National Technical University of Athens
School of Mechanical Engineering
Fluids Sector
Parallel CFD & Optimization Unit

Advancements in an Implicit Grid Displacement Method based on Rigid Body Motion Theory for use in Aerodynamic Shape Optimization

Master Thesis

by

Charalampos Spyropoulos
A Thesis submitted in partial fulfillment of the Joint Postgraduate Program
"Computational Mechanics"

Supervisor: Kyriakos C. Giannakoglou,Professor NTUA

Athens, February 2025

National Technical University of Athens
School of Mechanical Engineering
Fluids Sector
Parallel CFD & Optimization Unit

Advancements in an Implicit Grid Displacement Method based on Rigid Body Motion Theory for use in Aerodynamic Shape Optimization

Charalampos Spyropoulos

Abstract

This thesis presents the development, programming, and evaluation of the Distance-Weighted Rigid Body Motion (DWRBM) method, a mesh displacement technique designed to preserve mesh quality during large geometric deformations. By integrating rigid body motion principles with a distance-based weighting mechanism, the method effectively maintains mesh quality near boundaries and minimizes the risk of inverted cells, even in complex cases.

Building upon the non-linear and linearized methods introduced in a previous diploma thesis[5], this work enhances their performance through the application of the distance-based weighting technique. A GMRES solver was developed in C++ utilizing a modified Compressed Sparse Row (mCSR) storage scheme, to solve the systems, arising from the optimization problem. Two preconditioning approaches, diagonal and Gauss-Seidel, were implemented and assessed. The algorithm was rigorously tested across various 2D and 3D cases.

The DWRBM method has demonstrated its versatility and effectiveness in mesh deformation, showing considerable promise for integration into modern CFD workflows and aerodynamic optimization processes. Compared to previous implementations [5], the method achieved faster performance, improved mesh quality, and facilitated significantly larger displacements in the same test cases.

More specifically, the use of the weighting technique rendered the displacement feasible without requiring sub-steps in non-extreme cases. Additionally, the combination of weighting and linearization through the small-angle approximation significantly improved the algorithm's speed and computational efficiency. Moreover, the preconditioning in the GMRES solver proved highly effective, greatly accelerating convergence.

Keywords:
Computational Fluid Dynamics (CFD), mesh displacement, Unstructured meshes

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Μηχανολόγων Μηχανικών
Τομέας Ρευστών
Μονάδα Παράλληλης Υπολογιστικής Ρευστοδυναμικής & Βελτιστοποίησης

Αναβαθμίσεις σε μια Μέθοδο Μετατόπισης Πλέγματος βασισμένη στη Θεωρία Κίνησης Απαραμόρφωτου Σώματος για χρήση στην Αεροδυναμική Βελτιστοποίηση Μορφής

Χαράλαμπος Σπυρόπουλος

Περίληψη

Αυτή η διατριβή παρουσιάζει την ανάπτυξη, τον προγραμματισμό και την αξιολόγηση της μεθόδου Κίνησης Απαραμόρφωτου Σώματος, ενισχυμένης με βάρη ανάλογα της απόστασης (Distance Weighted Rigid Body Motion, DWRBM), μιας τεχνικής μετατόπισης πλέγματος που έχει σχεδιαστεί για τη διατήρηση της ποιότητας του πλέγματος κατά τη διάρκεια μεγάλων γεωμετρικών παραμορφώσεων. Με την ενσωμάτωση των αρχών κίνησης απαραμόρφωτου σώματος και ενός μηχανισμού βαρύτητας με βάση την απόσταση, η μέθοδος διατηρεί αποτελεσματικά την ποιότητα του πλέγματος κοντά στα όρια και ελαχιστοποιεί τον κίνδυνο αναστροφής των κελιών.

Βασιζόμενη στις μη γραμμικές και γραμμικοποιημένες μεθόδους που παρουσιάστηκαν σε προηγούμενη διπλωματική εργασία [5], αυτή η μελέτη βελτιώνει την απόδοσή τους μέσω της εφαρμογής της τεχνικής βαρύτητας με βάση την απόσταση.Ένας επιλύτης GMRES αναπτύχθηκε σε C++ χρησιμοποιώντας ένα τροποποιημένο σχήμα αποθήκευσης (modified Compressed Sparse Row, mCSR), για την επίλυση των συστημάτων που προκύπτουν από το πρόβλημα βελτιστοποίησης. Επίσης αναπτύχθηκαν και δοκιμάστηκαν δύο τεχνικές προσταθεροποίησης, η διαγώνια και η Gauss-Seidel. Τέλος ο αλγόριθμος δοκιμάστηκε σε διάφορες δισδιάστατες (2Δ) και τρισδιάστατες (3Δ) εφαρμογές.

Η μέθοδος DWRBM έδειξε ευελιξία και αποτελεσματικότητά στην παραμόρφωση πλέγματος, δείχνοντας σημαντική προοπτική για ενσωμάτωση σε σύγχρονες ροές εργασίας Υπολογιστικής Ρευστοδυναμικής και αεροδυναμικής βελτιστοποίησης. Συγκριτικά με προηγούμενες υλοποιήσεις[5], η μέθοδος πέτυχε ταχύτερη απόδοση, βελτιωμένη ποιότητα πλέγματος και επέτρεψε σημαντικά μεγαλύτερες μετατοπίσεις στις ίδιες περιπτώσεις δοκιμών.

Πιο συγκεκριμένα, η χρήση της τεχνικής βαρύτητας κατέστησε εφικτές τις μετατοπίσεις χωρίς την ανάγκη υπο-βημάτων για μη ακραίες περιπτώσεις. Επιπλέον, ο συνδυασμός της τεχνικής βαρύτητας και της γραμμικοποίησης μέσω της προσέγγισης μικρών γωνιών βελτίωσε σημαντικά την ταχύτητα και την υπολογιστική αποδοτικότητα του αλγορίθμου. Πέραν αυτών, η προσταθεροποίηση στον GMRES αποδείχθηκε εξαιρετικά αποτελεσματική, επιταχύνοντας σημαντικά τη σύγκλιση.

Λέξεις-κλειδιά:
Υπολογιστική Ρευστοδυναμική (ΥΡΔ), Μετατόπιση πλέγματος, Μη Δομημένα Πλέγματα

Acronyms

2D	Two-Dimensional.
3D	Three-Dimensional.
CFD	Computational Fluid Dynamics.
DoF	Degrees of Freedom.
DWRBM	Distance-Weighted Rigid Body Motion.
EAs	Evolutionary Algorithms.
FSI	Fluid Structure Interaction.
GMRES	Generalized Minimal Residual Method.
IDW	Inverted Distance Weighting.
mCSR	modified Compressed Sparse Row.
N-GMRES	Newton Generalized Minimal Residual Method.
PDE	Partial Differential Equation.
qN-GMRES	Quasi-Newton Generalized Minimal Residual Method.
RBF	Radial Basis Functions.
RBM	Rigid Body Motion.
w.r.t.	with respect to.

Contents

Acknowlegdements

First of all, I would like to express my gratitude to my advisor, Prof. Kyriakos Giannakoglou, for the invaluable knowledge and mindset he has shared with me, both through the courses he taught and our discussions on the technical aspects of this thesis. This work would not have been possible without his guidance and the trust he placed in me throughout the past year. His support, both technical and theoretical, has been crucial in my development as an engineer and researcher.

I would also like to extend my sincere thanks to Dr. Varvara Asouti for her constant support whenever I needed assistance, and to Vaggelis Christianos, whose work I continued, for his availability in the early stages of the project and his help in explaining key technical concepts.

Furthermore, I want to express my gratitude to my parents and brothers for their unwavering support and belief in me throughout this journey.

I would also like to thank my friends from the Computational Mechanics master's program—Kyriakos, Kostas, and Dimitris as well as Vaggelis D. and Natalia, for their friendship and encouragement.

Last but certainly not least, I want to thank Vaggelis F. and Angeliki for being role models for me and motivating me to become a better version of myself and pursue my dreams.

"The highest virtue is not to be free, but to fight for freedom."

Nikos Kazantzakis

Chapter 1

Introduction

1.1 CFD and meshes

Nowadays, the continuously search for climate-friendly transportation has led engineers to explore in-depth the design space and achieve more efficient designs in order to reduce emissions. Aerospace engineers working in the field of fluid dynamics (aerodynamic designers, propulsion engineers etc) are trying to optimize their component's designs in order to achieve the above mentioned objectives. The basic tool to do so is the Computational Fluid Dynamics (CFD) software. Solving numerically the governing equations (Navier Stokes Equations for example) help engineers to evaluate the aerodynamic performance of the air-vehicle. In order for these equations to be solved on a computer they have to be discretized. This numerical discretization also implies the spatial discretization of the physical space i.e. mesh generation. This is the so called numerical grid or mesh. The accuracy and reliability of the CFD simulations are strongly dependent on the mesh, both its resolution and its quality. On the other hand, a low quality mesh may diverge the simulation and do not even produce results. So, a high quality mesh is necessary in order to carry out a CFD simulation.

These days, as CFD solvers are reliable, researchers have focused in the development of automatic aerodynamic optimization algorithms. These algorithms significantly reduce both the design process duration and the cost by altering the geometry in order to optimize its aerodynamic performance. This change of geometry implies the need for a new computational mesh. This can be done either by remeshing or by adapting the existing mesh to the new geometry. The first approach, requires both time and human intervention, destroying the automatic feature of these algorithms. So, mesh adaptation to the new geometry is deemed necessary.

1.2 Aerodynamic Optimization

As already mentioned, there is significant technological interest in the ability to automatically design optimal aerodynamic bodies in order to enhance their performance. For example, the design process may aim at minimizing the drag of a body in external flow, maximizing the lift of a body, minimizing total pressure losses in internal flow, or maximizing the efficiency of flow in a compressor or turbine blade. These problems are known as shape optimization problems. To meet this need, optimization methods have been developed, aiming at minimizing or maximizing one or more functions, known as cost, fitness or objective functions.

Optimization methods can be divided into two main categories, stochastic and deterministic methods[8]. Stochastic optimization methods are characterized by seeking the optimal solution in a random or organized random way. These algorithms are general, meaning they can easily be applied to solve different problems, and they can identify the global optimum of an objective function, regardless of their initialization. However, they have the disadvantage of requiring the evaluation of many different solutions before converging to the optimum, which makes them slow. A characteristic example of stochastic algorithms is Evolutionary Algorithms (EAs)[8].

On the other hand, deterministic optimization algorithms are based on the generalized concept of the derivative of the objective function to find the optimal solution, that is, the solution that minimizes or maximizes the value of any given objective function. Developing a deterministic method requires a significant time investment and further development and programming to apply it to similar problems. For example, changing the objective function requires changes to the implementation code of the method. The advantage of deterministic methods is that they can converge quickly to the optimal solution; however, there is a risk of finding a local optimum, depending on the initialization point. The main issue with these methods is the need of the derivatives of the objective function with respect to the design variables, also known as sensitivity derivatives. Some notable methods for computing these derivatives are:

- *Finite Differences*[8]
- *Complex Variable Method*[8]
- *Direct Differentiation*[8]
- *Algorithmic Differentiation*[8]
- *Adjoint Method*[8], [10]

Of the above methods for computing sensitivity derivatives, the most widely used in aerodynamic design problems is the adjoint method [8], [10]. A major advantage of this method is that the computational cost is independent of the number of design variables, a crucial factor for solving modern aerodynamic optimization problems where the number of design variables may be of the order of thousands, exceeding by far the number of desired responses (objectives and constraints), of which the adjoint method cost scales. In problems with a high number of design variables, using stochastic methods is unfeasible or even impossible because their computational cost is many times higher than that of deterministic methods. However, implementing a deterministic optimization algorithm using the adjoint method requires deriving the adjoint equations and boundary conditions of the problem, and program the corresponding software.

In all the optimization methods mentioned above, evaluating each solution requires the parameterization of the geometry to identify the design variables, and then, the generation of a computational mesh that will adapt precisely to each new geometry. Due to the cost of generating the mesh, the optimization process often starts with an initial mesh around the starting geometry, which is not regenerated after each geometry change, since doing so would incur a high computational cost as well as human intervention, but is adapted to each new shape resulting from the optimization method used. For this reason, the mesh deformation method is a key component of the aerodynamic optimization process.

1.3 Mesh Deformation

Mesh deformation, displacement, or adaptation in CFD, is a fundamental part of solving problems involving moving bodies or bodies with changing geometry. Examples of such problems include unsteady problems with relative motion of two or more bodies, such as the movement of control surfaces or aeroelasticity computations, which involve the deformation of wings according to applied aerodynamic loads, as well as aerodynamic design applications, where the geometry changes after each optimization cycle.

The change in the geometry of a given problem requires the adaptation of the computational mesh to the new geometric boundary, so that the flow can be solved again. Creating a new mesh around the geometry would be an expensive option, as it requires significant computational time. For this reason, the displacement of the existing mesh based on the deformation of the problem's geometry is chosen. This displacement is done at each time step (for unsteady problems) or after each optimization cycle (for aerodynamic design problems). After each displacement, the previous flow solution can be used as the initial value for the new solution (provided the mesh topology is maintained), allowing for faster convergence. Therefore, the development of a mesh displacement method is crucial, one that maintains the quality of the mesh at each step while performing this adaptation in minimal computational time.

The goal of a mesh displacement method is to propagate the displacement of the surface (boundary) mesh into the interior of the computational domain, so that all interior nodes are displaced and the mesh adapts to the new geometry. This adaptation can be done either by remeshing the regions of the new computational domain that are not properly parameterized by the existing mesh, or by deforming the existing mesh. While the first method ensures better mesh quality, the second method is preferred as it preserves the topology of the mesh. Using an appropriate deformation method, a mesh can maintain its quality at acceptable levels, even for large deformations of the geometry.

In the literature, a wide range of methods has emerged for the deformation (or displacement or adaptation) of structured and unstructured computational meshes. These various mesh deformation techniques can be categorized into methods based on partial differential equations (PDE), physical analogy methods, algebraic methods, and combinations thereof.[14]

PDE-based methods compute the displacement of the mesh by solving differential equations with appropriate boundary conditions. Two widely used operators in this context are the Laplacian operator[4] and the Biharmonic operator[9]. The Biharmonic operator has the advantage over the Laplacian operator of better preserving mesh orthogonality near the boundaries. These methods are relatively straightforward to implement; however, they have limited flexibility in mesh displacement and often require many smaller incremental steps to reach the final deformed mesh. Consequently, PDE methods are primarily used for problems involving small deformations.

The second category of mesh deformation methods consists of algebraic methods. These methods determine the displacement of each mesh node using algebraic relations that depend on the displacement of boundary nodes and the relative position of the node to be moved. Algebraic methods have seen substantial development in recent years due to their high speed in computing mesh displacements. However, they typically do not account for the mesh topology, i.e., the connectivity of the mesh nodes. As a result, they can be applied easily and reliably to arbitrary types of meshes, whether structured or unstructured, with polyhedral elements or cells with high aspect ratios typical of viscous flow meshes.

One such method is interpolation using radial basis functions (RBF)[6], which distributes the displacement of the boundary nodes to the interior nodes based on their distance from specific centers. The RBF method is straightforward to apply and produces high-quality meshes, maintaining satisfactory orthogonality of the cells near boundaries. However, the direct application of this method to large 3D problems is computationally expensive. This cost can be reduced by using appropriate preconditioners.

Another algebraic method is the inverse distance weighting (IDW) method[16], which ensures that the mesh near the boundaries remains relatively rigid, while the mesh farther away is more flexible and deforms easily. This method effectively preserves mesh quality near the boundaries.

The last category is that of physical analogy methods. Physical analogy methods are the most commonly employed mesh deformation techniques. Two of the most popular and reliable techniques in this category are the spring analogy method and the elastic method. In the spring analogy method[3], the entire computational domain is modeled as a system of interconnected linear springs, which are connected at the nodes of the mesh. The deformation of the mesh is determined by solving the equilibrium equations of the overall system after the boundary nodes are displaced according to the geometry's deformation. While the spring analogy method is easy to implement, it exhibits stability issues in cases of large displacements and dense meshes, often leading to the appearance of negative volumes in the mesh. An improvement to this method is the approach by Farhat[7], which incorporates nonlinear torsional springs to avoid the formation of negative volumes.

In the elastic analogy method[15], the entire domain is modeled as an elastic solid, and the mesh deformation is governed by the classical laws of solid elasticity. This method offers significant flexibility compared to the spring analogy method but comes with a higher computational cost.

In all mesh deformation methods, geometric complexity, computational cost, and reliability represent conflicting factors. PDE methods facilitate straightforward deformation but offer limited flexibility. Physical analogy methods can handle large deformations of complex geometries with reliability but at a high computational cost. Algebraic methods provide fast mesh deformation but are less reliable for complex geometries or intricate displacements.

The Rigid Body Motion (RBM) method developed in this thesis, is a physical analogy method based in Rigid Body Motion equations. In order to maintain the mesh quality close to the boundary it's combined with the Inverse Distance Weighting (IDW) method resulting to the Distance-Weighted RBM (DWRBM) algorithm. The DWRBM method is capable of large deformations in both structured and unstructured meshes, and its cost is reasonable even for dense 3D meshes, as presented in this thesis.

1.4 RBM Method

The RBM Method is a physical analogy mesh displacement method, which adapts the mesh under the principles of the RBM theory, shown in fig. 1.1. Liatsikouras[11] in his doctoral thesis, performed at the PCOpt Unit of NTUA, developed a mesh displacement algorithm, based on rigid body motion, defining clouds of nodes covering the entire domain and trying to keep them as rigid as possible during mesh displacement. This was a connectivity-agnostic algorithm and could work in every type of mesh. These clouds were defined based in a user-defined radius, forming a circle in 2D or a sphere in 3D. Weights were also used, to ensure higher quality mesh in regions of interest(such as boundary layers). The first method he developed, was solving a linear optimization problem, assuming infinite small rotations. This method required the use of sub-steps increasing the computation cost of the method. He improved this algorithm, firstly by reformulating the problem without the infinite small rotations assumption and secondly by using edge-based stencils. Christianos[5] in his diploma thesis at the same group, developed another mesh displacement algorithm, also based in rigid body motion theory, working with stencils concluding by each internal node with its neighbors, and trying to displace the mesh keeping these stencils as-rigid-as possible. He solved the optimization problem, expressing the above mentioned hypothesis both in a decoupled way(optimizing each stencil's displacement independently from the others) and in a coupled one(defining a total objective function). Christianos' method also needed the use of sub-steps for large displacement cases, increasing the cost. In this thesis, an improvement to the Christianos' coupled algorithm is presented, by introducing weights in the total objective function definition, giving more importance in sensitive to inverted cell regions. The DWRBM algorithm, was capable of larger displacements, in reasonably less time.

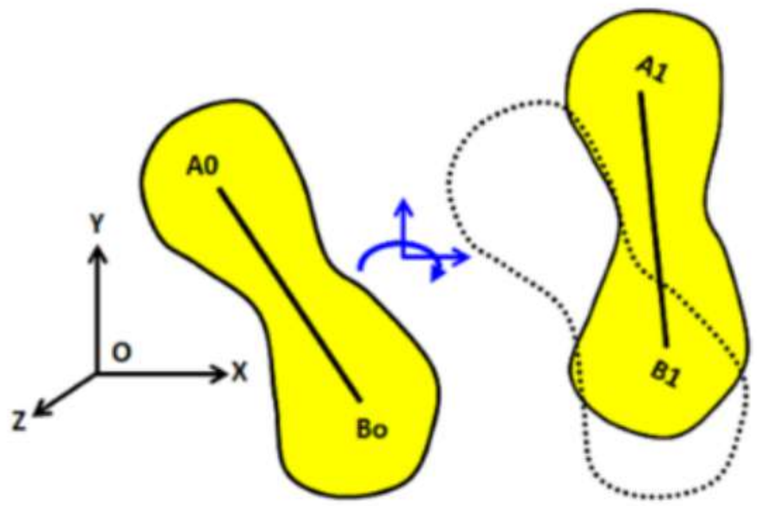


Figure 1.1: Rigid Body Motion in 2D. The distance between any two given points A and B of the body remains constant.[5]

1.5 Thesis Structure

This thesis focuses on the implementation, development, and testing of the Distance-Based Weighted RBM method on structured, unstructured, and hybrid meshes in both 2D and 3D. The method has been implemented using the C++ programming language, while post-processing was conducted using ParaView [1]. For the solution of the system of equations arising from the optimization problem, a preconditioned GMRES solver was developed, using a modified Compressed Sparce Row (mCSR) format. The structure of the thesis is outlined as follows:

Chapter 1: Introduction to mesh displacement methods in general, highlighting their significance and providing a brief overview of the RBM method.

Chapter 2: Presentation of the theoretical background and mathematical formulation of the DWRBM technique in 2D and 3D.

Chapter 3: Detailed explanation of the computational and numerical implementation, including a thorough presentation of the algorithm.

Chapter 4: Application of the DWRBM technique to a set of 2D and 3D cases, examining its performance, capabilities, and limitations.

Chapter 5: Summary and conclusions of the thesis, along with suggestions for future work.

Chapter 2

The RBM Method

2.1 Introduction

As stated in the Introduction, the RBM technique is a physical analogy mesh displacement method which adapts the mesh under the principles of the rigid body motion theory. Each node, connected with its neighbors, it's assumed to behave like a rigid body. So, by moving a set of boundary nodes, the internal ones move to adapt to the displacement, trying to keep their initial shape. This is achieved via solving an optimization problem, which minimizes the deviation between the final positions of the mesh nodes and the ideal ones (which corresponds to a pure rigid motion). Two approaches exist for the above mentioned technique, the decoupled and the coupled one. In the decoupled approach, the solution of each node i is computed independently from the solution of the other nodes. The final positions of the neighboring nodes are considered known, regardless of whether they are or not, as in the case of internal nodes. So, the iterative solution of the resulting systems of equations (3×3 in the case of 2D or 6×6 in the case of 3D for every internal node i) till convergence is needed. On the other hand, in the coupled approach, the objective function to be minimized is defined globally taking the displacements of all the internal nodes into account. So, the resulting system to be solved ($3N_I \times 3N_I$ in the case of 2D or $6N_I \times 6N_I$ in the case of 3D, where N_I is the number of internal nodes) gives the final displacements of all the internal nodes.

2.2 Theoretical Foundation

In Continuum Mechanics, rigid body is defined as a body in which deformation is zero during its displacement. If we consider the rigid body as an infinite set of particles then these particles do not move relatively to each other. In our method, considering each internal node i with its neighbors as a rigid body, we compute the final positions of each node, trying to respect the rigid body principles. However the pure rigid body motion for the entire mesh cannot be achieved because of the unrestricted movement of the boundaries, thus the method is applied minimizing the deviation from the ideal state. So, the objective function to be minimized is defined as this deviation, and the exact definition depends on the approach used, as mentioned in section 2.1.

A node is a zero-dimension entity which is inherently linked to the topology of the mesh, with 2 or 3 Degrees of Freedom(DoF), its coordinates, depending on the dimension of the mesh. An edge is a segment that links two nodes in the mesh, establishing the mesh's connectivity. A face is the surface formed by a closed loop of connected edges, with no other nodes within it. These concepts can be extended to 3D meshes. Crucially, these elements do not possess intrinsic physical meaning; instead, they serve as tools for discretizing the domain in order to compute the desired quantities. The nodes are classified in two main categories, the boundary and the internal nodes. The boundary nodes are then classified in 2 sub-categories, these of standing boundary nodes whose final positions are their old positions, and these that moved (with known displacements) during the optimization loop(or any other application such as fluid-structure interaction (FSI) or moving mesh application). In fig. 2.1, an example of the topology in an unstructured mesh is shown, for the central node M with its neighbors, defining the rigid body analogy as explained before. Its obvious that this can be extended to both structured and hybrid meshes, and in 3 dimensions as well.

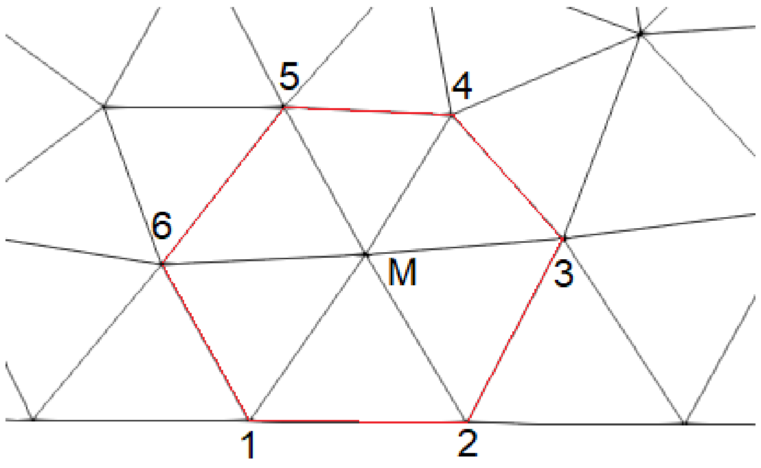


Figure 2.1: Example of 2D unstructured stencil,with red the rigid body analogy around the central node M with its neighbors

In this example, nodes 1 and 2 are boundary nodes with known final positions and nodes 3 to 6 and M are internal nodes, whose final positions is the desired outcome of the algorithm. If the decoupled approach is used, the final positions of nodes 3 to 6 are considered known when solving for M, and the RBM technique, applied to central node M, is able to specify its final position. But when the RBM technique applies to node 6 for example (now node 6 is the new central node), its final position differs from the one considered in the previous step for the computation of node's M displacement. Thus, an iterative procedure until convergence has to be applied. On the other hand, if the coupled approach is used, the final positions are not considered known and the RBM technique is not applied locally to each node, but a more global definition of the objective function is considered, that will be explained in next sections.

2.3 Formulation of the Decoupled Approach

As mentioned above, the decoupled approach strives to minimize an objective function locally defined at each internal node. This objective function is the deviation between the final coordinates of the central node M and the ideal ones, which corresponds to a pure rigid body motion. To compute the final position of the central node M, the final positions of its neighbors are required. If the neighbor is a boundary node, its final position is known. However, if the neighbor is an internal node, its final position might not be available when computing the displacement of the central node. As internal nodes are sequentially displaced during each iteration, the final position of a neighbor becomes known only if it has been displaced earlier in that iteration; otherwise, it remains unknown. To address with this discrepancy, the final position of the neighbor is assumed to be the same as its initial position in this iteration. After this adjustment, the nodes are successively displaced, starting with those nearest to the boundary nodes. This implies that a number of iterations are needed until the objective function convergence for each node.The algorithm of the above mentioned procedure is given above:

Algorithm 1 Decoupled RBM Algorithm

```
procedure DECOUPLED DISPLACEMENT($nodes, edges, faces$)
    for all $InternalNodes$ do
        $InternalNode.NewCoordinates \leftarrow InternalNode.OldCoordinates$
    end for
    $iteration \leftarrow 0$
    while ($iteration < maxIteration$ & $notConverged$) do
        for all $InternalNodes$ do
            Update $InternalNode.NewCoordinates$
        end for
        Check $Convergence$
        $iteration \leftarrow iteration + 1$
    end while
end procedure
```

To minimize the objective function, the partial derivatives with respect to the displacement parameters (which is discussed in the next sub-section) are needed. Setting these partial derivatives to zero gives the set of equations to be solved. For each internal node i, the solution of this 3×3 system for 2D or 6×6 system for 3D gives us the final position of each internal node i. The mathematical formulation of the above mentioned procedure will be discussed in sections 2.3.1 and 2.3.2.

2.3.1 2D Decoupled Mathematical Formulation

The mathematical expressions that describe the 2D motion of a rigid body are shown below:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \tag{2.1}$$

In eq. (2.1) the (x, y),(x', y') denote the initial and final coordinates respectively, θ denotes the angle of rotation around the z axis and $(\Delta x, \Delta y)$ denote the pure translation along x and y directions respectively. Obviously, this set expresses the motion from the center (0,0), however the generalization is trivial if another center of rotation is considered.
As mentioned above, in the decoupled approach the objective function F of each internal node i to be minimized is:

$$F_i = \frac{1}{2} \sum_{j \in N(i)} [(x_j^{ideal} - x_j^{new})^2 + (y_j^{ideal} - y_j^{new})^2] \tag{2.2}$$

where i denotes the central node of the stencil (M in fig. 2.1), j denotes each neighboring node of the central one, $N(i)$ the set of all the neighboring nodes of the central node, $(x_j^{ideal}, y_j^{ideal})$ denotes the set of the ideal final positions, assuming the stencil moves as a rigid body and (x_j^{new}, y_j^{new}) denotes the set of the final positions.
The expressions in eq. (2.2) are:

$$x_j^{ideal} = x_j cos\theta_i + y_j sin\theta_i + \Delta x_i \tag{2.3}$$

$$y_j^{ideal} = -x_j sin\theta_i + y_j cos\theta_i + \Delta y_i \tag{2.4}$$

$$x_j^{new} = \begin{cases} x_j cos\theta_j + y_j sin\theta_j + \Delta x_j & \text{, if } j \in IN(i) \\ x_j^* & \text{, if } j \in BN(i) \end{cases} \tag{2.5}$$

$$y_j^{new} = \begin{cases} -x_j sin\theta_j + y_j cos\theta_j + \Delta y_j & \text{, if } j \in IN(i) \\ y_j^* & \text{, if } j \in BN(i) \end{cases} \tag{2.6}$$

where $IN(i)$ indicates the internal neighbors of central node i, $BN(i)$ indicates the boundary neighbors of central node i and asterisk (*) indicates that these values are the known final positions, since the neighbor j is a boundary node. Interpreting this objective function, the aim is to minimize the deviation of this new final positions of each node and the ideal ones, which correspond to a pure rigid motion. To perform the minimization, the partial derivatives of the objective function w.r.t. the displacement parameters $(\Delta x, \Delta y, \theta)$ have to been set to zero. These three parameters, describe the final positions of each node i as shown above:

$$x_i^{new} = x_i^{old} cos\theta_i + y_i^{old} sin\theta_i + \Delta x_i \tag{2.7}$$

$$y_i^{new} = -x_i^{old} sin\theta_i + y_i^{old} cos\theta_i + \Delta y_i \tag{2.8}$$

The three equations to be solved are:

$$\frac{\partial F_i}{\partial \Delta x_i} = \sum_{j\in N(i)} \Big[x_j cos\theta_i + y_j sin\theta_i + \Delta x_i - x_j^*\Big] = 0 \tag{2.9}$$

$$\frac{\partial F_i}{\partial \Delta y_i} = \sum_{j\in N(i)} \Big[-x_j sin\theta_i + y_j cos\theta_i + \Delta y_i - y_j^*\Big] = 0 \tag{2.10}$$

$$\begin{aligned}\frac{\partial F_i}{\partial \Delta \theta_i} = \sum_{j\in N(i)} &\Big[(x_j \cos\theta_i + y_j \sin\theta_i + \Delta x_i - x_j^*)(-x_j \sin\theta_i + y_j \cos\theta_i)\Big]+ \\ &+\Big[(-x_j \sin\theta_i + y_j \cos\theta_i + \Delta y_i - y_j^*)(-x_j \cos\theta_i - y_j \sin\theta_i)\Big] = 0\end{aligned} \tag{2.11}$$

The solution procedure of the eqs. 2.9, 2.10 and 2.11 is described in three steps. Firstly compute Δx_i and Δy_i solving eqs. 2.9 and 2.10 respectively:

$$\Delta x_i = \frac{1}{n}\sum_{j\in N(i)} (x_j^* - x_j cos\theta_i - y_j sin\theta_i) \tag{2.12}$$

$$\Delta y_i = \frac{1}{n}\sum_{j\in N(i)} (y_j^* + x_j sin\theta_i - y_j cos\theta_i) \tag{2.13}$$

where n is the number of neighboring nodes of central node i.
Then eq. (2.11), can be rewritten as:

$$\frac{\partial F_i}{\partial \Delta \theta_i} = A sin\theta_i + B cos\theta_i = 0 \tag{2.14}$$

where A, B are given by:

$$A = \sum_{j\in N(i)} (x_j x_j^* + y_j y_j^* - x_j \Delta x_i - y_j \Delta y_i) \tag{2.15}$$

$$B = \sum_{j\in N(i)} (x_j y_j^* + y_j x_j^* + y_j \Delta x_i - x_j \Delta y_i) \tag{2.16}$$

eq. (2.14) can easily be solved using the Newton-Raphson method as shown:

$$set : G_i = A sin\theta_i + B cos\theta_i = 0 \tag{2.17a}$$

$$compute : G_i^{'} = \frac{\partial G_i}{\partial \Delta \theta_i} = A cos\theta_i - B sin\theta_i \tag{2.17b}$$

$$update : \theta_i^{new} = \theta_i^{old} - \frac{G_i}{G_i^{'}} \tag{2.17c}$$

The steps described in eq. (2.17) are repeated until convergence of θ_i. Then, Δx_i and Δy_i are recomputed using the updated value θ_i^{new}. As it can easily be seen in eqs. 2.12 to 2.17, the neighboring nodes j haven't been categorized to boundary or internals, as their final positions are assumed known, regardless the type of the node. From algorithm 1, the first step is to set the internal nodes' final positions equal to their initial positions and as the algorithm is applied to each node, their positions are updating. So, after computing the 3 displacement parameters for the first internal node, its final position is known, so the updated value is used in the next computation. The decoupled RBM algorithm is applied sequentially to each internal node, and the procedure is repeated until the objective function for every node converges.

2.3.2 3D Decoupled Mathematical Formulation

The mathematical expressions describing the 3D motion of a rigid body are shown below:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = R \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} \tag{2.18}$$

where R is the total rotation matrix which is equal to the multiplication of the three rotation matrices around each axis:

$$R = R_z R_y R_x \tag{2.19}$$

$$R_x(\theta_x) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta_x & -sin\theta_x \\ 0 & sin\theta_x & cos\theta_x \end{bmatrix} \tag{2.20}$$

$$R_y(\theta_y) = \begin{bmatrix} cos\theta_y & 0 & sin\theta_y \\ 0 & 1 & 0 \\ -sin\theta_y & 0 & cos\theta_y \end{bmatrix} \tag{2.21}$$

$$R_z(\theta_z) = \begin{bmatrix} cos\theta_z & -sin\theta_z & 0 \\ sin\theta_z & cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.22}$$

So, the combined rotation matrix R becomes:

$$R = \begin{bmatrix} cos\theta_y cos\theta_z & -cos\theta_x sin\theta_z + sin\theta_x sin\theta_y cos\theta_z & sin\theta_x sin\theta_z + cos\theta_x sin\theta_y cos\theta_z \\ cos\theta_y sin\theta_z & cos\theta_x cos\theta_z + sin\theta_x sin\theta_y sin\theta_z & -sin\theta_x cos\theta_z + cos\theta_x sin\theta_y sin\theta_z \\ -sin\theta_y & sin\theta_x cos\theta_y & cos\theta_x cos\theta_y \end{bmatrix} \tag{2.23}$$

In eqs. 2.18 to 2.23, (x, y, z),(x', y', z') denote the initial and final coordinates respectively, $(\theta_x, \theta_y, \theta_z)$ denote the angles of rotation around the (x, y, z) axis respectively and $(\Delta x, \Delta y, \Delta z)$ denote the pure translation along (x, y, z) directions respectively with the global origin (0, 0, 0) as the center of rotation. In 3D, the objective function F_i of each internal node i to be minimized is:

$$F_i^{3D} = \frac{1}{2} \sum_{j \in N(i)} [(x_j^{ideal} - x_j^{new})^2 + (y_j^{ideal} - y_j^{new})^2 + (z_j^{ideal} - z_j^{new})^2] \tag{2.24}$$

where i denotes the central node of the stencil, j denotes each neighboring node of the central one, $N(i)$ the set of all the neighboring nodes of the central node, $(x_j^{ideal}, y_j^{ideal}, z_j^{ideal})$ denotes the set of the ideal final positions, assuming the stencil moves under the disciplines of a rigid body and $(x_j^{new}, y_j^{new}, z_j^{new})$ denotes the set of the final positions. The expressions in eq. (2.24) are the below:

$$\begin{aligned} x_j^{ideal} =& x_j cos\theta_{yi} cos\theta_{zi} + \\ &+ y_j(-cos\theta_{xi} sin\theta_{zi} + sin\theta_{xi} sin\theta_{yi} cos\theta_{zi}) + \\ &+ z_j(sin\theta_{xi} sin\theta_{zi} + cos\theta_{xi} sin\theta_{yi} cos\theta_{zi}) + \\ &+ \Delta x_i \end{aligned} \tag{2.25}$$

$$\begin{aligned} y_j^{ideal} =& x_j cos\theta_{yi} sin\theta_{zi} + \\ &+ y_j(cos\theta_{xi} cos\theta_{zi} + sin\theta_{xi} sin\theta_{yi} sin\theta_{zi}) + \\ &+ z_j(-sin\theta_{xi} cos\theta_{zi} + cos\theta_{xi} sin\theta_{yi} sin\theta_{zi}) + \\ &+ \Delta y_i \end{aligned} \tag{2.26}$$

$$z_j^{ideal} = -x_j sin\theta_{yi} + y_j sin\theta_{xi} cos\theta_{yi} + z_j cos\theta_{xi} cos\theta_{yi} + \Delta z_i \tag{2.27}$$

$$x_j^{new} = \begin{cases} x_j \cos\theta_{yj} \cos\theta_{zj} + y_j(-\cos\theta_{xj} \sin\theta_{zj} + \sin\theta_{xj} \sin\theta_{yj} \cos\theta_{zj}) + \\ + z_j(\sin\theta_{xj} \sin\theta_{zj} + \cos\theta_{xj} \sin\theta_{yj} \cos\theta_{zj}) + \Delta x_j & \text{, if } j \in IN(i) \\ x_j^* & \text{, if } j \in BN(i) \end{cases} \tag{2.28}$$

$$y_j^{new} = \begin{cases} x_j \cos\theta_{yj} \sin\theta_{zj} + y_j(\cos\theta_{xj} \cos\theta_{zj} + \sin\theta_{xj} \sin\theta_{yj} \sin\theta_{zj}) + \\ + z_j(-\sin\theta_{xj} \cos\theta_{zj} + \cos\theta_{xj} \sin\theta_{yj} \sin\theta_{zj}) + \Delta y_j & \text{, if } j \in IN(i) \\ y_j^* & \text{, if } j \in BN(i) \end{cases} \tag{2.29}$$

$$z_j^{new} = \begin{cases} -x_j \sin\theta_{yj} + y_j \sin\theta_{xj} \cos\theta_{yj} + z_j \cos\theta_{xj} \cos\theta_{yj} + \Delta z_j & \text{, if } j \in IN(i) \\ z_j^* & \text{, if } j \in BN(i) \end{cases} \tag{2.30}$$

where $IN(i)$ indicates the internal neighbors of central node i, $BN(i)$ indicates the boundary neighbors of central node i and asterisk (*) indicates that these values are the known final positions, since the neighbor j is a boundary node. The 6×6 system to be solved arises from setting the partial derivatives of the objective function w.r.t the displacement parameters $(\Delta x, \Delta y, , \Delta z, \theta_{xi}, \theta_{yi}, \theta_{zi})$ to zero as shown above:

$$Eq.1 = \frac{\partial F_i^{3D}}{\partial \Delta x_i} = 0 \tag{2.31}$$

$$Eq.2 = \frac{\partial F_i^{3D}}{\partial \Delta y_i} = 0 \tag{2.32}$$

$$Eq.3 = \frac{\partial F_i^{3D}}{\partial \Delta z_i} = 0 \tag{2.33}$$

$$Eq.4 = \frac{\partial F_i^{3D}}{\partial \Delta \theta_{xi}} = 0 \tag{2.34}$$

$$Eq.5 = \frac{\partial F_i^{3D}}{\partial \Delta \theta_{yi}} = 0 \tag{2.35}$$

$$Eq.6 = \frac{\partial F_i^{3D}}{\partial \Delta \theta_{zi}} = 0 \tag{2.36}$$

So solving this 6×6 system of equations(as explained in algorithm 1 and, in a 2D case, in section 2.3.1) the final positions of each internal node i are:

$$\begin{aligned} x_i^{new} =& x_i^{old} cos\theta_{yi} cos\theta_{zi} + \\ &+ y_i^{old}(-cos\theta_{xi} sin\theta_{zi} + sin\theta_{xi} sin\theta_{yi} cos\theta_{zi}) + \\ &+ z_i^{old}(sin\theta_{xi} sin\theta_{zi} + cos\theta_{xi} sin\theta_{yi} cos\theta_{zi}) + \\ &+ \Delta x_i \end{aligned} \tag{2.37}$$

$$\begin{aligned} y_i^{new} =& x_i^{old} cos\theta_{yi} sin\theta_{zi} + \\ &+ y_i^{old}(cos\theta_{xi} cos\theta_{zi} + sin\theta_{xi} sin\theta_{yi} sin\theta_{zi}) + \\ &+ z_i^{old}(-sin\theta_{xi} cos\theta_{zi} + cos\theta_{xi} sin\theta_{yi} sin\theta_{zi}) + \\ &+ \Delta y_i \end{aligned} \tag{2.38}$$

$$z_i^{new} = -x_i^{old} sin\theta_{yi} + y_i^{old} sin\theta_{xi} cos\theta_{yi} + z_i^{old} cos\theta_{xi} cos\theta_{yi} + \Delta z_i \tag{2.39}$$

2.4 Formulation of the Coupled Approach

As mentioned in the Introduction, in the coupled RBM Approach, the objective function to be minimized is defined globally taking the displacements of all the internal nodes into account. This will ensure a better propagation of the displacement in the entire mesh, one of the drawbacks in the decoupled approach. Also, using weights, in the coupled approach we can control the quality of the mesh locally in regions with high interest. Such regions are the boundary layer mesh, around aerodynamic bodies for example, where a high quality mesh is required to capture the viscous phenomena. Also these regions are close in the external defined displacement (in aerodynamic shape optimization or FSI for example) and are prone to exhibit inverted cells. So, the total objective function to be minimized is:

$$F_{total} = \sum_{i \in I} w_i F_i \tag{2.40}$$

where F_i is the objective function as defined in the decoupled approach, I denotes the set of the internal nodes and w_i denotes the weights of each internal node.
In order to minimize the total objective function, the partial derivatives w.r.t the displacement parameters are set to zero. In the coupled approach, the displacement parameters are each node's $(\Delta x, \Delta y, \Delta z, \theta_x, \theta_y, \theta_z)$ in the case of 3D for example. The arising system ($3N_I \times 3N_I$ in the case of 2D or $6N_I \times 6N_I$ in the case of 3D, where N_I is the number of internal nodes) is much more computationally expensive than this of the decoupled method, but it needs to be solved just once. The algorithm in the coupled approach is shown in algorithm 2:

Algorithm 2 Coupled DWRBM Algorithm

```
procedure COUPLED DISPLACEMENT($nodes, edges, faces$)
    for all $InternalNodes$ do
        Calculate $Internal.Node.weight$
    end for
    Create System of Equations
    Solve System
    for all $InternalNodes$ do
        Update $InternalNode.NewCoordinates$
    end for
end procedure
```

2.4.1 2D Coupled Mathematical Formulation

As mentioned in section 2.4, in the coupled approach the objective function to be minimized, in the case of 2D, and taking into consideration the eqs. 2.2 to 2.6 takes the form below:

$$\begin{aligned} F_{total} =& \frac{1}{2}\sum_{i\in I}\sum_{j\in IN(i)} w_i\Big[(x_j cos\theta_i + y_j sin\theta_i + \Delta x_i) - (x_j cos\theta_j + y_j sin\theta_j + \Delta x_j)\Big]^2 + \\ &+\frac{1}{2}\sum_{i\in I}\sum_{j\in BN(i)} w_i\Big[(x_j cos\theta_i + y_j sin\theta_i + \Delta x_i) - (x_j^*)\Big]^2 + \\ &+\frac{1}{2}\sum_{i\in I}\sum_{j\in IN(i)} w_i\Big[(-x_j sin\theta_i + y_j cos\theta_i + \Delta y_i) - (-x_j sin\theta_j + y_j cos\theta_j + \Delta y_j)\Big]^2 + \\ &+\frac{1}{2}\sum_{i\in I}\sum_{j\in BN(i)} w_i\Big[(-x_j sin\theta_i + y_j cos\theta_i + \Delta y_i) - (y_j^*)\Big]^2 \end{aligned} \tag{2.41}$$

To minimize the objective function, its partial derivatives w.r.t. the displacement parameters $(\Delta x, \Delta y, \theta)$ have to set to zero. So, the system to be solved is:

$$\begin{aligned} Eq.1(i) = \frac{\partial F_{total}}{\partial \Delta x_i} &= \sum_{j \in IN(i)} w_i \Big[(x_j cos\theta_i + y_j sin\theta_i + \Delta x_i) - (x_j cos\theta_j + y_j sin\theta_j + \Delta x_j) \Big] - \\ &- \sum_{j \in IN(i)} w_j \Big[(x_i cos\theta_j + y_i sin\theta_j + \Delta x_j) - (x_i cos\theta_i + y_i sin\theta_i + \Delta x_i) \Big] + \\ &+ \sum_{j \in BN(i)} w_i \Big[(x_j cos\theta_i + y_j sin\theta_i + \Delta x_i) - (x_j^*) \Big] = 0 \end{aligned} \tag{2.42}$$

$$\begin{aligned} Eq.2(i) = \frac{\partial F_{total}}{\partial \Delta y_i} &= \sum_{j \in IN(i)} w_i \Big[(-x_j sin\theta_i + y_j cos\theta_i + \Delta y_i) - (-x_j sin\theta_j + y_j cos\theta_j + \Delta y_j) \Big] - \\ &- \sum_{j \in IN(i)} w_j \Big[(-x_i sin\theta_j + y_i cos\theta_j + \Delta y_j) - (-x_i sin\theta_i + y_i cos\theta_i + \Delta y_i) \Big] + \\ &+ \sum_{j \in BN(i)} w_i \Big[(-x_j sin\theta_i + y_j cos\theta_i + \Delta y_i) - (y_j^*) \Big] = 0 \end{aligned} \tag{2.43}$$

$$\begin{aligned} Eq.3(i) = \frac{\partial F_{total}}{\partial \Delta \theta_i} &= \sum_{j \in IN(i)} \Big[w_i [(x_j cos\theta_i + y_j sin\theta_i + \Delta x_i) - (x_j cos\theta_j + y_j sin\theta_j + \Delta x_j)] \cdot \\ &\qquad \cdot (-x_j sin\theta_i + y_j cos\theta_i) \Big] - \\ &- \sum_{j \in IN(i)} \Big[w_j [(x_i cos\theta_j + y_i sin\theta_j + \Delta x_j) - (x_i cos\theta_i + y_i sin\theta_i + \Delta x_i)] \cdot \\ &\qquad \cdot (-x_i sin\theta_j + y_i cos\theta_j) \Big] + \\ &+ \sum_{j \in BN(i)} \Big[w_i [(x_j cos\theta_i + y_j sin\theta_i + \Delta x_i) - (x_j^*)] \cdot \\ &\qquad \cdot (-x_j sin\theta_i + y_j cos\theta_i) \Big] + \\ &+ \sum_{j \in IN(i)} \Big[w_i [(-x_j sin\theta_i + y_j cos\theta_i + \Delta y_i) - (-x_j sin\theta_j + y_j cos\theta_j + \Delta y_j)] \cdot \\ &\qquad \cdot (-x_j cos\theta_i - y_j sin\theta_i) \Big] - \\ &- \sum_{j \in IN(i)} \Big[w_j [(-x_i sin\theta_j + y_i cos\theta_j + \Delta y_j) - (-x_i sin\theta_i + y_i cos\theta_i + \Delta y_i)] \cdot \\ &\qquad \cdot (-x_i cos\theta_j - y_i sin\theta_j) \Big] + \\ &+ \sum_{j \in BN(i)} \Big[w_i [(-x_j sin\theta_i + y_j cos\theta_i + \Delta y_i) - (y_j^*)] \cdot \\ &\qquad \cdot (-x_j cos\theta_i - y_j sin\theta_i) \Big] = 0 \end{aligned} \tag{2.44}$$

where w_i and w_j the weights of the central node and each neighbor respectively. As in the decoupled equations, IN denotes the internal neighbors while BN the boundary ones. These three equations have to be satisfied for each node simultaneously forming a $3NI \times 3NI$ non linear system. As the system is non linear, its Jacobian (exact or an approximation) is required in order to get the solution. More details about the solution method will be discussed in chapter 3, however the terms of the exact Jacobian will be presented here for mathematical completeness. The Jacobian matrix contains the partial derivatives of the system's equations w.r.t. the unknowns, the displacement parameters of each node in other words. The Jacobian terms involving each equation's node i displacement parameters $(\Delta x_i, \Delta y_i, \theta_i)$ are shown below:

$$\frac{\partial Eq.1(i)}{\partial \Delta x_i} = \sum_{j \in IN(i)} (w_i + w_j) + \sum_{j \in BN(i)} w_i \tag{2.45}$$

$$\frac{\partial Eq.1(i)}{\partial \Delta y_i} = 0 \tag{2.46}$$

$$\begin{aligned}\frac{\partial Eq.1(i)}{\partial \Delta \theta_i} = &\sum_{j \in IN(i)} \Big[w_i(-x_j sin\theta_i + y_j cos\theta_i) + w_j(-x_i sin\theta_i + y_i cos\theta_i) \Big] + \\ &+ \sum_{j \in BN(i)} \Big[w_i(-x_j sin\theta_i + y_j cos\theta_i) \Big]\end{aligned} \tag{2.47}$$

$$\frac{\partial Eq.2(i)}{\partial \Delta x_i} = 0 \tag{2.48}$$

$$\frac{\partial Eq.2(i)}{\partial \Delta y_i} = \sum_{j \in IN(i)} (w_i + w_j) + \sum_{j \in BN(i)} w_i \tag{2.49}$$

$$\begin{aligned}\frac{\partial Eq.2(i)}{\partial \Delta \theta_i} = &\sum_{j \in IN(i)} \Big[w_i(-x_j cos\theta_i - y_j sin\theta_i) + w_j(-x_i cos\theta_i - y_i sin\theta_i) \Big] + \\ &+ \sum_{j \in BN(i)} \Big[w_i(-x_j cos\theta_i - y_j sin\theta_i) \Big]\end{aligned} \tag{2.50}$$

$$\begin{aligned}\frac{\partial Eq.3(i)}{\partial \Delta x_i} = &\sum_{j \in IN(i)} \Big[w_i(-x_j sin\theta_i + y_j cos\theta_i) + w_j(-x_i sin\theta_i + y_i cos\theta_i) \Big] + \\ &+ \sum_{j \in BN(i)} \Big[w_i(-x_j sin\theta_i + y_j cos\theta_i) \Big]\end{aligned} \tag{2.51}$$

$$\begin{aligned}\frac{\partial Eq.3(i)}{\partial \Delta y_i} = &\sum_{j \in IN(i)} \Big[w_i(-x_j cos\theta_i - y_j sin\theta_i) + w_j(-x_i cos\theta_i - y_i sin\theta_i) \Big] + \\ &+ \sum_{j \in BN(i)} \Big[w_i(-x_j cos\theta_i - y_j sin\theta_i) \Big]\end{aligned} \tag{2.52}$$

$$\begin{aligned}\frac{\partial Eq.3(i)}{\partial \Delta\theta_i} = \sum_{j\in IN(i)} \Big[&(\Delta x_j - \Delta x_i)[w_i(x_j cos\theta_i + y_j sin\theta_i) + w_j(x_i cos\theta_i + y_i sin\theta_i)]+\\ &+(\Delta y_j - \Delta y_i)[w_i(-x_j sin\theta_i + y_j cos\theta_i) + w_j(-x_i sin\theta_i + y_i cos\theta_i)]+\\ &+(sin\theta_i sin\theta_j + cos\theta_i cos\theta_j)[w_i(x_j^2 + y_j^2) + w_j(x_i^2 + y_i^2)]\Big]+\\ + \sum_{j\in BN(i)} \Big[&w_i(\Delta x_i - x_j^*)[(-x_j cos\theta_i - y_j sin\theta_i)]+\\ &+(\Delta y_i - y_j^*)[(x_j sin\theta_i - y_j cos\theta_i)]\Big]\end{aligned} \tag{2.53}$$

The Jacobian terms involving the neighbors' displacement parameters $(\Delta x_j, \Delta y_j, \theta_j)$ are:

$$\frac{\partial Eq.1(i)}{\partial \Delta x_j} = -(w_i + w_j) \tag{2.54}$$

$$\frac{\partial Eq.1(i)}{\partial \Delta y_j} = 0 \tag{2.55}$$

$$\frac{\partial Eq.1(i)}{\partial \Delta\theta_j} = w_i(x_j sin\theta_j - y_j cos\theta_j) + w_j(x_i sin\theta_j - y_i cos\theta_j) \tag{2.56}$$

$$\frac{\partial Eq.2(i)}{\partial \Delta x_j} = 0 \tag{2.57}$$

$$\frac{\partial Eq.2(i)}{\partial \Delta y_j} = -(w_i + w_j) \tag{2.58}$$

$$\frac{\partial Eq.2(i)}{\partial \Delta\theta_j} = w_i(x_j cos\theta_j + y_j sin\theta_j) + w_j(x_i cos\theta_j + y_i sin\theta_j) \tag{2.59}$$

$$\frac{\partial Eq.3(i)}{\partial \Delta x_j} = w_i(x_j sin\theta_i - y_j cos\theta_i) + w_j(x_i sin\theta_i - y_i cos\theta_i) \tag{2.60}$$

$$\frac{\partial Eq.3(i)}{\partial \Delta y_i} = w_i(x_j cos\theta_i + y_j sin\theta_i) + w_j(x_i cos\theta_i + y_i sin\theta_i) \tag{2.61}$$

$$\frac{\partial Eq.3(i)}{\partial \Delta\theta_i} = (-sin\theta_i sin\theta_j - cos\theta_i cos\theta_j)[w_i(x_j^2 + y_j^2) + w_j(x_i^2 + y_i^2)]\Big] \tag{2.62}$$

2.4.2 3D Coupled Mathematical Formulation

As mentioned in section 2.4, in the coupled approach the objective function to be minimized, a in 3D case, is:

$$F_{total}^{3D} = \sum_{i\in I} w_i F_i^{3D} \tag{2.63}$$

where w_i are the weights of each node and F_i^{3D} is each nodes objective function as defined in eq. (2.24). The extension from 2D to 3D is straightforward, and for the sake of space, it will be omitted. However, the basic equations which the terms arise from are given below.

$$Eq.1(i) = \frac{\partial F^{3D}_{total}}{\partial \Delta x_i} = 0 \tag{2.64}$$

$$Eq.2(i) = \frac{\partial F^{3D}_{total}}{\partial \Delta y_i} = 0 \tag{2.65}$$

$$Eq.3(i) = \frac{\partial F^{3D}_{total}}{\partial \Delta z_i} = 0 \tag{2.66}$$

$$Eq.4(i) = \frac{\partial F^{3D}_{total}}{\partial \Delta \theta_{xi}} = 0 \tag{2.67}$$

$$Eq.5(i) = \frac{\partial F^{3D}_{total}}{\partial \Delta \theta_{yi}} = 0 \tag{2.68}$$

$$Eq.6(i) = \frac{\partial F^{3D}_{total}}{\partial \Delta \theta_{zi}} = 0 \tag{2.69}$$

As in 2D, this $6NI \times 6NI$ system of linear equations , where NI is the number of internal nodes, needs to be solved. As the system is non-linear, the Jacobian matrix is required. The entries to the Jacobian matrix of the system consist of the partial derivatives of the above mentioned eqs. (2.64 to 2.69) w.r.t. to the displacement parameters $(\Delta x_i, \Delta y_i, \Delta z_i, \theta_{xi}, \theta_{yi}, \theta_{zi})$ and $(\Delta x_j, \Delta y_j, \Delta z_j, \theta_{xj}, \theta_{yj}, \theta_{zj})$, where $j \in IN(i)$ with $IN(i)$ denoting the internal neighbors of i.

2.5 Linearization via Approximation

As mentioned in section 2.4, the system arising from the RBM method is non-linear. Assuming small displacement, an assumption valid in aerodynamic shape optimization applications, the above mentioned equations can be linearized. This assumption serves the purpose of reducing the computational cost, without affecting the quality of the resulting mesh[5]. It must be noted, that this linearization is irrelevant to the linearization of non linear systems from the numerical analysis aspect. Thus, this linearization arises from a physics based assumption that the angle of rotation is infinitesimal so as θ tends to zero:

$$\theta \to 0 \Rightarrow \begin{cases} sin\theta \to \theta \\ cos\theta \to 1 \end{cases} \tag{2.70}$$

Using eq. (2.70) the linearization is trivial. The 2D Coupled linear equations will be presented:

$$\begin{aligned}F_{total}^{approx} =&\frac{1}{2}\sum_{i\in I}\sum_{j\in IN(i)} w_i\Big[(y_j\theta_i+\Delta x_i)-(y_j\theta_j+\Delta x_j)\Big]^2+\\ &+\frac{1}{2}\sum_{i\in I}\sum_{j\in BN(i)} w_i\Big[(y_j\theta_i+\Delta x_i)-(x_j^*)\Big]^2+\\ &+\frac{1}{2}\sum_{i\in I}\sum_{j\in IN(i)} w_i\Big[(-x_j\theta_i+\Delta y_i)-(-x_j\theta_j+\Delta y_j)\Big]^2+\\ &+\frac{1}{2}\sum_{i\in I}\sum_{j\in BN(i)} w_i\Big[(-x_j\theta_i+\Delta y_i)-(y_j^*)\Big]^2\end{aligned} \tag{2.71}$$

$$\begin{aligned}Eq.1(i)=\frac{\partial F_{total}^{approx}}{\partial \Delta x_i}=&\sum_{j\in IN(i)}\Big[w_i[y_j(\theta_i-\theta_j)+(\Delta x_i-\Delta x_j)]+w_j[y_i(\theta_i-\theta_j)+(\Delta x_i-\Delta x_j)]\Big]+\\ &+\sum_{j\in BN(i)}\Big[w_i[x_j+y_j\theta_i+\Delta x_i-x_j^*]\Big]=0\end{aligned} \tag{2.72}$$

$$\begin{aligned}Eq.2(i)=\frac{\partial F_{total}^{approx}}{\partial \Delta y_i}=&\sum_{j\in IN(i)}\Big[w_i[x_j(\theta_j-\theta_i)+(\Delta y_i-\Delta y_j)]+w_j[x_i(\theta_j-\theta_i)+(\Delta y_i-\Delta y_j)]\Big]+\\ &+\sum_{j\in BN(i)}\Big[w_i[-x_j\theta_i+y_j+\Delta y_i-y_j^*]\Big]=0\end{aligned} \tag{2.73}$$

$$\begin{aligned}Eq.3(i)=\frac{\partial F_{total}^{approx}}{\partial \Delta \theta_i}=&\sum_{j\in IN(i)}\Big[w_i[(y_j(\Delta x_i-\Delta x_j)-x_j(\Delta y_i-\Delta y_j)+(x_j^2+y_j^2)(\theta_i-\theta_j)]+\\ &+w_j[(y_i(\Delta x_i-\Delta x_j)-x_i(\Delta y_i-\Delta y_j)+(x_i^2+y_i^2)(\theta_i-\theta_j)]\Big]+\\ &+\sum_{j\in BN(i)}\Big[w_i[y_j(\Delta x_i-x_j{}^*)-x_j(\Delta y_i-y_j{}^*)+\theta_i(x_j^2+y_j^2)]\Big]=0\end{aligned} \tag{2.74}$$

$$\frac{\partial Eq.1(i)}{\partial \Delta x_i}=\sum_{j\in IN(i)}(w_i+w_j)+\sum_{j\in BN(i)}w_i \tag{2.75}$$

$$\frac{\partial Eq.1(i)}{\partial \Delta y_i}=0 \tag{2.76}$$

$$\frac{\partial Eq.1(i)}{\partial \Delta \theta_i}=\sum_{j\in IN(i)}\Big[w_iy_j+w_jy_i\Big]+\sum_{j\in BN(i)}w_iy_j \tag{2.77}$$

$$\frac{\partial Eq.2(i)}{\partial \Delta x_i}=0 \tag{2.78}$$

$$\frac{\partial Eq.2(i)}{\partial \Delta y_i} = \sum_{j\in IN(i)} (w_i + w_j) + \sum_{j\in BN(i)} w_i \tag{2.79}$$

$$\frac{\partial Eq.2(i)}{\partial \Delta \theta_i} = \sum_{j\in IN(i)} \Big[w_i(-x_j) + w_j(-x_i)\Big] + \sum_{j\in BN(i)} -w_i x_j \tag{2.80}$$

$$\frac{\partial Eq.3(i)}{\partial \Delta x_i} = \sum_{j\in IN(i)} \Big[w_i y_j + w_j y_i\Big] + \sum_{j\in BN(i)} w_i y_j \tag{2.81}$$

$$\frac{\partial Eq.3(i)}{\partial \Delta y_i} = \sum_{j\in IN(i)} \Big[w_i(-x_j) + w_j(-x_i)\Big] + \sum_{j\in BN(i)} -w_i x_j \tag{2.82}$$

$$\frac{\partial Eq.3(i)}{\partial \Delta \theta_i} = \sum_{j\in IN(i)} \Big[w_i(x_j^2 + y_j^2) + w_j(x_i^2 + y_i^2)\Big] + \sum_{j\in BN(i)} \Big[w_i(x_j^2 + y_j^2)\Big] \tag{2.83}$$

$$\frac{\partial Eq.1(i)}{\partial \Delta x_j} = -(w_i + w_j) \tag{2.84}$$

$$\frac{\partial Eq.1(i)}{\partial \Delta y_j} = 0 \tag{2.85}$$

$$\frac{\partial Eq.1(i)}{\partial \Delta \theta_j} = -(w_i y_j + w_j y_i) \tag{2.86}$$

$$\frac{\partial Eq.2(i)}{\partial \Delta x_j} = 0 \tag{2.87}$$

$$\frac{\partial Eq.2(i)}{\partial \Delta y_j} = -(w_i + w_j) \tag{2.88}$$

$$\frac{\partial Eq.2(i)}{\partial \Delta \theta_j} = w_i x_j + w_j x_i \tag{2.89}$$

$$\frac{\partial Eq.3(i)}{\partial \Delta x_j} = -(w_i y_j + w_j y_i) \tag{2.90}$$

$$\frac{\partial Eq.3(i)}{\partial \Delta y_i} = w_i x_j + w_j x_i \tag{2.91}$$

$$\frac{\partial Eq.3(i)}{\partial \Delta \theta_i} = -w_i(x_j^2 + y_j^2) - w_j(x_i^2 + y_i^2) \tag{2.92}$$

2.6 Sub-Step Method

The previously discussed linearization, is valid when the assumption of small displacements is accurate not far from truth. In aerodynamic shape optimization applications, this is true, but in other applications such us aeroelasticity or rotating geometries, this assumption does not hold. In these more extreme displacement scenarios, the sub-step method needs to be applied. In this method, the displacement is split in smaller sub-steps where the assumption of small displacements is valid. Thus, this method gradually displaces the mesh nodes, in return of increasing the computational cost. This cost increases depending on the number of sub-steps applied, thus a wise choice of their number is required. To balance the computation cost versus the quality of the mesh, but also to make the selection more robust and independent of the user's experience, Christianos[5] implemented a robust algorithm to calculate the number of the sub-steps. The number of steps is calculated based on the total displacement of each node relative to the size of its stencil. Specifically, for each node, the steps are determined by finding the minimum number of divisions required so that each step's displacement is smaller than the shortest distance to any neighboring node. The final number of steps used is the maximum value found across all boundary nodes. This approach helps minimize the occurrence of inverted cells in the final mesh. Throughout the process, the magnitude of displacement for each step remains constant, adjusted to fit the specific characteristics of the problem. In fig. 2.2 a rotation of an airfoil in 3 sub-steps is illustrated. It's also noted that the sub-step method should also be applied in the non-linear RBM method if it exhibits inverted cells, in the case of extreme displacemnts.

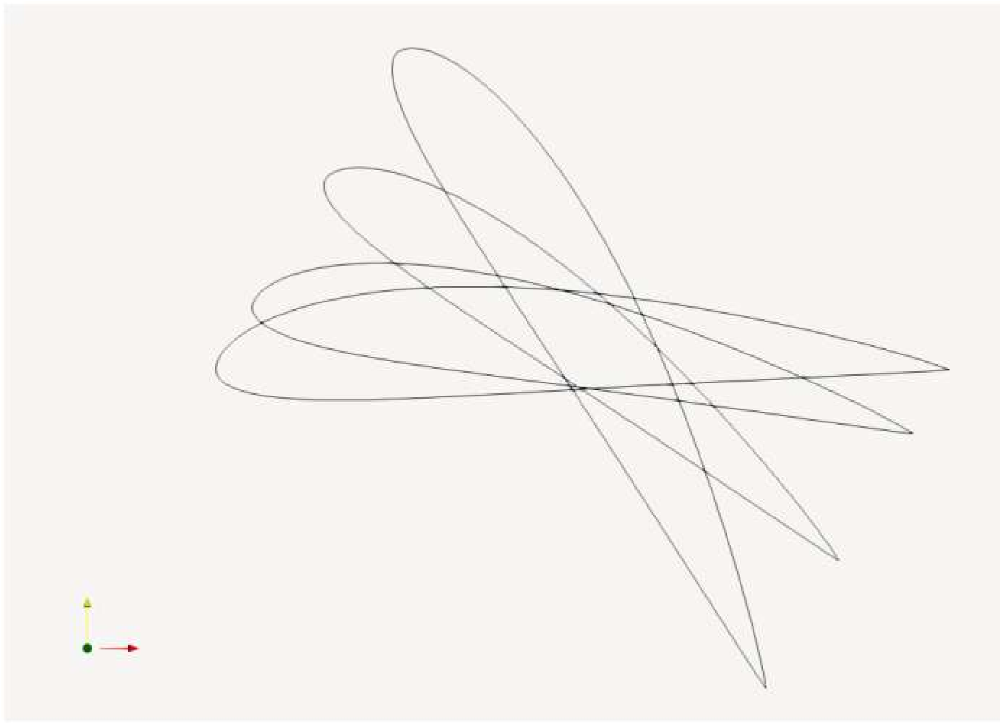


Figure 2.2: Airfoil rotation in 3 sub-steps[5]

2.7 3D mesh Quality metrics

To evaluate the capabilities of the RBM, the quality of the resulting mesh must be assessed. In 2D cases, visual inspection is often sufficient, as it allows for an intuitive evaluation of the final mesh and the detection of any inverted cells. However, in 3D cases, visual evaluation becomes significantly more challenging, necessitating the use of quantitative metrics.

For 3D meshes, the primary quality criterion used will be the scaled Jacobian, as defined below. The Jacobian, in the context of mesh generation, is a mathematical term used to define the transformation from physical space (x,y,z) to computational space (ξ, η, ζ) given by

$$J = \frac{\partial(x,y,z)}{\partial(\xi,\eta,\zeta)}$$

The mesh is invalid if the determinant of this matrix is zero or negative. For linear meshes this represents the case where the cell volumes are less than or equal to zero using right-hand-rule notation. The scaled Jacobian is the ratio of the minimum Jacobian value inside an element divided by the maximum value in the element. It is a measure of the variation in the Jacobian across the element. This metric ranges from $-\infty$ to 1, where a value of 1 represents a perfect mesh element. Values below zero indicate inverted cells, rendering the mesh invalid for CFD simulations.

Chapter 3

Computational and Numerical Framework

3.1 Introduction

As stated in the previous chapter, the RBM displacement algorithm requires careful computational and numerical treatment. Firstly, advanced data structure and storage schemes are required to storage the topology and the characteristics of the mesh, as it is unstructured in general. Even when the mesh is structured, it is treated as unstructured, to prevent a second customized code developement. For the storage of the matrices (the Jacobian for example) a modified Compressed Sparse Row (mCSR) format is used, in order to save memory[12]. The modification aims to have easy access to the diagonal of the matrix, as well as the upper and lower triangular matrices (they are used in preconditioning). Also, one key point of the Coupled RBM approach is the use of weights. The weights are based in a local distance of each node, defined as the minimum distance of each node to the closest boundary node. Appropriate expressions are suggested such as inverted distance weighting (IDW), with variable exponential factor, as well as sigmoid expressions. As the non-linear system is formed, the solution of this system is the next step. In order to solve the non-linear system, the Generalized Minimal Residual method (GMRES) is used. GMRES [13] is a Krylov subspace method which is suitable for solving large sparse systems. It is combined with the Newton method to handle the non-linearity, so when we use the exact Jacobian matrix we refer to the solver as Newton-GMRES (N-GMRES), otherwise if an approximation to the Jacobian matrix is used, we refer to the solver as quasi-Newton-GMRES (qN-GMRES). In order to speed up convergence left-preconditioning is used, choosing between Diagonal or Gauss-Seidel preconditioner. Finally, the algorithmic framework is presented and explained step by step.

3.2 Data structure and Storage

The RBM algorithm needs information about the topology and connectivity of the mesh in order to be applied. Firstly, it needs the dimension of the mesh, the number of nodes, the type of each element (triangles or quads in 2D, tetrahedra, hexahedra, pyramids or prisms in 3D) and the number of each element type. It also needs the coordinates of each node, as well as the type of the node. The type of the node is stored using flags(0 for unknown internal nodes, 1 for moving boundary nodes and 2 for standing boundary nodes). Then for each node, its neighbors are stored extracting information from the elements. Also a global to local (unknown) numbering needs to be stored for use in the Jacobian. So for each node, we also store its unknown index, as its shown in the fig. 3.1. If the flag of the node is 1 or 2, which indicates the boundary nodes, their unknown index is left blank, as its not defined. In the fig. 3.1 N indicates the number of nodes, while NI indicates the number of internal nodes. This mapping helps us save computational cost, for defining the Jacobian matrix, as this unknown index indicates the column position of the contribution coefficient of each internal node. This information is gained via 2 input files, one for the nodes' information and one for the elements' information. One more input file which contains the displacement of each moving node is required. Also the user can provide any other format, and the code will generate these 2 input files in the format needed. So, when the data structure is created, the next step is to form the non-linear system to be solved based in the equations of chapter 2.

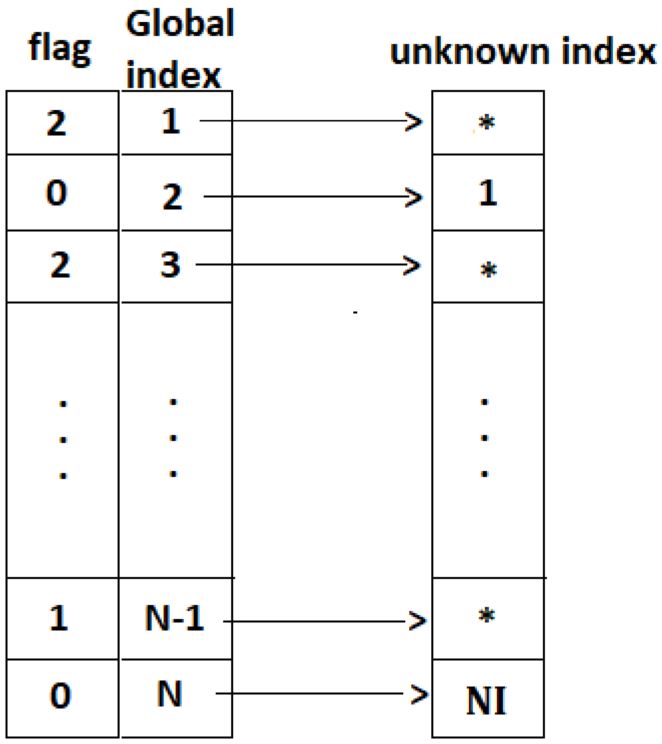
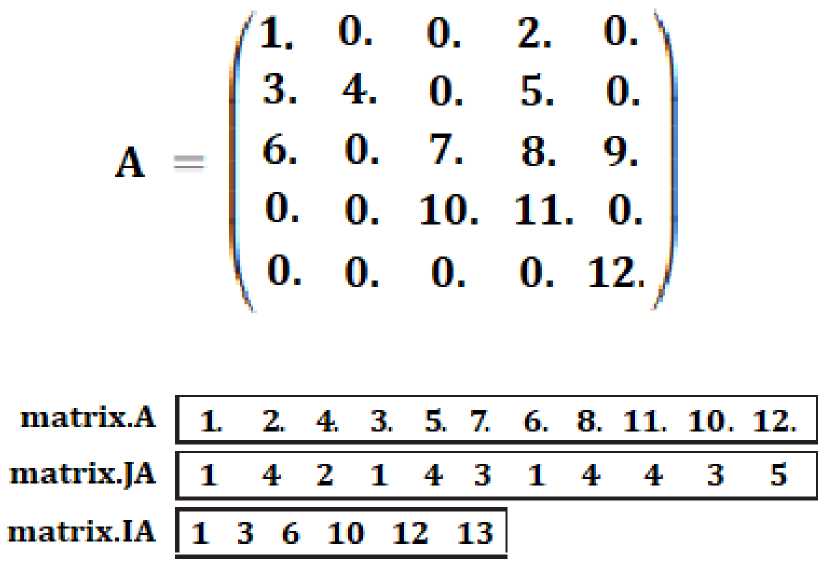


Figure 3.1: Global to Unknown index definition

In order to take advantage of the large number of zero elements, a modified Compressed Sparse Row (mCSR) format is used. In the Compressed Sparse Row format[12], the sparse matrix is stored in 3 vectors. The first vector, from now referred to as matrix.A, stores the non zero elements values of the matrix. The second one, matrix.JA, is an integer vector which stores the column of each non zero element of the matrix and the third one, matrix.IA, is an integer vector which stores the pointers to the beginning of each row in the vector matrix.A. The modification is the storage of each diagonal element as first in every row, which saves computational time each time we need access to the diagonal element. One example of the mCSR storage format is shown in fig. 3.2. In the example, memory saving does not occur, but as the size of the matrix grows and the sparsity increases, the effectiveness of this format becomes clear. Having easy access to the diagonal element and, thus, its column we can also have access to the upper or lower triangular matrix for use in preconditioning. Thus, the storage needed is $2 \times NnZeros$ plus $1 \times 3NI$ or $1 \times 6NI$, where $NnZeros$ indicates the number of non zero elements instead of $3NI \times 3NI$ in 2D or $6NI \times 6NI$ in 3D that the full storage would require, as computational time is saved from not calculating the diagonal elements or the upper/lower triangular matrices. An interesting question here is why the easy access to the upper or lower triangular matrix is an asset. This speeds up the convergence of the preconditioner, for example in the case of the Gauss-Seidel preconditioner, one forward substitution using the D+L matrix is used instead of 5-10 G-S iterations, converging much faster and without even needing the storage of the preconditioning matrix. Of course, all the basic sparse matrix operations are based in this modified CSR format. For more information about the storage of sparse matrices and operations employing sparse storage formats the reader is referred to [12].



$$\mathbf{A} = \begin{pmatrix} 1. & 0. & 0. & 2. & 0. \\ 3. & 4. & 0. & 5. & 0. \\ 6. & 0. & 7. & 8. & 9. \\ 0. & 0. & 10. & 11. & 0. \\ 0. & 0. & 0. & 0. & 12. \end{pmatrix}$$

matrix.A	1.	2.	4.	3.	5.	7.	6.	8.	11.	10.	12.
matrix.JA	1	4	2	1	4	3	1	4	4	3	5
matrix.IA	1	3	6	10	12	13					

Figure 3.2: The modified Compressed Sparse Row (mCSR) format

3.3 Distance-based Weighting

As mentioned above, in the coupled approach the use of weights is necessary in order to ensure sufficient mesh quality, especially in the region close to the aerodynamic body. In the boundary layer region, the displaced mesh is extremely vulnerable to inverted cells, as the elements there have high aspect ratio and can't handle large displacement. The importance of using weights is shown in fig. 3.3. In this case, the leading and trailing edges are lifted in order to reduce the curvature.

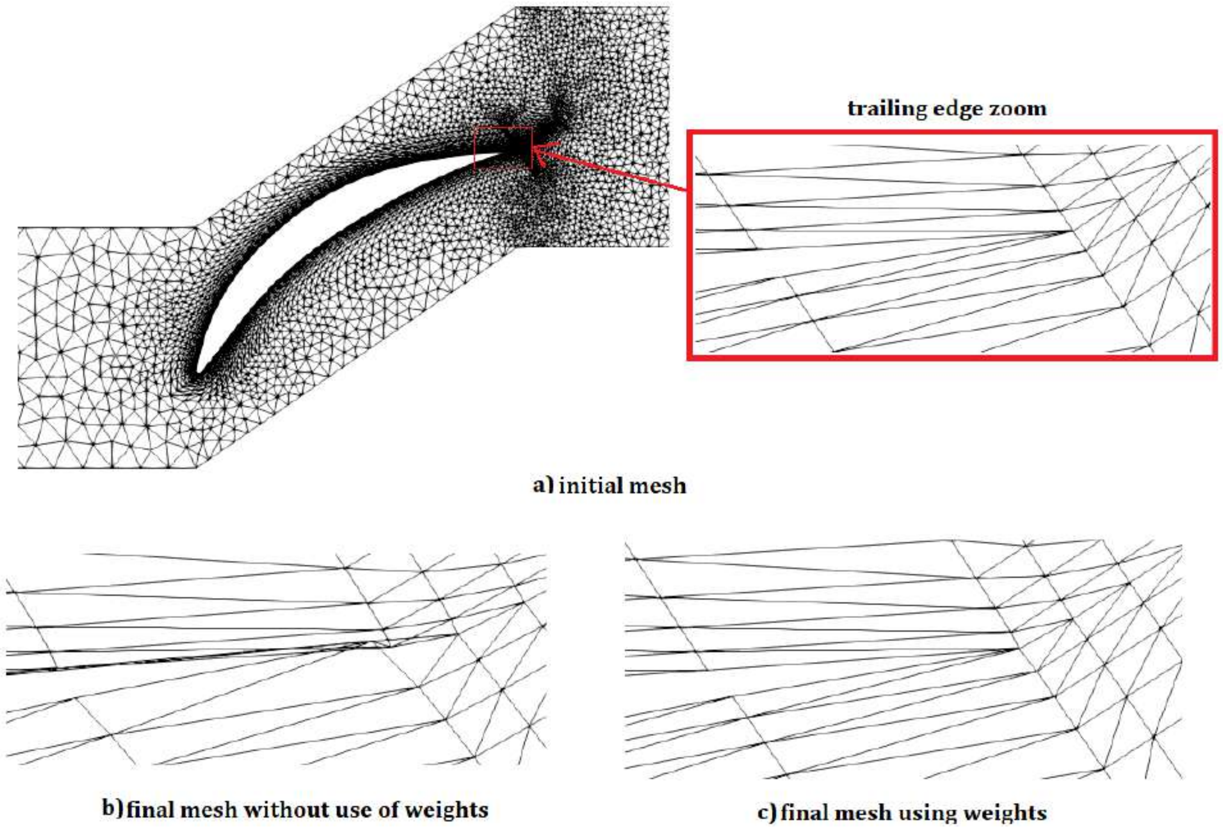

trailing edge zoom
a) initial mesh
b) final mesh without use of weights
c) final mesh using weights

Figure 3.3: Upper: a) initial mesh around a compressor cascade airfoil, zoomed at the trailing edge. Lower: final mesh zoomed at the trailing edge after coupled RBM algorithm, b) without the use of weights, c) using weights

The distance-based weighting is implemented to enhance the contribution of nodes near the boundary in the objective function. This approach specifically emphasizes the rigid body motion of the cells close to the boundaries. This distance is defined as the distance of the node to the closest boundary node (moving or not) and its calculated as shown in algorithm 3. The minimum distance, of all the nodes, is also stored, in case the weight expression needs to be normalized.

Algorithm 3 Distance Finding

```
procedure DISTANCE FINDING($nodes$)
    for all $i \in InternalNodes$ do
        $i.distance \leftarrow 10e9$                    ▷ Initializing with one large value
        for all $j \in BoundaryNodes$ do
            $distance \leftarrow \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}$
            if $i.distance > distance$ then
                $i.distance \leftarrow distance$
            end if
        end for
    end for
end procedure
```

Different expressions are used for the distance-based weighting, the one is based in the inverse-distance weighting (IDW) and the others belong to the sigmoid functions family. The two most important criteria for the selection of the weighting expression is the robustness and the quality of the resulting mesh. So different cases were tested, including coarse and fine meshes, and the IDW approach gave better results. Also using the exponential factor as a parameter enables having full control of the mesh quality, so this expression is used in this theses. However the optimal selection of the weighting expression is still under investigation and more expressions have to be tested. The expression of the weight used is shown in eq. (3.1).

$$w = \frac{1}{D^p} \tag{3.1}$$

where D is the distance as defined in the algorithm 3, and p is the exponential factor used as parameter with higher values improving the quality near the boundaries, with the price to pay of lower quality mesh far from these boundaries. The value range is between 1 and 2, with $p = 1.3$ being a good compromise.

3.4 Solution Algorithms

3.4.1 Linearization-Newton Method

As mentioned above, the system derived from the coupled RBM method is non-linear. In order to be solved, the linearization of the system is needed. Using the Newton method, the non-linear system w.r.t. $\boldsymbol{x}$ of eq. (3.2) (here $\mathbf{R}$ indicates a non linear operator) becomes linear w.r.t. $\boldsymbol{\delta x}$ as shown below:

$$\mathbf{R}(x) = 0 \tag{3.2}$$

$$\mathbf{J}^{old}\boldsymbol{\delta x}^{new} = -\mathbf{R}(x)^{old} \tag{3.3}$$

where $\mathbf{J}$ is the Jacobian matrix and $\boldsymbol{\delta x}$ is defined as :

$$\boldsymbol{x}^{new} = \boldsymbol{x}^{old} + \boldsymbol{\delta x}^{new} \tag{3.4}$$

As it's obvious in the above mentioned equations, the solution $\boldsymbol{x}$ is obtained after a number of iterations, from now called Newton iterations. In eq. (3.3), either the exact Jacobian matrix can be used as described in the equations of chapter 2 or an approximation to the Jacobian using finite difference schemes as:

$$\mathbf{J}_{ij} = \frac{\mathbf{R_i}(\boldsymbol{x}) - \mathbf{R_i}(\boldsymbol{x} + \epsilon_\mathbf{j})}{\epsilon_j} \tag{3.5}$$

where ϵ_j indicating small step of the j component of unknowns vector $\boldsymbol{x}$. Of course, central differences scheme can be used for second order approximation, but with increasing cost (double function evaluation cost). The linear system of eq. (3.3) will be solved using the General Minimal Residual method, a Krylov subspace method developed by Yousef Saad [13].

3.4.2 Arnoldi's Method

Arnoldi's method[2][12] is an orthogonal projection technique onto the Krylov subspace K_m, specifically designed for general non-Hermitian matrices. Introduced in 1951, the method was originally proposed as a way to reduce a dense matrix to Hessenberg form using a unitary transformation. In his original work, Arnoldi suggested that the eigenvalues of the Hessenberg matrix, obtained after fewer than n steps, could serve as accurate approximations to some of the eigenvalues of the original matrix. This insight later evolved into an effective method for approximating eigenvalues of large sparse matrices, and the approach was further adapted for solving large sparse linear systems.

Algorithm 4 Arnoldi's Method

```
1: Choose a vector $v_1$, such that $\|v_1\|_2 = 1$
2: for $j = 1, 2, \ldots, m$ do
3:     for $i = 1, 2, \ldots, j$ do
4:         Compute $h_{ij} = (Av_j, v_i)$
5:     end for
6:     Compute $w_j := Av_j - \sum_{i=1}^{j} h_{ij} v_i$
7:     Compute $h_{j+1,j} = \|w_j\|_2$
8:     if $h_{j+1,j} = 0$ then
9:         Stop
10:    else
11:        Set $v_{j+1} = \frac{w_j}{h_{j+1,j}}$
12:    end if
13: end for
```

At each step, the algorithm multiplies the previous Arnoldi vector v_j by A and then orthonormalizes the resulting vector w_j against all previous vectors v_i by a standard Gram-Schmidt procedure. It will break if the vector w_j computed in line 4 is zero (or under a user defined small value), indicating that the dimension of the Krylov subspace is sufficient.

3.4.3 GMRES

The Generalized Minimum Residual Method (GMRES)[13] is a projection method based on taking $K = K_m$ and $L = AK_m$, where K_m is the m-th Krylov subspace with $v_1 = \frac{r_0}{\|r_0\|_2}$. As is evident from its name, this method minimizes the residual norm over all vectors in $x_0 + K_m$. The GMRES algorithm is shown below:

Algorithm 5 GMRES

```
1: Compute $r_0 = b - Ax_0$, $\beta := \|r_0\|_2$, and $v_1 := \frac{r_0}{\beta}$
2: for $j = 1, 2, \ldots, m$ do
3:     Compute $w_j := Av_j$
4:     for $i = 1, \ldots, j$ do
5:         $h_{ij} := (w_j, v_i)$
6:         $w_j := w_j - h_{ij}v_i$
7:     end for
8:     $h_{j+1,j} = \|w_j\|_2$
9:     if $h_{j+1,j} = 0$ then
10:        Set $m := j$ and go to step 14
11:    end if
12:    $v_{j+1} = \frac{w_j}{h_{j+1,j}}$
13: end for
14: Define the $(m+1) \times m$ Hessenberg matrix $\bar{H}_m = \{h_{ij}\}_{1 \le i \le m+1, 1 \le j \le m}$
15: Compute $y_m$ the minimizer of $\|\beta e_1 - \bar{H}_m y\|_2$ and $x_m = x_0 + V_m y_m$
16: If satisfied Stop, else set $x_0 = x_m$ and go to step 1
```

In step 15 of algorithm 5 the computation of y_m is inexpensive as it's computed by solving a $m + 1 \times m$ least squares problem, where m is typically small. This least squares problem is solved using plane rotations transforming the Hessenberg matrix into an upper triangular form. So multiplying the Hessenberg matrix with each plane rotation matrix as defined in eq. (3.6):

$$\Omega_i = \begin{pmatrix} 1 & & & & & & & \\ & \ddots & & & & & & \\ & & 1 & & & & & \\ & & & c_i & s_i & & & \\ & & & -s_i & c_i & & & \\ & & & & & 1 & & \\ & & & & & & \ddots & \\ & & & & & & & 1 \end{pmatrix} \tag{3.6}$$

with $c_i^2 + s_i^2 = 1$. Generally, the scalars c_i and s_i of the i_{th} rotation Ω_i are defined as:

$$s_i = \frac{h_{i+1,i}}{\sqrt{(h_{ii}^{(i-1)})^2 + h_{i+1,i}^2}}, \quad c_i = \frac{h_{ii}^{(i-1)}}{\sqrt{(h_{ii}^{(i-1)})^2 + h_{i+1,i}^2}} \tag{3.7}$$

So multiplying Hessenberg matrix with Ω_1, h_{21} is eliminated, and repeating until all h_{i+1i} terms to be eliminated. For example in the case of $m = 5$ the Heisenberg matrix and the corresponding right-hand side shown in eq. (3.7):

$$\bar{H}_5 = \begin{pmatrix} h_{11} & h_{12} & h_{13} & h_{14} & h_{15} \\ h_{21} & h_{22} & h_{23} & h_{24} & h_{25} \\ & h_{32} & h_{33} & h_{34} & h_{35} \\ & & h_{43} & h_{44} & h_{45} \\ & & & h_{54} & h_{55} \\ & & & & h_{65} \end{pmatrix}, \bar{g}_0 = \begin{pmatrix} \beta \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}. \tag{3.8}$$

After 5 rotations are transformed into:

$$\bar{H}_5^{(5)} = \begin{pmatrix} h_{11}^{(5)} & h_{12}^{(5)} & h_{13}^{(5)} & h_{14}^{(5)} & h_{15}^{(5)} \\ & h_{22}^{(5)} & h_{23}^{(5)} & h_{24}^{(5)} & h_{25}^{(5)} \\ & & h_{33}^{(5)} & h_{34}^{(5)} & h_{35}^{(5)} \\ & & & h_{44}^{(5)} & h_{45}^{(5)} \\ & & & & h_{55}^{(5)} \\ & & & & 0 \end{pmatrix}, \quad \bar{g}_5 = \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \vdots \\ \gamma_6 \end{pmatrix}. \tag{3.9}$$

Defining Q_m as the product of the plane rotations Ω_i:

$$Q_m = \Omega_m \Omega_{m-1} \dots \Omega_1 \tag{3.10}$$

and

$$\bar{R}_m = \bar{H}_m^{(m)} = Q_m \bar{H}_m \tag{3.11}$$

$$\bar{g}_m = Q_m(\beta e_1) = (\gamma_1, \dots, \gamma_{m+1})^T \tag{3.12}$$

so

$$\min \|\beta e_1 - \bar{H}_m y\|_2 = \min \|\bar{g}_m - \bar{R}_m y\|_2 \tag{3.13}$$

The solution to the above least-squares problem is obtained by simply solving the triangular system resulting from deleting the last row of the matrix $\bar{R}_m$ and right-hand side $\bar{g}_m$. In addition, it is clear that for the solution y_*, the "residual" $\|\beta e_1 - \bar{H}_m y_*\|_2$ is nothing but the last element of the right-hand side, i.e., the term γ_6. In the GMRES algorithm, the only possibility of breakdown is in the Arnoldi loop, if the $w_j = 0$, i.e., when $h_{j1,j=0}$ at a given step j. In this situation, the algorithm stops, because it cannot generate the next orthonormal vector. In fact, this is not a problem but the opposite, as it means that the residual is zero and that we will converge at this step. In this thesis the Restarted GMRES is used.

Algorithm 6 Restarted GMRES

```
1: Compute $r_0 = b - Ax_0$, $\beta = \|r_0\|_2$, and $v_1 = \frac{r_0}{\beta}$
2: Generate the Arnoldi basis and the matrix $\bar{H}_m$ using the Arnoldi algorithm starting with $v_1$
3: Compute $y_m$ which minimizes $\|\beta e_1 - \bar{H}_m y\|_2$ and $x_m = x_0 + V_m y_m$
4: if satisfied then
5:     Stop
6: else
7:     Set $x_0 := x_m$ and GoTo 1
8: end if
```

3.5 Preconditioning

Although, GMRES is well founded theoretically, it can suffer from slow convergence or even divergence problems. Preconditioning is a key ingredient for the success of Krylov subspace methods like GMRES as both the efficiency and robustness of the solver can be improved. The idea behind Preconditioning, is transforming the original linear system into one which has the same solution, but which is likely to be easier to solve with an iterative solver. In general, the reliability of iterative techniques, when dealing with various applications, depends much more on the quality of the preconditioner than on the particular Krylov subspace method used. The first step is to find the preconditioning matrix M. The matrix M can be defined in many different ways but it must satisfy a few minimal requirements. From a practical point of view, the most requirement for M is that it is inexpensive to solve linear systems $Mx = b$. This is because the preconditioned GMRES will require a linear system solution with the matrix M at each step. Also M should be close to A in some sense and it should clearly be nonsingular. Once the preconditioning matrix M is available, the way it will be applied has to be chosen. There are three ways to apply the preconditioner: left-preconditioning:

$$M^{-1}Ax = M^{-1}b \tag{3.14}$$

right-preconditioning:

$$AM^{-1}u = b, x = M^{-1}u \tag{3.15}$$

and split-preconditioning (if the preconditioning matrix M is in the form of $M = M_L M_R$):

$$M_L^{-1}AM_R^{-1}x = M_L^{-1}b, x = M_R^{-1}u \tag{3.16}$$

In this thesis, left preconditioning is used.

3.5.1 Fixed Point Iteration based Preconditioners

A Fixed Point Iteration for solving a linear system takes the general form of:

$$x_{k+1} = M^{-1}Nx_k + M^{-1}b \tag{3.17}$$

where M and N realize the splitting of A into:

$$A = M - N \tag{3.18}$$

The above iteration is of the form:

$$x_{k+1} = Gx_k + \beta \tag{3.19}$$

where $\beta = M^{-1}b$ and G is the Iteration matrix as defined below:

$$G = M^{-1}N = M^{-1}(M - 1) = I - M^{-1}A \tag{3.20}$$

In this thesis, Jacobi (or Diagonal) and Gauss Seidel preconditioning have been used in order to accelerate the convergence of GMRES. The iteration matrices are shown below:

$$G_J(A) = I - D^{-1}A \tag{3.21}$$

$$G_{GS}(A) = I - (D + L)^{-1}A \tag{3.22}$$

where $A = D + L + U$, D is the Diagonal of matrix A, L is the lower triangular and U is the upper triangular one. So using the expression for G of eq. (3.17) the preconditioning matrices M are:

$$M_J = D \tag{3.23}$$

$$M_{GS} = D + L \tag{3.24}$$

We can further improve convergence,by applying relaxation to the preconditioner, for example in the case of Gauss Seidel Preconditioner the preconditioning matrix is transformed into:

$$M_{SOR} = (1/\omega)(D + \omega L) \tag{3.25}$$

where SOR denotes the Succesive over Relaxation method and ω is the relaxation factor.

3.5.2 Left-Preconditioned GMRES

In this thesis Left-Preconditioned is applied as mentioned above. Thus the linear system to be solved is transformed into:

$$M^{-1}Ax = M^{-1}b \tag{3.26}$$

The Arnoldi loop construct an orthogonal basis of the left preconditioned Krylov subspace as shown below:

$$K_m(A, M, r_0) = span[r_0, M^{-1}Ar_0, (M^{-1}A)^2 r_0, ..., (M^{-1}A)^{m-1} r_0] \tag{3.27}$$

The residual vectors and their norms calculated by the algorithm correspond to the preconditioned residuals, specifically $z_m = M^{-1}(b - Ax_m)$,rather than the original (unpreconditioned) residuals $b - Ax_m$. Moreover, the unpreconditioned residuals aren't directly accessible unless explicitly computed, such as by multiplying the preconditioned residuals by M. This can present challenges if a stopping criterion based on the actual residuals, rather than the preconditioned ones, is required.In algorithm 7 the Left Preconditioned GMRES algorithm is shown.

Algorithm 7 Left Preconditioned GMRES

```
1: Compute $r_0 = M^{-1}(b - Ax_0)$, $\beta := \|r_0\|_2$, and $v_1 := \frac{r_0}{\beta}$
2: for $j = 1, 2, \ldots, m$ do
3:     Compute $w_j := M^{-1}Av_j$
4:     for $i = 1, \ldots, j$ do
5:         $h_{ij} := (w_j, v_i)$
6:         $w_j := w_j - h_{ij}v_i$
7:     end for
8:     $h_{j+1,j} = \|w_j\|_2$
9:     if $h_{j+1,j} = 0$ then
10:        Set $m := j$ and go to step 14
11:    end if
12:    $v_{j+1} = \frac{w_j}{h_{j+1,j}}$
13: end for
14: Define the $(m+1) \times m$ Hessenberg matrix $\bar{H}_m = \{h_{ij}\}_{1 \le i \le m+1, 1 \le j \le m}$
15: Compute $y_m$ the minimizer of $\|\beta e_1 - \bar{H}_m y\|_2$ and $x_m = x_0 + V_m y_m$
16: If satisfied Stop, else set $x_0 = x_m$ and go to step 1
```

3.6 The Coupled RBM Algorithmic Framework

In this section, the whole step-by-step algorithmic framework is presented. The whole procedure is split into 3 main processes. The pre-processing, the solution procedure and the post-processing. At first, the user has to provide the necessary information about the mesh. These are the mesh's dimension (2D or 3D), the number of nodes with the initial coordinates of each node, specified as (x, y) for 2D or (x, y, z) for 3D as well as the type of each node (using a flag) as below:

- flag 0: Boundary nodes with fixed positions throughout the procedure,
- flag 1: Boundary nodes with known displacements determined by external conditions (FSI, aerodynamic shape optimization)
- flag 2: The internal nodes, those with unknown final positions

Also the number of each element type in the mesh (e.g., triangle, quadrilateral, etc., in 2D or tetrahedron, pyramid, prism, hexahedron in 3D) has to be provided, as also the node sequences comprising each element, specified in the appropriate order. Finally the user (or the external tool, the optimization software for example) has to provide the known displacements of the type 1 nodes. These pieces ofinformation are provided via input files, different formats are available, as they will be transformed in the default one. The next step is for the user to define the algorithmic parameters, such as the method used (Implicit/Explicit or Linearized/Non-Linear), the convergence criteria, the GMRES iterations (inner iterations) for each Newton Iteration (outer loops), the GMRES parameter m etc. In the pre-processing, the input files are translated into the default format. Then the data structure is created, determining the topology and the connectivity of each node, and storing the necessary information (such as the distances of each node).

After the data structure is created the solution procedure takes place. The first step is the calculation of each node's weight. The next step is the creation and solution of the non-linear system. Starting with the initialization (all zero's if not defined otherwise) the Objective Function to be minimized is computed. This will be the right-hand side of our system. Then the Jacobian matrix is calculated, since in the case of Newton GMRES (N-GMRES) it's required. In case of the quasi Newton GMRES (qN-GMRES) the Jacobian is not required, but an approximation is calculated using Finite Differences. The cost of the quasi Newton approach is O(NI). where NI is the number of internal nodes, as it requires $3 \times NI$ in the case of 2D (or $6 \times NI$ in the case of 3D) function calls. So as the exact expression of the Jacobian is available the N-GMRES approach is prefered. After the initialization of the system, the Restarted GMRES algorithm is employed in order to solve the linearized system. The GMRES is not run till convergence, but instead a small number of "inner" iterations are applied and the Newton procedure starts again using the last updated solution. The convergence (or stopping) criteria is applied in the Newton iterations, also called outer loops in this thesis. The optimal selection of the number of inner iterations is case sensitive, but a typical value is 5.

When the system is converged (or the maximum Newton iterations defined by the user are reached) the final positions of the internal nodes are updated. Then, the output files are generated, in order to visualize the mesh or use it in the CFD Solver. The Coupled DWRBM algorithmic aramework is presented in the flowchart of fig. 3.4.

If the Linearized Coupled algorithm is chosen, the system is linear and no Newton iterations are required. The convergence criteria is the same, and the maximum iteration stopping criteria is applied to GMRES iterations (previously defined as inner). The Linearized Coupled DWRBM algorithmic framework is presented in fig. 3.5.

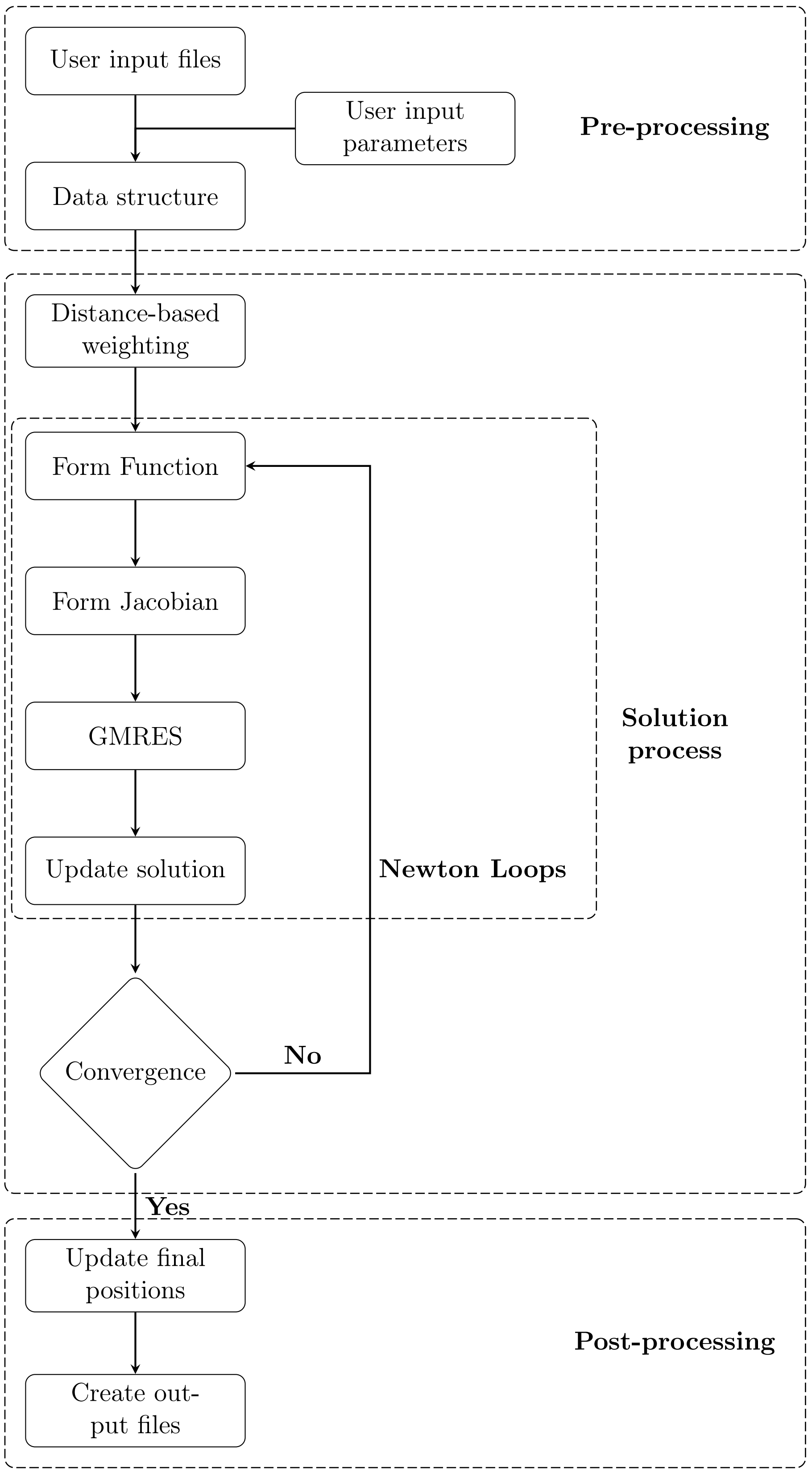


Figure 3.4: Implicit DWRBM algorithm framework.

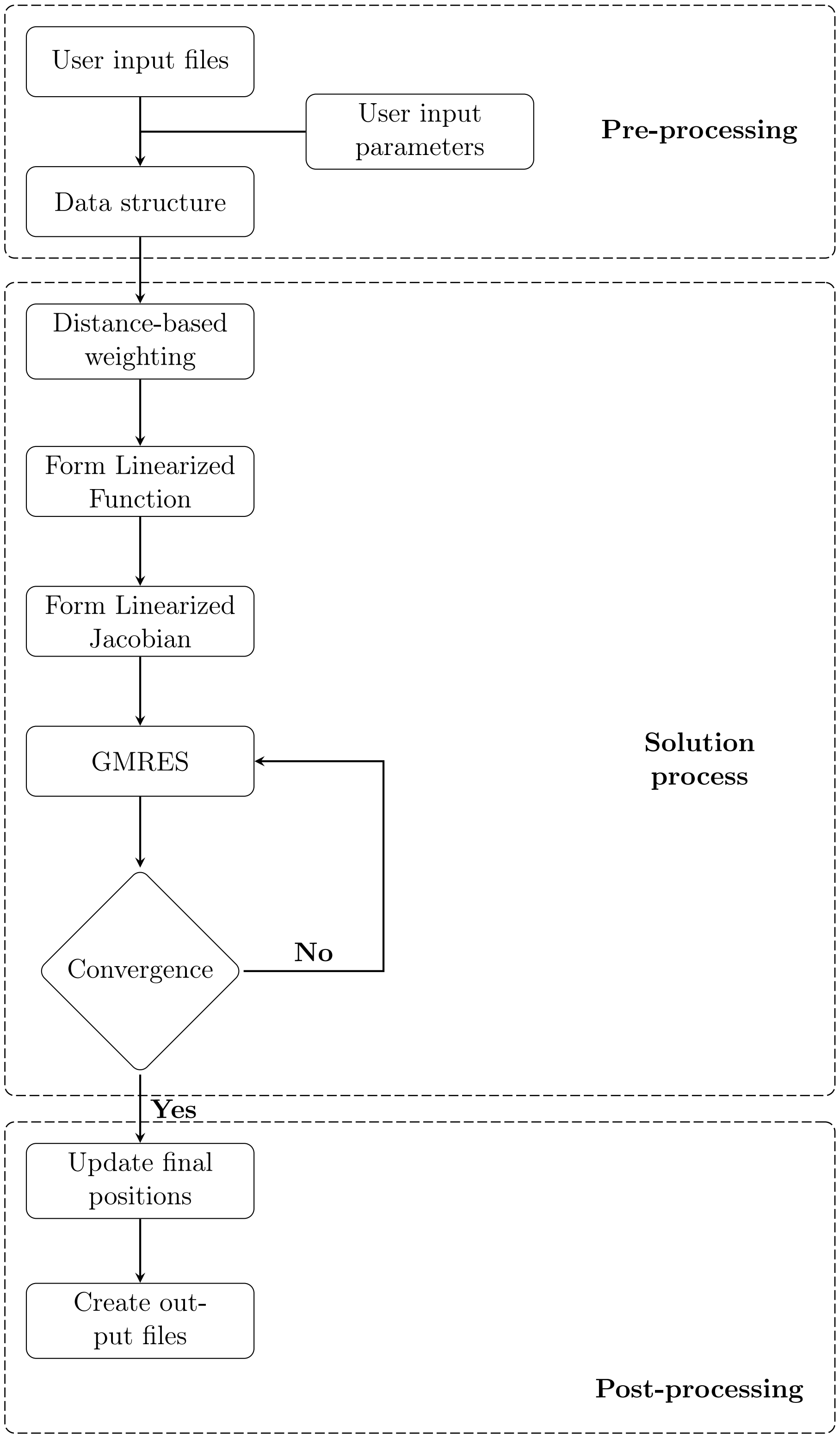


Figure 3.5: Implicit Linearized DWRBM algorithm framework.

Chapter 4

Applications

In this Chapter, the RBM algorithm is tested to assess its capabilities. It is tested in both fundamental cases, such as rotations and translations of squares and cubes, but also in real CFD applications, such as 2D airfoils and 3D wings. At first, the 2D cases are presented, starting with simple rotations of concentric squares. Then, more complex cases, such as combined rotation and translation of the inner square, take place and finally an airfoil is tested under different displacements. The final case of the airfoil is of crucial importance in evaluating the RBM algorithm in larger and harder cases. The difficulties arise from the finer mesh close to the airfoil, as high aspect ratio and skinny cells are prone to being inverted. Different settings, such as the weighting parameter or the coupled/decoupled algorithm, are compared based on the computational cost and the quality of the mesh. Then, as the algorithm is validated in the 2D cases, it is also tested in 3D ones. The first 3D case tested is the torsion of a 3D cube, consisting of a hybrid mesh (hexahedras, prisms, pyramids, tetrahedras) in order to evaluate our algorithm's capabilities. As our algorithm is tested in this basic case, more realistic cases are run, such as a 3D Wing bending and torsion and a realistic shape optimization case of a s-bend pipe. In the 3D cases, the algorithmic settings chosen are those of better performance in 2D.

4.1 2D Applications

4.1.1 Concentric squares rotation with free outer boundary

This first case, tests the algorithm's capabilities in a 2D structured mesh consisting of 96 quads. A rotation of 45° anti-clockwise around its center is applied to the inner square. The outer is free to move. The initial mesh and the ideal one are shown in fig. 4.1. Both the coupled and decoupled approaches are tested and the results are shown in fig. 4.2. In the coupled approach, no weighting is required and as it's shown in fig. 4.2 the concluding mesh is the same with the ideal. On the other hand, the decoupled algorithm didn't manage to displace the mesh ideally. The final mesh strongly depends on the order in which we loop through the unknown nodes.

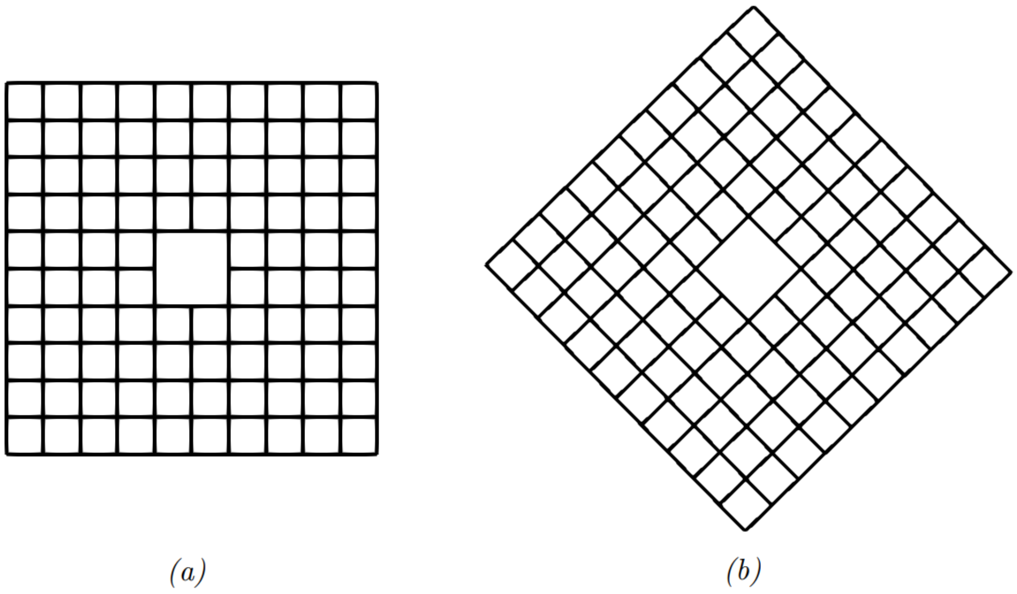


Figure 4.1: Concentric squares rotation with free outer boundary. (a): Initial mesh. (b): Ideal final mesh.

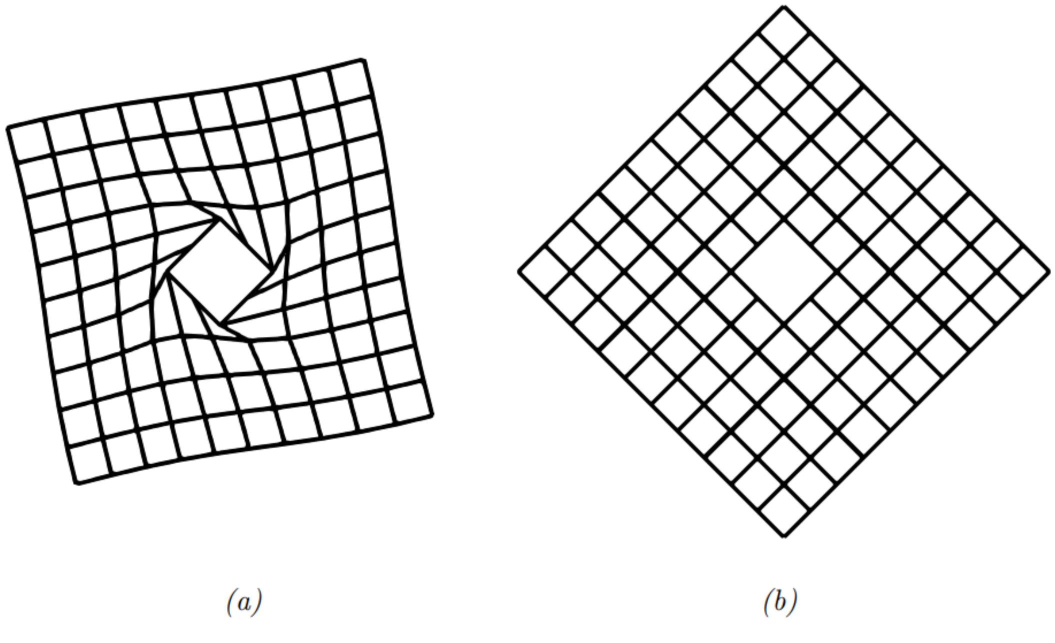


Figure 4.2: Concentric squares rotation with free outer boundary. Results of the (a) Decoupled and (b) Coupled approach.

4.1.2 Concentric squares with Fixed outer boundary

In this case, a finer mesh of two concentric squared is investigated. The mesh consists of 9936 quads (or 10152 nodes) and is shown in fig. 4.3. The outer square remains fixed while the inner one undergoes different displacements. These various displacements test the algorithm in more realistic cases and are also providing valuable information about the parameters of the algorithm.

Rotation of inner square

The first case is the rotation of the inner square anti-clockwise around its center. Different angles of rotation are tested until inverted cells occur. The max angle that does not produce inverted elements, is the quantity of interest to compare the algorithm's capabilities. Both the decoupled and the coupled approaches are tested, as well as the linearized coupled algorithm. The initial rotation angle is set to 15° and it is increasing in each sub-case until inverted cells occur. For GMRES, base m=5 is selected and the diagonal preconditioner is used. As presented, for the initial rotation both the coupled and decoupled approaches manage to displace the mesh with sufficient quality. In the decoupled approach, the algorithm requires 3 seconds to converge to the mesh shown in fig. 4.4. In the coupled approach, without using the weighting method, explained in section 3.3, the algorithm requires 14 seconds, resulting in a mesh of similar quality to that of the decoupled algorithm. The linearized coupled method give corresponding results, shown and compared to the Decoupled method in fig. 4.6. The run time was 6 seconds. Using weights the algorithm greatly improves the mesh close to the inner square. Different exponential parameters are tested, and the results are presented below.

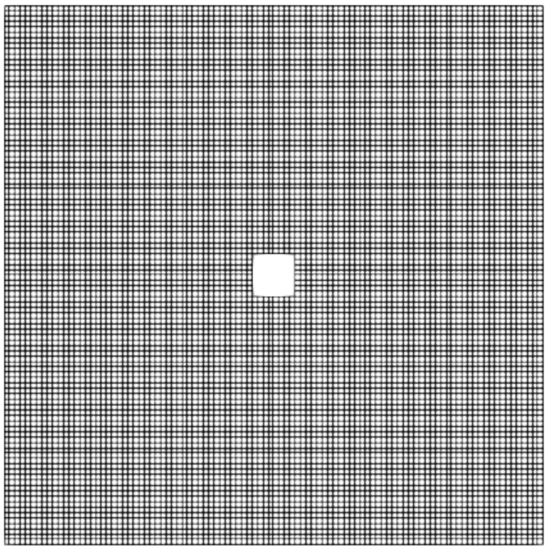


Figure 4.3: Two Concentric squares Case: Initial mesh.

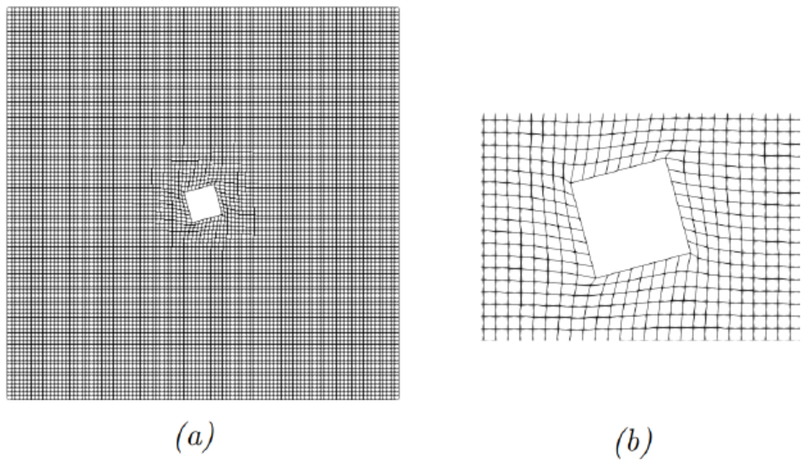


Figure 4.4: Two Concentric squares 15° rotation case: Decoupled approach Results. (a): Final mesh. (b): Zoom close to the inner square.

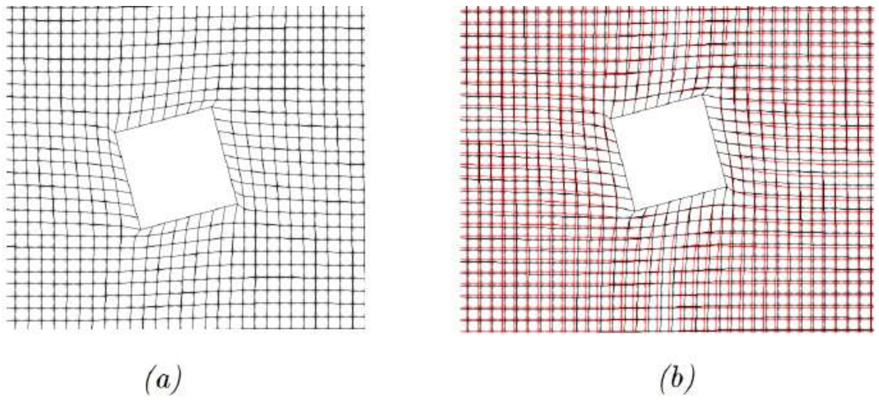


Figure 4.5: Two Concentric squares 15° rotation case: Coupled approach without the use of weighting. (a): Zoom of the final mesh close to the inner square. (b): Comparison between Coupled (black) and Decoupled (red) approach.

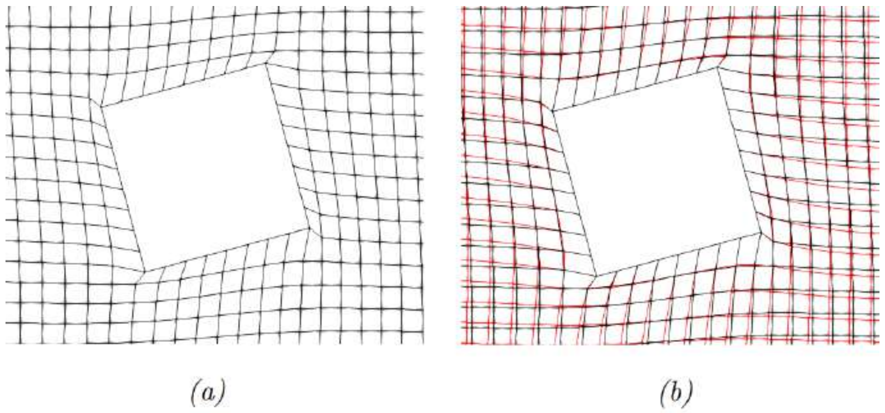


Figure 4.6: Two Concentric squares 15° rotation case: Linearized Coupled approach without the use of weighting. (a): Zoom of the final mesh close to the inner square. (b): Comparison between Linearized Coupled (black) and Decoupled (red) approach.

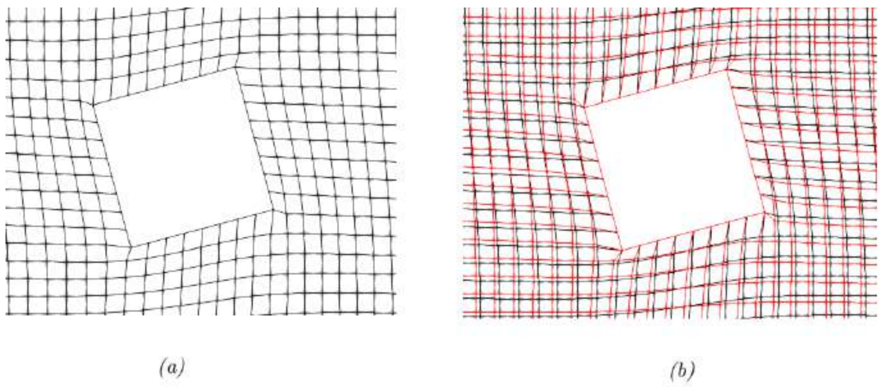


Figure 4.7: Two Concentric squares 15° rotation case: Weighted Coupled approach, p=1. (a): Zoom of the final mesh close to the inner square. (b): Comparison between Weighted Coupled (p=1)(black) and Coupled without weights (red).

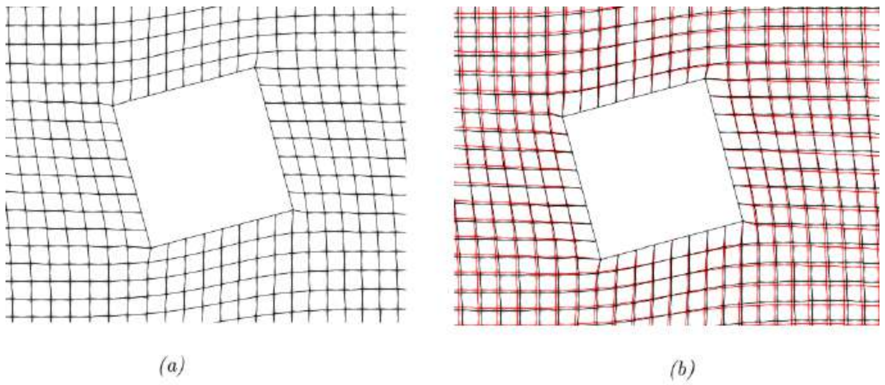


Figure 4.8: Two Concentric squares 15° rotation case: Weighted Coupled approach, p=2. (a): Zoom of the final mesh close to the inner square. (b): Comparison between Weighted Coupled (p=2) (black) and Weighted Coupled (p=1) (red).

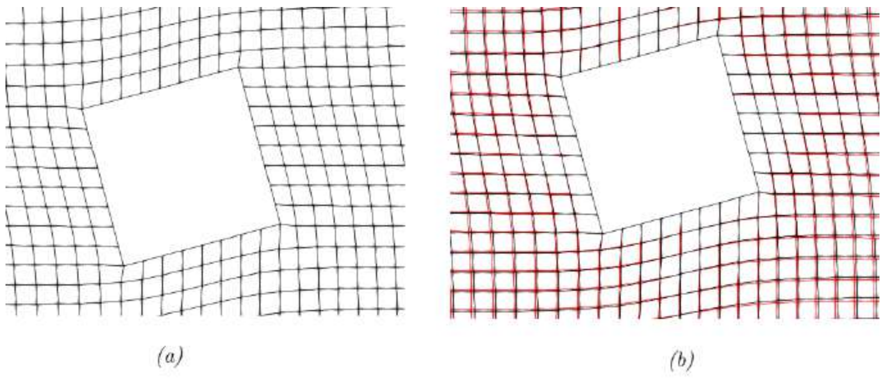


Figure 4.9: Two Concentric squares 15° rotation case: Weighted Coupled approach, p=4. (a): Zoom of the final mesh close to the inner square. (b): Comparison between Weighted Coupled (p=4) (black) and Weighted Coupled (p=2) (red).

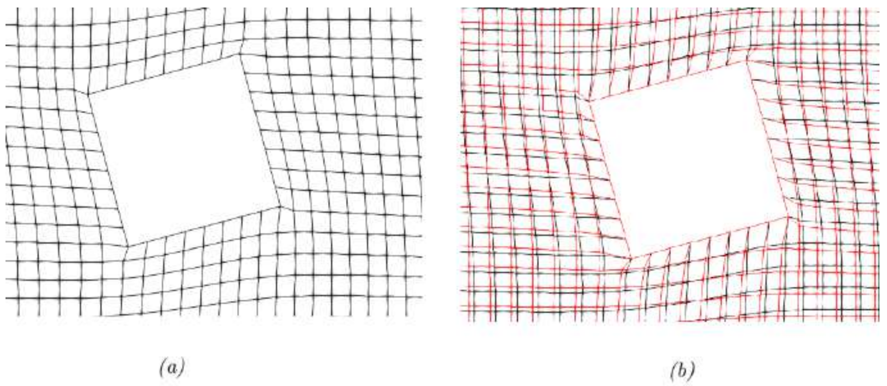


Figure 4.10: Two Concentric squares 15° rotation case: Linearized Weighted Coupled approach, p=1. (a): Zoom of the final mesh close to the inner square. (b): Comparison between Linearized Weighted Coupled (p=1)(black) and Linearized Coupled without weights (red).

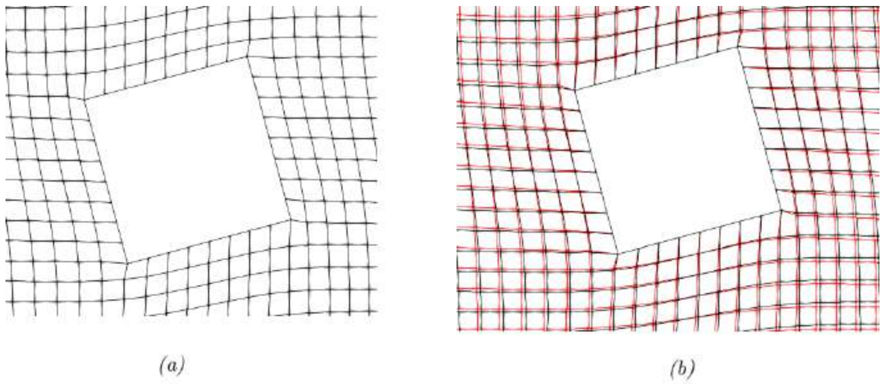


Figure 4.11: Two Concentric squares 15° rotation case: Linearized Weighted Coupled approach, p=2. (a): Zoom of the final mesh close to the inner square. (b): Comparison between Linearized Weighted Coupled (p=2) (black) and Linearized Weighted Coupled (p=1) (red).

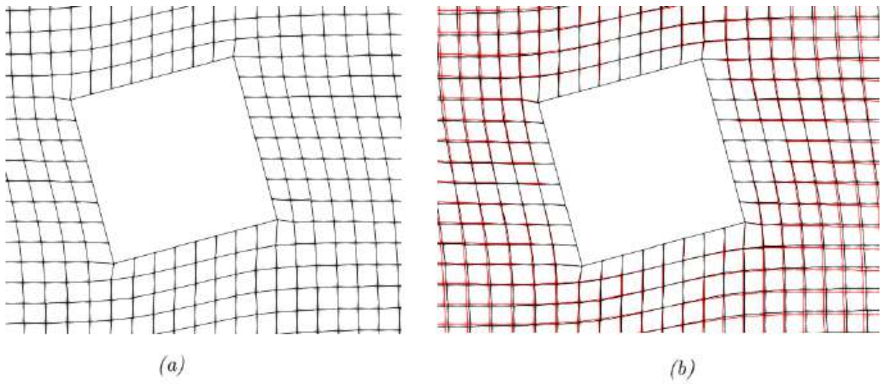
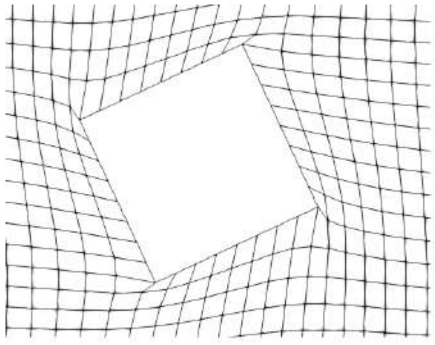
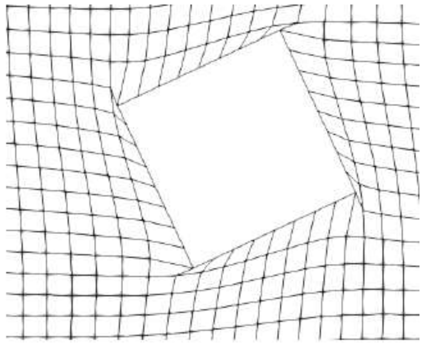
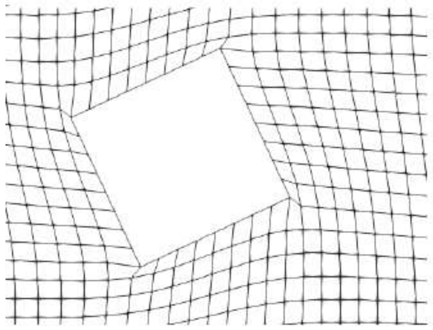
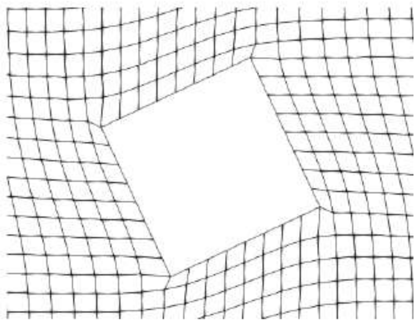


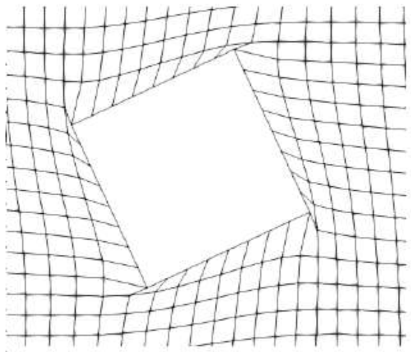
Figure 4.12: Two Concentric squares 15° rotation case: Linearized Weighted Coupled approach, p=4. (a): Zoom of the final mesh close to the inner square. (b): Comparison between Linearized Weighted Coupled (p=4) (black) and Linearized Weighted Coupled (p=2) (red).

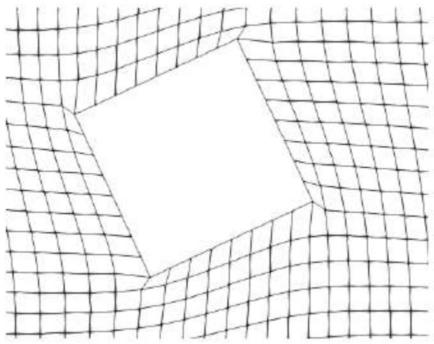

(a)

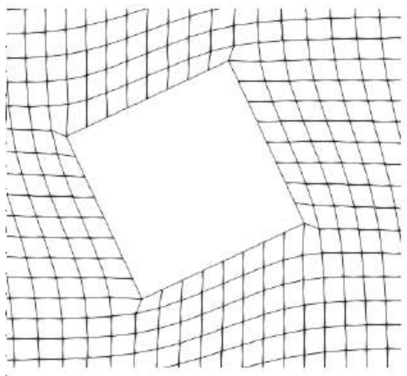

(b)


(c)


(d)


(e)


(f)


(g)

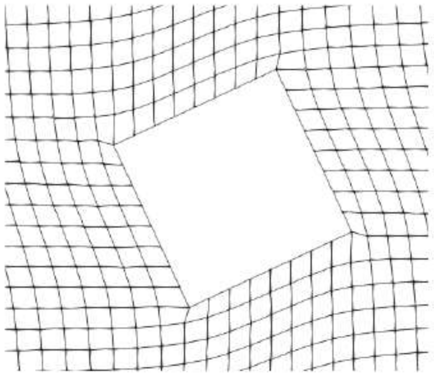

(h)

Figure 4.13: Two Concentric squares 25° rotation case compared results. (a) Decoupled approach. (b): Linearized Coupled approach. (c): Linearized Weighted Coupled approach, p=1. (d): Linearized Weighted Coupled approach, p=2. (e): Coupled approach, no weighting. (f): Weighted Coupled approach, p=1. (g): Weighted Coupled approach, p=2. (h): Weighted Coupled approach, p=4.

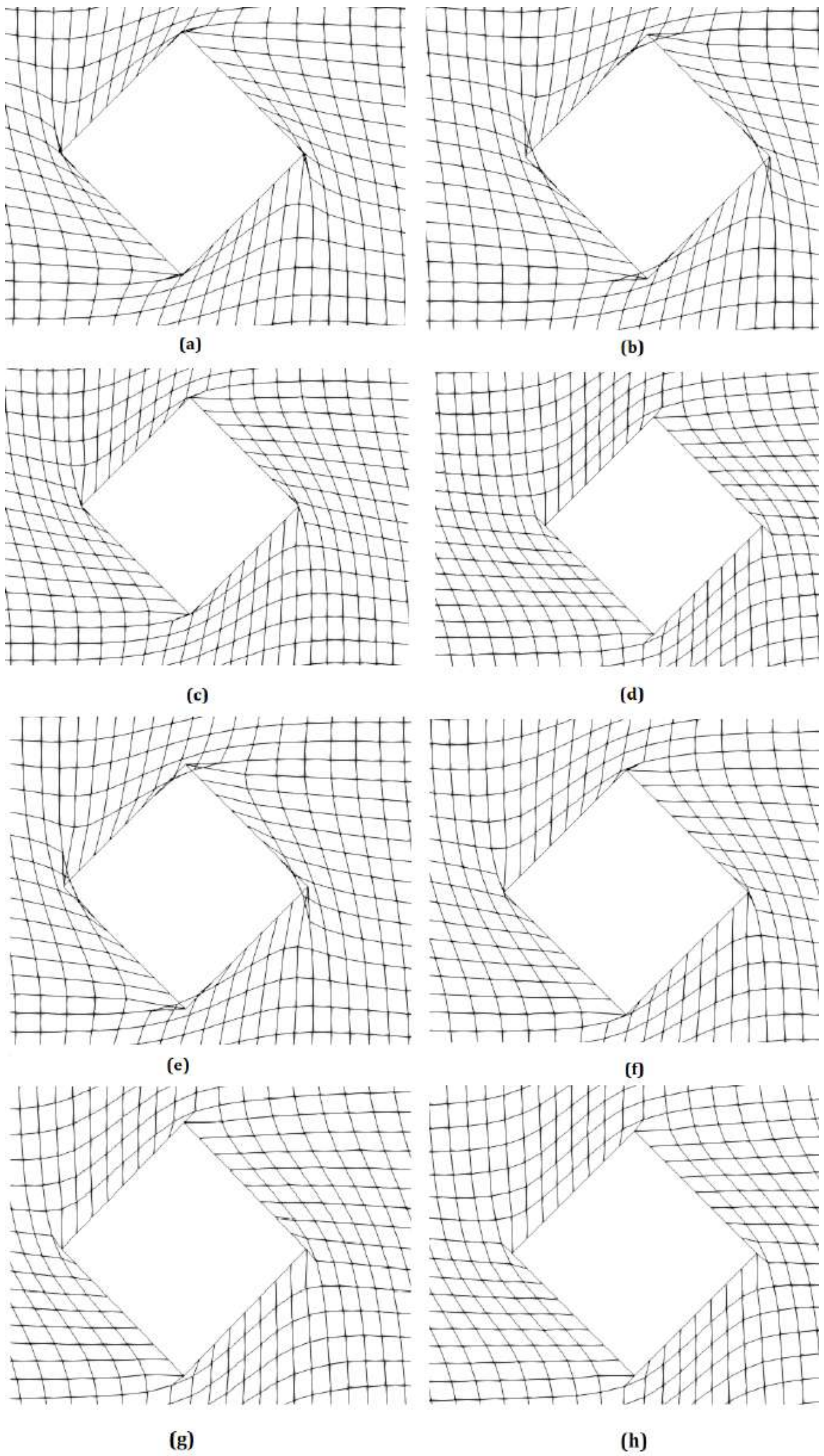


Figure 4.14: Two Concentric squares 45° rotation case compared results. (a) Decoupled approach. (b): Linearized Coupled approach. (c): Linearized Weighted Coupled approach, p=1. (d): Linearized Weighted Coupled approach, p=4. (e): Coupled approach, no weighting. (f): Weighted Coupled approach, p=1. (g): Weighted Coupled approach, p=2. (h): Weighted Coupled approach, p=4.

As it's shown in fig. 4.14 the use of weights is necessary, especially in large displacements, ensuring high quality mesh close to the boundaries. In this example, values above p=4 did not affect the results. The comparison of the methods is shown in table 4.1:

Table 4.1: Max rotation angle to avoid ivnerted cells for each method.

Method	**Max rotation angle**
Decoupled	25°
Linearized Coupled, no weights	25°
Linearized Coupled, p=1	35°
Linearized Coupled, p=2	45°
Linearized Coupled, p=4	55°
Coupled, no weights	25°
Coupled, p=1	35°
Coupled, p=2	45°
Coupled, p=4	55°

This case revealed that the linearized coupled algorithm, with the use of weights is superior. Even in large displacements, the final mesh is of same quality with that of the non-linear coupled one, in significantly lower cost (in the case of 55° rotation, the linearized one required just 6 seconds, while the non linear required 11). A comparison between the 2 resulting meshes is shown in fig. 4.15.

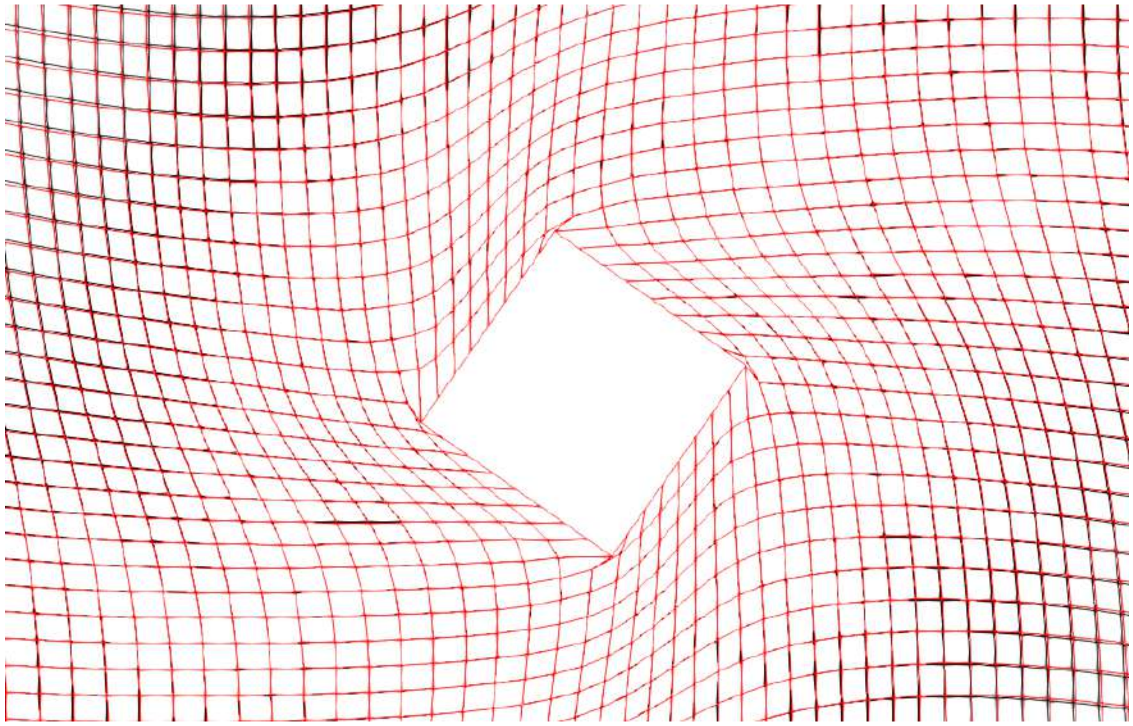


Figure 4.15: Two Concentric squares 55° rotation case compared results. Linearized Weighted Coupled, p=4 (red), Non-linear Weighted Coupled, p=4 (black).

Combined rotation and Translation of inner square

In this case, the combined rotation and translation of the inner square is investigated. Only the Coupled approach is tested, and the aim of this case is to study the sensitivity of the weighting parameter (the exponential parameter p of eq. (3.1)). The initial mesh is the same with this in fig. 4.3. The inner square is rotated 20° anti-clockwise and translated two times the length of its side in each direction. The results are shown below:

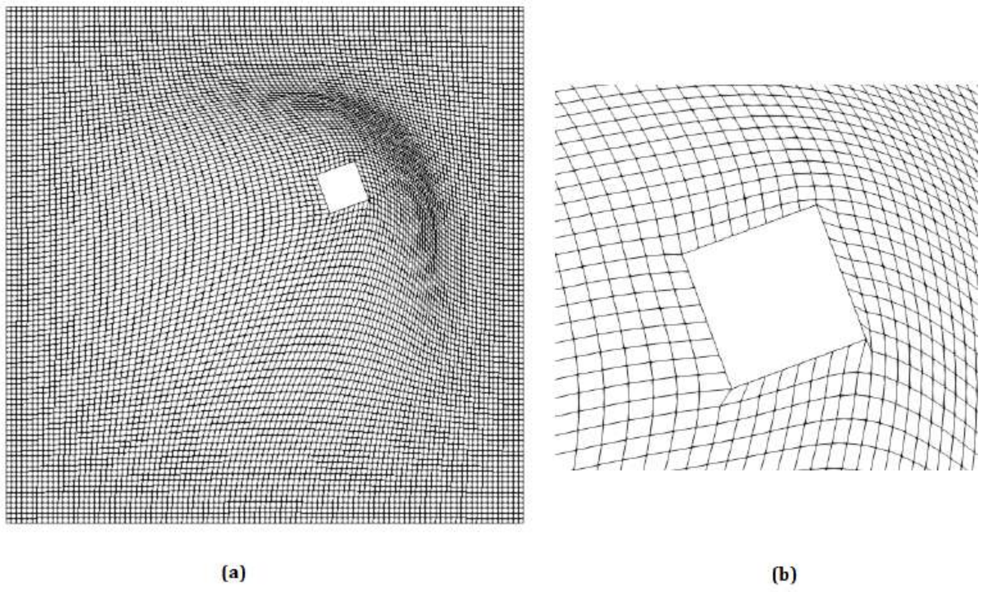


Figure 4.16: Two Concentric squares. Combined rotation and translation case. Weighted Coupled method with p=1. (a): Final mesh. (b): zoom near the inner square.

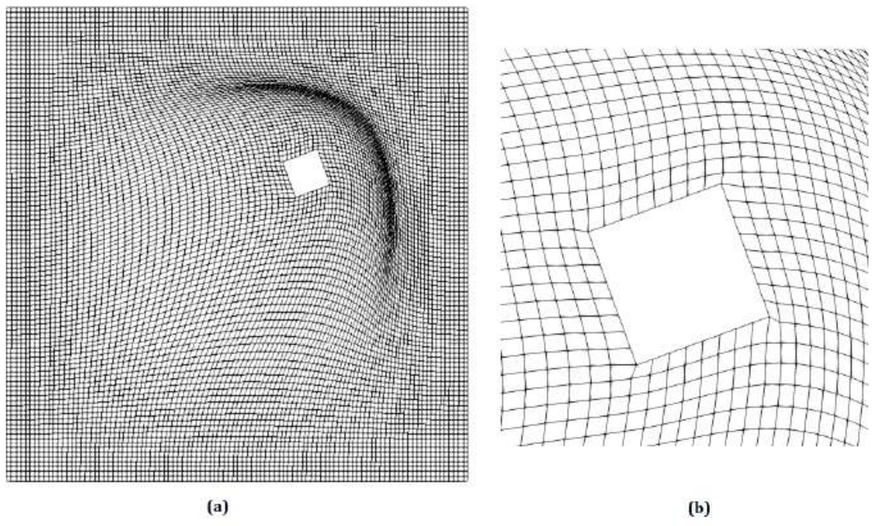


Figure 4.17: Two Concentric squares. Combined rotation and translation case. Weighted Coupled method with p=1.5. (a): Final mesh. (b): zoom near the inner square.

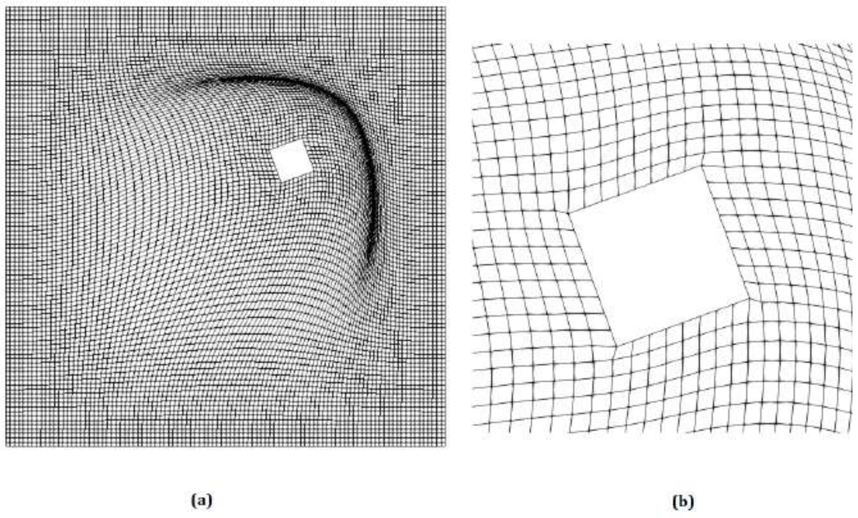


Figure 4.18: Two Concentric squares. Combined rotation and translation case. Weighted Coupled method with p=2. (a): Final mesh. (b): zoom near the inner square.

As shown in figures 4.13-15, increasing the value of the exponential parameter p enhances the mesh quality near the inner square. However, it can lead to issues in the far-field mesh. This is clearly illustrated in fig. 4.18, where the far-field mesh becomes unsuitable for CFD applications due to the presence of inverted cells, despite the mesh near the inner square retaining almost the same quality as the initial one. These issues arise because the parameter is designed to improve the mesh near the boundaries, resulting in most of the displacement being concentrated in the far-field region. A comparison between p=2 and p=1.5 is presented in fig. 4.19.

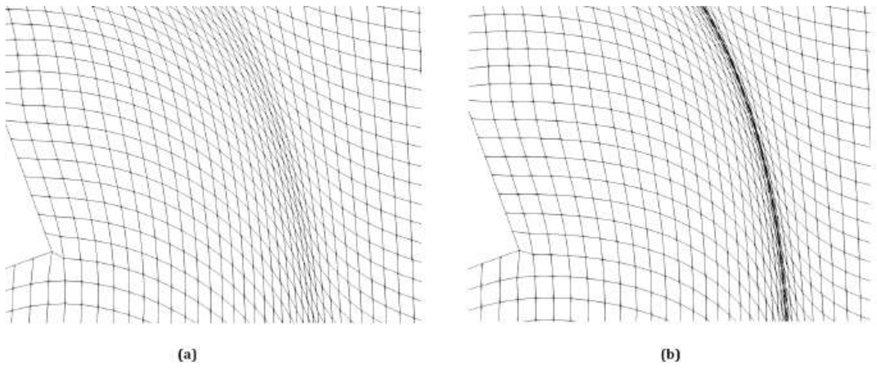


Figure 4.19: Two Concentric squares. Combined rotation and translation case. Zoom in far-field mesh (a): p=1.5. (b): p=2.

4.1.3 Isolated Airfoil

In this case, a more CFD-oriented application is considered: the rotation of an isolated airfoil. The center of rotation is set at one of three locations: the aerodynamic center, the leading edge, or the trailing edge. The airfoil has a unit length, and the mesh consists of 2242 nodes and 4383 triangles. This mesh is generated for inviscid flows, as no streched cells close to the wall were included in its generation. The initial mesh is shown in fig. 4.20.

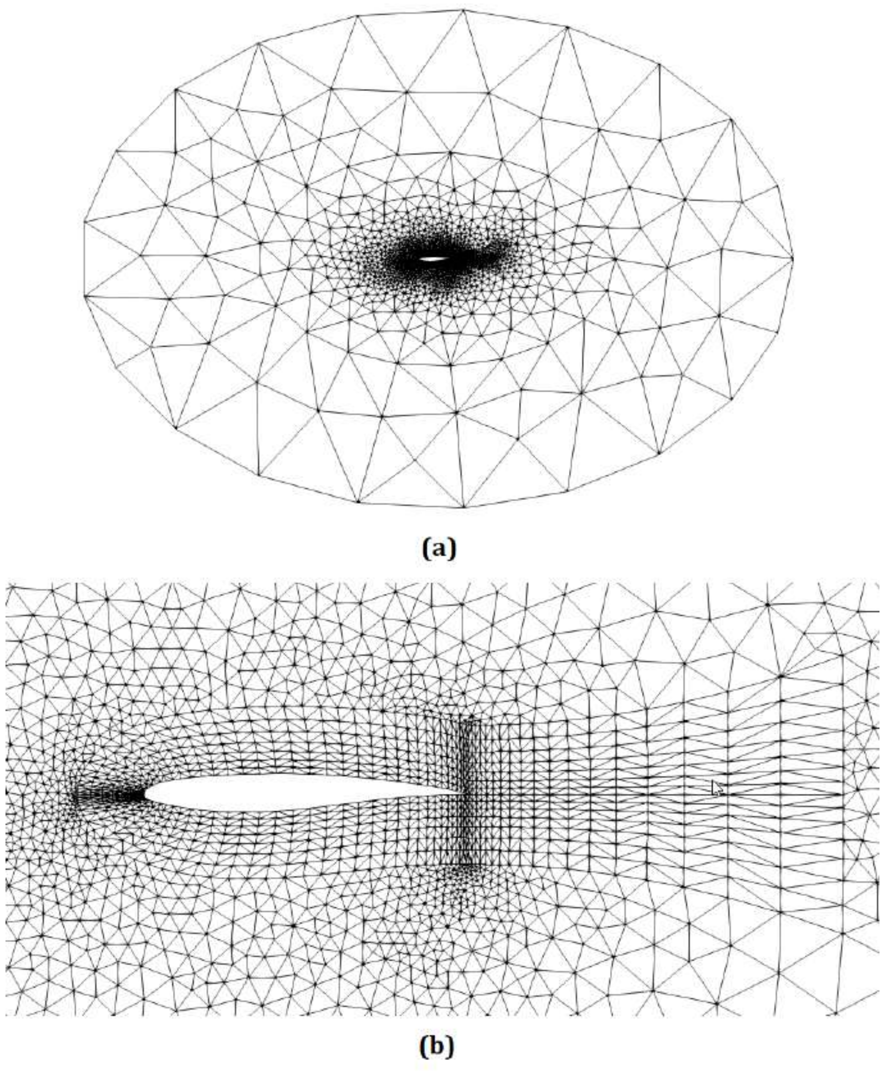


Figure 4.20: Isolated airfoil case (a): Initial mesh. (b): Zoom in airfoil.

Both the non-linear and the linearized coupled algorithms are tested. In table 4.2, the algorithmic settings are presented. In case of the linearized one, no outer iterations are used, and the max iterations criteria is applied to the GMRES instead.

Table 4.2: DWRBM Algorithm Settings

Method	**Non Linear & Linearized Coupled**
Pre-conditioner	Diagonal-Gauss Seidel
Stopping criteria	Relative residual$<10^{-3}$
Max Newton (outer) Iterations	100
GMRES base m	10-20
GMRES (inner) Iterations	3
Weighting Parameter value	various

Convergence Study

The GMRES convergence is evaluated in this section. The two preconditioners are compared and the GMRES was run with two different bases m, 10 and 20. The convergence, in the case of 25° rotation for the non-linear algorithm, is presented in fig. 4.21. The x-axis, in this figure, denotes the Newton (outer) iterations. The run took 9 seconds for the diagonal preconditioner with m=10 and 11 seconds for m=20. In case of the Gauss-Seidel preconditioner, the run took 11 seconds for m=10 and 21 seconds for m=20. So for the rest of the isolated airfoil cases, the diagonal preconditioner with m = 10 is used.

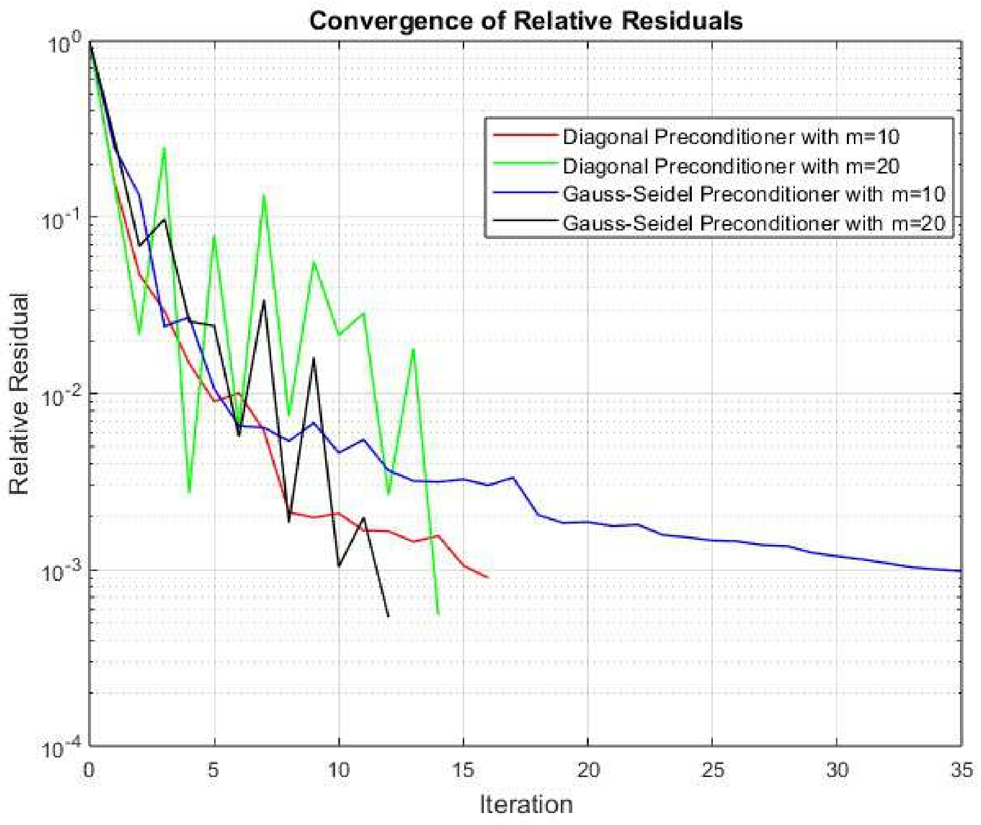


Figure 4.21: Convergence analysis of GMRES, non-linear coupled algorithm.

The same case is run with the linearized algorithm to assess its computational efficiency. For diagonal preconditioning, the run took 3 seconds for m=10 and 6 seconds for m=20. For diagonal preconditioning, the times were 4 and 13 seconds respectively. The convergence history is shown in fig. 4.22. The two final meshes are presented in fig. 4.23. As the linearized coupled algorithm is much faster, this algorithm is selected in the rest of the thesis.

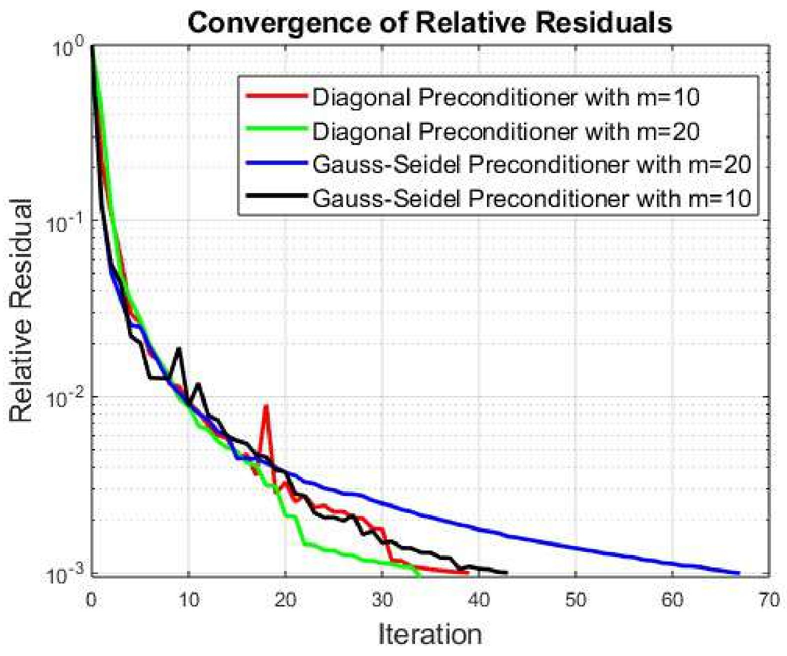


Figure 4.22: Convergence analysis of GMRES, linearized coupled algorithm.

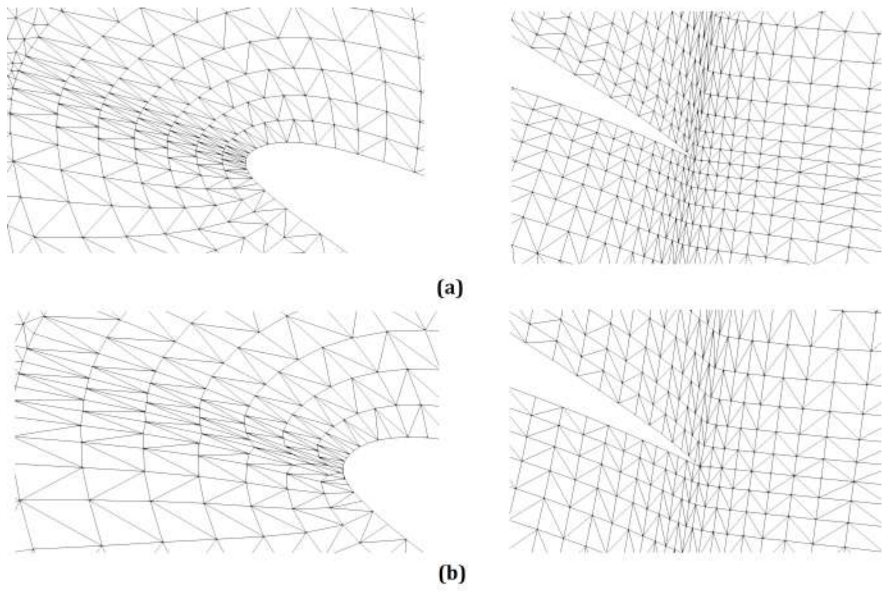


Figure 4.23: Isolated airfoil 25° rotation case. Final meshes. a) Linearized Weighted Coupled algorithm, p=4. b) Non-Linear Weighted Coupled algorithm, p=4.

Rotation around the Aerodynamic Center

In this case, the airfoil rotates around its aerodynamic center. The initial angle is set to 25° and it increases until inverted cells occur. Different values of the weighting parameter are evaluated and compared. The figures below present the results for the initial 25° case. The algorithm successfully adapted the mesh to the airfoil's rotation, resulting in a mesh of comparable quality to the initial configuration. As shown in fig. 4.24, the algorithm with p=2 and p=4 produced a higher quality mesh compared to that with p=1. For subsequent rotation angles, only the final mesh using the optimal parameter selection is presented. Also, it is observed that for rotation angles exceeding 60°, the use of sub-steps becomes necessary.

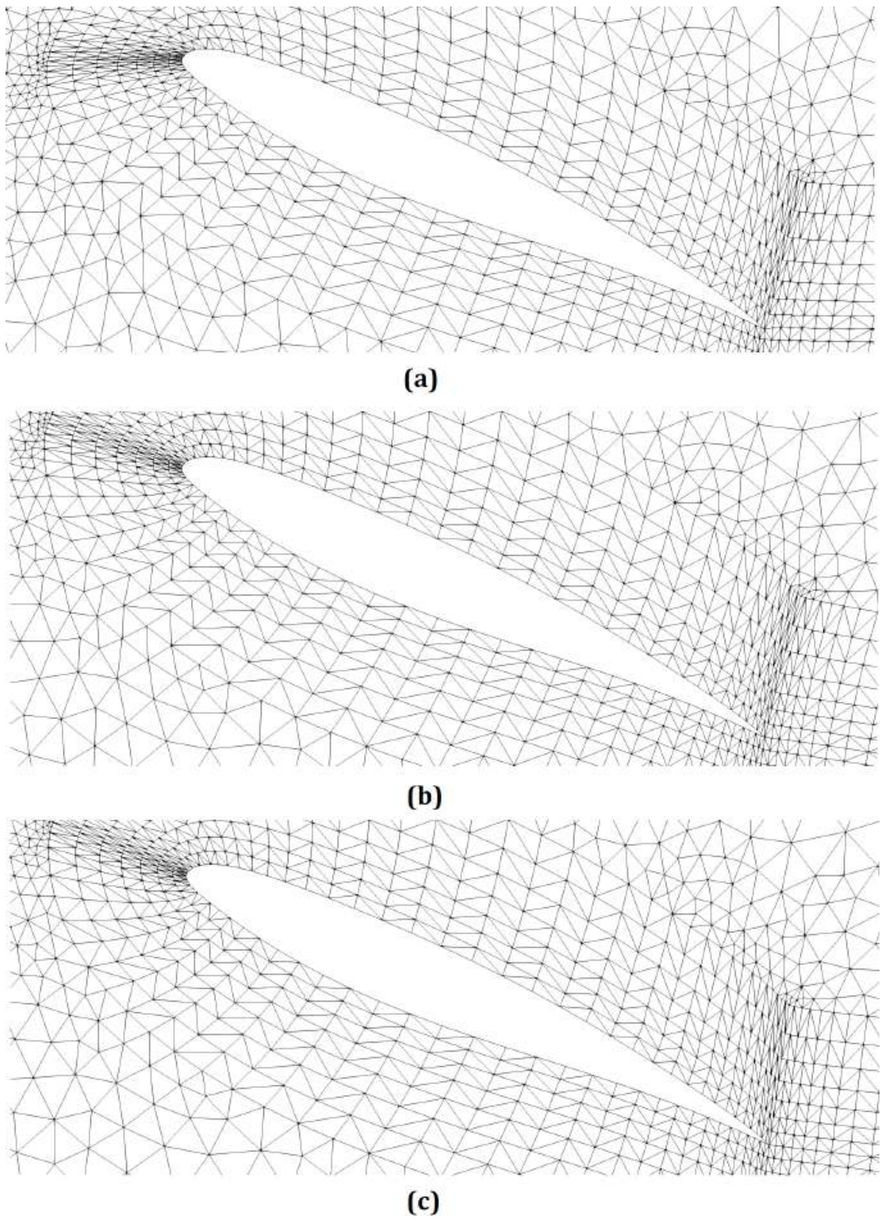


Figure 4.24: Isolated airfoil case: 25° rotation around the aerodynamic center. (a): p=1.(b): p=2. (c) p=4.

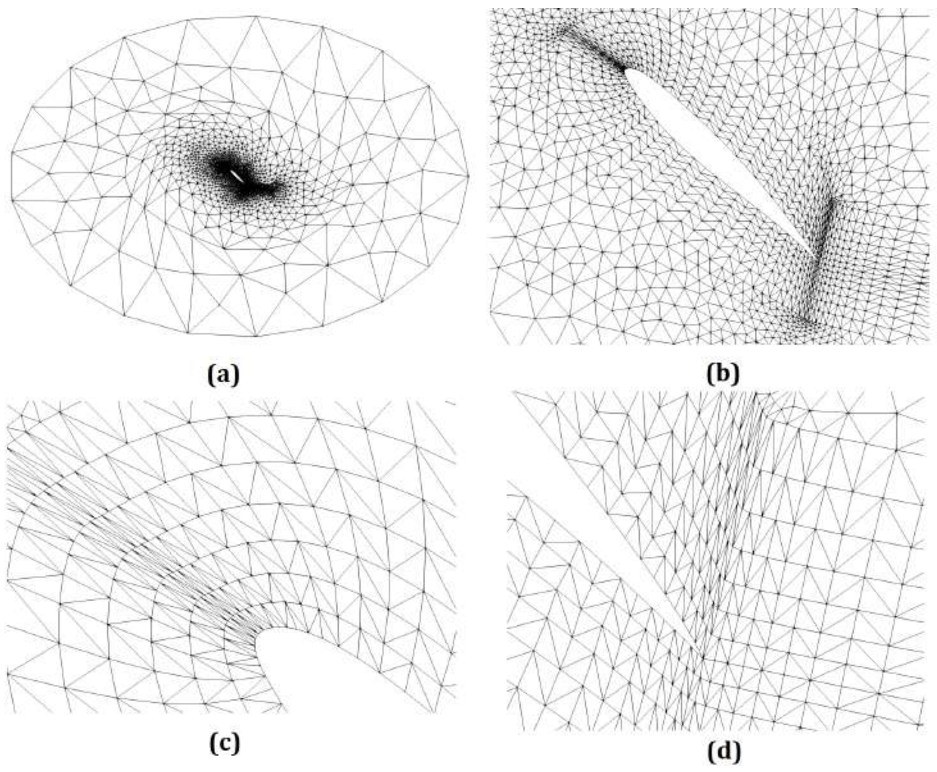


Figure 4.25: Isolated airfoil case: 45° rotation around the aerodynamic center, p=4. (a): Final mesh. (b): Zoom at the airfoil. (c): Zoom at the leading edge. (d): Zoom at the trailing edge.

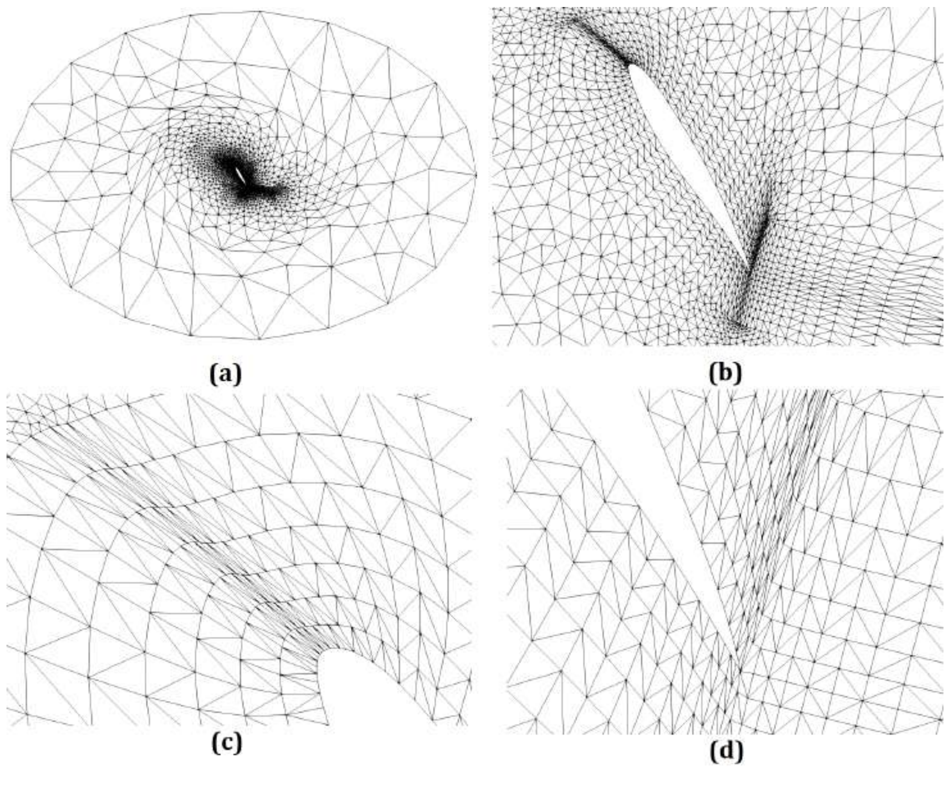


Figure 4.26: Isolated airfoil case: 60° rotation around the aerodynamic center, p=4. (a): Final mesh. (b): Zoom at the airfoil. (c): Zoom at the leading edge. (d): Zoom at the trailing edge.

As the rotation angle becomes higher, the linearized method fails. For angles larger than 60° the method requires the use of sub-steps, increasing its cost. Also from 75° and above even with sub-steps, the linearized method fails. The use of the non-linear coupled algorithm can still displace the mesh, with the use of sub-steps. The 90° rotation around the aerodynamic center is presented, using the non-linear coupled algorithm with 12 sub-steps.

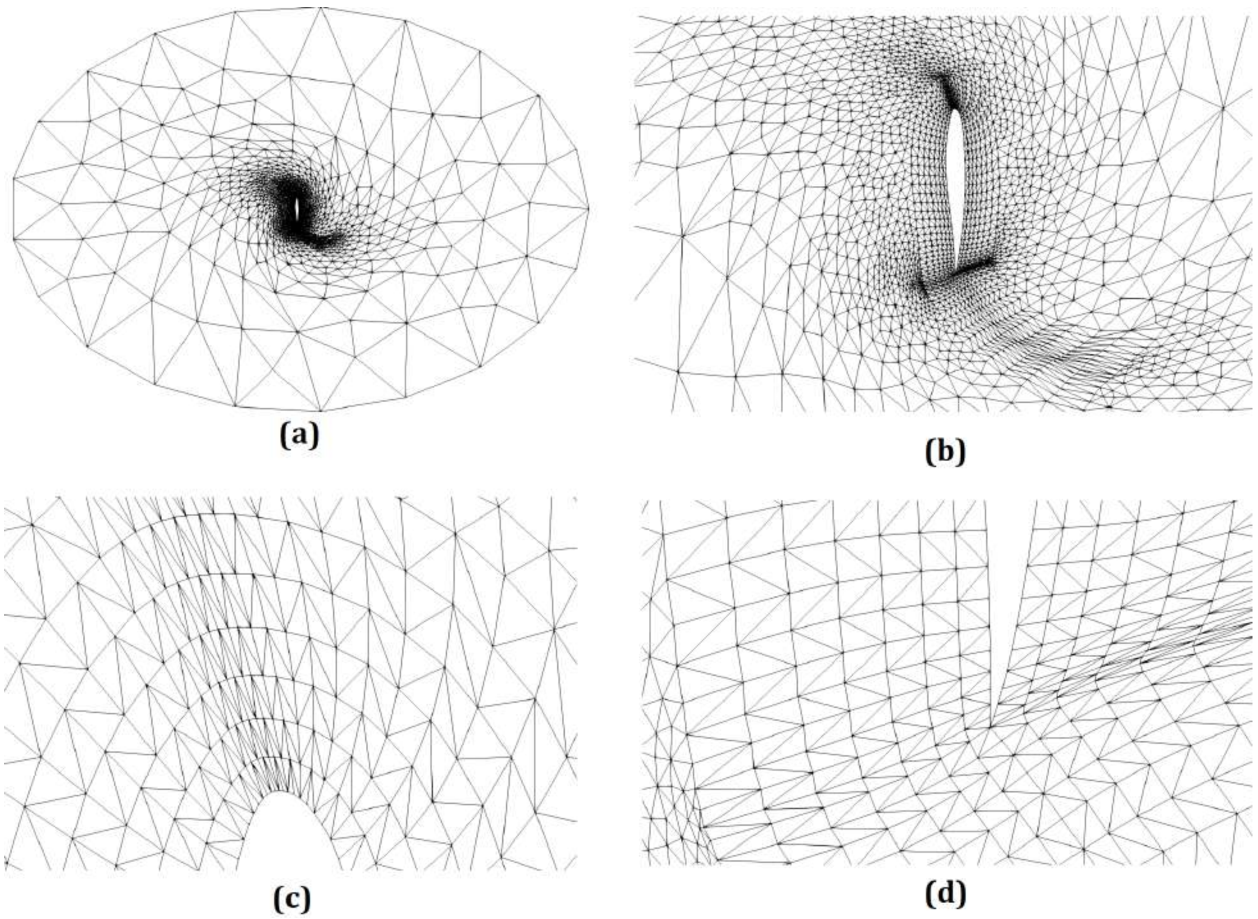


Figure 4.27: Isolated airfoil case: 90° rotation around the aerodynamic center, non-linear coupled algorithm with 12 sub-steps and p=4. (a): Final mesh. (b): Zoom at the airfoil. (c): Zoom at the leading edge. (d): Zoom at the trailing edge.

Rotation around the Leading Edge

In this case, the airfoil rotates around its leading edge. The linearized coupled algorithm successfully adapted the mesh to the airfoil's rotation, maintaining a quality comparable to the initial configuration. As in the previous scenario, the linearized algorithm did not manage to displace the mesh for rotation angles higher than 60° without the use of sub-steps. The extreme scenario of the 90° rotation is presented, using the non-linear coupled algorithm with 4 sub-steps. It was observed that while p=4 produced a higher-quality mesh near the airfoil, inverted cells appeared in the far field, as shown in fig. 4.29. Therefore, p=2 was selected as a compromise, achieving a balance between high-quality mesh near the airfoil and avoiding inverted cells.

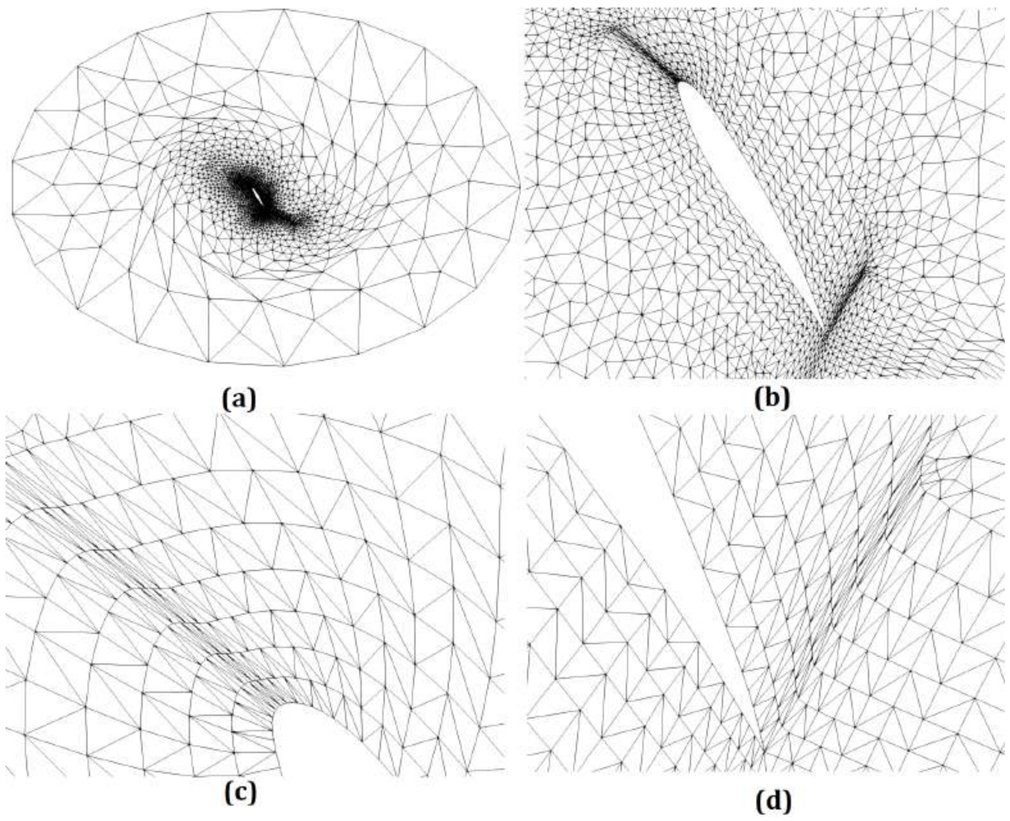


Figure 4.28: Isolated airfoil case: 60° rotation around the leading edge, linearized coupled algorithm without sub-steps and p=4. (a): Final mesh. (b): Zoom at the airfoil. (c): Zoom at the leading edge. (d): Zoom at the trailing edge.

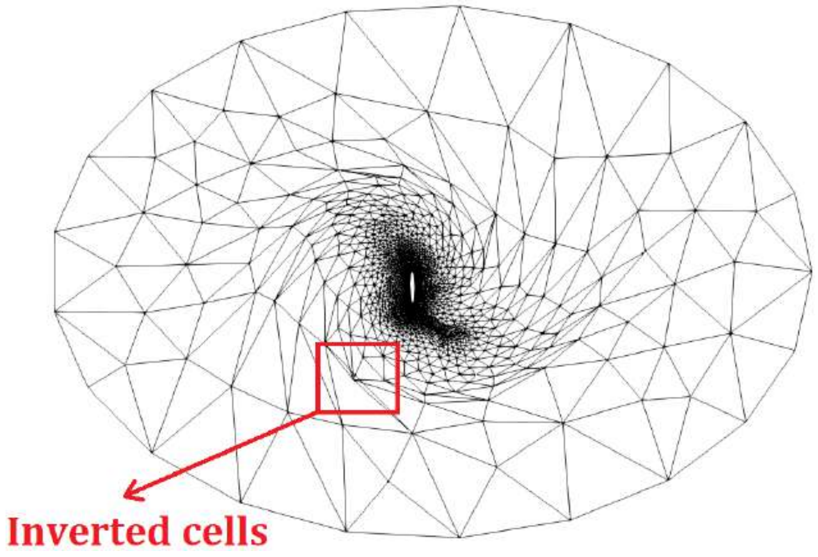


Figure 4.29: Isolated airfoil case: 90° rotation around the leading edge, non-linear coupled algorithm with 4 sub-steps and p=4. Final mesh.

As presented in fig. 4.29, the selection of p=4 resulted in the formation of inverted cells in the far field. This outcome stems from the fact that higher exponential weighting values prioritize mesh quality near the solid boundary by concentrating displacement into cells farther away from the boundaries. While this approach works well for small displacements, where far-field cells can tolerate this displacement due to their relatively regular shapes and lower aspect ratios compared to boundary-layer cells, it becomes problematic for large displacements.

In cases involving substantial deformations, a balance must be struck to preserve mesh quality near the airfoil while also preventing inverted cells in the far field. Achieving this balance is crucial, as inverted cells disrupt the solution process and hinder the optimization workflow. Thus, selecting an intermediate weighting parameter, such as p=2, ensures a compromise that maintains mesh quality around the airfoil and avoids inverted cells in the far field. The final mesh is presented in fig. 4.30.

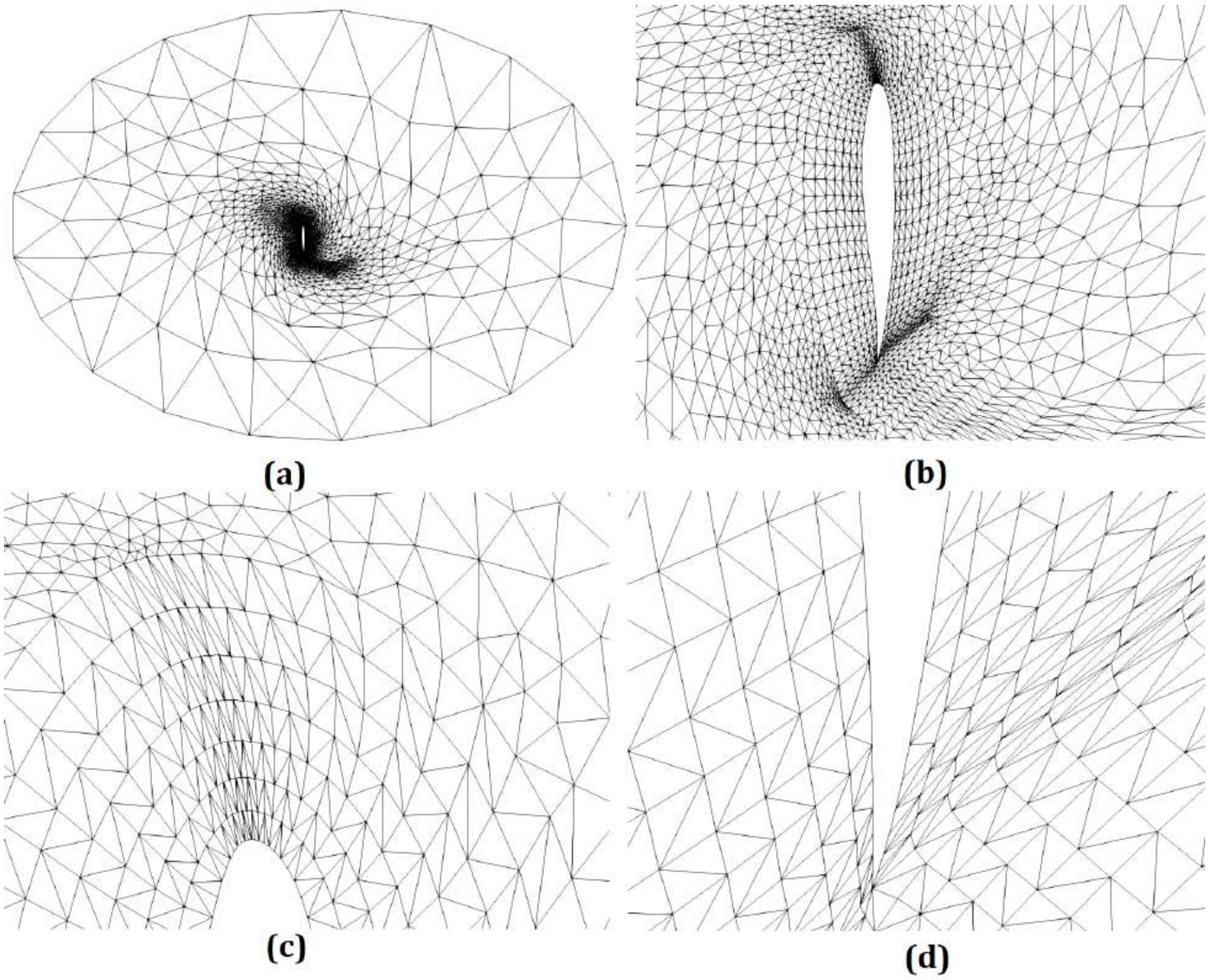


Figure 4.30: Isolated airfoil case: 90° rotation around the leading edge, non-linear coupled algorithm with 4 sub-steps and p=2. (a): Final mesh. (b): Zoom at the airfoil. (c): Zoom at the leading edge. (d): Zoom at the trailing edge.

Rotation around the Trailing Edge

In this case, the airfoil rotates around its trailing edge, which poses the most challenging scenario. The mesh near the trailing edge is significantly denser and consists of smaller elements, making it less capable of handling excessive compression. The linearized coupled algorithm is used, with p=2. For angles of rotation until 60°, no sub-steps are required. For 90° even with 10 sub-steps both linearized and non-linear algorithm, did not manage to displace the mesh. The resulting mesh of the 60° rotation is presented in fig. 4.31.

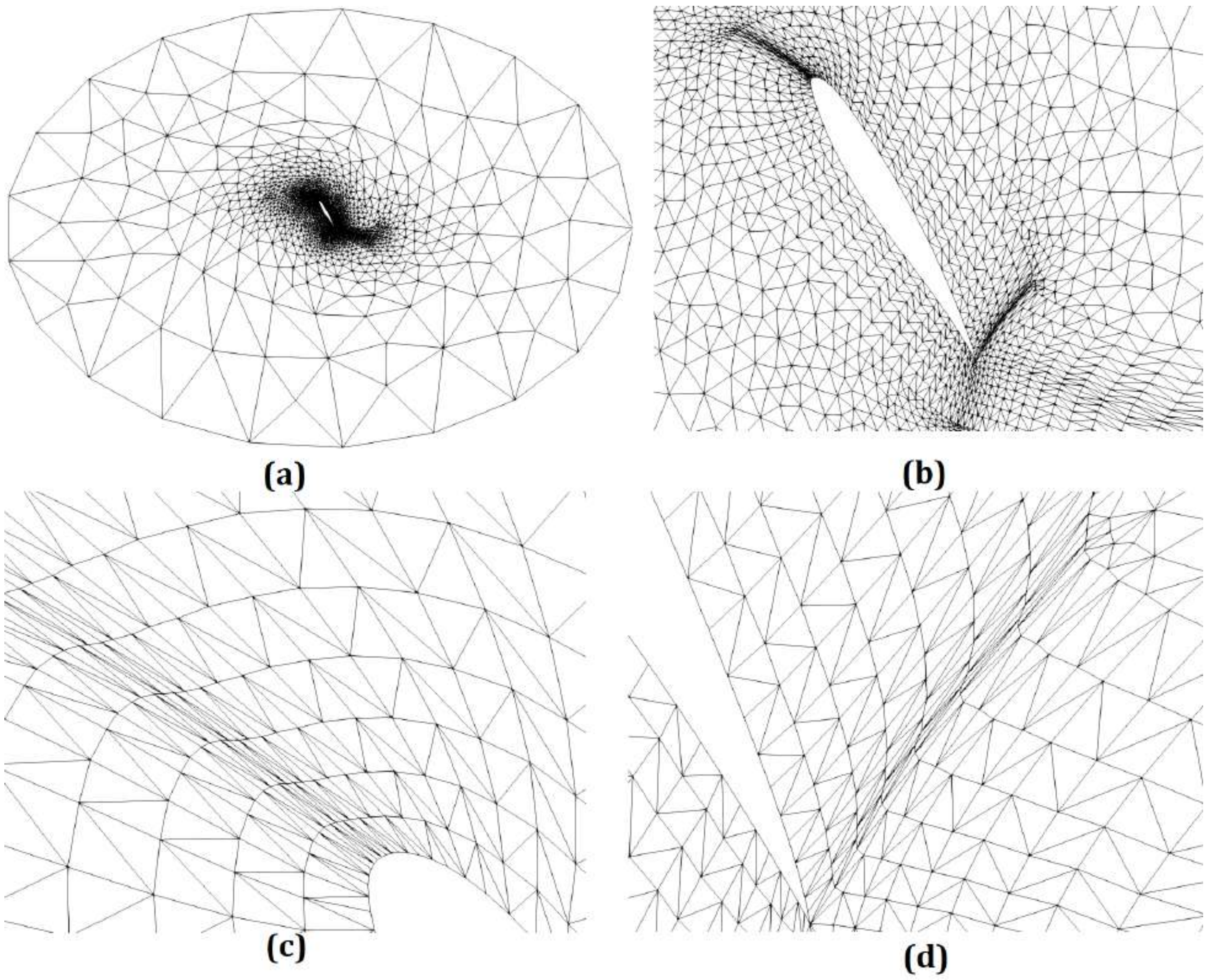


Figure 4.31: Isolated airfoil case: 60° rotation around the trailing edge, linearized coupled algorithm without sub-steps and p=2. (a): Final mesh. (b): Zoom at the airfoil. (c): Zoom at the leading edge. (d): Zoom at the trailing edge.

4.2 3D Applications

4.2.1 3D Cube Torsion

This first case evaluates the algorithm's performance on 3D hybrid meshes. The cube is composed of tetrahedra, pyramids, prisms, and hexahedra, containing 1340 nodes and 3740 elements. The cube undergoes distortion through a rotation around the y-axis. The initial and final meshes are presented below:

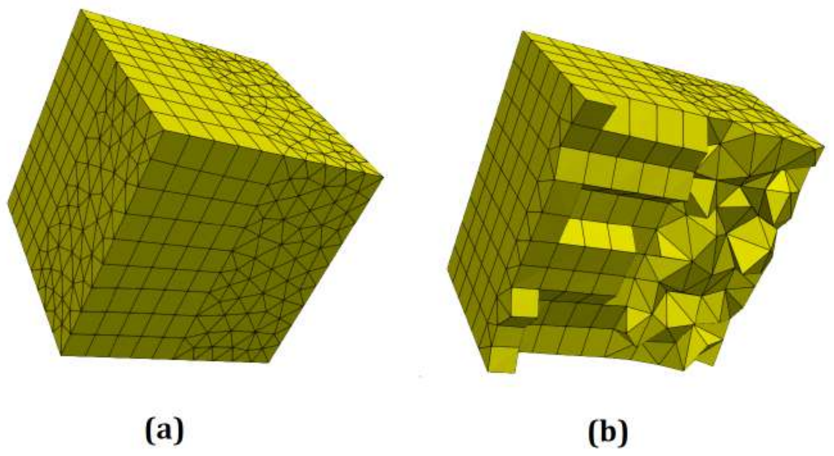


Figure 4.32: 3D cube torsion case. (a): Initial mesh.(b): Interior clip.

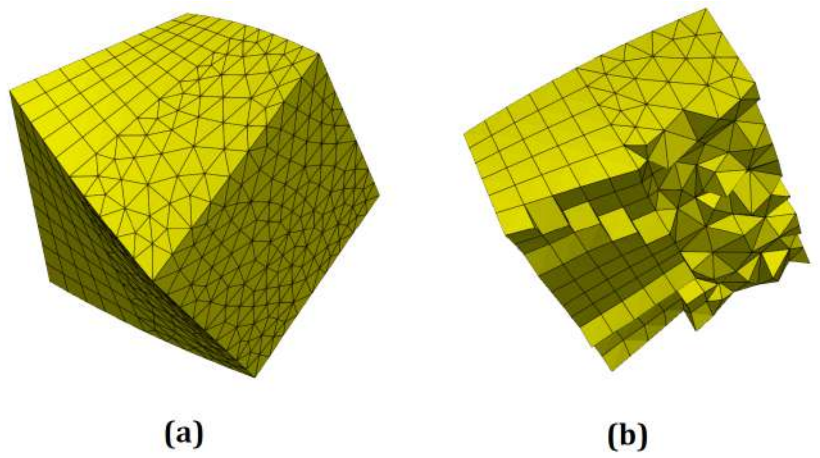


Figure 4.33: 3D cube torsion case. (a): Final mesh.(b): Interior clip.

4.2.2 ONERA M6 Wing Bending

This case represents a more realistic 3D scenario, with an application in the field of aeroelasticity. The initial mesh consists of 72,791 nodes and 341,797 tetrahedral elements, making it significantly larger than the previous cases. The purpose of this test is to evaluate the algorithm's performance in realistic, large-scale 3D applications. The computational domain is shown in fig. 4.34.

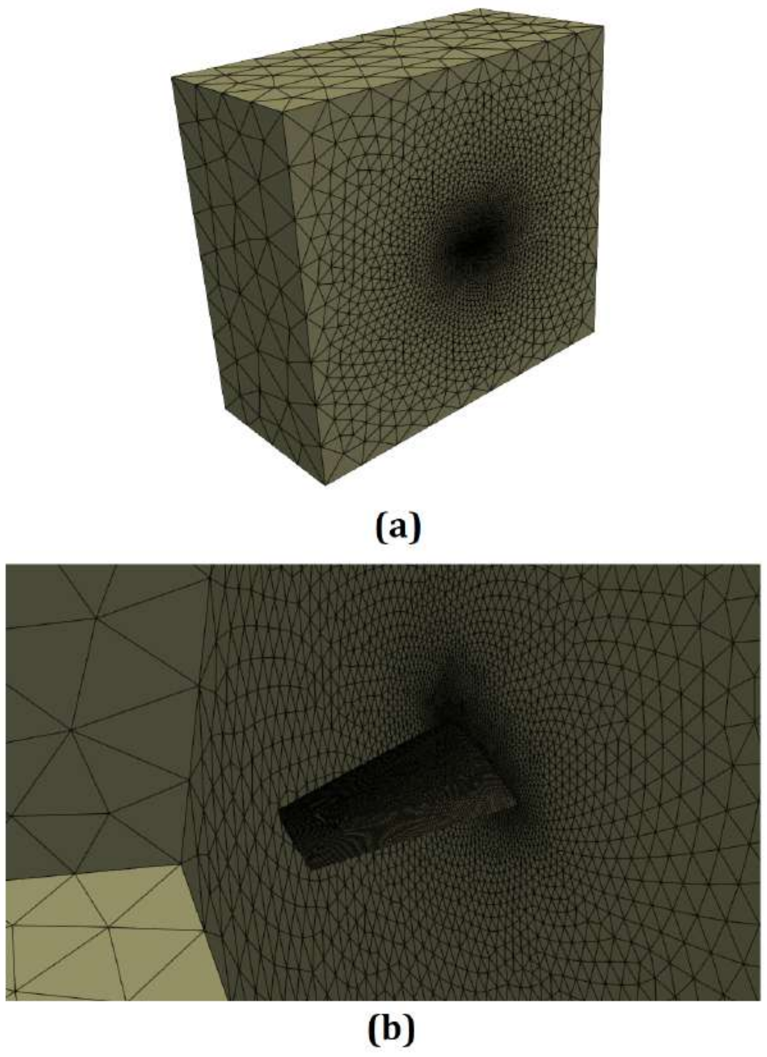


Figure 4.34: ONERA M6 Bending case. (a): Initial mesh. (b): Zoom to wing.

In this case, the ONERA M6 wing undergoes bending, along the spanwise direction (i.e.,the z-axis) which is described by the following equations:

$$x^{\text{new}} = x^{\text{old}} \tag{4.1a}$$

$$y^{\text{new}} = y^{\text{old}} + \alpha (z^{\text{old}})^2 \tag{4.1b}$$

$$z^{\text{new}} = z^{\text{old}} \tag{4.1c}$$

The variable α controls the magnitude of the bending, with a starting value of 0.1 and increasing until inverted cells appear. The Linearized Coupled algorithm is employed, with the parameter p also investigated but not shown here. Only the results for the optimized value of p are presented. The settings of the linearized Coupled algorithm are shown in table 4.3.

Table 4.3: DWRBM Algorithm Settings

Method	**Linearized Coupled**
Pre-conditioner	Diagonal
Stopping criteria	Relative residual$<10^{-4}$
Max Newton (outer) Iterations	-
GMRES base m	various
Max GMRES Iterations	100
Weighting Parameter value	4

Convergence Study

This section evaluates the convergence of the GMRES method. A diagonal preconditioner is used and compared to the unpreconditioned case to assess computational efficiency. The impact of varying the GMRES basis size m is also examined. The convergence study for a=0.1 is illustrated in fig. 4.35. As shown, the diagonal preconditioner significantly improves convergence. It is particularly noteworthy that the cost per iteration for the preconditioned GMRES remains unchanged compared to the unpreconditioned case, provided the same GMRES basis size m is used. In terms of computational time, the preconditioned GMRES with a basis size of m=3 was the fastest, completing in 240 seconds. So for this case the Diagonal Preconditioner with basis m=3 is selected. The non-linear coupled approach was also tested for comparison, requiring approximately 7 minutes to achieve the same quality mesh.

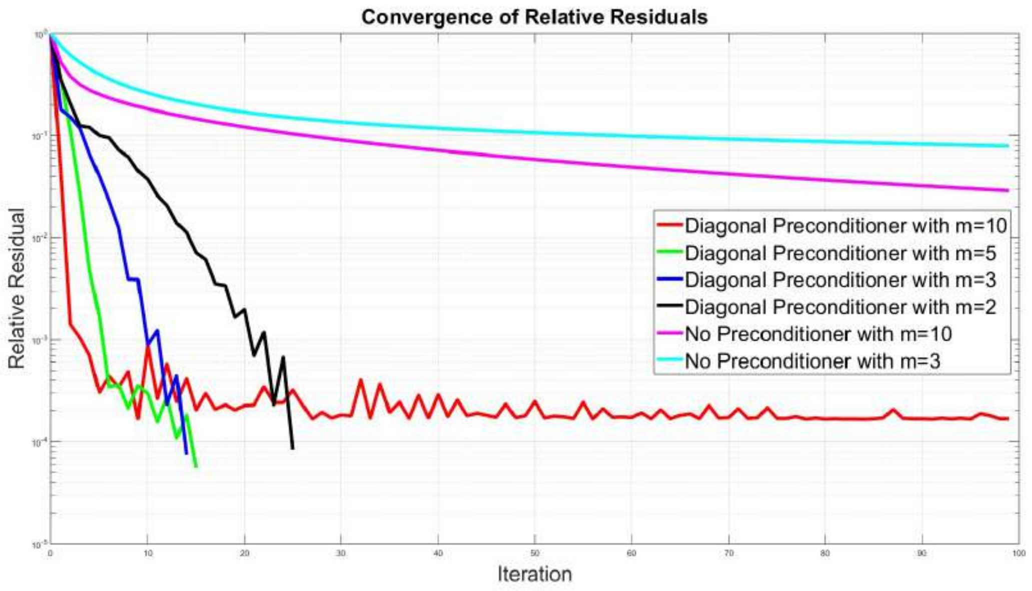


Figure 4.35: Convergence analysis of GMRES, ONERA M6 case.

Bending parameter $\alpha = 0.1$

The final position of the wing is shown in fig. 4.36. For this case, p=4 was selected, and no sub-steps were required. The final mesh is presented in fig. 4.37. No inverted cells appeared and the final mesh maintained the same quality as the initial one, as demonstrated by the histogram in fig. 4.38.

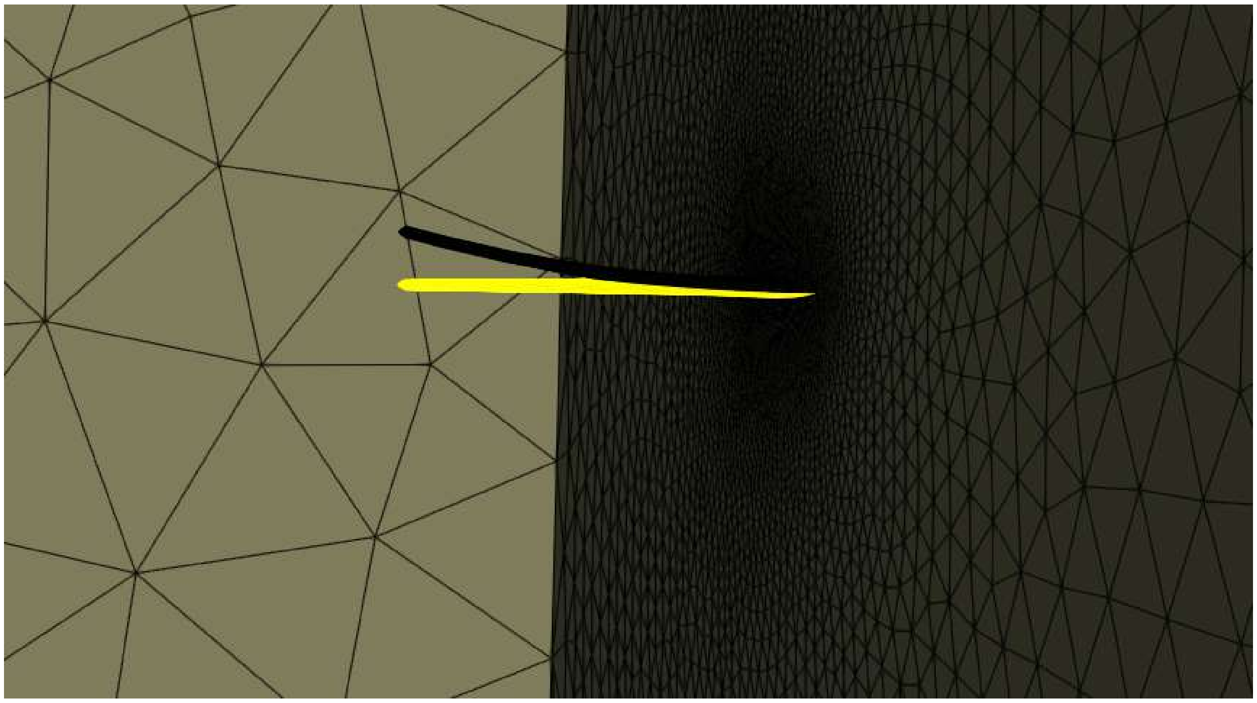


Figure 4.36: ONERA M6 Bending case. a=0.1 Black: Final wing. Yellow: Initial wing.

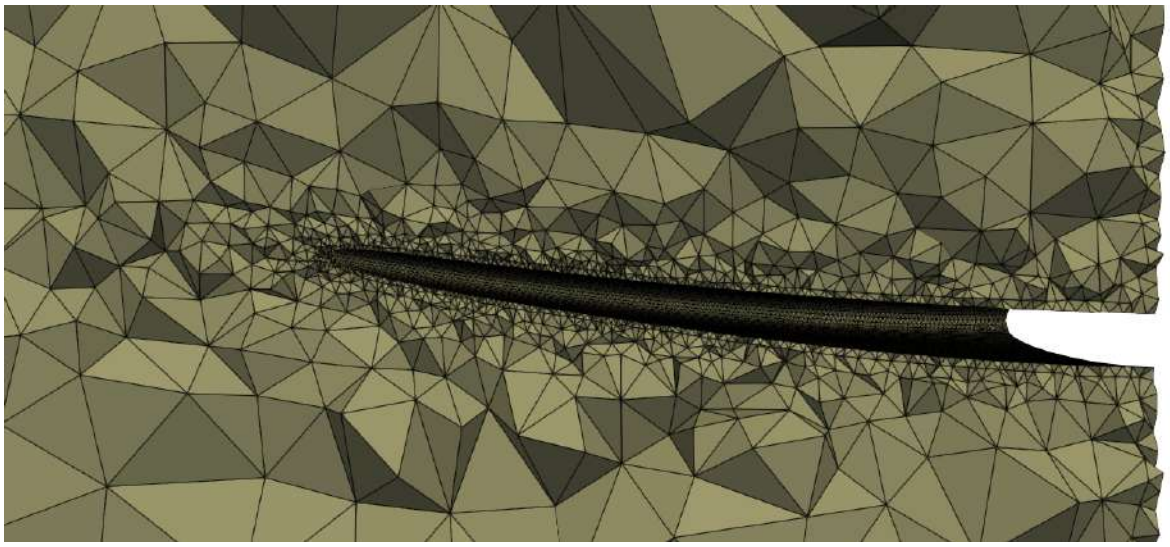


Figure 4.37: ONERA M6 Bending, a=0.1 case. mesh close to the wing.

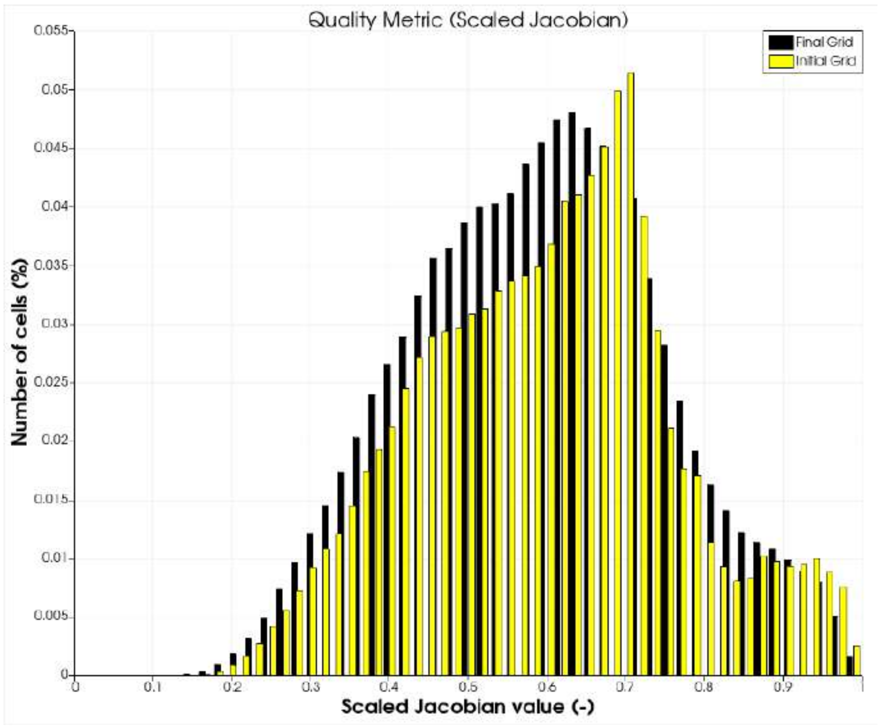


Figure 4.38: ONERA M6 Bending, a=0.1 case. Quality histogram, scaled Jacobian metric. Black: Final wing. Yellow: Initial wing.

Bending parameter $\alpha = 0.2$

The final position of the wing is shown in fig. 4.39. For this case, p=4 was also selected, and no sub-steps were required. The clock time was approximately 4 minutes and the final mesh is presented in fig. 4.40. No inverted cells appeared and the final mesh maintained the same quality as the initial one, as demonstrated by the histogram in fig. 4.41.

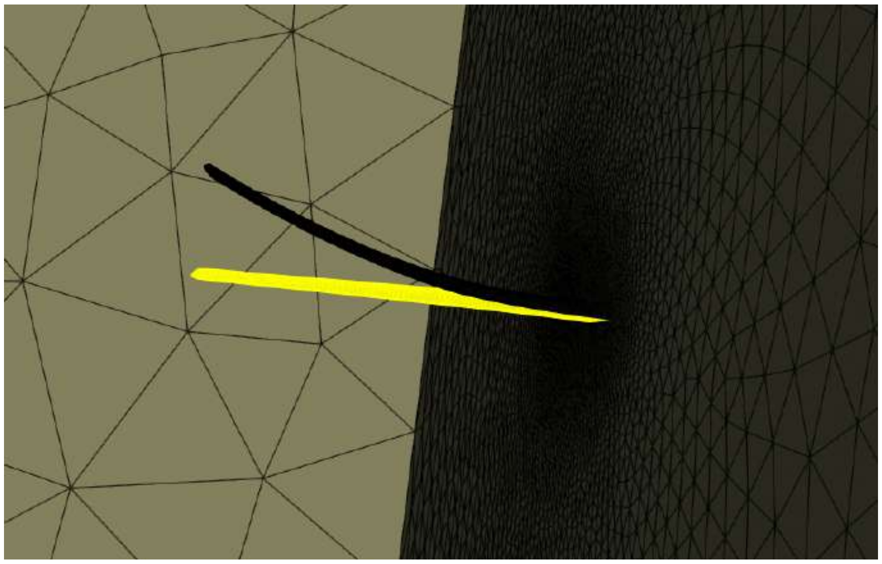


Figure 4.39: ONERA M6 Bending case. a=0.2 Black: Final wing. Yellow: Initial wing.

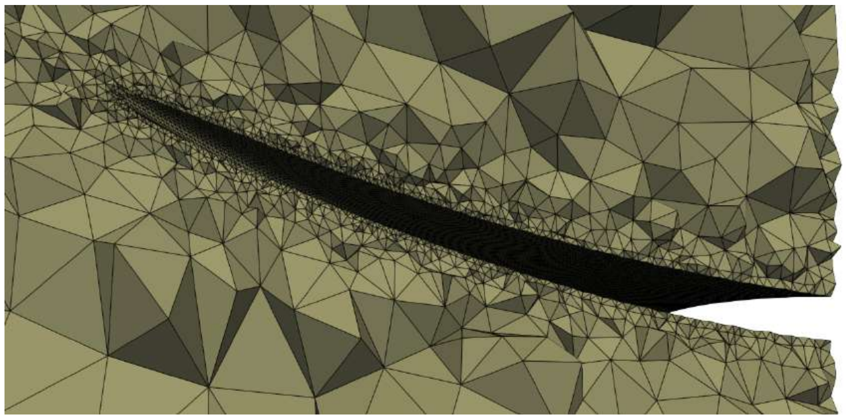


Figure 4.40: ONERA M6 Bending, a=0.2 case. mesh close to the wing.

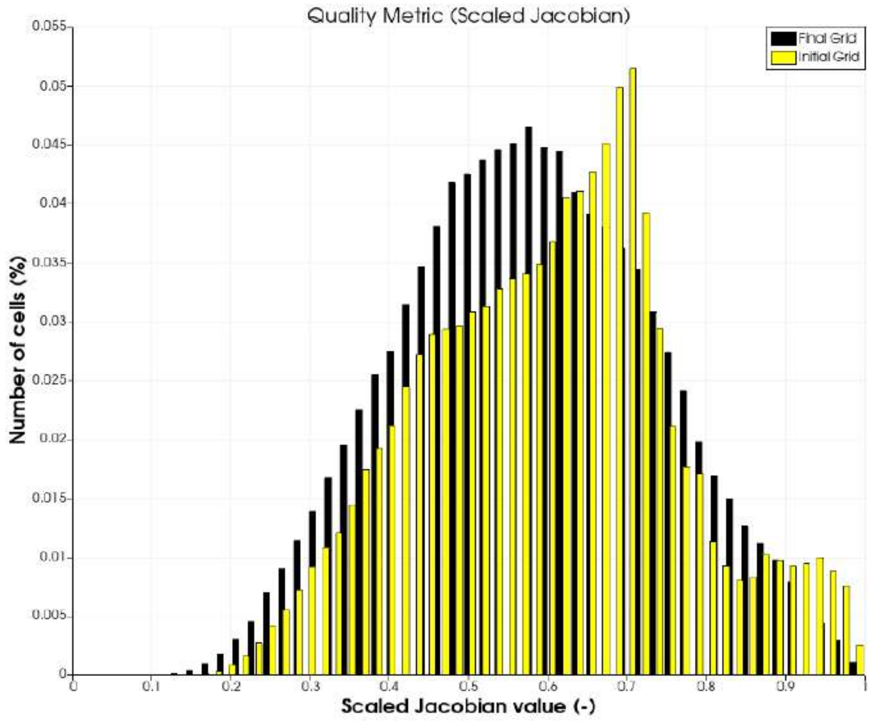


Figure 4.41: ONERA M6 Bending, a=0.2 case. Quality histogram, scaled Jacobian metric. Black: Final wing. Yellow: Initial wing.

Bending parameter $\alpha = 0.3$

For this case, p=4 was selected without the use of sub-steps. However, as shown in the histogram in fig. 4.42, inverted cells appeared, indicated by negative values of the scaled Jacobian in some mesh cells. Identifying the location of these inverted cells, as illustrated in fig. 4.43, provides valuable insight into addressing this issue. The inverted cells were found close to the wing surface, suggesting the need for a larger weighting parameter or the introduction of sub-stepping. As an initial approach, the parametric study of p is recommended. If the problem persists, further investigation will be required combined with the use of sub-steps.

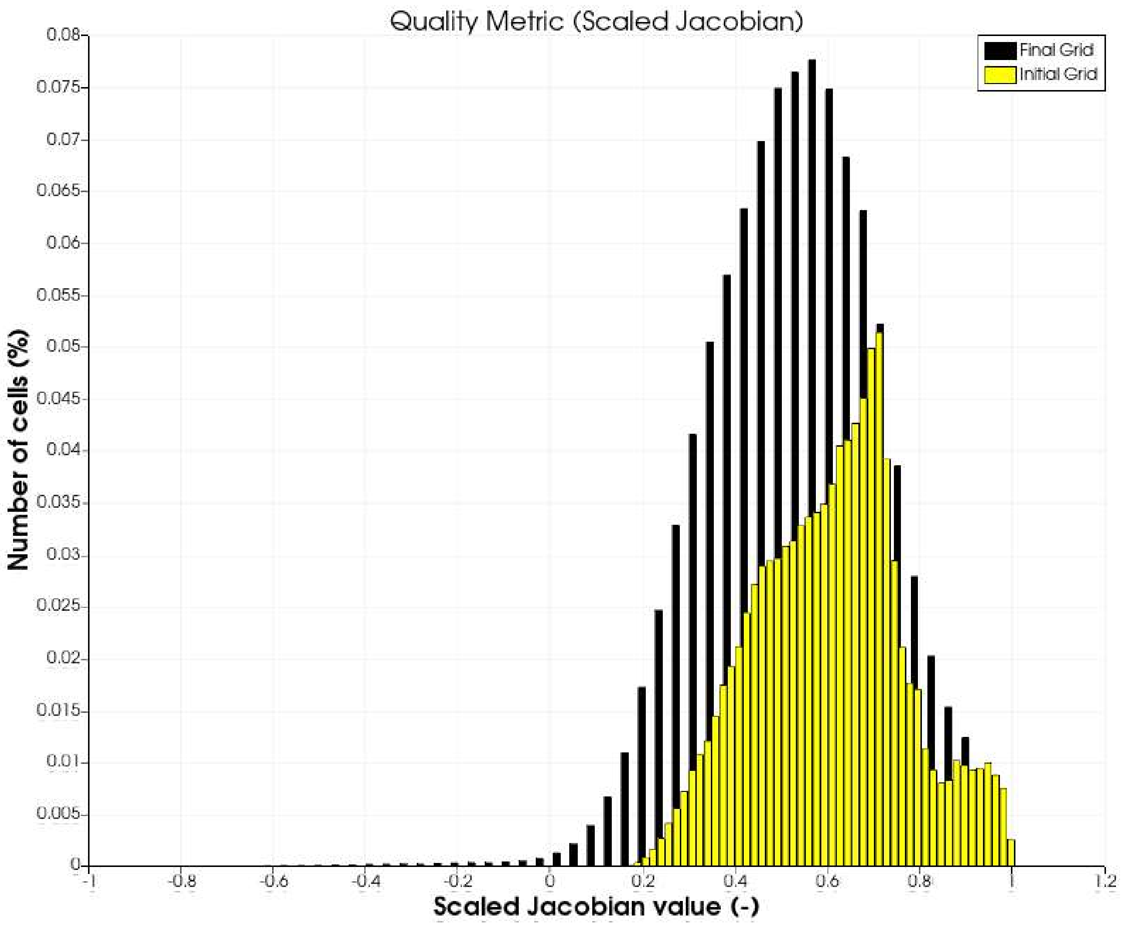


Figure 4.42: ONERA M6 Bending, a=0.3 case, with no sub-steps, Quality histogram, scaled Jacobian metric. Black: Final wing. Yellow: Initial wing.

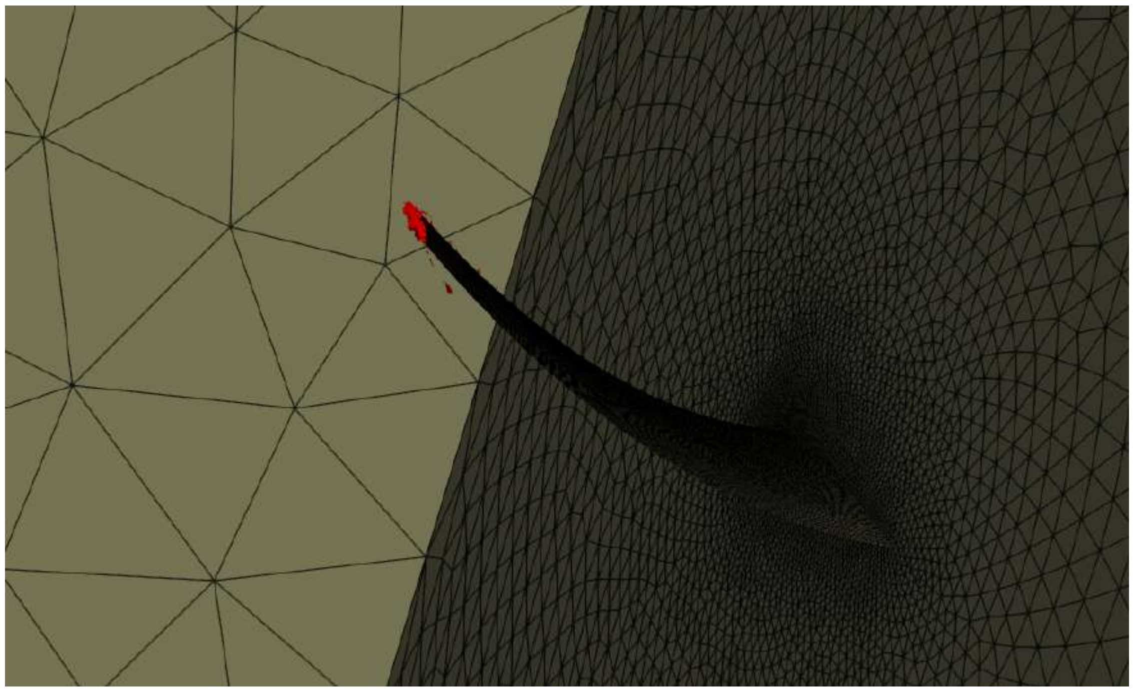


Figure 4.43: ONERA M6 Bending, a=0.3 case. Inverted cells in red.

4.2.3 ONERA M6 Wing Combined Bending and Torsion

This case represents a more complex and challenging displacement scenario, involving both the bending and torsion of the wing. The updated coordinates of the wing are described by the following equations:

$$x^{\text{new}} = \cos\phi(x_0 - x_r) - \sin\phi(y_0 - y_r) + x_r \tag{4.2a}$$

$$y^{\text{new}} = \cos\phi(y_0 - y_r) + \sin\phi(x_0 - x_r) + y_r + \alpha z_0^2 \tag{4.2b}$$

$$z^{\text{new}} = z_0 \tag{4.2c}$$

where,

$$x_r = 5.25 + 0.45 z_0 \tag{4.3}$$

$$y_r = 5.0 \tag{4.4}$$

$$\phi = \alpha z_0^2 \tag{4.5}$$

Combined Bending and Torsion, parameter $\alpha = 0.1$

For this case, p=4 was selected, and the use of sub-steps was not required. The final position of the wing is shown in fig. 4.44. No inverted cells were observed, and the algorithm converged in approximately 5 minutes. The final mesh is presented in fig. 4.45, and its quality is comparable to the initial mesh, as demonstrated in the histogram in fig. 4.46.

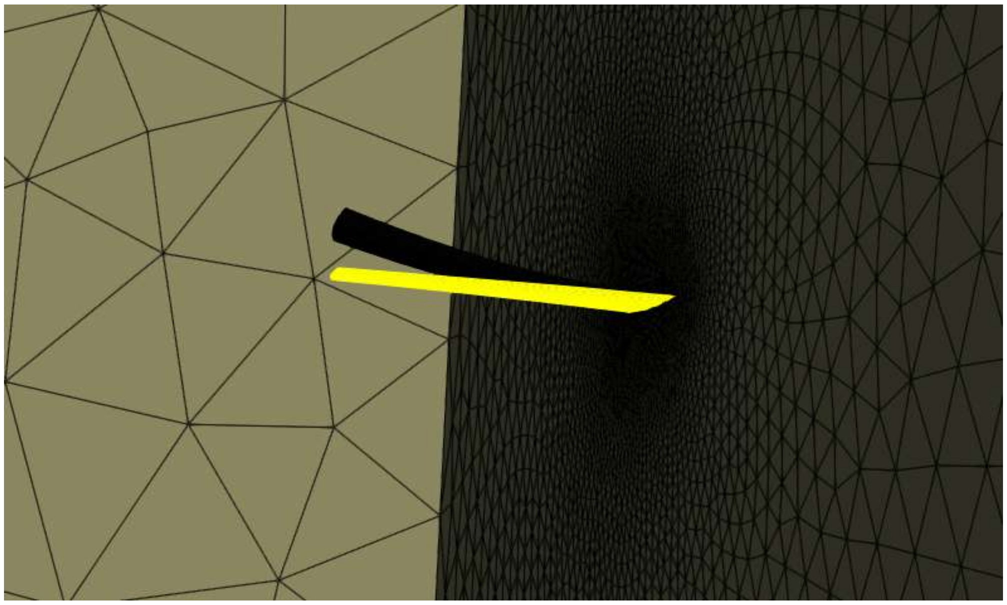


Figure 4.44: ONERA M6 Combined Bending and Torsion case. a=0.1 Black: Final wing. Yellow: Initial wing.

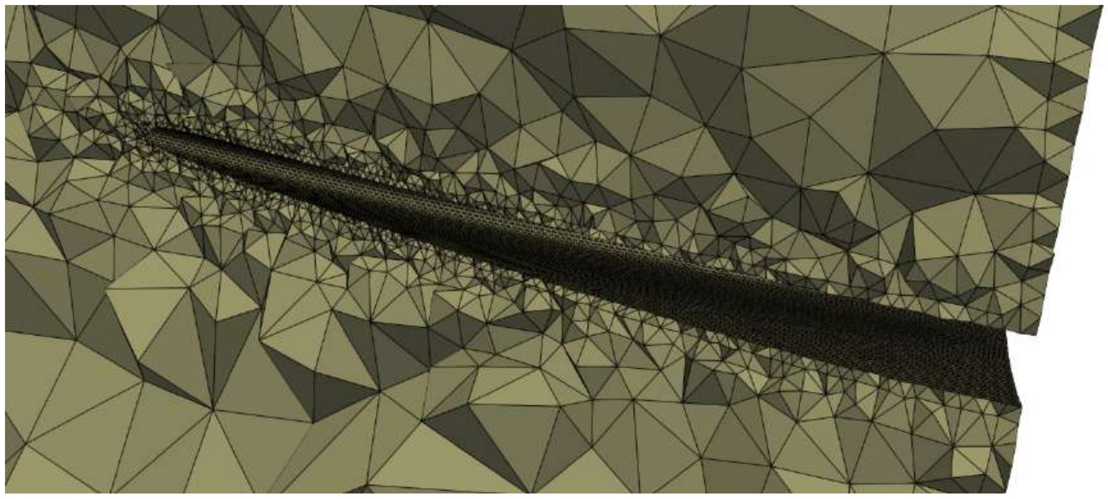


Figure 4.45: ONERA M6 Combined Bending and Torsion case, a=0.1. mesh close to the wing.

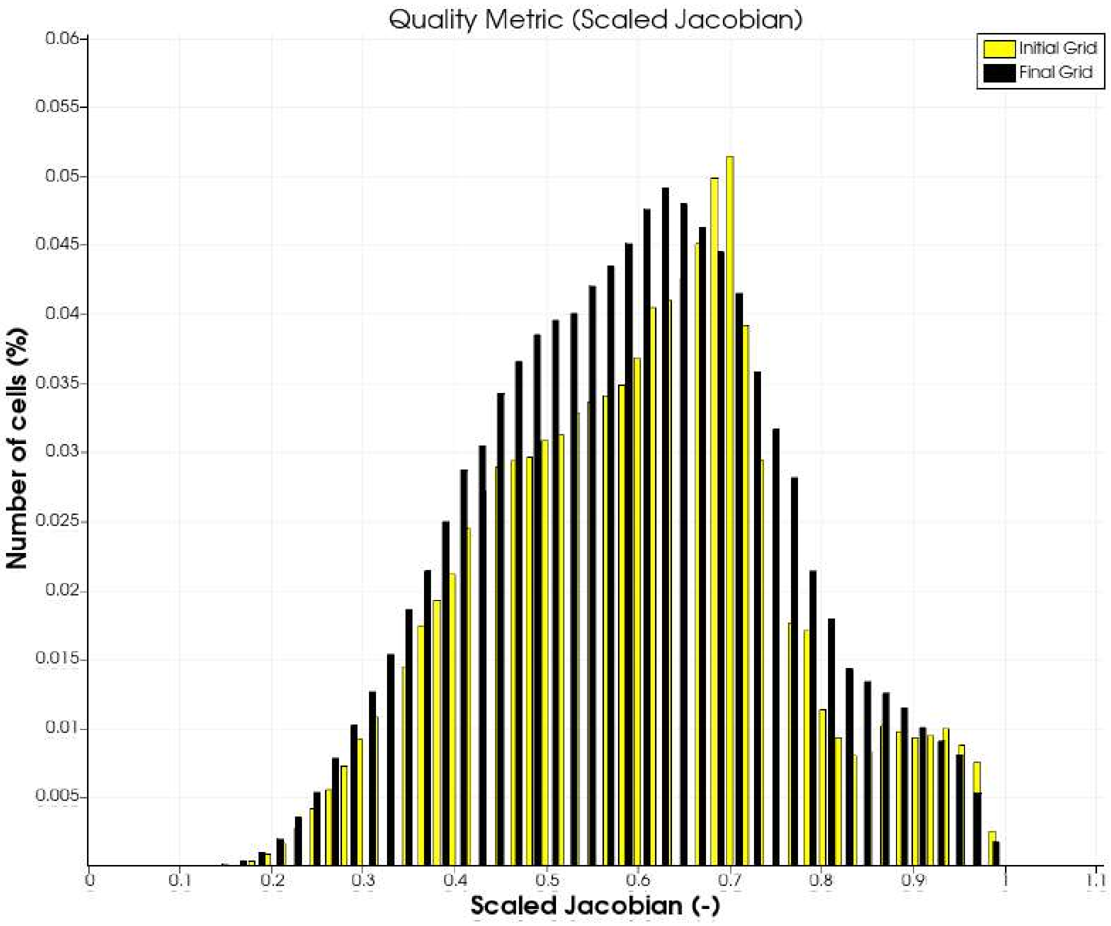


Figure 4.46: ONERA M6 Combined Bending and Torsion case, a=0.1. Quality histogram, scaled Jacobian metric. Black: Final wing. Yellow: Initial wing.

4.2.4 S-Bend pipe

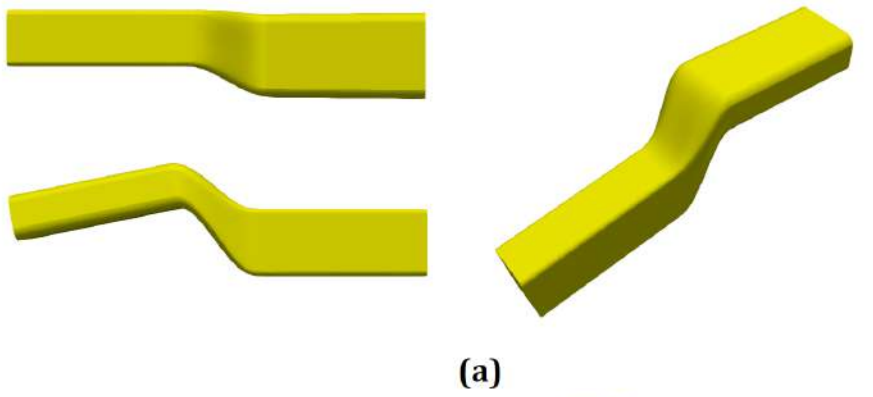
This case represents a realistic shape deformation of a s-bend pipe. The geometry deformation resembles of that in a shape optimization procedure, and the aim of this case is to test the algorithm in a realistic 3D shape optimization case. The mesh is structured and is consisted of 479688 nodes and 465976 hexahedra. The initial geometry and the corresponding mesh is presented in fig. 4.47. The displacement acted on the geometry is shown in fig. 4.48 and is described by this set of equations:

$$x^{\text{new}} = x^{\text{old}} \tag{4.6a}$$

$$y^{\text{new}} = y^{\text{old}} + \alpha(|x^{\text{old}}_{\text{min}}| - |x^{\text{old}}|)(|x^{\text{old}}_{\text{max}}| - |x^{\text{old}}|) \tag{4.6b}$$

$$z^{\text{new}} = z^{\text{old}} + \alpha(|x^{\text{old}}_{\text{min}}| - |x^{\text{old}}|)(|x^{\text{old}}_{\text{max}}| - |x^{\text{old}}|) \tag{4.6c}$$

The linearized coupled algorithm is used, with same settings of the wing case, described in table 4.3. The algorithm managed to adapt the mesh to the deformation without the use of sub-steps for parameter a=0.3. The final mesh is presented in fig. 4.50 and its quality compared to the initial is presented in the histogram of fig. 4.49.



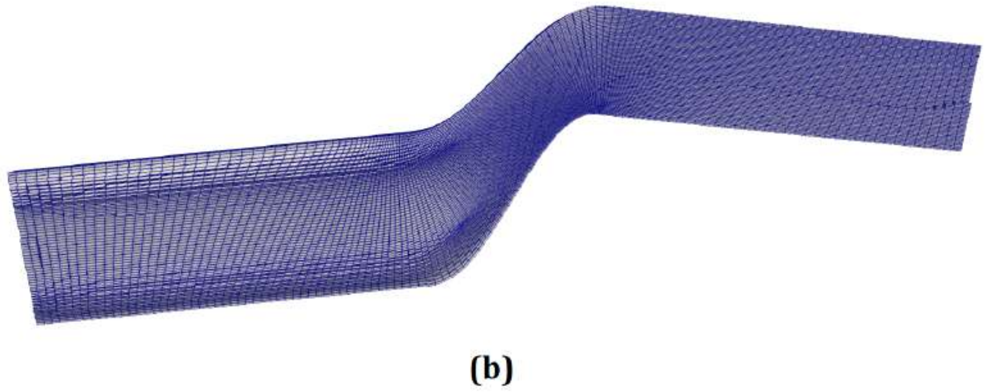


Figure 4.47: S-Bend pipe. (a): Initial geometry. (b): Initial mesh.

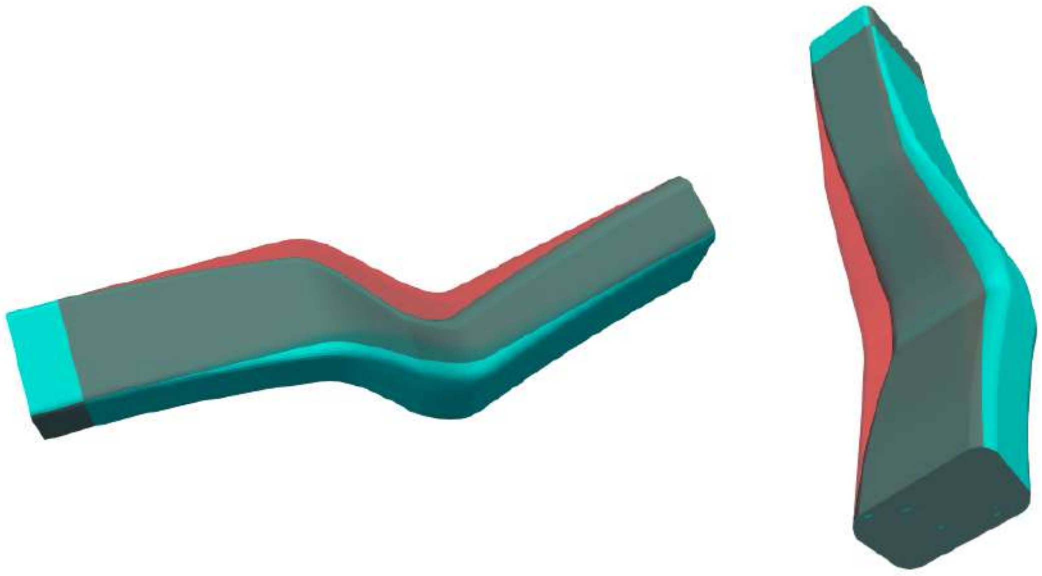


Figure 4.48: S-Bend pipe case. Red: Initial geometry. Cyan: Deformed geometry.

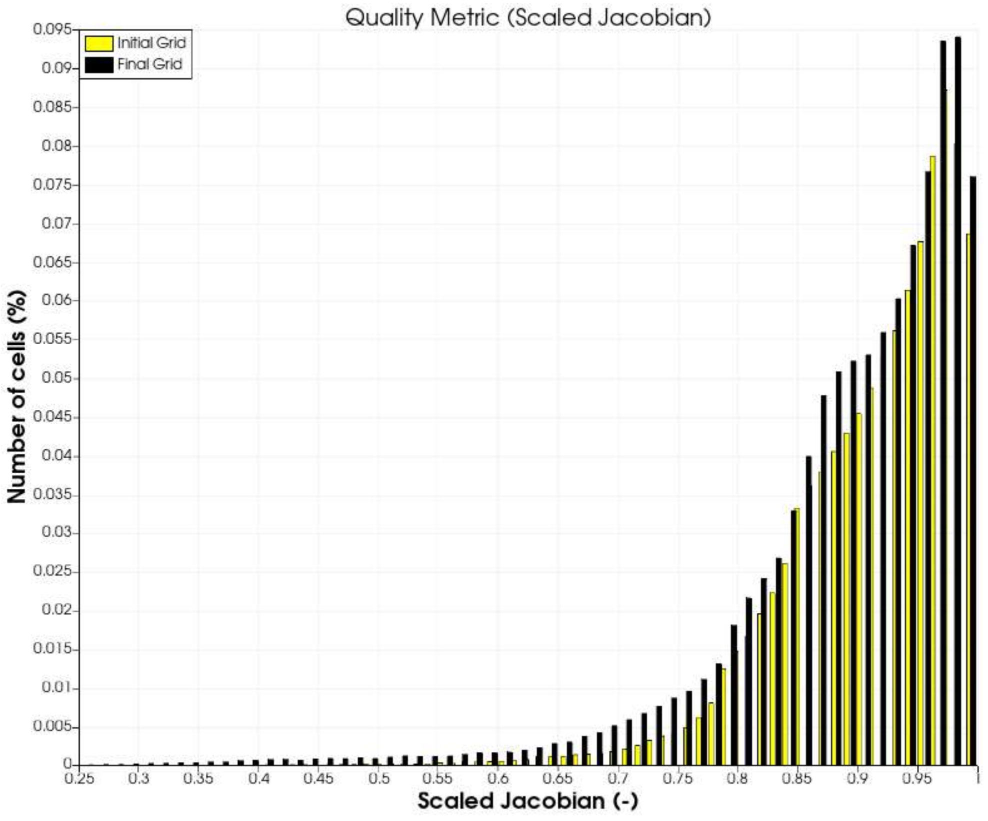


Figure 4.49: S-Bend pipe case, a=0.3: Quality histogram, scaled Jacobian metric. Black: Final mesh. Yellow: Initial mesh.

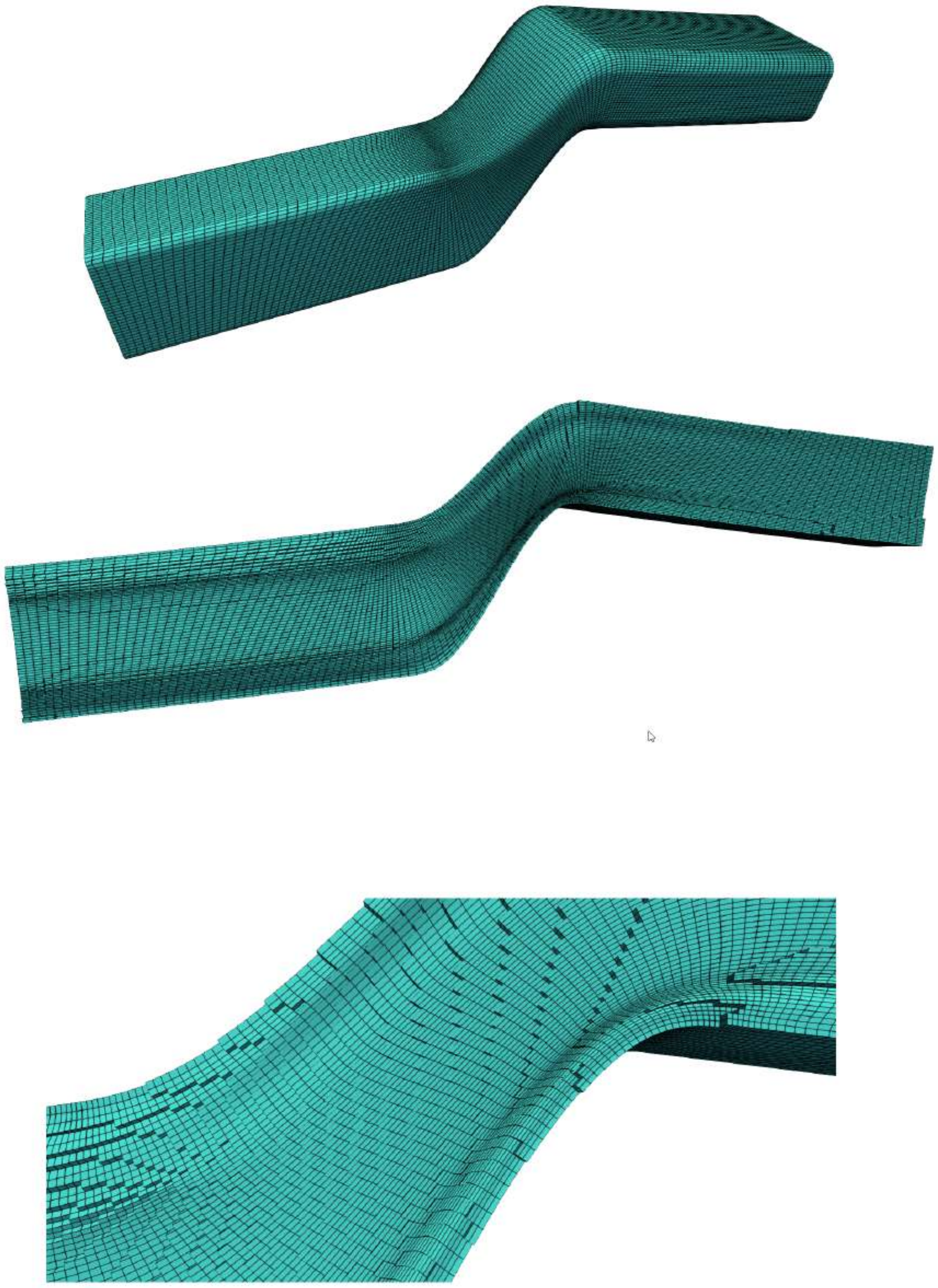


Figure 4.50: S-Bend pipe case. Final mesh

Chapter 5

Concluding Remarks & Future Work

5.1 Concluding Remarks

This thesis presents the implementation and evaluation of the Distance-Based Weighted Rigid Body Motion (DWRBM) method for mesh deformation across structured, unstructured, and hybrid meshes in both 2D and 3D domains. The proposed method was designed to address the challenges associated with maintaining mesh quality during large geometric deformations, ensuring robustness and computational efficiency.

The algorithm leverages a weighting mechanism based on node distance to boundaries, allowing enhanced mesh quality near critical regions, such as boundary layers, while mitigating the risk of inverted cells. This mechanism is case sensitive and a proper value of the exponential parameter p has to be selected. A good value that works is that of p=2, but it can vary from case to case.

Using an optimization framework, the method adjusts the internal mesh nodes in a way that closely resembles rigid body motion, minimizing deformation-induced distortions. The non-linear system arising from the minimization problem is solved using the Newton-GMRES method. A linear coupled algorithm is also developed, based on small angle approximation. A GMRES solver based on a modified Compressed Sparse Row (mCSR) format was developed, enabling easier preconditioning (easy acces to diagonal and upper/lower triangular matrices).

Extensive testing is conducted on a variety of cases, ranging from simple 2D geometries to complex 3D aerodynamic models like the ONERA M6 wing. The results demonstrated the method's capability to handle significant deformations while maintaining mesh quality. Notably:

-In 2D applications, the method efficiently managed both rotation and translation, maintaining high-quality meshes with minimal computational cost.

-In 3D applications, the algorithm performed robustly in scenarios involving bending and torsion of a 3D wing, as well as shape deformation of s-bend pipe.

The inclusion of distance-based weighting proved essential for avoiding inverted cells, particularly in areas with complex geometry or high aspect ratio elements. Extensive testing demonstrates that the non-linear weighted coupled algorithm is the most robust approach, capable of handling large displacements at a reasonable computational cost without requiring sub-stepping. However, the linearized version of the coupled algorithm proved more efficient in some cases due to its lower computational cost. By incorporating weights, the linearized coupled algorithm can also manage large displacements without sub-stepping, as demonstrated in the ONERA M6 case. Therefore, for computationally expensive scenarios, the linearized coupled algorithm is recommended, particularly for cases involving smaller displacements. In general, the linearized coupled algorithm displaced the mesh in approximately 50% of the computational time compared to the non-linear one. However, for extreme deformations the linearized algorithm failed, and the use of the non-linear one was necessary. Particularly in cases involving extreme deformations (90° airfoil rotation for example) or meshes with highly variable elements size the method required additional sub-steps or parameter tuning to achieve convergence. This highlights the potential for further investigation in the weighting strategy.

In conclusion, the Distance-Based Weighted RBM method has proven to be a versatile and effective tool for mesh deformation in computational fluid dynamics (CFD), aerodynamic shape optimization and aeroelastic analysis. Its ability to maintain mesh quality while handling significant deformations makes it a valuable addition to the field.

5.2 Future Work

As already discussed, the method developed in this thesis has demonstrated high effectiveness. However, there are several opportunities for further improvement.

Firstly, a more robust and adaptive weighting function needs to be explored. The current polynomial-based weighting requires user experience and is sensitive to specific cases, which can limit its applicability. Developing a more automated and generalized weighting approach would enhance the method's reliability and usability.

Furthermore, further optimization of the data structure is essential, as it accounts for a significant portion of the computational time in the algorithm. This optimization, combined with parallelization of the algorithm, could significantly reduce the overall computational cost, making the method more efficient for large-scale problems.

Finally, the method should be tested on larger 3D cases involving real-world aerodynamic optimization problems or in cases with even greater displacements. Such tests will provide a more comprehensive evaluation of the algorithm's performance in large-scale computations and highlight its contribution to reducing the computational time required in practical applications.

Bibliography

[1] Ahrens, J., Geveci, B., and Law, C. (2005). ParaView: An end-user tool for large data visualization. In *Visualization Handbook*. Elesvier. ISBN 978-0123875822.

[2] Arnoldi, W. E. (1951). The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quarterly of Applied Mathematics*, 9:17–29.

[3] Blom, F. J. (2000). Considerations on the spring analogy. *International journal for numerical methods in fluids*, 32(6):647–668.

[4] Burg, C. (2006). Analytic study of 2d and 3d grid motion using modified laplacian. *International journal for numerical methods in fluids*, 52(2):163–197.

[5] Christianos, E. (2024). An implicit grid displacement method based on a rigid body motion model. Master's thesis, National Technical University of Athens (NTUA).

[6] De Boer, A., Van der Schoot, M. S., and Bijl, H. (2007). Mesh deformation based on radial basis function interpolation. *Computers & structures*, 85(11-14):784–795.

[7] Farhat, C., Degand, C., Koobus, B., and Lesoinne, M. (1998). Torsional springs for two-dimensional dynamic unstructured fluid meshes. *Computer methods in applied mechanics and engineering*, 163(1-4):231–245.

[8] Giannakoglou, K. C. (2006). *Optimization Methods in Aerodynamics*. National Technical University of Athens.

[9] Helenbrook, B. T. (2003). Mesh deformation using the biharmonic operator. *International journal for numerical methods in engineering*, 56(7):1007–1021.

[10] Jameson, A. (1988). Aerodynamic design via control theory. *Journal of scientific computing*, 3:233–260.

[11] Liatsikouras, A. (2023). *Rigid Body Motion Techniques in Grid Displacement and CAD–Free Shape Parameterization, in Aerodynamic Optimization*. PhD thesis, National Technical University of Athens (NTUA).

[12] Saad, Y. (2003). *Iterative methods for sparse linear systems*. SIAM.

[13] Saad, Y. and Schultz, M. H. (1986). Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3):856–869.

[14] Selim, M. and Koomullil, R. (2016). Mesh deformation approaches–a survey.

[15] Stein, K., Tezduyar, T., and Benney, R. (2003). Mesh moving techniques for fluid-structure interactions with large displacements. *J. Appl. Mech.*, 70(1):58–63.

[16] Witteveen, J. and Bijl, H. (2009). Explicit mesh deformation using inverse distance weighting interpolation. In *19th AIAA Computational Fluid Dynamics*, page 3996.

Εκτενής Περίληψη στα Ελληνικά

Εισαγωγή

Αυτή η διατριβή παρουσιάζει την ανάπτυξη, τον προγραμματισμό και την αξιολόγηση της μεθόδου Κίνησης Απαραμόρφωτου Σώματος, ενισχυμένης με βάρη ανάλογα της απόστασης (Distance Weighted Rigid Body Motion, DWRBM), μιας τεχνικής μετατόπισης πλέγματος που έχει σχεδιαστεί για τη διατήρηση της ποιότητας του πλέγματος κατά τη διάρκεια μεγάλων γεωμετρικών παραμορφώσεων. Με την ενσωμάτωση των αρχών κίνησης απαραμόρφωτου σώματος και ενός μηχανισμού βαρύτητας με βάση την απόσταση, η μέθοδος διατηρεί αποτελεσματικά την ποιότητα του πλέγματος κοντά στα όρια και ελαχιστοποιεί τον κίνδυνο αναστροφής των κελιών.

Βασιζόμενη στις μη γραμμικές και γραμμικοποιημένες μεθόδους που παρουσιάστηκαν σε προηγούμενη διπλωματική εργασία [5], αυτή η μελέτη βελτιώνει την απόδοσή τους μέσω της εφαρμογής της τεχνικής βαρύτητας με βάση την απόσταση.Ένας επιλύτης GMRES αναπτύχθηκε σε C++ χρησιμοποιώντας ένα τροποποιημένο σχήμα αποθήκευσης (modified Compressed Sparse Row, mCSR), για την επίλυση των συστημάτων που προκύπτουν από το πρόβλημα βελτιστοποίησης. Επίσης αναπτύχθηκαν και δοκιμάστηκαν δύο τεχνικές προσταθεροποίησης, η διαγώνια και η Gauss-Seidel. Τέλος ο αλγόριθμος δοκιμάστηκε σε διάφορες δισδιάστατες (2Δ) και τρισδιάστατες (3Δ) εφαρμογές.

Μαθηματική Διατύπωση

Η μαθηματική έκφραση για τη μετατόπιση ενός κόμβου στον διδιάστατο χώρο είναι:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \tag{1}$$

Σύμφωνα με την υπόθεση πως το πλέγμα μετατοπίζεται προσεγγίζοντας όσο το δυνατόν περισσότερο την κίνηση ενός απαραμόρφωτου σώματος, η Συνάρτηση Κόστους F που ζητείται να ελαχιστοποιηθεί για κάθε κόμβο $i, i \in I$, όπου I αναπαριστά το σύνολο των εσωτερικών κόμβων στο πλέγμα, είναι:

$$F_i = \frac{1}{2} \sum_{j \in N(i)} [(x_j^{ideal} - x_j^{new})^2 + (y_j^{ideal} - y_j^{new})^2] \tag{2}$$

όπου ο δείκτης i δηλώνει τον κεντρικό κόμβο, ο δείκτης j τους γείτονες του κεντρικού κόμβου, ενώ το $N(i)$ αποτελεί το σύνολο όλων των γειτονικών κόμβων του κεντρικού. Τα $(x_j^{ideal}, y_j^{ideal})$ υποδηλώνουν τις x, y συντεταγμένες του γείτονα j αντίστοιχα, μετά τη μετατόπισή του, υποθέτοντας πως πράγματι μετατοπίζεται σύμφωνα με τις ιδιότητες ενός στερεού σώματος. Τα (x_j^{new}, y_j^{new}) υποδηλώνουν τις πραγματικές x, y συντεταγμένες του γείτονα j αντίστοιχα, μετά τη μετατόπισή του. Για να εξασφαλιστεί το βέλτιστο σύνολο $(\Delta x, \Delta y, \theta)$ για κάθε εσωτερικό κόμβο i, είναι απαραίτητο να υπολογιστούν οι παράγωγοι της συνάρτησης κόστους ως προς τις άγνωστες ποσότητες $\Delta x_i, \Delta y_i$, και θ_i, και να τεθούν ίσες με μηδέν.

Αποσυζευγμένη Προσέγγιση

Στην Αποσυζευγμένη Προσέγγιση, για να υπολογιστεί η μετατόπιση του κεντρικού κόμβου, θεωρούνται γνωστές οι μετατοπίσεις όλων των γειτόνων του. Αυτό ισχύει μόνο στην περίπτωση των γειτονικών κόμβων που είναι οριακοί, ενώ για τους υπόλοιπους είναι απλά υπόθεση. Έτσι λύνεται ενα 3×3 σύστημα (ή 6×6 στην περίπτωση του 3Δ) υπολογίζοντας τις τελικές θέσεις του κεντρικού κόμβου. Η διαδικασία αυτή επαλαμβάνεται για κάθε εσωτερικό κόμβο, και λόγω της υπόθεσης που αναφέρεται παραπάνω, ξαναξεκινάει η διαδικασία από την αρχή μέχρι να συγκλίνουν.

Συζευγμένη Προσέγγιση

Στην Συζευγμένη Προσέγγιση, ορίζεται η συνολική συνάρτηση κόστους, η οποία λαμβάνει υπόψιν της την αλληλεξάρτηση μεταξύ όλων των κόμβων του πλέγματος. Η συνολική συνάρτηση κόστους ορίζεται ως:

$$F_{total} = \sum_{i \in I} w_i F_i \tag{3}$$

όπου w_i είναι το βάρος κάθε κόμβου (που θα οριστεί στην συνέχεια) και F_i είναι η συνάρτηση κόστους ορισμένη στην εξ. (2).

Οπότε απο την συζευγμένη μέθοδο, προκύπτει ένα μη γραμμικό σύστημα, η επίλυση του οποίου παρέχει το βέλτιστο σύνολο τιμών των ποσοτήτων για την ταυτόγχρονη μετατόπιση όλων των κόμβων.

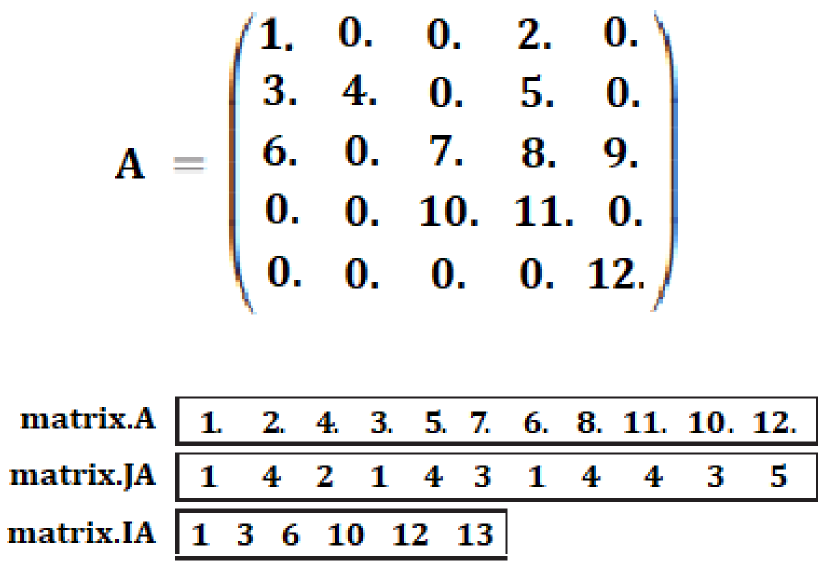
Γραμμικοποίηση μέσω προσέγγισης

Ακόμη, παρουσιάζεται μια προσέγγιση με στόχο την αντικατάσταση του μη-γραμμικού συστήματος με το γραμμικό αντίστοιχο. Αυτή η αντικατάσταση εξυπηρετεί τον σκοπό της μείωσης του υπολογιστικού κόστους της επίλυσης και βασίζεται στην προσέγγιση μικρών γωνιών. Θεωρώντας ότι η γωνία περιστροφής κάθε συστοιχίας κόμβων είναι μικρή έχουμε:

$$\theta \to 0 \Rightarrow \begin{cases} sin\theta \to \theta \\ cos\theta \to 1 \end{cases} \tag{4}$$

Υπολογιστικό πλαίσιο

Για να επιλυθούν τα παραπάνω συστήματα αναπτύχθηκε ένας επιλύτης GMRES με δύο προδιαθέτες, διαγώνιο και Guass-Seidel. Οι πίνακες αποθηκεύονται χρησιμοποιώντας ένα τροποποιημένο σχήμα αποθήκευσης (modified Compressed Sparse Row, mCSR), με σκοπό την αποδοτική αποθήκευση αλλά και διαχείριση των αραιών αυτών πινάκων. Το παραπάνω τροποποιημένο σχήμα αποθήκευσης παρουσιάζεται στην παρακάτω εικόνα:



$$\mathbf{A} = \begin{pmatrix} 1. & 0. & 0. & 2. & 0. \\ 3. & 4. & 0. & 5. & 0. \\ 6. & 0. & 7. & 8. & 9. \\ 0. & 0. & 10. & 11. & 0. \\ 0. & 0. & 0. & 0. & 12. \end{pmatrix}$$

matrix.A	1.	2.	4.	3.	5.	7.	6.	8.	11.	10.	12.
matrix.JA	1	4	2	1	4	3	1	4	4	3	5
matrix.IA	1	3	6	10	12	13					

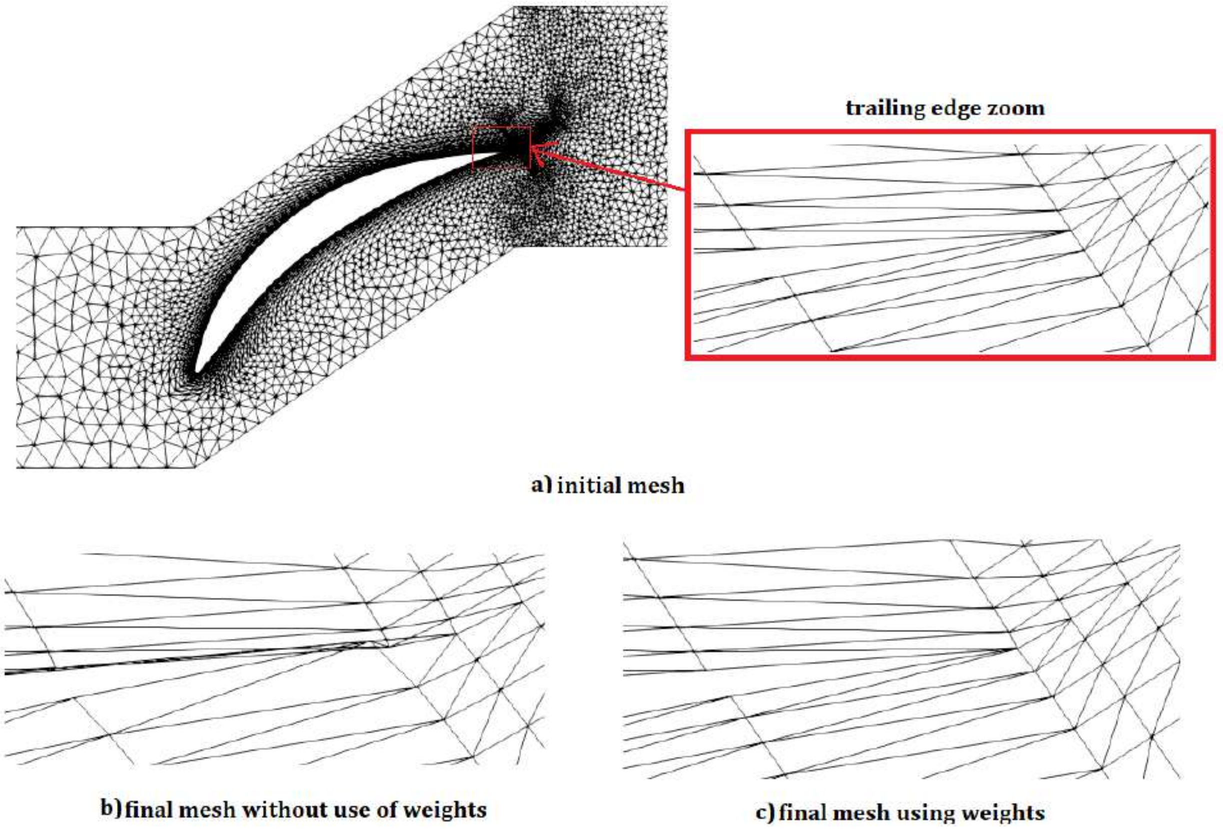
Σχήμα 1: Τροποποιημένο σχήμα αποθήκευσης

Βάρη ανάλογα της απόστασης

Όπως αναφέρθηκε προηγουμένως, στην συζευγμένη προσέγγιση γίνεται η χρήση βαρών με σκοπό την βελτίωση της ποιότητας του πλέγματος σε περιοχές κοντά στα όρια. Οι περιοχές αυτές είναι πιο ευάλωτες σε ανεστραμένα κελιά, αφού συνήθως αποτελούνται από μακρόστενα κελιά που δεν μπορούν να παραλάβουν εύκολα την παραμόρφωση. Η έκφραση των βαρών είναι εμπνευσμένη από την λογική της σταθμισμένης αντίστροφης απόστασης (inverse-distance weighting (IDW)). Η απόσταση αυτή ορίζεται σε κάθε κόμβο, ως η απόσταση από το κοντινότερο όριο. Η έκφραση που χρησιμοποιήθηκε είναι:

$$w = \frac{1}{D^p} \tag{5}$$

όπου D είναι η απόσταση και p είναι η εκθετική παράμετρος που μας επιτρέπει τον πλήρη έλεγχο των βαρών. Η σημασία των βαρών γίνεται αντιληπτή στην παρακάτω περίπτωση παραμόρφωσης, όπου οι ακμές εκφυγής και προσφυγής της αεροτομής ανυψώνονται με σκοπό την μείωσης της καμπυλότητας. Χωρίς την χρήση βαρών, εμφανίζονται ανεστραμένα κελιά, ενώ με την χρήση τους, το τελικό πλέγμα είναι αντίστοιχης ποιότητας με το αρχικό.

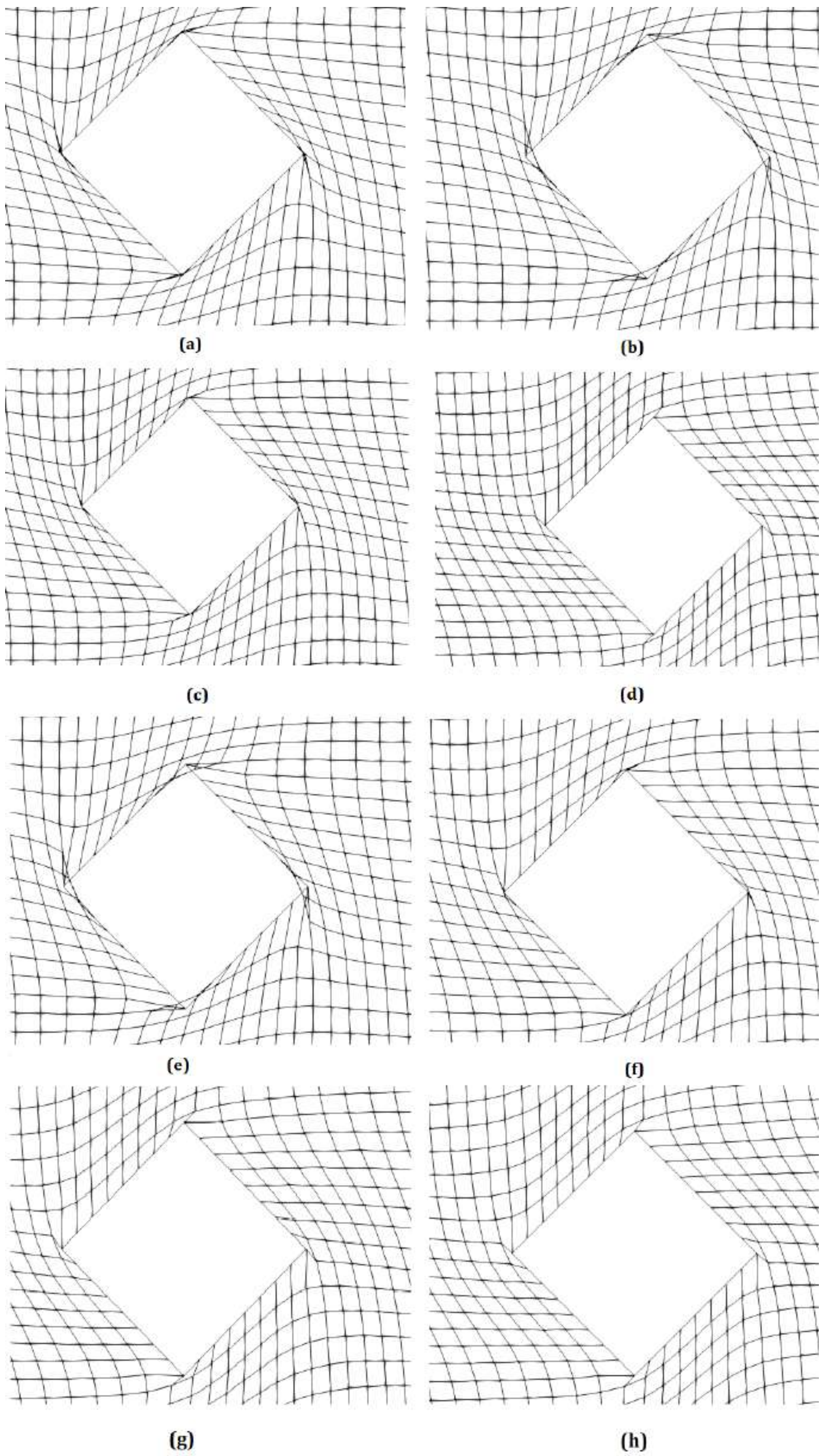

trailing edge zoom
a) initial mesh
b) final mesh without use of weights
c) final mesh using weights

Σχήμα 2: Πάνω: a) Αρχικό πλέγμα, με εστίαση στην ακμή εκφυγής. Κάτω: Τελικό πλέγμα εστιασμένο στην ακμή εκφυγής b) χωρίς την χρήση βαρών, c) με χρήση βαρών

Εφαρμογές Παραμόρφωσης Πλεγμάτων

Ομόκεντρα Τετράγωνα

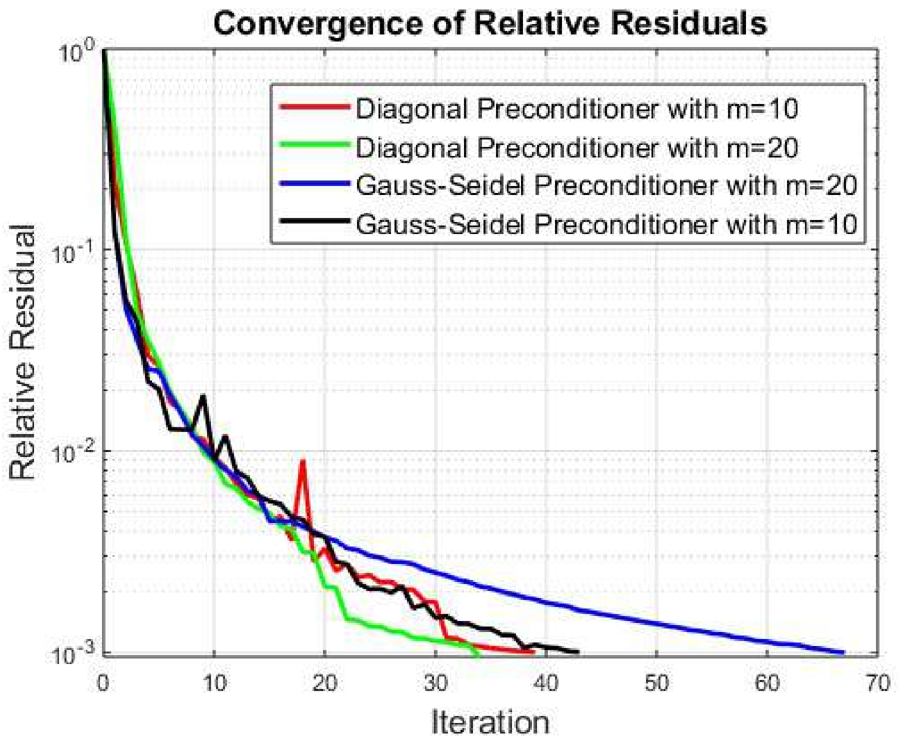
Η πρώτη εφαρμογή που θα παρουσιαστεί είναι η αυτή των 2 ομόκεντρων τετραγώνων. Το έξω τετράγωνο παραμένει σταθερό ενώ το εσωτερικό περιστρέφεται. Δοκιμάστηκαν όλες οι μέθοδοι (συζευγμένη, αποσυζευγμένη), εφαρμόζοντας γραμμικοποίηση και μη, όπως επίσης και βάρη. Υπολογιστικά καταλληλότερη κρίθηκε η γραμμικοποιημένη συζευγμένη μέθοδος με χρήση βαρών αφού πέτυχε ίδια αποτελέσματα με την μη γραμμική αντίστοιχη μέθοδο, στον μισό χρόνο. Αξίζει αν σημειωθεί ότι δεν χρειάστηκαν καθόλου υποβήματα. Τα αποτελέσματα για γωνία στροφής 45° φαίνονται στην παρακάτω εικόνα.



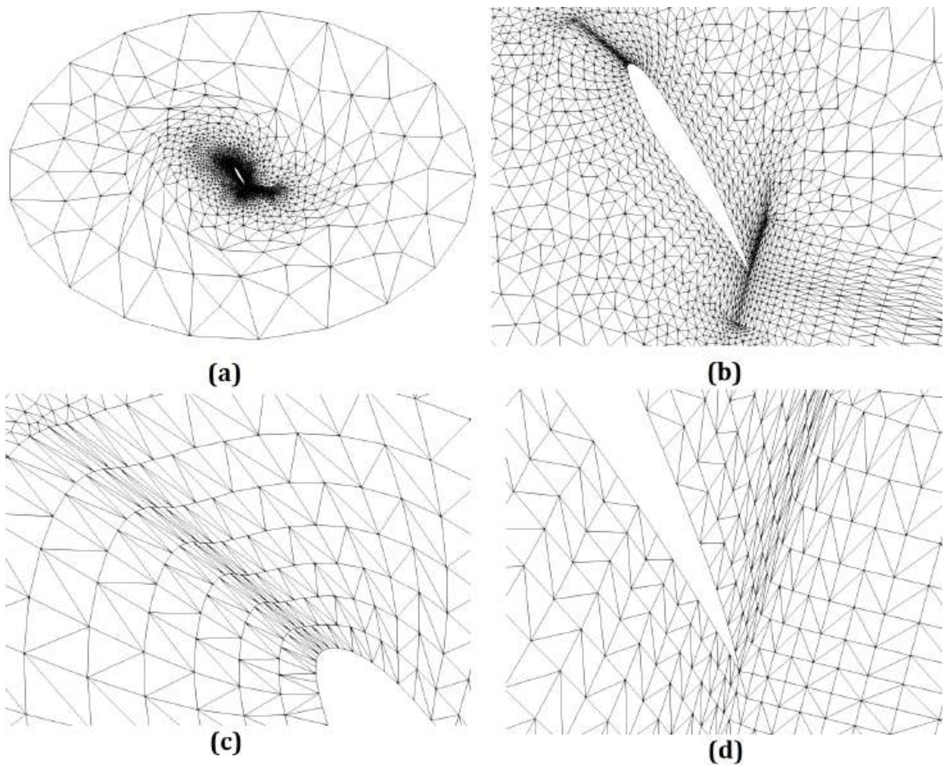
Σχήμα 3: Ομόκεντρα τετράγωνα με 45° περιστροφή συγκεντρωτικά αποτελέσματα. (a): Αποσυζευγμένη μέθοδος. (b): Γραμμικοποιημένη συζευγμένη μέθοδος. (c): Γραμμικοποιημένη συζευγμένη μέθοδος με χρήση βαρών, $p = 1$. (d): Γραμμικοποιημένη συζευγμένη μέθοδος με χρήση βαρών, $p = 4$. (e): Συζευγμένη μέθοδος. (f): Συζευγμένη μέθοδος με χρήση βαρών, $p = 1$. (g): Συζευγμένη μέθοδος με χρήση βαρών, $p = 2$. (h): Συζευγμένη μέθοδος με χρήση βαρών, $p = 4$.

2Δ αεροτομή

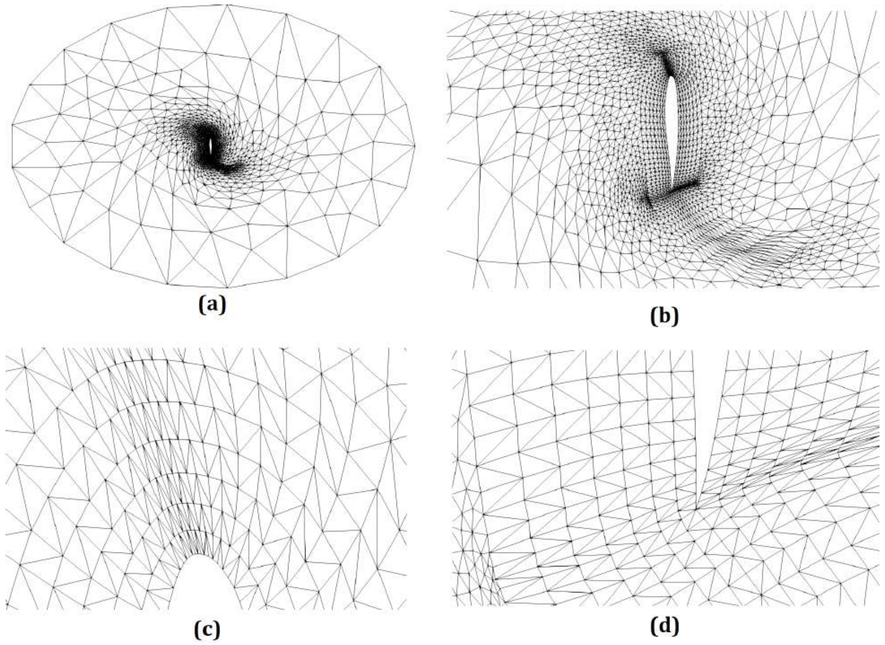
Για τη συγκεκριμένη εφαρμογή, μελετήθηκε η περιστροφή μιας μεμονωμένης αεροτομής. Χρησιμοποιήθηκε η γραμμικοποιημένη συζευγμένη μέθοδος, χωρίς την χρήση υποβημάτων. Η μέθοδος κατάφερε να προσαρμόσει το πλέγμα μέχρι και σε 60° στροφή, παίρνοντας ως σημείο περιστροφής 3 διαφορετικά σημεία, την ακμή προσβολής, το αεροδυναμικό κέντρο αλλά και την ακμή εκφυγής. Επίσης δοκιμάστηκαν και συγκρίθηκαν οι δύο προσταθεροποιητές, για διαφορετικές βάσης του GMRES όπως φαίνεται στο παρακάτω διάγραμμα σύγκλισης. Ταχύτερος φάνηκε να είναι ο διαγώνιος προσταθεροποιητής με βάση $m = 10$. Αξίζει να σημειωθεί ότι η συζευγμένη μέθοδος κατάφερε την περιστροφή της αεροτομής μέχρι και 90°, με την χρήση υποβημάτων. Τέλος παρουσιάζονται τα τελικά πλέγματα για κάθε περίπτωση.



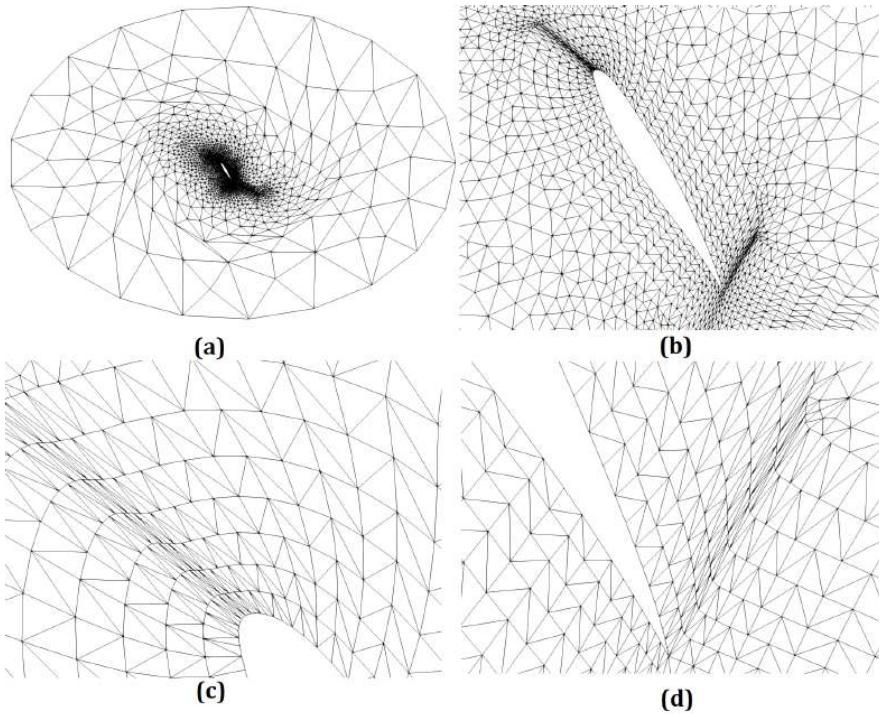
Σχήμα 4: Μελέτη σύγκλισης του $GMRES$, γραμμικοποιημένη συζευγμένη μέθοδος.



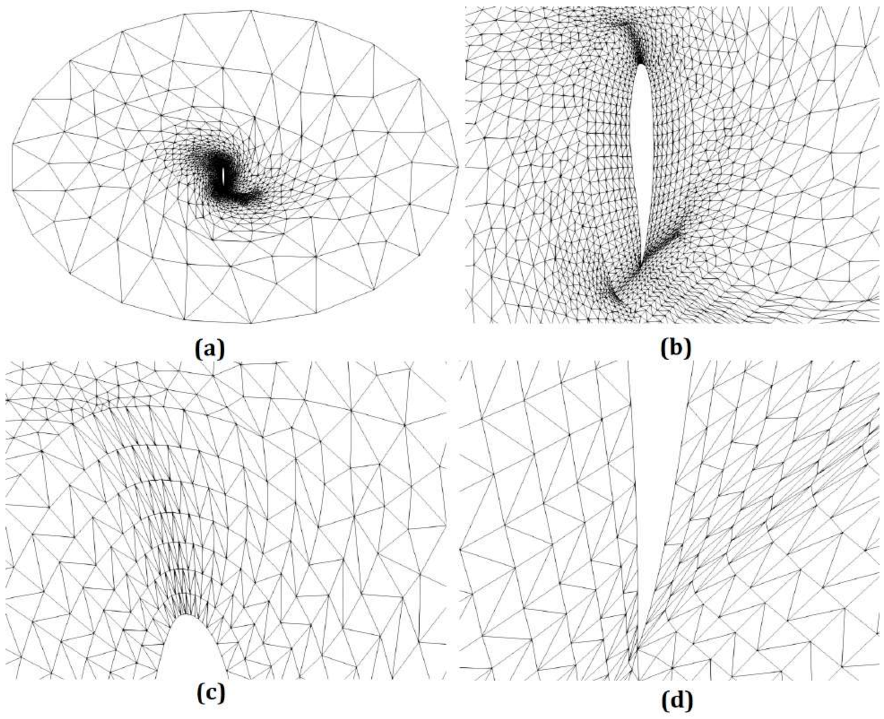
Σχήμα 5: 2Δ αεροτομή: 60° περιστροφή γύρω από το αεροδυναμικό κέντρο, γραμμικοποιημένη συζευγμένη μέθοδος με $p = 4$. (a) : Τελικό πλέγμα. (b) : Εστίαση στην αεροτομή. (c) : Εστίαση στην ακμή προσβολής. (d) : Εστίαση στην ακμή εκφυγής.



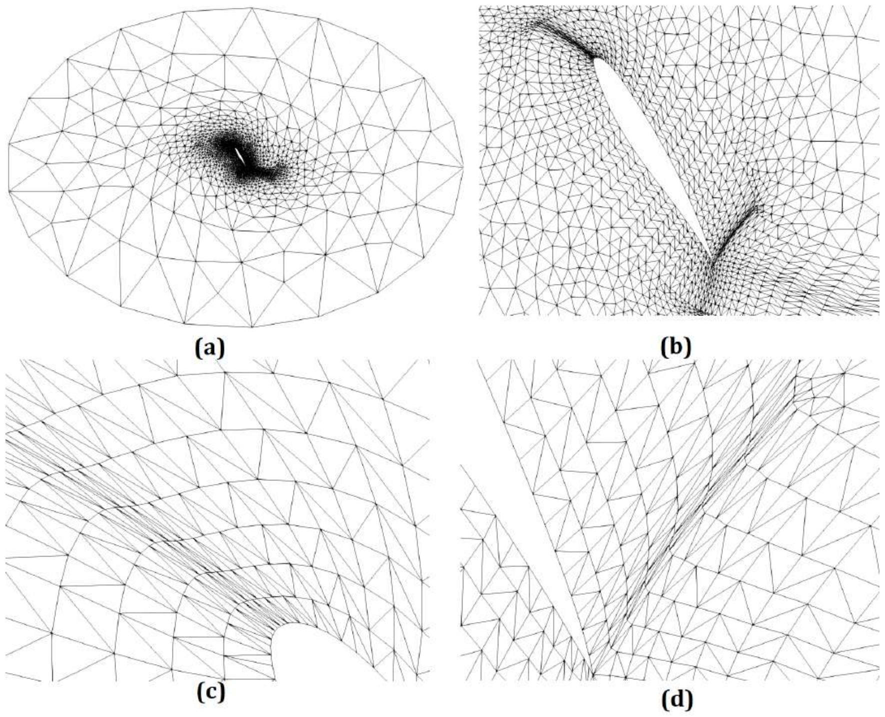
Σχήμα 6: 2Δ αεροτομή: 90° περιστροφή γύρω από το αεροδυναμικό κέντρο, μη γραμμική συζευγμένη μέθοδος με 12 υποβήματα και $p = 4$. (a) : Τελικό πλέγμα. (b) : Εστίαση στην αεροτομή. (c) : Εστίαση στην ακμή προσβολής. (d) : Εστίαση στην ακμή εκφυγής.



Σχήμα 7: 2Δ αεροτομή: 60° περιστροφή γύρω από την ακμή προσβολής, γραμμικοποιημένη συζευγμένη μέθοδος με $p = 4$. (a) : Τελικό πλέγμα. (b) : Εστίαση στην αεροτομή. (c) : Εστίαση στην ακμή προσβολής. (d) : Εστίαση στην ακμή εκφυγής.



Σχήμα 8: 2Δ αεροτομή: 90° περιστροφή γύρω από την ακμή προσβολής, μη γραμμική συζευγμένη μέθοδος με 4 υποβήματα και $p = 2$. (a) : Τελικό πλέγμα. (b) : Εστίαση στην αεροτομή. (c) : Εστίαση στην ακμή προσβολής. (d) : Εστίαση στην ακμή εκφυγής.



Σχήμα 9: 2Δ αεροτομή: 60° περιστροφή γύρω από την ακμή εκφυγής, γραμμικοποιημένη συζευγμένη μέθοδος με $p = 2$. (a) : Τελικό πλέγμα. (b) : Εστίαση στην αεροτομή. (c) : Εστίαση στην ακμή προσβολής. (d) : Εστίαση στην ακμή εκφυγής.

3Δ πτέρυγα

Στη συγκεκριμένη εφαρμογή μελετήθηκε η κάμψη, αλλά και η συνδιασμένη κάμψη και συστροφή της πτέρυγας ONERA M6. Η κάμψη περιγράφεται από τις παρακάτω εξισώσεις:

$$x^{\text{new}} = x^{\text{old}} \tag{6a}$$
$$y^{\text{new}} = y^{\text{old}} + \alpha(z^{\text{old}})^2 \tag{6b}$$
$$z^{\text{new}} = z^{\text{old}} \tag{6c}$$

ενώ η συνδυασμένη κάμψη και συστροφή από τις εξής:

$$x^{\text{new}} = \cos\phi(x_0 - x_r) - \sin\phi(y_0 - y_r) + x_r \tag{7a}$$
$$y^{\text{new}} = \cos\phi(y_0 - y_r) + \sin\phi(x_0 - x_r) + y_r + \alpha z_0^2 \tag{7b}$$
$$z^{\text{new}} = z_0 \tag{7c}$$

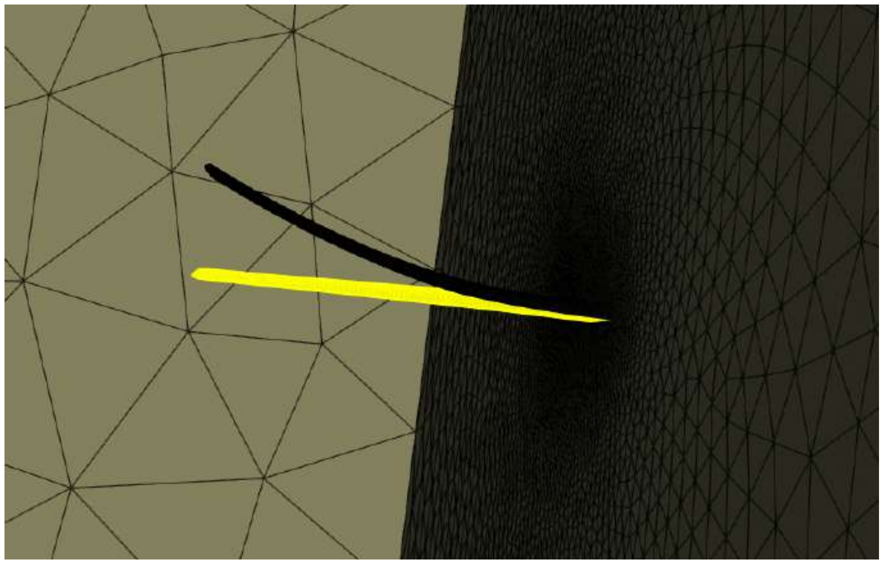
όπου,

$$x_r = 5.25 + 0.45 z_0 \tag{8}$$

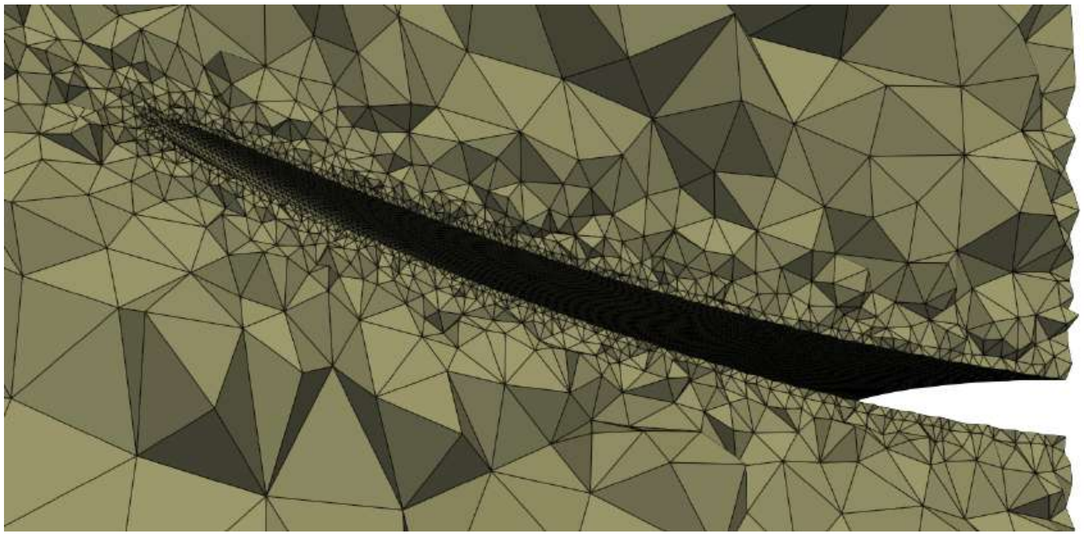
$$y_r = 5.0 \tag{9}$$

$$\phi = \alpha z_0^2 \tag{10}$$

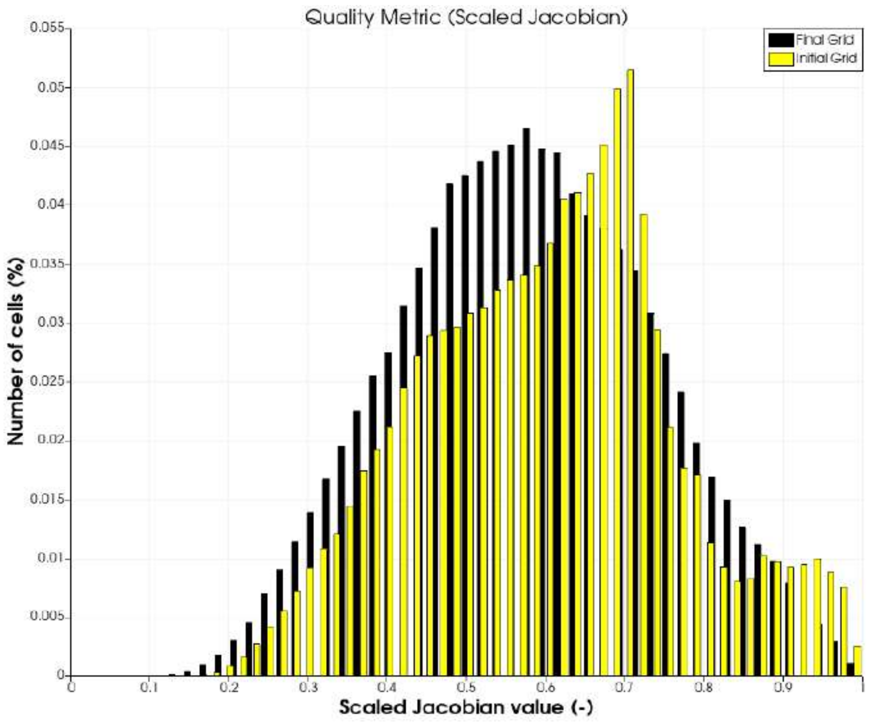
Χρησιμοποιήθηκε η γραμμικοποιημένη συζευγμένη μέθοδος, χωρίς χρήση υποβημάτων, και κατάφερε να προσαρμώσει το πλέγμα στην παραμόρφωση της γεωμετρίας, διατηρόντας την ποιότητα του. Τα αποτελέσματα φαίνονται παρακάτω.



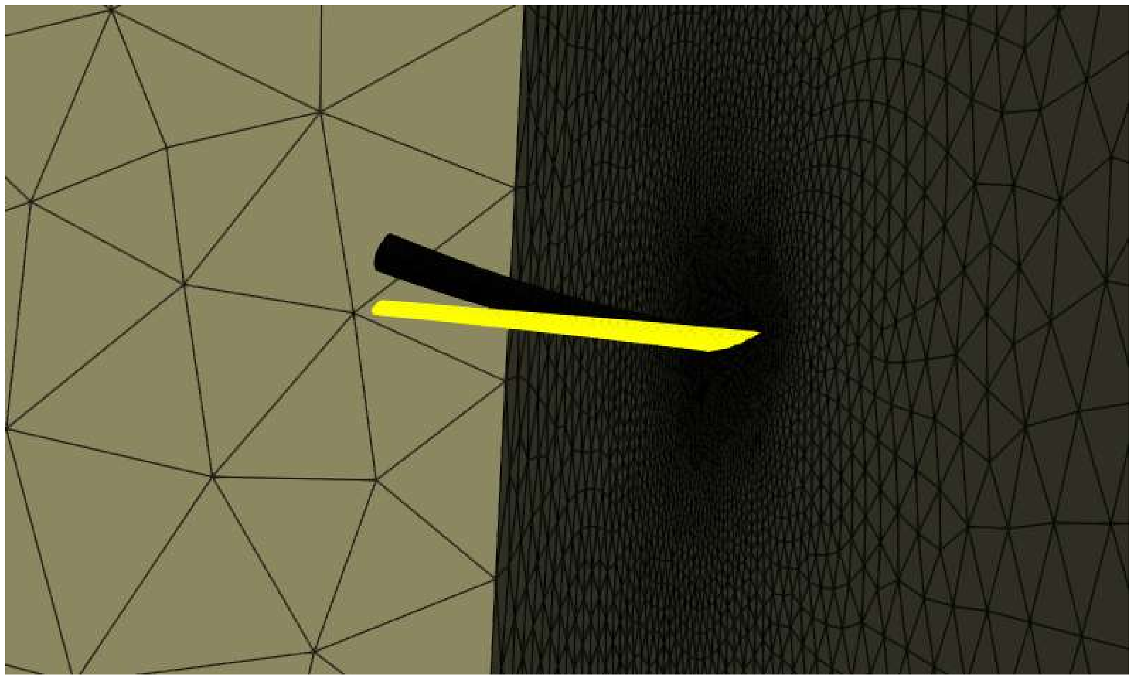
Σχήμα 10: $ONERA$ M6 κάμψη, $a = 0.2$. Μαύρο: Τελική θέση φτερού. Κίτρινο: Αρχική θέση φτερού.



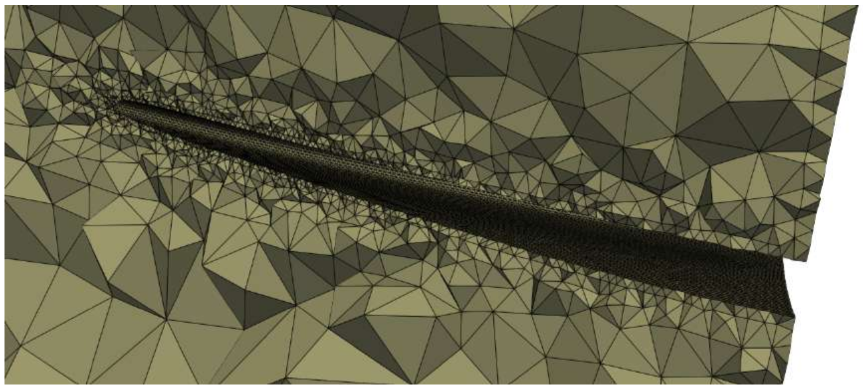
Σχήμα 11: $ONERA$ M6 κάμψη, $a = 0.2$. Πλέγμα κοντά στο φτερό.



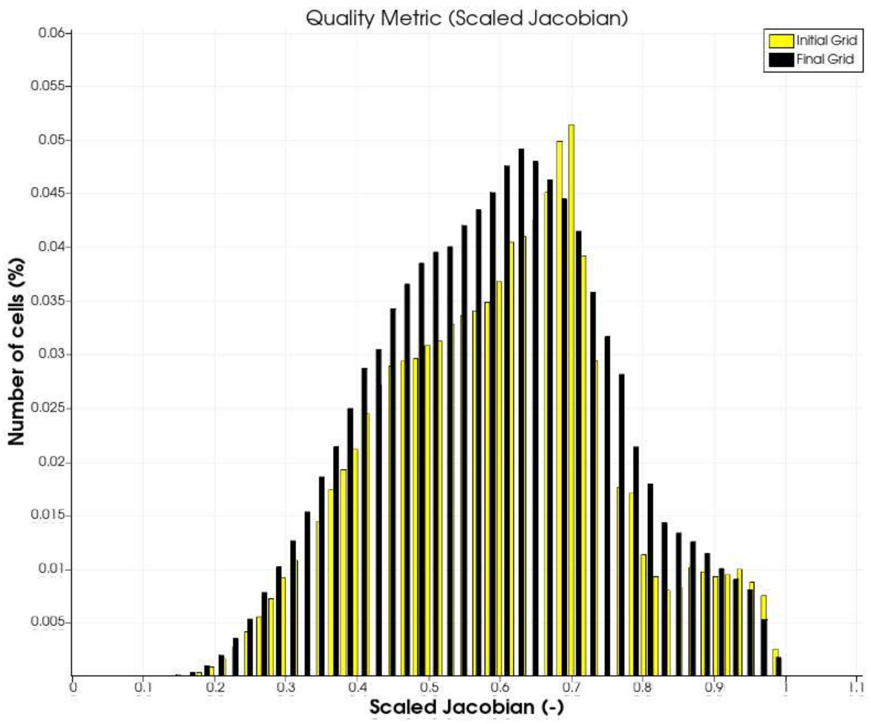
Σχήμα 12: *ONERA* M6 κάμψη, $a = 0.2$. Ιστόγραμμα ποιότητας, μετρική Scaled Jacobian Μαύρο: Τελικό πλέγμα. Κίτρινο: Αρχικο πλέγμα.



Σχήμα 13: *ONERA* M6 συνδυασμένη κάμψη και συστροφή, $a = 0.1$. Μαύρο: Τελική θέση φτερού. Κίτρινο: Αρχική θέση φτερού.



Σχήμα 14: $ONERA$ M6 συνδυασμένη κάμψη και συστροφή, $a = 0.1$. Πλέγμα κοντά στο φτερό.



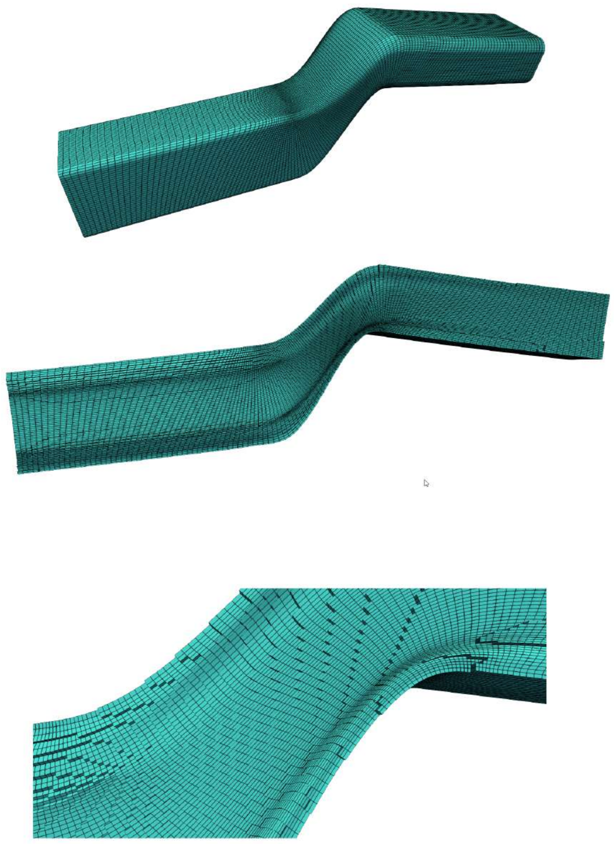
Σχήμα 15: $ONERA$ M6 συνδυασμένη κάμψη και συστροφή, $a = 0.1$. Ιστόγραμμα ποιότητας, μετρική Scaled Jacobian Μαύρο: Τελικό πλέγμα. Κίτρινο: Αρχικο πλέγμα.

3Δ αγωγός σχήματος S

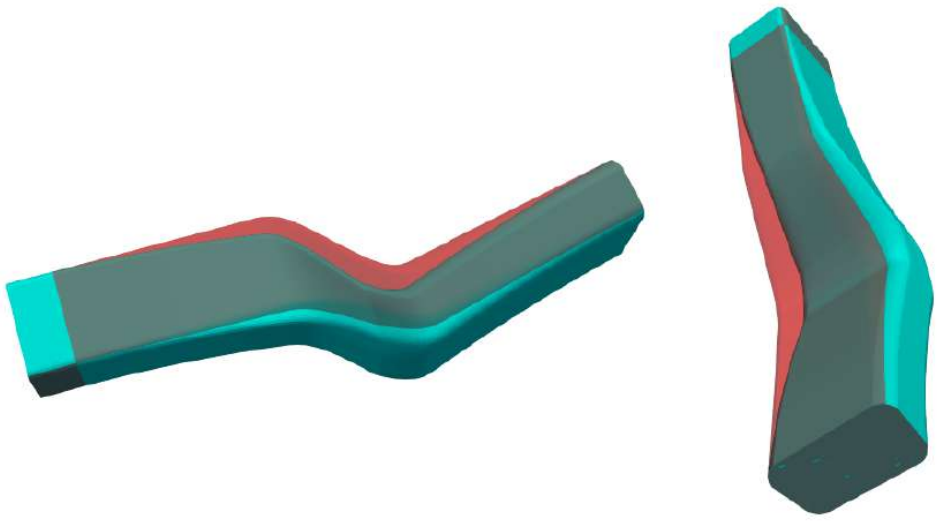
Στη συγκεκριμένη εφαρμογή, ένας αγωγός σχήματος S παραμορφώνεται παρομοιάζοντας την παραμόρφωση που θα δεχόταν κατά την αεροδυναμικής βελτιστοποίησης μορφής. Η παραμόρφωση που του επιβλήθηκε εκφράζεται από τις παρακάτω εξισώσεις:

$$x^{\text{new}} = x^{\text{old}} \tag{11a}$$
$$y^{\text{new}} = y^{\text{old}} + \alpha(|x^{\text{old}}_{\text{min}}| - |x^{\text{old}}|)(|x^{\text{old}}_{\text{max}}| - |x^{\text{old}}|) \tag{11b}$$
$$z^{\text{new}} = z^{\text{old}} + \alpha(|x^{\text{old}}_{\text{min}}| - |x^{\text{old}}|)(|x^{\text{old}}_{\text{max}}| - |x^{\text{old}}|) \tag{11c}$$

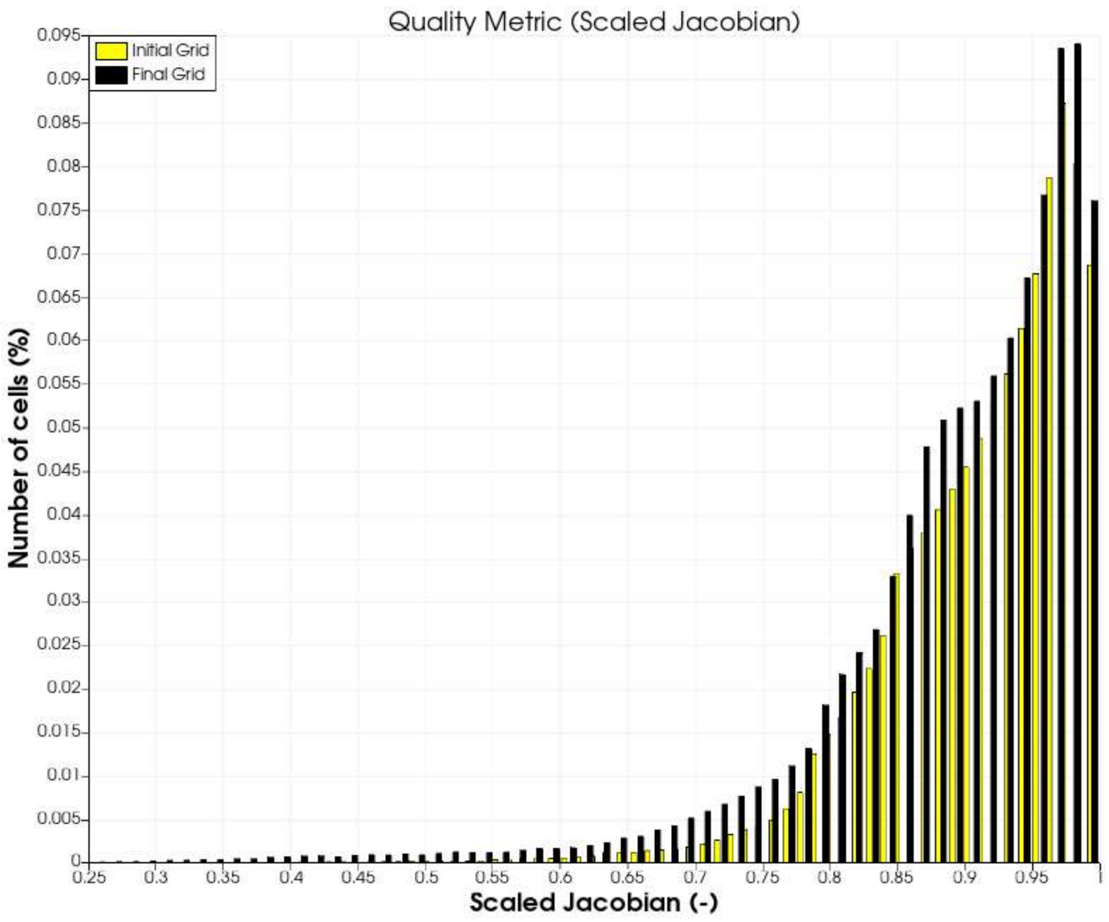
Η γραμμικοποιημένη συζευγμένη μέθοδος χρησιμοποιήθηκε και το τελικό πλέγμα παρουσιάζεται στην παρακάτω εικόνα. Στην συνέχεια παρουσιάζονται το αρχικό και τελικό σχήμα του αγωγού, όπως επίσης το ιστόγραμμα με την σύγκριση της ποιότητας των αντίστοιχων πλεγμάτων.



Σχήμα 16: Αγωγός σχήματος S, παραμόρφωση μορφής με $a = 0.3$.Τελικό πλέγμα.



Σχήμα 17: Αγωγός σχήματος S, παραμόρφωση μορφής με $a = 0.3$. Κόκκινο: Αρχική γεωμετρία. Γαλάζιο: Τελική γεωμετρία.



Σχήμα 18: Αγωγός σχήματος S, παραμόρφωση μορφής με $a = 0.3$: Ιστόγραμμα ποιότητας, μετρική Scaled Jacobian Μαύρο: Τελικό πλέγμα. Κίτρινο: Αρχικο πλέγμα.

Ανακεφαλαίωση-Συμπεράσματα

Η ανάλυση των αποτελεσμάτων αποκαλύπτει ότι η χρήση των γραμμικοποιημένων εξισώσεων της συζευγμένης μεθόδου αποδεικνύεται ως η βέλτιστη μέθοδος όσον αφορά το υπολογισιτκό κόστος αλλά και την ποιότητα του τελικού πλέγματος. Η χρήση βαρών την ενίσχυσε έτσι ώστε ακόμα και σε μεγάλες παραμορφώσεις, να καταφέρνει να προσαρμόσει το πλέγμα χωρίς την εμφάνιση ανεστραμένων κελιών, και χωρίς την ανάγκη χρήσης υποβημάτων. Παρόλαυτα, σε ακραίες παραμορφώσεις κρίνεται ακατάλληλη και προτείνεται η χρήση της μη γραμμικής συζευγμένης μεθόδου, αφού παρόλο το αυξημένο κόστος της, μπορεί να προσαρμόσει το πλέγμα σε αυτές τις ακραίες παραμορφώσεις. Όσον αφορά τον επιλύτη, η χρήση του διαγώνιου προσταθεροποιητή κρίνεται αρκετά σημαντική, αφού επιταγχύνει την σύγκλιση σε μεγάλο βαθμό.