas of erasing and reordering.

becomes larger. Thus, less and less control points are taken into account for the internal node displacements with the consequence of a discrete increasing of the computational error.

Obviously, the RIDW technique can be applied to meshes of any spatial dimension. Following this consideration, we will present at the end of this chapter several applications for different spatial dimensions, as we have done in the previous chapter for the *standard* IDW approach.

3.2 An IDW application in dimensional cascade

As anticipated in Chapter 2 (see Figure 2.17), it is very difficult to solve mesh motion problems where only the displacement of the fluid-structure interface is assigned and the displacement of the other boundaries of fluid mesh is unknown. With the previous implementation of the IDW technique it is possible to impose a displacement to the boundary nodes as a function of the interface motion. Nevertheless, if there is a structural object immersed in the fluid domain and the structure solver provides us the new position only of the interface nodes, it becomes our prerogative to move the other

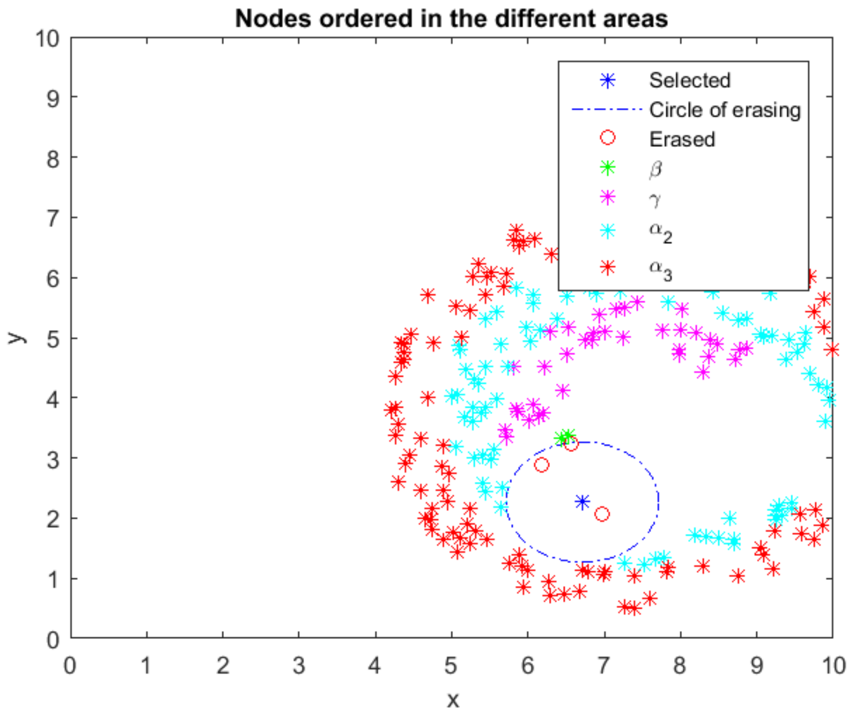


Figure 3.7: Fourth control point selected and its areas of erasing and reordering.

fluid mesh boundary nodes according to the movement of the interface. However, it is not always simple to obtain an analytical function that describes the movement of the other boundary nodes.

From this difficulty, we have the necessity to compute the overall displacement of the fluid mesh boundary nodes, following the movement of the structure immersed in the fluid domain. We have found a solution to this issue by the implementation of the **Dimensional Cascade IDW** (**DC-IDW**).

The idea is to use IDW interpolation technique to compute at first the movement of the boundary nodes of the fluid mesh which do not belong to the interface in function of the nodes of the interface. Then, the process can move to the computation of the internal node displacement depending on the position of all the boundary nodes that, now, are correctly moved.

For example, let us suppose to move several nodes belonging to the external edges and faces of the mesh as in Figure 3.10.

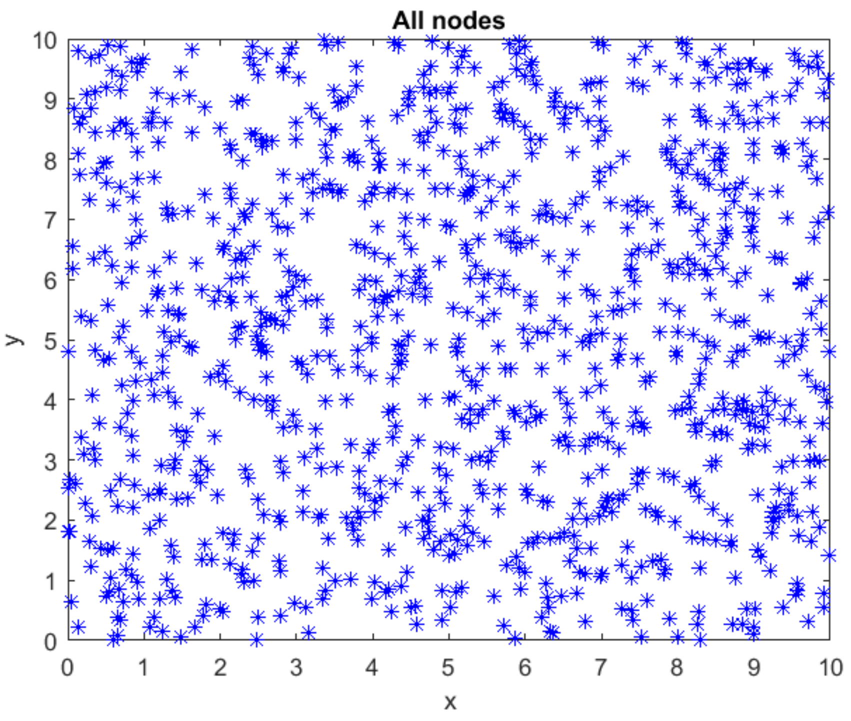


Figure 3.8: Starting random distribution of 1000 points on the square $(0, 10)^2$

The DC-IDW strategy is divided in to three phases.

- **First phase**: *1D analysis of the evolution of each boundary **edge**.*

  In this first phase the DC-IDW process computes the displacement of the boundary nodes belonging to each edge via IDW technique. Vertices and the originally moved nodes belonging to each edge play the role of control points, while the other nodes lying on the selected edge are the 'internal' (*slave*) nodes.

  In Figure 3.11 the configuration obtained by IDW application to each edge is shown.

- **Second phase**: *2D analysis of the evolution of each boundary **face**.*

  Starting from the configuration obtained after the first phase of DC-IDW procedure, it is possible to move to the update of the position of the nodes belonging to boundary faces. In this case, all the nodes lying on the edges of the considered face and the originally moved nodes belonging to the same

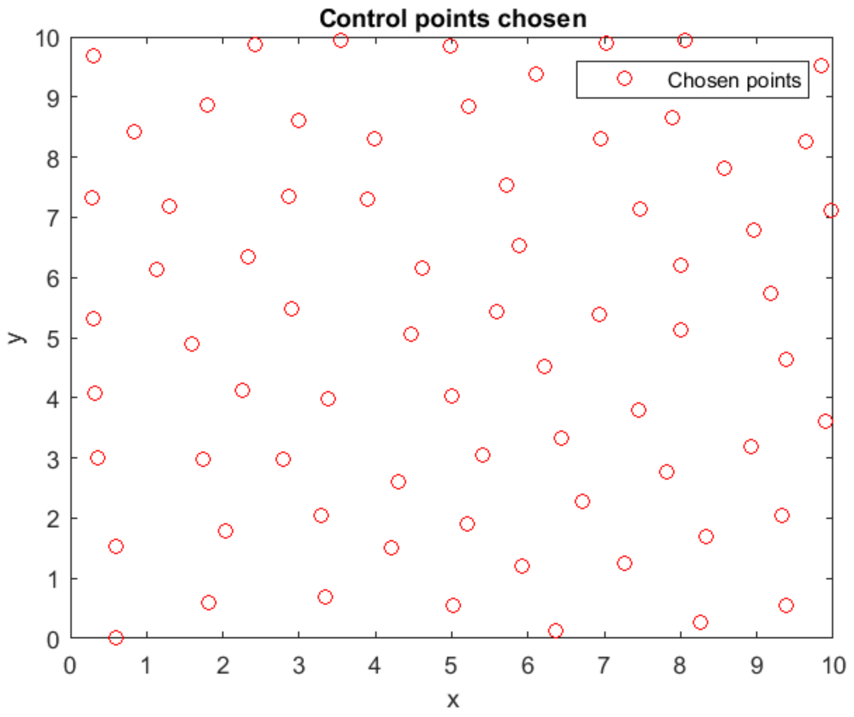


Figure 3.9: Distribution of the control points selected by RIDW process

face are selected as control points, while the other nodes of the selected face play the role of slave nodes.

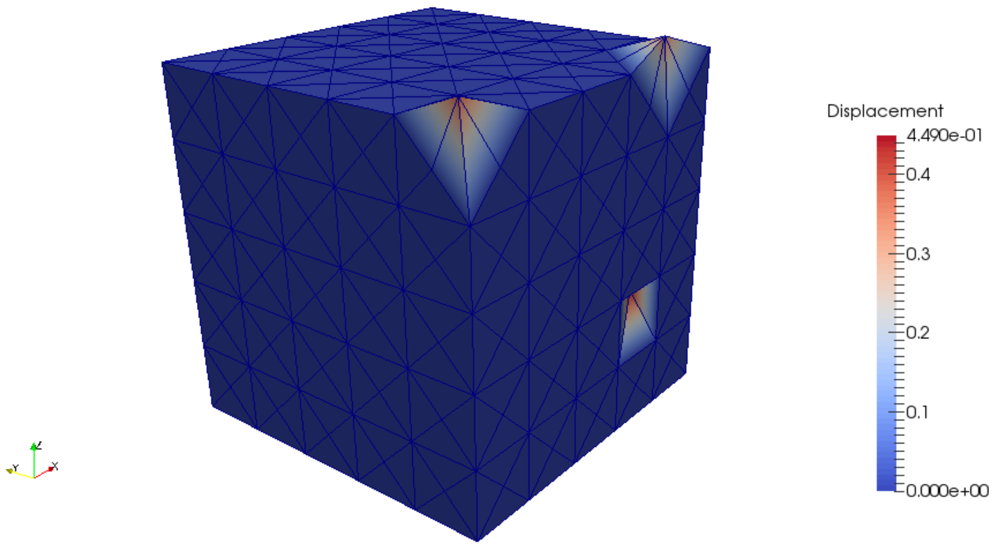
At the end of this second phase, the position of each node on the boundary faces is updated in agreement with the displacement initially imposed, as shown in Figure 3.12.

- **Third phase**: *3D displacement computation of the* ***internal*** *nodes of the mesh.*

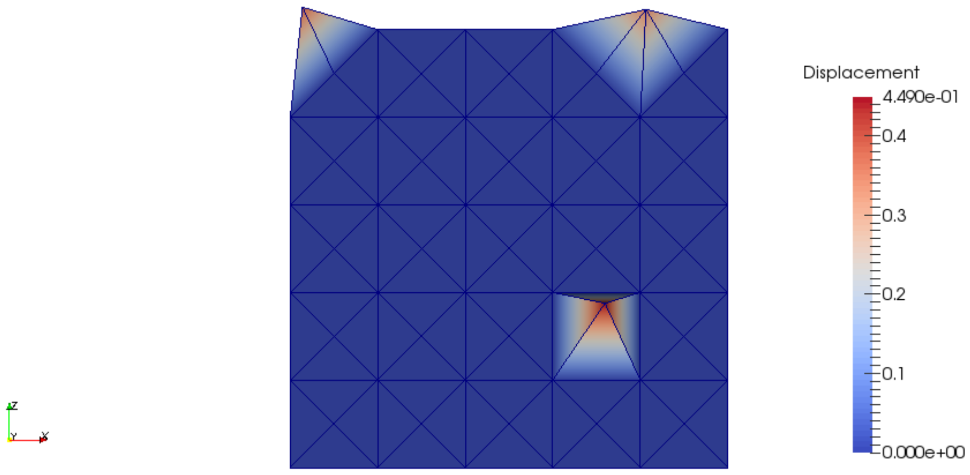
  All the boundary nodes of the mesh are correctly displaced now. In this last DC-IDW phase the *standard* IDW technique, or eventually the RIDW formulation, can be applied. The control points are all the boundary nodes or possibly the RIDW selected ones, while all the internal nodes of the mesh play the role of the slave nodes.

  An internal view of the resulting configuration is represented in Figure 3.13

For the sake of completeness, we show the result obtained with the DC-IDW for the challenging case of the hull described in the previous chapter. The deformed configura-



(a) Three-dimensional view

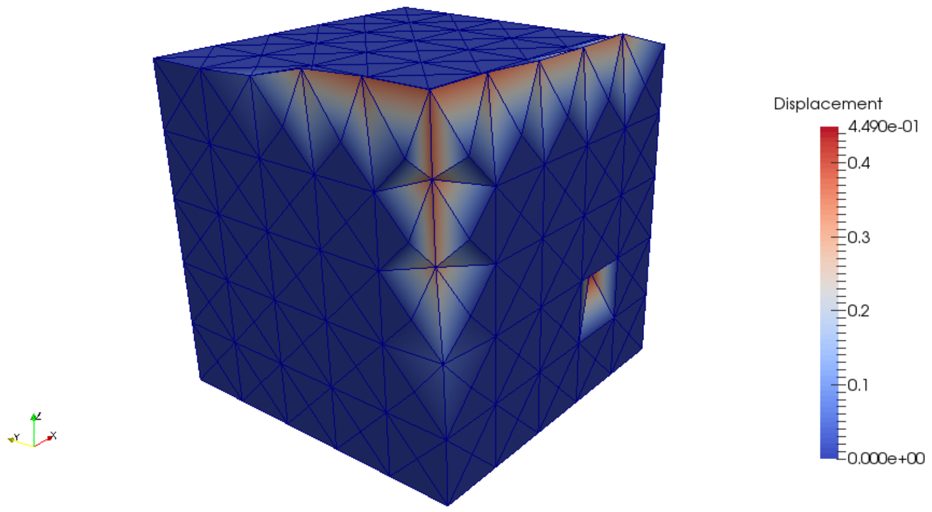


(b) Orthogonal projection

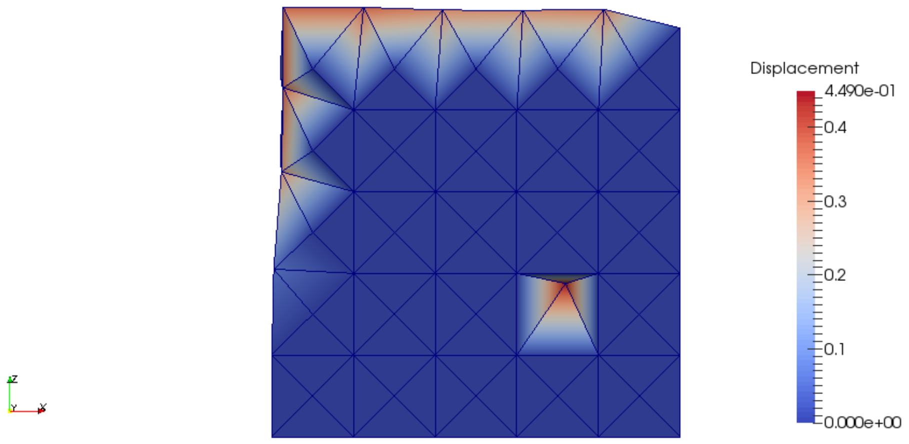
Figure 3.10: The starting configuration in which the only boundary nodes displaced are the ones selected by the user.

tion obtained with a *standard* implementation of IDW is provided in Figure 2.17, while the new configuration obtained via DC-IDW is shown in Figure 3.14.

Now, all the boundary nodes of the upper face of the fluid mesh have been moved following the motion of the hull walls.



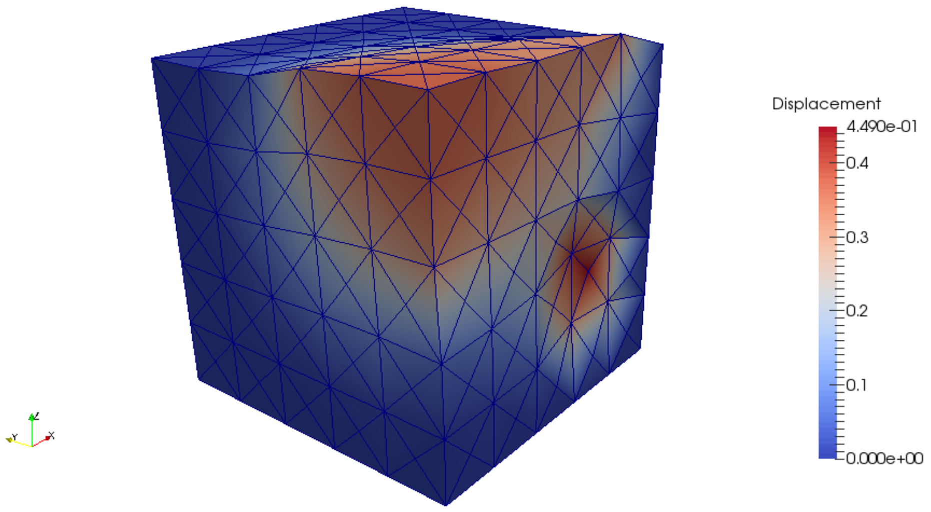
(a) Three-dimensional view



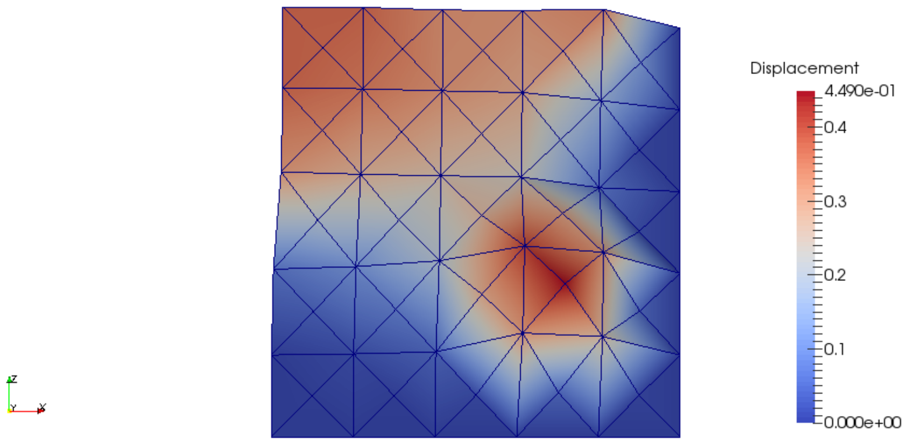
(b) Orthogonal projection

Figure 3.11: The configuration obtained after the first phase of the DC-IDW application.

DC-IDW process allows also the user to impose a given displacement to any node belonging to the boundary mesh and to compute the resulting configuration not only for the internal nodes, but also for all the boundary nodes. In this way the scope can be considerably extended and the IDW technique may become an important resource for several kinds of studies.



(a) Three-dimensional view



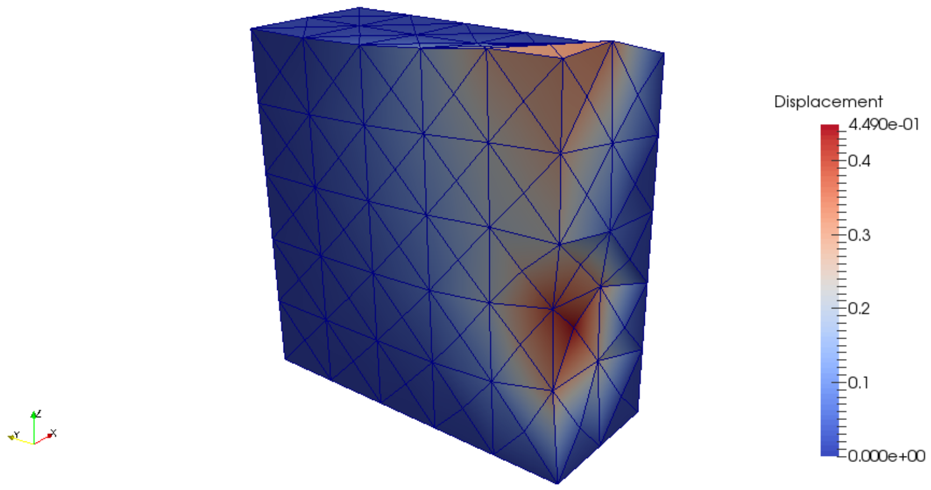
(b) Orthogonal projection

Figure 3.12: The updated boundary nodes position after the second phase of the DC-IDW application.

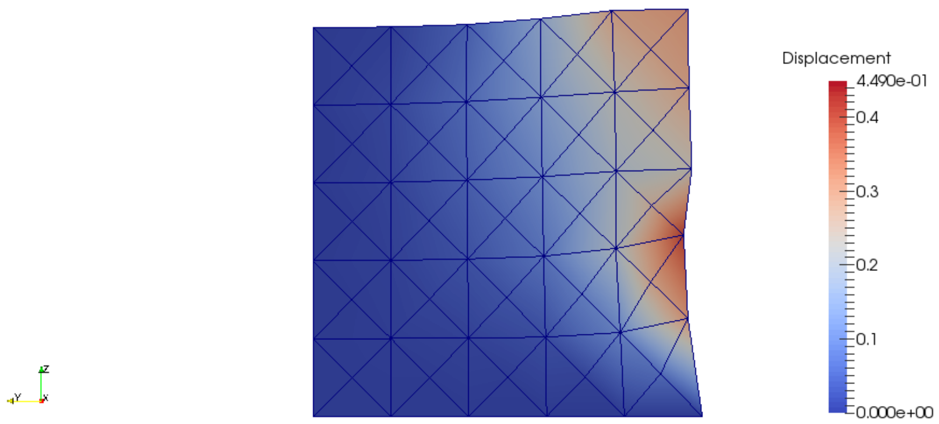
3.3 RIDW: applications and comparisons

In this last section, we repeat some of the applications presented in the previous chapter applying the RIDW technique, to analyse the advantages and the drawbacks of the new procedure.

Notice that, in all the following applications, just in the *last phase* of the DC-IDW strategy, in addition to the control points selected by RIDW, we will consider **all** the



(a) Three-dimensional view



(b) Orthogonal projection

Figure 3.13: Internal view of the mesh nodes displaced at the end of the DC-IDW application.

vertices and, in particular, for three-dimensional applications we add to the reduced set of control points, all the nodes belonging to the *edges* of the boundary. The reduced distribution is preserved on the faces.

We have done this choice to preserve the information about the deformation limits imposed by the boundary node position, to observe the *no-compenetration constraint*.

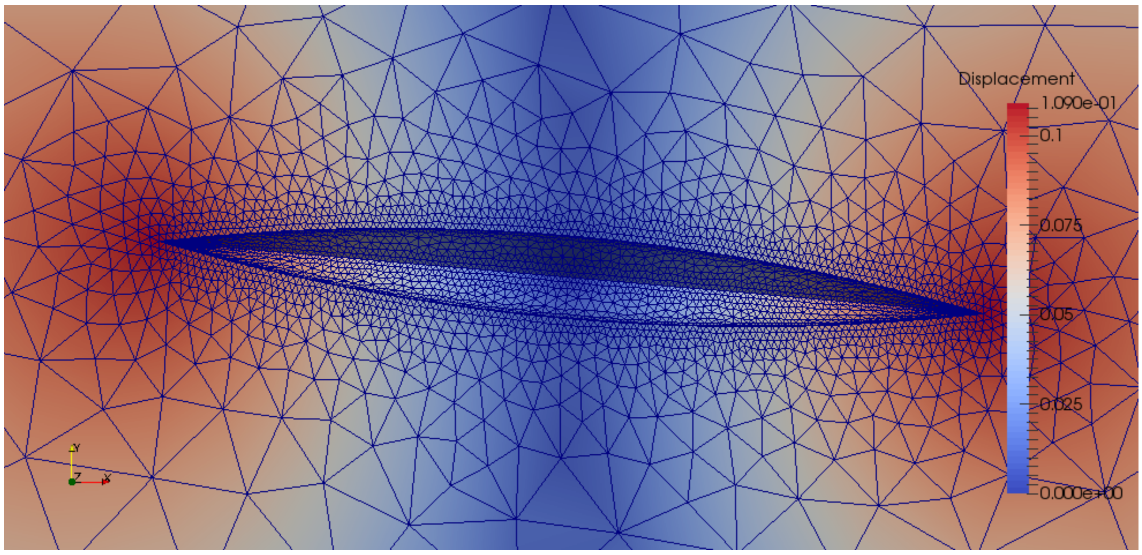


Figure 3.14: Correct boundary nodes motion around a rotating hull.

3.3.1 The RIDW square test case

To begin, we consider the example in Section 2.2.1.

A visualization of the starting configuration and the deformed resulting one in Figures 3.15 and 3.16. We have included in the undeformed configuration the control points selected by RIDW algorithm, via small red spheres.

While respect to the mesh shown in Section 2.2.1, now the control points do not coincide with all the boundary nodes but just with a subset of them. We can summarize the quantitative information of the RIDW procedure in Table 3.1.

We can observe a sensible increase of the CPU time, from 0.097 s to 0.126 s, due to the time spent for the reduction procedure.

Nevertheless, we aim at reducing the memory effort demanded by the IDW technique, properly resizing the IDW matrix. In the specific case, the number of control points has dropped drastically from 60 (all the boundary nodes for the *standard* IDW technique) to 24. Consequently, the matrix will be characterized by the same number of rows (the number of internal points has not changed), but by only 24 columns instead of 60, with a considerable saving of memory.

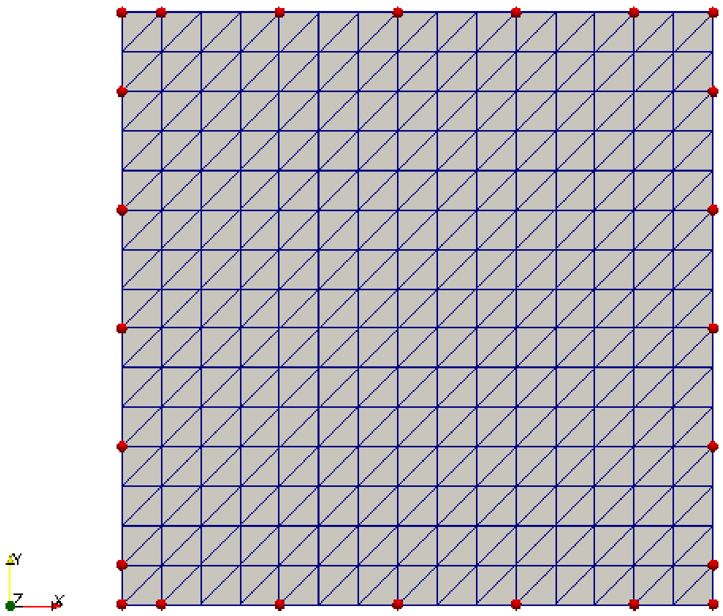


Figure 3.15: Undeformed configuration of the square test case with the control points (red spheres) selected by RIDW process.

Lenght of square side	π m
Number of total elements	450
Number of total nodes	256
Number of control points	24
Radius of reduction	$R_{top} = 0.5$ m
	$R_{bottom} = 0.5$ m
	$R_{right} = 0.5$ m
	$R_{left} = 0.5$ m
Number of internal nodes	196
Number of boundary nodes	60
CPU Time	0.126 s
Relative error respect to the displacement	1.85 %

Table 3.1: Quantitative information related to the RIDW deformation of the square in Figure 3.15.

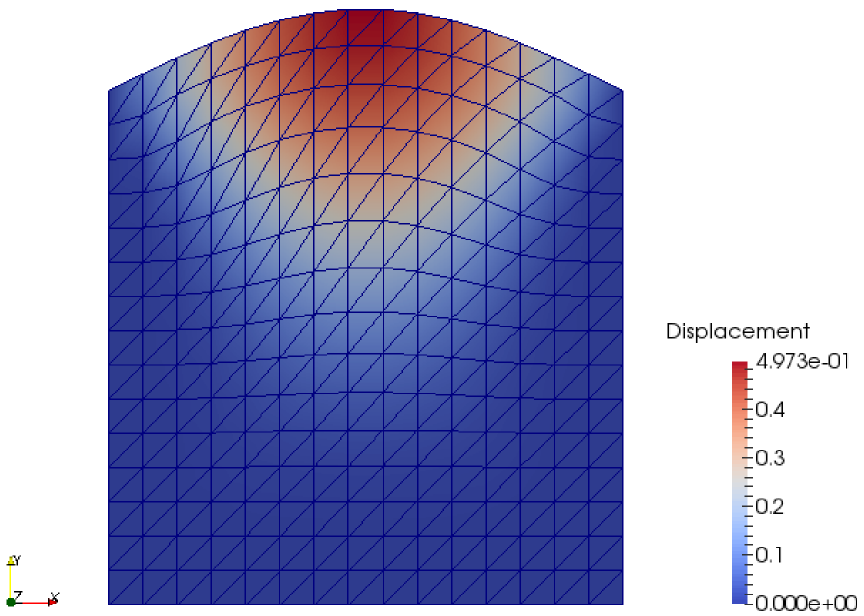


Figure 3.16: Deformed square with internal nodes displacement computed via RIDW.

As expected, the reduction procedure introduces an error on the internal node displacement with respect to the *standard* IDW result, expressed as

$$e = \frac{\sum_{s=1}^{\mathcal{N}_s} (\| \mathbf{u}^{IDW}(\mathbf{x}_s) \| - \| \mathbf{u}^{RIDW}(\mathbf{x}_s) \|)}{\sum_{t=1}^{\mathcal{N}_s} \| \mathbf{u}^{IDW}(\mathbf{x}_t) \|} \tag{3.2}$$

which, however, remains limited (about 1.85 %).

3.3.2 The RIDW cantilever beam test case

Another interesting application of RIDW is the bidimensional grid around the rectangular hole of Section 2.2.3. The starting configuration is the same as in Section 2.2.3, for the *standard* IDW. Nevertheless, in this section, the value of the internal displacements are computed according to the RIDW technique to reduce the number of control points.

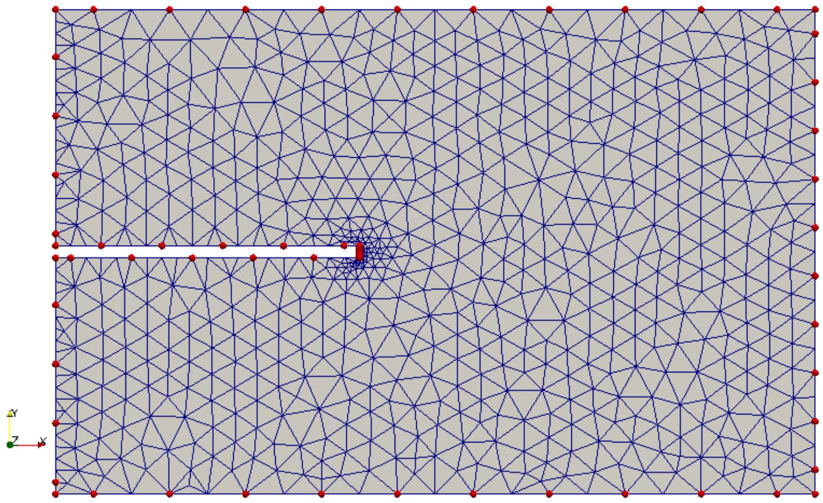


Figure 3.17: Undeformed starting mesh and control points selected by RIDW process for a deformable cantilever beam immersed in a fluid domain.

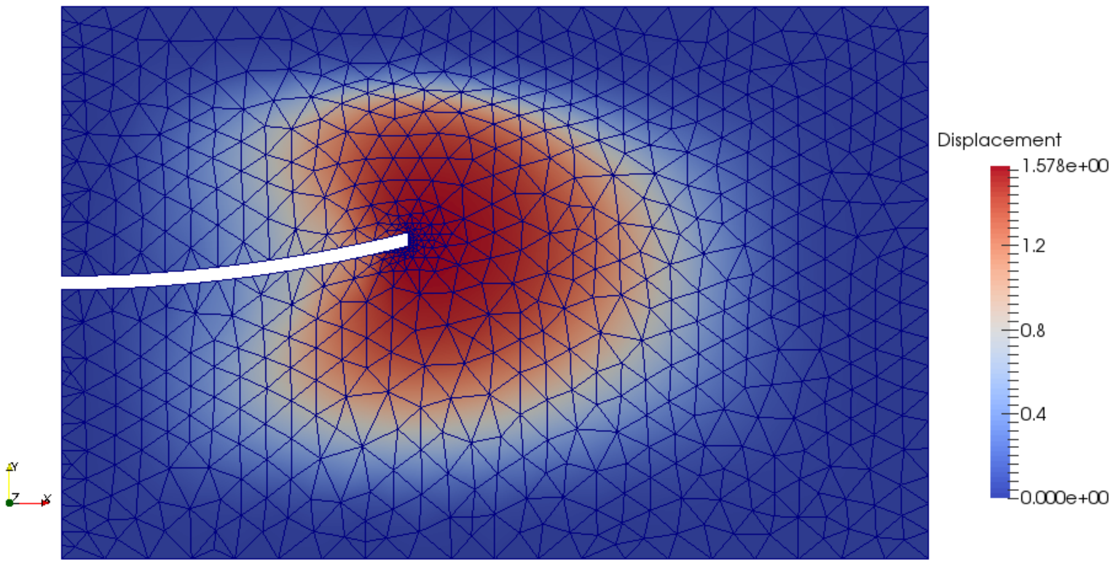


Figure 3.18: Quantitative information related to the RIDW obtained fluid mesh motion around the cantilever beam in Figure 3.17.

From the quantitative results collected in Table 3.2 and from Figures 3.17 and 3.18, we remark that also for this application the CPU time is greater than the one required by the *standard* IDW. In fact, the CPU time is increased from 0.255 s to 0.280 s. However,

Mesh dimensions	6π m $\times$ 16 m	
Rectangular hole dimensions	2π m $\times$ 1 m	
Number of total elements	1360	
Number of total nodes	761	
Number of control points	66	
Radius of reduction	$R_{top} = 2$ m	
	$R_{bottom} = 2$ m	
	$R_{right} = 2$ m	
	$R_{upleft} = 2$ m	
	$R_{downleft} = 2$ m	
	$R_{tophole} = 2$ m	
	$R_{bottomhole} = 2$ m	
	$R_{righthole} = 0.03$ m	
Number of internal nodes	601	
Number of boundary nodes	160	
CPU Time	0.280 s	
Relative error respect to the displacement	4.60 %	

Table 3.2: Quantitative information related to the fluid mesh motion around the cantilever beam in Figure 3.17 computed via RIDW.

the reduction of control points, from 160 (all boundary nodes) to 66 points, implies an IDW matrix of size [601×66], fairly smaller with respect to the one computed via standard IDW ([601×160]). Operating the reduction on the number of control points, it has been produced an error equal to 4.60 % compared with the result obtained with standard IDW technique.

3.3.3 Fluid mesh motion around a wing via RIDW strategy

We focus now on the three-dimensional configuration presented in Section 2.2.4.

The RIDW applicated to a three dimensional case (see Figures 3.19 and 3.20) become more interesting because it better validates the advantages of the RIDW in terms of complexity reduction, also for cases closer to actual FSI and CFD studies. The feature which emphasizes the RIDW advantages is represented by the reduced number of control points. In fact, as shown in Table 3.3, the number of the data points diminishes from 14126 to 3611, with a considerable memory saving in assembling the IDW matrix, which now assumes a dimension of [21910×3611].

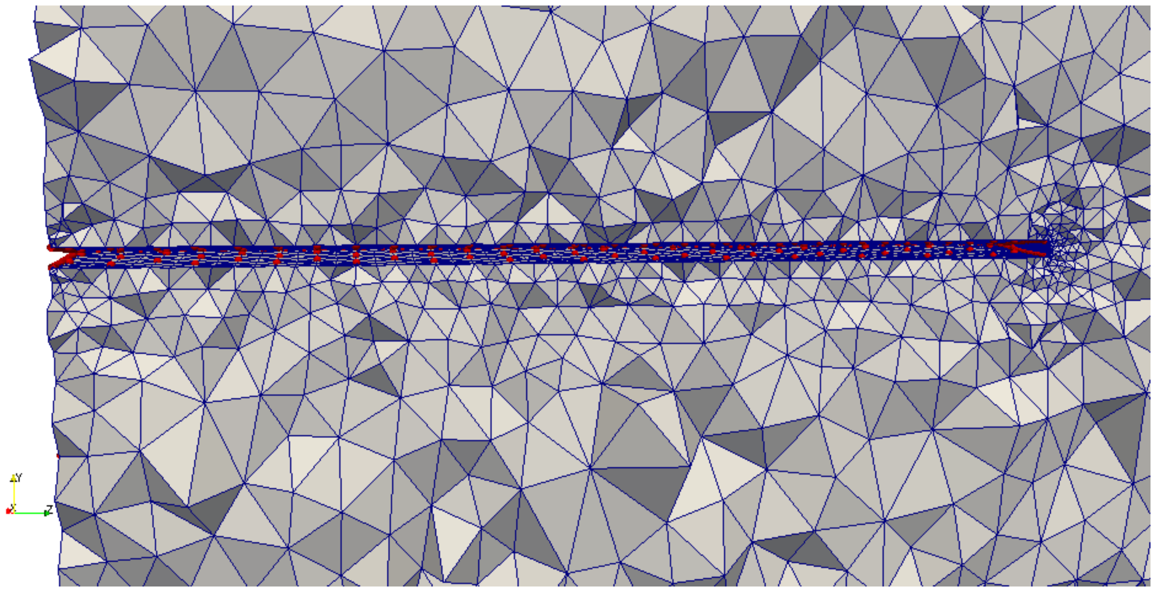


Figure 3.19: Undeformed configuration internal view of the mesh representing a wing in a wind tunnel with the control nodes (red spheres) selected by RIDW process.

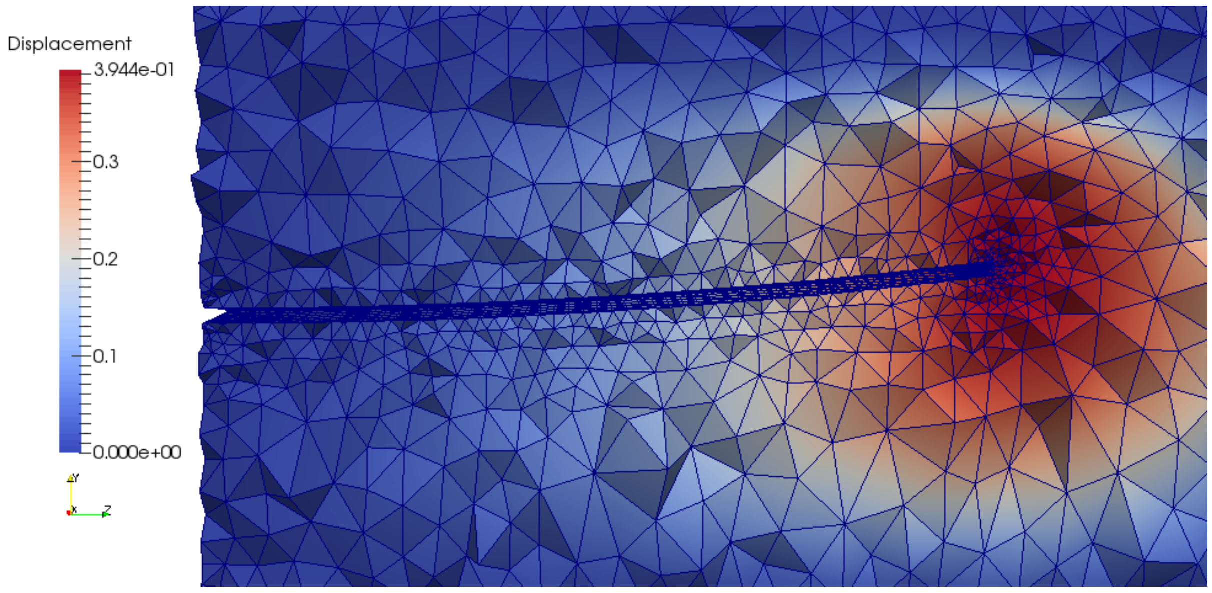


Figure 3.20: Mesh deformation internal view obtained at the end of RIDW application for a shape representing a deformed wing in a wind tunnel.

Mesh dimensions	10 m $\times$ 5 m $\times$ 4π m
Wing longitudinal dimension	2π m
NACA0012 profile chord length	1.01 m
Number of elements	169598
Number of nodes	36036
Number of control points	3611
Radius of reduction	$R_{top} = 0.5$ m
	$R_{bottom} = 0.5$ m
	$R_{right} = 0.25$ m
	$R_{left} = 0.25$ m
	$R_{front} = 0.5$ m
	$R_{rear} = 0.5$ m
	$R_{tophole} = 0.25$ m
	$R_{bottomhole} = 0.25$ m
	$R_{righthole} = 0.25$ m
Number of internal nodes	21910
Number of boundary nodes	14126
CPU Time	151.392 s
Relative error respect to the displacement	5.64 %

Table 3.3: Quantitative information about the RIDW deformed 3D fluid mesh represented in Figure 3.20.

In this case the reduction process has produced also a decrease of the CPU time, from 446.479 s required by the *standard* IDW process to 151.392 s for the reduced approach. This balances the time spent for the reduction procedure. However, the results obtained are affected by a relative error on the displacement of the internal nodes equal to 5.64 $\%$.

3.3.4 Mesh motion around a hull by RIDW technique

To investigate the potentialities of RIDW also in the naval field, we have to come back to the hull example presented in Section 2.2.5.

The trend of the resulting values, summarized in Table 3.4, is the expected one. The number of the control points is considerably decreased, from 3864 to 712 (despite all the boundary edge nodes are included), influencing the internal node displacement with an error equal to 2.42 %.

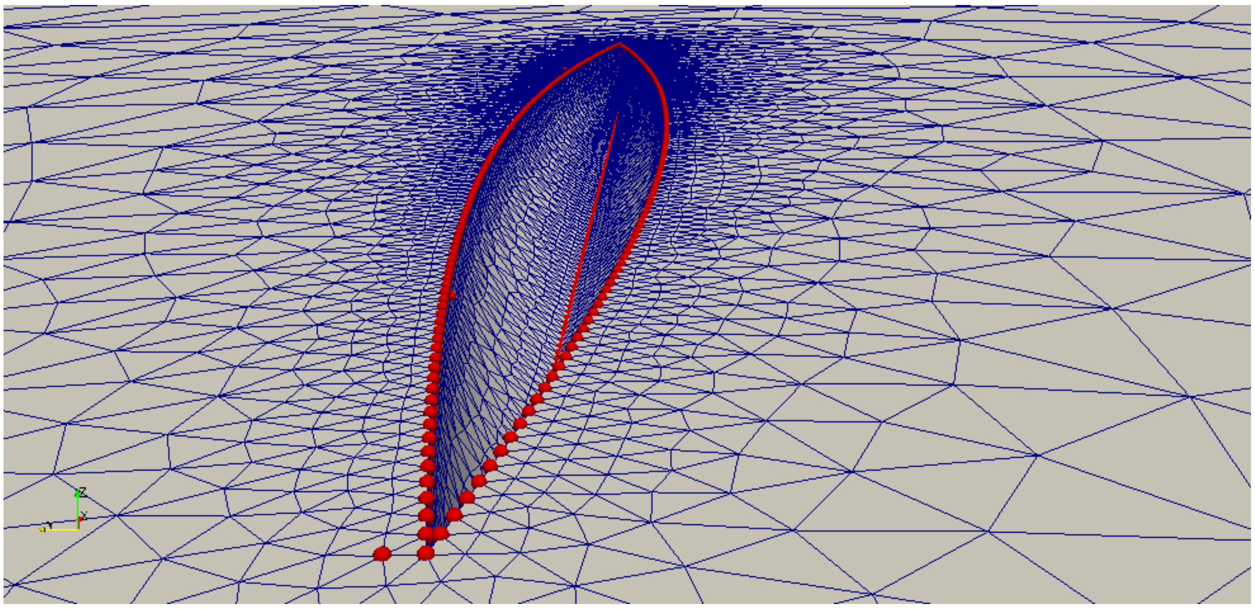


Figure 3.21: Undeformed mesh for hull walls immersed in a fluid domain with control points selected by RIDW algorithm (red spheres).

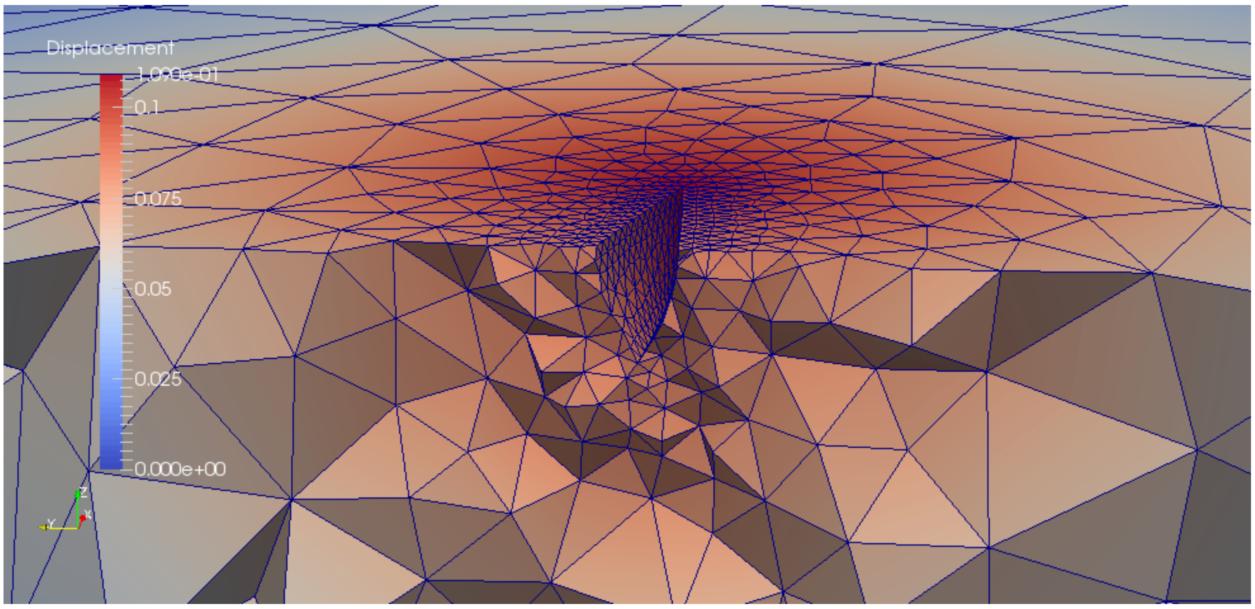


Figure 3.22: Mesh motion obtained via the RIDW applied to the mesh in Figure 3.21.

Mesh dimensions	$50\ \mathrm{m} \times 25\ \mathrm{m} \times 10\ \mathrm{m}$
Hull length	2.50 m
Number of elements	30265
Number of nodes	7186
Number of control points	712
Radius of reduction	$R_{top} = 2$ m
	$R_{bottom} = 2$ m
	$R_{right} = 2$ m
	$R_{left} = 2$ m
	$R_{front} = 2$ m
	$R_{rear} = 2$ m
	$R_{righthole} = 2$ m
	$R_{lefthole} = 2$ m
Number of internal nodes	3322
Number of boundary nodes	3864
CPU Time	8.976 s
Relative error respect to the displacement	2.42 %

Table 3.4: Quantitative information about the RIDW computed 3D mesh motion showed in Figure 3.22.

Furthermore, the reduction on the number of the control points allows to obtain the good results in Figures 3.21 and 3.22, with a lower CPU time (8.976 s) with respect to the *standard* IDW (19.9525 s).

3.3.5 RIDW for structural mesh motion: the wing

We complete investigation of the performances of RIDW by considering the structural mesh in Section 2.2.6. We use the three-dimensional mesh of the wing to integrate the knowledge about the RIDW application on the mesh motion of a typical aeronautic structural mesh.

The mesh motion operated by the RIDW technique on the structural grid shown in Figures 3.23 and 3.24, requires 1.760 s, using 308 control points. As for the previous cases, the price to pay to reduce the dimension of the IDW matrix is an error on the internal node displacement equal to 1.10 % with respect to the IDW standard method.

In Figure 3.23 it is possible to observe the distribution of the control points (red spheres). Also in this case, the nodes belonging to the edges are all selected as control points in the last phase of the DC-IDW process.

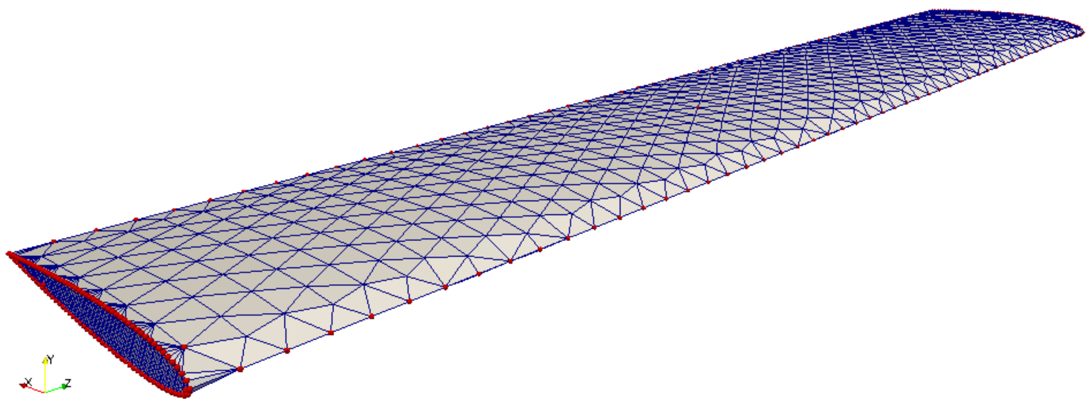


Figure 3.23: Undeformed mesh of a wing where the control points selected by RIDW are represented by th red spheres.

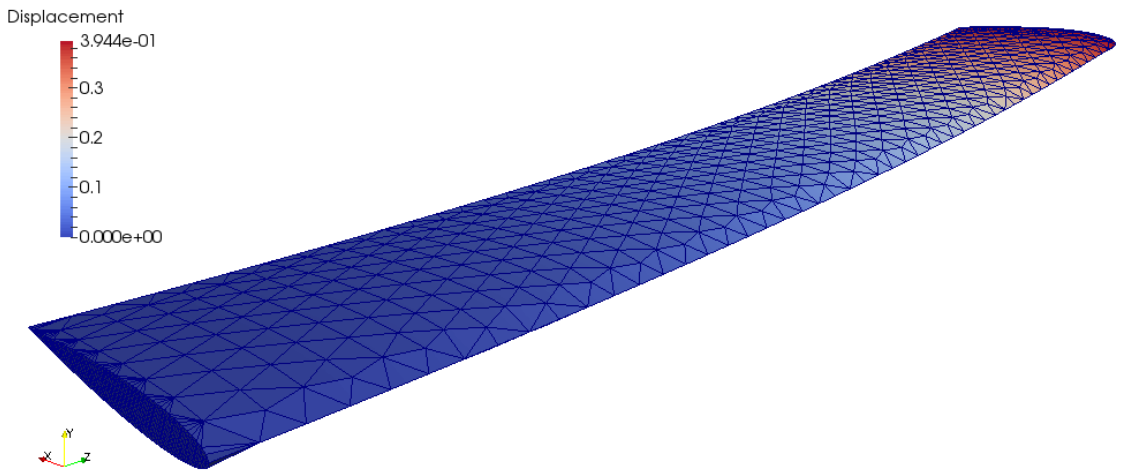


Figure 3.24: Mesh motion obtained via the RIDW process for a structural mesh representing a wing of NACA0012 profile.

Wing longitudinal dimension	2π m
NACA0012 profile chord length	1.01 m
Number of elements	8850
Number of nodes	2463
Number of control points	308
Radius of reduction	$R_{top} = 0.7$ m
	$R_{bottom} = 0.7$ m
	$R_{right} = 2.5$ m
	$R_{left} = 2.5$ m
Number of internal nodes	797
Number of boundary nodes	1666
CPU Time	1.760 s
Relative error respect to the displacement	1.10 %

Table 3.5: Quantitative information about the 3D wing mesh showed in Figure 3.23.

3.4 Identification of rigid movements

The consequences of rigid movements of an interface (such as, the hull test case of Section 2.2.5) significantly affect the solution of mesh motion problems from the beginning. Recently in the literature, a decomposition strategy has been proposed by Romanelli [68] in order to split the rigid roto-translation and the small deformation analysis starting from the solution obtained by the structure solver.

In this section we provide the main steps of the strategy proposed in [68] to emphasize its capability to manage fluid-structure interaction problems involving, free-bodies (not constrained ones) as structure.

During the simulation process, the structure may be subject to both deformations and rigid movements, due to the effect of the fluid forces acting on the external surfaces of the structure. In the presence of free-bodies, in order to efficiently solve the mesh motion problem, we should be able to know the amplitude of both the two displacement components (rigid and deformative). Therefore, we can split the structure displacement into two different components: a global *large* one, related to structural **rigid translations and/or rotations** and, a local one, due to *small* **elastic deformations**. If we denote by $\varsigma \in \Omega^S$ the total structural displacement field, we can decompose ς as:

$$\varsigma = \varsigma_R + \varsigma_D \tag{3.3}$$

where:

- ς_R, represents the structural rigid movements;
- ς_D, indicates the small deformative component.

It is important to remark that we have classified as *large* and *small*, the rigid and the elastic component of the displacement, respectively, since the first one involves a perturbation of the whole structural domain, while, the second one generally leads to localized shape deformations.

The goal is to set up a methodology in order to identify both rigid translations and rotations, while, for the recovery of the deformative component, the shape parametrization technique introduced in Chapter 2 will be employed.

In particular, we can observe that the motion of the fluid mesh, when considering free-bodies, can be known a priori (e.g., in the case of imposed rotations of an airfoil, aileron deflections, etc.) or it can be the output returned by the structural subsystem. Here, we focus on the latter case, that leads to an identification problem of the structural translations and rotations.

In Figures 3.25 and 3.26, we illustrate the idea of separating the global displacement in to two different components: starting from an initial reference configuration of a wing, we obtain the final one by simulating wing deformations and applying a rigid pitch rotation.

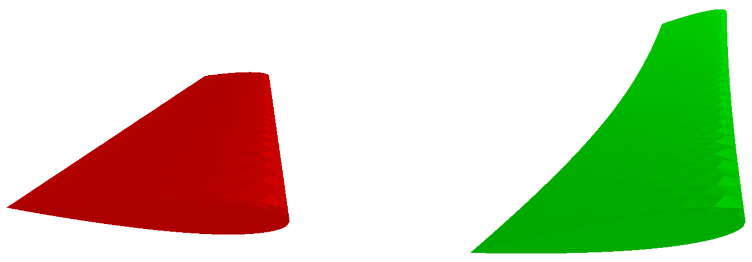


Figure 3.25: Initial (left) and target (right) shape configurations of a wing.

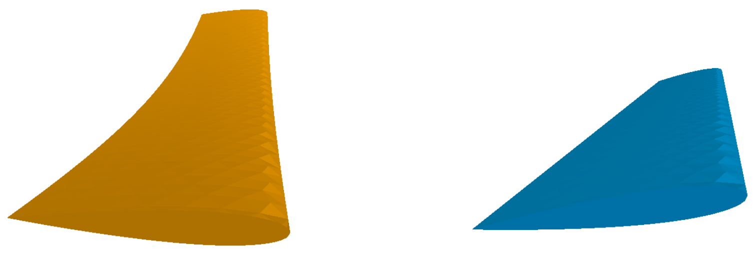


Figure 3.26: Visualization of the different displacement components defining the final configuration: deformative (left) and rigid (right) components.

Thus, in order to analyse this kind of problems and, at the same time, to achieve the best compromise among accuracy, robustness and efficiency, the strategy proposed in [68] uses a hierarchical mesh deformation strategy, based on a modified version of the IDW multivariate interpolation kernel.

The basic idea of this deformation procedure consists of adding in the kinematic description of the grid motion a convenient intermediate frame between the reference and the target configurations.

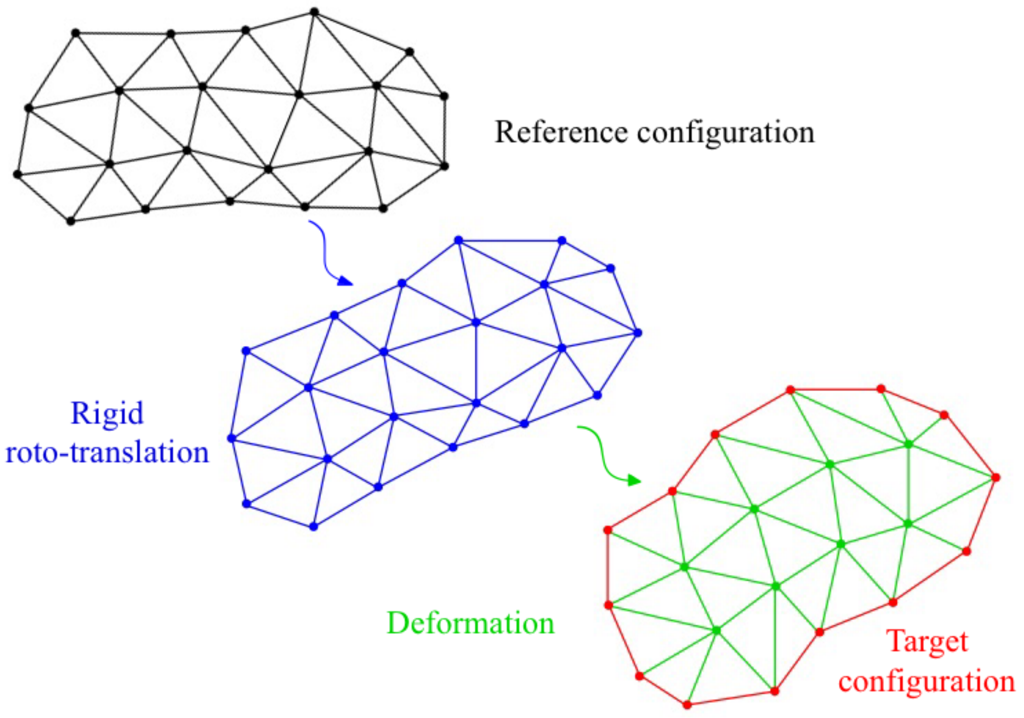


Figure 3.27: Mesh deformation problem with focus on the rigid and deformative components.

As shown in Figure 3.27, the problem can be decomposed into two parts: the identification of the *large* rigid rotation-translation contribution between the reference and the intermediate configurations, and the update of the mesh according to the residual *small* elastic displacement field with a small computational effort and an improved robustness. Thus, such a procedure allows to rewrite the structural problem decoupling the rigid and the elastic degrees of freedom. In particular, it is possible to subdivide the grid motion process into three steps.

First of all, using the information about the interpolated structural displacement field $\mathbf{\Delta x}_j$, relative to the reference undeformed configuration $\mathbf{x}_j$ of each $j-th$ boundary degree of freedom, it is possible to identify by means of a Weighted Least Squares (WLS) strategy only the *rigid* translation-rotation part of the overall general big displacements field as

$$\Delta \mathrm{x}_j = \boldsymbol{s} + \boldsymbol{T}\mathrm{x}_j + \boldsymbol{\epsilon}_j = \boldsymbol{s} + (\boldsymbol{R} - \boldsymbol{I})\mathrm{x}_j + \boldsymbol{\epsilon}_j \quad \forall j = 1, ..., \mathcal{N}_c. \tag{3.4}$$

The mean translation vector is indicated by $\boldsymbol{s}$ and the linear map tensor with $\boldsymbol{T}$. This choice leads to an easier implementation with respect to the identification of the mean rotation tensor $\boldsymbol{R}$, since it is not necessary to include within the formulation the orthogonality constraints $\boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{I}$ by means of Lagrange multipliers. In fact, the same result can be retrieved a posteriori using the *Euler-Rodriguez formula* to split the mean rotation tensor $\boldsymbol{R}$ into two contribution: an emisymmetric one given by the tensor $\boldsymbol{E}$ and a symmetric one,$\boldsymbol{S}$, so that

$$(\boldsymbol{R} - \boldsymbol{I}) = \sin\phi \boldsymbol{K}_x + (1 - \cos\phi)\boldsymbol{K}_x\boldsymbol{K}_x = \boldsymbol{E} + \boldsymbol{S}, \tag{3.5}$$

where ϕ is the magnitude of the rotation vector $\boldsymbol{\psi} = \phi\hat{\boldsymbol{k}}$ in the direction $\hat{\boldsymbol{k}}$, while $\boldsymbol{K}_x$ is the emisymmetric tensor representing the linear cross-product operator $\hat{\boldsymbol{k}}\times$. Requiring that the emisymmetric part of the linear map tensor $\boldsymbol{T}$ matches $\boldsymbol{E}$, given by $\boldsymbol{E} = (\boldsymbol{T} - \boldsymbol{T}^T)/2$, it is straightforward to retrieve $\boldsymbol{\psi}$ and assembly $\boldsymbol{R}$ with (3.5). The limit of such a procedure lies in the fact that the residual linear deformation tensor $\boldsymbol{D} = \boldsymbol{T}(\boldsymbol{R} - \boldsymbol{I})$ is always symmetric.

As a result of the identification procedure, in the second step of this procedure, the mesh is adjusted only to the smaller residual ϵ_j of each $j - th$ boundary degree of freedom, with a computational effort which is expected to be significantly small. In fact, rather than using algorithms that require solving a system of equations (expensive in terms of computational cost and memory occupation for realistic simulations), it is possible to use an IDW multivariate interpolation scheme. For each $k - th$ internal degree of freedom the displacement $\boldsymbol{\Delta}\mathbf{x}_k$ relative to the reference undeformed configuration x_k can be evaluated by performing the following matrix-vector multiplication, similarly to (2.3):

$$\boldsymbol{\Delta}\mathbf{x}_k = \sum_{j=0}^{\mathcal{N}_c} \frac{\boldsymbol{IDW}_{(k,j)}}{\mid \boldsymbol{IDW}_{(k,:)} \mid}\boldsymbol{\epsilon}_j \qquad \forall k = 1, ..., \mathcal{N}_s. \tag{3.6}$$

Finally, to achieve the best trade-off between memory-efficiency and time-efficiency, one of the modifications applied to the original implementation of IDW technique and proposed in [68] is the so called Sparse Inverse Distance Weighting (SIDW). In this version the IDW matrix is stored in a sparse format as follows:

$$\boldsymbol{SIDW}_{(k,j)} = \boldsymbol{IDW}_{(k,j)} \qquad \text{if} \qquad \frac{\boldsymbol{IDW}_{(k,j)}}{\mid \boldsymbol{IDW}_{(k,:)} \mid} > \xi \tag{3.7}$$

where ξ is a threshold set by the user. Particular attention is required for the tuning of parameter ξ, generally chosen close to 0.005. In fact, if its value is too large and consequently there is not any active $j-th$ column for the $k-th$ row, the internal node displacement can be significantly affected by the missed contribution of the neglected nodes.

The algorithm of the described hierarchical mesh deformation toolbox can be summarized by tackling the simple but representative test problem of computing the internal mesh of an arbitrary object given the reference configuration and the new boundary nodes position in the target configuration. The prescribed displacement field is composed by both *large* rigid rotation-translation and *small* deformative contributions. It is worthwhile to compare the results of the Weighted Least-Squares (WLS) identification of the mean translation vector $\boldsymbol{s}$ and linear map tensor $\boldsymbol{T}$ with those of the a-posteriori recovery of the rotation tensor $\boldsymbol{R}$ using the *Euler-Rodriguez formula*. Not only the identification of the mean linear map tensor $\boldsymbol{T}$ is computationally more efficient than that of the rotation tensor $\boldsymbol{R}$, but the intermediate configuration is also significantly closer to the target one. Therefore, the additional effort requested to go from the intermediate to the target configuration can be expected to be smaller. In other terms, the mesh quality after applying the IDW interpolation can be expected to be higher, improving the robustness of the overall procedure. For examples and further discussion please refer to [68].

3.5 Conclusions and perspectives about RIDW

As shown in all the applications of this chapter, the RIDW algorithm produces a considerable decrease of the number of the control points. Moreover, for more complex meshes, typically with a very large number of boundary nodes, the selection procedure of the control points allows to compute the internal node displacement with a very lower (about 80% less) number of parameters, leading to an important gain in terms of CPU time with respect to *standard* IDW. This is the case, for instance, of the test cases described in Sections 3.3.3 and 3.3.4, where the CPU time is almost 70% less of the one required by the *standard* IDW strategy.

On the other hand, the relative error results significantly influenced by the value of the reduction radius R. High values of R, in fact, will lead the RIDW technique to eliminate a lot of control points whose contribution will be not taken into account for the computation of the internal node displacement. For these reasons, the radius choice has to be operated by considering the degree of accuracy required for the analysis and by avoiding an important loss of information.

The operation of tuning of the parameter R is not always easy, especially when it is necessary to have different values R for differently discretized boundaries. For this reason, a future perspective is to make the tuning process automatic, starting for instance from the local mesh characteristic dimension (i.e., the mean distance between two nodes belonging to the considered boundary). The goal is to take into account a moderate number of control points for each boundary, without neglecting excessive information about the boundary movement, property required for an accurate computation of the internal node motion.

Several kinds of selection of control points can be realized, also depending on the physical phenomenon described with the considered mesh motion. One among these applies a procedure similar to the one described above but substituting the circles with ellipsis drawn in the fluid main stream direction, in order to extract information, discretized but not redundant, about the fluid motion. Another possibility is to thicken the control points distribution in the maximum interface displacement zones to mostly weight the displacement localized in specific parts of the domain.

Obviously this kind of implementation assumes to analyse geometrically the interface motion to point out the parts of the domain with maximum displacement and the direction of fluid motion. However, often it is not easy to extract these information about the domain, especially when the displacements are not amenable to analytical functions. In addition, the risk to overweight the displacements of some control points and inevitably neglect other points can become source of high errors on the internal nodes displacement computation.

The RIDW technique implementation here presented allows to obtain good results in terms of accuracy and of limited CPU time, nevertheless, the time required remains too large for complex engineering applications. For this reason, we resort to a ROM implementation where we use the RIDW technique or the *standard* IDW approach. Doing so, it will be possible to distinguish the *Offline* phase, which collects data about possible and admissible mesh movements, from the *Online* phase which, starting from the data collected, computes rapidly the internal mesh motion in function of an imposed boundary movement.

Chapter 4

IDW and Reduced Order Methods

In this chapter we discuss all the details related to the construction of a Reduced Order Method (ROM) for IDW interpolation techniques.

Since the matrix formulation (2.3) leads to a system of parametrized equations, we focus on the reduction procedure used for parametrized *Partial Differential Equations* (PDEs) [37] which has inspired our implementation of ROM for IDW. First of all, in Section 4.1, we introduce the basic components of a ROM for PDEs. In particular, the algebraic version of a reduced order method which allows to obtain a problem of reduced dimensions is showed in Section 4.1.1 (see also [37]). Section 4.1.2 describes in detail the *Offline-Online* computational splitting procedure. In Section 4.1.3 a possible strategy to choose the basis functions that span the reduced order space is discussed. This strategy is based on Proper Orthogonal Decomposition (POD), one of the most common and suitable techniques used for parametrized problems.

Finally, in Section 4.2 we show how to build a procedure for deformations by implementing a POD-based ROM for RIDW interpolation preserving an efficient *Offline-Online computational splitting* process. Section 4.3 presents the results obtained by the new procedure for the examples shown in the previous chapters, highlighting a particular efficiency in terms of computational and memory effort. In the last section some conclusions about the IDW and RIDW in a ROM context are drawn.

4.1 Reduced Order Methods for parametrized PDEs: main ingredients

The goal of a ROM is to compute in an inexpensive way a low-dimensional approximation of the expensive high-fidelity (parameter dependent) solution. The reduced solution is obtained by a projection onto a small subspace, constituted by *global* (specific) basis functions, instead of a large space of *local* (generic) basis functions. In fact, we can assume that the family of the full-order solutions, obtained for different values of the parameter $\boldsymbol{\mu}$ through a suitable high-fidelity approximation technique, is accurate. Therefore, we expect any high-fidelity solution (that is, for any value of $\boldsymbol{\mu}$ belonging to a defined domain $\mathcal{D}$) to be well approximated in terms of these (global and specific) basis functions. In particular, referring to parametrized PDEs, we can summarize the main ingredient of a ROM as follows:

i. *High-fidelity method*: ROMs do not aim at replacing the high-fidelity method, rather, they build upon it approximating a 'given' solution. The full-order problem reads: find $u_h(\boldsymbol{\mu}) \in V^h$ such that

$$a(u_h(\boldsymbol{\mu}), v_h; \boldsymbol{\mu}) = f(v_h; \boldsymbol{\mu}) \quad \forall v_h \in V^h, \tag{4.1}$$

where V^h is a finite dimensional space of (possibly large) dimension $\mathcal{N}$. The solution of (4.1) can be kept as close as desired to the physical solution u by choosing a suitable discretization space.

ii. *Galerkin projection*: a ROM consists in selecting few **basis functions** $\{\zeta_i\}_{i=1}^N$ and finding the reduced order solution $u_N(\boldsymbol{\mu})$ as a linear combination of these basis functions. The choice of the basis functions differs for the different methods. After choosing the basis functions, the space where we seek the reduced solution is

$$V^N = \text{span}\,\{\zeta_i, i = 1, \dots N\} \in V^h, \tag{4.2}$$

with $N = dim(V^N) \ll \mathcal{N}$. Therefore, the reduced order problem becomes: find $u_N(\boldsymbol{\mu}) \in V^N$ such that

$$a(u_N(\boldsymbol{\mu}), v_N; \boldsymbol{\mu}) = f(v_N; \boldsymbol{\mu}) \quad \forall v_N \in V^N \tag{4.3}$$

iii. *Offline-Online computational splitting*: the most important goal is to perform an extensive snapshots generation only once (*Offline stage*), followed by a rapid evaluation of the solution for any new value $\boldsymbol{\mu}$ (*Online stage*). Unfortunately, although problem (4.3) is nominally of small size N, assembling its structures (matrix and right hand side) involves entities associated with our $\mathcal{N}$-dimensional interpolation space. However, the matrix assembly procedure is confined in the *Offline stage* thus, does not affect the rapidity of solution evaluation performed in the *Online stage*.

4.1.1 From high-fidelity to reduced order methods

In this Section, we first introduce a reduced order method to discuss the algebraic connection between the reduced order problem and the high-fidelity one [70].

As shown in the previous section, we assume that the reduced order solution is given by

$$u_N(\boldsymbol{\mu}) = \sum_{m=1}^{N} u_N^m(\boldsymbol{\mu})\zeta^m. \tag{4.4}$$

Inserting (4.4) in (4.3) and choosing $v_N = \zeta^n$, for $n = 1 \dots N$, we obtain the reduced order algebraic system

$$\sum_{m=1}^{N} a(\zeta^m, \zeta^n; \boldsymbol{\mu}) u_N^m(\boldsymbol{\mu}) = f(\zeta^n; \boldsymbol{\mu}), \quad n = 1 \dots N. \tag{4.5}$$

Hence, we can write the problem as

$$\boldsymbol{A}_N(\boldsymbol{\mu})\boldsymbol{u}_N(\boldsymbol{\mu}) = \boldsymbol{b}_N(\boldsymbol{\mu}), \tag{4.6}$$

where, in particular

$$\begin{cases} (\boldsymbol{A}_N(\boldsymbol{\mu}))_{nm} & = & a(\zeta^m, \zeta^n; \boldsymbol{\mu}), \\ (\boldsymbol{b}_N(\boldsymbol{\mu}))_n & = & f(\zeta^n; \boldsymbol{\mu}) \end{cases}$$

and $\boldsymbol{u}_N(\boldsymbol{\mu}) = (u_N^1(\boldsymbol{\mu}), \dots, u_N^N(\boldsymbol{\mu}))^T$ is the vector of the degrees of freedom related to the reduced order solution. Since each basis function ζ^n belongs to the space V^h, we can express each of them as:

$$\zeta^n = \sum_{i=1}^{\mathcal{N}} \zeta_i^n \eta_i, \quad n = 1 \dots N, \tag{4.7}$$

that is, as a linear combination of the high-fidelity space basis functions $\{\eta_i\}_{i=1}^{\mathcal{N}}$. Then, by inserting (4.7) in (4.5), and after denoting by

$$\mathcal{Z} = [\boldsymbol{\zeta}_1, \dots, \boldsymbol{\zeta}_N] \in \mathbb{R}^{\mathcal{N} \times N} \tag{4.8}$$

the matrix of the N degree of freedom $\boldsymbol{\zeta}$ related to each reduced basis function, we have that

$$\begin{array}{rcl} \boldsymbol{A}_N(\boldsymbol{\mu}) & = & \mathcal{Z}^T \boldsymbol{A}(\boldsymbol{\mu}) \mathcal{Z} \\ \boldsymbol{b}_N(\boldsymbol{\mu}) & = & \mathcal{Z}^T \boldsymbol{b}(\boldsymbol{\mu}), \end{array} \tag{4.9}$$

where $\boldsymbol{A}(\boldsymbol{\mu})$ and $\boldsymbol{b}(\boldsymbol{\mu})$ are the structures given by the discretization of the full-order problem (4.1):

$$\boldsymbol{A}(\boldsymbol{\mu})\boldsymbol{u}(\boldsymbol{\mu}) = \boldsymbol{b}(\boldsymbol{\mu}). \tag{4.10}$$

A graphical sketch related to the assembling of matrix $\boldsymbol{A}$ is provided in Figure 4.1.

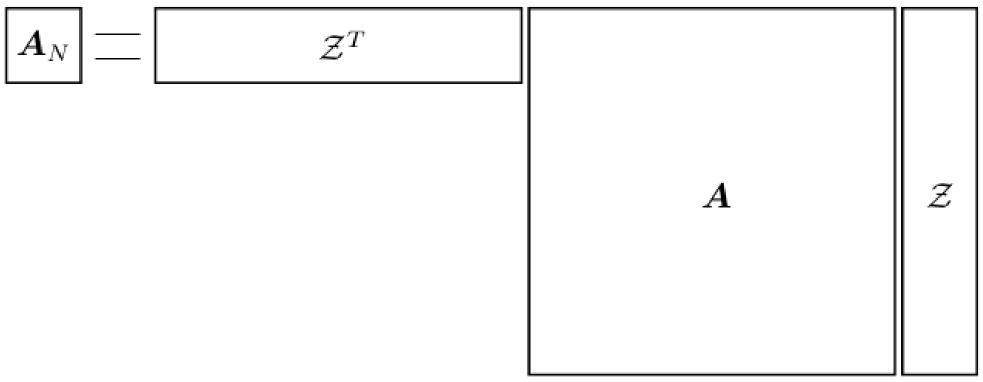


Figure 4.1: Schematic representation of the reduced matrix assembly procedure.

We can now characterize the error between the reduced order solution and the high-fidelity one

$$\boldsymbol{e}_N(\boldsymbol{\mu}) = \boldsymbol{u}(\boldsymbol{\mu}) - \mathcal{Z}\boldsymbol{u}_N(\boldsymbol{\mu}) \tag{4.11}$$

in terms of the high-fidelity residual of the approximated solution, given by

$$\boldsymbol{r}(\boldsymbol{\mu}) = \boldsymbol{b}(\boldsymbol{\mu}) - \boldsymbol{A}(\boldsymbol{\mu})\mathcal{Z}\boldsymbol{u}_N(\boldsymbol{\mu}). \tag{4.12}$$

In fact, the following relation holds:

$$\boldsymbol{A}(\boldsymbol{\mu})\boldsymbol{e}_N(\boldsymbol{\mu}) = \boldsymbol{r}(\boldsymbol{\mu}) \tag{4.13}$$

4.1.2 Offline-Online computational procedure

System (4.6) is nominally of small size coinciding with a set of N linear equations in N unknowns. However, the assembling of the matrix and of the right hand side, involves entities associated with the $\mathcal{N}$-dimensional approximation space as required by (4.9). To obtain an efficient model for the *Offline-Online procedure* [37], we can rely on affine parameter dependence by assuming that:

$$\begin{array}{rcl} \boldsymbol{A}(\boldsymbol{\mu}) & = & \sum_{l=1}^{L} \gamma^l(\boldsymbol{\mu})\boldsymbol{A}^l(\boldsymbol{\mu}) \\ \boldsymbol{b}(\boldsymbol{\mu}) & = & \sum_{m=1}^{M} \tau^m(\boldsymbol{\mu})\boldsymbol{b}^m(\boldsymbol{\mu}). \end{array} \tag{4.14}$$

Thus, we can substitute this expression in (4.9) to compute:

$$\boldsymbol{A}_N^l = \mathcal{Z}^T\boldsymbol{A}^l\mathcal{Z}, \quad \boldsymbol{b}_N^m = \mathcal{Z}^T\boldsymbol{b}^m. \tag{4.15}$$

At this point, system (4.6) can be expressed in matrix form as

$$\left(\sum_{l=1}^{L} \gamma^l(\boldsymbol{\mu}) \boldsymbol{A}_N^l\right) \boldsymbol{u}_N(\boldsymbol{\mu}) = \sum_{m=1}^{M} \tau^m(\boldsymbol{\mu}) \boldsymbol{b}_N^m. \tag{4.16}$$

In this way, computation entails an expensive $\boldsymbol{\mu}$-independent *Offline stage* performed only once, and just an *Online stage* for any chosen value of the parameter $\boldsymbol{\mu} \in \mathcal{D}$. During the former phase, the structures $\{\boldsymbol{A}_N^l\}_{l=1}^L$ and $\{\boldsymbol{b}_N^m\}_{m=1}^M$, as well as the basis matrix $\mathcal{Z}$, are computed and stored. In the latter phase, for any given $\boldsymbol{\mu}$, all the $\gamma^l(\boldsymbol{\mu})$ and $\tau^m(\boldsymbol{\mu})$ coefficients are evaluated, and the $N \times N$ linear system (4.16) is assembled and solved, in order to get the reduced basis approximation $\boldsymbol{u}_N(\boldsymbol{\mu})$.

The *Online operation* cost is $O(LN^2)$ to assemble the matrix, $O(N^3)$ to invert it and $O(MN)$ to get the right hand side in (4.16). The *Online cost* to evaluate $\boldsymbol{u}^N(\boldsymbol{\mu})$ is thus independent of $\mathcal{N}$.

4.1.3 Reduced space construction by Proper Orthogonal Decomposition (POD)

In the last decades a considerable progress has characterized in strategies for the construction of reduced order spaces [13]. We limit to the **Proper Orthogonal Decomposition** (POD) method [21, 20]. Among the several reduced order modelling techniques proposed in the literature [32, 60, 59, 67, 78, 34, 3], one of the most popular and suitable for the aeronautical and naval engineering applications is surely POD and for this reason has inspired our ROM implementation.

The Proper Orthogonal Decomposition (POD) technique reduces the dimensionality of a system by transforming the original variables into a new set of uncorrelated variables (called **POD modes**, or principal components), so that the first few modes ideally retain most of the 'energy' present in all of the original variables. For this reason, POD relies on the use of the singular value decomposition (SVD) algorithm (see, e.g., [82, 62, 25]). Consider a discrete set of n_{train} values for the parameters and assemble the snapshot matrix $\mathcal{U} \in \mathbb{R}^{\mathcal{N} \times n_{\text{train}}}$ as follows:

$$\mathcal{U} = [\boldsymbol{u}(\boldsymbol{\mu}_1), \cdots, \boldsymbol{u}(\boldsymbol{\mu}_{n_{\text{train}}})], \tag{4.17}$$

where the columns are the solution vectors for the n_{train} selected values for the parameters. The SVD of $\mathcal{U}$ reads

$$\mathcal{V}^T \mathcal{U} \mathcal{W} = \begin{pmatrix} \Sigma & 0 \\ 0 & 0 \end{pmatrix}, \tag{4.18}$$

where

$$\mathcal{W} = [\zeta_1, \zeta_2, \cdots, \zeta_{\mathcal{N}}] \in \mathbb{R}^{\mathcal{N} \times \mathcal{N}} \tag{4.19}$$

and

$$\mathcal{V} = [\Psi_1, \Psi_2, \cdots, \Psi_N] \in \mathbb{R}^{n_{\text{train}} \times n_{\text{train}}} \tag{4.20}$$

are orthogonal matrices, whereas

$$\Sigma = \text{diag}(\sigma_1, \cdots, \sigma_r) \tag{4.21}$$

with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r$; here $r \leq n_{\text{train}}$ is the rank of $\mathcal{U}$.

For any $N \leq n_{\text{train}}$, the POD basis, of dimension N, is defined as the set of the first N left singular vectors $[\boldsymbol{\zeta}_1, \cdots, \boldsymbol{\zeta}_N]$ of $\mathcal{U}$. Thus, the basis matrix is given by

$$\mathcal{Z} = [\boldsymbol{\zeta}_1, \cdots, \boldsymbol{\zeta}_N] \in \mathbb{R}^{\mathcal{N} \times N}. \tag{4.22}$$

By construction, the POD basis is orthonormal. Furthermore, it can be shown that the energy contained in the $\mathcal{N} - N$ neglected modes is

$$E(\mathcal{Z}) = \sum_{i=N+1}^{\mathcal{N}} \sigma_i^2, \tag{4.23}$$

so that, the error associated with the POD approximation is equal to the sum of the squares of the singular values corresponding to the neglected POD modes. In this way, we can select N so that $E(\mathcal{Z}) \leq \epsilon_{tol}^*$, for a prescribed tolerance ϵ_{tol}^*. To do this, it is sufficient to choose N such that

$$I(N) = \frac{\sum_{i=1}^{N} \sigma_i^2}{\sum_{i=1}^{\mathcal{N}} \sigma_i^2} \geq 1 - \delta, \tag{4.24}$$

which means to require that the energy retained by the last $\mathcal{N} - N$ modes is at most equal to δ, being δ as small as desired. Quantity $I(N)$ is referred to as the relative information content of the POD basis.

Thus, the POD approach requires to compute n_{train} high-fidelity candidate snapshots, as well as the solution of a SVD problem, while (4.23) provides information about the amount of energy neglected by the selected POD modes.

4.2 A ROM for IDW interpolation technique

As anticipated, solving a problem via IDW or Reduced IDW techniques allows to obtain good results for the computation of the solution at the grid points, but, also with the application of a geometrical reduction it is computationally expensive.

To limit this drawback, we adapt reduced order modelling to the IDW framework (and, in particular, to its geometrically reduced version RIDW), carrying out, first of all, an *Offline stage* inspired by the POD strategy. In this way, during the *Offline phase*, the algorithm extracts the POD modes from the displacement solutions for several values of the parameters. At this point, a specific displacement evaluation for a *given* set of parameters can be performed in the *Online phase*, exploiting the reduced model advantages with an important reduction in terms of computational effort and demanded time.

In our specific case, the decomposition applied to the snapshot matrix $\mathcal{U}$ allows to reconstruct the solution for a generic parameter variation that affects the data point position. The main role in the solution reconstruction is represented by the sum of the basis functions, suitably weighted, that allows to preserve important information. So, denoting by $\boldsymbol{x}_{k_{undef}}$ the original position of the $k-th$ point and with $\boldsymbol{x}_{k_{def}}$ the position of the same point after the deformation, it is possible to write:

$$\boldsymbol{x}_{k_{def}} \approx \boldsymbol{x}_{k_{undef}} + \sum_{i=1}^{N} \beta_i \boldsymbol{d}_i \tag{4.25}$$

where N is the number of POD basis functions chosen to represent any deformation, $\boldsymbol{d}_i$ are the POD basis functions and β_i are the weight coefficients, our unknowns.

It is possible to rewrite Eq. (4.25) in matrix form as

$$\boldsymbol{x}_{k_{def}} \approx \boldsymbol{x}_{k_{undef}} + \mathcal{Z}\boldsymbol{\beta} \tag{4.26}$$

where $\mathcal{Z}$ is the matrix collecting the POD basis functions selected by POD and $\boldsymbol{\beta}$ is the vector of the weights. In this way we perform the generalization of the reduction strategy applied to PDEs in Section 4.1 to the construction process of a ROM to describe mesh deformations.

Considering the matrix formulation of IDW in (2.3), to improve the notation, we introduce a new matrix denoted by $\boldsymbol{CIDW}$ to express the ratio in (2.1), whose elements are defined by the following relation:

$$\boldsymbol{CIDW}_{k,j} = \frac{\boldsymbol{IDW}(k,j)}{\sum_{j=1}^{\mathcal{N}} \boldsymbol{IDW}(k,j)}, \tag{4.27}$$

where $\boldsymbol{IDW}(k,j)$ is defined as in (2.4). Thus, it is possible to define the matrix composed by the elements in (4.27) as the product of a diagonal matrix, of dimensions $[\mathcal{N}_s \times \mathcal{N}_s]$, whose $k-th$ entry coincides with the inverse of the sum of the entries $\boldsymbol{IDW}(k,j)$:

$$\boldsymbol{CIDW} = Diag\left[\frac{1}{\sum_{j=1}^{\mathcal{N}} \boldsymbol{IDW}(k,j)}\right] \boldsymbol{IDW} \quad for\ k = 1, ..., \mathcal{N}_s. \tag{4.28}$$

Furthermore, we have decided to assemble a $\boldsymbol{CIDW}$ matrix considering all the nodes of the mesh, control points included. Thus, after the first $\mathcal{N}_s$ rows, where $\mathcal{N}_s$ is the number of slave nodes, a $[\mathcal{N}_c \times \mathcal{N}_c]$ identity matrix is added to $\boldsymbol{CIDW}$ so that the resulting $[(\mathcal{N}_s + \mathcal{N}_c) \times \mathcal{N}_c]$ matrix is defined by

$$\boldsymbol{TIDW} = [\,\boldsymbol{CIDW};\ \boldsymbol{I}_{N_c}\,] \tag{4.29}$$

$\boldsymbol{TIDW}$ is assembled as shown in Figure 4.2. .

This choice allows to obtain, after the Galerkin projection [16], basis functions influenced by all the grid points. Indeed, the snapshots matrix includes the control points displacement.

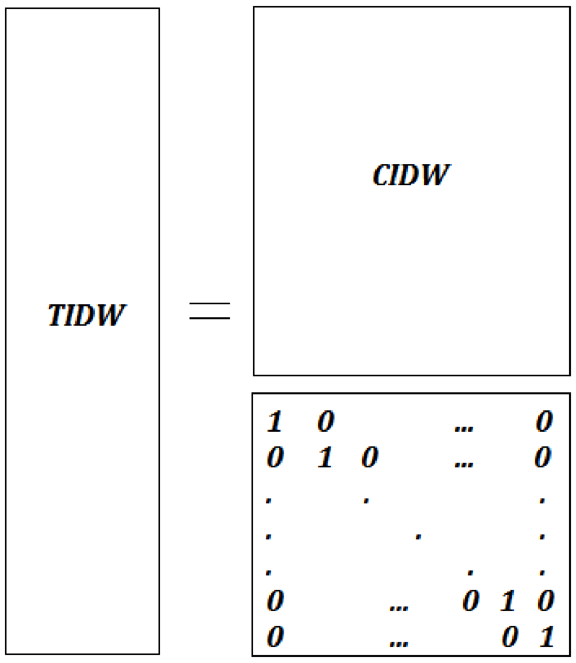


Figure 4.2: Schematic representation of the TIDW matrix

Thus, it is possible to write the IDW formula for all the mesh nodes as follows:

$$\boldsymbol{\delta} = \boldsymbol{TIDW}\,\boldsymbol{\delta_c} \tag{4.30}$$

where $\boldsymbol{\delta}$ denoted the displacement column vector $(\boldsymbol{x}_{def} - \boldsymbol{x}_{undef})$ of all the $(\mathcal{N}_s + \mathcal{N}_c)$ points and, similarly, $\boldsymbol{\delta_c}$ is the displacement column vector of $(\mathcal{N}_c)$ control points. The similarity between the expressions (4.30) and (4.26) is evident:

$$\begin{array}{rcccc} \boldsymbol{x}_{k_{def}} & = & \boldsymbol{x}_{k_{undef}} & + & \boldsymbol{TIDW}\,\boldsymbol{\delta_c} \\ \boldsymbol{x}_{k_{def}} & \approx & \boldsymbol{x}_{k_{undef}} & + & \mathcal{Z}\boldsymbol{\beta}. \end{array} \tag{4.31}$$

Now we aim at obtaining a value for $\boldsymbol{\beta}$ in order to have results as similar as possible for both these expressions.

To reach this goal we resort to the Galerkin projection using a least squares approach. First of all, we have to pre-multiply Eq.(4.30) by $\boldsymbol{TIDW}^T$ in order to obtain:

$$\boldsymbol{TIDW}^T\boldsymbol{\delta} = \boldsymbol{TIDW}^T\,\boldsymbol{TIDW}\,\boldsymbol{\delta_c} \tag{4.32}$$

and thus we express $\boldsymbol{\delta_c}$ in terms of $\boldsymbol{\delta}$ as

$$\boldsymbol{\delta_c} = (\boldsymbol{TIDW}^T\,\boldsymbol{TIDW})^{-1}\,\boldsymbol{TIDW}^T\,\boldsymbol{\delta} \tag{4.33}$$

so that, denoting by $\boldsymbol{IDW}^+$ the matrix $[(\boldsymbol{TIDW}^T\,\boldsymbol{TIDW})^{-1}\,\boldsymbol{TIDW}^T]$, we have

$$\boldsymbol{\delta_c} = \boldsymbol{IDW}^+\,\boldsymbol{\delta}, \tag{4.34}$$

having a form very similar to (4.10). Here, $\boldsymbol{IDW}^+$ plays the role of $\boldsymbol{A}(\boldsymbol{\mu})$, $\boldsymbol{\delta}$ the one of $\boldsymbol{u}(\boldsymbol{\mu})$, while $\boldsymbol{\delta_c}$ replaces the right-hand side $\boldsymbol{b}(\boldsymbol{\mu})$.

Moreover, if we exploit the second equation of system (4.31), we can approximate $\boldsymbol{\delta}$ with the product $\mathcal{Z}\boldsymbol{\beta}$ so that

$$\boldsymbol{\delta_c} \approx \boldsymbol{IDW}^+\,\mathcal{Z}\boldsymbol{\beta} \tag{4.35}$$

The term multiplying vector $\boldsymbol{\beta}$ is a Galerkin projection, so that multiplying (4.32) on the left side by $\mathcal{Z}^T\,(\boldsymbol{IDW}^+)^T$, we obtain:

$$\mathcal{Z}^T\,(\boldsymbol{IDW}^+)^T\,\boldsymbol{TIDW}^T\boldsymbol{\delta} \approx \mathcal{Z}^T\,(\boldsymbol{IDW}^+)^T\,\boldsymbol{TIDW}^T\,\boldsymbol{TIDW}\,\boldsymbol{\delta}_c \tag{4.36}$$

i.e., using relation $\boldsymbol{\delta} \approx \mathcal{Z}\boldsymbol{\beta}$ written in (4.31),

$$\mathcal{Z}^T\,\boldsymbol{IDW}^*\,\mathcal{Z}\,\boldsymbol{\beta} \approx \mathcal{Z}^T\,\boldsymbol{IDW}^*\,\boldsymbol{TIDW}\,\boldsymbol{\delta_c} \tag{4.37}$$

where the product $(\boldsymbol{IDW}^+)^T\,\boldsymbol{TIDW}^T$ is denoted by $\boldsymbol{IDW}^*$.

At this point, the *Offline phase* is finished with the assembling of system (4.37) and the storage of all the matrices which compose it.

Information collected in the different matrices involved in (4.37), will be used, after the assignment of a control points displacement vector $\boldsymbol{\delta_c}$, in the execution of the two steps representing the *Online computational phase*, i.e.,

- computation of $\boldsymbol{\beta}$ vector;
- solution reconstruction.

To compute $\boldsymbol{\beta}$, we have to solve a system with a very reduced dimension (N) with respect to the original one ($\mathcal{N}$). To solve the solution at each grid point, we exploit relation (4.26), simply correcting the original position of grid points with the update vector $\mathcal{Z}\boldsymbol{\beta}$.

At the end of this model order reduction, the computational effort of the *Online phase* has been considerably reduced ($N << \mathcal{N}$) and, for a given and generic (admissible) control point displacement, it is possible to evaluate the displacement solution for all the mesh nodes by the vector $\boldsymbol{\delta}$.

4.3 Relevant applications

We show in this section the results obtained with the ROM technique for IDW for some of the examples considered in the previous chapters.

In particular, these examples allow to numerically evaluate the computational advantages obtained by the implementation of the reduced order method.

As in the previous sections, the most relevant parameter to evaluate the numerical benefits is the CPU time, without neglecting the importance of the error value with respect to the solution computed by *standard* IDW.

The *Offline-Online computational splitting* plays a very important role in this assessment, thus, the CPU time spent to execute the entire algorithm, will be provided split into two values:

1) the *Offline phase CPU time*: this is the time spent to collect the snapshots and to compute all the matrices which will be used in the next (*Online*) phase to evaluate the final solution;

2) the *Online phase CPU time*: this is the time spent to evaluate the final solution. In this phase, the algorithm evaluates the vector $\boldsymbol{\delta_c}$ of the control point displacement and then computes $\boldsymbol{\beta}$ using the matrices assembled in the previous phase and the relation (4.37). This allows to extract the final solution for any given parameter variation.

It is possible to use both the strategies to compute the snapshot solutions in the *Offline phase*: the *standard* IDW and the *reduced* IDW. However, to corroborate the advantages of the conjunction between RIDW and reduce order models, we are going to show only the results associated with RIDW.

In the following applications, the reduction radius R is chosen as in the previous examples. We will specify its value in the quantitative tables.

Finally, we remark that for a prescribed tolerance $\epsilon^*_{tol} = 10^{-5}$, for each of the following examples, it turns out that it is sufficient to consider at most the first two POD modes to obtain accurate results, with a neglected energy content $E(\mathcal{Z})$ very small (of the order of the machine epsilon).

4.3.1 The ROM-RIDW square test case

We exactly refer to the configuration in Section 2.2.1. A visualization of the undeformed and deformed configurations is furnished in Figures 4.3 and 4.4, respectively. In particular, Figure 4.4 shows the deformation recovered by the ROM-RIDW strategy.

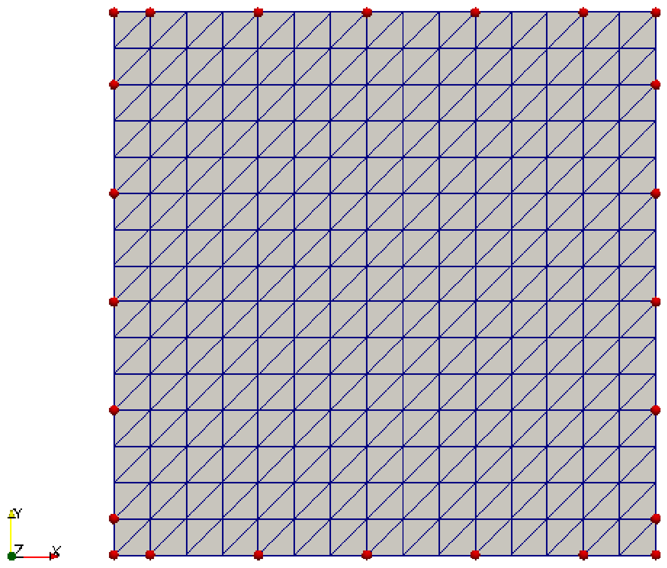


Figure 4.3: Undeformed configuration of the square with control points (red spheres) selected by RIDW process.

The mesh obtained is very similar to the ones in the previous chapters. The main features are summarized in Table 4.1.

We immediately recognize more data in the table with respect to the previous validation, i.e., we have

- the set of values for the parameter used as training set during the *Offline stage*;
- the CPU time distinguished into the *Offline time* and the *Online* time;
- the δ_c vector size;
- the number of considered POD modes, i.e., the size of the vector $\boldsymbol{\beta}$.

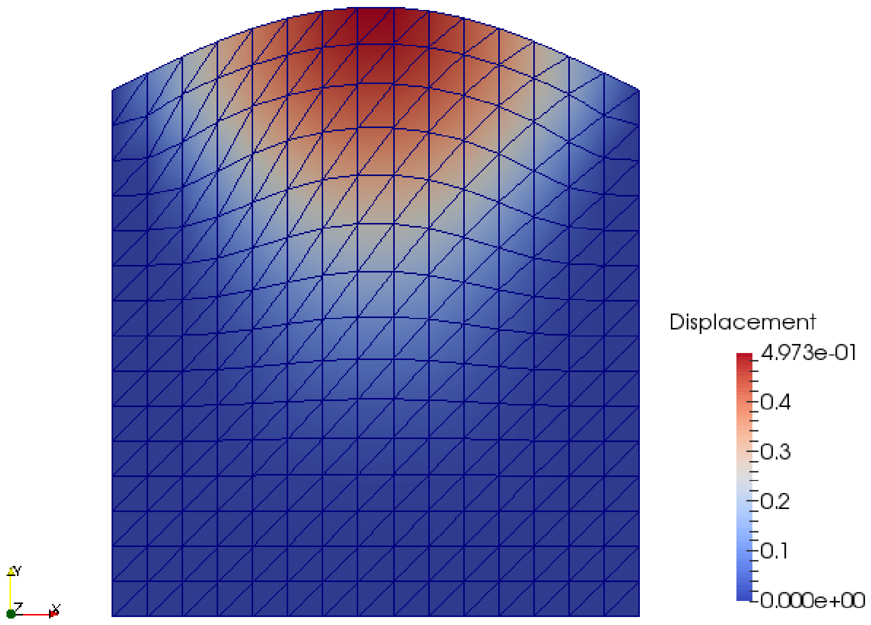


Figure 4.4: Deformed square with internal node displacement computed via ROM-RIDW.

Concerning the first information, with reference to the specific test case, we parametrize the movement of the boundary used in the POD snapshot collection as

$$y(x) = \mu_i \text{sin}(x) \qquad i = 1, ..., 10, \tag{4.38}$$

where μ_i is the *i-th* value of the parameter. As anticipated, in the *Online phase*, for the $\boldsymbol{\delta_c}$ assignment required to find $\boldsymbol{\beta}$, we use the given target configuration identified by the law $y(x) = 0.5\text{sin}(x)$ (see Figure 2.8 in Chapter 2).

The sizes of the vectors $\boldsymbol{\delta}_c$ and $\boldsymbol{\beta}$ allow to numerically quantify the reduction of the system dimension: from $\hat{\mathcal{N}_c}$ (number of control points considered by RIDW) to N ($\boldsymbol{\beta}$ size). In this case, the energy of the system is almost contained in the first POD mode ($N = 1$).

The results obtained in terms of CPU time and error are promising.

The time demanded to compute the same solution using the RIDW strategy is 0.126 s against 0.034 s of the ROM-RIDW approach, while the error does not change significantly (1.85% for RIDW versus 1.98% for ROM-RIDW). The *price* to be paid is the *Offline CPU time* equal to 0.792 s. However, we have to consider that this part of the algorithm is executed once and allows to find the solution for several new parameter values requiring a very short time. In fact, the advantage of this ROM approach is the

Lenght of square side	π m	
Number of total elements	450	
Number of total nodes	256	
Number of internal nodes	196	
Number of boundary nodes	60	
Radius of reduction	$R_{top} = 0.5$ m	
	$R_{bottom} = 0.5$ m	
	$R_{right} = 0.5$ m	
	$R_{left} = 0.5$ m	
Number of control points	24	
$\boldsymbol{\delta}_c$ size	$[24 \times 2]$	
Parameter values	$\{0.05, 0.04, 0.08, 0.03, 1, 0.09, 1.3, 0.06, 0.07, 0.01\}$ m	
Number of POD modes selected	1	
Offline CPU Time	0.792 s	
Online CPU Time	0.034 s	
Percentage of error respect to the displacement	1.98 %	

Table 4.1: Quantitative information related to the deformation computed by ROM-RIDW of the square in Figure 3.15.

fast computation in the case of several repeated deformations (as for *real-time* or *shape optimization* problems). If we want to know just the displacement solution for a single assigned interface movement, the simple *standard* IDW or the new RIDW strategies result fast enough, without using ROM, which in such a case simply slows down the evaluation of the solution without any gain in terms of accuracy.

4.3.2 The ROM-RIDW beam test case

We consider the configuration in Section 2.2.3.

As previously remarked, the structure is not meshed. The mesh considered is the fluid one, and part of the boundary constitutes the interface with the structure. In this case the target configuration used for the *Online* phase is the one which moves the interfaces between the structure and the fluid according to function $y(x) = 0.01x^2$. We provide the undeformed starting mesh in Figure 4.5 and the deformed mesh obtained by merging RIDW with ROM in Figure 4.6.

We furnish in Table 4.2 the setting adopted for the test case and the obtained results.

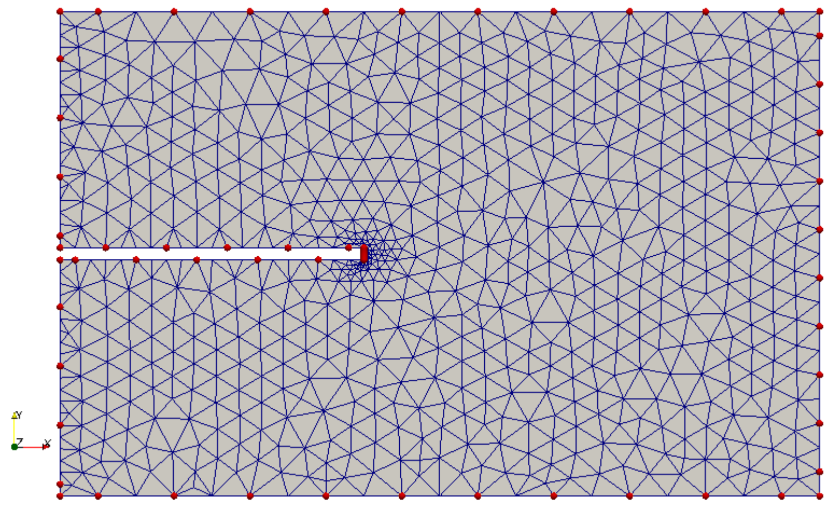


Figure 4.5: Undeformed configuration for the cantilever beam test case.

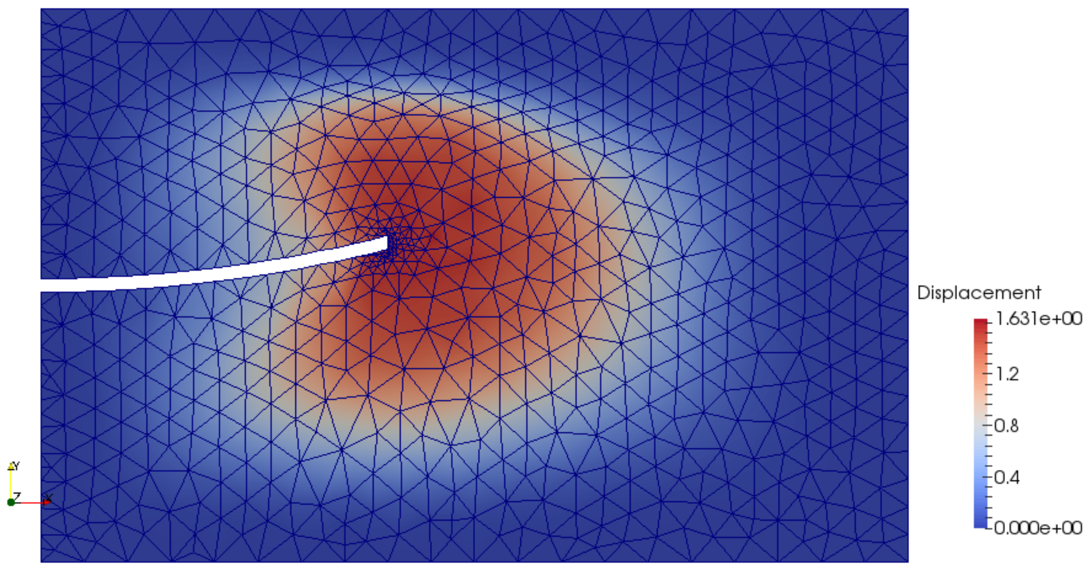


Figure 4.6: Deformed fluid domain mesh of the cantilever beam test case computed via ROM-RIDW.

The parameters used in the snapshot collection are identified by the function:

$$y(x) = \mu_i x^2 \qquad for\ i = 1, ..., 10 \tag{4.39}$$

where μ_i is the *i-th* value for the parameter.

Mesh dimensions	6π m $\times$ 16 m
Rectangular hole dimensions	2π m $\times$ 1 m
Number of total elements	1360
Number of total nodes	761
Number of internal nodes	601
Number of boundary nodes	160
Number of control points	66
$\boldsymbol{\delta}_c$ size	$[66 \times 2]$
Radius of reduction	$R_{top} = 2$ m
	$R_{bottom} = 2$ m
	$R_{right} = 2$ m
	$R_{upleft} = 2$ m
	$R_{downleft} = 2$ m
	$R_{tophole} = 2$ m
	$R_{bottomhole} = 2$ m
	$R_{righthole} = 0.03$ m
Parameter values	$\{0.001, 0.02, 0.018, 0.03, 0.07,$ $0.015, 0.009, 0.005, 0.0013, 0.003\}$ m
Number of POD modes selected	1
Offline CPU Time	2.893 s
Online CPU Time	0.045 s
Percentage of error respect to the displacement	4.80 %

Table 4.2: Quantitative information related to the fluid mesh motion around the cantilever beam in Figure 4.5 obtained by ROM-RIDW.

The results underline the advantages of using the RIDW strategy and, in particular, its reduced order model implementation.

In fact, also in the previous test case, the *Online CPU time* is very small (just 0.045 s) while the RIDW computation requires 0.280 s to compute the solution of the same example. The percentage of error with respect to the displacement computed by *standard* IDW remains under 5%, as for the RIDW method.

Such a gain in terms of CPU time is due to the twofold reduction process: the reduction of the number of control points by RIDW strategy and successively the reduction of the dimension of the system to be solved via ROM, from the initial $\hat{\mathcal{N}}_c$ dimension ($\boldsymbol{\delta}_c$ size) to the smaller one N ($\boldsymbol{\beta}$ size).

The efficiency and the accuracy of the ROM-RIDW approach justify the employment of such a strategy. The *Offline phase* requires about 3 s, a very long time with respect to the solution evaluation of the *Online phase*. However, as previously remarked, in the *Offline phase* the algorithm computes ten solutions, one for each parameter value. Furthermore, the *Offline steps* are executed just one time, allowing to find several solutions for several new value of μ with a considerable saving of CPU time.

4.3.3 Mesh motion around a deformed wing via ROM-RIDW

A more complex, but very interesting, application is represented by the configuration in Section 2.2.4, i.e., the NACA0012-based wing immersed in a tridimensional fluid domain. The geometry considered is a parallelepiped with a wing-shaped hole fixed at the left wall towards the center of the domain. The interface between fluid and structure in the target configuration results moved according to function $y(z) = 0.01z^2$.

First of all, we show the undeformed, i.e., the initial, configuration in Figure 4.7, and the deformed one obtained via RIDW combined with the ROM in Figure 4.8.

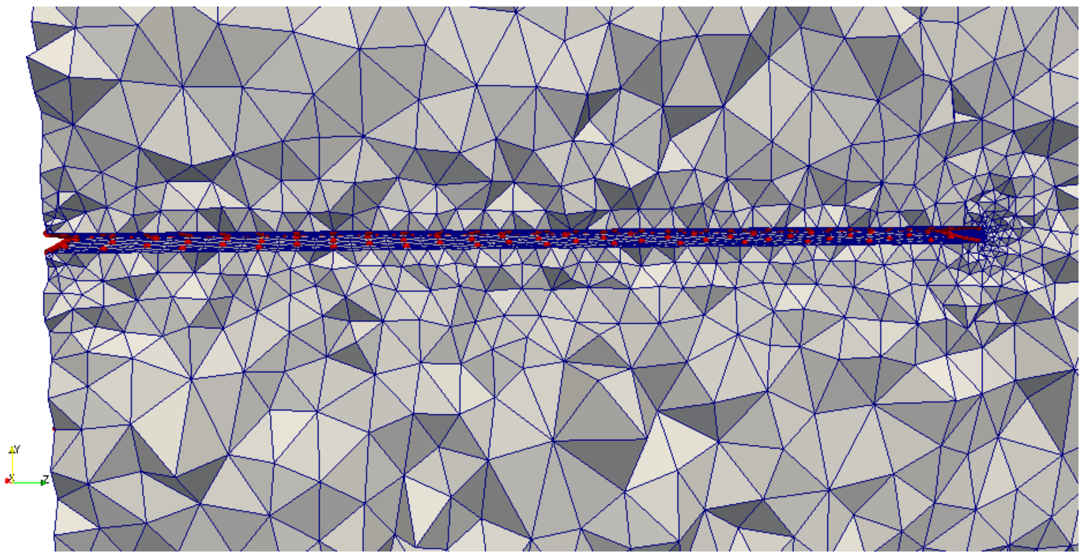


Figure 4.7: Undeformed configuration of the fluid domain around a 3D NACA0012-based wing.

Table 4.3 collects the setting adopted for the validation and the results obtained during the computation of the deformed structure.

Also for this example, the results show the potentialities of the joint use of RIDW and ROM.

The error is about the same as the one characterizing the simple RIDW strategy, i.e., 5.66% for ROM-RIDW against 5.64% for RIDW.

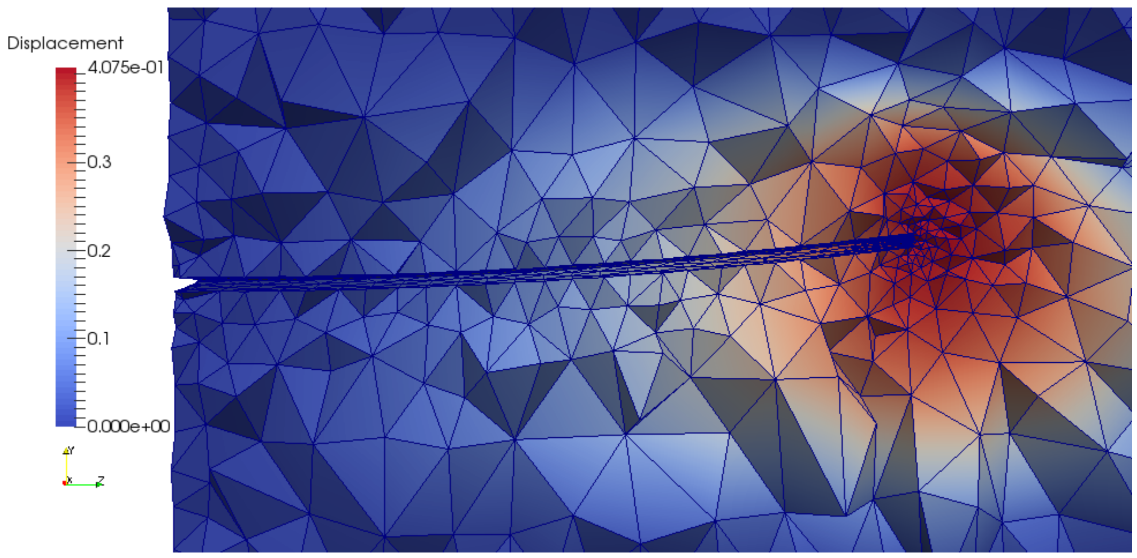


Figure 4.8: Deformed tridimensional fluid mesh computed via ROM-RIDW algorithm.

The *Online CPU time* is very lower with respect to the time required by RIDW: only 2.200 s for the ROM application versus 151.392 s demanded by RIDW.

The parametric function is the same as the one used for the previous bidimensional example, namely,

$$y(z) = \mu_i z^2 \qquad for\ i = 1, ..., 10 \tag{4.40}$$

The μ_i parameters are shown in Table 4.3 and the solutions collected in the snapshots matrix are associated with these values.

The considerable saving of CPU time, also for this more complex application, together with the reduced memory demand represent the significant advantages of the ROM-RIDW process and balance the *Offline CPU time* (751.410 s).

For this example, we explicitly provide the first singular values identified by the POD process to check their decay.

$$\lambda_1 = 518944;\ \lambda_2 = 7.005e^{-11};\ \lambda_3 = 2.948e^{-11};\ \cdots \tag{4.41}$$

The first eigenvalue, λ_1, is significantly higher with respect to the other ones. This means that most of the 'system energy' is contained in the first POD mode. Thus, to accurately describe the system, it is sufficient to consider only the first mode in the ROM-RIDW algorithm.

Moreover, since the *energy content* of the neglected POD modes is meaningless, the error (5.66%) in Table 4.3 is entirely due to the RIDW process. In fact, repeating the algorithm execution in a ROM-IDW version (instead of the ROM-RIDW combination),

Mesh dimensions	$10\text{ m} \times 5\text{ m} \times 4\pi\text{ m}$
Wing longitudinal dimension	2π m
NACA0012 profile chord length	1.01 m
Number of elements	41152
Number of nodes	9172
Number of internal nodes	4972
Number of boundary nodes	4200
Number of control points	3611
$\boldsymbol{\delta}_c$ size	$[3611 \times 3]$
Radius of reduction	$R_{top} = 0.5$ m
	$R_{bottom} = 0.5$ m
	$R_{right} = 0.25$ m
	$R_{left} = 0.25$ m
	$R_{front} = 0.5$ m
	$R_{rear} = 0.5$ m
	$R_{tophole} = 0.25$ m
	$R_{bottomhole} = 0.25$ m
	$R_{righthole} = 0.25$ m
Parameter values	$\{0.05, 0.04, 0.08, 0.03, 1, 0.1, 1.3, 0.06, 0.07, 0.5\}$ m
Number of POD modes selected	1
Offline CPU Time	751.410 s
Online CPU Time	2.200 s
Percentage of error respect to the displacement	5.66 %

Table 4.3: Quantitative information about the deformed 3D fluid mesh represented in Figure 4.8 computed by ROM-RIDW approach.

the error on the displacement with respect to the solution computed by a single *standard* IDW evaluation is comparable to the machine epsilon.

4.3.4 A ROM-RIDW naval application

The target configuration considered in this naval case is the same as the one computed in Section 3.3.4, i.e., the one yielded by a rotation of the hull with respect to the z axis of an angle equal to -5^o. The starting undeformed configuration and the deformed one obtained by the ROM-RIDW algorithm are furnished in Figure 4.9 and Figure 4.10,

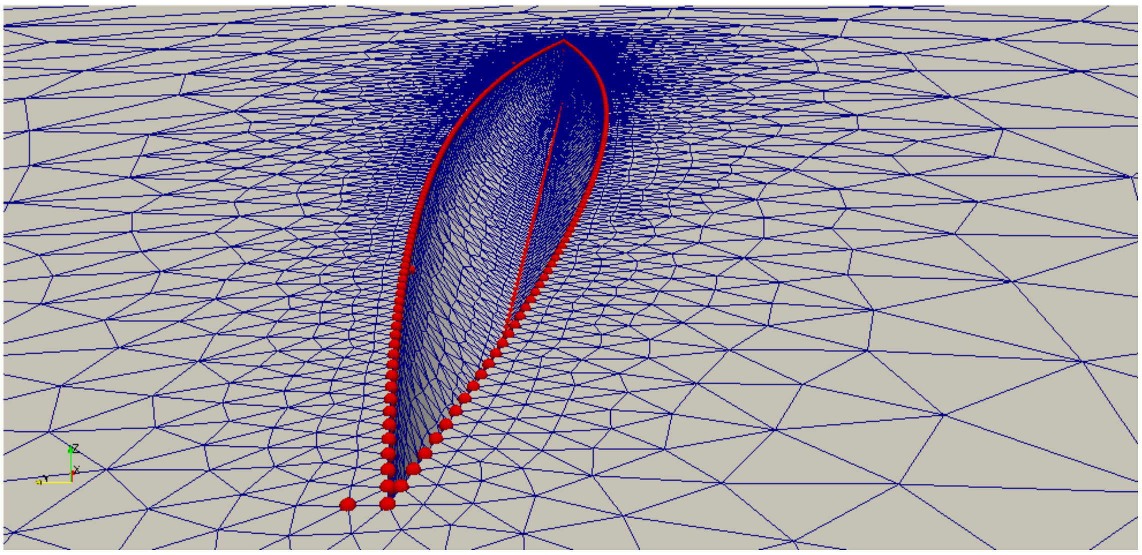


Figure 4.9: Undeformed configuration of a fluid domain around a hull.

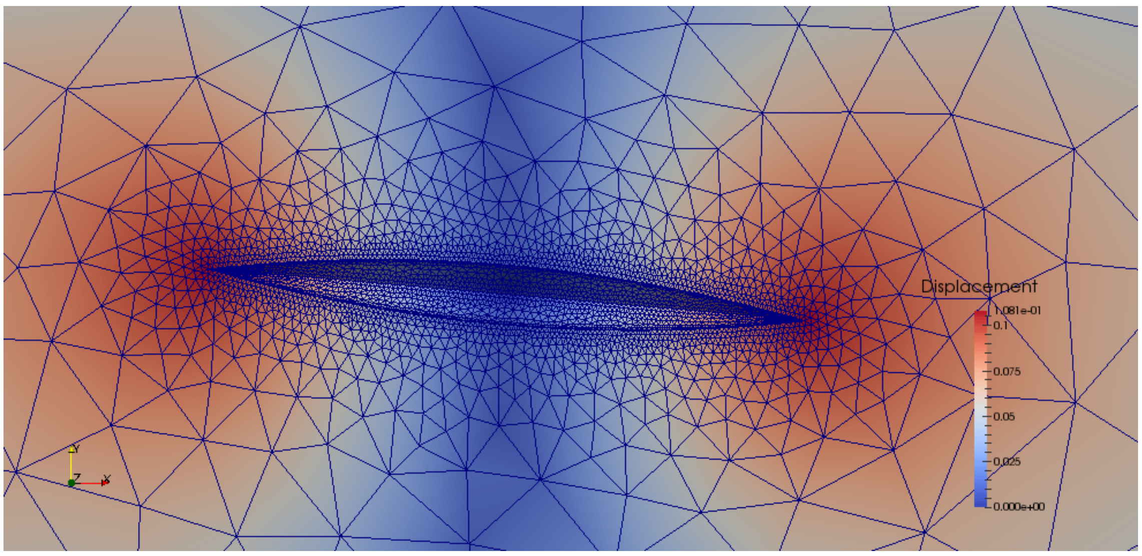


Figure 4.10: Deformed 3D mesh for the domain computed by ROM-RIDW.

respectively.

In the Table 4.4 the setting used for this application and the numerical results obtained are gathered.

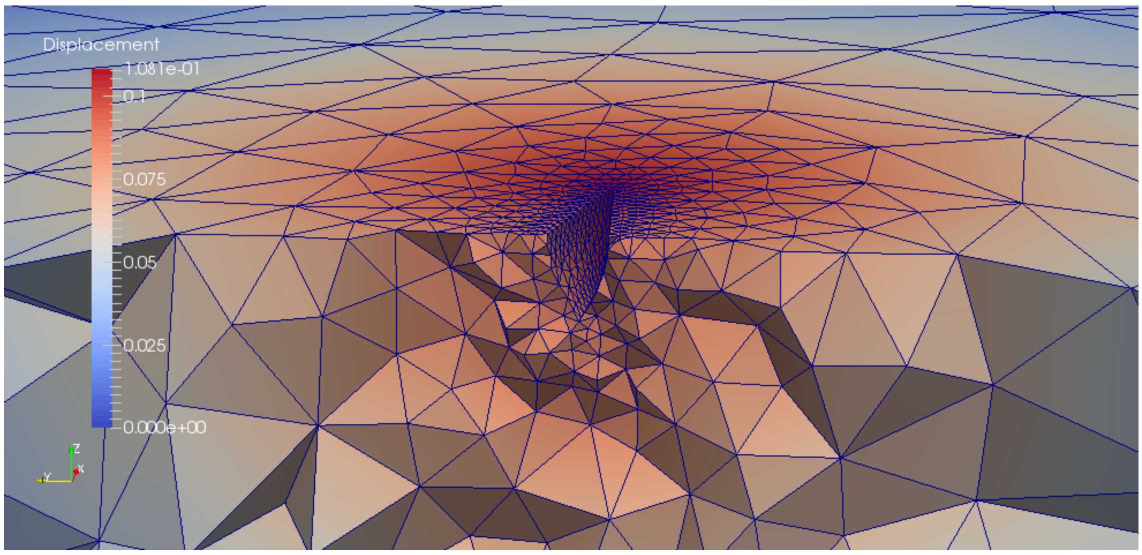


Figure 4.11: A detail representation of the final configuration of the 3D fluid domain represented in Figure 4.10.

In this case, the parameter values μ_i (with $i = 1, ..., 10$) are ten possible angles of rotation for the hull.

The CPU time required to execute the *Online phase* is very reduced, shorter (about one quarter) than the one demanded by the RIDW application on the same example (see Table 3.4). Furthermore, the percentage of error (respect to the solution computed by a *standard* IDW technique) is quite close to the same obtained with RIDW: 2.43% versus 2.42%.

For this application, the neglected energy content $E(\mathcal{Z})$ remains below the tolerance ϵ^*_{tol} by considering only two POD modes ($N = 2$).

The *Offline* execution requires a long time (about 432.314 s), but it is fully justified by the reduced memory storage and computational effort achieved by this ROM-RIDW implementation.

To be consistent with the same application considered in the previous chapters, we have applied a rigid rotation to the hull. Clearly, rotations are not the only deformations computable with the proposed algorithm. Alternatively, we can apply interface movements similar to the one considered for the wing example in Section 4.3.3.

Thus, for the sake of completeness, we show in Figures 4.12 and 4.13 the configuration obtained using the ROM-RIDW algorithm when considering a hull interface target movement given by $y(x) = 0.01x^2$.

Mesh dimensions	50 m $\times$ 25 m $\times$ 10 m
Hull length	2.50 m
Number of elements	30265
Number of nodes	7186
Number of internal nodes	3322
Number of boundary nodes	3864
Number of control points	712
$\boldsymbol{\delta}_c$ size	$[712 \times 3]$
Radius of reduction	$R_{top} = 2$ m
	$R_{bottom} = 2$ m
	$R_{right} = 2$ m
	$R_{left} = 2$ m
	$R_{front} = 2$ m
	$R_{rear} = 2$ m
	$R_{righthole} = 2$ m
	$R_{lefthole} = 2$ m
Parameter values	$\{-\frac{\pi}{20}, -\frac{\pi}{50}, -\frac{\pi}{40}, -\frac{\pi}{18}, -\frac{\pi}{80}, -\frac{\pi}{60}, -\frac{\pi}{100}, -\frac{\pi}{10}, -\frac{\pi}{5}, -\frac{\pi}{30}\}$ m
Number of POD modes selected	2
Offline CPU Time	432.314 s
Online CPU Time	1.420 s
Percentage of error respect to the displacement	2.43%

Table 4.4: Quantitative information about the deformation obtained by ROM-RIDW strategy of the 3D mesh showed in Figure 3.22.

The deformed fluid mesh in Figure 4.12 is obtained spending 1.44 s for the *Online phase*. The parameter values used for the construction of the snapshot matrix are the same used for the wing example (see Table 4.3) and the percentage error, after selecting a single POD mode, is equal to 3.41% with respect to the displacement solution obtained via the *standard* IDW interpolation technique.

4.3.5 A wing structural mesh motion via ROM-RIDW

We finally apply the ROM-RIDW method to the structural mesh of the wing in Section 4.3.3. Thus, the target configuration is represented by the boundaries of the wing when moved according to the function $y(z) = 0.01z^2$. The undeformed mesh is the one shown in Figure 4.14, while the deformed configuration obtained by the combined

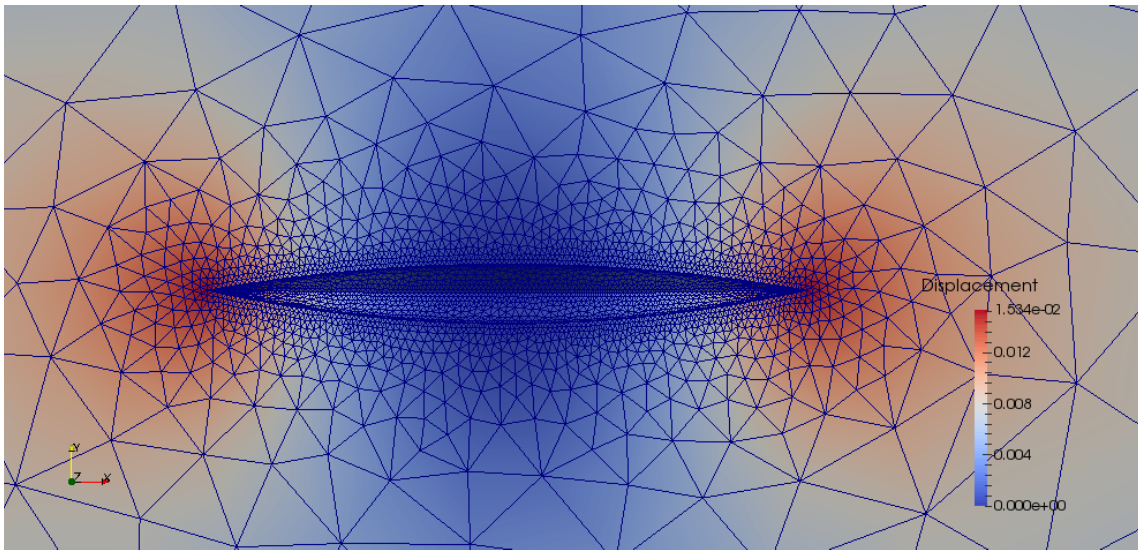


Figure 4.12: Deformed 3D mesh for the fluid domain computed by ROM-RIDW method.

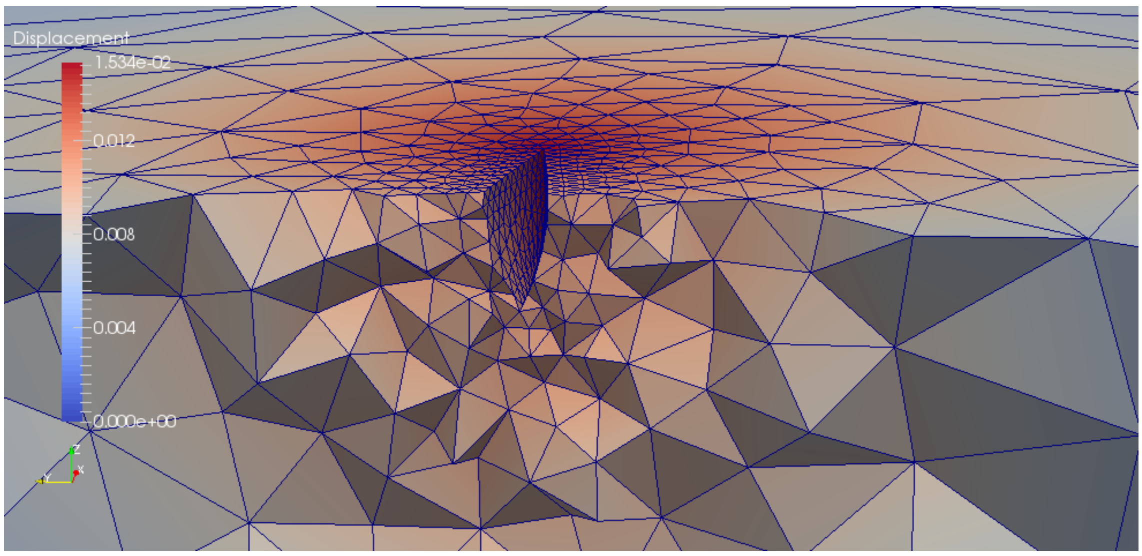


Figure 4.13: A detail view of the internal nodes of the 3D fluid domain shown in Figure 4.12.

ROM-RIDW method is provided in Figure 4.15.

We present the adopted setting and the obtained results in Table 4.5.

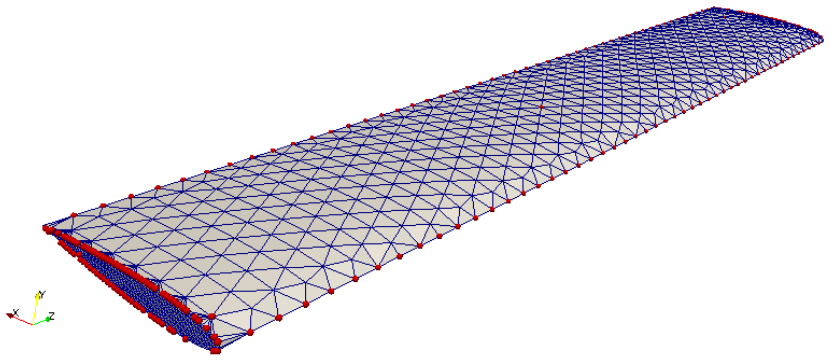


Figure 4.14: Undeformed configuration of a tridimensional wing based on a NACA0012 airfoil.

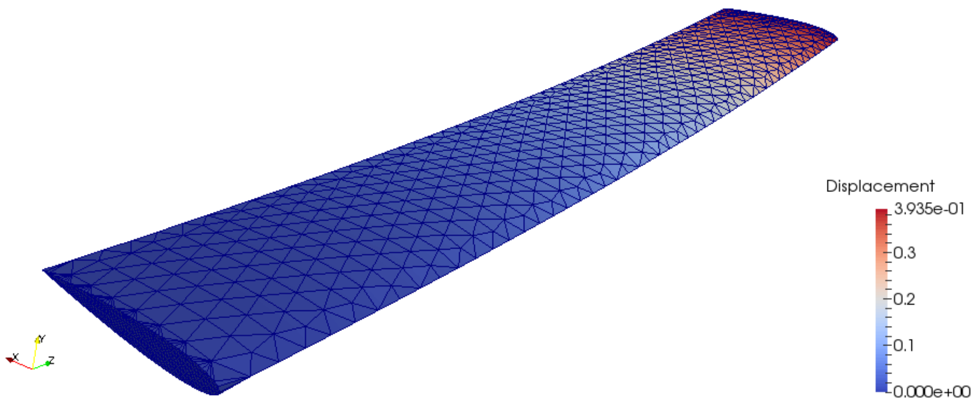


Figure 4.15: Deformed 3D wing structural mesh computed via ROM-RIDW process.

The same conclusion drawn for the examples presented in the previous sections can be repeated for this structural case. The memory effort is reduced by the RIDW application, so that each solution to be collect in the snapshot matrix is based only on 308 control points.

Moreover, the computational time to evaluate the displacement of internal nodes of the mesh, given a target deformation of the interface, is considerably reduced. In fact, with ROM, the algorithm requires 0.35 s to complete the *Online phase* providing

Wing longitudinal dimension	2π m
NACA0012 profile chord length	1.01 m
Number of elements	8850
Number of nodes	2463
Number of internal nodes	797
Number of boundary nodes	1666
Number of control points	308
$\boldsymbol{\delta}_c$ size	$[308 \times 3]$
Radius of reduction	$R_{top} = 0.7$ m
	$R_{bottom} = 0.7$ m
	$R_{right} = 2.5$ m
	$R_{left} = 2.5$ m
Parameter values	$\{0.05, 0.04, 0.08, 0.03, 1, 0.1, 1.3, 0.06, 0.07, 0.5\}$ m
Number of POD modes selected	1
Offline CPU Time	40.640 s
Online CPU Time	0.352 s
Percentage of error respect to the displacement	1.12 %

Table 4.5: Quantitative information about the mesh deformation of the 3D wing showed in Figure 4.15.

an acceptable error (1.12%) with respect to the deformed solution computed by the *standard* IDW technique.

Also for this example, the number of POD modes required to obtain accurate results is the minimum possible, being equal to one ($N = 1$).

4.4 Conclusions and perspectives of IDW in the ROM context

As noted in this chapter, a ROM construction for RIDW interpolation in the context of mesh motion problems gives considerable advantages in terms of computational and memory effort.

On the one hand, the RIDW strategy, discussed in Chapter 3, allows to compute accurate solutions used to assemble the snapshot matrix with a reduced number of control points, with a consequent lower memory effort. On the other hand, the POD-based Reduced Order Method, in conjunction with RIDW, produces an important decrease of CPU time via the *Offline-Online computational splitting*. In fact, the *Offline* CPU times are compensated by the reduced *Online* CPU times, considering that the former process takes place only once while the latter is the CPU time actually required to compute the displacement of the mesh nodes.

In Table 4.6 the estimate of the costs of the different strategies implemented with respect to the *standard* IDW formulation.

Strategy	*Online CPU time*	*Error*	*Number of control points*
RIDW	40 %	3.1 %	20 %
ROM-IDW	8.5%	0 %	100 %
ROM-RIDW	0.6%	3.2 %	20 %

Table 4.6: Summary table of comparison between the estimated costs of the different startegies implemented with respect to the *standard* IDW formulation.

The Table 4.6 highlights the ability of the ROM-RIDW process to exploit the advantages of the RIDW strategy (reduced number of control points) and of the *Offline-Online* computational splitting procedure (CPU time considerably reduced), containing, at the same time, the error on the displacement of the nodes.

The capabilities of the ROM-RIDW strategy becomes particularly evident in the last examples, when dealing with more complex 3D configurations. Further improvements are clearly possible, for instance, with reference to the choice of the position of the control points.

Conclusions

The main goal of the present work has been the development of an Inverse Distance Weighting (IDW) within a parametrization strategy for efficient mesh motion applications.

The advantage of using shape parametrization technique for this kind of problems is due to the possibility of representing shape deformations via low dimensional spaces (control points). In particular, the IDW technique shows good properties in terms of mesh quality conservation and ease of implementation.

Despite these important qualities, its computational cost and memory requirement are often prohibitive, particularly for complex meshes as the ones characterizing aeronautical and naval applications due to the large number of involved nodes.

For these reasons, we have developed and implemented a new toolbox which improves the performances of a *standard* IDW technique so that it can become useful also for problems too demanding so far.

After a brief introduction about the state of art of the shape parametrization strategies and the motivation behind this work, in Chapter 2 we have detailed the original formulation of the IDW technique, highlighting in particular the corresponding benefits and drawbacks.

In Chapter 3 we have proposed two different strategies of *geometrical reduction* in order to reduce the number of the control points involved in the solution computation and to exploit the IDW advantages. The first new approach is based on a geometric user-driven procedure to reduce the control points used to compute the internal node displacement, by user-definable circular annuli. The second procedure, instead, allows to use IDW technique in '*dimensional cascade*', i.e., to apply IDW technique in three different phases (1D, 2D and 3D) to compute, first of all, the movement of the interface starting from the displacement of some specific nodes, and then the deformation of the whole mesh.

Since the geometrical reduction has produced good results in terms of memory effort (by guaranteeing a reduction up to one third with respect to the original formulation), but it is still characterized by excessive CPU times, in Chapter 4 we have focused on the construction of a *Reduced Order Method* to exploit the advantage of a possible splitting between *Offline* and *Online* phases. Inspired by a Proper Orthogonal Decomposition, in the *Offline* stage we have assembled a matrix of solutions (snapshots) for several interface movements to obtain a reduced solution space. Thus, by a projection on this space, in the *Online* stage we have extracted the solution for new arbitrary values of the parameters.

This procedure has reduced the computation time of about an order of magnitude, so that the IDW technique turns out to be now available also for real-time and shape optimization applications, in aeronautical and naval fields that, typically, require almost instantaneous solutions.

Each development of the original IDW formulation has been successfully tested on different configurations, starting from simple benchmark test cases to complex meshes typical of aeronautical and naval settings.

Concerning the future developments of this work, an extensive testing of the entire mesh motion toolbox on a real CFD application could fully certificate the capabilities of the new IDW formulation. Moreover, since the code is optimized to work for the *serial computing*, in order to further improve the algorithm performances we are thinking to reorganize it for the *parallel computing*. Also the geometric reduction of the number of the control points could be more deeply investigated, driven, for instance, by the physics of the problem under investigation. A possibility could be realizing a reduction process based on the gradient of the interface displacement, rather than on the main direction of the fluid. Furthermore, ROMs are more and more increasing their potentialities in scientific computing and new challenges arise each day, pushing the researcher to innovative ideas to solve complex problems. Reduced order methods are spreading and very soon it will be possible to associate their efficient computational properties also with parametric interfaces management algorithms like the one presented in this thesis.

Bibliography

[1] I. H. Abbott and A. Von Doenhoff. *Theory of wing sections: including a summary of airfoil data*. Dover Publications, 1959.

[2] J. Ahrens, B. Gavenci, and C. Law. *ParaView: An End-User Tool for Large Data Visualization, Visualization Handbook*. Elsevier, 2005.

[3] N. Ardjmandpour, C. C. Pain, F. Fang, A. G. Buchan, J. Singer, M. A. X. X. Player, I. M. Navon, and J. Carter. Reduced order borehole induction modelling. *International Journal of Computational Fluid Dynamics*, 28(3–4):140–157, 2014.

[4] N. Aubry. On the hidden beauty of the proper orthogonal decomposition. *Theoretical and Computational Fluid Dynamics*, 2:339–352, 1991.

[5] N. Aubry, W. Lian, and E. Titi. Preserving symmetries in the proper orthogonal decomposition. *SIAM Journal on Scientific Computing*, 14:483–505, 1993.

[6] K. Baker. Singular Value Decomposition tutorial. Available at `https://www.ling.ohio-state.edu`, 2013.

[7] F. Ballarin. *Reduced–order models for patient–specific haemodynamics of coronary artery bypass grafts*. PhD thesis, Politecnico di Milano, Italy, 2015.

[8] F. Ballarin, A. Manzoni, G. Rozza, and S. Salsa. Shape optimization by Free-Form Deformation: existence results and numerical solution for Stokes flows. *Journal of Scientific Computing*, 60(3):537–563, 2014.

[9] S. Barnes. A technique for maximizing details in numerical weather map analysis. *Journal of Applied Meteorology and Climatology*, 3:396–409, 1964.

[10] G. Batchelor. *An Introduction to Fluid Dynamics*. Cambridge Mathematical Library, 1973.

[11] U. Baur, P. Benner, B. Haasdonk, C. Himpe, I. Maier, and M. Ohlberger. Comparison of methods for parametric model order reduction of instationary problems. Preprint MPIMD/15-01, Max Planck Institute Magdeburg, 2015.

[12] A. Beckert and H. Wendland. Multivariate interpolation for fluid-structure-interaction problems using Radial Basis Functions. *Aerospace Science and Technology*, 5:125–134, 2001.

[13] P. Benner, S. Gugercin, and K. Willcox. A Survey of Projection-Based Model Reduction Methods for Parametric Dynamical Systems. *SIAM review*, 57(4):483–531, 2015.

[14] G. Berkooz, P. Holmes, and J. L. Lumley. The Proper Orthogonal Decomposition in the analysis of turbulent flows. *Annual Reviews Fluid Mechanics*, 25:539–575, 1993.

[15] A. Buffa, Y. Maday, A. T. Patera, C. Prud'homme, and G. Turinici. A priori convergence of the greedy algorithm for the parametrized reduced basis method. *ESAIM: Mathematical Modelling and Numerical Analysis*, 46(03):595–603, 2012.

[16] M. Buffoni and K. Willcox. Projection-based model reduction for reacting flows. In *40th Fluid Dynamics Conference and Exhibit*. AIAA, 2010.

[17] M. D. Buhmann. *Radial Basis Functions*, volume 12 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, UK, 2003.

[18] T. Bui-Thanh, M. Damodaran, and K. Willcox. Aerodynamic data reconstruction and inverse design using proper orthogonal decomposition. *AIAA Journal*, 42(8):1505–1516, 2004.

[19] T. Bui-Thanh, K. Willcox, O. Ghattas, and B. van Bloemen Waanders. Goal-oriented, model-constrained optimization for reduction of large-scale systems. *Journal of Computational Physics*, 224(2):880–896, 2007.

[20] J. Burkardt, M. Gunzburger, and H.-C. Lee. POD and CVT-based reduced-order modeling of Navier–Stokes flows. *Computer methods in applied mechanics and engineering*, 196:337–355, 2006.

[21] K. Carlberg and C. Farhat. A low-cost, goal-oriented 'compact proper orthogonal decomposition' basis for model reduction of static system. *International Journal for Numerical Methods in Engineering*, 86(3):381–402, 2011.

[22] J. R. Cebral and R. Lohner. Conservative load projection and tracking for fluid-structure problems. *AIAA Journal*, 35(4):687–692, 1997.

[23] D. Chen, R. Taki, and W. Fan. Using a genetic algorithm to optimize the Inverse Distance Weight (IDW) Interpolation method. Available at `http://www.giscience.org`, Queen's University, Kingston, Canada 2012.

[24] F. Chinesta, A. Huerta, G. Rozza, and K. Willcox. Model Order Reduction. *Encyclopedia of Computational Mechanics*, 2016.

[25] A. K. Cline and I. S. Dhillon. *Computation of the Singular Value Decomposition*. CRC Press, 2006.

[26] G. Cressman. An operational objective analysis system. *Monthly Weather Review*, 87:367–374, 1959.

[27] A. de Boer, A. H. van Zuijlen, and H. Bijl. A higher-order panel method based on B-splines. In *European Conference on Computational Fluid Dynamics*. ECCOMAS CFD, 2006.

[28] S. Deparis, D. Forti, and A. Quarteroni. A rescaled localized radial basis function interpolation on non-Cartesian and non-conforming grids. *SIAM Journal on Scientific Computing*, 36(6), 2014.

[29] M. Diez, E. F. Campana, and F. Stern. Conservative load projection and tracking for fluid-structure problems. *Computer Methods in Applied Mechanics and Engineering*, 283:1525–1544, 2015.

[30] R. P. Dwight. Robust mesh deformation using the linear elasticity equations. In H. Deconinck and E. Dick, editors, *Proceedings of the Fourth International Conference on Computational Fluid Dynamics*, pages 401–406. Springer, 2006.

[31] D. Forti. Comparison of shape parametrization techniques for Fluid-Structure Interaction problems. Master's thesis, Politecnico di Milano, Italy, 2012.

[32] D. Forti and G. Rozza. Efficient geometrical parametrization techniques of interfaces for reduced-order modelling: application to fluid–structure interaction coupling problems. *International Journal of Computational Fluid Dynamics*, 28(3–4):158–169, 2014.

[33] C. Günther. Reduced Basis Method for the Shape Optimization of Racing Car Components. Master's thesis, Ecole Polyechnique Fédérale de Lausanne, 2008.

[34] B. Haasdonk. Reduced Basis Methods: a tutorial introduction for stationary coercive parametrized PDEs. Universität Stuttgart, Institut für Angewandte Analysis und Numerische Simulation, 2013.

[35] F. Hanzer. Spatial interpolation of scattered geoscientific data. Available at `http://www.uni-graz.at`, 2012.

[36] B. T. Helenbrook. Mesh deformation using the biharmonic operator. *International Journal for Numerical Methods in Engineering*, 56(7):1007–1021, 2003.

[37] J. S. Hesthaven, G. Rozza, and B. Stamm. *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*. MS&A series. Springer International Publishing, 2016.

[38] B. S. Kirk, J. W. Peterson, R. H. Stogner, and G. F. Carey. `libMesh`: A C++ Library for Parallel Adaptive Mesh Refinement/Coarsening Simulations. *Engineering with Computers*, 22(3–4):237–254, 2006.

[39] G. Kosec. *Local meshless method for multi-phase thermo-fluid problems*. PhD thesis, University of Nova Gorica, 2011.

[40] G. Kosec, M. Depolli, A. Rashkovska, and R. Trobec. Super linear speedup in a local parallel meshless solution of thermo-fluid problems. In K. Bathe and B. Topping, editors, *Computers & Structures*, volume 133, pages 30–38. Elsevier, 2014.

[41] G. Kosec and B. Šarler. Solution of a low Prandtl number natural convection benchmark by a local meshless method. *International Journal of Numerical Methods for Heat & Fluid Flow*, 23(1):189–204, 2013.

[42] A. Koshakji. Free Form Deformation techniques for 3D shape optimization problems. Master's thesis, Politecnico di Milano, Italy, 2010.

[43] A. Koshakji, A. Quarteroni, and G. Rozza. Free Form Deformation techniques applied to 3D shape optimization problems. *Communication in Applied and Industrial Mathematics*, 4, 2013.

[44] K. Kunisch and S. Volkwein. Galerkin proper orthogonal decomposition methods for a general equation in fluid dynamics. *SIAM Journal on Numerical Analysis*, 40(2):492–515, 2002.

[45] K. Kunisch and S. Volkwein. Proper orthogonal decomposition for optimality systems. *Mathematical Modelling and Numerical Analysis*, 42:1–23, 2008.

[46] T. Lassila, A. Manzoni, A. Quarteroni, and G. Rozza. Model order reduction in fluid dynamics: challenges and perspectives. In A. Quarteroni and G. Rozza, editors, *Reduced Order Methods for Modeling and Computational Reduction*, volume 9, pages 235–274. Springer, MS&A Series, 2013.

[47] T. Lassila, A. Manzoni, A. Quarteroni, and G. Rozza. A reduced computational and geometrical framework for inverse problems in haemodynamics. *International Journal for Numerical Methods in Biomedical Engineering*, 29(7):741–776, 2013.

[48] T. Lassila, A. Quarteroni, and G. Rozza. A reduced basis model with parametric coupling for fluid-structure interaction problem. *SIAM Journal on Scientific Computing*, 34(2):A1187–A1213, 2012.

[49] T. Lassila and G. Rozza. Model reduction of steady fluid-structure interaction problems with free-form deformations and reduced basis methods. In *Proceedings of the 10th Finnish Mechanics Days*, pages 454–465, 2009.

[50] T. Lassila and G. Rozza. Parametric free-form shape design with PDE models and reduced basis method. *Computer Methods in Applied Mechanics and Engineering*, 199(23-24):1583–1592, 2010.

[51] M. Lombardi. *Numerical simulation of a sailing boat: free surface, fluid-structure interaction and shape optimization*. PhD thesis, Ecole Polyechnique Fédérale de Lausanne, 2012.

[52] M. Lombardi, N. Parolini, A. Quarteroni, and G. Rozza. Numerical simulation of sailing boats: dynamics, FSI, and shape optimization. In G. B. A. Frediani, editor, *Variational Analysis and Aerospace Engineering: Mathematical Challenges for Aerospace Design*, volume 66, pages 339–377. Springer US, 2012.

[53] G. Lu and D. Wong. An adaptive inverse-distance weighting spatial interpolation technique. *Computers & Geosciences*, 34:1044–1055, 2008.

[54] Y. Maday, A. T. Patera, and G. Turinici. A priori convergence theory for reduced-basis approximations of single-parameter elliptic partial differential equations. *Journal of Scientific Computing*, 17(1-4):437–446, 2002.

[55] A. Manzoni. *Reduced models for optimal control, shape optimization and inverse problems in haemodynamics*. PhD thesis, Ecole Polyechnique Fédérale de Lausanne, 2012.

[56] A. Manzoni, A. Quarteroni, and G. Rozza. Shape optimization for viscous flows by reduced basis methods and free-form deformation. *International Journal for Numerical Methods in Fluids*, 70(5):646–670, 2012.

[57] MATLAB. *version 7.10.0 (R2015b)*. The MathWorks Inc., Natick, Massachusetts, 2015.

[58] A. M. Morris, C. B. Allen, and T. C. S. Rendall. CFD-based optimization of airfoils using radial basis functions for domain element parametrization and mesh deformation. *International Journal for Numerical Methods in Fluids*, 58:827–860, 2008.

[59] S. Perotto. Hierarchical model (Hi-Mod) reduction in non-rectilinear domains. *Domain Decomposition Methods in Science and Engineering*, 58:477–485, 2014.

[60] S. Perotto and A. Veneziani. Coupled model and grid adaptivity in hierarchical reduction of elliptic problems. *Journal of Scientific Computing*, 60(3):505–536, 2014.

[61] S. Perotto and A. Zilio. Hierarchical model reduction: three different approaches. *Numerical Mathematics and Advanced Applications*, pages 851–859, 2013.

[62] A. Quarteroni. *Numerical models for differential problems.* MS&A series. Springer-Verlag Italia, Milan, 2009.

[63] A. Quarteroni, G. Rozza, and A. Manzoni. Certified reduced basis approximation for parametrized partial differential equations and applications. *Journal of Mathematics in Industry*, 1(3), 2011.

[64] A. Quarteroni, R. Sacco, and F. Saleri. *Numerical Mathematics.* Series 37. Springer-Verlag, New York, 2000.

[65] B. Raghavan, P. Breitkopf, Y. Tourbier, and P. Villon. Towards a space reduction approach for efficient structural shape optimization. *Structural and Multidisciplinary Optimization*, 48:987–1000, 2013.

[66] S. Ravindran. A reduced-order approach for optimal control of fluids using proper orthogonal decomposition. *International Journal for Numerical Methods in Fluids*, 34(5):425–448, 2000.

[67] M. Ripepi. *Model Order Reduction for Computational Aeroelasticity.* PhD thesis, Politecnico di Milano, Italy, 2015.

[68] G. Romanelli. *Computational Aeroservoelasticity of free-flying deformable aircraft.* PhD thesis, Politecnico di Milano, Italy, 2012.

[69] G. Rozza. Reduced basis approximation and error bounds for potential flows in parametrized geometries. *Communications in Computational Physics*, 9:1–48, 2011.

[70] G. Rozza. Fundamentals of Reduced Basis Method for problems governed by parametrized PDEs and applications. In *Separated representations and PGD-based model reduction: fundamentals and applications*, volume 554. Springer, 2014.

[71] G. Rozza, D. B. P. Huynh, N. C. Nguyen, and A. T. Patera. Real-time reliable simulation of heat transfer phenomena. In *Proceedings of HT2009: 2009 ASME Summer Heat Transfer Conference*, 2009.

[72] G. Rozza, D. B. P. Huynh, and A. T. Patera. Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations. *Archives of Computational Methods in Engineering*, 15(3):229–275, 2008.

[73] F. Salmoiraghi. Reduced-order models for potential flows past parametrized NACA airfoils based on an isogeometric Boundary Element Method. Master's thesis, Politecnico di Milano, Italy, 2014.

[74] F. Salmoiraghi, F. Ballarin, G. Corsi, A. Mola, M. Tezzele, and G. Rozza. Advances in geometrical parametrization and reduced order models and methods for Computational Fluid Dynamics problems in applied sciences and engineering: overview and perspectives. In *ECCOMAS Congress 2016 – VII European Congress on Computational Methods in Applied Sciences and Engineering*, 2016.

[75] J. A. Samareh. Aerodynamic shape optimization based on Free-Form Deformation. In *Proceedings of the 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. AIAA, 2004.

[76] T. W. Sederberg and S. R. Parry. Free-Form Deformation of solid geometric models. In *Proceedings of SIGGRAPH - Special Interest Group on GRAPHics and Interactive Techniques*, pages 151–159. SIGGRAPH, 1986.

[77] D. Shepard. A two-dimensional interpolation function for irregularly-space data. In *Proceedings-1968 ACM National Conference*, pages 517–524. ACM, 1968.

[78] W. A. Silva, P. Chwalowski, and B. Perry III. Evaluation of linear, inviscid, viscous, and reduced-order modelling aeroelastic solutions of the AGARD 445.6 wing using root locus analysis. *International Journal of Computational Fluid Dynamics*, 28(3–4):122–139, 2014.

[79] P. Thévenaz, T. Blu, and M. Unser. Interpolation revisited. *IEEE Transactions on Medical Imaging*, 19(7):739–758, 2000.

[80] R. Trobec, G. Kosec, M. Šterk, and B. Šarler. Comparison of local weak and strong form meshless methods for 2-D diffusion equation. *Engineering Analysis with Boundary Elements*, 36(3):310–321, 2012.

[81] K. Veroy, C. Prud'homme, D. Rovas, and A. Patera. A posteriori error bounds for reduced-basis approximation of parametrized noncoercive and nonlinear elliptic partial differential equations. In *Proceedings of the 16th AIAA computational fluid dynamics conference*, volume 3847, 2003.

[82] S. Volkwein. Proper orthogonal decomposition: Theory and reduced-order modeling. *Lecture Notes, University of Konstanz*, 2012.

[83] J. A. Witteveen. Explicit and robust Inverse Distance Weighting mesh deformation for CFD. In *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*. AIAA, 2010.

[84] J. A. Witteveen and H. Bijl. Explicit mesh deformation using Inverse Distance Weighting interpolation. In *19th AIAA Computational Fluid Dynamics*. AIAA, 2009.