TÉCNICO LISBOA

Unstructured polyhedral mesh generator with improved grid quality metrics

Manuel Dias da Costa

Thesis to obtain the Master of Science Degree in

Mechanical Engineering

Supervisors: Prof. José Carlos Fernandes Pereira
Dr. Duarte Manuel Salvador Freire Silva de Albuquerque

Examination Committee

Chairperson: Prof. Carlos Frederico Neves Bettencourt da Silva
Supervisor: Dr. Duarte Manuel Salvador Freire Silva de Albuquerque
Member of the Committee: Prof. José Paulo Baptista Moitinho de Almeida

June 2018

”After a while I went out and left the hospital
and walked back to the hotel in the rain.”
- Ernest Hemingway, *A Farewell to Arms*

Acknowledgements

”Um desporto individual, ganho por equipas”, é uma frase muito repetida para descrever o ciclismo de estrada e que se aplica perfeitamente à execução desta Tese. Aqui ficam os agradecimentos à minha ”equipa”:

O meu primeiro agradecimento é dirigido à minha família. Particularmente, Mãe, Pai, Bárbara e Avó São que sempre me apoiaram e fizeram tudo o que podiam para que conseguisse concluir este trabalho. Sei que durante os últimos anos nem sempre lhes dei o devido valor mas fica aqui o meu agradecimento pela compreensão e carinho.

A todos os meus amigos, e começando por aqueles a quem nem consigo explicar sobre o que é esta Tese e que mesmo assim compreendem as, por vezes longas, ausências. Afonso, Carolina, Pedro, Raminhos e Ramos o meu muito obrigado. Um agradecimento especial também ao Miguel, que sabe do que estou a falar quando digo ”malhas”, por partilhar comigo as suas experiências e com isso melhorar este trabalho. Finalmente uma palavra para aqueles que me acompanharam ao longo dos últimos anos e a quem consigo explicar a Tese, compreendem os desabafos e sugerem soluções. Especialmente ao Francisco e à Catarina, com quem estudei e trabalhei muito nos últimos anos e que tiveram de ouvir todas as minhas teorias e opiniões.

Ao Dr. Duarte Albuquerque, orientador desta Tese, fica o meu agradecimento pelas ideias sugeridas, pela revisão deste documento e por todo o conhecimento partilhado, sem os quais este trabalho não seria possível. Fica ainda algo, que para mim é, ainda mais importante que estes contributos, o interesse demonstrado por este trabalho e disponibilidade para ajudar. Sei que fui um privilegiado e que não é comum ter um orientador tão acessível e disposto a ajudar e por isso aqui fica o meu sincero agradecimento e reconhecimento.

Resumo

Nesta tese, um gerador de malhas em duas dimensões, designado como *Formiga* (Forced Mesh Improvement and Generation Algorithm), e capaz de gerar tanto malhas triangulares como poliédricas é desenvolvido com o intuito de melhorar o erro numérico em aplicações com o método de volume finito. Este gerador tem como principal característica o algoritmo de suavização, iterativo, baseado num método das forças. Este algoritmo processa a informação fornecida pelos critérios de qualidade da malha, específicos ao método de volume finito, e desloca os vértices da malha de forma a minimizar os seus valores.

Alterações de conectividades em ambos os tipos de malha e rotinas para que o gerador de malhas poliédricas seja capaz de lidar com domínios côncavos são também características basilares deste novo gerador.

Durante este trabalho são apresentadas várias malhas, geradas sobre vários casos geométricos com o fim de demonstrar a robustez do algoritmo implementado. São também estabelecidas comparações com outros geradores de malhas triangulares e poliédricas tendo em conta os critérios de qualidade de malha estabelecidos.

As malhas geradas são usadas na simulação numérica de problemas com solução analítica para que se comprove que os métodos desenvolvidos surtem o efeito desejado no erro numérico. Como consequência deste estudo surge uma comparação entre a precisão de malhas triangulares e poliédricas e também uma comparação entre quatro famílias de esquemas de discretização especificas para malhas não-estruturadas.

Palavras-chave: Gerador de malhas; Malhas poliédricas não estruturadas; Método de Volume Finito; Algoritmo de suavização de malha; Critérios de qualidade da malha; Precisão numérica

Abstract

In this Thesis, a two dimensional mesh generator, named *Formiga* (acronym for Forced Mesh Improvement and Generation Algorithm), capable of creating triangular and polyhedral grids is developed with the end goal of increasing numerical accuracy in simulations with the Finite Volume Method. The main characteristic of the generator is the iterative smoothing algorithm based in the forcing approach and Finite Volume specific grid quality metrics, which displaces the vertices of the mesh so that the said metrics are improved.

Other important features include the change in mesh connectivities for different cell types and an algorithm to endow the polyhedral meshes with the ability of handling non-convex geometries.

Meshes over several geometries are presented to showcase the robustness of the developed algorithm and comparisons with other triangular and polyhedral mesh generators are established to prove the quality of the meshes obtained using the proposed approach.

These meshes are also used to perform the numerical simulation of problems with a known analytical solution to prove that the applied methods decreases the numerical error. Given that both triangular and polyhedral meshes are tested a comparison between cell shapes is also performed along with a review of four families of discretization schemes for unstructured grids.

Keywords: Mesh generation; Unstructured polyhedral meshes; Finite volume method; Grid smoothing algorithm; Grid quality metrics; Numerical accuracy

Contents

List of Tables

List of Figures

Nomenclature

Abbreviations

1D	One-dimensional
2D	Two-dimensional
3D	Three-dimensional
AIAA	American Institute of Aeronautics and Astronautics
`bbox`	Bounding box
CFD	Computational Fluid Dynamics
CV	Control Volumes
CV	Centroidal Voronoi Tessellation
FEM	Finite Element Method
`fd`	Signed distance function
`fh`	Desired edge length function
Formiga	Forced Mesh Improvement and Generation Algorithm
FVM	Finite Volume Method
LASEF	Laboratory of Simulation of Energy and Fluids
`pfix`	Fixed points
SIMPLE	Semi-Implicit Method for Pressure-Linked Equations
V&V	Verification and Validation
WLS	Weighted Least Squares Method

Symbols

α_N	Warp angle or Non-orthogonality angle
γ	Concavity metric
`dist`	Output of `fd`
$\mathbf{d}$	Vector between $\mathbf{P}_0$ and $\mathbf{P}_1$ for interior faces and $\mathbf{f}$ and $\mathbf{P}_0$ for exterior ones
ΔC	Displacement
η	Blending factor
f	Face
$\mathbf{f}$	Coordinates of the face centroid f
$\mathbf{f_i}$	Coordinates of the intersection of $\mathbf{d}$ with face f
$\mathbf{F}$	Sum of used forces
$\mathbf{F_{dist}}$	*Distmesh* correction force
$\mathbf{F_{ort}}$	Non-orthogonality correction force
$\mathbf{F_{skew}}$	Skewness correction force
$\mathbf{F_{unif}}$	Uniformity correction force

f_x	Uniformity metric
h_{ref}	Target distance between points in initial distribution
h_{max}	Maximum distance allowed between points in initial distribution
κ	Diffusion coefficient
$\mathbf{m}$	Skewness vector
ν	Kinematic viscosity
$\mathbf{p}$	Coordinates of mesh vertice
P_n	Cell or Control volume
$\mathbf{P}_n$	Geometric centre of cell P_n
r, θ	Polar coordinates
$\mathbf{S}_f$	Outward pointing face area vector
$\mathbf{u}$	Velocity field
ϕ	Transported variable or computational variable
ψ	Skewness metric
Ω	Arbitrary cell or control volume
$\nabla \cdot \mathbf{u}$	Divergence
$\lvert\lvert\mathbf{u}\rvert\rvert$	Euclidean norm
$\nabla\phi$	Gradient

Chapter 1

Introduction

1.1 Motivation

The interest in Computational Fluid Dynamics (CFD) has been increasing over time following the progresses in computing power observed in recent decades. These advances allow for the solution of difficult and complex problems that make CFD useful not only in industrial applications, but also in medicine, and even sports like road cycling or F1 racing.

Along with its widespread use comes the need to guarantee accurate and reliable solutions. To achieve this goal these solutions must undergo a process usually called Verification & Validation (V&V). AIAA [1] defined Verification as “the process of determining that a model implementation accurately represents the developer’s conceptual description of the model and the solution to the model.” and Validation as “the process of determining the degree to which a model is an accurate representation of the real world from the perspective of the intended uses of the model”.

The Verification process may be split into two different stages Code Verification and Solution Verification [2]. The former occupies itself with identifying and correcting programming errors which are one of five main sources of numerical error in CFD solutions [2]. Solution Verification tackles the other four main sources ”insufficient spatial discretization convergence, insufficient temporal discretization convergence, insufficient convergence of an iterative procedure, computer round-off” [2].

Spatial discretization error and temporal discretization error as in [3] or just discretization error is often the main error source in CFD [3]. This type of numerical error comes from both the discretization of the mathematical model to be solved and the discretization of the domain on which the simulation is carried out. In order to discretize the mathematical model there are several methods available, being the Finite Element Method (FEM) and the Finite Volume Method (FVM) the most common ones. This last one is used on most commercial, open source CFD softwares and in SOL, the in-house code at the Laboratory of Simulation of Energy and Fluids (LASEF).

Domain discretization, commonly referred as meshing, consists in dividing the geometry in analysis into small finite parts in which the discretized equations may be reasonably approximated and solved. The large scope of existing meshes may be divided in two families: structured and unstructured. The first are hard to automatically generate for anything but simpler geometries while the later require much less user interference to generate a valid mesh even in complex geometries. However this comes at a cost, unstructured meshes cause less accurate results than their structured counterparts. There is plenty of research focused in improving unstructured meshes accuracy spread over many a field, amongst them being the improvement of meshes grid quality estimations and the study of different cell types. The first focuses on developing algorithms that measure and improve metrics thought to influence the accuracy of a numeric simulation while the later is focused on studying alternatives to triangular (2D) and tetrahedral (3D) meshes that are the norm in CFD applications. Two previous studies [4, 5] done in LASEF revealed that polyhedral cells with arbitrary number of faces were more accurate than triangular ones.

This Thesis contributes through a new mesh generator called *Formiga* (acronym for Forced Mesh Improvement and Generation Algorithm), capable of generating polyhedral meshes from existing triangular meshes and improving either one of them using several methods based on mesh quality metrics. The meshes created are then tested in SOL with the purpose of verifying if the changes introduced positively affect the accuracy of the Finite Volume Method.

1.2 Topic Overview

Unstructured meshes don't obey a regular order of nodes or cells and are not restricted to a single shape of cell or element, existing in triangles, quadrilaterals and polygons with an arbitrary number of faces when in 2D and the 3D equivalents tetrahedra, hexahedra, prisms and other type of cells [2]. There's a wide variety of methods available to generate this kind of mesh depending on the shape of cell preferred.

For triangular or tetrahedral meshes the three main groups identified [6] are Delaunay triangulation, Advancing Front and Quadtree (in 2D or Octree in 3D). Details about each one of these may be found in [6, 7]. Quadrilateral and hexahedral meshes have drawn in much interest over the years since these meshes are known for their improved convergence speed and accuracy when compared to their triangular/tetrahedral counterparts [8]. In what regards to quadrilateral and hexahedral mesh generation there's still a long way to go for it to reach the level of automation of triangular/tetrahedral mesh generation. There are nonetheless several generation methods available split into direct and indirect [6]. Direct methods include [9] grid-based, skeleton-based, whisker weaving and plastering. In these methods the mesh is created by placing hexahedral cells without resorting to a previous tetrahedral mesh. In indirect methods, a tetrahedral mesh is first placed in the domain and is then converted to hexahedral. Several algorithms exist such as Q-Morph [10], others may be found in [11].

Polyhedral meshes try and bridge the gap between triangular and strictly quadrilateral meshes being easier to automatically generate than the later and more accurate than the former. Similarly to quadrilateral meshes, polyhedral mesh generation algorithms are also split between direct and indirect methods. The direct method would be to use a method called the Centroidal Voronoi Tessellation (CVT), that may also be used to create triangular meshes, in which starting from an initial point distribution a Voronoi Tessellation is created where the generating points or seeds are the geometric center of the resulting Voronoi cell. A Voronoi Tesselation is a division of a plane into parts, creating convex polygons.

In [12] there's a more in dept description of this method by the same authors who made available the *Polymesher* [13], that is a 2D MATLAB polyhedral mesh generator that uses CVT, which is very dependent on the initial point distribution to provide high quality meshes hence there exist several algorithms to propose the best placement for the seeds, such as the Lloyd's method and the MacQueen's method, both described in [14] along with other related methods.

The indirect methods use a triangular mesh as a primal mesh from which a polyhedral one is created. In [15] two ways of achieving this goal are mentioned. In the first polygon's are created by bisecting the lines between each triangle's central point and it's neighbors. According to [15] this method is used by both *Qhull* [16] and *Voro++* [17]. A full description of this method is given in [18].

The second indirect method is to compute the dual of a Delaunay triangulation, this method ,often referred to as dual method, is explained in detail in both [15, 19] and shall be addressed during this work as this is the method chosen to produce polyhedral meshes.

Over the last few paragraphs some remarks were made regarding the accuracy of each mesh type with no justification, here some studies that compared cell shapes are presented. Firstly in [20] Juretić compared triangular, quadrilateral and polyhedral meshes in two dimensions concluding that quadrilateral cells were the most accurate (for an equivalent cell number) followed by polyhedral and only then triangular meshes, justifying these results with the existence of face pairs in the first two shapes, which cancel a contribution to the discretization error. Perić [21] compares polyhedral meshes with triangular ones noting that the first ones have more accurate cell centered gradients, attributed to the larger number of neighbors, and also that polyhedral meshes need a smaller number of control volumes to achieve the same level of accuracy. The authors of [22] compared tetrahedral and polyhedral meshes in cerebral aneurysm geometries and verified that polyhedral meshes required less control volumes to solve the problem at hand and therefore they converge faster. Finally in [23] several radiation problems are solved in both triangular and polyhedral meshes from where the main conclusion was that the simulations with the later ran much faster than with the former.

To sum the information above, quadrilateral meshes are preferable when a relatively easy geometry is used. In the case that it is not possible, polyhedral meshes appear to be the second choice, being almost as easy to generate as triangular ones and by all accounts requiring less cells to achieve the same level of accuracy.

Choosing the "best" mesh topology does not guarantee the most accurate result. Like with most things, there are good and bad meshes, independently of cell shape. To judge grid quality several metrics exist though for the purpose of this work only criteria applicable to cells with arbitrary number of faces are considered. Juretić in [20] described three metrics: non-orthogonality, skewness and uniformity, which are directly related to the discretization schemes for unstructured meshes. These metrics measure some grid's characteristics, at the face of the control volumes, known to affect the accuracy of numerical solutions.

In [24] four metrics are addressed of which the *Hybrid mesh quality metric I* and *II* appear to be the most promising. These particular metrics were created with information about the cell's size and shape. In the same study these metrics are used to evaluate mesh quality but also to improve it by the means of an optimization based smoothing algorithm.

Smoothing algorithms are mesh improvement methods that alter node/vertex position to improve grid quality. Among these methods are the above mentioned optimization based algorithms in which an objective function is minimized, as an example, in [24] the objective function might be given by *Hybrid mesh quality metric I* for each vertex and minimized by Newton's method. Other optimization based algorithms are mentioned in [25] and [26]. Another possibility is called *Laplacian smoothing* [27] in which a node is moved to the geometric centre of it's nearest neighbours. Improvements on this method exist and the reader is referred to [25, 28] for more details. Other approach is the one proposed in [29] by Persson and Strang, called *force-based method* in which displacements proportional to a metric of interest are applied iteratively, improving node distribution and consequently its mesh quality. In their work, these displacements are applied during the generation of triangular meshes however it stands to reason that the same concept may be applied to any already defined mesh.

1.3 Objectives

The main goal of this work is to create a 2D mesh generator whose grids are able to positively impact the accuracy of numerical simulations with the Finite Volume Method, by improving related grid quality parameters. To achieve this main goal several tasks must be completed, first:

I. Robust 2D polyhedral mesh generator capable of handling non-convex geometries;

II. Smoothing algorithm based on the *force based* method and FVM specific grid quality metrics, capable of improving cells with an arbitrary number of faces;

III. Compare the grid quality metrics of the created meshes with ones from other established mesh generators;

IV. Study the numerical performance of these grids for different analytical solutions;

V. Compare the numerical accuracy of triangular and polyhedral meshes.

1.4 Present Contributions

This Sections shines a light over the contributions made in this work, most of them associated with the mesh generator created.

Formiga was developed in MATLAB with the initial concept of taking triangular meshes from the well established mesh generator, *Distmesh* [29] and compute it's dual corresponding mesh, resulting in a polyhedral mesh that would then be improved upon. A function to compute this dual mesh was taken from a previous work done in LASEF [4], however that version was not able to handle non-convex geometries and so it was slightly modified to accommodate for a new function to handle concave cells in it's place.

Aside from these two contributions from previous works, the remaining part of the generator was almost all developed by the current author. In broad strokes this includes: improvement of *Distmesh*'s robustness and speed; measuring and computing grid quality metrics; applying connectivity changes to triangular grids, being through edge eliminations, edge swaps or Constrained Delaunay triangulations, the later using MATLAB function `delaunayTriangulation`; improving vertex position through the soon to be presented smoothing algorithm and changing polyhedral grids connectivities by both edge and cell eliminations. These in addition to the triangular mesh generation and application of the dual method constitute the essential building blocks of *Formiga* that at a certain point of its development evolved from not only being able to improve polyhedral grids but triangular ones as well. This did not imply a big overhaul of the already programmed functions as these were already designed for cells of arbitrary number of faces but rather punctual modifications to cater to triangular meshes characteristics that resulted in new functions in order to change their connectivities.

Other non essential but useful functions were created to represent the created meshes, plot the grid quality values, map said grid quality values on the actual meshes and export them to an Excel file.

All numerical simulations were carried out in SOL therefore functions to export the created meshes into an usable format, STARCD/PROSTAR (v4), were needed. These were already available for both triangular and polyhedral meshes, although the later had to be corrected to work properly for some cases.

In addition to all these routines for 2D mesh generation, a 3D version of *Formiga* was also created in MATLAB with the main objective of proving the smoothing algorithm's effectiveness in this higher dimension. Following the same principles as the 2D version it computes the dual of a primal, tetrahedral mesh generated through `distmeshnd`, a 3D version of *Distmesh*, and improves it with the aforementioned smoothing algorithm, that may also be applied to the tetrahedral meshes.

Even though this 3D algorithm does not handle non-convex geometries nor is capable of performing connectivity modifications as these are not essential to prove the concept, code modifications were required to compute the dual mesh, measure grid quality metrics, apply the smoothing algorithm and create graphical representations. New functions to export the meshes to the STARCD/PROSTAR (v4) file format also had to be implemented in the current work for both tetrahedral and polyhedral meshes.

Several readers for different mesh file formats were also created, this includes *Gmsh*'s file format, ”MSH”, *Triangle*'s `.node` and `.ele` files, the STARCD/PROSTAR (v4) file format used on OpenFOAM and STAR CCM+'s 9 boundary shell format.

1.5 Thesis Outline

This work is divided in five Chapters.

Chapter 2 starts by introducing the grid quality metrics used in this work and the proposed techniques to minimize them. It then moves on to a brief review of the FVM mathematical formulation and the discretization methods used. The rest of the chapter is concerned with detailing all steps necessary to create *Formiga*'s meshes.

Chapter 3 showcases the improvements made with the generated meshes in regards to the grid quality parameters used. Comparisons are then established with other mesh generators to provide some perspective over the presented values.

In the 4th Chapter the meshes created are used in several numerical simulations, for which the analytical solutions are known, to find if the changes made have a measurable impact in the numerical accuracy.

The 5th and final Chapter evaluates the work done, focusing on the relation between the used grid quality metrics and the numerical error of the solution, an analysis of the obtained results with both triangular and polyhedral meshes is also carried out. This work is concluded with a collection of topics to be addressed in future works.

Chapter 2

Numerical Methods and Grid Generator Algorithm

In this Chapter all methods concerning the generation and improvement of both triangular and polyhedral meshes are addressed. Both 2D and 3D meshes with an arbitrary number of faces are referred to as polyhedral meshes. Section 2.1 refers to the quality metrics used and corresponding proposed corrections. Section 2.2 sums up the Finite Volume schemes used. Sections 2.3 through 2.8 resume the techniques used to create the featured generator, *Formiga*. Being primarily focused on 2D, throughout these Sections there are instances where issues related to 3D mesh generation are addressed as well.

2.1 Mesh Quality Criteria and Proposed Corrections

2.1.1 Mesh Quality Criteria

The mesh quality criteria chosen for this study are the ones proposed by Juretić [20]: non-orthogonality, skewness and uniformity. All of which are measured on the faces of the Control Volumes (CV). Figure 2.1 introduces the nomenclature used.

A face is considered non-orthogonal when the angle α_N between vectors $\mathbf{d}$ and $\mathbf{S}_f$, which may be

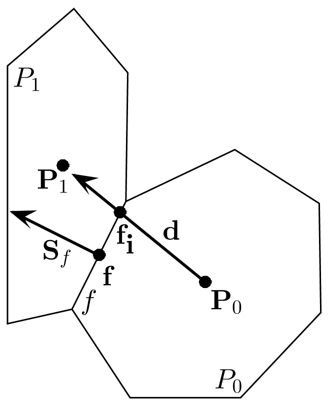


Figure 2.1: Two polyhedral cells with significant entities represented.

expressed by Eq. 2.1, is different from zero. With $\mathbf{d} = \mathbf{P}_1 - \mathbf{P}_0$ and $\mathbf{S}_f$ exterior face normal. $\mathbf{P}_1$ and $\mathbf{P}_0$ are the coordinate vectors of the geometric centers of cells P_1 and P_0, respectively.

$$\alpha_N = \arccos\left(\frac{\mathbf{d}.\mathbf{S}_f}{||\mathbf{d}||\,||\mathbf{S}_f||}\right) \tag{2.1}$$

This quality criterion, that may also be referred to as warp angle, is evaluated in every face of the mesh, including boundary faces. In these situations vector $\mathbf{d}$ is defined by $\mathbf{d} = \mathbf{f} - \mathbf{P}_0$, where $\mathbf{f}$ represents the face geometric centre.

Skewness or eccentricity is a measure of the distance between a face geometric centre and the intersection of vector $\mathbf{d}$ with said face, $\mathbf{f_i}$ represents this intersection point.

In this work this criterion shall be computed through Equation 2.2 which was adapted from [5] for 2D cases. Where $\mathbf{m}$ is defined as $\mathbf{m} = \mathbf{f} - \mathbf{f_i}$. The reasoning behind this Equation is that in this form it's easy to gage the position of $\mathbf{f_i}$. When $\psi = 1$, means that $\mathbf{f_i}$ coincides with one of the vertices that defines the face and when $\psi = 0$, $\mathbf{f_i}$ coincides with $\mathbf{f}$. In [30] Jasak considers that $||\mathbf{m}||$ must be smaller than $||\mathbf{d}||$ for a grid to be of reasonable quality, meaning that skewness should be smaller than 1 through Juretić's metric [20] ($\psi = \frac{||\mathbf{m}||}{||\mathbf{d}||}$). A parallel is hard to establish among both metrics though for this study efforts will be carried out to keep skewness values smaller than 1, meaning that $\mathbf{f_i}$ must always lie within face f.

$$\psi = 2\frac{||\mathbf{m}||}{||\mathbf{S}_f||} \tag{2.2}$$

For 3D cases the metric employed is the same as in [5] though corrected as it should be multiplying by 2 instead of $\frac{1}{2}$, Equation 2.3. Again the principle is to create a dimensionless measure of $||\mathbf{m}||$ that has significance regarding $\mathbf{f_i}$'s position within a face. This concept does not pan out as well in 3D as in 2D, since the denominator in Eq. 2.3 represents the side of a square of equivalent area as the face in discussion. Meaning that for other shapes than a square, values larger than 1 won't necessarily mean that $\mathbf{f_i}$ lies outside the face's limits.

$$\psi = 2\frac{||\mathbf{m}||}{\sqrt{||\mathbf{S}_f||}} \tag{2.3}$$

In the way it is defined it only makes sense to measure skewness values, unlike non-orthogonality, in interior faces. The same goes for the following quality metric, uniformity.

Uniformity is a measure of the relative position of a face in regards to the cells that share it. If a mesh is uniform Eq. 2.4 will return $f_x = 0.5$.

$$f_x = \frac{||\mathbf{f_i} - \mathbf{P}_1||}{||\mathbf{d}||} \tag{2.4}$$

All three grid quality criteria have a close relation with the numeric schemes to be presented in Section 2.2, some of these schemes even propose methods to correct or mitigate the effects caused by these grid quality metrics. In the following Sub-Section corrections to be applied to these problems through a smoothing algorithm will be proposed, with the goal to reduce the prevalence of these induced errors.

2.1.2 Proposed Corrections

Before introducing the proposed corrections, matters to describe the *force-based method* employed in [29] and that shall be used to enforce the proposed corrections.

The *force-based method* is an iterative smoothing algorithm inspired by the analysis of trusses. By definition [31] ”A truss consists of straight members connected at joints. Truss members are connected at their extremities only; thus no member is continuous through a joint.”. The resemblance with computational meshes is striking and one could almost change the words ’truss’, ’members’ and ’joints’ for ’mesh’, ’edges’ and ’points/nodes’, respectively, and obtain a, though incomplete, mesh definition. Taking advantage of this analogy the method developed by Persson and Strang [29] applies forces to the points/nodes of a mesh, that are later converted in displacements to help the improvement of mesh quality throughout the generation process.

The conversion from Forces into displacements is done through Equation 2.5. In [29] and this work the same equation is approximated with the Forward Euler method (Eq. 2.6).

$$\frac{d\mathbf{p}}{dt} = \mathbf{F}(\mathbf{p}) \tag{2.5}$$

$$\mathbf{p_{n+1}} = \mathbf{p_n} + \mathbf{\Delta t F}(\mathbf{p_n}) \tag{2.6}$$

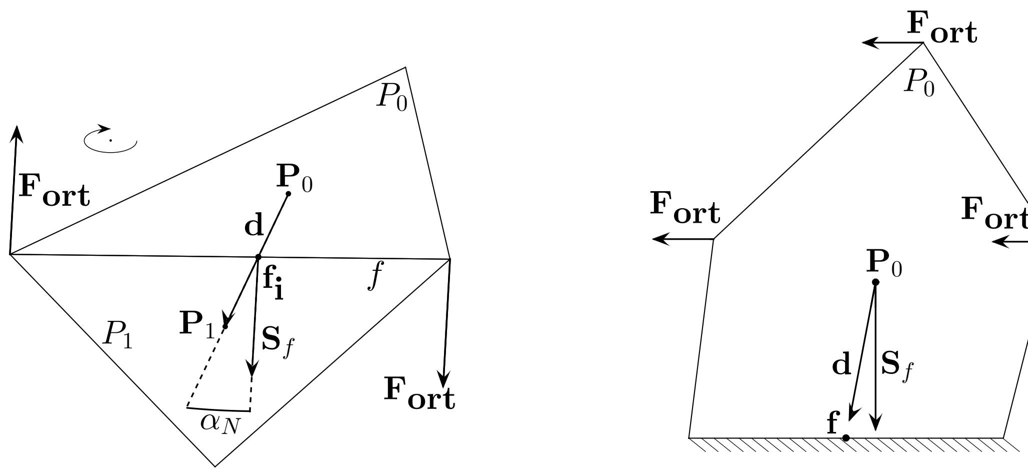
The force, $\mathbf{F(p)}$, used in *Distmesh* is modelled after Hooke’s law. The idea behind it being to make every edge in the mesh comply to the specified *desired edge length* function. This is done by computing the difference between each edge’s desired and actual length. For cases where this difference is larger than zero the computed value is multiplied by a constant k to obtain a force. If the above mentioned difference is smaller than zero the resulting force is zero. This means that no edge is allowed to reduce its size, they are only allowed to increase which is relevant as will be seen in Section 2.5.

Three corrections are proposed next, one for each of the presented quality metrics. All are directly proportional to a relevant metric concerning the respective quality criterion.

To reduce non-orthogonality or warp angles the strategy adopted was to apply forces with equal intensity but opposite directions to the vertexes that compose the edge, f. These forces are perpendicular to f and their intensity is directly proportional to the angle, α_N, subject to improvement, just like the previous force was directly proportional to the difference between the desired and the measured edge length. Figure 2.2(a) is an illustration of the described process.

This approach achieved the best results however there are other forms of achieving the same goal. One possibility might be applying a displacement to one of the edge’s points instead of both. The main advantage against the former method is that in situations where the improvement of this quality criterion directly reduces the quality of other metric, only moving one point allows more ”room” for the other corrective force to work with.

Another way to approach the situation is by going the opposite route and instead of trying to improve the warp angle by moving the face defining vertices, move the remaining cell’s points so that the centroids position change and vector $\mathbf{d}$ becomes parallel to $\mathbf{S}_f$. This approach did not achieve better results

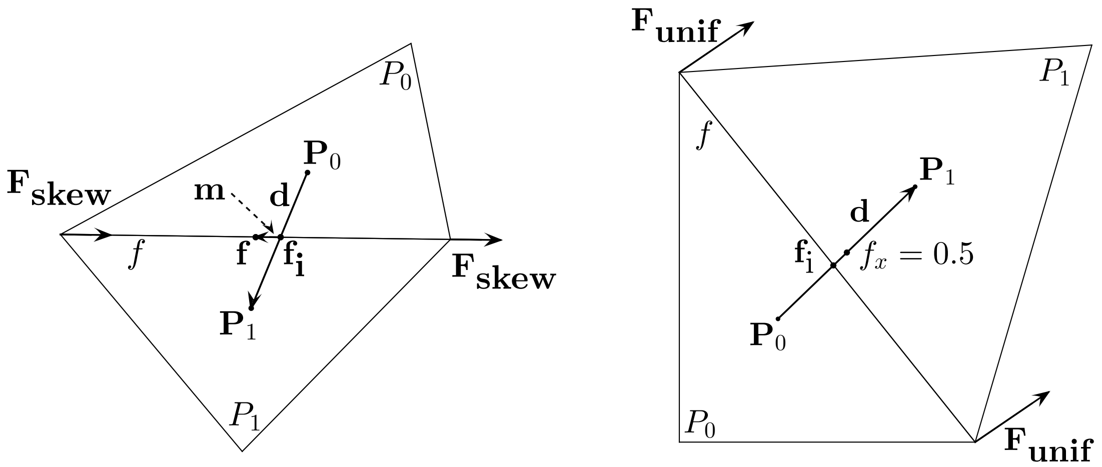


(a) Correction force in inner faces. (b) Correction force in boundary faces.

Figure 2.2: Corrections proposed for the non-orthogonality angle α_N.

than the ones already presented, however in situations where moving the face's vertexes is not an option (when improving boundary edges) is a valid option. As such, it is used to improve the non-orthogonality angle in boundary faces.. Figure 2.2(b) illustrates this method where, for clarity, the normal vector $\mathbf{S}_f$ was represented in the cell's centroid instead of in the actual face for the image to be more readable as in the presented way it's easier to gage the angle α_N. In the same spirit of making the images that make Figure 2.2, and the following, easily digested the corrections shown are for one face only.

From Eqs. 2.2 and 2.3 stands that to reduce skewness values we must shorten the distance between the edge's midpoint $\mathbf{f}$ and $\mathbf{f_i}$, and if possible, make them coincide. By applying a force in the opposite direction of $\mathbf{m}$ to the two vertexes that form the edge to be improved, Fig. 2.3(a), $\mathbf{f}$ is moving closer to $\mathbf{f_i}$. Note that by doing this $\mathbf{f_i}$ will change position as well however it shouldn't move at the same rate as $\mathbf{f}$ allowing for an approximation. Following the same principles as the forces above, this force is proportional to the length of vector $\mathbf{m}$.



(a) Correction force for skewness at face f. (b) Correction force for uniformity at face f.

Figure 2.3: Correction forces for skewness and uniformity.

Similarly to the non-orthogonality case one could try to change cell's centroid position to improve mesh quality. This would be done by applying forces parallel to the target edge to all cell's vertexes, except those that define the mentioned edge. This force would have the direction that would correct $\mathbf{f_i}$ position. As an example in Fig. 2.3(a) one would move the "free" points in the same direction as $\mathbf{m}$.

The method used to improve the uniformity criterion follows the exact same guidelines as the previous and is shown in Fig. 2.3(b). Unlike the other two quality criteria where it's clear that the goal is to make them as close to zero as possible, uniformity poses a different problem as this is the only error that may be introduced voluntarily. For an uniform mesh is clear that the displacements applied must render $f_x = 0.5$ on all faces. However for an user-defined size distribution different than that it's not as clear. To truly correct uniformity one would enforce $f_x = 0.5$ nonetheless but this would step over the user input resulting in an unwanted mesh.

Given this situation it was decided that these forces purpose would be to enforce the size distribution defined by the user and not mesh uniformity *per se*. The implications of this decision is that, depending on the situation, the forces applied might not correct uniformity at all.

In order to achieve the defined goal one must establish the reference values against which the uniformity will be measured. The relative size of each cell is measured using it's geometric centre and a ratio, is computed between it's size and the neighbouring cells, with whom it shares a face, Eq. 2.7. In *Formiga* this is achieved by using the *Distmesh*'s [29] input variable `fh` that represents a *desired edge length function*. This ratio will be the value the algorithm will try to enforce. To keep this reference values as unpolluted as possible this calculation is not performed in every iterative step but rather only in the first iteration and every time the grid's connectivities are modified.

$$cell\ ratio = \frac{\texttt{fh}(\mathbf{P}_0)}{\texttt{fh}(\mathbf{P}_0) + \texttt{fh}(\mathbf{P}_1)} \tag{2.7}$$

Considering everything mentioned about the uniformity criterion, it was felt that though used in the smoothing process the numeric values of uniformity are hard to interpret and easy to misjudge so these won't be presented in Chapters 3 and 4.

2.2 Discretization Schemes

The discretization of the mathematical model, i.e. conversion of the governing differential equations into an algebraic or system of algebraic equations is an essential part of any numerical simulation and to it belongs a part of the, so called, discretization error in addition to domain discretization. In this work only second order accurate schemes are used, all of which were already implemented in SOL code. As such there are already lengthy recounts and explanations about the schemes implemented in previous studies [4, 5, 32, 33]. This Section will only introduce the schemes used in future Chapters.

A brief introduction to the Finite Volume Method is made using the convection-diffusion equation, Eq. 2.8, where κ is the diffusion coefficient, and $\mathbf{u}$ the velocity field.

$$\frac{\partial \phi}{\partial t} + \nabla.(\mathbf{u}\phi) - \nabla.(\kappa \nabla \phi) = S_\phi \tag{2.8}$$

Considering a volume Ω, that in the FVM context may represent one cell, Eq. 2.8 may be integrated resulting in Eq. 2.9.

$$\underbrace{\frac{\partial}{\partial t}\int_{\Omega}\phi dV}_{Temporal\ term} + \underbrace{\int_{\Omega}\nabla.(\mathbf{u}\phi)dV}_{Convection\ term} - \underbrace{\int_{\Omega}\nabla.(\kappa\nabla\phi)dV}_{Diffusion\ term} = \underbrace{\int_{\Omega}S_{\phi}dV}_{Source\ term} \tag{2.9}$$

By applying the Divergence or Gauss theorem to the convective and diffusive terms one obtains Eq. 2.10, where $f = \partial\Omega$ and represents the limits (faces) of Volume Ω.

$$\frac{\partial}{\partial t}\int_{\Omega}\phi dV + \oint_{f}(\mathbf{u}\phi).d\mathbf{S}_f - \oint_{f}(\kappa\nabla\phi).d\mathbf{S}_f = \int_{\Omega}S_{\phi}dV \tag{2.10}$$

Until this point no approximations have been made, therefore Eq. 2.10 is still exact however to further advance in the Finite Volume Method and compute the presented integrals values of ϕ_f and $\nabla\phi_f$ are needed for the convective and diffusive terms, respectively. To get them the surface integrals above are approximated by the respective sum of face integrals, that may be approximated by the Gauss-Legendre quadrature.

In [32] four different families of convective and diffusive schemes are presented. In the following Sub-Sections each family is discussed, ahead of their application in Chapter 4. Discretization schemes for the temporal and source terms are not discussed here as they are not as relevant for this work.

2.2.1 Convective Schemes for classic and grid quality families

The first family of schemes does not take into account any information about the grid quality and as such may be described by Eq. 2.11, where η is the blending factor:

$$\phi_f = \eta\phi_{P_0} + (1-\eta)\phi_{P_1} \tag{2.11}$$

This blending factor is what defines the different schemes that belong to this family. Several possibilities exist for the definition of η though the chosen representative of this family for this Thesis is defined by Eq. 2.12. Using this blending factor the interpolation point ϕ_f is computed in the intersection of the face surface with vector $\mathbf{d}$, which should sound familiar as this position was already defined in Section 2.1 as $\mathbf{f_i}$ and results in the first scheme, $C-LIN$. As seen in the same Section, $\mathbf{f}$ and $\mathbf{f_i}$ don't always coincide, as matter of fact, in unstructured meshes, they rarely coincide. This proves the importance of the already introduced grid quality metric, skewness, that effectively measures the distance between these two points.

$$\eta_{LIN} = \frac{(\mathbf{f}-\mathbf{P}_0).\mathbf{S}_f}{(\mathbf{P}_1-\mathbf{P}_0).\mathbf{S}_f} \tag{2.12}$$

Taking this quality metric in consideration another family of convective schemes was proposed by Juretić [18] that tries to mitigate the skewness problem with a correction term. This family of schemes is introduced in Equation 2.13, where $(\overline{\nabla\phi})_f$ is the interpolated cell centered gradient vector.

$$\phi_f = \eta\phi_{P_0} + (1-\eta)\phi_{P_1} + \underbrace{(\mathbf{f} - \mathbf{P}_0 - \eta\mathbf{d}).(\overline{\nabla\phi})_f}_{\textbf{skewness correction}} \tag{2.13}$$

Considering the same blending factor $\eta_{_{LIN}}$ the correction term may be rewritten as in Eq. 2.14 in terms of the previously defined vector $\mathbf{m}$, resulting in a scheme that shall be referred to as $C-LIN-SK$.

$$(\mathbf{f} - \mathbf{P}_0 - \eta_{_{LIN}}\mathbf{d}).(\overline{\nabla\phi})_f = (\mathbf{f} - (\mathbf{P}_0 + \eta_{_{LIN}}\mathbf{d})).(\overline{\nabla\phi})_f = (\mathbf{f} - \mathbf{f_i}).(\overline{\nabla\phi})_f = \mathbf{m}.(\overline{\nabla\phi})_f \tag{2.14}$$

2.2.2 Diffusive Schemes for classic and grid quality families

Following the same logic as for convective schemes, first, the diffusive scheme belonging to the classic family that does not take any grid information into account is presented and then a scheme with corrections based on grid quality, that belongs to the second family, is presented.

Diffusive schemes are all about computing the gradient of the dependent variable at the geometric center, $\mathbf{f}$, of face f. A common choice for a scheme within the realm of the first defined family is the called central differences scheme, Eq. 2.15, that from here onwards will be called $D-NRML$.

$$(\nabla\phi)_f = (\phi_{P_1} - \phi_{P_0})\frac{\mathbf{S}_f}{\mathbf{S}_f.\mathbf{d}} \tag{2.15}$$

The scheme with grid quality correction belongs to a group of schemes proposed by Jasak in [30] where similarly to $C-LIN-SK$ a corrective term is added, Eq. 2.16. Three possible definitions for $\mathbf{q}$ are given in the same study, with Magalhães [5] concluding that the most accurate was the one given in Eq. 2.17, whose subscript, OR, stands for over-relaxed correction and names this scheme for the purpose of this thesis, $D-OR$. If for convective schemes the quality metric of interest was the skewness, for diffusive schemes the most relevant quality metric is the warp angle as it is a measure of how significant is the non-orthogonal contribution of $\mathbf{S}_f$ to the computation of $(\nabla\phi)_f$.

$$(\nabla\phi)_f = (\phi_{P_1} - \phi_{P_0})\mathbf{q} - \underbrace{[(\overline{\nabla\phi})_f - (\mathbf{d}.(\overline{\nabla\phi})_f)\mathbf{q}]}_{\textbf{warp angle correction}} \tag{2.16}$$

$$\mathbf{q}_{_{OR}} = \frac{\mathbf{S}_f}{\mathbf{S}_f.\mathbf{d}} \tag{2.17}$$

2.2.3 Ghost Points

Another family of schemes, initially presented in [34], noted that though the last introduced scheme corrected the inaccuracy related to the non-orthogonality angle in diffusive schemes it still left unattended the problem caused by skewed grids that force the computation of $(\nabla\phi)_f$ to occur in other point than $\mathbf{f}$. To address said problem the projection of points $\mathbf{P}_0$ and $\mathbf{P}_1$ onto a line parallel to the outward face normal ($\mathbf{S}_f$) that passes through $\mathbf{f}$ is proposed. This projection solves the accuracy issue of the non-orthogonality angle and skewness at the same time. Said projected points share their name with this

Sub-section and may be computed through Eq. 2.18, $n = 0, 1$. Their computational value may be interpolated through Equation 2.19, $n = 0, 1$.

$$\mathbf{P}'_n = \mathbf{f} - \frac{[(\mathbf{f} - \mathbf{P}_n).\mathbf{S}_f]\,\mathbf{S}_f}{\mathbf{S}_f.\mathbf{S}_f} \tag{2.18}$$

$$\phi_{P'_n} = \phi_{P_n} + (\mathbf{P}'_n - \mathbf{P}_n).(\nabla\phi)_{P_n} \tag{2.19}$$

Using these projected values in Eq. 2.15 one obtains $D-GP$, presented in Eq. 2.21 with $\mathbf{d}' = \mathbf{P}'_1 - \mathbf{P}'_0$. The same approach can be used to create a convective scheme, $C - GP$, represented in Eq. 2.20.

$$\phi_f = \eta\phi_{P'_0} + (1-\eta)\phi_{P'_1}, \quad \eta = \frac{(\mathbf{f} - \mathbf{P}'_0).\mathbf{S}_f}{(\mathbf{P}'_1 - \mathbf{P}'_0).\mathbf{S}_f} \tag{2.20}$$

$$(\nabla\phi)_f = (\phi_{P'_1} - \phi_{P'_0})\frac{\mathbf{S}_f}{\mathbf{S}_f.\mathbf{d}'} \tag{2.21}$$

2.2.4 Face Weighted Least Squares

The final family of numerical schemes uses the values from nearby faces to fit a linear polynomial centered in $\mathbf{f}$ containing both convective and diffusive values, Eq. 2.22. The regression analysis model, Weighted Least Squares Method, that lends it's name to this family of schemes ($C - FLS$ and $D - FLS$) is used to compute both the convective and diffusive terms, is presented and discussed in detail in [32]. Similarly to the previous family these schemes aim at correcting both orthogonality and skewness values by considering only the points location.

$$\phi(\mathbf{x}) = \phi_f + (\nabla\phi)_f.(\mathbf{x} - \mathbf{f}) \tag{2.22}$$

2.3 Distmesh and Primal Mesh Generation

Distmesh [29] is a MATLAB code created to generate triangular (2D) and tetrahedral (3D) meshes using the Delaunay triangulation algorithm. Starting with an initial random point distribution across a user defined bounding box (`bbox`), an initial triangulation of the set of points is carried out. The spacing between the vertices in this initial distribution is given by h_{ref} and set by the user. Making use of a *signed distance function* (`fd`) to represent the domain, the centroid of each newly formed triangle is evaluated. If a triangle's centroid is found outside the prescribed region then this cell will be eliminated. Details about *distance functions* may be found in [13, 35] though, in short, it is a function that evaluates vertex position returning values smaller than zero if the input is inside the defined domain and larger otherwise. Values of around 0 identify boundary points.

In order to improve mesh quality *Distmesh* employs the *force-based method* introduced in Section 2.1 after which is fairly common for points to be found outside of the domain. To amend this situation these points are projected back to the boundary, more on this will be explained in Section 2.5.

Given that it is public domain and the documented results [36] show good quality meshes, *Distmesh* was chosen to provide the initial mesh for *Formiga* to improve upon and generate polyhedral grids from. Through the duration of this work *Distmesh* proved to be a solid choice. However, a few problems were noted: the algorithm provides different results for the same user input when the *desired edge length function* (`fh`) is not uniform; there is no verification that the final mesh respects the prescribed domain. Both were also noted in [37] and this last one also identified by Persson and Strang [29]; and finally it was found that, though rarely, there were instances when cells were formed by three collinear points. Of these three flaws the last two are the most serious since they directly interfere with the ability of the proposed algorithm to convey a usable grid. Therefore solutions were created to correct them.

To verify that the boundary of the primal mesh is in agreement with the user input, the boundary of *Distmesh*'s grid is computed and compared to an estimate of said boundary, determined by finding which mesh points are within an established distance of `fd` = 0, that defines the domains's limits, and using them to create edges that best replicate the boundary. Once both structures are set the number of edges on the estimate is compared with the number of edges on the actual discrete boundary, easily found by identifying edges that belong only to one cell. If they don't agree the domain's frontier estimate is recalculated with slightly increased tolerances and re-compared. The aim of this repetition is to ensure the cause of disagreement is related to *Distmesh* and not to the boundary estimation. If they do agree a second, more restrictive step is reached where all the edges that form the boundary estimate have to be found in the list containing all mesh's edges. If there's at least one edge not found the boundary is re-estimated and the comparison process restarted.

Each mesh subjected to this evaluation process is allowed only one disagreement with the boundary estimation, upon the second one the mesh is discarded and *Distmesh* recalled to provide another mesh. To account for possible disagreements due to incorrect utilization of the input variables *Distmesh* may only be summoned up to five times. After reaching this limit the algorithm will terminate and return an error message.

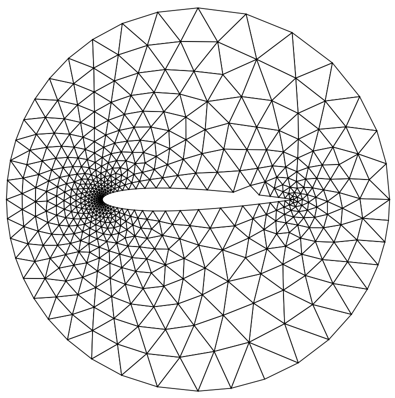


Figure 2.4: Example of primal mesh that doesn't conform to user specified boundary.

In Figure 2.4 is an example of a failed attempt to generate a mesh around an airfoil.

As for the possibility of existing cells composed by three collinear points the method employed is the following: Since this mistake was only found in cells near the boundary, the easiest solution found was to not allow the existence of triangles within a certain small distance to the boundary. In order to accomplish this, the centroids of said cells are computed and distance to the boundary is checked using the aforementioned *distance function* (`fd`). If this distance is inferior to a pre-established tolerance then the triangle is eliminated.

In Fig.2.5 is an example of a mesh where the situation was found. The predicament is shown in finer detail in Figure 2.5(b) where it is possible to see that there is no proper triangle, only three collinear points.

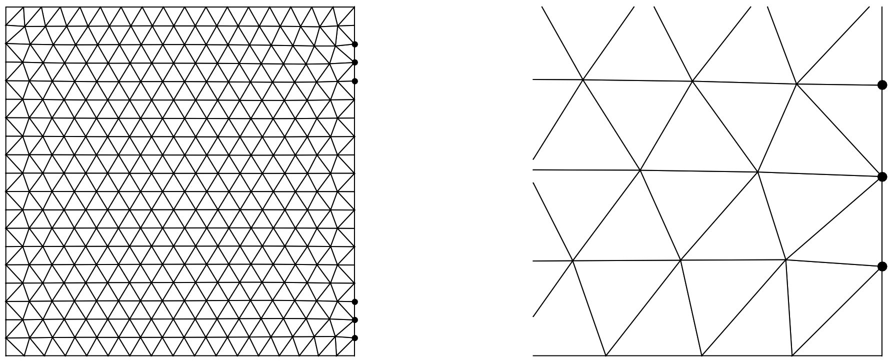

(a) Mesh where the first problem was identified. (b) Detail of the figure to the left.

Figure 2.5: Two "triangles" composed by collinear points obtained when creating an uniform mesh on a square domain with $h_{ref} = 0.06$.

Coupled with these additions aimed at increasing robustness a small change aimed at improving performance was made to `distmesh2d`. This change consisted of replacing the following line:

```
Ftot=full(sparse(bars(:,[1,1,2,2]),ones(size(F))*[1,2,1,2],[Fvec,-Fvec],N,2));
```

Whose responsible for converting the computed force, for each edge, to their nodes so that a displacement can effectively occur. The next few lines do the exact same thing though faster:

```
Ftot(:,1) = accumarray(bars(:,1),Fvec(:,1),[N 1]);
Ftot(:,1) = Ftot(:,1) - accumarray(bars(:,2),Fvec(:,1),[N 1]);
Ftot(:,2) = accumarray(bars(:,1),Fvec(:,2),[N 1]);
Ftot(:,2) = Ftot(:,2) - accumarray(bars(:,2),Fvec(:,2),[N 1]);
```

For this to be a good alternative to the previously existing code, `Ftot = [];` must be added inside the `if` statement called "Density control" to account for changes in the number of mesh points through the various iterations.

Large time savings are gained through this modification especially when the number of cells increases. As shown in Fig. 2.6 over a certain number of cells the time savings are so significant that in the time saved is possible to run the entire polyhedral version of *Formiga* meaning that in the time originally required to produce one triangular mesh, are now generated and improved one triangular and one polyhedral mesh. As a disclaimer the results shown in the mentioned Figure were obtained for meshes over an unit square with uniform point distribution. The times presented for each mesh size are averaged out of a 10 run pool using an AMD Ryzen R5 1600 with a base clock of 3.2 GHz running Windows 10.

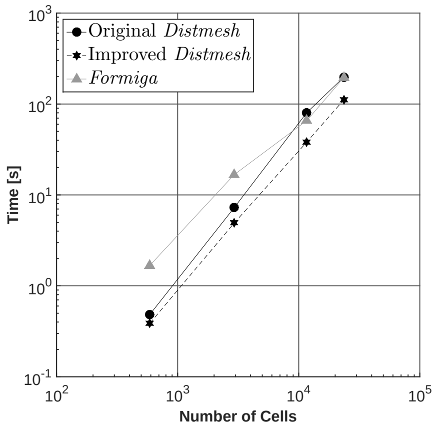


Figure 2.6: Example of primal mesh that doesn't conform to user specified boundary and time savings.

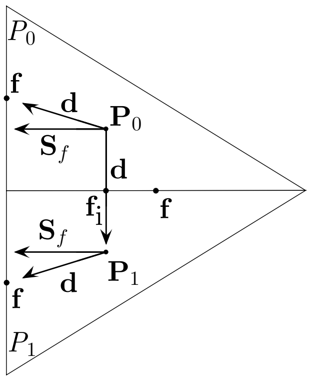
2.4 Changes in Triangular Meshes Connectivities

Distmesh provides good quality triangular meshes with appreciable consistency but *Formiga* is intended for use in Finite Volume applications therefore a couple of changes in mesh topology are employed to make the generated grids as fit, as possible, for the task at hand.

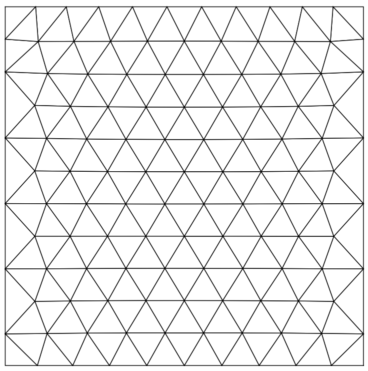
During the course of this study a characteristic was identified in several triangular meshes that results in poor quality faces according to the Finite Volume specific quality criteria introduced in Section 2.1. This is the presence of right-angled triangles on the domain's boundary. The term 'right-angled triangles' is employed loosely in this context since the triangles don't have in fact to be right-angled for this topology to be problematic, a more correct way of describing the situation would be: two boundary defining cells that also share an edge between them. Despite the remark this term shall be used during this study when referring to this problem.

Found in most meshes generated through the triangular mesh generators most used in the development of the present work, *Distmesh*, *Gmsh* [38] and *Triangle* [39], this mesh topology results in faces with high warp angle values and higher skewness values. In Fig 2.7(a) one can see a large (for triangular mesh standards) distance between $\mathbf{f}$ and $\mathbf{f_i}$ (skewness) on the edge shared by cells P_1 and P_0 and big values of warp angle (over 20°) on the edges that these cells share with the boundary (again, $\mathbf{S}_f$ is represented in the cell's center for ease of interpretation). The situation cannot be solved using any smoothing algorithm as can be understood by trying to improve the non-orthogonality values mentioned. Any correction applied from cell P_0 perspective will have the exact contrary result in cell P_1, and vice-versa. For the *force based method* introduced in Section 2.1 this means that the forces applied to each cell will cancel and the position of the face dividing the two will remain unchanged.

To solve the predicament several changes in the mesh topology are performed starting with the elimination of the edge shared by both cells in discussion. This alone would be enough to improve both



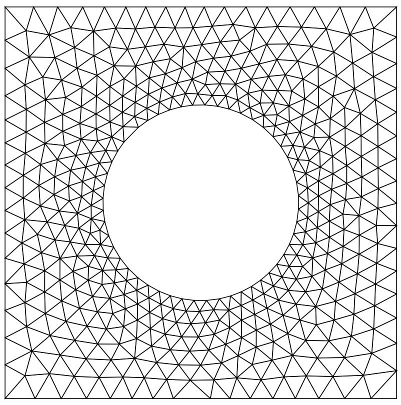
(a) Warp angle and skewness in 'right-angled triangles'.



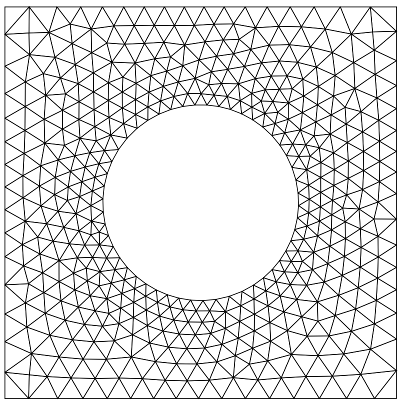
(b) Mesh unrecuperable by the smoothing algorithm.

Figure 2.7: Quality criteria in 'right-angled triangles' and example of mesh in need of constrained Delaunay triangulation.

the non-orthogonality and skewness values however would result in a very large cell. It was found that if only a few instances occurred the smoothing algorithm introduced would be able to reshape these large triangles making them indistinguishable from the rest, situation depicted in Fig.2.8 where the relevant difference is in the bottom boundary.



(a) Mesh after edge elimination step.



(b) Same mesh as on the left, after smoothing.

Figure 2.8: 'Right-angled triangles' solved only by edge elimination.

On the other hand in cases like the one shown in Fig. 2.7(b) where several instances occur it is not possible for the smoothing algorithm to correct these cells. The solution found for these cases was to do a constrained Delaunay triangulation where the constraints are the new edges created inside the aforementioned large triangles. This is done by adding the midpoint of the two boundary edges that have since been merged (after the edge elimination) to the list of points and removing the one vertex that belongs to both edges. The new edges will be defined by the newly added points and the one vertex left that defined the eliminated edge. The process is illustrated in Fig. 2.9, where the dashed gray lines represent the initial triangular mesh and the black lines the added constraints.

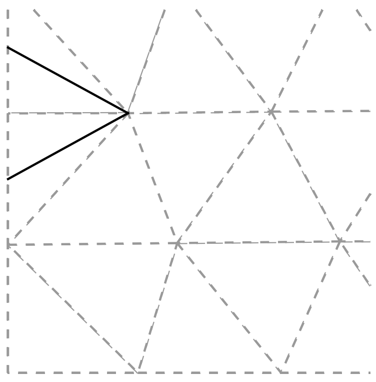


Figure 2.9: Constraint applied to avoid ’right-angled triangles’.

On the same subject of low quality cells it was noted that when ”corner” cells are shaped like the ones in the bottom left of Figure 2.8(a) the quality metrics are improved significantly when compared to the ”corners” of Fig. 2.8(b) as the later have high warp angles in the boundary defining faces (Fig. 2.10). For the sake of clarity ”corner” is defined as the point where two different boundaries intersect.

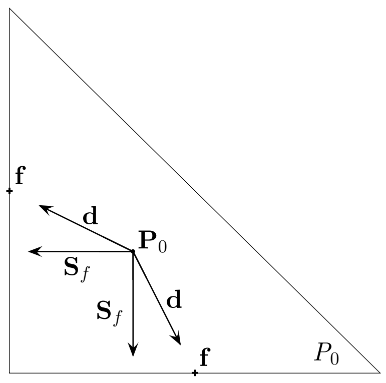


Figure 2.10: Warp angle for current corner topology.

In *Formiga* all triangular cell ”corners” are shaped like the mentioned corner of Fig. 2.8(a), the only exception is when these triangular meshes are the basis for a dual polyhedral mesh. In this latter case the corners are forced to be like in Fig. 2.8(b). The reasoning behind this is that the polyhedral mesh obtained by computing the dual of meshes like the one in Fig. 2.8(b) have ”square” cells in it’s corners while the other topology results in pentagons. This is relevant because these pentagons are, usually, anything but regular having one significantly smaller edge than the rest. Being smaller the chances of having significant skewness values are higher. These changes in corner topology are enforced through the use of simple edge swaps.

2.5 Smoothing Algorithm

The basis of the smoothing algorithm have already been laid in Section 2.1 although important factors concerning the execution were left out. It has been established in the mentioned Section that this is an iterative method where in each iteration the mesh quality is evaluated and corrective forces calculated and applied to the entire mesh, making this a ”global” [28] method despite being possible to adapt and perform ”local” improvements, if required.

The major concern with a method of this sort is to guarantee that the resulting final grid respects the domain set by the initial grid. In *Distmesh* [29] only ”repulsive” forces exist which is to say that the boundary may only be compromised by points leaving the domain. So the solution proposed by it’s authors, and already mentioned in Section 2.3, was to project them back to the boundary if found outside, by applying a force equivalent to the numerical gradient of the output of the *distance function*. This allows for the mesh to comply with the established limits and for the nodes in discussion to move along the boundary. Points defined as `pfix` aren’t given this luxury and are simply fixed.

The corrections applied in this work allow for, to use the *Distmesh*’s terminology, ”attractive” and ”repulsive” forces meaning that the boundary defining points may now leave the domain or enter it. To address the situation one can adapt the method used in *Distmesh* and project these points back to the boundary to which they belong, changing only the condition to identify them from `dist>0` to `abs(dist) < tolerance`, where `dist` is the output of function `fd`. This is the method employed when improving triangular meshes in the *Formiga* code.

To improve polyhedral grids another approach to ensure that the boundary keeps it’s integrity was used. It may only be applied to the polyhedral meshes as it relies on the primal mesh generated and the distance function `fd`. In this instance the primal mesh is used to define a 1D space where the polyhedral mesh’s boundary points are allowed to move. That said space is defined by the computed boundary, identified when verifying the primal meshes validity in Section 2.3.

The dual mesh’s vertices location is evaluated and those belonging to the primal grid’s boundary edges get their movement restricted to the direction of the edges. In other words, if a point belongs to a boundary edge the sum of forces to which he would be subjected are projected onto the edge to where it belongs. If one of the dual mesh’s nodes belongs to two non collinear boundary edges it is fixed.

In a way one could say that this approach is more preventive than the former since it preserves the boundary by not allowing points to leave it in the first place. While the approach used in triangular meshes let’s them leave and gets them back. Both are found to work properly and if required may even be used together although this was not found to be necessary.

Other discussion worth having is the selection of forces to be applied. There are presently four different corrective forces to choose from in the *Formiga* code: the one introduced in *Distmesh* and three introduced in Section 2.1. From these four corrective measures it is possible to create several combinations depending on the problem at hand. In here resides one of the major advantages of this smoothing algorithm, it’s flexibility. By creating a function in the form of, for example, $\mathbf{F} = a\,\mathbf{F_{ort}} + b\,\mathbf{F_{skew}} + c\,\mathbf{F_{unif}} + d\,\mathbf{F_{dist}}$, one may easily change the results obtained.

Coupled with the possibility of defining a specific force combination for a given problem is also the alternative to choose a stopping criterion for the iterative process. For the current work stopping criteria based on grid quality are used. The approach selected was to monitor both the average and maximum values of the metrics more relevant for the problem at hand and stop the iterative process when the improvements are smaller than a defined tolerance.

Both the choice of the force combination and the stopping criteria carry enormous impact in the success of the algorithm as results can vary from solid improvement to an unusable grid. Special attention must be paid to the fact that the grid quality parameters don't use all the same scale therefore coefficients a, b, c, d must reflect these variations and eventually the stopping criterion case more than one grid quality metric is used. For these situations where two metrics with different scales are used a way of giving them similar importance is by multiplying them [40].

2.6 Polyhedral Mesh Generation

The generation of polyhedral meshes by the way of computing the dual of a primal mesh is not new and there are several works in literature that describe the process [15, 19]. In this Section a brief explanation of the algorithm is presented, starting with the 2D case, as this is the main focus of this thesis, and discussing the 3D case at the end.

2.6.1 Polyhedral meshing in two-dimensions

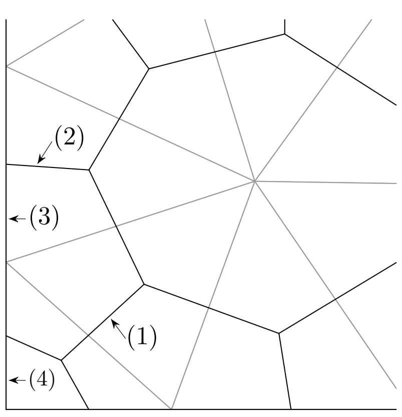
The process starts by gathering every primal mesh's cell central point. For the 2D case the central points will be each triangle's centroid, in a perfect world one would use the circumcenters however it's not uncommon for boundary triangles to have their circumcenters outside the domain resulting in an invalid dual mesh. When discussing the 3D case this issue will be readdressed as the implications of this decision are more severe.

In addition, the mid point of every boundary defining edge is collected as well as the points where these edges intersect. All these points mentioned over the last two paragraphs constitute the dual mesh vertices.

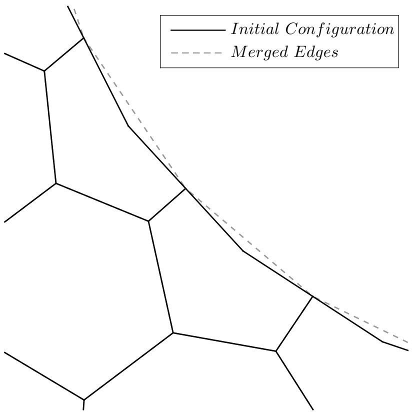
In the interior of the domain, the edges of the dual mesh are created by joining the centroids of neighbouring triangles. Intersecting the primal face shared amongst them. In addition, edges are also added by connecting the central point of boundary triangles to the mid point of their boundary edge. At this point all internal edges are constructed, the external edges are created by connecting the mid point of adjacent boundary primal edges and by connecting ”corner” vertex with the mid point of the primal edges that define it. Examples of all four types of edge described are on Fig. 2.11(a) numbered by order of appearance in the text.

The polyhedra are now all but formed being just a matter of defining with a cell number the group of dual edges that surround each primal vertex.

For interior primal vertex a dual cell is created, composed by the dual edges (type (1)) that intersect



(a) Examples of types of dual edge.



(b) Correction of concave cells through edge merging.

Figure 2.11: Types of dual edge and concave cells formed by almost collinear faces.

the primal edges containing a specific dual vertex.

For primal vertices on the domain's edges, in addition to the edges gathered by following the same method as before (where one gets edges of types (1) and (2)) one also needs to add a boundary edge (type (3)). This may be achieved by finding the dual edge that contains a given primal vertex.

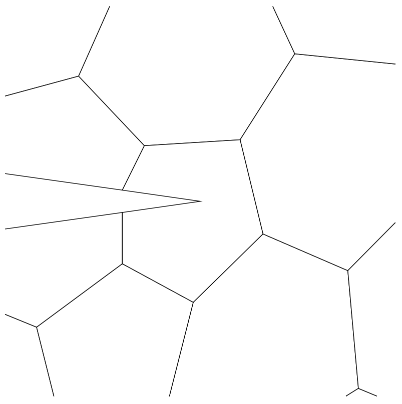
For primal vertexes that coincide with dual vertexes, meaning "corner" vertices, a cell is created by gathering the type (2) dual edges that surround it as well as the dual edges (type (4)) that contain it.

With these steps completed the result should be, for most cases, an all convex polyhedral mesh that respects the defined geometry, provided the primal mesh respected the domain as well. The reason why this isn't true for all cases is that the dual mesh generation method is known to create invalid concave cells where the discrete boundary of the domain is itself concave. In Fig. 2.11(b) a detail of a mesh created around a circle proves the existence of said cells though cases like these aren't a particular cause for concern as the two edges that render the polygon concave may be merged into a single one with no real negative consequences.

Concerning this issue, a more complicated problem is shown in Figure 2.12 where two instances of concave cells are presented. In these situations one must resort to other methods in order to obtain only convex polygons. In [41] three ways to "eliminate" these concave cells are presented for 3D meshes. Despite being possible to convert any of them to be applied in 2D it was decided to use a new approach in *Formiga*.

This novel approach was designed to work together with the already introduced smoothing algorithm meaning that the alterations described over the following lines won't render an high quality mesh until the smoothing algorithm is applied.

The first order of business, once a concave cell is identified, is to ensure the cell in question is correctly defined. It's not uncommon in severe cases of concavity for the cell's centroid to be found outside of the cell. This may result in an incorrect ordering of the vertexes that make the cell, which causes situations like the ones in Figure 2.13(a) that was supposed to represent the same cell as in Fig.


(a) Mesh around the trailing edge of a NACA 0012 airfoil.

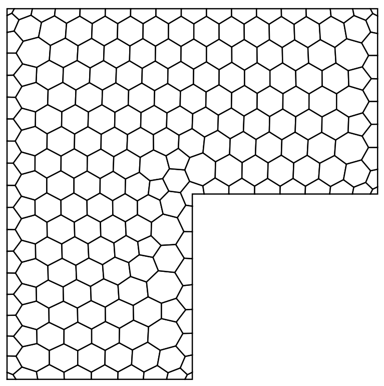
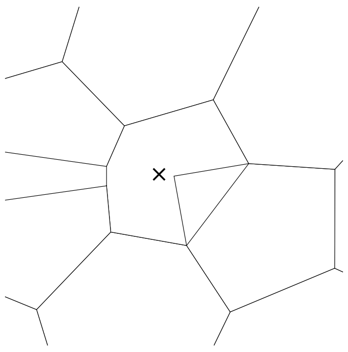

(b) Mesh over a L-shaped domain.

Figure 2.12: Examples of concave cells with non trivial solution.

2.12(a) though it fails do to so due to incorrect ordering of the vertices. The 'x' represents the centroid estimation through the average of the vertex position which is outside of the cell and of the domain, if the cell were properly ordered. To ensure that the cell respects the defined domain the following steps are taken:

1. The concave point is used as reference around which all the other points of the cell are ordered counter-clockwise so that they form a convex polygon;
2. Decompose the resulting polygon in triangles using the concave point as pivot, 2.13(b);
3. Evaluate the position of the centroid of each triangle using `fd`, the one found outside the domain represents the model's concavity. Knowing this it's trivial to reorder the points in the correct way.


(a) Incorrect point's ordering due to concavity.

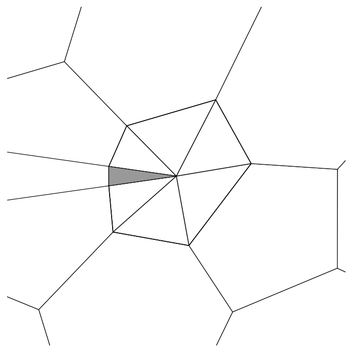

(b) Decomposition in triangles of the cell to the left.

Figure 2.13: Same concave cell as in Fig. 2.12(a) before reordering and after triangle decomposition.

Provided that the previous steps are completed, the mesh should always respect the domain's limits which will make possible the application of the following algorithm:

4. The vector that connects the centroid of the triangle found outside the domain to the concave point is computed;

5. Compute the midpoints of the edges of that cell and search for the one whose vector formed by itself and the concave point is closer to making an angle of 180º with the one computed in the previous step;

6. Once found, the edge to which this midpoint belongs is deleted and its two vertices are connected to the concave point, effectively creating two new edges;

7. Delete the ”concave cell” from the cell list and add two new ones that will split amongst each other the vertices. In Fig. 2.14 this method is applied on a mesh generated around the same airfoil as Fig. 2.12(a) , and on the same L-shaped domain as in Fig. 2.12(b).

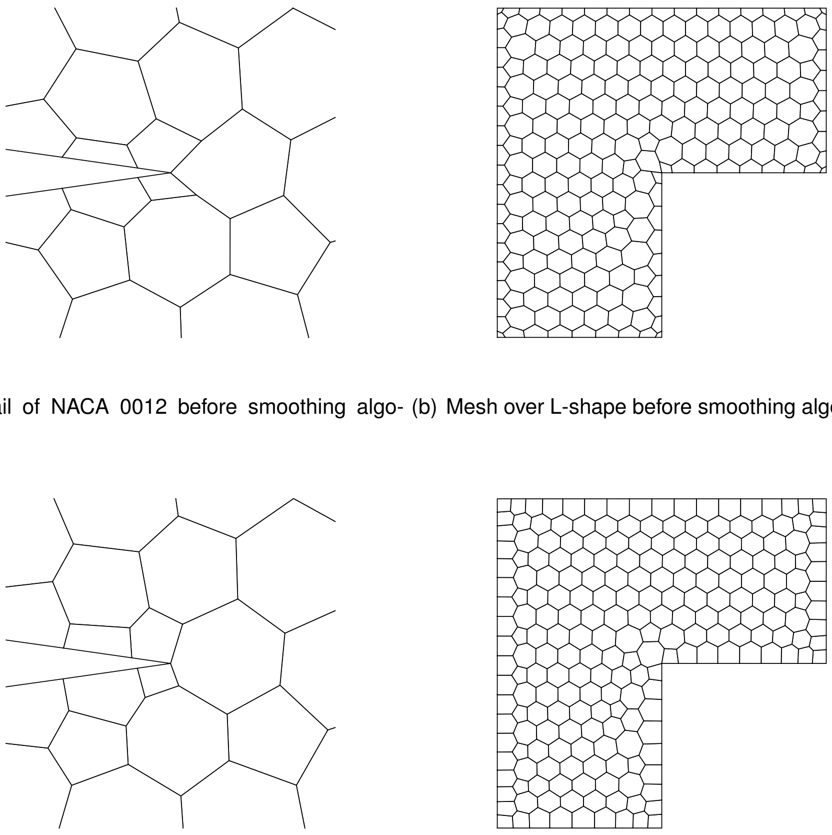
Effectively the proposed method distributes the vertices that composed the previous concave cell over three different cells, two of them created for the effect. In Fig. 2.14 the end result of the current algorithm is presented. As mentioned before the outcome of the present method by itself is nothing to write home about with highly distorted cells and poor mesh quality values. However when coupled with the smoothing algorithm the result is a high quality convex polyhedral mesh. Numeric values of grid quality for the airfoil are found in Chapter 3, while Table 2.1 has the quality metrics regarding the L-shaped domain. Important to note that the values in Chapter 3 refer to the entire mesh while in Table 2.1 the values presented concern only to edges affected by this method, being therefore directly aimed at comparing methodologies to resolve this concavity issue.

Through the development of *Formiga* many approaches to this particular problem were tried, one of them resulted in something very close to the mesh generated in STAR CCM+®, presented in Fig. 2.15(a), and was important in the development of the already explained algorithm. Though it’s not possible to comment on the method followed by STAR CCM+®, the approach previously used had the same first 5 steps as the already described method however instead of eliminating the edge found a new one is added composed by the concave point and the selected edge’s midpoint.

Despite performing well under most circumstances it was found that in situations such as the one in Figure 2.12(a) the edge introduced to split the concave cell was, a lot of times, very small. This solves the concavity problem but causes another since small edges tend to result in high skewness values. These values often exceed $\psi = 1$. Sometimes it is possible to correct this situation by applying a Smoothing method however it was found that with the force based method employed here one of two things would usually happen, either the skewness value of the mentioned edge would improve at the expense of the warp angle of neighbouring edges or the warp angle of said neighbouring edges would be corrected leading to an even smaller dividing edge and an even higher skewness value.

The reason why this failed approach is regarded as an important step towards the development of the current algorithm is that the later would not have been found if not by using the first coupled with the yet un-introduced edge elimination feature that will be explained in Section 2.7.

Examining Fig. 2.15(a) the dividing edge is not small when compared to the ones around and scores pretty respectable values of both skewness ($\psi \simeq 0.5$) and warp angle ($\alpha_N \simeq 0^\circ$) . The overall result of it’s introduction, on the other hand, is not great resulting in high warp angle in the boundary edges and high skewness values in other neighbouring ones.

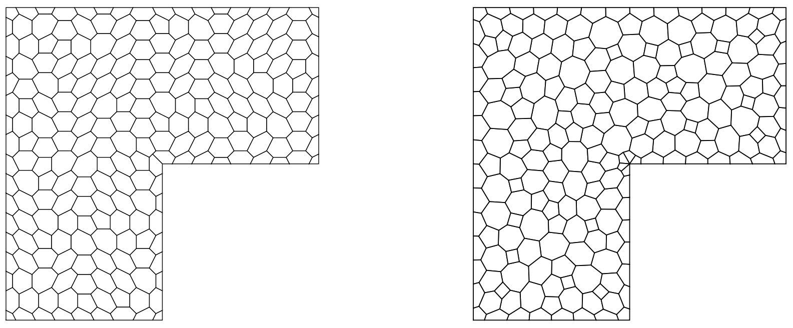


(a) Detail of NACA 0012 before smoothing algorithm. (b) Mesh over L-shape before smoothing algorithm.

(c) Detail of NACA 0012 after smoothing algorithm. (d) Mesh over L-shape after smoothing algorithm.

Figure 2.14: Results of introduced algorithm to correct the concave cells before and after application of the smoothing algorithm.

Another approach found is the one used by the open source CFD code OpenFOAM, that for the same problem decomposes the formerly concave cell in several smaller quadrilaterals, Fig.2.15(b). The variation in cell area introduced by the method has consequences in terms of uniformity quality but



(a) Mesh over the L-shape created with STAR CCM+®. (b) Mesh over the L-shape created with OpenFOAM.

Figure 2.15: Meshes created with other mesh generators to compare methods of removing concave cells.

achieves better skewness values than STAR CCM+® and similar warp angle values.

In Table 2.1 values from the meshes generated through *Formiga*, OpenFOAM and STAR CCM+® are presented ordered from worst to best performing.

		$\alpha_N [deg]$		ψ	
	$N.Cells$	Avg	Max	Avg	Max
STAR CCM+®	226	14.0238	21.2904	0.4853	0.8203
OpenFOAM	209	9.2025	21.1908	0.213	0.368
Formiga	219	1.2285	3.0095	0.077	0.1843

Table 2.1: Comparison of quality metrics from the faces affected by the concave cell removal.

2.6.2 Polyhedral meshing in three-dimensions

For the 2D version of *Formiga* the cell's central vertices are the primal cells centroids as these are easy to obtain and always produce valid meshes. In three dimensions, it is still possible to use them however the resulting polyhedral mesh will have non-planar faces which, have poor computational performance [19] and will undoubtedly make the evaluation of quality metrics harder. Just like in the two dimensional case, the primal meshes circumcenter's is not always a valid option, for the very same reasons. This led to the appearance of other ideas to define these central points. Garimella *et al.* [15] suggest using the circumcenters whenever possible and the closest point to the circumcenter that still lies within the cell and falls on the line between the circumcenter and the cell's centroid. In [19] several remarks regarding the quality of this approach are made and improvements suggested.

As a first approach, and fully aware of the consequences, *Formiga* won't follow these suggestions and will use the circumcenters as central point every time they fall within it's cell and the geometric center otherwise. To compute the centroids each cell is decomposed in pyramids.

Just like previously, the dual mesh is computed by defining all entities starting with vertexes, edges, faces and finally the cells. Since the main focus of this work is to generate 2D meshes there won't be a detailed description of the 3D generation algorithm, however the main differences and particularities of 3D mesh generation will be pointed out.

One of the aspects of two dimensional mesh generation that may be salvaged is the definition of dual vertices. All dual vertexes defined in 2D will be needed in 3D and added to the central cell points already discussed. However some renaming is in order as what were cells in 2D are now faces, faces and edges had the same meaning in 2D but now are two different entities. In short the centroid of each boundary primal face is a dual vertex, every mid point of the edges that limit the domain is a dual vertex and every "corner" vertex is also a dual vertex. To clear any doubts concerning the definition of cells, faces, and vertexes in 3D, Figure 2.16 shows a cell P_0 compose by several faces, where face f is marked in gray and is composed by vertexes v_1, v_2, v_3, v_4, v_5

Edges are created in the same way as before, it's just important to be cautious when defining exterior

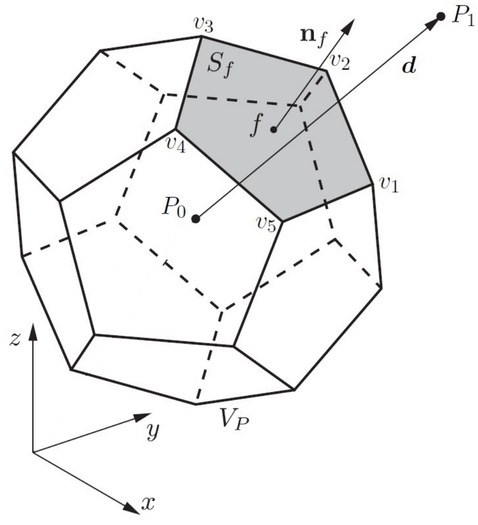


Figure 2.16: 3D polyhedral cell and relevant entities - from [32].

dual edges to avoid connecting vertexes that belong to different boundaries. Face construction is different for interior faces. Exterior faces follow the same procedure as cells in 2D, by ordering the dual edges around a primal vertex. Special attention must be paid to primal vertexes on model's boundary edges and "corners" as the dual vertices that surround these belong to several boundaries therefore defining distinct dual faces. Interior dual faces are constructed by gathering dual edges that surround a primal edge.

To assemble the cells it's just a matter of collecting all the dual faces of each primal vertex.

For it's current purpose there is no need for the algorithm to be a complete mesh generator. So two compromises were made that restrict it's capabilities, the first was to not add "corner" vertexes as dual points. What this effectively means is that *Formiga* only handles geometries in which a cell may belong to up to two boundaries. The second is that no algorithm to resolve concave cells caused by a concave discrete boundary was implemented. This is not to be confused with concave cells created due to the choice of central cell points that shall be addressed in Section 2.8.

2.7 Changes in Polyhedral Meshes Connectivities

Similarly to what was said in Section 2.4 for triangular meshes, there may occur situations in the polyhedral ones that are left unresolved by the proposed smoothing algorithm. In general cells of arbitrary number of faces are more "malleable" than triangular cells so there isn't a particular topology that must be avoided in order to the smoothing algorithm to work. On the other hand its application may cause small edges/faces. As mentioned previously from the perspective of grid quality, smaller edges have higher skewness values and should therefore be avoided.

The criterion defined to identify small edges is to compute the average length of a cell's edges and compare it against the length of each individual one. Case an individual edge is found to be smaller than a pre-established percentage of the average, the edge number is stored for it to be later eliminated.

The edge elimination process is in theory very simple, as one must only collapse a small edge into

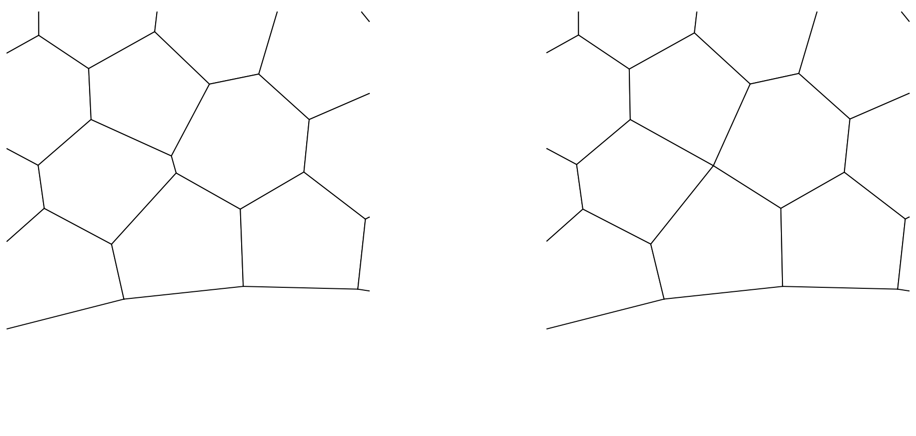

(a) Initial mesh. (b) Mesh after edge elimination.

Figure 2.17: Detail of a mesh before and after the edge elimination process.

a single point keeping track of the changes caused in nearby cells. This approach of collapsing edges into points is also followed in the *Polymesher* code [13]. In Figure 2.17 there's an example of an edge elimination process.

Another unwanted scenario is to have a small cell. Despite being a very rare occurrence the ability to eliminate cells was added to *Formiga*. To eliminate a cell, each cell's area is evaluated and compared to its "neighbours". If the area is smaller than a fraction of the neighbouring cells area average then the cell is eliminated. Just like for the edge case, the eliminated cell is collapsed to a single point, in situations where the cell to be eliminated shares some edges with the boundary, this point must guarantee that the user's domain input is respected. Otherwise the cell is collapsed to it's centroid. A similar approach is also used for the edge elimination case. If an edge shares one vertex with the boundary or contains a fixed vertex it will collapse to that specific point. In any other case the edge will collapse to it's midpoint, instead.

In order to avoid unnecessary changes and to improve the chances of having a positive impact in the quality metrics the modifications are kept to a minimum. The process of edge and cell elimination only starts when both edges and cells to be deleted are found. Given that there are edges and cells to delete, the edges are eliminated first as these produce less impact in the mesh. In *Formiga* a cell is only eliminated if there are no edges to delete.

2.8 Algorithm structure

With all the major components needed for both mesh generation and improvement, introduced, matters to show where they fit depending on the type of mesh as well as introducing smaller pieces that also make the code.

2.8.1 Two dimensional

The 2D version of *Formiga* has all the algorithms presented in the previous Sections. In Figures 2.18(c) and 2.18(a) the polyhedral and triangular versions of *Formiga* are represented through their essential blocks, the name of each block in said figures refers to the methods described in Sections with similar name. Off all blocks the one that still lacks explanation is the 'Post-processing' one.

Once in the 'Post-processing' stage the mesh is completed and only a few operations need to be carried out to complete the execution. The Reverse Cuthill-Mckee algorithm [42] is applied in order to reduce the bandwidth of the matrix containing the cell's connectivities. The algorithm is applied through MATLAB's function `symrcm`. In addition the generated mesh is exported to the STARCD/PROSTAR (v4) format to be tested in SOL. Other non-essential tasks may be carried out in this stage such as the graphical representation of the mesh and saving the grid quality values into an Excel® file.

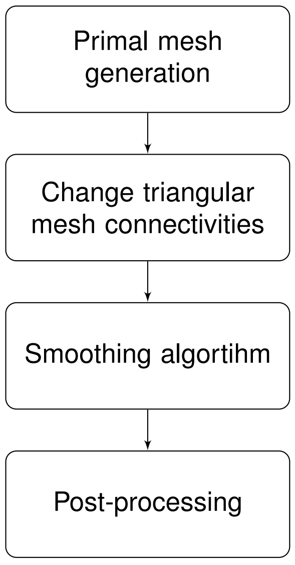
From the mentioned Figures one can see that the approach to topology modification differs for triangular and polyhedral meshes. Triangular meshes undergo these modifications before the application of the smoothing algorithm while polyhedral meshes undergo after. The explanation has been alluded to in Sections 2.4 and 2.7 though, in short, for triangular meshes there are topologies that prevent the smoothing algorithm from being successful and as a consequence they need to be dealt with beforehand while in polyhedral meshes this doesn't happen and changes in the connectivities are used afterwards only to enhance the results obtained with the smoothing algorithm.

Another worthwhile remark is that in *Formiga* the triangular mesh always goes through the smoothing process independently of it's final use. This is done to account for situations like the one in Figure 2.8 however, apart from cases like these, the improvement of the primal mesh's quality metrics through the smoothing algorithm was not found to be of critical significance for the generation of polyhedral meshes as the changes in vertices position have a small impact in the dual vertex used.

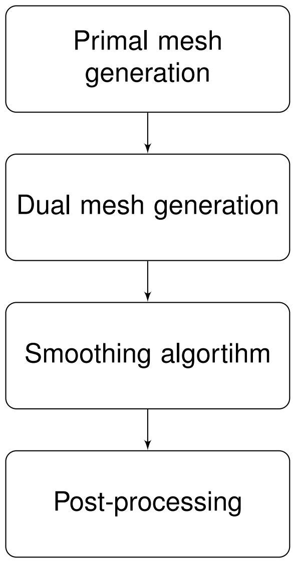
2.8.2 Three dimensional

As a proof of concept a 3D version of *Formiga* was created. This code uses the `distmeshn` function as primal mesh generator when possible though for one of the test cases in Section 3.6, *Gmsh* was used since the first code was unable to create a valid mesh misrepresenting the interior circle. The dual mesh is computed using the methods in Section 2.6.2 after which the smoothing algorithm is applied. Tetrahedral and polyhedral meshes do not suffer any changes in topology. As mentioned before without changes in the primal meshes connectivities there is no need to apply the smoothing algorithm in order to improve the dual mesh therefore it is only applied to the polyhedral mesh. The flowchart for tetrahedral mesh generation is not presented as this is the simplest of them all, constituting of only the mesh generation, application of the implemented smoothing algorithm and post-processing.

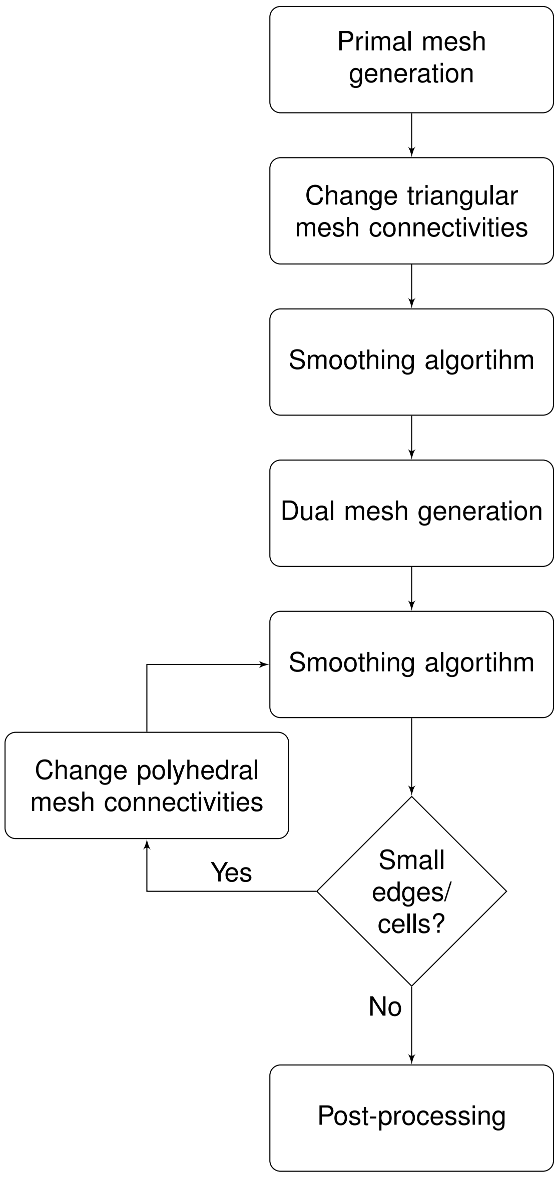
Some modifications were made to the smoothing algorithm in order to keep the it simpler. Namely, the method used to keep the vertexes inside the domain is the same used for 2D triangular meshes, by projecting the vertices that left the domain. Other important difference is the correction of the non-orthogonality angle.



(a) 2D triangular flowchart.



(b) 3D polyhedral flowchart.



(c) 2D polyhedral flowchart.

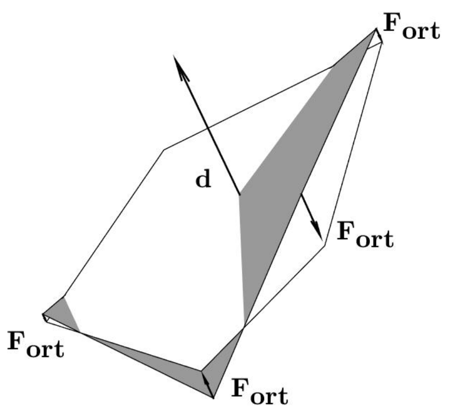
Figure 2.18: Flowcharts for triangular mesh generation in 2D and the two versions for polyhedral meshes in 2D and in 3D.

There are details in the implementation of the warp angle correction in 2D, that make it difficult to adapt for three dimensions. Therefore a new correction is used in addition to the warp angle one that makes an effort to mitigate the occurrence of non-planar faces caused by the dual meshing step.

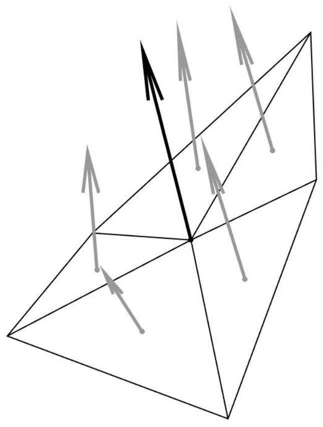
This new approach computes for each internal face the plane whose normal is given by the corresponding vector $\mathbf{d}$ and passes through the face centroid $\mathbf{f}$. This plane is where all face's vertexes should lie in order to obtain a 0° warp angle and a planar face. As forcing all face's vertices to lie in this plane would be too drastic of a change, the vector between each one of the face's vertex and its projected position is computed and it's length measured. With this information it's possible to create a new corrective force in which, for every iteration, each vertex moves a fraction of the length of the computed vectors. In Fig. 2.19(a) an example of the application of these forces is depicted. The face marked in gray represents a non-planar, non-orthogonal face and the white marked one the target planar face.

Figure 2.19(b) shows the same gray face of Fig. 2.19(a) decomposed in triangles and each one has its normal vector marked in gray, additionally their average is represented in black at the centre of the face. The intent behind this picture is to present a quality metric used to quantify how non-planar a face is and if the forces applied managed to improve it. The metric γ simply measures the angle between each triangles normal vector and the average shown in black. The angles are afterwards averaged. Just like the non-orthogonality metric these angles are measured in degrees.

The other corrections proposed for 2D (skewness and uniformity) were easily adapted for 3D and work in the same way, while the "Post-processing" stage in three dimensions consists of only exporting the mesh to the STARCD/PROSTAR (v4) file format.



(a) Non-orthogonality angle correction for 3D faces.



(b) Non-planar face decomposed in triangles and "concavity" correction.

Figure 2.19: Non-orthogonality angle and "concavity" correction in non-planar 3D face.

Chapter 3

Grid Quality Improvements and Comparison with other Mesh Generators

In this chapter, the grid quality values obtained through the described algorithm are presented. In order to make these numbers meaningful, comparisons with other mesh generators are also included. In summary, Sections 3.1 through 3.3 are concerned with the triangular meshes results, Sections 3.4 and 3.5 with polyhedral meshes results and Section 3.6 with 3D polyhedral mesh results.

3.1 Distmesh comparison

The triangular version of *Formiga* rose from the need to improve *Distmesh's* grids so it stands to reason that this should be the first comparison. As mentioned previously, one of the drawbacks of *Distmesh* is that no two meshes are equal when using the same non-uniform size distribution, since parts of the algorithm depend on random factors.

So in order to obtain meaningful results in this and the following sections, the chosen methodology goes as follows: for each entry parameters set by the user the *Formiga* code is ran n times generating n different meshes. From the data collected the sample's average $(\overline{x})$ and standard deviation (s) are calculated. Having these, a level of significance α is set so that conclusions may be reached about the overall algorithm's efficiency. The data collected refers to both the average and maximum value of each grid quality metric.

Since the distribution of the population from which the initial sample was extracted is unknown, a sample size of $m = 40$ was set for every example. The central limit theorem guarantees that for a sample size large enough the sample's average $\overline{x}$ distribution tends for the standard normal distribution. Though concepts like "large enough" can leave room for debate it's felt that the sample size chosen meets the requirement for the standard normal distribution to be used. From the set level of significance

α and by defining a margin of error $E = \alpha \times \overline{x}$, the number of simulations n required for each grid quality metric to be computed with $(1-\alpha)\times 100\%$ certainty is given by equation 3.1, in which $z_{\alpha/2}$ is the quantile of order $\alpha/2$ of the standard normal distribution and s the sample's standard deviation:

$$n = \left(\frac{z_{\alpha/2} \times s}{E}\right)^2 \tag{3.1}$$

In this work a level of significance $\alpha = 0.1$ was set which corresponds to a confidence level of 90%.

The example in Table 3.1 doesn't require this analysis because an uniform point distribution was used meaning that *Distmesh* always comes up with the same mesh (same points and same connectivities) when using the same user input. The Rose Shaped Domain name and equation was taken from [43].

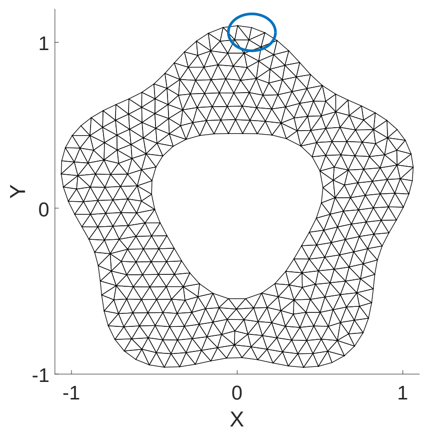
			$\alpha_N[deg]$		ψ	
	h_{ref}	$N.Cells$	Avg	Max	Avg	Max
Rose Shaped Domain						
Distmesh		662	4.9226	26.9435	0.0459	0.2743
Formiga	0.084	655	1.5327	18.4545	0.0405	0.1934
Improvement			68.86%	31.51%	11.72%	29.51%
Distmesh		2486	3.6364	30.0031	0.0253	0.2846
Formiga	0.045	2484	1.7116	14.4106	0.0220	0.1937
Improvement			52.93%	51.97%	12.96%	31.94%
Distmesh		13169	2.2711	14.6921	0.0114	0.2465
Formiga	0.02	13169	1.2960	14.1533	0.0098	0.1974
Improvement			42.94%	3.67%	14.47%	19.93%
Distmesh		40513	1.3022	32.8582	0.0060	0.2588
Formiga	0.0115	40511	0.7085	14.2740	0.0052	0.2018
Improvement			45.59%	56.56%	14.65%	22.05%
Average Improvement			52.58%	35.93%	13.45%	25.86%

Table 3.1: Quality metric's improvement on meshes over a Rose Shaped domain.

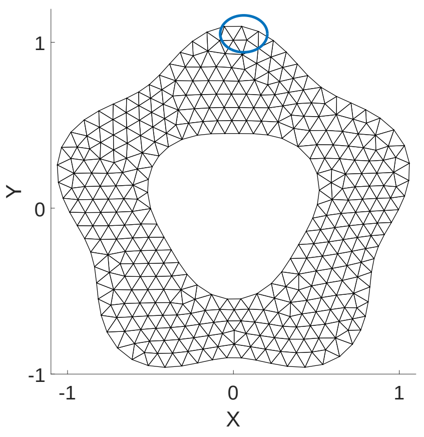
For the three examples introduced in Table 3.2 the meshes were generated with the following *distance function* (`fd`) and *desired edge length function* (`fh`) for Square with circular hole, Square with size function based in a point and a line source and NACA 0012 airfoil, respectively:

```
fd=@(p) ddiff(drectangle(p,0,1,0,1),dcircle(p,0.5,0.5,0.25));
fh=@(p) min(href+0.1*dcircle(p,0.5,0.5,0.25),hmax);
fd=@(p) drectangle(p,0,1,0,1);
fh=@(p) min(min(href+0.1*abs(dcircle(p,0,0,0)),href+0.1*abs(dpoly(p,[0.3,0.7; 0.7,0.5]
))),hmax);
fd=@(p) ddiff(dcircle(p,0.5,0,1),(abs(p(:,2))-polyval([a(5:-1:2),0],p(:,1))).^2-a(1)^2*
p(:,1));
fh=@(p) min(min(hlead+0.2*dcircle(p,0,0,0),htrail+0.3*dcircle(p,1,0,0)),hmax).
```

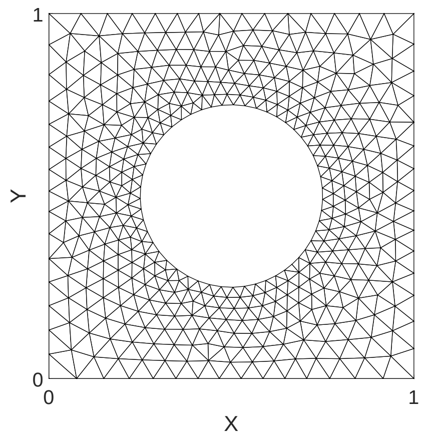
All these size functions have the same meaning, that the desired cell face size varies linearly across the domain. The only difference between the three is the locations where the mesh is more refined. As a result of these functions, Table 3.2 shows both the face length desired h_{ref} and the maximum face length allowed h_{max}, both defined by the user. For the last example `hlead` $= h_{ref}$ and `htrail` $= [0.05, 0.02, 0.01, 0.008]$, for each mesh number and `a` is a vector that helps define the airfoil shape. All three are used as examples in the freely available version of *Distmesh* though with some minor differences. Namely changing the position of the square with circular hole, centering the square and interior circle in $(0.5, 0.5)$ and changing their dimensions. The square side is now 1 and the circle radius is $r = 0.25$. The slope of the size functions used in all three examples is also different from the original.



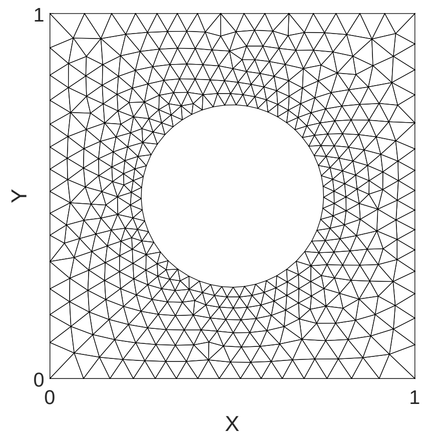
(a) Rose Shaped Domain - *Distmesh*'s Mesh.



(b) Rose Shaped Domain - *Formiga*'s Mesh.



(c) Square with circular hole - *Distmesh*'s Mesh.

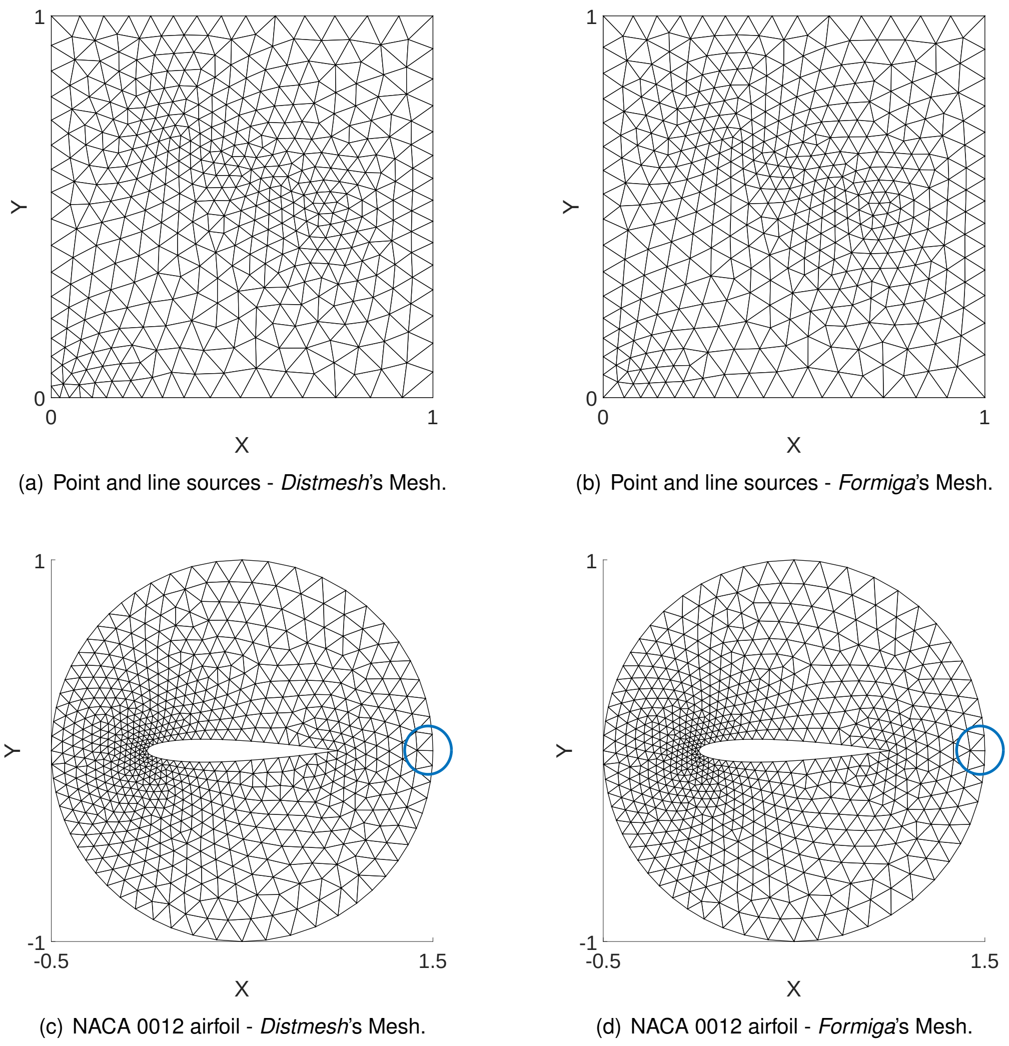


(d) Square with circular hole - *Formiga*'s Mesh.

Figure 3.1: Examples of meshes for the first two examples - Rose Shaped domain and Square with circular hole, obtained with *Distmesh* and *Formiga* codes.

From the values in the mentioned Table one can conclude that there's no need to repeat the generation process any further for all examples and reference sizes tested because the results show that the sample size collected is enough to compute the average value μ of all quality criteria with a confidence level of 90%. Despite *Formiga* not having any random variable of its own, it uses *Distmesh*'s grids as input and so its results suffer from the same flaw. No need for an equivalent Table for *Distmesh* as it's sample size required will never be higher than the shown here as can be understood considering that different mesh topologies have different levels of susceptibility to improvement therefore resulting in a larger amplitude of grid quality values.

As for the actual differences between generators Tables 3.1 and 3.3 show very significant improvements in both quality metrics across all examples. The reduction in warp angle, especially it's average value, is particularly large while the reduction in skewness values is more modest. This behaviour may be explained by two factors, the first being the force combination chosen. All results presented in this Chapter and the next are dependent on the relative importance given to each corrective measure introduced in Section 2.1. For this chapter the goal was to improve the overall grid quality not focusing in a particular metric however, if better skewness values are pretended, a more skewness focused force could be applied in order to further improve the skewness values though with possible sacrifice of the previous gains in the warp angle.



(a) Point and line sources - *Distmesh*'s Mesh. (b) Point and line sources - *Formiga*'s Mesh.

(c) NACA 0012 airfoil - *Distmesh*'s Mesh. (d) NACA 0012 airfoil - *Formiga*'s Mesh.

Figure 3.2: Examples of Meshes for the first two examples - Square with size function based in a point and a line source and NACA 0012 airfoil. Comparison between *Distmesh* and *Formiga* codes.

			$\alpha_N[deg]$		ψ	
	$Mesh\,\#$	$h_{ref} - h_{max}$	Avg	Max	Avg	Max
$Square\ w/\ circular\ hole$						
$Average$			1.6903	18.1347	0.0431	0.2164
$Standard\,Deviation$	1	$0.03 - 0.1$	0.2342	0.9012	0.0033	0.0074
$Sample\,Size\,Required$			6	1	2	1
$Average$			1.5354	18.3874	0.0410	0.2198
$Standard\,Deviation$	2	$0.02 - 0.1$	0.2210	0.5884	0.0021	0.0054
$Sample\,Size\,Required$			6	1	1	1
$Average$			1.5180	18.9274	0.0422	0.2199
$Standard\,Deviation$	3	$0.01 - 0.06$	0.1245	0.7810	0.0015	0.0041
$Sample\,Size\,Required$			2	1	1	1
$Average$			1.5560	18.6773	0.0424	0.2249
$Standard\,Deviation$	4	$0.006 - 0.04$	0.0865	0.9709	0.0012	0.0054
$Sample\,Size\,Required$			1	1	1	1
$Point\ and\ line\ sources$						
$Average$			1.4367	18.2935	0.0431	0.2146
$Standard\,Deviation$	1	$0.03 - 0.08$	0.1364	1.1332	0.0032	0.0074
$Sample\,Size\,Required$			3	2	2	1
$Average$			1.4850	18.8731	0.0434	0.2172
$Standard\,Deviation$	2	$0.02 - 0.07$	0.1414	1.0749	0.0026	0.0064
$Sample\,Size\,Required$			3	1	1	1
$Average$			1.5628	18.9180	0.0427	0.2193
$Standard\,Deviation$	3	$0.01 - 0.06$	0.1017	0.8374	0.0014	0.0070
$Sample\,Size\,Required$			2	1	1	1
$Average$			1.7566	19.0607	0.0416	0.2236
$Standard\,Deviation$	4	$0.005 - 0.04$	0.1654	1.1435	0.0013	0.0062
$Sample\,Size\,Required$			3	1	1	1
$NACA\ 0012\ airfoil$						
$Average$			2.8755	27.4843	0.0565	0.2894
$Standard\,Deviation$	1	$0.008 - 0.45$	0.5669	6.1041	0.0025	0.0408
$Sample\,Size\,Required$			11	14	1	6
$Average$			3.1040	27.4652	0.0580	0.3056
$Standard\,Deviation$	2	$0.006 - 0.4$	0.5292	7.4761	0.0019	0.0492
$Sample\,Size\,Required$			8	21	1	8
$Average$			3.0311	28.6062	0.0412	0.3066
$Standard\,Deviation$	3	$0.004 - 0.06$	0.4111	6.4800	0.0017	0.0588
$Sample\,Size\,Required$			6	14	1	11
$Average$			3.1324	34.2233	0.0403	0.3378
$Standard\,Deviation$	4	$0.003 - 0.05$	0.2838	5.3605	0.0019	0.0316
$Sample\,Size\,Required$			3	7	1	3

Table 3.2: Sample size required to achieve a 90% confidence level on triangular meshes generated through *Formiga*.

			$\alpha_N[deg]$		ψ	
	$Mesh\,\#$	$N.Cells$	Avg	Max	Avg	Max
Square w/ circular hole						
Distmesh		816	4.7312	31.6658	0.0467	0.3402
Formiga	1	813	1.6903	18.1347	0.0431	0.2164
Improvement			64.27%	42.73%	7.89%	36.39%
Distmesh		1451	4.4825	31.7030	0.0449	0.3492
Formiga	2	1447	1.5354	18.3874	0.0410	0.2198
Improvement			65.75%	42.00%	8.59%	37.07%
Distmesh		3465	4.3209	32.7940	0.0464	0.3468
Formiga	3	3458	1.5180	18.9274	0.0422	0.2199
Improvement			64.87%	42.28%	9.13%	36.59%
Distmesh		6124	4.2502	33.1261	0.0465	0.3418
Formiga	4	6109	1.5560	18.6773	0.0424	0.2249
Improvement			63.39%	43.62%	8.92%	34.20%
Average Improvement			64.57%	42.66%	8.63%	36.06%
Point and line sources						
Distmesh		843	4.5829	32.0672	0.0468	0.3115
Formiga	1	839	1.4367	18.2935	0.0431	0.2146
Improvement			68.65%	42.95%	7.89%	31.12%
Distmesh		1394	4.4162	32.3206	0.0471	0.3120
Formiga	2	1390	1.4850	18.8731	0.0434	0.2172
Improvement			66.37%	41.61%	7.86%	30.39%
Distmesh		3014	4.1969	32.3284	0.0462	0.3160
Formiga	3	3010	1.5628	18.9180	0.0427	0.2193
Improvement			62.76%	41.48%	7.67%	30.60%
Distmesh		5998	4.0609	32.3224	0.0451	0.3218
Formiga	4	5993	1.7566	19.0607	0.0416	0.2236
Improvement			56.74%	41.03%	7.69%	30.50%
Average Improvement			63.63%	41.77%	7.78%	30.65%
NACA 0012 *airfoil*						
Distmesh		901	4.4735	29.7813	0.0585	0.3226
Formiga	1	901	2.8755	27.4843	0.0565	0.2894
Improvement			35.72%	7.71%	3.42%	10.28%
Distmesh		1086	4.5476	30.2784	0.0600	0.3365
Formiga	2	1085	3.1040	27.4652	0.0580	0.3056
Improvement			31.75%	9.29%	3.33%	9.16%
Distmesh		2883	4.3053	32.0502	0.0444	0.3399
Formiga	3	2879	3.0311	28.6062	0.0412	0.3066
Improvement			29.60%	10.75%	7.18%	9.79%
Distmesh		3406	4.2568	36.5612	0.0435	0.3520
Formiga	4	3401	3.1324	34.2233	0.0403	0.3378
Improvement			26.42%	6.39%	7.47%	4.02%
AverageImprovement			30.87%	8.54%	5.35%	8.31%

Table 3.3: Quality improvements on triangular meshes over a Square with circular hole, Square with size function based in a point and a line source and NACA 0012 airfoil.

The second, and perhaps most significant, factor is that the skewness values shown by *Distmesh*, in particular it's maximum values of around 0.3 may not, as will be shown in the following Sections, be considered high not only for triangular meshes but especially when considering polyhedral meshes as well.

Across Figures 3.1 and 3.2 one can clearly identify some of the main characteristics of the proposed algorithm such as the re-triangulation of "right angled" boundary defining triangles (top "petal" of Rose Shaped Domain, Figs. 3.1(a) and 3.1(b)) and the forced connectivities on the "corners" of the domain (bottom right corner of Figs. 3.2(a) and 3.2(b), as an example). In Figure 3.2(d) however there is still one instances of 'right-angled triangles' on the point of coordinates $(1.5, 0)$. This happens due to a known limitation of the current algorithm where if any fixed point belongs to one these triangles they are not corrected. This approach was taken to protect and preserve the boundary that had already been verified as valid. To the best of the author's knowledge the incidence of this limitation should be very small. Though this explains why the improvement percentage of the quality metrics is smaller when compared to the previous examples.

What also stands out from the mentioned Figures is the high quality cells produced by *Distmesh* in the interior of the domain. A testament to this fact is the difficulty in detecting visually differences between meshes, in this region, on the right and left column of these Figures. From this fact one can deduce that almost equilateral triangles that make the interior of the mesh have borderline zero skewness and warp angle. As a consequence, stands that the improvements made by *Formiga*, in these cases and shown in Tables 3.1 and 3.3, were conquered mainly by improving boundary cells, which are cells that have at least one boundary face.

3.2 Triangle comparison

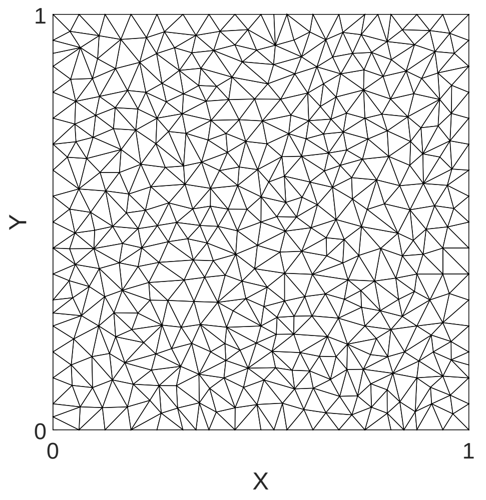
Triangle [39] is a renown Delaunay mesh generator whose latest version dates back to 2005. To generate meshes using this software the following command line switches were used : '-Dqa'. According to the available documentation [44] the letter 'D' ensures all mesh's triangles are Delaunay, letter 'q' guarantees that these triangles have internal angles ranging from $20°$ to $140°$ and letter 'a' imposes a maximum triangle area which was used to adjust the number of cells in the mesh. Once generated, the output files `.ele` and `.node` were read and the meshes evaluated in MATLAB. Equivalent meshes of approximate number of cells were then created through *Formiga*. The chosen example for this comparison was a square size 1 and uniform size distribution.

The differences showed in Tab. 3.4 are striking where some grid quality values are cut by an order of magnitude. In the previous section was mentioned that skewness maximum values of around 0.3 were acceptable and here is further confirmation. The maximum skewness values seen in *Triangle*'s meshes almost double the ones found in *Distmesh* in the previous section (in what may be considered harder examples). From Fig. 3.3 one also gets a sense for the quality of the grids, Fig. 3.3(b) shows more evenly sized and shaped triangles which contributes for a more "structured" feel. While Figure 3.3(a) shows, as previously mentioned that it would, plenty of instances of 'right-angled triangles' that have a

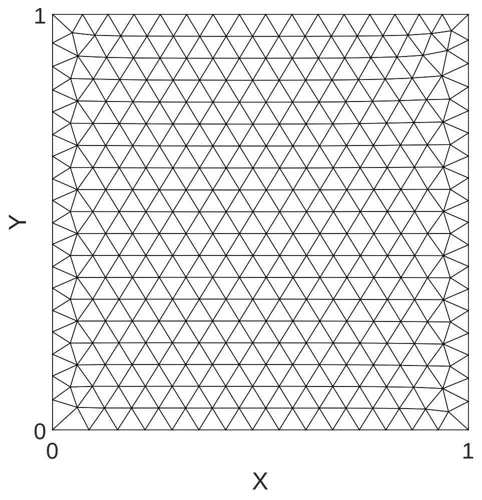
		$\alpha_N[deg]$		ψ	
	$N.Cells$	Avg	Max	Avg	Max
$Triangle$	787	11.0507	45.9806	0.1192	0.5541
$Formiga$	694	1.0648	17.4316	0.0177	0.1586
$Improvement$		90.36%	62.09%	85.11%	71.37%
$Triangle$	2216	10.3088	44.5079	0.1118	0.5548
$Formiga$	1789	0.4552	17.8990	0.0114	0.2104
$Improvement$		95.58%	59.78%	89.82%	62.08%
$Triangle$	10308	10.2110	48.2583	0.1127	0.5979
$Formiga$	10026	0.3737	18.0735	0.0048	0.2107
$Improvement$		96.34%	62.55%	95.71%	64.76%
$Triangle$	51602	10.3262	48.2534	0.1154	0.6384
$Formiga$	49511	0.2530	16.6002	0.0022	0.1707
$Improvement$		97.55%	65.60%	98.09%	73.26%
$Average$		94.96%	62.51%	92.19%	67.87%

Table 3.4: Comparison between Triangle and *Formiga* codes - Square domain case.

sizeable negative impact in the quality metrics. It's not difficult to conclude that the poor quality cells in *Triangle* are not solely placed on the boundary, like in *Distmesh* but scattered through the interior of the domain.



(a) Mesh over square domain generated with *Triangle*.



(b) Mesh over square domain generated with *Formiga*.

Figure 3.3: Triangular meshes over a square domain.

3.3 Gmsh comparison

Unlike the mesh generators in previous Sections, *Gmsh* [38] claims to be a Finite Element specific mesh generator. Which makes for a good comparison since it will further relay the need for specific Finite Volume algorithms.

Two cases were tested and both using an uniform point distribution. Concerning *Gmsh*, all meshes were generated using the Delaunay algorithm. Just like in the previous Section an output file from the software was read and the mesh analysed in MATLAB.

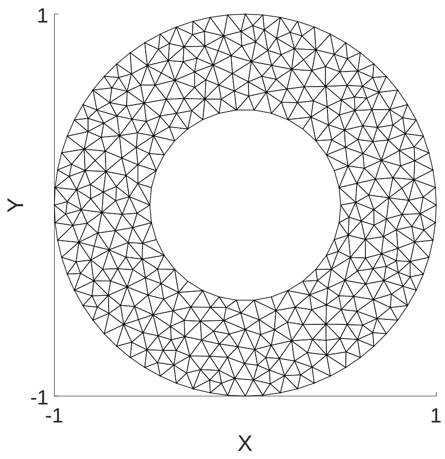
3.3.1 Annulus Domain

This domain consists of two concentric circles of radius $r = 0.5$ and $r = 1$ that shall be used again in Section 4.4.

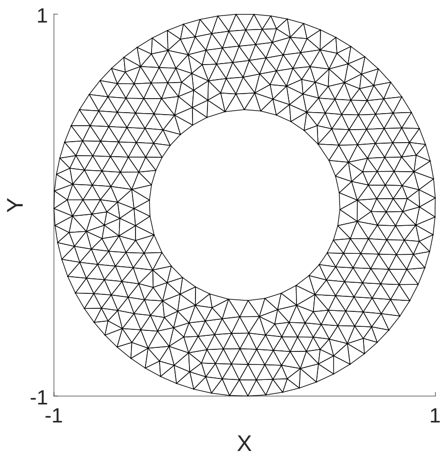
As in the previous Sections there is a big difference in the quality metrics between algorithms. Despite the fact that the maximum skewness and warp angle of *Gmsh*'s grids are closer to *Triangle*'s level than *Distmesh*'s, the average values of these quality criteria behaves in the opposite way. This may be indication that the presence of several instances of 'right-angled triangles', easily spotted in the interior boundary of Fig. 3.4(a), is as in *Distmesh* the main reason for less than ideal grid quality values.

		$\alpha_N[deg]$		ψ	
	$N.Cells$	Avg	Max	Avg	Max
$Gmsh$	698	5.6226	39.7106	0.1089	0.4104
$Formiga$	668	2.3509	16.7082	0.0468	0.1912
$Improvement$		58.19%	57.93%	57.02%	53.42%
$Gmsh$	2587	4.9492	40.4105	0.0976	0.3780
$Formiga$	2481	1.8891	14.2556	0.0226	0.1929
$Improvement$		61.83%	64.72%	76.88%	48.96%
$Gmsh$	13350	5.9947	40.6962	0.0934	0.4738
$Formiga$	13130	1.2501	14.1422	0.0101	0.1997
$Improvement$		79.15%	65.25%	89.21%	57.85%
$Gmsh$	40364	5.7622	44.7658	0.0935	0.5101
$Formiga$	40290	0.8110	14.6183	0.0053	0.2027
$Improvement$		85.93%	67.34%	94.31%	60.26%
$Average$		71.27%	63.81%	79.36%	55.12%

Table 3.5: Comparison with *Gmsh* - Annulus domain.



(a) Mesh generated with *Gmsh*.



(b) Mesh generated with *Formiga*.

Figure 3.4: *Gmsh*'s and *Formiga*'s meshes over the annulus domain case.

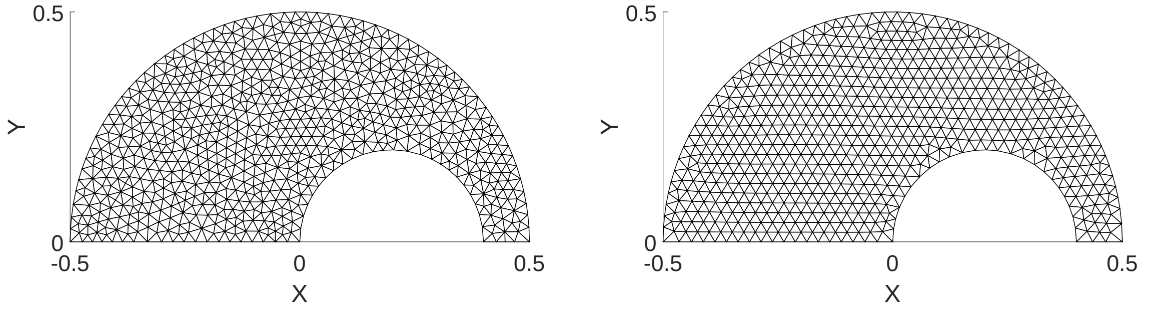
3.3.2 Uncentered semi-circles

Similarly to the previous case, this domain will also be used again, to solve a diffusive problem (Section 4.2). What this means is that more attention was put into improving the warp angle although the skewness was not completely despised.

Concerning only mesh generation purposes the domain consists of two semi-circles with $r = 0.5$ and $r = 0.2$, respectively, being the last one's centre shifted by $\Delta C = 0.2$ to the right in regards to the first. The results of both algorithms are in Table 3.6 and show again *Formiga*'s meshes pulling ahead and the *Gmsh*'s ones exhibiting a similar behaviour as in the previous case.

		$\alpha_N[deg]$		ψ	
	$N.Cells$	Avg	Max	Avg	Max
$Gmsh$	268	6.8941	39.5778	0.1005	0.3708
$Formiga$	236	2.9493	17.9970	0.0373	0.2580
$Improvement$		57.42%	54.53%	62.87%	30.42%
$Gmsh$	1350	5.9493	36.3349	0.0937	0.4362
$Formiga$	1237	1.1496	19.8948	0.0193	0.2060
$Improvement$		80.68%	45.25%	79.43%	52.77%
$Gmsh$	12062	5.8193	38.6064	0.0924	0.4913
$Formiga$	11634	0.7731	17.7052	0.0063	0.1966
$Improvement$		86.71%	54.14%	93.22%	59.99%
$Gmsh$	38131	5.6586	41.6409	0.0897	0.5126
$Formiga$	37149	0.5090	18.1830	0.0035	0.2025
$Improvement$		91.00%	56.33%	96.12%	60.49%
$Average$		78.95%	52.56%	82.91%	50.92%

Table 3.6: Comparison with *Gmsh* - Uncentered semi-circles.



(a) Mesh generated with *Gmsh*. (b) Mesh generated with *Formiga*.

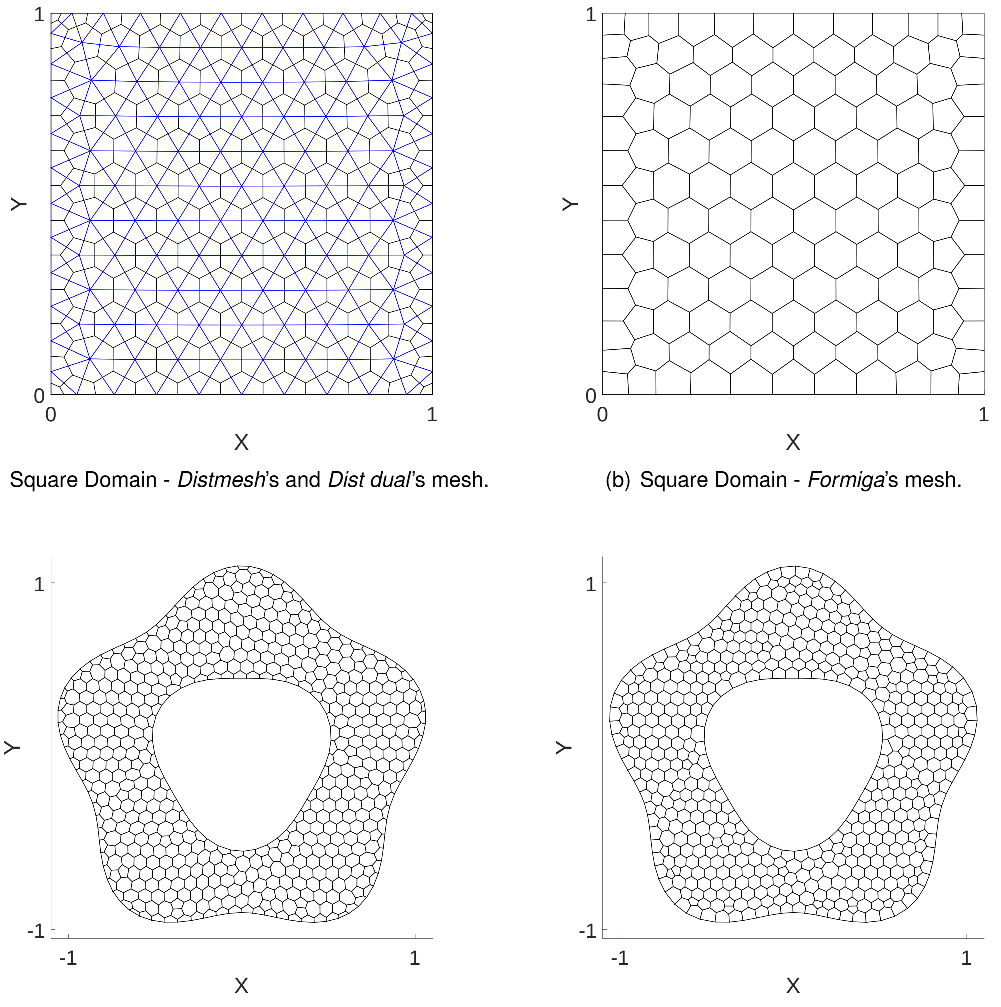
Figure 3.5: *Gmsh*'s and *Formiga*'s meshes over uncentered semi-circles.

3.4 Distmesh and *Formiga*’s polyhedral grids comparison

In this Section, the dual of *Distmesh* generated grids (from here onwards these meshes will be referred to as *Dist dual*) will be computed to showcase the implemented smoothing algorithm and the reasons why *Formiga* does not use *Distmesh*’s grids directly but a close version of the meshes seen in previous Sections as primal mesh for generation of polyhedral grids.

The examples in this Section will be the same as in Section 3.1 together with to the unit length square with uniform size distribution used in the comparison with *Triangle*. The algorithm used to compute the dual grid of *Distmesh* is the same used in *Formiga*, this should provide good grounds to establish a real comparison between the two with exception of the cases where concave cells are created since the proposed algorithm relies heavily on the smoothing stage to convert these cells in high quality convex ones. The reader will be reminded of this when relevant.

The same study done in Section 3.1 regarding the number of mesh generations needed to ensure results with a confidence level of at least 90% is required, these results are presented in Table 3.7.



(a) Square Domain - *Distmesh*’s and *Dist dual*’s mesh.

(b) Square Domain - *Formiga*’s mesh.

(c) Rose Shaped Domain - *Dist dual*’s mesh.

(d) Rose Shaped Domain - *Formiga*’s mesh.

Figure 3.6: Polyhedral meshes over a Square and Rose Shaped domains.

			$\alpha_N [deg]$		ψ	
	$Mesh\,\#$	$h_{ref} - h_{max}$	Avg	Max	Avg	Max
$Square\ w/\ circular\ hole$						
$Average$			0.7129	3.7677	0.0669	0.6178
$Standard\,Deviation$	1	$0.03 - 0.1$	0.0487	0.3454	0.0033	0.1821
$Sample\,Size\,Required$			2	3	1	24
$Average$			0.6382	4.7377	0.0591	0.5043
$Standard\,Deviation$	2	$0.02 - 0.1$	0.0547	0.9453	0.0047	0.0964
$Sample\,Size\,Required$			3	11	2	10
$Average$			0.6119	8.1719	0.0529	0.6173
$Standard\,Deviation$	3	$0.01 - 0.06$	0.0231	0.7003	0.0015	0.0752
$Sample\,Size\,Required$			1	2	1	5
$Average$			0.6091	11.1333	0.0501	0.7251
$Standard\,Deviation$	4	$0.006 - 0.04$	0.0201	0.8480	0.0010	0.0644
$Sample\,Size\,Required$			1	2	1	3
$Point\ and\ line\ sources$						
$Average$			0.7126	4.4594	0.0373	0.4248
$Standard\,Deviation$	1	$0.03 - 0.08$	0.0411	0.5893	0.0012	0.1060
$Sample\,Size\,Required$			1	5	1	17
$Average$			0.7207	6.4943	0.0330	0.3582
$Standard\,Deviation$	2	$0.02 - 0.07$	0.0226	1.1980	0.0015	0.0567
$Sample\,Size\,Required$			1	10	1	7
$Average$			0.6481	11.2083	0.0505	0.6110
$Standard\,Deviation$	3	$0.01 - 0.06$	0.0318	4.2428	0.0018	0.1563
$Sample\,Size\,Required$			1	40	1	18
$Average$			0.6399	14.1541	0.0383	0.5058
$Standard\,Deviation$	4	$0.005 - 0.04$	0.0285	2.6118	0.0010	0.0501
$Sample\,Size\,Required$			1	10	1	3
$NACA\ 0012\ airfoil$						
$Average$			1.9773	24.0362	0.0773	0.7496
$Standard\,Deviation$	1	$0.006 - 0.4$	0.3864	5.7594	0.0079	0.1069
$Sample\,Size\,Required$			11	16	3	6
$Average$			1.2700	16.7664	0.0442	0.5899
$Standard\,Deviation$	2	$0.005 - 0.1$	0.2211	3.5820	0.0052	0.1024
$Sample\,Size\,Required$			9	13	4	9
$Average$			1.2746	15.7093	0.0408	0.6407
$Standard\,Deviation$	3	$0.004 - 0.06$	0.3700	4.1681	0.0077	0.1244
$Sample\,Size\,Required$			23	20	10	11
$Average$			0.8211	14.4227	0.0332	0.6381
$Standard\,Deviation$	4	$0.003 - 0.04$	0.1032	3.5799	0.0024	0.0785
$Sample\,Size\,Required$			5	17	2	5

Table 3.7: Sample size required to achieve a 90% confidence level on polyhedral meshes generated through *Formiga*.

The five cases, spread over Tables 3.8 and 3.9, are a true testament to the advantages of the smoothing algorithm. If in the triangular cases the changes in mesh topology play a huge factor in mesh quality improvement the same cannot be said in here where changes in topology are used only to correct punctual situations providing minor improvements to the grid quality metrics.

			$\alpha_N[deg]$		ψ	
	h_{ref}	$N.Cells$	Avg	Max	Avg	Max
Square						
Dist dual		128	4.5597	24.6311	0.1129	0.7562
Formiga	0.1	130	0.4496	3.4777	0.0724	0.6008
Improvement			90.14%	85.88%	35.90%	20.55%
Dist dual		588	2.2988	25.5558	0.0519	0.7539
Formiga	0.045	589	0.3278	3.1826	0.0321	0.4951
Improvement			85.74%	87.55%	38.13%	34.33%
Dist dual		1310	1.5783	23.3094	0.0394	1.0111
Formiga	0.03	1311	0.5973	3.9385	0.0167	0.3810
Improvement			62.15%	83.10%	57.65%	62.32%
Dist dual		2931	0.7640	23.1727	0.0291	0.7958
Formiga	0.02	2929	0.1173	3.1856	0.0137	0.5830
Improvement			84.64%	86.25%	52.95%	26.74%
Average			80.67%	85.70%	46.16%	35.98%
Rose Shaped Domain						
Dist dual		556	4.5611	21.1243	0.1272	1.2927
Formiga	0.07	550	0.3464	6.7079	0.0782	0.7994
Improvement			92.40%	68.25%	38.53%	38.16%
Dist dual		2233	3.2515	24.4897	0.0660	1.0190
Formiga	0.035	2230	1.2257	11.8301	0.0431	0.7942
Improvement			62.30%	51.69%	34.69%	22.06%
Dist dual		3485	2.8050	22.8498	0.0536	1.0320
Formiga	0.028	3484	0.1955	2.4186	0.0257	0.4596
Improvement			93.03%	89.42%	51.97%	55.46%
Dist dual		10682	1.8485	21.1582	0.0302	1.0335
Formiga	0.016	10680	0.1091	3.2197	0.0138	0.3916
Improvement			94.10%	84.78%	54.07%	62.11%
Average			85.46%	73.53%	44.82%	44.45%

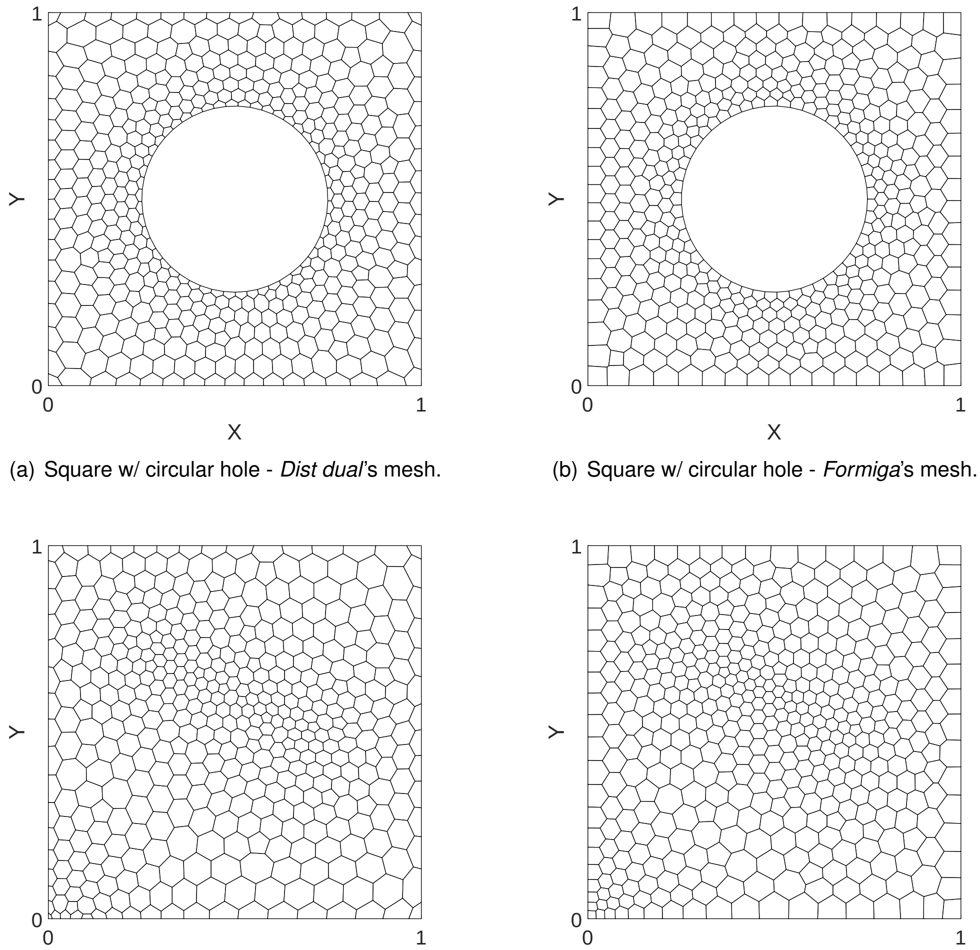
Table 3.8: Rose Shaped and Square domains quality metrics improvement.

As hinted at in Section 3.1, skewness values in polyhedral meshes are generally superior to their triangular counterparts. Examining the results it's not uncommon to find maximum skewness values in excess of 1 in the polyhedral meshes resulting from *Dist dual* which is far from ideal. *Formiga*'s skewness maximum values though higher than desired in some situations are for these cases all under 1 and represent a sizeable improvement over the other meshes.

	$Mesh \#$	$N.Cells$	$\alpha_N[deg]$ Avg	$\alpha_N[deg]$ Max	ψ Avg	ψ Max
Square w/ circular hole						
Dist dual		464	4.6377	23.8181	0.1226	0.9806
Formiga	1	463	0.7129	3.7677	0.0669	0.6178
Improvement			84.63%	84.18%	45.48%	37.00%
Dist dual		797	4.4295	24.3158	0.1055	1.0131
Formiga	2	801	0.6382	4.7377	0.0591	0.5043
Improvement			85.59%	80.52%	43.98%	50.22%
Dist dual		1853	4.1057	25.3994	0.0898	1.0461
Formiga	3	1851	0.6119	8.1719	0.0529	0.6173
Improvement			85.10%	67.83%	41.03%	40.99%
Dist dual		3243	3.9663	26.3371	0.0825	1.0649
Formiga	4	3229	0.6091	11.1333	0.0501	0.7251
Improvement			84.64%	57.73%	39.22%	31.91%
Average Improvement			84.99%	72.56%	42.43%	40.03%
Point and line sources						
Dist dual		453	4.4699	23.8199	0.0956	0.9967
Formiga	1	452	0.7126	4.4594	0.0373	0.4248
Improvement			84.06%	81.28%	61.00%	57.38%
Dist dual		743	4.0836	23.7484	0.0836	1.0349
Formiga	2	735	0.7207	6.4943	0.0330	0.3582
Improvement			82.35%	72.65%	60.50%	65.39%
Dist dual		1552	3.7823	24.5515	0.0692	1.0329
Formiga	3	1554	0.6481	11.2083	0.0505	0.6110
Improvement			82.86%	54.35%	27.03%	40.84%
Dist dual		3055	3.6115	24.3762	0.0614	1.0553
Formiga	4	3054	0.6399	14.1541	0.0383	0.5058
Improvement			82.28%	41.93%	37.59%	52.07%
Average Improvement			82.89%	62.55%	46.53%	53.92%
NACA 0012 airfoil						
Dist dual		582	4.4225	42.8351	0.1064	1.1407
Formiga	1	575	1.9773	24.0362	0.0773	0.7496
Improvement			55.29%	43.89%	27.39%	34.29%
Dist dual		1537	4.4561	45.0655	0.1017	1.0086
Formiga	2	1527	1.2700	16.7664	0.0442	0.5899
Improvement			71.50%	62.80%	56.52%	41.51%
Dist dual		2634	3.8302	46.6341	0.0761	0.9422
Formiga	3	2630	1.2746	15.7093	0.0408	0.6407
Improvement			66.72%	66.31%	46.35%	32.00%
Dist dual		4400	2.8624	44.9231	0.0607	0.9715
Formiga	4	4400	0.8211	14.4227	0.0332	0.6381
Improvement			71.31%	67.89%	45.24%	34.32%
Average Improvement			66.21%	60.22%	43.87%	35.53%

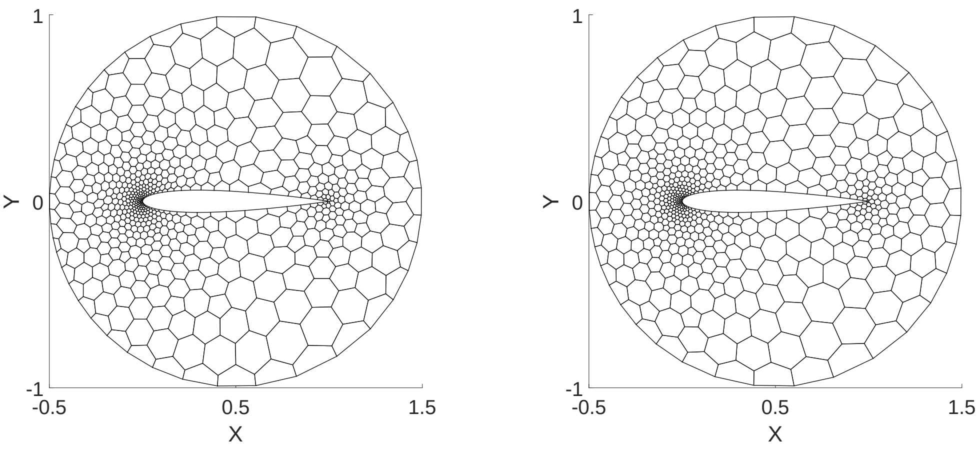
Table 3.9: Quality improvements on polyhedral meshes over a Square with circular hole, Square with size function based in a point and a line source, and NACA 0012 airfoil.

The higher skewness values are usually found in the edges shared between two boundary defining cells, that is why one can clearly see in the right hand side meshes of Figures 3.6 and 3.7 a clear increase in these edges dimension and consequently cell size. This is a result of the application of the skewness correction forces.



(a) Square w/ circular hole - *Dist dual*'s mesh. (b) Square w/ circular hole - *Formiga*'s mesh.

(c) Point and Line Sources - *Dist dual*'s mesh. (d) Point and Line Sources - *Formiga*'s mesh.



(e) NACA 0012 airfoil - *Dist dual*'s mesh. (f) NACA 0012 airfoil - *Formiga*'s mesh.

Figure 3.7: Examples of meshes from cases in Table 3.9.

Still on the topic of boundary cells, one of the reasons why *Formiga* doesn't compute the dual of *Distmesh*'s grids may be seen in a couple of images though more clearly in Figure 3.6. On the left and right boundaries of Fig. 3.6(a) one may notice that the cell's size is highly irregular, while in Figure 3.6(b) it's much more uniform. Surely the smoothing algorithm plays a factor in this difference however the main explanation for this situation is that the cells mentioned in Figure 3.6(a) are the dual of 'right-angled triangles'. Despite not having the same negative impact seen in triangular meshes these still cause variations in cell size which do not favour, for example, gradient computations. In Figure 3.6(a) the primal mesh is represented in gray to showcase the 'right-angled triangles'.

Non-orthogonality values are of similar order of magnitude compared to triangular meshes with no improvements, and are further enhanced by the proposed algorithm. The airfoil case scores higher warp angle values than the remaining examples, especially when no corrections are applied. As addressed in the beginning of this Section the algorithm developed to transform concave cells in convex ones only makes changes in topology for the smoothing algorithm to improve upon. Since the meshes created with *Dist dual* are not improved with the smoothing algorithm the changes in connectivities made are left unattended and result in significant non-orthogonality angles.

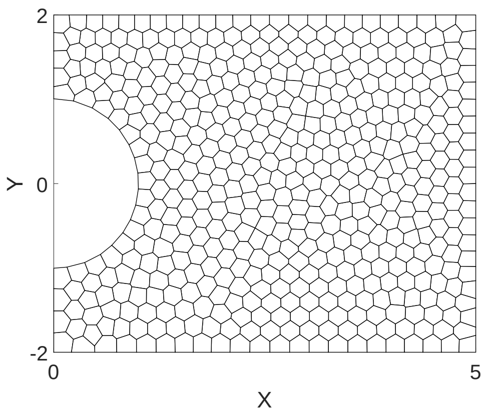
In similar fashion to what was said regarding the quality of *Distmesh*'s inner triangles the same holds true for its dual polyhedra as what were equilateral, or close to it, triangles become regular hexagons. With this said the main differences between *Dist dual* and *Formiga* are still found in cells that define the domain.

3.5 Polymesher comparison

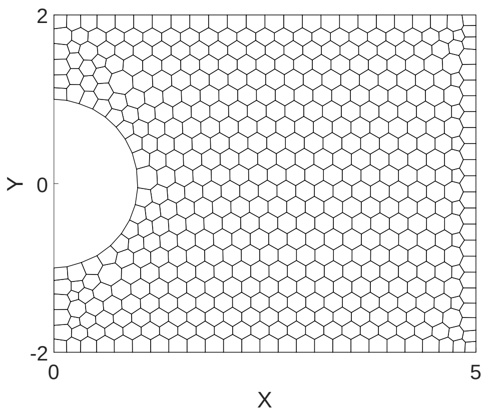
Polymesher [13] is a MATLAB code that uses the Centroidal Voronoi Tessalation coupled with Loyd's algorithm to produce 2D polyhedral meshes. The two test cases chosen for the comparison are the Michell cantilever domain and a Wrench domain both shown in [13] and readily available in the downloadable version of *Polymesher*. Since *Polymesher* requires the maximum number of iterations allowed as input, this number was used so that the time spent by both *Polymesher* and *Formiga* was approximately the same.

3.5.1 Michell's cantilever domain

For the first example, Table 3.10 shows closer warp angle values between both generators when compared to previous Sections but with *Formiga* still in the lead in most mesh sizes. The big disparity is in skewness values, the meshes attained through *Polymesher* show very large maximum skewness values undoubtedly due to the presence of very small edges that may be spotted in Fig. 3.8(a). *Polymesher* has an algorithm in place to detect and remove those edges although not all were removed. At this point one has to consider the possibility that method used to compare both generators may have hindered this generator's results in a way though each *Polymesher* value presented was averaged out of a simulation set of size $m = 40$, in this test case and the following.



(a) Michell Cantilever Domain - *Polymesher*'s Mesh.



(b) Michell Cantilever Domain - *Formiga*'s Mesh.

Figure 3.8: Michell cantilever domain - Meshes comparison.

Fig. 3.8 shows the meshes created with both generators. The mesh generated with *Formiga* appears to be more "ordered" but the one obtained with *Polymesher* also presents well shaped cells, including in the boundaries. This leads to the belief the removal of small edges will produce a mesh of similar quality to the one in Fig. 3.8(b).

		$\alpha_N[deg]$		ψ	
	$N.Cells$	Avg	Max	Avg	Max
$PolyMesher$	500	0.2258	7.2941	0.1060	2.3061
$Formiga$	493	0.2056	3.3111	0.0388	0.5424
$Improvement$		8.95%	54.61%	63.43%	76.48%
$PolyMesher$	1000	0.1119	6.4478	0.0844	2.0839
$Formiga$	959	0.1354	3.5416	0.0284	0.4515
$Improvement$		-21.03%	45.07%	66.35%	78.33%
$PolyMesher$	2000	0.1400	7.8442	0.0923	2.5226
$Formiga$	1956	0.0856	2.4793	0.0196	0.4565
$Improvement$		38.82%	68.39%	78.76%	81.90%
$PolyMesher$	5000	0.0834	7.5170	0.0811	2.7679
$Formiga$	4980	0.0583	2.7532	0.0136	0.6685
$Improvement$		30.11%	63.37%	83.16%	75.85%

Table 3.10: Comparison between *PolyMesher* and *Formiga* - Michell cantilever domain.

3.5.2 Wrench cantilever domain

In this example, *Polymesher* scores remarkably low warp angle average values, better than *Formiga* in a few mesh sizes, but suffers from the same problem as the previous example in what concerns to the maximum skewness and the occurrence of small edges. What's very different from the previous case, is the maximum warp angle values. At first these values were thought to be wrong though upon closer inspection instances like the one shown in Fig. 3.9 were detected. What the mentioned Figure details is a situation where the domain is not properly represented due to the presence of a small edge that, in it's turn, results in very significant non-orthogonality values. Since this discovery the situation has been identified in [13], as well, Figs. 11 and 12(d) of said publication show the same situation found here.

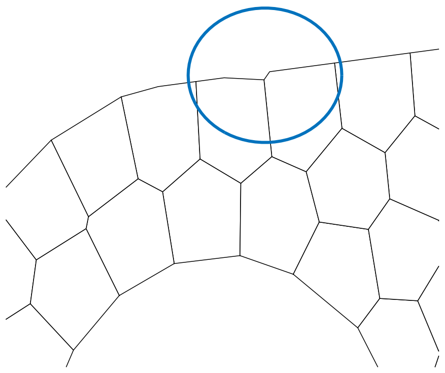


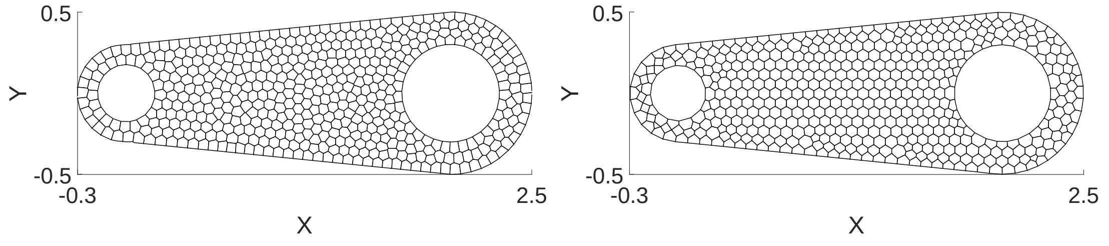
Figure 3.9: Detail of mesh generated through PolyMesher.

Formiga represents the geometry properly but has high values of maximum skewness, partly due to the difficulties from this geometry, partly due to the low cell number used, however for the most refined mesh these values drop considerably.

		$\alpha_N[deg]$		ψ	
	$N.Cells$	Avg	Max	Avg	Max
$PolyMesher$	500	0.4182	42.7477	0.1024	2.0165
$Formiga$	497	0.3312	2.5840	0.0892	0.8812
$Improvement$		20.81%	93.96%	12.97%	56.30%
$PolyMesher$	1000	0.2195	41.7440	0.0816	1.9672
$Formiga$	994	0.2256	2.4970	0.0666	0.9098
$Improvement$		−2.77%	94.02%	18.41%	53.75%
$PolyMesher$	2000	0.1484	43.4126	0.0816	2.2723
$Formiga$	2006	0.2008	2.4644	0.0585	0.9722
$Improvement$		−35.33%	94.32%	28.30%	57.22%
$PolyMesher$	5000	0.0820	43.1078	0.0731	2.4718
$Formiga$	4589	0.0803	2.0624	0.0256	0.4682
$Improvement$		2.08%	95.22%	65.01%	81.06%

Table 3.11: Comparison between *PolyMesher* and *Formiga* - Wrench domain.

Again, the overall quality of *Polymesher*'s grids is quite good as proved by it's mesh quality average values though the presence of small edges results in high maximum values that hinder it's overall quality. *Formiga*, on the other hand, can solve and reduce these maximum values.



(a) Michell Cantilever Domain - *Polymesher*'s Mesh. (b) Michell Cantilever Domain - *Formiga*'s Mesh.

Figure 3.10: Wrench domain - Polyhedral meshes comparison.

3.6 3D polyhedral grid generator

In this Section, the smoothing algorithm, with 3D specific corrections presented in Sub-Section 2.8.2, is applied to 3D polyhedral meshes. As mentioned previously and unlike the previous Sections, there are not modifications in cell connectivities and all improvements are achieved through the smoothing algorithm. Similarly to what was verified in 2D for triangular meshes, tetrahedral meshes require modifications in cell connectivities for the smoothing algorithm to be effective. Since these are not developed yet, no results are shown for tetrahedral meshes.

In the previous Chapter was mentioned that no 3D polyhedral mesh generation algorithm guarantees that all its faces are planar. This represents a problem for this work as the quality metrics used here are measured in these planar or quasi-planar faces. Despite the existence of a correction to make the problematic faces closer to planar faces, there is no guarantee that the results presented in the next Tables are not affected.

		α_N $[deg]$		ψ		γ $[deg]$	
	$N.Cells$	Avg	Max	Avg	Max	Avg	Max
$Dist\ dual$		11.5769	61.9178	0.7059	3.8378	5.4278	35.4952
$Formiga$	172	7.5611	26.7624	0.5765	3.6097	3.8899	22.9536
$Improvement$		34.69%	56.78%	18.33%	5.94%	28.33%	35.33%
$Dist\ dual$		10.0801	50.3444	0.5321	3.6693	4.9664	34.7282
$Formiga$	587	7.2607	34.6195	0.4334	2.5099	3.8205	18.7082
$Improvement$		27.97%	31.23%	18.55%	31.60%	23.07%	46.13%
$Dist\ dual$		9.5505	69.0543	0.4791	3.7868	4.9880	46.0200
$Formiga$	1069	6.9942	32.4461	0.3981	3.5010	3.9130	31.4415
$Improvement$		26.77%	53.01%	16.90%	7.55%	21.55%	31.68%
$Dist\ dual$		8.5494	68.6166	0.4177	4.3026	4.9461	47.7171
$Formiga$	2485	5.9369	35.7989	0.3387	3.3261	3.7124	27.8585
$Improvement$		30.56%	47.83%	18.90%	22.70%	24.94%	41.62%
$Average$		30.00%	47.21%	18.17%	16.95%	24.48%	38.69%

Table 3.12: Grid quality improvements in 3D polyhedral meshes - Sphere domain case.

Two examples were tested. In the first one, tetrahedral meshes generated with *Distmesh*, inside a sphere centred in $(0, 0, 0)$ and with radius $r = 1$, are used as primal grids from which the corresponding polyhedral meshes are computed. Similarly to the previous Sections these meshes are referred as *Dist dual*. These meshes are then subjected to the smoothing algorithm resulting in an, hopefully, improved grid which is simply referred as *Formiga*.

The results in Table 3.12 show significant improvements in all metrics though it is clear that the order of magnitude of the presented parameters is higher than the ones in 2D meshes. The maximum skewness values are the most evident case though as mentioned previously the metric used is not as easy to interpret as in 2D and depending on the case, for ratio skewness values of around 3, $\mathbf{f_i}$ may still lie within it's corresponding face. The reduction of the concavity angle γ is similar to the one observed in the warp angle α_N.

		α_N $[deg]$		ψ		γ $[deg]$	
	$N.Cells$	Avg	Max	Avg	Max	Avg	Max
$Gmsh\ dual$		14.4834	52.3444	0.6266	19.8873	5.1998	33.6861
$Formiga$	657	8.4499	37.2393	0.5045	5.1333	3.8720	24.3293
$Improvement$		41.66%	28.86%	19.50%	74.19%	25.54%	27.78%
$Gmsh\ dual$		13.6636	53.5918	0.5664	3.1936	5.1533	30.3898
$Formiga$	980	8.6706	28.7646	0.4726	2.4376	3.9992	24.5013
$Improvement$		36.54%	46.33%	16.55%	23.67%	22.40%	19.38%
$Gmsh\ dual$		12.8122	48.6323	0.5696	28.0968	5.2656	35.7589
$Formiga$	1699	7.2658	30.2027	0.4152	3.2409	4.4490	29.0150
$Improvement$		43.29%	37.90%	27.10%	88.47%	15.51%	18.86%
$Gmsh\ dual$		12.1274	50.7878	0.5358	34.4055	5.6124	38.0194
$Formiga$	3607	6.7832	43.3126	0.4048	3.4627	4.6120	26.9722
$Improvement$		44.07%	14.72%	24.45%	89.94%	17.83%	29.06%
$Average$		41.39%	31.95%	21.90%	69.07%	20.32%	23.77%

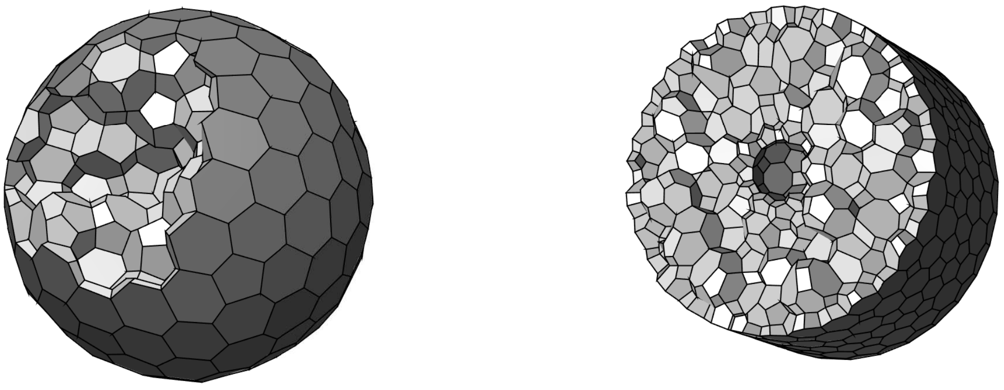
Table 3.13: Grid quality improvements in 3D polyhedral meshes - Cylinder with sphere inside case.

The second case tested consists of a cylinder given by equation 3.2, with $x \in [0, 1]$, $a = 0.5$ and $r = 0.5$ with a sphere of radius $r = 0.1$ and centred at $(0.5, 0.5, 0.5)$ inside.

$$(y - a)^2 + (z - a)^2 = r^2 \tag{3.2}$$

As mentioned previously it was not possible to generate a valid tetrahedral mesh with *Distmesh*, so *Gmsh* was used to provide the primal grids. The results of the grid quality metrics are in Table 3.13 and show similar improvements to the ones seen in the sphere case with exception to the maximum skewness values that were improved by a huge margin as some of the initial polyhedral meshes had very poor quality in this regard.

The presented values are harder to validate than in two dimensional cases, as visual representations offer little help. Despite not helping the interpretation, of the grid quality values, the representation of the obtained polyhedral meshes for each example is presented in Fig. 3.11 with cut-outs to show some of the internal cells. Both presented meshes were generated with the *Formiga* 3D version.



(a) Sphere case.

(b) Cylinder with sphere inside case.

Figure 3.11: Examples of 3D meshes generated with *Formiga*.

Chapter 4

Numerical Results

In this Chapter problems with analytical solution are solved for several meshes so that conclusions may be reached regarding the most accurate cell type and the impact of the improvements of *Formiga* made on the numerical accuracy. To do it, triangular and polyhedral meshes from *Formiga*, *Distmesh* and *Dist dual* with different number of cells are used for each analytical case and their average and maximum numerical errors computed.

The discretization schemes used are the ones introduced in Section 2.2, as a reminder four families of schemes are used where the first $D-NRML, C-LIN$ does not use any correction for unstructured grids while the other three have different contingencies in place to deal with grid quality issues.

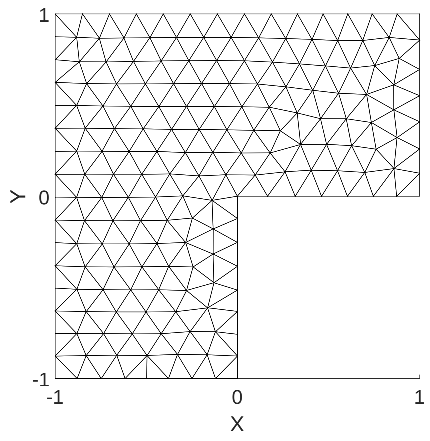
4.1 Laplace Equation in an L-Shaped Domain

The Laplace equation clearly only has a diffusive term so the grid quality parameter that should receive the largest amount of attention will be the warp angle. The other criteria are not despised and are also improved by *Formiga*, as shown in the previous Chapter, though not presented to try and keep this Section and others in similar situations easier to read.

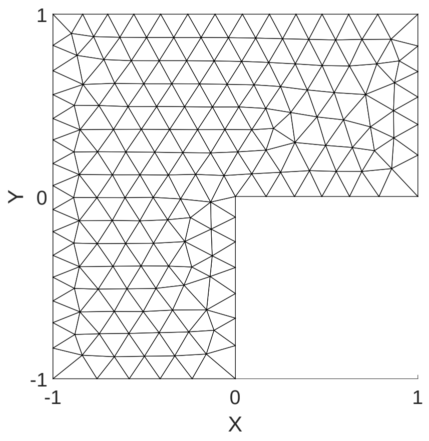
Over Tables 4.1 and 4.2 the non-orthogonality angles of all 16 meshes used are presented. The main conclusion to draw from this is that *Formiga*'s grids have better warp angles across all sizes and cell types. Similarly to what happened in Section 3.4 with the NACA 0012 airfoil the polyhedral meshes generated through *Dist dual* have higher warp angles caused by the elimination of the concave cell create by the concavity in the domain. Despite this, the difference in quality between these meshes and the ones from *Formiga* goes way beyond the cells affected by this modification, Fig. 4.3(a) also shows small boundary defining cells that cause higher skewness values and also sizable variations in cell volume that affect cell's gradients computations. As mentioned before these variations in cell size are caused by the presence of 'right-angled triangles' in the primal mesh and may be spotted in Fig. 4.1(a).

To verify if the proposed smoothing algorithm and connectivity modifications bare any weight in the increase of the numerical accuracy, Dirichlet boundary conditions are applied and the solution of the

Laplace equation compared against the known analytical solution, presented in Eq. 4.1. Although the solution is showed in polar coordinates the numerical solution is computed in Cartesian coordinates by SOL.



(a) Triangular mesh generated with *Distmesh*.



(b) Triangular mesh generated with *Formiga*.

Figure 4.1: Examples of triangular meshes over a L-shaped domain.

$$\phi(r,\theta) = r^{\frac{2}{3}} sin(\frac{2\theta}{3}) \tag{4.1}$$

	$Mesh\#$	$N.Cells$	$\alpha_N\ [deg]$ Avg	$\alpha_N\ [deg]$ Max
$Distmesh$		212	6.5178	32.1483
$Formiga$	1	206	3.6224	17.8596
$Improvement$			44.42%	44.45%
$Distmesh$		2124	2.4168	30.6401
$Formiga$	2	2119	1.6487	13.8971
$Improvement$			31.78%	54.64%
$Distmesh$		9922	1.1776	34.0130
$Formiga$	3	9915	0.9570	13.6313
$Improvement$			18.73%	59.92%
$Distmesh$		26740	0.6920	35.0352
$Formiga$	4	26733	0.5272	13.1603
$Improvement$			23.81%	62.44%

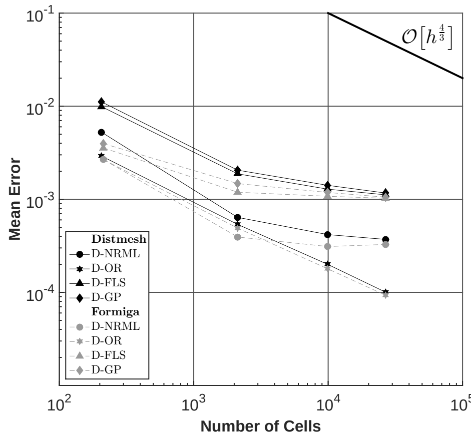
Table 4.1: L-shape domain - Triangular meshes warp angles.

	$Mesh\#$	$N.Cells$	$\alpha_N\ [deg]$ Avg	$\alpha_N\ [deg]$ Max
$Dist\ dual$		217	5.2730	28.4002
$Formiga$	1	219	0.4601	5.6465
$Improvement$			91.27%	80.12%
$Dist\ dual$		2209	1.2765	32.6320
$Formiga$	2	2207	0.1795	3.1428
$Improvement$			85.94%	90.37%
$Dist\ dual$		10782	0.7262	39.2428
$Formiga$	3	10783	0.0967	14.3688
$Improvement$			86.69%	63.38%
$Dist\ dual$		24138	0.4770	35.9490
$Formiga$	4	24138	0.0753	7.5522
$Improvement$			84.22%	78.99%

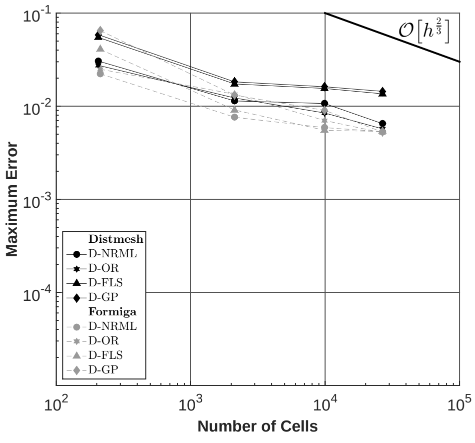
Table 4.2: L-shape domain - Polyhedral meshes warp angles.

Over Fig. 4.2 and Table 4.3 all important information regarding the simulations made with triangular meshes is condensed. The values in the Table reflect the improvement in accuracy when comparing the meshes from *Distmesh* with *Formiga*'s, meaning: $Improvement = \frac{error(Distmesh) - error(Formiga)}{error(Distmesh)}$.

The first and most important result extractable from the informations is that the improvements made with *Formiga* definitely show, that its meshes score more accurate results in all mesh sizes and numerical schemes. The scheme $D - OR$ obtained the most accurate results being the only one that managed to keep the average error slope through all grid sizes. This was also the one that showed smaller improvements with the application of all aforementioned corrections. As for the other schemes that have corrections for unstructured grids ($D - FLS$ and $D - GP$) they show a similar level of improvement to $D - NRML$ that has no warp angle correction of it's own.



(a) Average error.



(b) Maximum error.

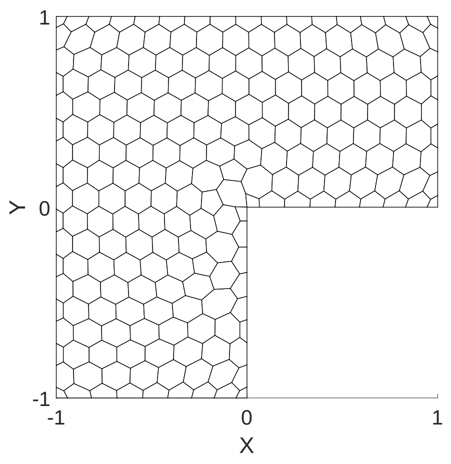
Figure 4.2: L-shaped domain - Triangular meshes numerical error.

		$D - NRML$		$D - OR$		$D - FLS$		$D - GP$	
	$Mesh\,\#$	Avg	Max	Avg	Max	Avg	Max	Avg	Max
$Improvement$	1	49.04%	26.88%	8.24%	9.11%	63.96%	25.45%	64.32%	−10.97%
$Improvement$	2	38.47%	33.19%	8.87%	−9.61%	36.62%	47.51%	28.31%	28.09%
$Improvement$	3	25.60%	44.94%	10.45%	16.99%	16.19%	64.45%	16.20%	44.22%
$Improvement$	4	11.70%	17.87%	7.31%	9.29%	8.19%	60.60%	9.70%	63.52%
$Average$		31.21%	30.72%	8.72%	6.44%	31.24%	49.50%	29.63%	31.21%

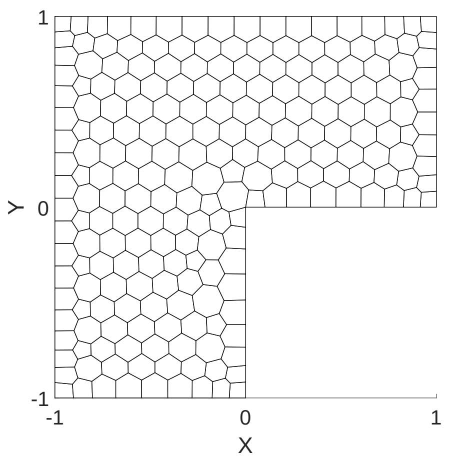
Table 4.3: L-shaped domain - Triangular meshes numeric error improvement.

The results from polyhedral meshes are in Fig. 4.4 and Table 4.4 and tell a similar tail to the one of the triangular meshes, the changes proposed in this work score more accurate results. The average improvement results show very large improvements of the error average for all schemes but $D - OR$, with $D - FLS$ and $D - GP$ improving by nearly an order of magnitude.

The maximum error for this case is located at the singularity vertex of coordinates $(0, 0)$ where the analytic gradient is infinite. With this being said the *Formiga* code meshes manage to obtain large improvements in this type of error. $D - OR$ average improvement values see little difference though a closer inspection of the values in it's respective columns of Tab. 4.4 proves that this scheme's results are improved by *Formiga* for more refined mesh sizes and worsened for less refined. Over Sections of the past Chapter was mentioned that the corrections made resulted in larger boundary defining cells, what may be confirmed by comparing Figs. 4.3(a) and 4.3(b). Despite this resulting in better quality faces and ultimately cells it also reduces definition near the limits of the domain what in this particular case may have had more importance than the quality improvements made.

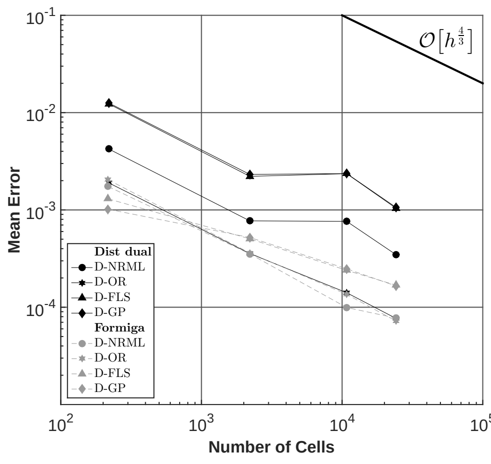


(a) Polyhedral mesh generated with *Dist dual*.

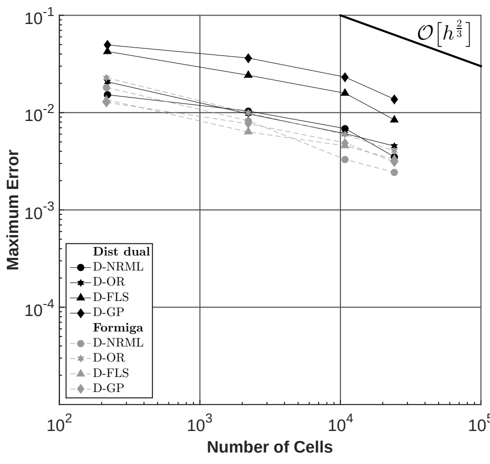


(b) Polyhedral mesh generated with *Formiga*.

Figure 4.3: Examples of polyhedral meshes over a L-shaped domain.



(a) Average error.



(b) Maximum error.

Figure 4.4: L-shaped domain - Polyhedral meshes numerical error.

		$D-NRML$		$D-OR$		$D-FLS$		$D-GP$	
	$Mesh\,\#$	Avg	Max	Avg	Max	Avg	Max	Avg	Max
$Improvement$	1	58.87%	−18.60%	−7.53%	−9.96%	89.45%	68.16%	91.92%	74.27%
$Improvement$	2	54.25%	20.67%	−1.32%	−2.61%	77.19%	73.88%	77.53%	78.90%
$Improvement$	3	86.98%	51.72%	3.71%	2.21%	89.85%	71.17%	89.50%	78.96%
$Improvement$	4	77.69%	31.02%	5.94%	10.75%	83.84%	60.74%	84.43%	77.38%
$Average$		69.45%	21.20%	0.20%	0.10%	85.08%	68.49%	85.85%	77.38%

Table 4.4: L-shaped domain - Polyhedral meshes numeric error improvement.

Interestingly the scheme without any grid quality corrections managed to be the most accurate for two of the four sizes tested though for the most refined mesh $D-OR$ managed to edge out $D-NRML$, but probably this is a specific result for this analytical case.

In Table 4.5 a comparison is established between triangular and polyhedral meshes by measuring the improvement when changing from the first to the later. The first line of the Table is labeled $Dist$ as it compares the triangular meshes generated with *Distmesh* against their dual counterparts create through *Dist dual* while the second line, labeled $Formiga$, compares the triangular and polyhedral meshes created with this generator.

	$D-NRML$		$D-OR$		$D-FLS$		$D-GP$	
	Avg	Max	Avg	Max	Avg	Max	Avg	Max
$Dist$	-19.50%	35.02%	30.35%	23.37%	-29.93%	4.48%	-21.17%	-30.83%
$Formiga$	47.36%	27.10%	24.09%	17.78%	70.59%	38.06%	75.58%	52.13%

Table 4.5: L-shaped domain - Triangular versus polyhedral meshes, numerical accuracy.

The results show that for the lower quality meshes it is difficult to draw a conclusion as to which mesh type has better accuracy. Triangular meshes score better average error results for every scheme but $D-OR$, however polyhedral meshes have the best maximum error values except for $D-GP$ and the best average error for the $D-OR$ scheme which was the most accurate for this example.

As for *Formiga*'s meshes there it is no doubt as to what mesh type is more accurate, polyhedral meshes lead the way in both average and maximum errors on all schemes, what should come as no surprise as values in Tabs. 4.1 and 4.2 showed a clear advantage in average warp angle values for polyhedral meshes. Maximum values are also, for the most part, better in polyhedral meshes. In here resides one characteristic of polyhedral meshes, though it is possible to further improve their average quality metrics, compared to triangular ones, sometimes it might be more challenging to control the maximum values as small faces may appear and hinder mesh quality. Obviously by controlling all the generator's parameters it is relatively easy to solve this predicament though in a generator that's intended to be automatic and to have as little human input as possible, it is not easy to always avoid these small edges.

4.2 Eccentric cylinder condensers

Taking the domain defined in Sub-Section 3.3.2 the Laplace equation is solved again. The solution of this equation will determine the electric potential field caused by two circular capacitors of different charges. Due to symmetry of the geometry, only the upper half of the complete problem is considered.

Mesh quality metrics of interest and representations are in Tables 4.6 and 4.7, and Figure 4.5. Grid quality values behave similarly to previous Sections, *Formiga*'s meshes score better average and maximum warp angle with the polyhedral meshes being superior to the triangular meshes values, especially in the average ones.

The analytical solution for this problem is known and given by Eq. 4.2, where the constants correspond to the following values: $a = 0.375,\ O = 0.625,\ K = 1.442695041$ and $\phi_0 = 1$. Obtained as a consequence of setting the inner capacitor's potential to one and the exterior to zero.

$$\phi(x,y) = K \ln\left(\frac{\sqrt{(x-a-O)^2+y^2}}{\sqrt{(x+a-O)^2+y^2}}\right) - \phi_0 \tag{4.2}$$

	$Mesh\#$	$N.Cells$	α_N $[deg]$ Avg	α_N $[deg]$ Max
$Distmesh$		244	5.8582	30.7789
$Formiga$	1	236	2.6847	13.1446
$Improvement$			54.17%	57.29%
$Distmesh$		1241	3.2804	36.4465
$Formiga$	2	1237	1.4663	8.5076
$Improvement$			55.30%	76.66%
$Distmesh$		11638	1.4271	28.3656
$Formiga$	3	11634	0.7803	8.5063
$Improvement$			45.32%	70.01%
$Distmesh$		37157	0.8466	31.9411
$Formiga$	4	37149	0.5150	11.7904
$Improvement$			39.16%	63.09%

Table 4.6: Eccentric cylinder condensers - Triangular meshes warp angles.

	$Mesh\#$	$N.Cells$	α_N $[deg]$ Avg	α_N $[deg]$ Max
$Dist\ dual$		246	4.0108	26.9205
$Formiga$	1	244	0.5891	6.6160
$Improvement$			85.31%	75.42%
$Dist\ dual$		1964	2.4040	34.2954
$Formiga$	2	1963	0.2495	8.1275
$Improvement$			89.62%	76.30%
$Dist\ dual$		13677	1.0213	20.9672
$Formiga$	3	13673	0.1160	7.1680
$Improvement$			88.64%	65.81%
$Dist\ dual$		42433	0.6230	24.0820
$Formiga$	4	42428	0.0778	8.1641
$Improvement$			87.52%	66.10%

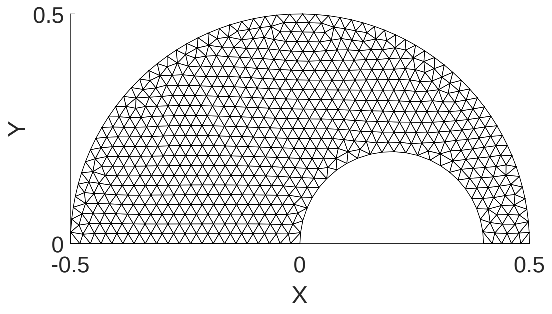
Table 4.7: Eccentric cylinder condensers - Polyhedral meshes warp angles.

To compute this problem Neumann boundary conditions are prescribed to the faces in the x axis to impose the symmetry and Dirichlet conditions are prescribed to both the inner and outer circles.

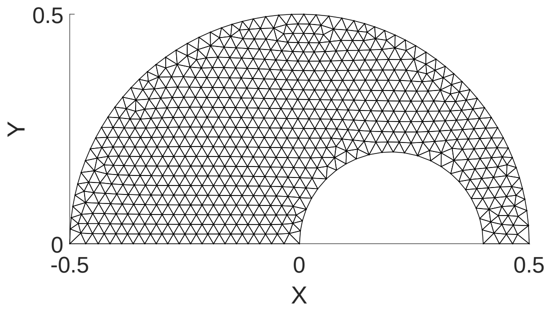
The results from the triangular meshes are in Table 4.8 and Fig. 4.6, and though similar to the last Section in the sense that the changes proposed improve accuracy for all four schemes, this case has some particularities of its own. The first major difference is the average improvement of average errors seen in schemes $D-OR$, $D-FLS$ and $D-GP$, that by far exceed the improvements seen in the previous example. If these outperformed the expectations the results from $D-NRML$ are underwhelming. Having no corrections for unstructured grids is expected that this scheme scores the least accurate results however it also stands to reason that improvements in mesh quality would result in the largest increase in accuracy of the bunch. This was not verified for this example that even though it scored the largest average reduction of maximum error, the reduction of average errors is smaller than other schemes and seems to drop with the number of cells despite not existing any reason from the warp angle point of view to do so.

Polyhedral meshes (Fig. 4.7 and Tab. 4.9) do not show this behavior with *Formiga*'s meshes improving by nearly an order of magnitude both the average and maximum errors in all meshes and numerical schemes.

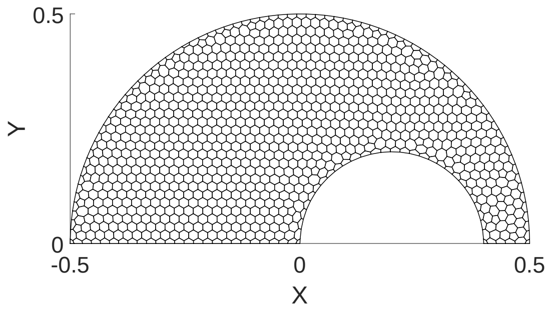
In this test case, the most accurate scheme for both mesh types proved to be $D-GP$. In Table 4.10 the same comparison between triangular and polyhedral meshes from last Section is presented and shows a much clear dominance of polyhedral meshes than before. Despite polyhedral meshes already being in the lead without any corrections the changes proposed in this work manage to extend their lead



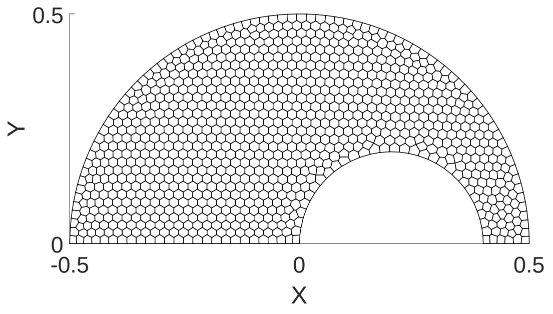
(a) Triangular mesh generated with *Distmesh*.



(b) Triangular mesh generated with *Formiga*.



(c) Polyhedral mesh generated with *Dist dual*.



(d) Polyhedral mesh generated with *Formiga*.

Figure 4.5: Eccentric cylinder condensers - Examples of the meshes used.

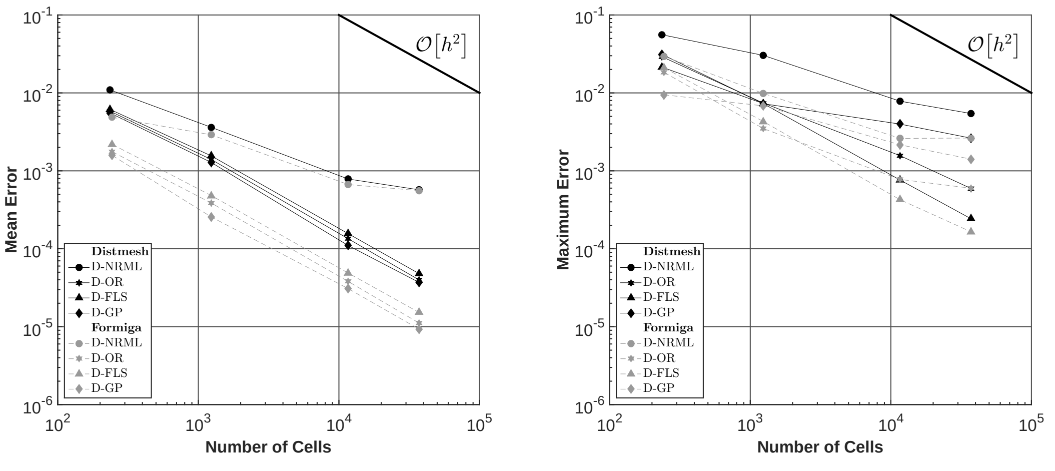
		$D-NRML$		$D-OR$		$D-FLS$		$D-GP$	
	$Mesh\,\#$	Avg	Max	Avg	Max	Avg	Max	Avg	Max
$Improvement$	1	55.42%	47.14%	69.77%	37.02%	64.36%	1.83%	71.43%	69.70%
$Improvement$	2	19.96%	67.55%	72.47%	53.06%	69.13%	41.27%	80.07%	6.28%
$Improvement$	3	15.1%	66.73%	71.72%	50.40%	68.79%	44.12%	72.02%	45.79%
$Improvement$	4	2.35%	51.54%	72.00%	−0.73%	68.28%	32.03%	74.89%	46.28%
$Average$		23.21%	58.24%	71.49%	34.94%	67.64%	29.81%	74.61%	42.01%

Table 4.8: Eccentric cylinder condensers - Triangular meshes numeric error improvement.

especially in the average and maximum error values of scheme $D-NRML$.

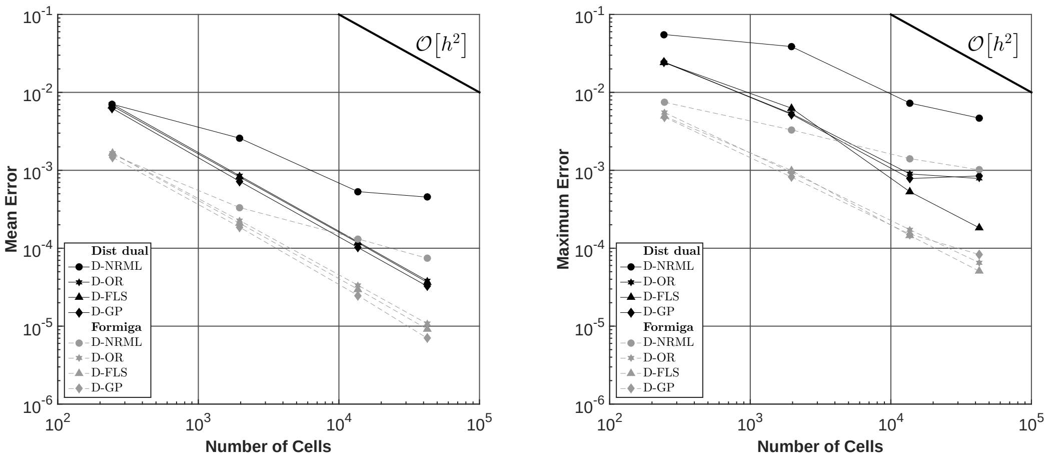
		$D-NRML$		$D-OR$		$D-FLS$		$D-GP$	
	$Mesh\,\#$	Avg	Max	Avg	Max	Avg	Max	Avg	Max
$Improvement$	1	77.66%	86.51%	76.51%	77.28%	75.41%	79.62%	76.51%	80.48%
$Improvement$	2	87.13%	91.47%	73.34%	82.40%	74.47%	84.13%	74.21%	84.33%
$Improvement$	3	75.27%	80.62%	72.32%	80.77%	74.70%	72.54%	75.76%	80.70%
$Improvement$	4	83.61%	78.31%	71.92%	91.67%	74.91%	72.22%	78.48%	90.18%
$Average$		80.92%	84.23%	73.52%	83.03%	74.87%	77.13%	76.24%	83.92%

Table 4.9: Eccentric cylinder condensers - Polyhedral meshes numeric error improvement.



(a) Average error. (b) Maximum error.

Figure 4.6: Eccentric cylinder condensers - Triangular meshes numerical error.



(a) Average error. (b) Maximum error.

Figure 4.7: Eccentric cylinder condensers - Polyhedral meshes numerical error.

	$D-NRML$		$D-OR$		$D-FLS$		$D-GP$	
	Avg	Max	Avg	Max	Avg	Max	Avg	Max
$Dist$	29.20%	−1.45%	7.91%	13.83%	21.60%	14.02%	12.52%	49.37%
$Formiga$	80.77%	62.12%	15.57%	77.46%	39.90%	72.08%	19.62%	81.09%

Table 4.10: Eccentric cylinder condensers - Triangular versus polyhedral meshes, numerical accuracy.

4.3 Convective-Diffusive Equation - Quarter annulus domain

Moving on from diffusive problems, this Section proposes the solution of the convection-diffusion equation in an annulus segment centered at $(0, 0)$, as seen in [45]. Given the nature of this problem both skewness and warp angle values are of great interest and as such are presented in Tables 4.11 and 4.12 for triangular and polyhedral meshes, respectively.

Over the previous Sections has been mentioned that it is common for polyhedral meshes to have higher maximum skewness values than triangular ones, that is also the case here. So this adds a new variable to compare, where polyhedral meshes had the advantage in what concerns to grid quality values, that also showed later in the numerical results. Besides this a closer analysis of the mentioned Tables reveals that the maximum warp angles of polyhedral meshes 3 and 4 are not quite to par with the first two putting them at a similar level to the triangular meshes of equivalent size. The fluctuations in maximum values happen as a consequence of the automatic mesh generation process where one does not cater directly to the improvement needs of the individual mesh.

			α_N $[deg]$		ψ	
	$Mesh\#$	$N.Cells$	Avg	Max	Avg	Max
$Distmesh$		503	4.5056	33.9281	0.0360	0.2383
$Formiga$	1	498	1.5174	18.1123	0.0322	0.2252
$Improvement$			66.32%	46.62%	10.43%	5.48%
$Distmesh$		1723	3.4162	32.7205	0.0205	0.3333
$Formiga$	2	1716	1.2006	17.7066	0.0176	0.1985
$Improvement$			64.86%	45.89%	14.09%	40.44%
$Distmesh$		12077	1.3896	32.0638	0.0073	0.3333
$Formiga$	3	12072	0.6823	17.6022	0.0061	0.1966
$Improvement$			50.90%	45.10%	16.63%	41.01%
$Distmesh$		53860	0.7136	32.0483	0.0035	0.3372
$Formiga$	4	53855	0.3902	18.9079	0.0030	0.1987
$Improvement$			45.32%	41.00%	15.41%	41.07%

Table 4.11: Quarter annulus - Triangular meshes quality metrics.

In Figure 4.8 some of the characteristics of *Formiga*'s meshes are easily identified like the treatment of the domain's boundaries, both in triangular meshes, where the change from one to two triangles in these regions is noticeable (Figs. 4.8(a) and 4.8(b)) and polyhedral meshes where one that confirms the corner cells are all quadrilaterals, Fig. 4.8(d).

Overall the average grid quality values still show polyhedral meshes ahead and pretty significant improvements from the changes proposed during this work that are reflected in *Formiga*'s meshes.

For this problem a constant diffusion coefficient of $\kappa = 0.01$ and an angular velocity of $u_\theta = \frac{1}{r}$ are set along with the Dirichlet boundary conditions on all the domain's limits except for the frontier parallel to the y axis where Neumann boundary conditions are applied. The solution for this example is given by Eqs. 4.3 and 4.4 in polar coordinates although actually solved in cartesian ones.

$$\phi(r,\theta) = \sin\left(\frac{\pi \ln r}{\ln 2}\right)\frac{s_1\exp(s_1\frac{\pi}{2} + s_2\theta) - s_2\exp(s_2\frac{\pi}{2} + s_1\theta)}{s_1\exp(s_1\frac{\pi}{2}) - s_2\exp(s_2\frac{\pi}{2})} \tag{4.3}$$

$$s_{1,2} = \frac{1 \pm \sqrt{1 + \left(\frac{2\pi\kappa}{\ln 2}\right)^2}}{2\kappa} \tag{4.4}$$

			α_N $[deg]$		ψ	
	$Mesh\,\#$	$N.Cells$	Avg	Max	Avg	Max
$Dist\ dual$		350	3.9515	23.9622	0.1023	1.1542
$Formiga$	1	349	0.3129	3.3118	0.0459	0.3937
$Improvement$			92.08%	86.18%	55.16%	65.89%
$Dist\ dual$		1723	2.3642	25.3238	0.0451	1.0572
$Formiga$	2	1725	0.1983	3.5399	0.0244	0.6988
$Improvement$			91.61%	86.02%	45.89%	33.90%
$Dist\ dual$		12151	1.0087	24.4224	0.0179	1.0619
$Formiga$	3	12145	0.3190	11.4282	0.0073	0.4420
$Improvement$			68.37%	53.21%	59.39%	58.37%
$Dist\ dual$		55636	0.5044	24.1743	0.0085	1.0257
$Formiga$	4	55631	0.1656	10.5087	0.0039	0.5020
$Improvement$			67.16%	56.53%	53.76%	51.06%

Table 4.12: Quarter annulus - Polyhedral meshes quality metrics.

Numerical results from triangular meshes are presented in Table 4.13 and Figure 4.9 and show, similarly to the previous Section a small improvement in the accuracy of the first family of schemes that scored by far the least accurate results, not being able to maintain the second order accuracy slope, in both the *Distmesh*'s and *Formiga*'s meshes. All three other families of schemes performed similarly with comparable reductions of average and maximum errors. So similarly that is not possible to discern from the presented information which was the most accurate, however, the ghost points family scored slightly better average error results than the rest, just edging out the family $D-OR,\ C-LIN-SK$.

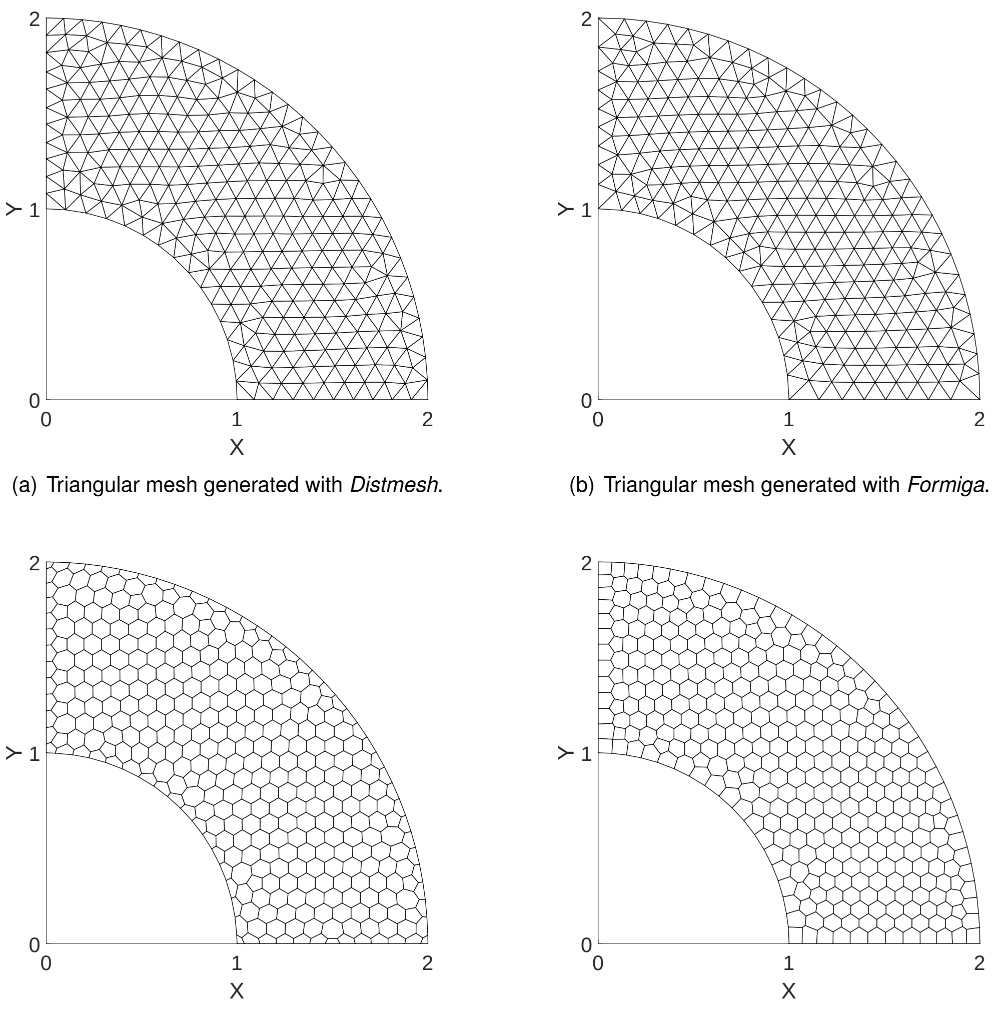
		$D-NRML,\ C-LIN$		$D-OR,\ C-LIN-SK$		$D-FLS,\ C-FLS$		$D-GP,\ C-GP$	
	$Mesh\,\#$	Avg	Max	Avg	Max	Avg	Max	Avg	Max
$Improvement$	1	5.17%	32.37%	20.69%	−4.03%	25.71%	−6.73%	17.84%	−7.76%
$Improvement$	2	5.22%	37.18%	36.14%	54.73%	43.67%	55.63%	33.10%	37.87%
$Improvement$	3	2.25%	29.11%	66.91%	−9.67%	70.04%	12.38%	65.89%	−18.16%
$Improvement$	4	10.07%	49.05%	70.33%	5.71%	69.82%	8.91%	70.49%	−1.38%
$Average$		5.68%	36.93%	48.52%	11.69%	52.31%	17.55%	46.83%	2.64%

Table 4.13: Quarter annulus - Triangular meshes numeric error improvement.

Polyhedral meshes on the other hand see *Formiga*'s meshes improve the average accuracy with all families of numerical schemes by roughly the same percentage, with the maximum error for the last three families being reduced by over an order of magnitude in the most refined meshes. The high maximum warp angle values mentioned before only seem to hinder the results of $D-NRML,\ C-LIN$ that by not having any grid corrections are, in theory, more susceptible to variations of the grid quality metrics.

No real debate as for what schemes were the most accurate for this example, with $D-OR,\ C-LIN-SK$ performing better, ahead of the ghost points family.

In this case polyhedral meshes always outperformed the triangular ones and, as before, benefited



(a) Triangular mesh generated with *Distmesh*. (b) Triangular mesh generated with *Formiga*.

(c) Polyhedral mesh generated with *Dist dual*. (d) Polyhedral mesh generated with *Formiga*.

Figure 4.8: Examples of meshes over the quarter annulus domain.

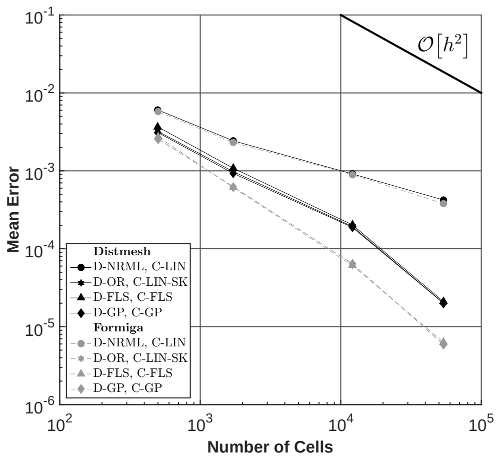
		$D-NRML,\ C-LIN$		$D-OR,\ C-LIN-SK$		$D-FLS,\ C-FLS$		$D-GP,\ C-GP$	
	$Mesh\,\#$	Avg	Max	Avg	Max	Avg	Max	Avg	Max
$Improvement$	1	38.61%	33.97%	−2.17%	68.93%	13.34%	66.33%	3.49%	69.32%
$Improvement$	2	77.00%	56.93%	53.42%	82.51%	51.15%	82.26%	59.32%	82.33%
$Improvement$	3	21.21%	51.37%	78.79%	92.00%	68.71%	91.03%	74.52%	91.67%
$Improvement$	4	25.50%	48.95%	68.25%	95.82%	66.06%	96.13%	71.39%	95.52%
$Average$		40.58%	47.80%	49.57%	84.81%	49.82%	83.94%	52.18%	84.71%

Table 4.14: Quarter annulus - Polyhedral meshes numeric error improvement.

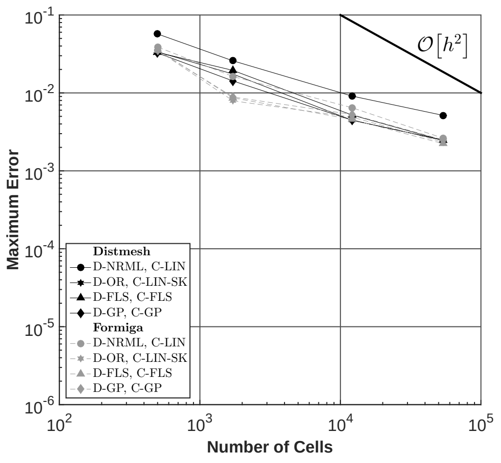
from the grid quality improvements made to further solidify the lead.

	$D-NRML,\ C-LIN$		$D-OR,\ C-LIN-SK$		$D-FLS,\ C-FLS$		$D-GP,\ C-GP$	
	Avg	Max	Avg	Max	Avg	Max	Avg	Max
$Dist$	13.10%	25.53%	50.13%	26.95%	33.62%	29.64%	38.81%	20.10%
$Formiga$	48.55%	56.81%	69.05%	89.55%	59.12%	89.27%	66.89%	88.46%

Table 4.15: Quarter annulus - Triangular versus polyhedral meshes, numerical accuracy.

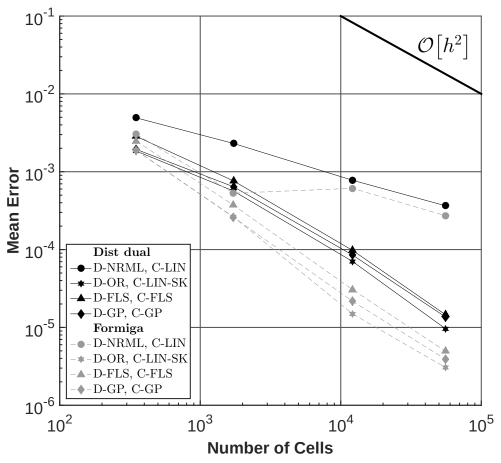


(a) Average error.

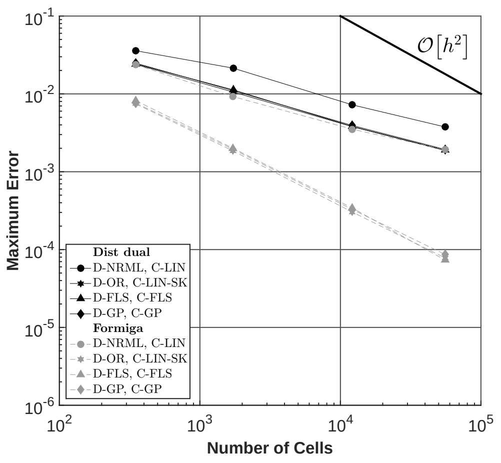


(b) Maximum error.

Figure 4.9: Quarter annulus - Triangular meshes numerical error.



(a) Average error.



(b) Maximum error.

Figure 4.10: Quarter annulus - Polyhedral meshes numerical error.

4.4 Taylor Couette Flow

The Taylor Couette flow describes the behaviour of a viscous fluid between two solid concentric cylinders at different angular speeds. For the purpose of this work only the fully developed, steady state of this flow is of interest as in these conditions and admitting two long enough cylinders the velocity field is known and given by Eq. 4.5. The constants $\Omega_o,\ \Omega_i,$ represent the angular velocity of the inner and outer cylinders, respectively, and $R_o,\ R_i$ their respective radius.

$$\mathbf{v}_\theta(r) = \frac{\Omega_o R^2{}_o - \Omega_i R^2{}_i}{R^2{}_o - R^2{}_i} r + \frac{(\Omega_i - \Omega_o) R^2{}_i R^2{}_o}{R^2{}_o - R^2{}_i} \frac{1}{r} \tag{4.5}$$

In the conditions described this problem may be reduced to 2D and the remaining boundary conditions set, $R_o = 1$, $R_i = 0.5$, $\Omega_i = 1$, $\Omega_o = 0$ and viscosity $\nu = 1$. Tables 4.16 and 4.17 have the quality metrics of all meshes used and their representation is showed in Figure 4.11.

Concerning triangular meshes the improvement values in the respective Table don't show anything out of the norm set in previous Sections, with significant improvements from *Formiga*. However it's worth noting that the maximum warp angles of the first two *Distmesh*'s grids are smaller than usually verified in previous examples. This has only one explanation, the absence of 'right-angled triangles', due to the use of only curvilinear boundaries in this case. This was one of the few geometries used in *Distmesh* that most often than not produced meshes without this particular topology. Fig. 4.11(a) confirms their absence and shows a *Distmesh* generated grid even more similar to *Formiga*'s than usual. The improvements made by *Formiga* in the maximum warp angle for these two meshes are only minor as no changes in connectivities are performed.

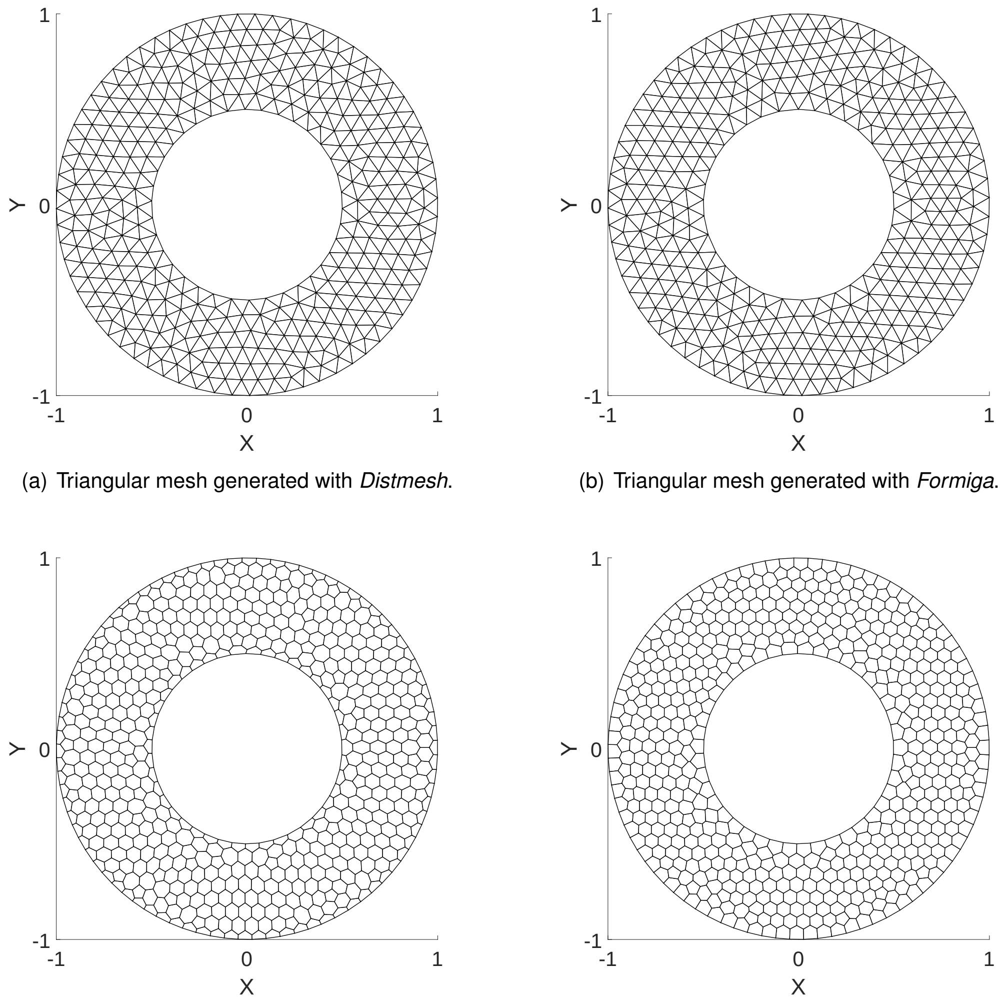
			α_N $[deg]$		ψ	
	$Mesh\#$	$N.Cells$	Avg	Max	Avg	Max
Distmesh		668	4.8828	18.5696	0.0519	0.2438
Formiga	1	668	2.3509	16.7082	0.0468	0.1912
Improvement			51.85%	10.02%	9.83%	21.59%
Distmesh		2481	3.7387	16.2806	0.0257	0.3078
Formiga	2	2481	1.8891	14.2556	0.0226	0.1929
Improvement			49.47%	12.44%	12.19%	37.33%
Distmesh		4185	3.2421	34.0978	0.0201	0.2809
Formiga	3	4182	1.6244	15.5195	0.0173	0.2083
Improvement			49.79%	54.49%	13.63%	25.84%
Distmesh		8326	2.5599	31.5046	0.0138	0.2799
Formiga	4	8324	1.4029	14.8328	0.0119	0.2068
Improvement			45.20%	52.92%	13.62%	26.12%

Table 4.16: Taylor Couette - Triangular meshes quality metrics.

			α_N $[deg]$		ψ	
	$Mesh\#$	$N.Cells$	Avg	Max	Avg	Max
Dist dual		556	4.6656	19.2378	0.1288	0.9327
Formiga	1	555	0.4518	2.5326	0.0718	0.4826
Improvement			90.32%	86.84%	44.23%	48.25%
Dist dual		2225	3.2250	22.3456	0.0642	1.1033
Formiga	2	2222	0.2539	5.4724	0.0354	0.6489
Improvement			92.13%	75.51%	44.89%	41.18%
Dist dual		4349	2.5921	20.6570	0.0455	1.0771
Formiga	3	4351	0.1862	4.7378	0.0244	0.4664
Improvement			92.82%	77.06%	46.44%	56.69%
Dist dual		8395	2.0644	23.1212	0.0327	1.0613
Formiga	4	8395	0.1701	5.1036	0.0187	0.7047
Improvement			91.76%	77.93%	42.89%	33.60%

Table 4.17: Taylor Couette - Polyhedral meshes quality metrics.

Formiga's polyhedral meshes show, like in previous Sections, large reductions in the warp angle, skewness, and sizable volume increase of boundary cells.



(a) Triangular mesh generated with *Distmesh*.

(b) Triangular mesh generated with *Formiga*.

(c) Polyhedral mesh generated with *Dist dual*.

(d) Polyhedral mesh generated with *Formiga*.

Figure 4.11: Examples of meshes generated over the annulus domain.

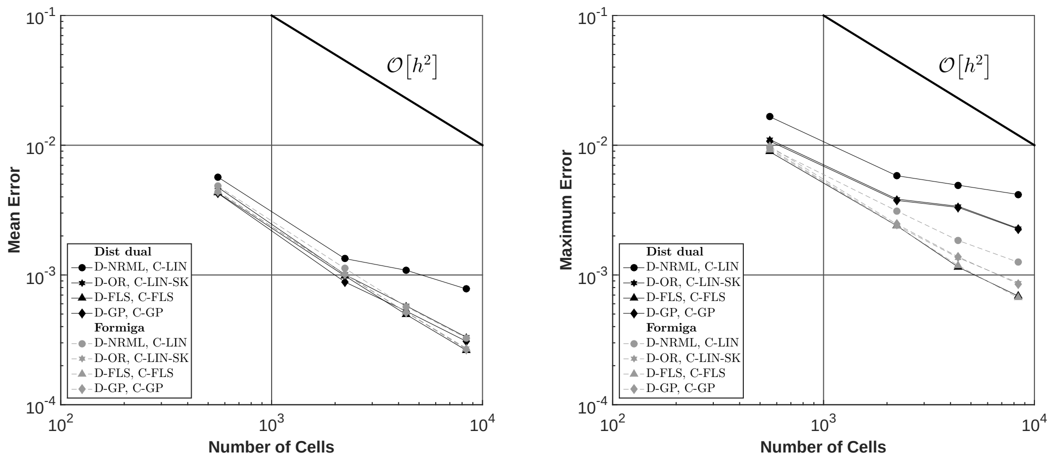
To solve this problem the SIMPLE [46] algorithm is used along with the discretization schemes of previous Sections for the diffusive and convective terms and a cell centered gradient scheme based on Weighted Least-Squares method. [32].

		$D-NRML,\ C-LIN$		$D-OR,\ C-LIN-SK$		$D-FLS,\ C-FLS$		$D-GP,\ C-GP$	
	$Mesh\,\#$	Avg	Max	Avg	Max	Avg	Max	Avg	Max
$Improvement$	1	14.89%	42.56%	7.36%	14.89%	-2.18%	-3.10%	-3.11%	9.00%
$Improvement$	2	15.89%	46.45%	-0.69%	37.36%	-3.48%	0.51%	-14.74%	34.29%
$Improvement$	3	47.57%	62.51%	10.58%	59.84%	-4.52%	-2.83%	2.01%	58.41%
$Improvement$	4	58.17%	69.96%	17.90%	61.99%	-2.87%	2.85%	13.42%	62.24%
$Average$		34.13%	55.37%	8.79%	43.52%	-3.26%	-0.64%	-0.60%	40.98%

Table 4.18: Taylor Couette - Polyhedral meshes numeric error improvement.

Unlike previous Sections the results on polyhedral meshes shall be addressed first. Table 4.18

and Figure 4.12 show large improvements on the maximum error in all but the Least Squares family of schemes. This family did not show any improvement in the accuracy, this may be attributed to the already low error magnitude before the smoothing algorithm. Improvements on the average error are only sensible on the family of schemes with no corrections and $D - OR,\ C - LIN - SK$, with the first one improved by a larger margin.



(a) Average error.

(b) Maximum error.

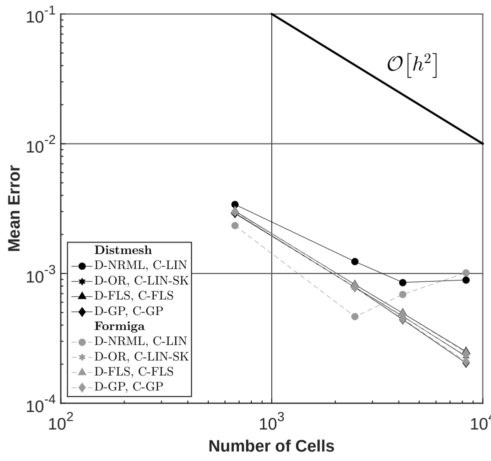
Figure 4.12: Taylor Couette - Polyhedral meshes numerical error.

The simulation results on the triangular meshes are spread over Table 4.19 and Figure 4.13, however before addressing those it is important to detail the process to achieve them. When trying to obtain results with the mentioned cell centered gradient scheme, the numeric error obtained with all discretization schemes increased with mesh size, independently of mesh generator. Given that the same gradient scheme had already been used, with success, in the polyhedral meshes a programming error is excluded and the fault has to rest with the cell shape of the meshes. Also note that the gradient scheme affects the pressure gradient computed in the momentum equations.

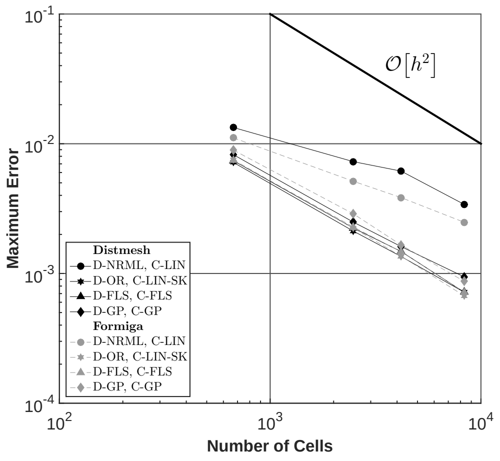
		$D - NRML,\ C - LIN$		$D - OR,\ C - LIN - SK$		$D - FLS,\ C - FLS$		$D - GP,\ C - GP$	
	$Mesh\,\#$	Avg	Max	Avg	Max	Avg	Max	Avg	Max
$Improvement$	1	31.25%	16.76%	−1.10%	−4.67%	−0.82%	−0.51%	−2.12%	−8.44%
$Improvement$	2	62.36%	29.42%	0.71%	−5.72%	0.70%	−1.97%	0.12%	−15.88%
$Improvement$	3	19.05%	37.56%	0.56%	−0.44%	0.65%	0.48%	−1.20%	−2.77%
$Improvement$	4	−13.82%	27.21%	−0.98%	6.21%	1.43%	0.08%	−2.45%	7.54%
$Average$		24.71%	27.74%	−0.20%	−1.16%	0.49%	−0.48%	−1.41%	−4.89%

Table 4.19: Taylor Couette - Triangular meshes numeric error improvement.

Determined to achieve results with triangular meshes the Gauss Method [32] was chosen as new cell centered gradient scheme and the results obtained are the ones in the already mentioned Table and Figure. What these show is that the improvements in grid quality made with *Formiga* produced no effect in the numerical errors of all three schemes that incorporate grid quality corrections, that also managed to maintain the slope error decay. The results of schemes $D - NRML,\ C - LIN$ are not up to this standard. For smaller number of cells *Formiga*'s grids score respectable improvements over the numerical accuracy of *Distmesh*'s grids, and achieve average errors inferior to all other schemes. Then, for larger grids, they show the same behavior seen with the previous gradient scheme.



(a) Average error.



(b) Maximum error.

Figure 4.13: Taylor Couette - Triangular meshes numerical error.

Despite all the difficulties in getting results, triangular meshes managed to outperform the polyhedral ones. Upon seeing this the polyhedral meshes simulations were repeated using the Gauss Method to verify if it was in any shape the responsible though, this was not found to be the case with the results being borderline the same as in Table 4.18. From Table 4.20 one may confirm that triangular meshes scored, indeed, better results but it's also fair to say that *Formiga*'s triangular and polyhedral meshes performed much more evenly than *Distmesh*'s and its dual grid. Needless to say the values presented for the first family of schemes aren't representative of much and not considered for the previous statements.

	$D-NRML,\ C-LIN$		$D-OR,\ C-LIN-SK$		$D-FLS,\ C-FLS$		$D-GP,\ C-GP$	
	Avg	Max	Avg	Max	Avg	Max	Avg	Max
$Dist$	-22.59%	-1.71%	-40.07%	-125.93%	-16.67%	-0.22%	-32.33%	-81.69%
$Formiga$	-40.93%	38.73%	-27.25%	-15.42%	-20.86%	-0.33%	-30.11%	5.94%

Table 4.20: Taylor Couette - Triangular versus polyhedral meshes, numerical accuracy.

Chapter 5

Conclusions

A new mesh generator capable of producing high quality triangular and polyhedral meshes was presented. Its main characteristics include the identification and elimination of poor quality cells in triangular meshes, a new method to eliminate concave cells in polyhedral meshes and a smoothing algorithm based in forces proportional to grid quality parameters specific to the Finite Volume Method (FVM).

This smoothing algorithm shows good ability to improve grid quality however it's fair to say that this algorithm is more effective in polyhedral meshes as it obtains larger improvements, most of the time, than in triangular ones. In addition improvements on triangular meshes are dependent on the elimination of some prejudicial connectivities.

The implementation of the smoothing algorithm in three dimensions showed solid improvements, although not at the same level as in 2D. These improvements suggest this algorithm has potential however more work is required, mainly in the connectivities correction.

The presented meshes, generated over several geometries prove the robustness and above all the algorithm's ability to produce high quality grids. Comparisons with several other available mesh generators see *Formiga*'s meshes scoring better grid quality metrics independently of the cell type used.

The meshes generated with the routines presented in this work prove to be more accurate than grids with worse quality metrics on the numerical solutions tested. Improving both the average and maximum error in most cases, independently of the family of discrete schemes. This performance on numerical simulations ultimately validates both the choice of grid quality metrics and the algorithm developed in this work.

As a by product of the last mentioned analysis it is possible to judge the accuracy of the discretization schemes used. The first conclusion, that should come as no surprise, is that corrections for unstructured meshes are indeed necessary as the family of schemes without any corrections is the worst performing one, in most cases, they are very susceptible to the quality of the mesh used. The remaining schemes used performed similarly amongst each other being able to sustain the order of convergence for each test case in *Formiga*'s meshes but not always in the case of the other selected meshes. This in addition, to the improvements in average and maximum errors verified with *Formiga*'s grids prove that despite the corrections for grid quality in the discretization schemes the development of better mesh improvement and generation algorithms is still of the utmost importance. The ghost points family and the quality metric based scheme split amongst them the best average error results while the Least Squares family took, in most cases, the lowest maximum error value.

Finally on the comparison between triangular and polyhedral meshes the results presented confirm that in most situations polyhedral grids are more accurate than triangular ones for a similar number of cells. From a grid quality perspective the skewness values are usually higher in polyhedral meshes, in it's average values but specially in the maximum ones while the warp angle results are lower on polyhedral meshes.

5.1 Future Work

The following steps in the life of *Formiga* may for the 2D code consist of a conversion from MATLAB to C, this would not only allow for faster execution times but also further integration within SOL that is also in this language. In doing so an avenue for using *Formiga*'s output with previous LASEF work concerning the study of adaptive meshes [47] is also opened, by relating the error estimators with the base size distribution

The three dimensional version needs, on the other hand, to undergo a large number of improvements before being considered a finished product. Some of these improvements concern only the current generator while others also concern the 3D polyhedral generation in it's entirety. Under the first category fall making the routine implemented capable of creating meshes in any geometry and not only in model's with limited number of boundaries, expand the algorithm for concave cell's elimination to 3D and implement the changes in connectivities proposed for triangular and polyhedral meshes in 2D, for 3D. On the second category falls the choice of primal mesh's central points to use as dual vertices. At this moment, no 3D polyhedral mesh generator guarantees that all its faces are planar. The quality metrics used in this work, being measured on each of the cell's faces, would be more reliable if all faces were planar or almost planar. The presented smoothing algorithm would also benefit from planar faces because of the more accurate grid quality estimations but also because this would represent one less item to improve.

Other line of work, will be to extend the current code for high-order meshing. Despite being a field of research significantly developed for triangular meshes, it has not yet been applied for unstructured polyhedral grids.

Bibliography

[1] American Institute of Aeronautics and Astronautics, Reston, VA. *Guide for the verification and validation of computational fluid dynamics simulations*, 1998. AIAA-G-077-1998.

[2] William L Oberkampf and Timothy G Trucano. Verification and validation in computational fluid dynamics. *Progress in Aerospace Sciences*, 38(3):209–272, 2002.

[3] William L Oberkampf and Christopher J Roy. *Verification and validation in scientific computing*. Cambridge University Press, 2010.

[4] Diogo Matias Coimbra Martins. On the suppression of spurious pressure oscillations in immersed boundary methods with unstructured grids. *Master Thesis, Instituto Superior Técnico, Universidade de Lisboa*, 2016.

[5] J. P. Magalhães. An adaptive framework for the numerical simulation of environmental flows in complex geometries. *PhD Thesis, Instituto Superior Técnico, Universidade Técnica de Lisboa*, 2011.

[6] Steven J Owen. A survey of unstructured mesh generation technology. In *IMR*, pages 239–267, 1998.

[7] Pascal Jean Frey and Paul L George. *Mesh generation: application to finite elements*. Wiley Online Library, 2000.

[8] Guojun Yu, Bo Yu, Shuyu Sun, and Wen-Quan Tao. Comparative study on triangular and quadrilateral meshes by a finite-volume method with a central difference scheme. *Numerical Heat Transfer, Part B: Fundamentals*, 62(4):243–263, 2012.

[9] Feifei Shang, Yangke Gan, and Yufei Guo. Hexahedral mesh generation via constrained quadrilateralization. *PloS one*, 12(5):e0177603, 2017.

[10] Steven J. Owen, Matthew L. Staten, Scott A. Canann, and Sunil Saigal. Q-morph: an indirect approach to advancing front quad meshing. *International Journal for Numerical Methods in Engineering*, 44(9):1317–1340, 1999.

[11] Yan Liu, HL Xing, and Zhenqun Guan. An indirect approach for automatic generation of quadrilateral meshes with arbitrary line constraints. *International Journal for Numerical Methods in Engineering*, 87(9):906–922, 2011.

[12] Cameron Talischi, Glaucio H. Paulino, Anderson Pereira, and Ivan F. M. Menezes. Polygonal finite elements for topology optimization: A unifying paradigm. *International Journal for Numerical Methods in Engineering*, 82(6):671–698.

[13] Cameron Talischi, Glaucio H. Paulino, Anderson Pereira, and Ivan F. M. Menezes. Polymesher: a general-purpose mesh generator for polygonal elements written in matlab. *Structural and Multidisciplinary Optimization*, 45(3):309–328, 2012.

[14] Yan Liu Yu-xin Jie, Xu-dong Fu. Mesh generation for FEM based on centroidal voronoi tessellations. *Mathematics and Computers in Simulation*, 97:68 – 79, 2014.

[15] Rao V Garimella, Jibum Kim, and Markus Berndt. Polyhedral mesh generation and optimization for non-manifold domains. In *Proceedings of the 22nd International Meshing Roundtable*, pages 313–330. Springer, 2014.

[16] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS)*, 22(4):469–483, 1996.

[17] Chris Rycroft. Voro++: A three-dimensional voronoi cell library in c++. 2009.

[18] F. Juretić. *Error Analysis in Finite Volume CFD*. PhD thesis, PhD Thesis, Imperial College London, 2004.

[19] G. Balafas. Polyhedral mesh generation for CFD-analysis of complex structures. *Master Thesis, TUM, Germany*, 2013.

[20] F. Juretić and A. D. Gosman. Error analysis of the finite-volume method with respect to mesh type. *Numerical heat transfer, part B: fundamentals*, 57(6):414–439, 2010.

[21] M. Perić. Simulation of flows in complex geometries:new meshing and solution methods. *NAFEMS Seminar*, May 3-4 2004. Germany.

[22] Martin Spiegel, Thomas Redel, Y. Jonathan Zhang, Tobias Struffert, Joachim Hornegger, Robert G. Grossman, Arnd Doerfler, and Christof Karmonik. Tetrahedral and polyhedral mesh evaluation for cerebral hemodynamic simulation—a comparison. In *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, pages 2787–2790. IEEE, 2009.

[23] Man Young Kim, Seung Wook Baek, and Il Seouk Park. Evaluation of the finite-volume solutions of radiative heat transfer in a complex two-dimensional enclosure with unstructured polygonal meshes. *Numerical Heat Transfer, Part B: Fundamentals*, 54(2):116–137, 2008.

[24] Jibum Kim and Jaeyong Chung. Untangling polygonal and polyhedral meshes via mesh optimization. *Engineering with Computers*, 31(3):617–629, Jul 2015.

[25] Jibum Kim, Myeonggyu Shin, and Woochul Kang. A derivative-free mesh optimization algorithm for mesh quality improvement and untangling. *Mathematical Problems in Engineering*, 2015, 2015.

[26] Michael L. Brewer, Lori Freitag Diachin, Patrick M. Knupp, Thomas Leurent, and Darryl J. Melander. The Mesquite mesh quality improvement toolkit. In *IMR*, 2003.

[27] David A Field. Laplacian smoothing and delaunay triangulations. *International Journal for Numerical Methods in Biomedical Engineering*, 4(6):709–712, 1988.

[28] Zhijian Chen, Joseph R. Tristano, and Wa Kwok. Construction of an objective function for optimization-based smoothing. *Engineering with Computers*, 20(3):184–192, Sep 2004.

[29] Per-Olof Persson and Gilbert Strang. A simple mesh generator in matlab. *SIAM review*, 46(2):329–345, 2004.

[30] Hrvoje Jasak. *Error analysis and estimation for finite volume method with applications to fluid flow*. PhD thesis, PhD Thesis,Imperial College, 1996.

[31] F. P. Beer. *Vector Mechanics for Engineers: Statics and dynamics*. McGraw-Hill Companies, 2009.

[32] Duarte M. S. Albuquerque. Numerical computation of incompressible flows on adaptive and unstructured grids. *PhD Thesis, Instituto Superior Técnico, Universidade de Lisboa*, 2013.

[33] André Luís de Almeida Leite. Structured and unstructured finite volume calculations of a laminar and a turbulent axisymmetric jet flow. *Master Thesis, Instituto Superior Técnico, Universidade de Lisboa*, 2014.

[34] J. Ferziger and M. Peric. *Computational methods for fluid dynamics*. Springer-Verlag, 2nd edition, 1999.

[35] Per-Olof Persson. *Mesh generation for implicit geometries*. PhD thesis, Massachusetts Institute of Technology, 2005.

[36] Hoa Nguyen, John Burkardt, Max Gunzburger, Lili Ju, and Yuki Saka. Constrained cvt meshes and a comparison of triangular mesh generators. *Computational Geometry*, 42(1):1 – 19, 2009.

[37] Jonas Koko. A matlab mesh generator for the two-dimensional finite element method. *Applied Mathematics and Computation*, 250:650 – 664, 2015.

[38] Christophe Geuzaine and Jean-François Remacle. Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331.

[39] Jonathan Richard Shewchuk. Triangle: Engineering a 2D quality mesh generator and delaunay triangulator. In *Applied computational geometry towards geometric engineering*, pages 203–222. Springer, 1996.

[40] R Timothy Marler and Jasbir S Arora. Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26(6):369–395, 2004.

[41] Sang Yong Lee. Polyhedral mesh generation and a treatise on concave geometrical edges. *Procedia Engineering*, 124:174–186, 2015.

[42] Iain S Duff. Computer solution of large sparse positive definite systems (alan george and joseph w. liu. *SIAM Review*, 26(2):289–291, 1984.

[43] Ricardo Costa, Stéphane Clain, Raphaël Loubère, and Gaspar J. Machado. Very high-order accurate finite volume scheme on curved boundaries for the two-dimensional steady-state convection–diffusion equation with dirichlet condition. *Applied Mathematical Modelling*, 54:752 – 767, 2018.

[44] Triangle's instructions. `https://www.cs.cmu.edu/~quake/triangle.help.html`. Accessed: 01-04-2018.

[45] Carl Ollivier-Gooch and Michael Van Altena. A high-order-accurate unstructured mesh finite-volume scheme for the advection–diffusion equation. *Journal of Computational Physics*, 181(2):729 – 752, 2002.

[46] A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. *International Journal of Heat and Mass Transfer*, 15(10):1787 – 1806, 1972.

[47] Duarte M. S. Albuquerque, José M. C. Pereira, and José C. F. Pereira. Residual least-squares error estimate for unstructured h-adaptive meshes. *Numerical Heat Transfer, Part B: Fundamentals*, 67(3):187–210, 2015.