POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Department of Aerospace Science and Technology
Doctoral Programme in Aeronautical Engineering

Development of Time-Accurate CFD Methodologies for Dynamic Mesh Handling in Reactive Flow Computations

Doctoral Dissertation of:
Daniel Costero Valero
ID: 10782573
daniel.costero@polimi.it

Advisor:
Prof. Federico Piscaglia

Tutor:
Prof. Paolo Bettini

The Chair of the Doctoral Program:
Prof. Pierangelo Masarati

Academic Year: 2022-23

to my family

Acknowledgments

I wish to extend my heartfelt gratitude to my advisor, Prof. Federico Piscaglia, for his invaluable guidance and unwavering support, which played a crucial role in the successful completion of my thesis. His expert advice and constructive feedback significantly influenced the trajectory of my research.

I am also thankful to my peers at PoliMi, whose assistance and insights throughout this journey have been indispensable. Our discussions and their input enriched my understanding of the topic and offered fresh viewpoints.

I extend my appreciation to the colleagues at AVL in Graz, who welcomed me with open arms from day one and fueled my passion for my work.

A special thank you goes to Raf, Rafa, and Marija, whose consistent support and encouragement have been a pillar of strength over the years. Your emotional support has been priceless and has not gone unnoticed; I am profoundly grateful.

Finally, I cannot express enough gratitude to my family, whose unwavering love and encouragement have been my source of strength and inspiration, pushing me to strive for excellence in this endeavor.

Abstract

The field of Computational Fluid Dynamics has undergone rapid evolution over the last few decades, and it remains a critical tool for engineers and researchers in the present for simulating and analyzing fluid flows across various applications, including aerodynamics, hydrodynamics, biomedical engineering, and environmental science. Despite recent advancements in computing technology, computational time remains a significant bottleneck in transient simulations involving complex phenomena, as very small time steps are required for accurate results. To address this challenge, the use of high-order temporal schemes represents a promising approach to reduce computational time as it allows the usage of larger time steps. This is particularly important in problems with dynamic grids and large boundary movements, where mesh movement can be a limiting factor. In particular, a change in the topology of the grid imposes additional challenges, as introducing the grid velocity in the transport terms of the governing equations is insufficient to ensure conservativeness.

In this thesis, a methodology to use second order time differencing schemes in the presence of topology changes that change the number of cells is presented. In particular, the Second Order Backward Euler and Crank Nicolson schemes are considered. Three numerical experiments are provided to illustrate the effectiveness of the presented methodology. First a one-dimensional Uniformly Accelerated Piston case is considered, as it presents an analytical solution. Then, a three-dimensional lid-driven cavity test with a moving region of cells is analyzed and, finally, the methodology is tested on an industrial configuration.

A possible application is the simulation of in-cylinder flows in Dual-Fuel Internal Combustion Engines (DFICE), where a higher time accuracy may improve the precision of the fuel-air mixing process. The simulation of DFICE also requires accurate methods to simulate spray injection. For this reason, the final section of the thesis is devoted to the implementation of spray models suitable for dual-fuel combustion. A hybrid spray model that combines the Huh-Gosman breakup model for primary atomization of the liquid core near the injector and the KHRT model for secondary breakup is implemented and extended to work in multiple multi-hole injector simulations. The spray model is combined to a dynamic flow solver able to handle moving overset grids. Tests on a benchmark dynamic test-case are proposed. Results obtained with different mesh motion techniques are compared and discussed.

Contents

1 Introduction

Many interesting scientific and engineering problems in Computational Fluid Dynamics involve the coupling of high-speed fluid flows with body-fitted meshes with moving and possibly deforming boundaries [8, 12, 26, 59]. With large displacements, the flow problem is often formulated in an Arbitrary Lagrangian–Eulerian (ALE) scheme [63, 62, 28, 29] and discretized on a moving grid. An ALE integrator is constructed by combining the same time integrator adopted in static grids together with a procedure to include the velocity of the moving grid points. In Finite Volume (FV) Eulerian solvers based on the colocated grid arrangement, the transposition in time-varying coordinates of the conservation equations is achieved by substituting all advection velocities by their relative (to grid movement) counterparts [22]. Despite the ALE method is applied to problems with prescribed boundary motion and large deformations [5, 33, 32, 45], it is well know that it does not necessarily preserve the order of time-accuracy of its fixed counterpart. To make the rezone calculation conservative of mass, momentum, and energy, the ALE method requires that an additional constraint is fulfilled, namely the Geometric Conservation Law (GCL). This was first discovered by Trulio [63, 62] who applied it to solve one-dimensional problems by a Finite Difference method. In [15], the GCL was applied to the Finite Volume method, while its application in arbitrarily moving geometries was presented in [16]. A mathematical analysis has demonstrated the role of the GCL for its time-space accuracy order [25, 47], while in [21] it is proved that the Discrete GCL is a sufficient condition for some schemes to satisfy the maximum principle for passive species. The failure to satisfy the Discretised Geometric Conservation Law (DGCL) on moving meshes introduces errors in the form of artificial mass sources [15]. In [24, 20], it is shown that satisfaction of the DGCL is a necessary condition for any temporal discretisation scheme to preserve on moving grids the non-linear stability properties of its fixed-grid counterpart but also that satisfaction of the DGCL is not a sufficient condition for a temporal discretisation scheme to preserve its order of time-accuracy on moving grids, as established on fixed grids [64]. When mesh topology modifications are combined with ALE schemes the fullfillment of the GCL is not trivial: grid points are added or removed, and the mesh resolution and connectivity dynamically change to accommodate large prescribed boundary deformations (re-zone phase). Additionally, the solution interpolation (remap) must be conservative, accurate and must preserve the monotonicity

of the solution. Topology changes may involve the addition/deletion of cell faces without affecting the number of the Control Volumes (CV) (as it happens when different mesh regions are attached/detached through non-conformal interfaces [53]) or it can also modify the number of CVs, as in adaptive methods, namely dynamic cell layering [45, 54], Adaptive Mesh Refinement (AMR) and remeshing. In a broad sense, the simplest example of topology change is the remeshing [6], that consists of a remap of the fluid-dynamic solution onto a different mesh at prescribed times, when the mesh quality is assumed to become critical. This approach is simple to perform, because the solver solution is decoupled by the mesh generation, that is usually done during the pre-processing; it may become quite unconvenient if several grids must be generated to prevent invalid elements from appearing, because it can be very costly. In general, any local re-meshing or error-driven adaptive refinement may be considered as an example of topological change, that implies to recompute the mesh connectivity. The overhead due to topology changes is strictly implementation-dependent and it is never negligible. When adaptive methods are applied on deforming grids, the solution transfer between meshes with different topology may be non-trivial.

The extension of ALE schemes to variable topology grids has been investigated in [38, 37], where the edge swapping technique has been exploited to modify the grid without changing the number of grid nodes. In [26, 39, 4], methods based on explicit interpolation of the solution from the old to the new grid is proposed for ALE schemes used with grid connectivity changes. This may originate problems in enforcing conservativeness and monotonicity and complicate the implementation of multi-step time integration algorithms [66]. An alternative strategy consists of interpreting the insertion or deletion of a new element as a series of fictitious continuous deformations of the finite volumes associated to the nodes involved in the modification. When applied to three-dimensional problems in the Finite Volume Framework, this approach was often labeled as inflation layer meshing, as it is done in [5, 32]. This is possible to achieve by a repair step after the re-zone phase where mass is re-distributed between neighboring cells to ensure local bound preservation [7, 34] or, if first order time-accurate methods are used, by considering the velocity from the newly added faces to sweep new volumes during the re-zone phase [34, 40, 23, 55]. No studies are available about the fulfillment of the DGCL with the ALE scheme when second time-accurate implicit discretization methods are applied with topology changes that involve a variation in the number of cells in the dynamic grid [52].

Motivation of this research

The research described in this document is motivated by the need to improve the temporal accuracy of the solution in simulations where moving meshes are combined with dynamic addition

or deletion of cell volumes, as it happens when adaptive mesh refinement (AMR) or dynamic cell layering are triggered. Examples of applications to CFD cases are the calculation of the aerodynamics of moving wings, the calculation of moving pistons in rapid compressing machines, or the simulation of hull hydrodynamics. Increasing the temporal accuracy is extremely important in cases where mixing is present, as small errors in the solution will propagate during the simulation. One major application is the simulation of dual-fuel engines, which require advanced hybrid spray models that must be solved on a static grid to achieve a high accuracy. Mesh motion based on topology changes or overset grids are two techniques that can provide a static grid locally for the region where the spray plume is formed, while achieving a high temporal accuracy.

Goals and highlights

The goal of the present work is to develop an efficient methodology to achieve second-order time accuracy when new volumes are added or deleted into a dynamic grid. In particular, the second-order backward Euler (SOBE) and the Crank–Nicolson (CN) schemes are considered. Numerical analysis are performed on the uniformly accelerated piston, for which an analytical solution is available, a three-dimensional cavity case and an industrial configuration of an Internal Combustion Engine (ICE). Comparisons of the proposed work with other mesh moving techniques that do not involve topology changes are performed to illustrate the accuracy and conservativeness of the methodology. Aiming at the simulation of dual-fuel engines, an advanced hybrid spray model is implemented and validated in order to take advantage of the increased accuracy of the dynamic mesh handling. The code is implemented as an open-source C++ code in the OpenFOAM® technology.

Thesis structure

The content of this work is organized as follows. In Chapter 1, an introduction to the problem has been presented. Chapter 2 presents a description of the Finite Volume discretization, as well as a review of the different errors introduced in the solution during the discretization. The different existing mesh configurations are also reviewed, indicating the individual sources of error they introduce. The formulation of the governing equations of the Arbitrary Lagrangian-Eulerian system is reviewed in Chapter 3, followed by an explanation of how to handle topology changes in dynamic grids in Chapter 4. In particular, the proposed methodology to achieve second order temporal accuracy is explained in Sec. 4.4. Chapter 5 presents the results of the methodology applied to three test cases: Uniformly Accelerated Piston (Sec. 5.1), lid-driven cavity (Sec. 5.2) and industrial configuration (Sec. 5.3). The hybrid Lagrangian spray model

is introduced in Chapter 6 along with its coupling with dynamic overset grids. Finally, the conclusions are drawn in Chapter 7. The numerical tool employed in the tests is included in a set of dynamic C++ libraries which have been used in combination with the OpenFOAM® FV code as released by the OpenFOAM Foundation in the development version [61].

2 | Finite Volume Discretization

Simulating fluid dynamics is a highly challenging task, given that it necessitates solving a set of nonlinear partial differential equations. Furthermore, the phenomena at play span a broad range of physical scales, demanding the use of complicated models. In this section, the different errors introduced to solve fluid dynamics problems are introduced, with special attention to the discretization errors. The domain is decomposed in small polyhedral cells on which the equations are solved. Different meshes introduce different simplifications of the domain and therefore introduce further errors in the solution. Different mesh configurations can be used trying to maximize the accuracy of the results, while keeping an acceptable computational time. Each mesh type has a different set of characteristics that might be more suitable for a particular problem. If more accuracy is required only locally, in some concrete parts of the domain, mesh refinement can also be introduced.

2.1. Introduction

The performance and functionality of many products, from small household appliances to complex aerospace systems, are not solely dependent on their structural properties. The characteristics of heat transfer, fluid flow, and even fluid-solid interaction can significantly impact their design and performance. Therefore, incorporating these fluid dynamics phenomena into numerical simulations is essential for optimizing their design and enhancing their functionality. However, the simulation of fluid dynamics problems is challenging as it involves solving a set of coupled nonlinear partial differential equations. In addition, the physical scales involved in the different phenomena vary very widely, requiring complicated models. As there is no analytical solution for these problems, numerical methods have to be used instead, introducing different types of errors in the solution.

In the initial stages, a *modeling error* is introduced due to the utilization of simplified mathematical equations to represent the fluid flow, which may not fully encapsulate the complexity of the actual physical phenomena. The assumptions and approximations made during the mod-

eling process introduce inaccuracies in the subsequent simulation results. Modeling errors can occur at different stages of the CFD simulation, including fluid properties, boundary conditions or turbulence models. For example, it is a common practice to neglect gravitational and electromagnetic forces or to assume that walls are adiabatic, turbulence isotropic and the fluid Newtonian. These simplifications are made to facilitate the solution of the resultant system of non-linear, partial differential equations, albeit at the cost of simulation precision.

In order to solve the system of non-linear PDEs used to model the physical phenomena, approximate numerical solutions are needed, as analytical solutions only exist for a very limited number of very simple problems. The difference between this potential analytical solution and the exact solution of the algebraic system of equations that results from the discretization of the equations on a given grid is called the *discretization error*. Temporal and spatial discretization will be responsible for this error, as well as the numerical schemes used in the approximation of the derivative terms appearing in the equations.

The resulting algebraic system of equations is usually solved using iterative methods. The difference between the exact solution of the algebraic system of equations and the solution obtained with iterative methods constitutes the *convergence error*. This error can be reduced by running a sufficiently large number of iterations.

Nowadays, the modeling error is very small as extremely accurate mathematical models have been developed, and even tuned with experimental constants, to ensure that all the physical phenomena is properly captured. In addition, with the computational power of modern computers, the convergence error can be reduced substantially. Therefore, to improve the results of CFD simulations, reducing the discretization error emerges as the most promising approach.

To construct the algebraic system of equations, a mesh (or grid) is created to discretize the continuous domain in a finite number of elements in which the exact solution of the equations is computed. A temporal discretization is also introduced in the case of transient simulations. In addition, a numerical scheme to treat the derivative terms is also needed, establishing a polynomial variation of the solution from one cell to another. There are therefore two main ways to reduce the discretization error: increasing the number of computational cells in the grid and increasing the accuracy of the numerical schemes. In this work, a methodology to use both techniques simultaneously is presented, introducing more cells in the solution to raise the spatial accuracy and using second-order accurate schemes to increase the temporal accuracy.

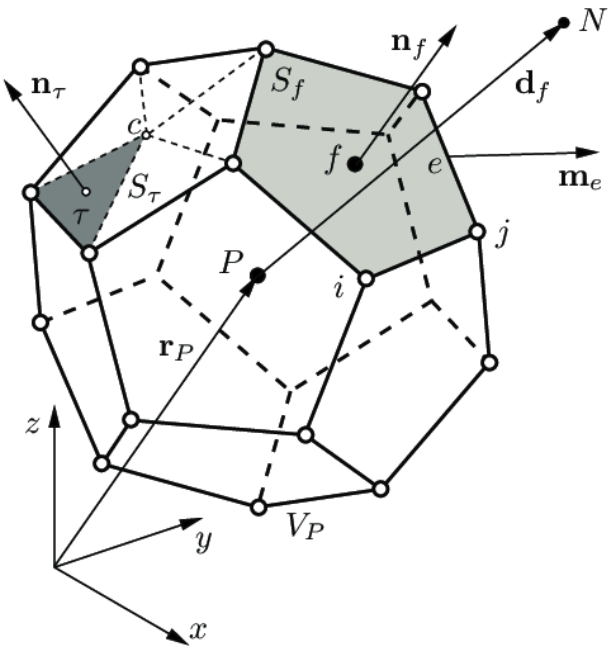


Figure 2.1: Polyhedral Control Volume [65]

2.2. Finite Volume Mesh

The solution of the algebraic equations will provide the values of the fluid variables at determined locations of the domain and at particular time instants. As time is a one-dimensional variable that only moves forward, indicating a time step is sufficient to indicate the temporal discretization. However, spatial discretization poses more challenges as the three dimensional space must be separated into individual Control Volumes (CV) that fill the domain completely and uniquely. The algebraic system of equations will provide the solution at the center of the CVs, indicated by P in Fig. 2.1. The computational point P corresponds to the centroid of the CV and can be calculated like:

$$\int_{V_P} (\boldsymbol{x} - \boldsymbol{x_P})\, \mathrm{d}V = 0 \tag{2.1}$$

The limits of the CV are set by an arbitrary number of flat faces, each of which is shared with only one neighboring cell (denoted by N in Fig. 2.1). Each CV is therefore a polyhedron and can have an arbitrary number of faces, edges and vertices. The faces will have a normal vector defined in the center of the face (named $\boldsymbol{n_f}$ in Fig. 2.1) that will point towards the direction of the cell with lower label, which will be the *owner* of the face. The other cell will be the *neighbor*. If a face has an owner and a neighbor it will be an internal face, while boundary faces will not have any neighbor cell. Knowing the owner and the neighbor cells of a face is

enough to define completely the mesh topology, while reducing the memory storage required, accelerating face accessing and providing considerable freedom in mesh generation.

Volume Calculation

A very important step in CFD simulations is the computation of the volume and the face surface vector of all the cells in the domain. In the discretization of any 3D geometry, it might occur that some of the boundaries are not planar and therefore, some of the cell faces are not necessary planar. To deal with these cells, the face is separated in a set of plane triangles, depending on the desired accuracy. This is clearly an approximation but introducing enough number of triangles, a perfect representation of the surface can be achieved. Once the cell is represented as an arbitrary number of planar, non-overlapping faces, its volume can be calculated with the following expression, derived from the Gauss theorem for any given field $\boldsymbol{B}$:

$$\int_{\tilde{\Omega}} (\nabla \cdot \boldsymbol{B})\, dV = \int_{S} \boldsymbol{B} \cdot d\boldsymbol{S} \tag{2.2}$$

If one chooses a field $\boldsymbol{B}$ such that $\nabla \cdot \boldsymbol{B} = 1$, the first term becomes immediately the volume of the cell. One such example could be the field $\boldsymbol{B} = x\boldsymbol{i}$, but also $y\boldsymbol{j}$, $z\boldsymbol{k}$ or $1/3(x\boldsymbol{i} + y\boldsymbol{j} + z\boldsymbol{k})$. If the first one is considered as an example, and the normal vector to a face is expressed as $d\boldsymbol{S_f} = S_{x,f}\boldsymbol{i} + S_{y,f}\boldsymbol{j} + S_{z,f}\boldsymbol{k}$, the equation becomes:

$$V = \int_{S} x\boldsymbol{i} \cdot (dS_x\boldsymbol{i} + dS_y\boldsymbol{j} + dS_z\boldsymbol{k}) = \int_{S} x dS_x \approx \sum_{f} x\ S_{x,f} \tag{2.3}$$

that allows to compute the volume of any polyhedral cell from its faces. To compute the volume of all the cells of the domain, one must ensure that the definition of the faces is the same for neighboring cells in order to avoid empty or overlapping spaces in the computational domain.

Surface Normal Calculation

However, it remains the question of how to compute the normal vector of any given surface, that might have any number of edges. The simplest approach for this matter is to calculate the face centroid (c) and decompose the surface in triangles (τ) having this vertex in common, like the pentagonal face S_τ in the top of Fig. 2.1. The area and normal vector ($\boldsymbol{n_\tau}$) of each one of the triangles can be easily computed with a cross product of two of its edges. The normal vector and the area of the full surface are computed now by just summing the results of each

individual triangle:

$$\boldsymbol{S_f} = \frac{1}{2}\sum_{i=1}^{N_v}\left[(\boldsymbol{r_{i-1}} - \boldsymbol{r_1}) \times (\boldsymbol{r_i} - \boldsymbol{r_1})\right] \tag{2.4}$$

where N_v represents the number of vertices of the surface and $\boldsymbol{r_i}$ the position of the vertex i. This expression holds even for twisted or convex faces and it is independent of the choice of the common vertex. The area of the surface is computed as the magnitude of the normal vector:

$$S_f = |\boldsymbol{S_f}| = \sqrt{S_{x,f}^2 + S_{y,f}^2 + S_{z,f}^2} \tag{2.5}$$

2.3. Mesh Induced Errors

The solution of the algebraic system of PDEs will provide the solution only at determined locations of the domain. The domain discretization will therefore introduce some error in the solution that must be handled carefully if a good accuracy is to be achieved. If a linear variation of the solution inside each CV is assumed (providing second order spatial accuracy), cell distribution across the whole domain is crucial to capture all the phenomena. In regions where the solution changes rapidly, small CVs are needed; otherwise, the mesh will have an insufficient resolution and some information will be lost. A particularly important region is the boundary layer, where small cells are usually required.

The quality of the generated grid is also very important, specially for complex geometries, as it can introduce large errors. Ideally, one should use an uniform Cartesian grid, as the midpoint rule used to compute the values in the faces provides the exact solution. However, it is almost never possible to use such grid as the boundaries are not always aligned with the grid lines. Not using Cartesian grids will introduce additional sources of error in the solution, as it will be explained below. The angle between the normal vector of a face and the line connecting the CV centers at both sides of the face is the crucial parameter defining the grid quality. However, the orthogonality of the face edges at the CV vertices is not important. The different sources of error introduced by non-Cartesian grids and how they affect to the precision of the results will be explained in the following sections.

2.3.1. Aspect Ratio

Aspect ratio refers to the ratio of the length of the longest side of a cell to the length of the shortest side. A cell with an aspect ratio close to 1 is considered to be well-shaped or isotropic, meaning that its dimensions are roughly equal in all directions. High aspect ratio cells can cause numerical diffusion, which occurs when a numerical scheme diffuses the solution across

the domain, resulting in a loss of accuracy. This can happen because the high aspect ratio cells have a large area-to-volume ratio, which makes the error in one direction more important than in other, thus diffusing the errors unevenly.

High aspect ratio cells can also affect the grid convergence of a CFD simulation, which refers to the ability of a simulation to produce accurate results as the grid is refined. High aspect ratio cells can lead to a slower convergence rate, meaning that more cells are required to achieve a certain level of accuracy. They can also cause problems with turbulence modeling, particularly in the near-wall region, where the velocity gradients are highest, resulting in incorrect predictions of turbulence intensity and length scales.

However, in some cases high aspect ratio cells are intentionally used in regions where the flow gradients are primarily in one direction as they can provide accurate results with less computational cost in those regions. For example, near the boundaries, the gradients in the direction of the wall are smaller than the ones in the normal direction. Using high aspect ratio cells in these regions provides higher resolution in the normal direction while keeping a relatively low number of computational cells. To mitigate the effects of high aspect ratio cells, one can use grid adaptation techniques, increase the number of cells in the problematic regions, or use specialized numerical methods that are better suited for handling high aspect ratio cells.

2.3.2. Non-Uniformity (Smoothness)

An uniform (smooth) mesh is a computational grid where the distance between neighboring CV centers is relatively uniform across the whole domain and there are no sudden changes in element size or shape. It is important because it reduces numerical errors, such as the interpolation error, and leads to better convergence of iterative solvers. An uniform mesh can be generated through careful selection of mesh parameters and refinement criteria, but it is a complicated task when dealing with complex geometries. Obtaining an appropriate uniformity of the grid is usually a compromise solution with the rest of the grid parameters, aiming at obtaining an final mesh that is accurate enough to capture all the important phenomena, while maintaining an affordable computational time.

2.3.3. Skewness

In the FV method, estimating the value of a quantity ϕ at element faces is crucial to evaluate many terms of the discretized transport equations. It is commonly assumed that the average value of ϕ over the entire face can be used as an estimate at the face. However, during different steps of the discretization process, linear variation of variables between nodes is assumed, and

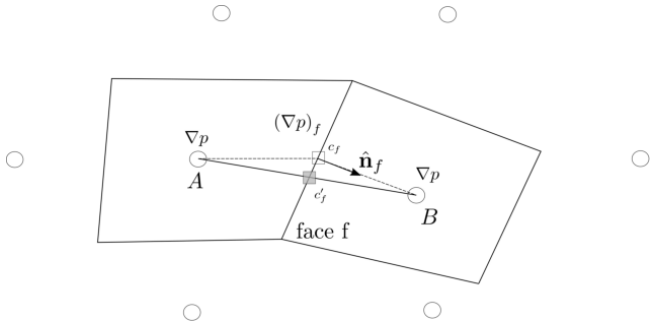


Figure 2.2: Skewed mesh

this assumption is extended to the variation of variables along the face. As a result, the average value of any variable ϕ over a face is expected to be found at its centroid.

To estimate the value of ϕ at the face, a linear interpolation profile is typically used. The value at the face is estimated at the intersection between the face and the line connecting the two nodes straddling the face. However, when the grid is skewed, the line does not necessarily pass through the centroid of the face, resulting in an intersection point c'_f that differs from the face centroid c_f (see Fig. 2.2). The skewness can be measured as:

$$\psi = \frac{|\boldsymbol{d}_{AB}|}{|\boldsymbol{c_f} - \boldsymbol{c'_f}|} \tag{2.6}$$

To maintain the overall spatial accuracy of the discretization method at second order, it is necessary to perform all face integrations at the face centroid. Therefore, a correction in a skewed mesh for the interpolated value at c'_f is required to obtain the value at c_f. This is particularly important in high-accuracy simulations where even small errors can have significant impacts on the results. If no correction is performed, a diffusion-type error will be introduced in the solution.

2.3.4. Non-Orthogonality

The non-orthogonality of a face is measured as the angle θ_{no} in Fig. 2.3. It is the angle formed between the line joining the cell centers of the owner and neighbor of the face ($\boldsymbol{d}$) and the normal vector of the face ($\hat{\boldsymbol{n}}$), and can be calculated as:

$$cos(\theta_{no}) = \frac{|\boldsymbol{d}_{AB}|}{|\boldsymbol{c_f} - \boldsymbol{c'_f}|} \tag{2.7}$$

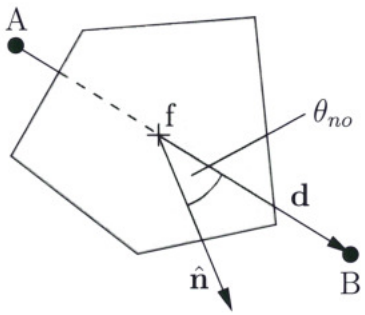


Figure 2.3: Non orthogonal face

The non orthogonality of a grid is important when solving the Laplacian terms of the transport equations, where the calculation of the gradient in the center of a face is required. If the mesh is orthogonal, this term can be computed simply as:

$$\nabla_n \Phi_f = \hat{\boldsymbol{n}}_{\boldsymbol{f}} \cdot (\nabla \Phi)_f = \frac{\Phi_B - \Phi_A}{|\boldsymbol{r_B} - \boldsymbol{r_A}|} = \frac{\Phi_B - \Phi_A}{d_{AB}} \tag{2.8}$$

Unfortunately, most structured curvilinear grids and unstructured grids are non orthogonal. In this case, the gradient normal to the surface will have a component perpendicular to the direction of the segment AB. Correcting this component is crucial as it affects the solution stability, smoothing any nonphysical oscillations that might appear in the iterative solution process. The correction used in this work is explained in Sec. 3.3. In addition, this type of error cannot be reduced by a mesh refinement and a different mesh topology is usually required.

2.4. Types of Mesh

When performing computational fluid dynamics simulations, the selection of the appropriate type of mesh is a crucial step in achieving accurate and stable results. Several factors need to be considered when choosing the mesh configuration. Firstly, the choice of mesh should be consistent with the numerical methods being used. For example, structured meshes are often used with finite-difference methods, while unstructured meshes are more commonly used with finite-volume and finite-element methods. Computational resources should also be taken into account as some mesh types require more computational power to be generated and to provide a solution.

Additionally, the selected mesh type must be able to accurately capture the relevant physical phenomena of the simulation. This is particularly important in the simulation of turbulent flows where appropriate turbulence models and mesh resolution are required to, for example,

provide good results in the boundary layer.

The quality of the mesh should also be considered. Various mesh quality criteria, such as aspect ratio, skewness, and orthogonality, should be examined to ensure that the mesh is of sufficient quality. Furthermore, the mesh should be fine enough to achieve convergence and the desired accuracy of the simulation. However, a finer mesh also increases computational cost, so a balance between accuracy and computational cost must be achieved.

The different types of mesh existing in the literature are reviewed in the following sections, explaining their characteristics and for which type of simulations each one of them are most suited. All of them present some advantages and drawbacks, being the choice of the mesh type a compromise decision the user has to make. This decision will impact not only the accuracy and computational time of the simulation, but also the engineering time required to generate the mesh by the user. Some mesh types can be generated automatically while others require a large time to be setup by expert users if a good quality grid is desired, potentially constituting a bottleneck in the procedure.

2.4.1. Non-Conformal Grid

The grid with the best possible quality, minimizing all the errors indicated in Sec. 2.3, would be a Cartesian one. It uses a set of orthogonal coordinate axes, typically x, y, and z, to define the location of the grid points or cells, which will be distributed uniformly. Cartesian grids have a regular structure that allows for efficient and accurate numerical methods. However, they are limited in their ability to handle problems with non-rectangular or curved boundaries, which require a lot of cells to accurately represent the geometry in a stepwise approximation. This can lead to large computational costs, especially in three dimensions, and can make the use of Cartesian grids impractical or inefficient for certain problems.

To take advantage of these facts, one could use non-conformal grids, where the grid does not adapt to the boundaries of the domain, like in Fig. 2.4. The most typical method using this approach is the Immersed Boundary Method (IBM), introduced by Peskin in 1972 [51]. In this method, the fluid flow equations are solved on a fixed Cartesian grid while the presence of a solid body or boundary is modeled by the introduction of a body force term in the equations of motion. This term acts as a virtual force that mimics the presence of the solid body and spreads out the information over the grid using an interpolation scheme.

The immersed boundary method is particularly useful in simulating complex and/or moving geometries, where it may be difficult or computationally expensive to generate a mesh that conforms to the solid boundaries. It is of particular importance in fluid-structure interaction problems, where it is a current research topic [44].

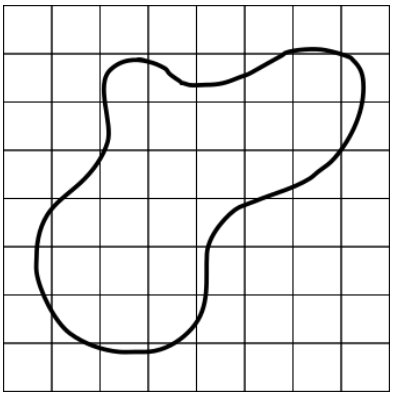


Figure 2.4: Non conformal grid

One of the main limitations of IBM is its dependence on the grid resolution. The method requires a fine grid resolution near the boundary to accurately capture the interactions between the fluid and the immersed objects. This can lead to high computational costs and may limit the applicability of IBM to relatively simple geometries or to small boundary motions. In addition, imposing the appropriate boundary conditions can be difficult and may require additional modeling assumptions or experimental data. It will limit the accuracy of the solution near the immersed boundaries and may introduce numerical instabilities.

2.4.2. Body-Fitted Grid

A body-fitted grid is a type of grid that is constructed to conform to the geometry of the computational domain. It can capture the geometry of complex shapes, providing a better resolution in regions close to the boundaries. In addition, body-fitted grids can often be more efficient than non conformal grids as they usually require fewer grid points next to the boundaries to achieve the desired level of accuracy, thus reducing the computational cost.

However, generating the grid for complex geometries can be time-consuming and require significant computational resources. This is even more critical for moving grids, where not only the initial grid has to be generated but also its motion during the simulation. Different types of body-fitted grids are explained in the following sections.

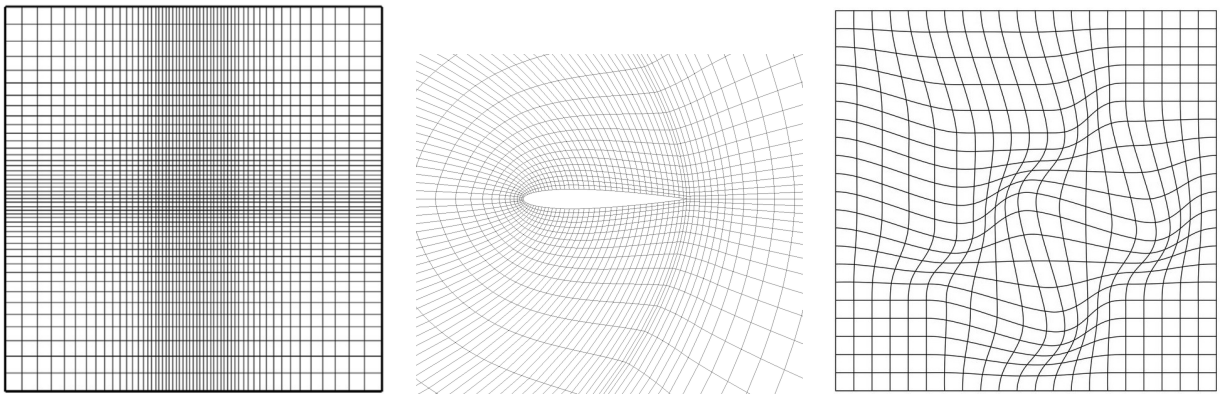
Structured (Regular) Grid

This type of grids is composed by several families of grid lines that do not cross with other grid lines of the same family, and cross each grid line of the other families only once. From these properties it follows that every point P in the grid can be uniquely identified by a set of three indices in a 3D geometry, e.g. (i, j, k). Each point has 6 neighbors and the indices of each

neighbor differs by ± 1 from one of the indices of P. This fact implies a constant connectivity across the mesh that translates into a matrix of the system of equations having a regular structure, which require less memory and computational resources to store and process. In addition, they provide faster convergence rates than other type of grids because special solvers optimized for structured grids can be used. The main disadvantage of this type of mesh is that they can only be used in simple geometries. The distribution of the cell size across the domain may be difficult to control as introducing small cells in one part of the domain will imply using them in another regions where they are not needed.

The structured grids can be further differentiated according to the distribution of the grid lines:

- **Orthogonal:** The grid lines are straight lines and the intersection with one another is always perpendicular, like in Fig. 2.5a. It provides the highest quality but it can not adapt correctly to the boundaries.
- **Curvilinear:** similar to the orthogonal mesh but with the grid lines expressed in curvilinear coordinates (Fig. 2.5b)
- **Non-orthogonal:** The grid lines are not orthogonal. It provides more flexibility to create the mesh in the boundaries, facilitating the imposition of the boundary conditions, but the grid quality diminishes (Fig. 2.5c).



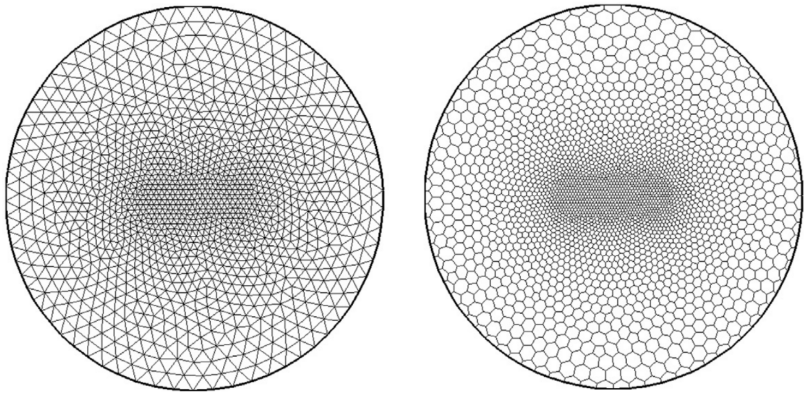
(a) (b) (c)

Figure 2.5: a) Structured Orthogonal grid; b) Structured Curvilinear grid; c) Structured Non-orthogonal grid. Figures taken from [43]

Unstructured Grid

This is the most flexible type of grid and is able to fit any arbitrary boundary. The cells may have any shape and any number of neighbors, therefore facilitating its automatic generation using, for example, a Delaunay triangulation. However, the mesh connectivity is no longer

constant and must be stored in additional memory space. In addition, the matrix of the algebraic equation to be solved does not have a regular, diagonal structure like in structured grids. This usually requires a reordering of the points to reduce the band width of the matrix. The solvers used to solve the algebraic system of equations will therefore be slower than those for structured grids. Two examples of this type of grids are shown in Figs. 2.6a and 2.6b for tetrahedral and polyhedral cells.



(a) (b)

Figure 2.6: a) Unstructured Tetrahedral grid; b) Unstructured Polyhedral grid

Unstructured meshes are preferred for complex geometries, as they allow for greater flexibility in mesh generation. They provide bigger control of the cell size and quality, allowing for local refinements like in Fig. 2.7, where smaller cells are introduced in the intake. In addition, they can reduce the number of cells in the domain compared to structured grids, as they can be concentrated in the most important regions.

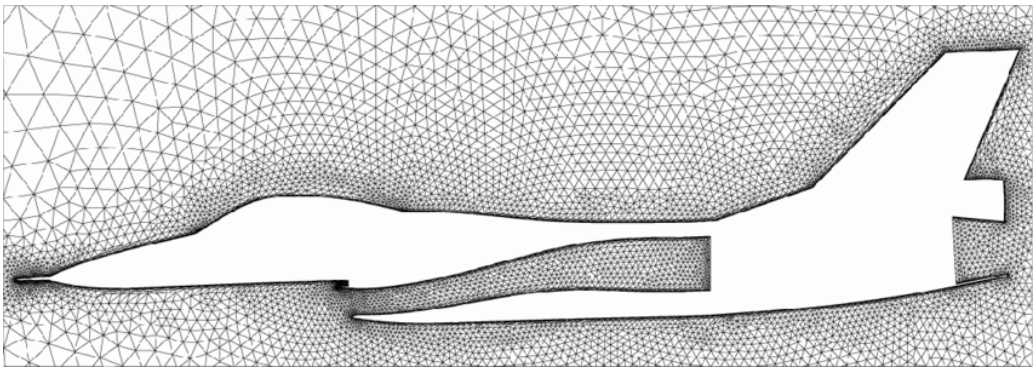


Figure 2.7: Unstructured mesh for the simulation of an F-16 plane [46]

Hybrid Grid

The hybrid mesh is such in which some region of the domain has structured mesh and the other has an unstructured one in order to take advantage of both strategies. The typical example is the meshing of the boundary layer, like in Fig. 2.8. If one wants to solve accurately the flow next to the surfaces, very small cells are needed in the direction normal to the boundary, where the gradients of the solution are bigger. However, along the surface the flow has smaller variations and bigger cells can be used. If triangular cells (or tetrahedral in 3D) are selected, the created elements will have a very high skewness and will introduce a large error in the solution. However, the use of structured grids can provide such high-aspect ratio cells, reducing the number of cells needed in the region and increasing the accuracy of the solution. This approach is usually used in most CFD simulations, where a small number of structured layers of cells are placed along the surfaces. Then, the rest of the mesh is generated with structured or non-structured distribution.

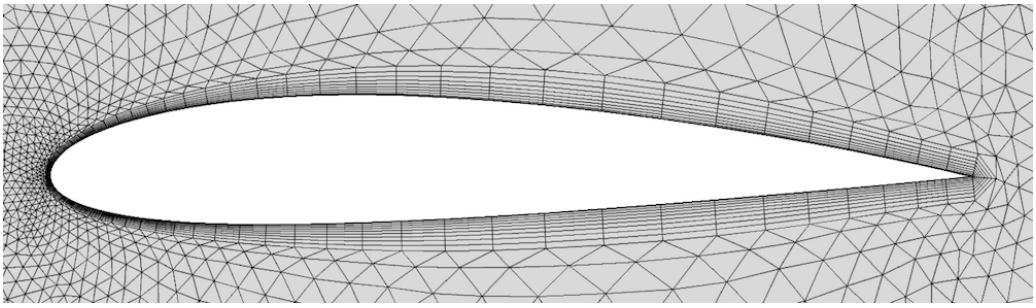


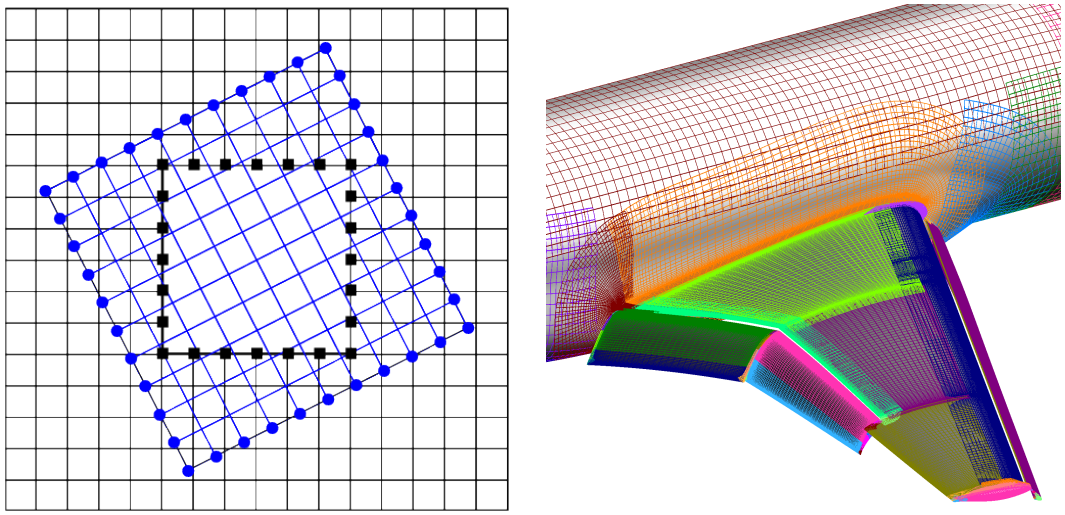
Figure 2.8: Hybrid mesh

Overlapping (Chimera) Grid

Overlapping grids is an innovative approach wherein multiple individual grids, each designed for a specific component or feature of the geometry, are overlaid to represent the entire flow domain. These grids are generated independently, allowing for greater flexibility and ease in meshing complex or moving parts. Instead of trying to fit a single grid around every intricate detail, one can focus on generating high-quality grids for each subdomain, ensuring accuracy where it is most needed. A general example of this technique is shown in Fig. 2.9a.

The main advantage of overlapping grids lies in their modular nature. As components move or as the simulation's focus shifts, grids can be adapted, refined, or even replaced without disrupting the entire mesh structure. This modularity is especially advantageous in simulations with relative motion between components, such as aircraft wing flaps or multi-body interactions in water. An example of this modularity is shown in Fig. 2.9b, provided by NASA [11], where each color represents an individual grid. It is evident how this technique provides high-quality

local meshes optimized for each part of the geometry. For instance, the leading edge of the wing has a finer mesh than the pressure and suction sides, as the field gradients are more significant in that region. Individual meshes are employed for each moving part, allowing them to move independently without affecting the rest of the individual grids.



(a) (b)

Figure 2.9: a) General scheme of 2 overlapping grids; b)Using of overlapping grids in a wing by NASA [11]

Solving fluid problems using overlapping grids requires a set of unique steps:

1. **Grid generation:** the first step is to separate the computational domain in smaller subdomains. Then, individual grids for each geometric component or region of interest are generated. These grids can be structured or unstructured, depending on the characteristics of each subdomain. Grids must sufficiently overlap one another, in order to ensure that there is an adequate buffer region for the interpolation.
2. **Grid connectivity and hole cutting:** once the individual grids have been created, the overlaying regions must be determined, identifying the cells belonging to one grid (the *donor*) and the other (the *receptor*). If more than two grids overlap in the same region, a hierarchical relationship has to be declared, stating which grids are donors and receptors to one another. Then, the hole cutting procedure takes places, where cells in the receptor grid that are overlapped by a higher-priority donor grid are marked as inactive. These cells will not be involved in the computation as they would only provide redundant information already present in the donor grid, adding computational time and memory. An example of the hole cutting procedure is shown in Fig. 2.9a, where some overlapping

cells in the background (black) mesh have been removed.

3. **Interpolation weights calculation:** the next step is to identify the interpolation stencil for each receptor cell. That is, for each boundary cell in the receptor grid (cells around the hole, depicted as solid black squares in Fig. 2.9a), identify a set of cells of the donor grid from which the solution values will be interpolated. This set of cells is called the interpolation stencil. Then, the interpolation weights are calculated based on the relative locations of the donor cells and the interpolation point in the receptor grid. In linear interpolation, these weights are directly proportional to the distance from the interpolation point to the stencil nodes. Higher-order interpolations, like quadratic or cubic schemes, will have more complex weight calculations. The interpolation method will also affect the interpolation stencil. For example, a bilinear interpolation in 2D would need a 2x2 stencil, while a trilinear interpolation in 3D would need a 2x2x2 stencil.

4. **Solve flow equations:** Before beginning an iteration of the flow computation, boundary values in the overlap regions have to be interpolated using the pre-computed interpolation weights. Using the determined weights (ω_i), flow variables (ϕ) (like velocity, pressure, temperature, etc.) are interpolated from the donor cells to the receptor cell like:

$$\phi_{receptor} = \sum_i \omega_i \ \phi_{i,donor} \tag{2.9}$$

This step ensures that the starting point of each iteration has consistent values at the boundaries of each subdomain. Then, the fluid equations are solved independently on each grid as one would do on any standard mesh. After the flow equations have been solved for an iteration, and before advancing to the next iteration, values from the donor grids to the receptor grids in overlapping regions have to be interpolated again. This ensures that the next iteration starts with boundary values that are consistent with the latest solution. This introduces some computational overhead, as one additional step has to be performed every iteration. However, depending on the problem's nature and the solver's performance, the interpolation frequency might be adjusted. For instance, at the beginning of the simulation, when the solution is far from convergence, interpolation might be performed every few iterations instead of every iteration. As the solution gets closer to convergence, interpolation might be performed more frequently, even every iteration, to maintain solution accuracy and stability.

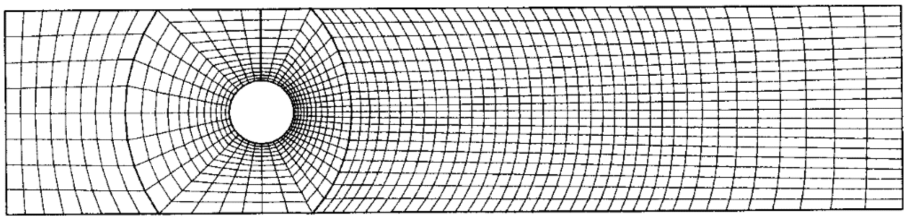
Proper synchronization of these steps is crucial. Too infrequent interpolation can lead to divergence or inaccuracies, while excessive interpolation can add computational overhead without significantly benefiting solution quality. Furthermore, other advanced techniques

like under-relaxation, multi-grid acceleration, or implicit time-stepping can be incorporated, each introducing its nuances to the interpolation's timing and frequency.

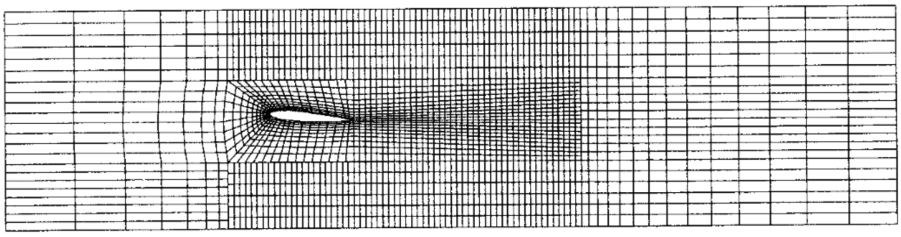
In addition, it has to be noticed that this methodology does not ensure conservation, as no conservation equation is solved through the boundaries. Even though the error introduced is very small and might be sufficient for some applications, one must use an interpolation scheme that ensures mass, momentum and energy conservation. This translates into choosing interpolation weights so that the amount of a conserved quantity (e.g., mass) taken from a donor cell equals the amount added to a receptor cell. Alternatively, one can ensure that the fluxes across the boundaries of overlapping grids are consistent, integrating the fluxes and forcing them to match.

Overset grids are particularly suited for problems with moving boundaries as each of the grids can move independently. If configured properly, they maintain a high quality grid even during large deformations, like in the case of ICE simulations. They are relatively easy to configure and allow to save a lot of time in the mesh generation process, allocating higher resolution to areas where it is most critical while adopting coarser resolutions in other regions. However, the grid motion introduces some additional steps in the methodology that will raise the computational time of the simulation. In particular, due to grid movement or deformation, the connectivity between overlapping grids can change with time. Thus, the hole-cutting process and grid connectivity determination must be frequently updated, as well as the recalculation of the interpolation weights. This also requires more complex interpolation techniques in order to ensure mass, energy and momentum conservation along the overlapping regions and to avoid the introduction of non-physical oscillations or artificial mass sources. Sometimes, it may also be necessary to reduce the time step to obtain smaller CFL number and ensure the accuracy and stability of the solution.

In terms of using multi-core processing for the computation, dynamic overset grids also introduce some challenges. With static grids, the entire computational domain, including all the grids, is decomposed into smaller sub-domains or chunks. Each sub-domain is then assigned to a separate processor or a set of processors. However, as grids move or deform, the overlap regions between them can change, leading to a situation where data for a particular sub-domain, initially present on one processor, now resides on a different one. This required inter-processor communication to transfer data between sub-domains, typically using MPI (Message Passing Interface) or similar parallel computing tools, which increases the computational time. In addition, dynamic movement can cause load imbalances; for instance, if one processor ends up with more active grid cells than others due to grid motion, it will have a disproportionate amount of work. This can be solved by performing periodic load balancing to redistribute the work more evenly among processors, which involves dynamically reassigning portions of the grids to



(a) Matching



(b) Non-matching

Figure 2.10: Block-structured grid

different processors. This will also affect to the interpolation process in the boundaries as the processors that need to communicate may change, leading to some synchronization overhead as some processors may need to wait for the others to finish before moving on.

Block-Structured Grid

In block-structured grids the domain is separated into different blocks, where each one has its own coordinate system that can be used to generate individual structured meshes. This makes it easier to handle complex geometries and to control the cell distribution, as each block can have its own resolution and orientation. In very complex geometries, blocks with unstructured grids could also be created, constituting a hybrid grid. If the blocks overlap each other, a similar approach than in Chimera grids can be used, interpolating from one block to the other in the overlapping region. If the blocks do not overlap, an interface will appear where cell vertices from one block may (Fig. 2.10a) or may not (Fig. 2.10b) match with the vertices from the other.

When the vertices do not match, a special treat is required. As shown in Fig. 2.11 for a 2D case, the cell L in the block A will now have 3 neighboring cells in its right side. To circumvent this situation, new vertices are added in the intersections of both faces, splitting the faces in several smaller ones. The topology of all the cells next to the interface will change, adding all the new faces and their corresponding connectivity. In the case of the cell L in Fig. 2.11, it will pass from having 4 faces to 6. The same methodology can be applied to three-dimensional problems and to interfaces between structured and non-structured grids.

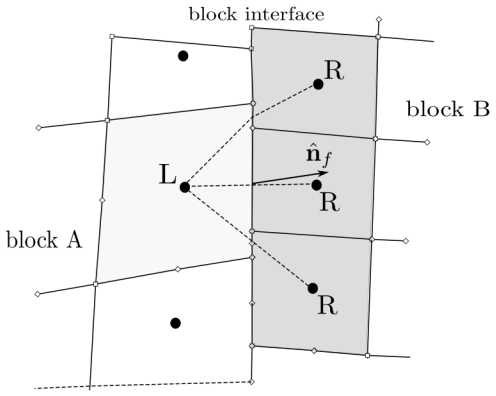


Figure 2.11: Interface between two non matching blocks

The use of block-structured grids provides several advantages in CFD simulations. Firstly, it allows for a grid refinement only in the regions of interest, and not in the entire domain, while keeping the good properties of structured grids locally. This can significantly reduce the computational cost and memory requirements of the simulation as specialized solvers for structured grids can be used on each block individually. Secondly, the pre-defined connectivity between blocks facilitates the implementation of parallel algorithms as each block can be solved in an separated node, allowing for faster computation times. On the other hand, problems may arise when the interface has a complex geometry, like in sharp corners. Also, a large difference in the cell size between two connected blocks can introduce a large error in the solution and curb its convergence. For this reason, a ratio below 2:1 in the cell sizes is usually recommended.

2.5. Mesh Refinement

Mesh refinement is a critical aspect of the mesh, as it allows for the accurate representation of complex geometries and fluid flows. By refining the mesh, the discretized equations can more accurately capture the physics of the flow, leading to improved accuracy and reduced numerical errors. There are various types of mesh refinement techniques that can be applied depending on the specific simulation requirements. Each technique has its advantages and disadvantages, and the appropriate technique must be selected for each case. Additionally, mesh refinement can have a significant impact on the computational cost of a simulation. As the mesh is refined, the computational time required to solve the equations increases. Therefore, it is essential to balance the level of mesh refinement with the available computational resources to ensure that the simulation can be completed within a reasonable time.

The most basic technique to refine the mesh consists of creating smaller cells in regions where the gradients are expected to be large or in regions with important geometric features like

corners, edges or fillets. This allows to have more cells in the places where they are needed the most, capturing all the relevant phenomena while keeping a relatively moderated number of cells. If these cells are generated by the division of initial cells into smaller ones, it is called *H-refinement*. It is a common technique used in adaptive mesh refinement (AMR) algorithms, where the mesh is dynamically refined and coarsened based on a set of criteria, like the solution gradient.

Another type of mesh refinement involves increasing the order of the polynomial used to represent the solution inside a given cell. It is known as *P-refinement* and provides more accurate solutions without increasing the number of cells in the mesh. P-refinement is particularly useful in regions of smooth flow where H-refinement would be unnecessary. However, it can introduce challenges related to numerical stability and may require more robust integration techniques to speed up the convergence of the solution. Both techniques can be used together in what is called *HP-refinement*, which allows for adaptive refinement where both the mesh size and polynomial order can be adjusted to achieve the desired accuracy in different regions of the computational domain.

3 Numerical Schemes

Some of the most relevant problems in the industry involve the solution of domains with deforming or moving boundaries, like IC engines, rotatory machines or fluid-structure interaction simulations. In this work, the Arbitrary Lagrangian-Eulerian method has been used, as it is particularly suited for large domain deformations. This technique requires to compute the relative fluxes in the convective term of the transport equations. The way of computing them is provided by solving the Geometric Conservation Law, that imposes a relationship between the numerical schemes used for the temporal and convective terms. Solving this equation is crucial to avoid the apparition of spurious mass sources in the solution. Once the convective terms have been computed, the system of equations is solved sequentially using the PIMPLE algorithm.

3.1. Arbitrary Lagrangian-Eulerian (ALE) Method

The Arbitrary Lagrangian-Eulerian (ALE) method is a numerical technique used in Computational Fluid Dynamics to simulate problems that involve large deformations, complex geometries, and moving boundaries. The ALE method was developed in an attempt to combine the advantages of the Lagrangian and Eulerian kinematical descriptions, while minimizing their respective drawbacks [18].

In the Lagrangian description, the computational mesh is made up of individual nodes that follow the motion of the associated fluid particles: the mass of fluid inside each cell remains constant during the simulation. However, since the mesh must move with the fluid, it can quickly become distorted. This description is useful for tracking free surfaces and interfaces between different materials, which is why it is often used for fluid-structure interaction problems. In contrast, the Eulerian description keeps the mesh fixed in space while the fluid moves with respect to it. This makes it easier to handle large distortions in the flow but often at the expense of precise interface definition and resolution of flow details.

In the ALE method, the nodes of the computational mesh are allowed to move in a variety of

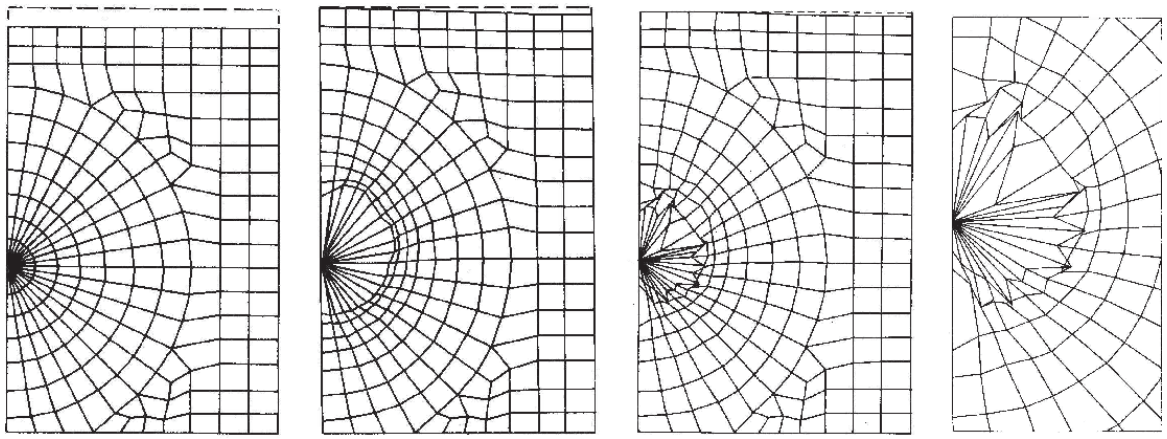


Figure 3.1: a) Initial mesh; b) ALE mesh; c) Lagrangian mesh; d) Details of the distortion in the Lagrangian description [18]

ways, providing flexibility in the description of the flow. This includes the ability to move with the continuum in a typical Lagrangian fashion, remain fixed in space in an Eulerian manner, or be moved in an arbitrarily specified way to allow for continuous rezoning. This freedom in the movement of the computational mesh allows for the handling of greater distortions of the fluid than what would be possible with a purely Lagrangian method, while providing better resolution than what is possible with a purely Eulerian approach (see Fig. 3.1).

To solve the governing equations with the ALE method, a time-splitting technique is typically used. This involves splitting the equations into a Lagrangian step and an Eulerian step. In the Lagrangian step, the equations are solved on a moving mesh that follows the motion of the fluid. The mesh is deformed using a displacement field that is calculated from the fluid velocity. The deformation of the mesh will affect the boundary conditions, which must be updated at each time step to account for the new location and orientation of the boundary. In the Eulerian step, the equations are solved on a fixed mesh that is aligned with the original undeformed geometry. The two steps are coupled through the use of a mapping function that maps the Lagrangian mesh onto the Eulerian mesh and ensures that the solution is conserved across the two meshes.

The interaction between the Lagrangian and Eulerian descriptions is a crucial aspect of the ALE method. The deformed mesh introduces distortions that can affect the accuracy and stability of the solution. To mitigate this, a remeshing or mesh adaptation strategy can be employed. Remeshing involves generating a new mesh at each time step based on the current configuration of the fluid, while mesh adaptation involves modifying the existing mesh to improve its quality and accuracy. The connection between meshes is typically a bijective mapping, meaning that

each point in the Lagrangian mesh is uniquely mapped to a point in the Eulerian mesh and vice versa. However, meshes with different topology and number of cells can also be used, as it will be explained in Sec. 4.

3.2. Governing Equations for the Fluid Transport in Time-Varying Domains

A lot of interesting problems in engineering involve solution domains that vary over time, as the boundaries move. This movement can be imposed, like in piston-driven flows, or can be calculated as part of the solution, like in fluid-structure interaction problems. In any event, as the solution domain changes, the grid must therefore accommodate to this movement. If one assumes a fixed, Cartesian reference frame, the continuity equation can be written as:

$$\frac{\partial}{\partial t}\int_{\Omega}\rho\,\mathrm{d}V+\int_{\partial\Omega}\rho\,\boldsymbol{U}\cdot\boldsymbol{n}\,\mathrm{d}S=0 \tag{3.1}$$

If the integral over a single one-dimensional, deforming control volume, whose boundaries move from $x_1(t)$ to $x_2(t)$, is considered, one can write:

$$\int_{x_1(t)}^{x_2(t)}\frac{\partial\rho}{\partial t}\,\mathrm{d}x+\int_{x_1(t)}^{x_2(t)}\frac{\mathrm{d}(\rho U)}{\mathrm{d}x}\,\mathrm{d}x=0 \tag{3.2}$$

Applying the Leibniz's rule to the first term and integrating the second one, it follows:

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{x_1(t)}^{x_2(t)}\rho\,\mathrm{d}x-\left[\rho_2\frac{\mathrm{d}x_2}{\mathrm{d}t}-\rho_1\frac{\mathrm{d}x_1}{\mathrm{d}t}\right]+\left[\rho_2U_2-\rho_1U_1\right]=0 \tag{3.3}$$

where the indices 1 and 2 represent the boundaries of the one-dimensional control volume. The term $\frac{dx_i}{dt}$ represents the velocity at which the boundary moves, denoted as $U_{b,i}$. As both terms have the same form, the equation can be rewritten as:

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{x_1(t)}^{x_2(t)}\rho\,\mathrm{d}x+\int_{x_1(t)}^{x_2(t)}\frac{\partial}{\partial x}\left[\rho\left(U-U_b\right)\right]\mathrm{d}x=0 \tag{3.4}$$

If a three-dimensional time-changing domain $\tilde{\Omega}(t)\subset\mathbb{R}^3$ is considered, the following version of the mass conservation equation is found:

$$\frac{\partial}{\partial t}\int_{\tilde{\Omega}(t)}\rho\,\mathrm{d}V+\int_{\partial\tilde{\Omega}(t)}\rho(\boldsymbol{U}-\boldsymbol{U}_b)\cdot\boldsymbol{n}\,\mathrm{d}S=0 \tag{3.5}$$

Comparing the mass conservation equation in static grids (Eq. 3.1) with its counterpart for dynamic grids (Eq. 3.5), it can be seen how the latter can be obtained by substituting the velocity vector in the convective term $\boldsymbol{U}$ with the velocity relative to the boundary: $\boldsymbol{U} - \boldsymbol{U_b}$. If the boundaries move with the same velocity than the fluid, no mass flux will go through the faces of the control volume and, therefore, the mass inside of it will remain constant: it becomes a control mass and the Lagrangian description of the fluid flow is obtained. On the other hand, if the boundaries do not move, the equations for a static grid are recovered, which corresponds to the Eulerian description of the fluid flow.

In fact, this transformation from control mass to control volume can be applied to any transport equation by using the Reynolds' transport theorem:

$$\frac{d}{dt}\int_{\Omega_{CM}} \rho\phi \,\mathrm{d}V = \frac{d}{dt}\int_{\Omega_{CV}} \rho\phi \,\mathrm{d}V + \int_{\partial\Omega_{CV}} \rho(\boldsymbol{U} - \boldsymbol{U}_b)\cdot\boldsymbol{n}\,\mathrm{d}S \tag{3.6}$$

Using this theorem, the conservation equation in integral form for a generic variable ϕ over a time-changing domain $\tilde{\Omega}(t) \subset \mathbb{R}^3$ reads:

$$\frac{\partial}{\partial t}\int_{\tilde{\Omega}(t)} \rho\phi \,\mathrm{d}V + \int_{\partial\tilde{\Omega}(t)} \rho[(\boldsymbol{U} - \boldsymbol{U}_b)\cdot\boldsymbol{n}]\phi \,\mathrm{d}S - \int_{\partial\tilde{\Omega}(t)} \Gamma_\phi \nabla\phi\cdot\boldsymbol{n}\,\mathrm{d}S = \int_{\tilde{\Omega}(t)} s_\phi \,\mathrm{d}V \tag{3.7}$$

where ρ is the density, $\boldsymbol{U}$ is the flow velocity, $\boldsymbol{U}_b$ is the velocity of the cell faces, Γ_ϕ is the diffusion coefficient and s_ϕ is any source/sink term of ϕ. From Eq. 3.7, continuity and momentum equations of a compressible flow can be obtained. By replacing $\phi = 1$, mass conservation is recovered (Eq. 3.5); making $\phi = \boldsymbol{U}$, the momentum equation can be obtained:

$$\frac{\partial}{\partial t}\int_{\tilde{\Omega}(t)} \rho\boldsymbol{U} \,\mathrm{d}V + \int_{\partial\tilde{\Omega}(t)} \rho\boldsymbol{U}[(\boldsymbol{U} - \boldsymbol{U}_b)\cdot\boldsymbol{n}]\,\mathrm{d}S - \int_{\tilde{\Omega}(t)} \nabla\cdot\tau \,\mathrm{d}V = -\int_{\tilde{\Omega}(t)} (\nabla p + \boldsymbol{F})\,\mathrm{d}V \tag{3.8}$$

with $p = \rho/RT$ as the thermodynamic pressure, τ as the resultant surface stress tensor and $\boldsymbol{F}$ as the resultant external volumetric forces. System closure is achieved by the so-called constitutive laws, whose formulation depends on the properties of the continuous medium, and, for compressible flows, by the energy equation.

In a co-located variable arrangement, like the one used in this work, all the primitive variables are assigned to the cell centroids, while face-centered variables are obtained by interpolation, denoted by the operator $[\![\cdot]\!]_f$. The mass fluxes, surface integrals from Eq. 3.8, are evaluated by either lower or higher order interpolation of velocity components at CV faces. However, in order to avoid decoupling of pressure and velocity when using co-located variable arrangement,

the interpolated pressure contribution to the cell face velocity is corrected [57]:

$$\frac{\partial}{\partial t}\rho\,\phi V + \sum_f \rho_f\,\phi_f\,(\varphi_f - \varphi_{M,f}) - \sum_f \Gamma_\phi \nabla\phi_f = s_\phi V \tag{3.9}$$

having defined the cell face flux φ_f:

$$\varphi_f = [\![\boldsymbol{U}]\!]_f \cdot \boldsymbol{n}_f\, S_f \tag{3.10}$$

being S_f the face area and $\boldsymbol{n_f}$ its normal unity vector. $\varphi_{M,f}$ is the corresponding mesh flux due to point motion (see [22]) and is expressed as:

$$\varphi_{M,f} = \boldsymbol{U}_{b,f} \cdot \boldsymbol{n}_f\, S_f \tag{3.11}$$

It represents the flux induced by the movement of the CV faces. If a face moves with the same velocity than the fluid, the mass flux through the CV face will be zero. If this is true for all the CV faces, the same fluid remains inside the CV and it becomes a control mass: a Lagrangian description of fluid motion is achieved. On the other hand, if a CV face does not move, its mesh flux will be zero and Eq. 3.7 simplifies to the traditional equation for static domains.

3.3. Variable Positioning and Spatial Discretization

In this work, the Finite Volume Method has been used, calculating the value of the intensive variables in the center of the cells with a collocated variable arrangement. Spatial discretization of the primary variables has been computed as in the following:

- *Diffusive term (Laplacian)* of a quantity Φ, e.g. $\nabla \cdot (\Gamma\nabla\Phi)$:

$$\int_V \nabla\cdot(\Gamma\nabla\Phi)\,dV = \int_S (\Gamma\nabla\Phi)_f \cdot \boldsymbol{n} dS \simeq \sum_f \Gamma_f \boldsymbol{S}_f \cdot (\nabla\Phi)_f = \sum_f \Gamma_f |\boldsymbol{S}_f| \nabla_n \Phi_f \tag{3.12}$$

  where $\nabla_n\Phi_f$ is the surface normal gradient of Φ. The subscript f in Eq. 3.12 indicates the cell-to-face interpolated quantities. Linear cell-to-face interpolation has been applied: for irregular polyhedral meshes, interpolation is generalized by defining a weight w for each face:

$$\boldsymbol{\Phi}_f = w\boldsymbol{\Phi}_P + (1-w)\boldsymbol{\Phi}_N \tag{3.13}$$

  where Φ_f is the face-interpolated quantity. Subscripts P and N indicate values at the

centers of two neighboring cells. The surface gradient of a quantity Φ is decomposed onto an orthogonal part and a (non-orthogonal) correction:

$$\nabla_n \Phi_f^n = \underbrace{\alpha(\Phi_P^n - \Phi_N^n)}_{\text{implicit}} + \underbrace{(\mathbf{n}_f - \alpha \mathbf{d}) \cdot (\overline{\nabla_n \Phi})_f^{n-1}}_{\text{correction (explicit)}} \tag{3.14}$$

where $\alpha = \frac{1}{\boldsymbol{n}_f \cdot \boldsymbol{d}}$, $\boldsymbol{d}$ the vector from P to N, and $\overline{\nabla_n \Phi_f}$ is the *uncorrected* normal gradient from the two values of the two cells sharing the face. The explicit part is computed from Eq. 3.13 as:

$$\nabla \Phi_f^{n-1} = w \nabla \Phi_P^{n-1} + (1 - w) \Phi_N^{n-1} \tag{3.15}$$

being $w = 0.5$ in this work; the normal gradient is computed as:

$$(\overline{\nabla_n \Phi})_f^{n-1} = \nabla \Phi_f^{n-1} \cdot \boldsymbol{n}_f^{n-1}. \tag{3.16}$$

- *Gradient terms*: have been discretized by the Green-Gauss theorem:

$$\nabla \phi_P = \frac{1}{V_P} \sum_f \phi_f \boldsymbol{S}_f \tag{3.17}$$

being V_P the volume of the polyhedral cell P, and $\boldsymbol{S}_f$ the surface vector of the f-th face of the cell.

- *Non-linear terms (convective terms)*: the convective term in momentum balance is linearized with the Picard approach: the mass flux ϕ is treated explicitly and the non-linear term is approximated by:

$$\phi u_{j,rel} \simeq \phi^{n-1} u_j^n - \varphi_M^{n-1} \tag{3.18}$$

the index n in Eq. 3.18 denotes that the values are taken from the result of the previous outer iteration of the segregated solver. A technique for momentum-based interpolation of mass fluxes on cell faces [57] is used to mimic the staggered-grid discretization to prevent checkerboard effects. Using the divergence theorem, the convective terms are rewritten as:

$$\int_V \nabla \cdot (\boldsymbol{u}\boldsymbol{u}) \simeq \sum_f \phi_f (u_f - u_b) \tag{3.19}$$

The velocity u_f is interpolated with the same approach presented in Eq. 3.13, while a second-order central differencing scheme is used for the fluxes.

- *Dynamic mesh*: in a dynamic grid, the position of the cell centers changes from one time step to the next one. Linear extrapolation has been used for mapping cell-centered quantities from the old to the new mesh, to favor the convergence rate of the solver:

$$\phi(\boldsymbol{x}^n, t^{n-1}) = \phi(\boldsymbol{x}^{n-1}, t^{n-1}) + \nabla\phi(\boldsymbol{x}^{n-1}, t^{n-1}) \cdot (\boldsymbol{x}^n - \boldsymbol{x}^{n-1})$$

  However, remapping of fields defined over the faces of the CVs cannot be applied to extensive quantities, as it strongly influences the conservation (and the convergence rate) of the p-U algorithm. To ensure conservation, fields in the CV faces are interpolated from the values in the CV centers, and a Helmholtz-like equation is then solved to ensure that the remapped state is fully conservative.

3.4. GCL: Geometric Conservation Law

When the cell moves rigidly and all the faces have the same velocity, no additional consideration is required and one can proceed to solve the governing equations just by using the relative velocities in the convective term. However, when the CV deforms, special attention must be paid to the calculation of the mass fluxes. In fact, if the grid velocities are used directly to compute the convective fluxes, conservation is not necessarily ensured. To prove this fact, one can consider the continuity equation discretized with the implicit Euler time integration, following [22]. For simplicity, let us consider the CV as a rectangle and that the fluid is incompressible and moves with a constant velocity $\boldsymbol{U} = (u, v)$. The faces move with constant (but different) velocities $\boldsymbol{U_b}$, like shown in Fig. 3.2, so that each one of the faces (named n, s, e, w) will have a constant velocity ($v_{b,n}$, $v_{b,s}$, $u_{b,e}$, $u_{b,w}$), respectively.

For this CV, the discretized equation becomes:

$$\begin{aligned}\frac{\rho\,(V^{n+1} - V^n)}{\Delta t} &+ \rho\left[(u - u_{b,e}) - (u - u_{b,w})\right]^{n+1} (\Delta y)^{n+1} \\ &+ \rho\left[(v - v_{b,n}) - (v - v_{b,s})\right]^{n+1} (\Delta x)^{n+1} = 0\end{aligned} \tag{3.20}$$

Since the fluid velocities are constant, the equation reduces to:

$$\frac{\rho\,(V^{n+1} - V^n)}{\Delta t} - \rho(u_{b,e} - u_{b,w})(\Delta y)^{n+1} - \rho(v_{b,n} - v_{b,s})(\Delta x)^{n+1} = 0 \tag{3.21}$$

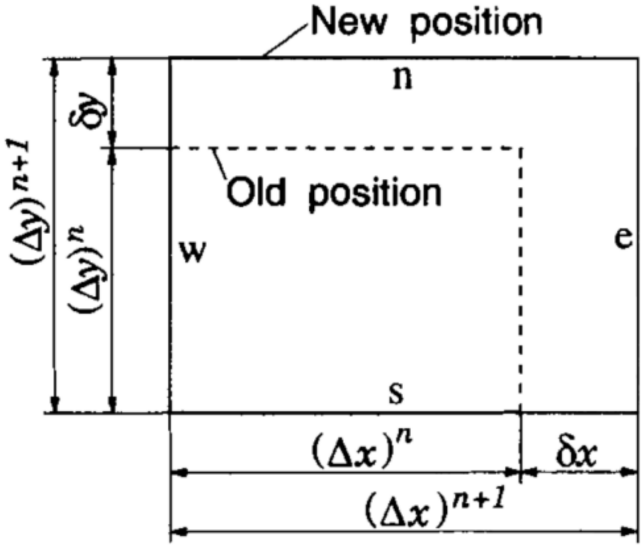


Figure 3.2: Deforming rectangular CV [22]

Following the notation in Fig. 3.2, the deformation along each axis can be introduced as:

$$\begin{aligned} \delta x &= \Delta t \left(u_{b,e} - u_{b,w}\right) \\ \delta y &= \Delta t \left(v_{b,n} - v_{b,s}\right) \end{aligned} \tag{3.22}$$

and the variation of the volume can be computed as:

$$\begin{aligned} V^{n+1} &= (\Delta x \Delta y)^{n+1} \\ V^{n} &= [(\Delta x)^{n+1} - \delta x][(\Delta y)^{n+1} - \delta y] \end{aligned} \tag{3.23}$$

If the values from Eq. 3.23 are substituted in Eq. 3.21, it can be seen how the mass conservation equation is no longer satisfied, introducing a spurious mass source:

$$\delta \dot{m} = \frac{\rho \delta x \delta y}{\Delta t} = \rho \left(u_{b,e} - u_{b,w}\right) \left(v_{b,n} - v_{b,s}\right) \Delta t \tag{3.24}$$

which corresponds with a first-order discretization error, as it is proportional to the time step. This error will accumulate with time as, every time the cell deforms, new mass sources will be introduced in the solution. This situation applies to other schemes, like the Second Order Backwards Scheme or the Crank Nicolson, specially when the fluid and grid velocities are not constant.

In order to restore mass conservation, an additional equation must be enforced: the Geometric

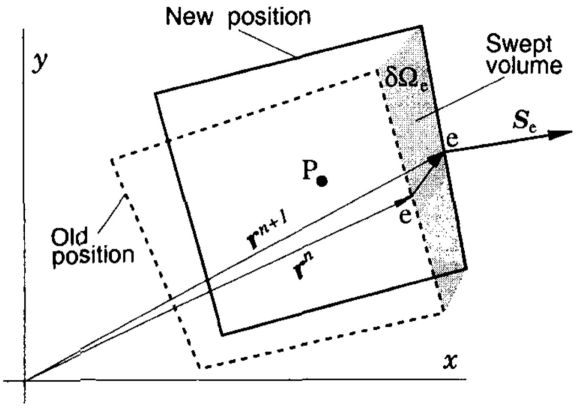


Figure 3.3: Shaded area represents volume swept by face e [22]

Conservation Law (GCL):

$$\frac{d}{dt}\int_{\tilde{\Omega}(t)} \mathrm{d}V - \int_S \boldsymbol{U}_\mathrm{b} \cdot \mathrm{d}\boldsymbol{S} = 0 \tag{3.25}$$

The GCL equation can be seen as a conservation equation for the space. It is equivalent to a conservation law of a fluid with zero velocity and, in continuous form, does not add any constrain to the problem as it is implicitly satisfied. If the semi-discrete counterpart (DGCL) of Eq. 3.25 is considered, it follows:

$$\frac{dV}{dt}\bigg|_{t=n-1}^{t=n} - \sum_f (\boldsymbol{U_b} \cdot \boldsymbol{n})_f S_f = 0 \tag{3.26}$$

Introducing the definition of mesh flux from Eq. 3.11, one obtains:

$$\frac{dV}{dt}\bigg|_{t=n-1}^{t=n} - \sum_f \varphi_{M,f} = 0 \tag{3.27}$$

introducing a restriction on the numerical schemes. More concretely, the calculation of mesh fluxes is defined by the temporal discretization. To illustrate this fact, let us consider the implicit Euler scheme for simplicity, even though its extension to higher order schemes is straightforward, as it will be shown later. In this case, the midpoint rule and central-difference schemes are used for spatial integration. Considering the domain of Fig. 3.4, the DGCL can be written as:

$$\frac{V^{n+1} - V^n}{\Delta t} = \sum_f (\boldsymbol{U_b} \cdot \boldsymbol{n})_f S_f, \quad with\ f = n, s, e, w \tag{3.28}$$

The volume difference can be computed as the sum of the volume swept from t^n to t^{n+1} by

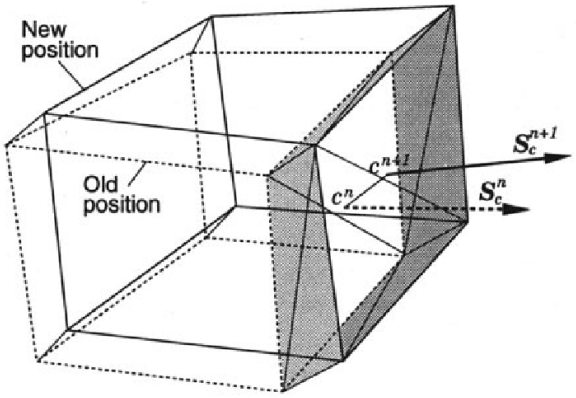


Figure 3.4: Volume swept by a face in a 3D CV [22]

each one of the CV faces $\delta\Omega_f$:

$$\frac{V^{n+1} - V^n}{\Delta t} = \frac{\sum_f \delta\Omega_f}{\Delta t} = \sum_f \dot{\Omega}_f \tag{3.29}$$

and comparing theses equations for one of the CV faces, one obtains:

$$\dot{\Omega}_f = (\boldsymbol{U_b} \cdot \boldsymbol{n})_f S_f = \varphi_{M,f} \tag{3.30}$$

stating clearly how the mesh flux of a CV face represents the volume swept by this face during the current time step. The problem now remains in the calculation of the velocity of each CV face center of the grid. If the position of every CV of the grid is know at every instant of the simulation, one could compute $\boldsymbol{U_b}$ explicitly like:

$$\boldsymbol{U}_{b,\boldsymbol{f}} \approx \frac{\boldsymbol{r_f}^{n+1} - \boldsymbol{r_f}^n}{\Delta t} \tag{3.31}$$

which will pose no problems when the grid moves in only one direction. However, if there is motion in more than one direction, this approach may not work, generating spurious mass sources. If the mesh flux is computed as the volume swept by each of the CV faces, the exact calculation can be performed, even for large time steps. The procedure to compute it is explained in the following section. If fully-implicit time integration schemes are considered, once the value of the mesh flux on each face is computed, all the terms of the transport equations are known and one can proceed to compute the solution in the next time step.

Swept Volume Calculation

Previously, it was stated that the mesh flux of a face can be calculated as the volume swept by this face during the time step (see Eq. 3.30). As the edges of the cell may turn during the motion, the swept volume is computed following the same procedure used to calculate the volume of a cell. For this, besides triangulating the old and new positions of the face, an additional triangulation of the areas swept by the edges (shaded areas in Fig. 3.4) is also needed. As these regions are shared with neighboring cells, the exact same triangulation must be made for both cells in order to guarantee that the space is conserved.

3.5. Solution Method: PIMPLE Algorithm

The DGCL provides the numerical scheme to compute the convective fluxes in the transport equations, for a given temporal scheme. With that, all terms in the transport equations can be discretized in a given mesh, forming an algebraic system of equations. To solve it, an iterative procedure has been followed. In this work, a compressible flow with frozen chemistry has been considered, so the energy equation has to be solved as well. For that, the traditional PIMPLE algorithm has been chosen (see Algorithm 3.1).

The coupling of momentum and continuity equation proceeds as follows. First, the momentum equation is discretized as:

$$\frac{\partial}{\partial t} V \rho \boldsymbol{U} + \sum_f \rho_f \left[\!\left[\boldsymbol{U}^n\right]\!\right]_f \left(\varphi_f^{n-1} - \varphi_{M,f}^n\right) - \sum_f \mu \operatorname{dev} \nabla \left[\!\left[\boldsymbol{U}^n\right]\!\right]_f \cdot \boldsymbol{n}\, S_f = -\sum_f \left[\!\left[p^{n-1}\right]\!\right]_f \boldsymbol{n}\, S_f \tag{3.32}$$

and it is solved using the pressure field computed at the previous iteration (or its initial value); then, mass conservation is achieved by combining the continuity and momentum equations into a discrete equation for the pressure:

$$\frac{\partial(\psi\, p)V}{\partial t} - \sum_f \left[\!\left[\frac{1}{\mathcal{A}_p}\right]\!\right]_f \left[\!\left[\nabla p^n\right]\!\right]_f \cdot \boldsymbol{n} S_f = \sum_f \rho_f \varphi_f^* \tag{3.33}$$

where $\psi = \frac{1}{RT}$ is obtained from the ideal gas law and the intermediate fluxes φ_f^* are computed using a single Jacobi iteration on the discretized momentum equation derived of the pressure gradient:

$$\varphi_f^* = \left[\!\left[\rho \frac{\mathcal{H}(\mathcal{U})}{\mathcal{A}_p}\right]\!\right]_f \cdot \boldsymbol{n}\, S_f \tag{3.34}$$

In the above Eq. 3.34, $\mathcal{A}_p$ are the diagonal coefficients of the matrix resulting from discretization

of Eq. 3.8, while $\mathcal{H}(\mathcal{U}) = \sum_l a_l \boldsymbol{U}_l^n$ are the off-diagonal contributions to velocity $\boldsymbol{U}_p$. Both have been interpolated on cell faces following the Rhie-Chow procedure [57] to avoid "checkerboarding" problems in the results. Solution of Eqs. 3.32 and 3.33 through Eq. 3.34 is iterated several times until convergence. Finally, continuity-preserving fluxes and velocities are computed independently:

$$\varphi_f^{n+1} = \varphi_f^* - [\![\nabla p^n]\!]_f \left[\!\!\left[\frac{1}{\mathcal{A}_p}\right]\!\!\right]_f S_f \tag{3.35}$$

$$\boldsymbol{U}^{n+1} = \frac{\mathcal{H}(\mathcal{U})}{\mathcal{A}_p} - \frac{1}{\mathcal{A}_p}\nabla p^n \tag{3.36}$$

Algorithm 3.1 PIMPLE loop.

1: Set initial value of variables at t^n
2: **while** Number of outer correctors && error > tolerance **do**
3: Assemble and solve momentum equation (Eq. 3.32): $\boldsymbol{U}^*$
4: **while** Number of inner correctors && error > tolerance **do**
5: Compute φ_f^n using the Rhie-Chow interpolation to obtain a momentum satisfying mass flux (Eq. 3.34): φ_f^*
6: Assemble and solve energy equation, if present.
7: Assemble and solve pressure correction equation with φ_f^* (Eq. 3.33): p'
8: Update pressure and velocity fields and mass fluxes to obtain continuity-satisfying fields (Eqs. 3.35 and 3.36)
9: **end while**
10: Update turbulence model variables and thermophysical transport quantities.
11: Update (Eq. 3.32) with the new values of p and φ_f
12: **end while**
13: Update all fields and advance in time

4 | Finite-Volume ALE Scheme for Dynamic Meshes with Topology Changes

The first objective of this section is to explain how topology changes are introduced in a dynamic grid when new cells are added or removed from the computational mesh. Firstly, the mesh topology is changed and a conservative remapping is applied, followed by the mesh motion in a mesh with constant topology. Then, the temporal discretization of the equations is introduced, with particular interest on second-order schemes. For them, an *equivalent ghost state* is generated, allowing to advance the solution in time. The Second Order Backward Euler and Crank-Nicolson schemes are discussed. Finally, an extension to Adaptive Mesh Refinement techniques is discussed.

4.1. Introduction

In problems formulated with the Arbitrary Lagrangian-Eulerian framework, the velocity of the mesh points can be imposed independently of the fluid velocity. If the displacement of the points becomes too large, the mesh may quickly get distorted and produce some invalid elements in the mesh. A possible solution for this case would be to change the grid topology, changing the number of entities of the grid (points, faces or cells) or the connectivity between them. A discussion on different topology changes, including connectivity variations, sliding interfaces, or element redefinition, can be found in [45, 54, 55]. In the FV formulation, the interest falls on how the topology change affects the calculation of volume integrals and face fluxes. In the present work, only the topological changes involving a variation in the number of grid cells are considered, because of their connection to the time discretization. In the following, Ω will represent the discrete approximation of the domain volume $\tilde{\Omega}(t)$, i.e. the subdivision of $\tilde{\Omega}(t)$ in a finite number of polyhedral, non-overlapping Control Volumes or cells V_i such that $\{V_i\} = \Omega$. Each polyhedral CV is delimited by an arbitrary number of faces $\{f_j\} = \partial V_i$, so that adjacent

cells share the same faces. The shorthand Ω_k^n will be used to identify the discretization Ω, whose topology is k, at time t^n. The value of a quantity ϕ in the cell i belonging to the discretization Ω_k^n will be expressed as $\phi_{i,k}^n$. If the topology change involves a variation in the number of grid cells, a remapping has to be made before solving the governing equations in the updated mesh. In a co-located variable arrangement, the commonly applied mesh-to-mesh mapping of intensive variables between Ω_{k-1}^n and Ω_k^n is no longer trivial as the one-to-one correspondence of cells is missing.

To perform a topology change in a dynamic grid, a two-step procedure has been followed, decoupling the topology change from the mesh motion. First, the topology change is performed on a static grid ($\Omega_{k-1}^{n-1} \rightarrow \Omega_k^{n-1}$), mapping the solution from the cell centers of the initial topology to the new ones ($\phi_{i,k-1}^{n-1} \rightarrow \phi_{j,k}^{n-1}$). Then, the value of the fluxes in the faces are computed by interpolation of the values in the cell centers and conservativeness is restored by solving a Helmholtz-like equation. Like this, an *equivalent state* is obtained: all the physical fields at t^{n-1} are now expressed in a grid with a different topology. From this equivalent state, the mesh can be moved following the traditional procedure as no topology change is involved ($\Omega_k^{n-1} \rightarrow \Omega_k^n$):

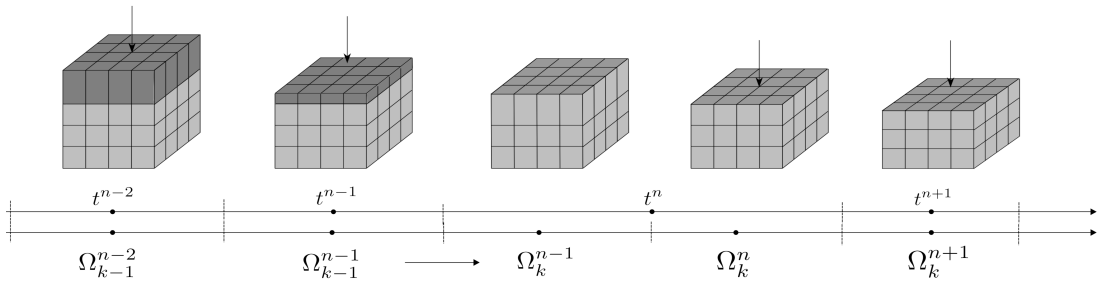
$$\Omega_{k-1}^{n-1} \rightarrow \Omega_k^n = \underbrace{(\Omega_{k-1}^{n-1} \rightarrow \Omega_k^{n-1})}_{\text{topology change}} + \underbrace{(\Omega_k^{n-1} \rightarrow \Omega_k^n)}_{\text{mesh motion}} \tag{4.1}$$

This procedure is illustrated in Fig. 4.1 for the particular case of dynamic cell layering, even though the methodology applies to any cell shape, mesh motion or topology change. In Fig. 4.1a, cell volumes are removed while the mesh is moved; in Fig. 4.1b, cell volumes are added during motion in the time interval $\Delta t = t^n - t^{n-1}$.

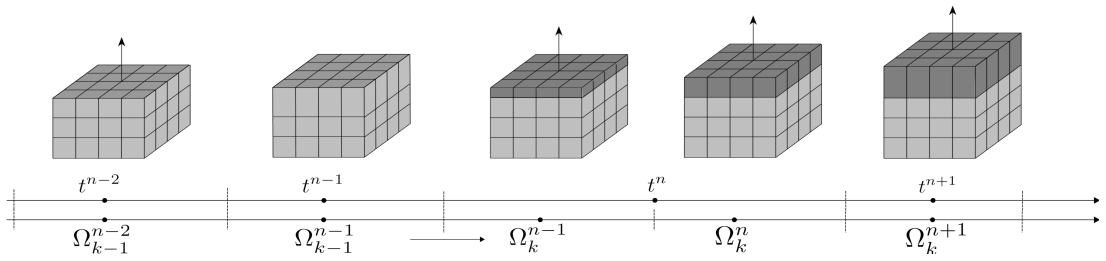
4.2. Conservative Remapping

The mapping from one mesh to the other is typically done with a linear interpolation. The fields computed in the cell centers of the old mesh are interpolated to the cell centers of the new mesh. From these values, the fluxes in the faces of the new mesh can be calculated again by interpolation. Linear interpolation is usually preferred, but higher order interpolation could also be used. It has to be noticed that the resulting field does not necessarily satisfy the transport equations, and therefore, continuity is not guaranteed. Continuity is restored by solving a Helmoltz-like equation for a pressure corrector p':

$$\frac{\partial \psi p'}{\partial t} - \nabla \cdot \left[\frac{1}{\mathcal{A}_p} \nabla p' \right] = \nabla \cdot (\llbracket \rho \boldsymbol{U} \rrbracket_f)_k^n - \nabla \cdot (\varphi)_k^n \tag{4.2}$$



(a) volume removal



(b) volume addition

Figure 4.1: Two-step execution of the dynamic cell layering. (a) cell compression is applied by the upper moving boundary and removal of cell layers is triggered at $t^{n-1} \rightarrow t^{n}$; (b) the upper boundary expands the neighboring cell and layer addition at $t^{n-1} \rightarrow t^{n}$ is triggered. Ω_k^n is the FV discretization of the domain; n is the global temporal index and k the index referring to the mesh topology.

which is eventually used to re-adjust the mapped fluxes as suggested by [31]:

$$\varphi^{(n+1)} = \varphi\big(t^{(n)}, \boldsymbol{x}^{(n+1)}\big) + \frac{1}{\mathcal{A}_p} \nabla p' \tag{4.3}$$

where $\psi = \frac{1}{RT}$ is obtained from the equation of state of an ideal gas. Finally, before solving the governing equations on the updated mesh, old values of $\boldsymbol{u}_k^n$, p_k^n and ρ_k^n must still satisfy continuity when they are remapped onto the new grid [31]: the old velocity field $\boldsymbol{u}_{k-1}^n$ might not be compliant with the continuity equation, when it is re-sampled onto the new mesh. Therefore, a modified form of Poisson equation needs to be solved for a pressure corrector p_{corr}:

$$\nabla^2 p_{\text{corr}} + \frac{1}{\Delta t} \nabla \cdot \left(\rho_k^n \cdot \boldsymbol{u}_k^n\right) = 0 \tag{4.4}$$

The differential equation (Eq. 4.4) must be completed with appropriate boundary conditions. On solid walls they have to be of Neumann type ($\partial p_{\text{corr}}/\partial n = 0$), whereas on permeable walls a Dirichlet boundary condition is applied ($p_{\text{corr}} = 0$). The pressure correction problem assumes therefore the following form:

$$\begin{cases} \nabla^2 p_{\text{corr}} + \dfrac{1}{\Delta t} \nabla \cdot \left(\rho_k^n \cdot \boldsymbol{u}_k^n \right) = 0 \\ \dfrac{\partial p_{\text{corr}}}{\partial n} = 0 \qquad \text{on solid boundaries} \end{cases} \tag{4.5}$$

4.3. Temporal Discretization with Topology Changes

The procedure described in Sec. 4.2 is well established in CFD solvers supporting mesh motion and topology changes [5, 61, 45, 52], when the first-order implicit Euler scheme is applied for time differencing in a generic flow transport equation:

$$\frac{\partial}{\partial t} \int_{\tilde{V}} \phi \, \mathrm{d}V \approx \frac{V_k^n \, \phi_k^n - V_{k-1}^{n-1} \, \phi_{k-1}^{n-1}}{\Delta t} \tag{4.6}$$

The transported variables ϕ_{k-1}^{n-1} across the topology change must be computed to ensure conservation with the ALE scheme. This requires particular attention as new volumes to the mesh are added (transition $\Omega_{k-1}^{n-1} \rightarrow \Omega_k^{n-1}$ in Fig. 4.1b). This can be done in two ways:

- with cell inflation: it is assumed that cell faces at t^{n-1} are duplicated to generate new zero-volume cells, that are then inflated to form the new cells at Ω_k^{n-1}. In Eq. 3.27:

$$V_k^{n-1} = 0 \tag{4.7}$$

  and the local topology change (volume addition and mesh motion) is accounted by the mesh fluxes of the newly added faces from position $\Omega_k^{n-1} \rightarrow \Omega_k^n$, that are computed as:

$$\sum_f \varphi_{M,f_i} = \frac{\Delta \boldsymbol{x}_{f_i} \cdot \boldsymbol{S}_f}{\Delta t} \tag{4.8}$$

  where $\Delta \boldsymbol{x}_{f_i}$ is the mesh flux corresponding to the volume swept by each face from state Ω_k^{n-1} to state Ω_k^n.

- without cell inflation: if the newly added faces are assumed to be inserted in their final positions at state Ω_k^{n-1}, their mesh fluxes will be zero:

$$\varphi_{M,f_i} = 0 \tag{4.9}$$

  and the local topology change will be accounted in the equation by a conservative remapping of the cell quantities between to different grids. Same reasoning can be applied if a static

face is being removed.

After this step, mesh fluxes are computed between $\Omega_k^{n-1} \rightarrow \Omega_k^n$ as in any deforming mesh. Once the mesh fluxes are updated, direct mapping of the primary variables is applied onto the updated topology, while fluxes are adjusted to ensure conservativeness. With first order temporal schemes, the described steps ensure the fulfillment of the DGCL during mesh motion with topology changes. On the other hand, if second order accurate temporal schemes are used, primary variables and volumes at the old-old time t^{n-2} are required (ϕ_k^{n-2}). However, transported variables $\phi_{i,k-1}^{n-2}$ are only available at t^{n-2} on the old topology Ω_{k-1}^{n-2}: a way of handling $\phi_{k-1}^{n-2} \rightarrow \phi_k^{n-2}$ is required.

4.4. Second-Order Temporal Discretization With Dynamic Mesh Refinement

With second-order time schemes and in presence of topology changes involving addition of control volumes (namely Adaptive Mesh Refinement and dynamic addition of cell layers [45]), all the primary variables and the cell volumes at time t^{n-2} must be estimated on the updated mesh topology k: ϕ_k^{n-2} (see Fig. 4.1b). In the following, this state will be referred as *equivalent ghost state* Ω_k^{n-2}. It has to be noticed that this equivalent ghost state is only needed for the cells being added or deleted and their neighboring ones. The rest of the mesh does not get influenced by the local topology change, which reduces the computational time as very few operations have to be performed in a very limited number of cells. While Adaptive Mesh Refinement and dynamic addition of cell layers are different techniques, the same theory applies to ensure that the DGCL is properly conserved with second-order time differencing schemes. In the following, it will be discussed how to apply Second-Order Backward Euler (SOBE) and the Crank-Nicolson (CN) differencing schemes to discretize the temporal derivatives in presence of dynamic, topologically changing grids.

4.4.1. Second-Order Backward Euler Scheme (SOBE)

The Second-Order Backward Euler is a two-step Adams-Moulton method [49, 50], where a linear combination of the current-time and the old-time derivatives is used in the left hand side:

$$\frac{3}{2}\frac{V^n\phi^n - V^{n-1}\phi^{n-1}}{\Delta t} - \frac{1}{2}\frac{V^{n-1}\phi^{n-1} - V^{n-2}\phi^{n-2}}{\Delta t} = V^n\mathcal{F}(\phi^n) \tag{4.10}$$

That yields the following expression for the transient term:

$$\frac{\partial}{\partial t}\int_{\tilde{\Omega}(t)}\phi\,\mathrm{d}V \approx \frac{1}{\Delta t}\left(\frac{3}{2}V^n\phi^n - 2V^{n-1}\phi^{n-1} + \frac{1}{2}V^{n-2}\phi^{n-2}\right) \tag{4.11}$$

The SOBE method is formally second-order and unbounded [27]. The DGCL for the SOBE takes the form:

$$\frac{\frac{3}{2}V_k^n - 2V_k^{n-1} + \frac{1}{2}V_k^{n-2}}{\Delta t} = \frac{3}{2}\frac{(V_k^n - V_k^{n-1})}{\Delta t} - \frac{1}{2}\frac{(V_k^{n-1} - V_k^{n-2})}{\Delta t} = \sum_f \varphi_{M,f}^n \tag{4.12}$$

If the cell volumes are added without inflation, it holds $V_k^n = V_k^{n-1} = V_k^{n-2}$ and mesh fluxes on the newly added faces are zero. If static cell volumes are deleted without inflation, the removed faces will have nil mesh flux and the volumes can be added to the value of the deforming ones. The DGCL will be naturally satisfied and no spurious oscillation in the solution will appear. Despite an uniform time step has been considered in the equations, the application of the equations to variable time-steps is straight forward and does not impact the validity of the methodology.

4.4.2. Crank-Nicolson Scheme (CN)

The Crank Nicolson (CN) [13] temporal scheme is a second-order accurate and bounded scheme. To derive this scheme, let us consider the case of a traditional transport equation:

$$\frac{\partial\phi}{\partial t} = \mathcal{F}(\phi) \tag{4.13}$$

where $\mathcal{F}(\phi)$ represent the spatial operator. One could divide the time-step in two halves and use the implicit Euler scheme to solve the first half:

$$\frac{\phi^{n-\frac{1}{2}} - \phi^{n-1}}{\frac{\Delta t}{2}} = \mathcal{F}(\phi^{n-\frac{1}{2}}) \tag{4.14}$$

and the explicit Euler scheme to solve the second one:

$$\frac{\phi^{n} - \phi^{n-\frac{1}{2}}}{\frac{\Delta t}{2}} = \mathcal{F}(\phi^{n-\frac{1}{2}}) \tag{4.15}$$

Both schemes provide only first order accuracy. However, if they are added, one can get:

$$\frac{\phi^n - \phi^{n-1}}{\Delta t} = \mathcal{F}(\phi^{n-\frac{1}{2}}) = \frac{1}{2}\left[\mathcal{F}(\phi^n) + \mathcal{F}(\phi^{n-1})\right] \tag{4.16}$$

The temporal term is treated as the implicit Euler scheme, whereas the spatial term $\mathcal{F}(\phi)$ is treated in a semi implicit manner. However, if the Eq. 4.13 evaluated at t^{n-1} is substituted in Eq. 4.16, one can get:

$$\frac{\phi^n - \phi^{n-1}}{\Delta t} = \frac{1}{2}\left[\mathcal{F}(\phi^n) + \frac{\partial \phi^{n-1}}{\partial t}\right] \tag{4.17}$$

where $\frac{\partial \phi^{n-1}}{\partial t}$ is known from the previous time step and corresponds to the evolution of ϕ between t^{n-2} and t^{n-1}. Re-arranging the equation, one can write:

$$2 \cdot \frac{\phi^n - \phi^{n-1}}{\Delta t} - \frac{\partial \phi^{n-1}}{\partial t} = \mathcal{F}(\phi^n) \tag{4.18}$$

This formulation requires to compute $\frac{\partial \phi^{n-1}}{\partial t}$ instead of $\mathcal{F}(\phi^{n-1})$. The time derivative at the previous time step $\frac{\partial \phi^{n-1}}{\partial t}$ is not readily available but it can be easily calculated with the values of ϕ^{n-2} as:

$$\frac{\partial \phi^{n-1}}{\partial t} = 2 \cdot \frac{\phi^{n-1} - \phi^{n-2}}{\Delta t^{n-1}} - \frac{\partial \phi^{n-2}}{\partial t} \tag{4.19}$$

where the values of $\frac{\partial \phi^{n-2}}{\partial t}$ have been stored in memory the previous time step using this same equation.

In order to increase the stability of the scheme, the method is usually blended with an implicit Euler scheme. This weighting/blending coefficient is called the "off-centering coefficient" $\theta \in [0; 1]$. Introducing θ in the previous equations, it follows:

$$(1+\theta) \cdot \frac{\phi^n - \phi^{n-1}}{\Delta t} - \theta \frac{\partial \phi^{n-1}}{\partial t} = \mathcal{F}(\phi^n) \tag{4.20}$$

where $\theta = 1$ recovers the original CN, while the Euler scheme applies with $\theta = 0$. The value of the off-centering coefficient is defined by the user and it has been set to 1 in this work to obtain the pure Crank Nicolson scheme.

With dynamic topologically changing grids, the Crank-Nicolson scheme must be reformulated to account for the variation of cell volumes during the different time steps. In particular, the transient term of the N-S equations can be therefore rewritten as:

$$\frac{\partial}{\partial t}\int_{\tilde{\Omega}(t)} \phi \, \mathrm{d}V \approx \frac{(1+\theta)}{\Delta t} \cdot \left[V^n \phi^n - V^{n-1}\phi^{n-1}\right] - \theta \left[\left(\frac{\partial \phi}{\partial t}\right) V\right]^{n-1} \tag{4.21}$$

with the previous time derivative being evaluated as:

$$\left[\left(\frac{\partial \phi}{\partial t}\right) V\right]^{n-1} = \frac{(1+\theta_0)}{\Delta t_0} \cdot \left[V^{n-1}\phi^{n-1} - V^{n-2}\phi^{n-2}\right] - \theta_0 \left[\left(\frac{\partial \phi}{\partial t}\right) V\right]^{n-2} \tag{4.22}$$

where the subscript 0 refers to the value of the coefficients at the previous time step. Similarly to the SOBE, the CN scheme also requires the value of the variables at t^{n-2}, so a similar procedure can be followed: old-old fields must be computed on the updated topology (equivalent ghost state, Ω_k^{n-2}). With the CN scheme, there is also the need of compute $\frac{\partial\phi}{\partial t}\big|_k^{n-2}$, that expresses the variation of the field ϕ_k in the time interval $\Delta t^{n-2} = t^{n-2} - t^{n-3}$ in the grid with the new topology. This, in turn, requires to know the values of the derivatives from previous times recursively until the beginning of the simulation. Therefore, the introduction of a topology change will inevitably create a discontinuity in the solution. With local mesh refinements, only a limited amount of cells will change their volumes. As the evaluation of the time derivatives with the Crank-Nicolson scheme depends on the evolution of the cell volumes, the quantity $\frac{\partial\phi}{\partial t}\big|_k^{n-2}$ also needs to be remapped onto the currently updated topology. If a cell refinement is triggered, the estimation of $\frac{\partial\phi}{\partial t}\big|_k^{n-2}$ is not trivial: a direct mapping of the field from the old topology does not ensure conservativeness, while the calculation of that term would require to reconstruct the history of the field on the recent topology reconstructed from the beginning of the simulation. This is clearly extremely demanding to do in some cases, and impossible to do with complex geometries. A possible solution to the problem consists of setting $\frac{\partial\phi}{\partial t}\big|_k^{n-2} = 0$ in Eq. 4.22 when a topology change is triggered, for the cells that are deforming, right after remapping. This simplification also translates to the DGCL, as follows:

$$\frac{(1+\theta)}{\Delta t}\cdot\left[V_k^n - V_k^{n-1}\right] - \frac{(1+\theta^0)}{\Delta t^0}\cdot\left[V_k^{n-1} - V_k^{n-2}\right] + \theta^0 V_k^{n-2}\,\cancelto{0}{\frac{\partial(\phi)_k^{n-2}}{\partial t}} = \sum_f \varphi_{M,f}^n = \varphi_{M,\text{moving}}^n \tag{4.23}$$

where $\varphi_{M,\text{moving}}^n$ corresponds to the mesh flux of the moving face, whose value is not affected by the topology change. This implies an imbalance between the mesh flux and the value of V_k^{n-2}, which provokes that the DGCL is no longer satisfied. However, V_k^{n-2} has been corrected to fulfill exactly the DGCL with the approximations introduced, like:

$$\widetilde{V}_k^{n-2} = V_k^{n-1} - \frac{\Delta t^0}{\Delta t}\left(\frac{1+\theta}{1+\theta^0}\right)\cdot\left[V_k^n - V_k^{n-1}\right] + \frac{\Delta t^0}{1+\theta^0}\cdot\varphi_{M,\text{moving}}^n \tag{4.24}$$

The corrected $\widetilde{V}_k^{n-2}$ of the cells in the dynamic layer will not fill all the domain completely and uniquely. However, as the topology of the equivalent ghost state is not being updated, this simplification is equivalent to changing just the value of the $\widetilde{V}_k^{n-2}$ in the cells belonging to the dynamic layer and will not have any consequences in the following time-steps. In other words, $\widetilde{V}_k^{n-2}$ is used to solve the governing equations in the newly added cells and compensates

the simplification introduced. In this way, all the information required to calculate the time derivative is available and the current solution at t^n can be computed.

4.4.3. Adaptive Mesh Refinement (AMR)

So far, the focus has been put on topological changes involving the addition or removal of layers of cells. However, this technique also applies with Adaptive Mesh Refinement, where only a single cell may be refined or unrefined according to some user defined criteria. In static grids no additional constrains are required and the normal procedure can be used as there is no need to ensure the Geometric Conservation Law. Similar reasoning can be applied if all the grid moves rigidly, without any cell being deformed.

The focus then is to apply refinement or unrefinement to a deforming cell. The procedure is similar to the one explained for a layer of cells in Fig. 4.1, if only one face of the cell moves along one of the logical axis of a structured grid. If all the added or removed faces during the refinement are static or deform along its own plane, such that their mesh flux is nil, the same procedure can be applied directly without any further consideration. The DGCL will be solved and the *equivalent ghost state* will again be calculated.

5 | Numerical Results

The methodology to use second order temporal schemes when the number of cells changes in dynamic grids has been tested in three different problems. First, the one-dimensional Uniformly Accelerated Piston case is examined because it presents an analytical solution that allows to quantify the accuracy of the methodology. Then, a three-dimensional lid-driven cavity test case has been considered, where a region of cells moves periodically in the domain. Finally, an industrial geometry has been considered. In particular, an Internal Combustion Engines has been chosen because it presents the relative motion of three different components (two valves and a piston) that undergo a large movement. More concretely, the TCC-III engine has been chosen as it represents a benchmark case for IC engines.

5.1. Uniformly Accelerated Piston

The Uniformly Accelerated Piston (UAP) test-case [35], for which the analytical solution is available, has been selected as numerical experiment to validate the theory proposed in this work. In the experiment, the wave propagation of the compressible flow inside a cylinder of infinite length is forced by the uniformly accelerated motion of one of its ends (the piston). A schematic of the problem setup is shown in Fig. 5.1. When the piston starts moving with constant acceleration, a pressure wave moving with a velocity c_0 over a gas at rest is formed, where c_0 represents the undisturbed speed-of-sound. If the piston has positive acceleration, a compression wave is formed; otherwise, a rarefaction wave is observed. The region of the cylinder where $x > c_0 t$ is at rest as it has not been influenced by the pressure wave yet. On the piston surface, the velocity of the gas is the same of the piston: $v_p = \pm\, at$ in $x = \pm \frac{a\, t^2}{2}$, with $x(t = 0) = 0$ and $v_p(t = 0) = 0$. If any effect of the side walls is neglected, gas velocity,

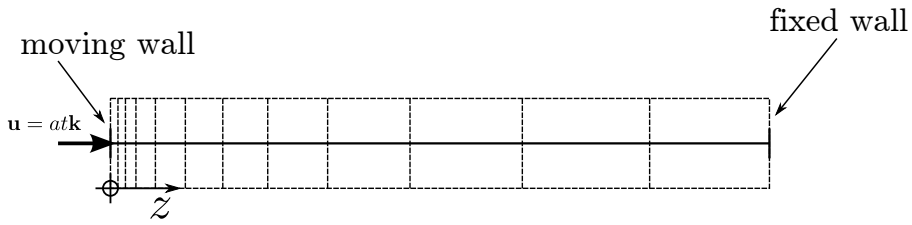


Figure 5.1: Uniformly Accelerated Piston, numerical experiment. A wall boundary (piston) moves along an infinite cylinder with uniform acceleration towards a fixed wall. With the reference frame used, positive acceleration is assumed for compression.

pressure and temperature in the region between the piston and $x < c_0 t$ can be calculated as:

$$U(x,t) = \begin{cases} -\frac{1}{\gamma}\left(c_0 + \frac{\gamma+1}{2}at\right) + \frac{1}{\gamma}\sqrt{\left(c_0 + \frac{\gamma+1}{2}at\right)^2 - 2a\gamma(c_0 t - x)} & \text{if } x \leq c_0 t \\ 0 & \text{if } x > c_0 t \end{cases} \tag{5.1}$$

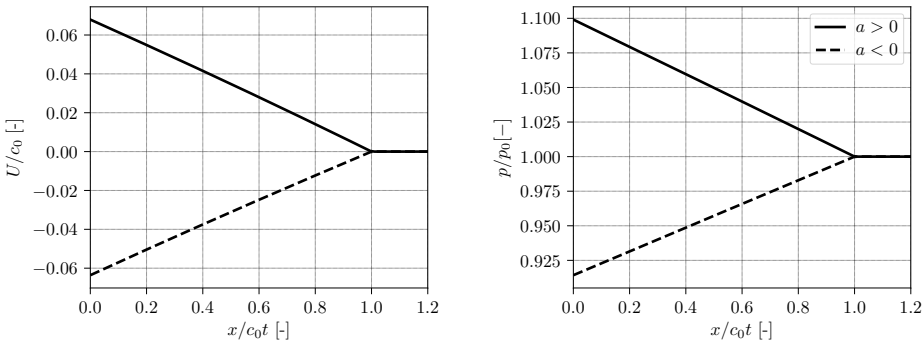
$$p(x,t) = \begin{cases} p_0 \left(1 \pm \frac{\gamma-1}{2}\frac{U}{c_0}\right)^{\frac{2\gamma}{\gamma-1}} & \text{if } x \leq c_0 t \\ p_0 & \text{if } x > c_0 t \end{cases} \tag{5.2}$$

$$T(x,t) = \begin{cases} T_0 \left(1 \pm \frac{\gamma-1}{2}\frac{U}{c_0}\right)^2 & \text{if } x \leq c_0 t \\ T_0 & \text{if } x > c_0 t \end{cases} \tag{5.3}$$

being γ the ratio of specific heats. In a similar way, density and speed of sound can be calculated using the ideal gas model. In Fig. 5.2, analytical velocity and pressure profiles across the wave front in the axis of the cylinder are plotted for positive and negative acceleration, where it is shown its linear evolution. In both cases, the duration of the experiment is limited to a time interval $[0, t_{\text{lim}}]$ to avoid the formation of a shock wave during compression or a void region during expansion. t_{lim} can be expressed as:

$$t_{\text{lim}} = \begin{cases} \frac{2c_0}{(\gamma+1)|a|} & \text{if} \quad a > 0 \\ \frac{2c_0}{(\gamma-1)|a|} & \text{if} \quad a < 0 \end{cases} \tag{5.4}$$

Without any loss of generality, Eqs. 5.1, 5.2 and 5.3 can be applied also to a cylinder of finite length L: in this case the aforementioned solution will be valid either until the end of the



(a) Velocity (b) Pressure

Figure 5.2: Analytical solution of velocity (a) and pressure (b) profiles along the axis of the uniformly-accelerated piston. The moving boundary is located at $x = 0$

cylinder is reached or a shock wave is formed. Therefore, it can be written:

$$t_{\text{lim}} = \begin{cases} \min\left[\dfrac{L}{c_0}, \dfrac{2c_0}{(\gamma+1)|a|}\right] & \text{if} \quad a > 0 \\ \min\left[\dfrac{L}{c_0}, \dfrac{2c_0}{(\gamma-1)|a|}\right] & \text{if} \quad a < 0 \end{cases} \tag{5.5}$$

This analytical solution will be used as a benchmark of the numerical simulations to account for the error introduced by the calculations with specific focus on velocity, pressure and temperature, as it is related to the mass conservation of the problem.

Case Setup and Simulation Strategy

A one-dimensional FV mesh has been used to discretize the UAP geometry of Fig. 5.1. Grid refinement is applied towards the moving wall with a ratio $\Delta x_{\text{max}}/\Delta x_{\text{min}} = 10$. The moving wall is physically translating with velocity $\boldsymbol{u} = at\,\boldsymbol{i}$, with $\boldsymbol{i}$ as the unit vector aligned with the piston axis; the same velocity is set on the moving face to enforce the non-permeability constraint. A free-slip boundary condition is applied on the side walls, while a no-slip boundary condition is applied to the upper wall. The total simulation time is set to avoid that the pressure wave reaches the fixed wall. Zero-diffusion on all boundaries is applied for pressure and temperature. The flow is assumed to be laminar. Pressure-velocity coupling is achieved by a transient compressible solver [45]. Two different strategies for mesh motion are compared (Fig. 5.3), namely: a) mesh motion based on cell stretching. The displacement of each mesh point is inversely proportional to its distance from the moving wall, so all cells undergo a deformation and change their size, while the topology of the mesh does not change; b) rigid

motion of a cell zone with dynamic layering [45] (see Fig. 5.4): only one cell is deforming, while dynamic addition/removal of cells is applied on the cells located at $x/L \approx 0.03$ when $t = 0$. This distance has been chosen to be far enough from the piston and avoid any influence of the boundary, but to allow the cells to be crossed by the evolving wave. Cell layers are dynamically added or removed when the deformation of the cell involves more than 25% of its thickness. Cell count was 10000 using the aforementioned 1 : 10 size ratio between domain ends. Adaptive time-stepping was disabled in this set of simulations, to favor a more accurate comparison among results obtained by different dynamic mesh handlings. In simulations with topology changes, cell addition (during expansion) and cell removal (during compression) were triggered at any time-step; in this way, the effect of the numerics used to handle topology changes could be accounted in the quantification of either the time-step and the global error. During the simulations, the size of the cell added during mesh refinement ensures to keep the mesh as uniform as possible, to minimize the non-uniformity error.

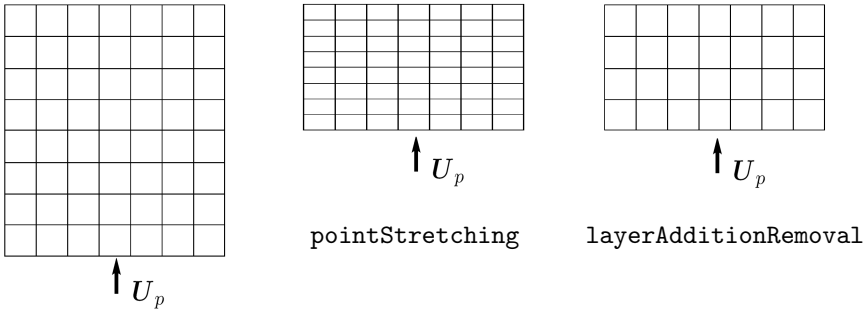


Figure 5.3: Uniformly Accelerated Piston (UAP), comparison between different dynamic mesh handling strategies. Left) discretized domain; center) cell stretching; right) cell layering.

For each configuration tested, the point-wise error has been computed as:

$$e(x,t) = \frac{f(x,t) - \tilde{f}(x,t)}{\tilde{f}(x,t)} \tag{5.6}$$

with $f(x,t)$ being one of $\{U(x,t), p(x,t), T(x,t)\}$ computed by the CFD solver, and $\tilde{f}(x,t)$ the same function evaluated using the analytical formula. The accuracy order has been computed with the normwise error $\|e\|_1(t)$:

$$\|e\|_1(t) = \|f(x,t) - \tilde{f}(x,t)\|_1 \tag{5.7}$$

All quantities have been calculated at the first time step after the first topology change is triggered. This choice has been made in regards of quantifying as accurately as possible the

properties of the topological change and avoiding any correction of the error that may be performed by later iterations. In addition, a longer simulation has also been carried out in order to check if the total mass is conserved after the addition/removal of a large number of cells. Results from simulations based on dynamic mesh handling with topological changes have been compared with the analytical solution and with the results provided by simulations based on a cell stretching strategy. As outlined in Fig. 5.3, the use of cell layering in dynamic mesh handling allows to preserve the same discretization independently on the position of the moving boundary. This allows to maintain the initial mesh quality (uniformity, resolution, non-orthogonality) constant during the whole simulation.

Variable	**Values**
Cell motion strategy	Cell stretching, layer A/R
Time scheme	Euler, SOBE, CN
N_{cells}	10000
Δt (μs)	0.125, 0.25, 0.5, 1, 2

Table 5.1: Summary of the setup parameters for the numerical experiments.

Code Verification

If the piston moves with positive acceleration (boundary moving inwards, Fig. 5.4-bottom), domain cells undergo a compression; this compression is distributed throughout the mesh if cell stretching is applied (deformed by a rate that is inversely proportional to their distance from the moving end), or it is concentrated in the volume between the translating and the fixed regions, where a removal of cells is eventually triggered, in case of mesh motion based on topology changes. With negative acceleration (boundary moving outwards, Fig. 5.4-top), cells expand throughout the mesh (cell stretching) or a refinement is applied to the few cells that are deforming. If cell layers are added or removed, the cell count globally changes.

In Figs. 5.5 and 5.6, the velocity, pressure and temperature profiles along the domain, together with their relative error, are plotted for the compressing and expanding piston, respectively. In the graphs, the top row includes the results obtained by cell stretching, while results obtained with layer addition or removal (A/R) are reported on the bottom. Curves for different values of the time-step integration are compared against the analytical solution [35]. The vertical dotted line represents the x-coordinate where layer A/R is triggered at the time data are sampled. Results from the simulations are in good agreement with the analytical solution. Small discrepancies are observed at the wave front ($x/(c_0 t) = 1$). Being this effect present either

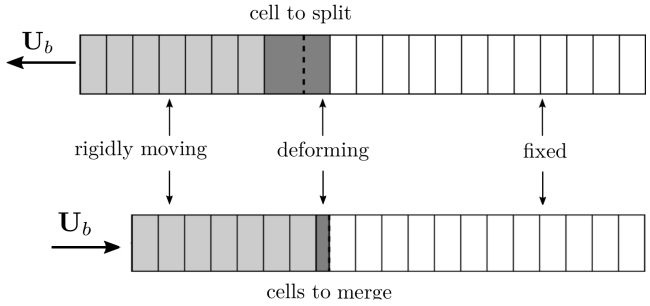


Figure 5.4: Basic principles of the dynamic addition/removal of cell layers.

with cell stretching or with cell layering, it is clearly not due to the mesh handling strategy used; it is rather a consequence of the non-conservative form of the governing equations, that are solved in a segregated fashion for velocity $\boldsymbol{U}$ and h instead of $(\rho\boldsymbol{U})$ and (ρh). A table summarizing the maximum value of error introduced by the topology change in all the schemes for each time step can be found in Tab. 5.2, showing how the absolute value of this error is very small in all the cases. Finally, the formulations of the proposed temporal differencing schemes provide an error with cell layering which is comparable to the error produced by the cell stretching strategy; in fact, the relative error is lower than 10^{-6} % for all the tested cases. This confirms once more the proper operation of the methodology proposed in this work.

In the case of the Crank Nicolson (CN) scheme, the results for the velocity, the pressure and the velocity fields for dynamic layering and cell stretching are shown in Figs. 5.5-c and 5.6-c. In all the cases, the main error is again located in the front wave and no error created by the topology change can be seen. The error with respect to the cell stretching strategy is around 0.01% for the velocity field in the compressing piston, which is two orders of magnitude bigger than for the SOBE scheme. This is expected as some simplifications have been performed to create the equivalent state. However, this value is still very small and negligible with respect to other sources of error. In addition, it can be seen that the error for the compressing piston is bigger than for the expanding one. This may seem counter intuitive because in layer addition, an approximation has been performed in 2 layers of cells. However, the volume these cells occupy is smaller than the volume occupied in the equivalent state when performing layer removal, thus reducing the total error of the solution.

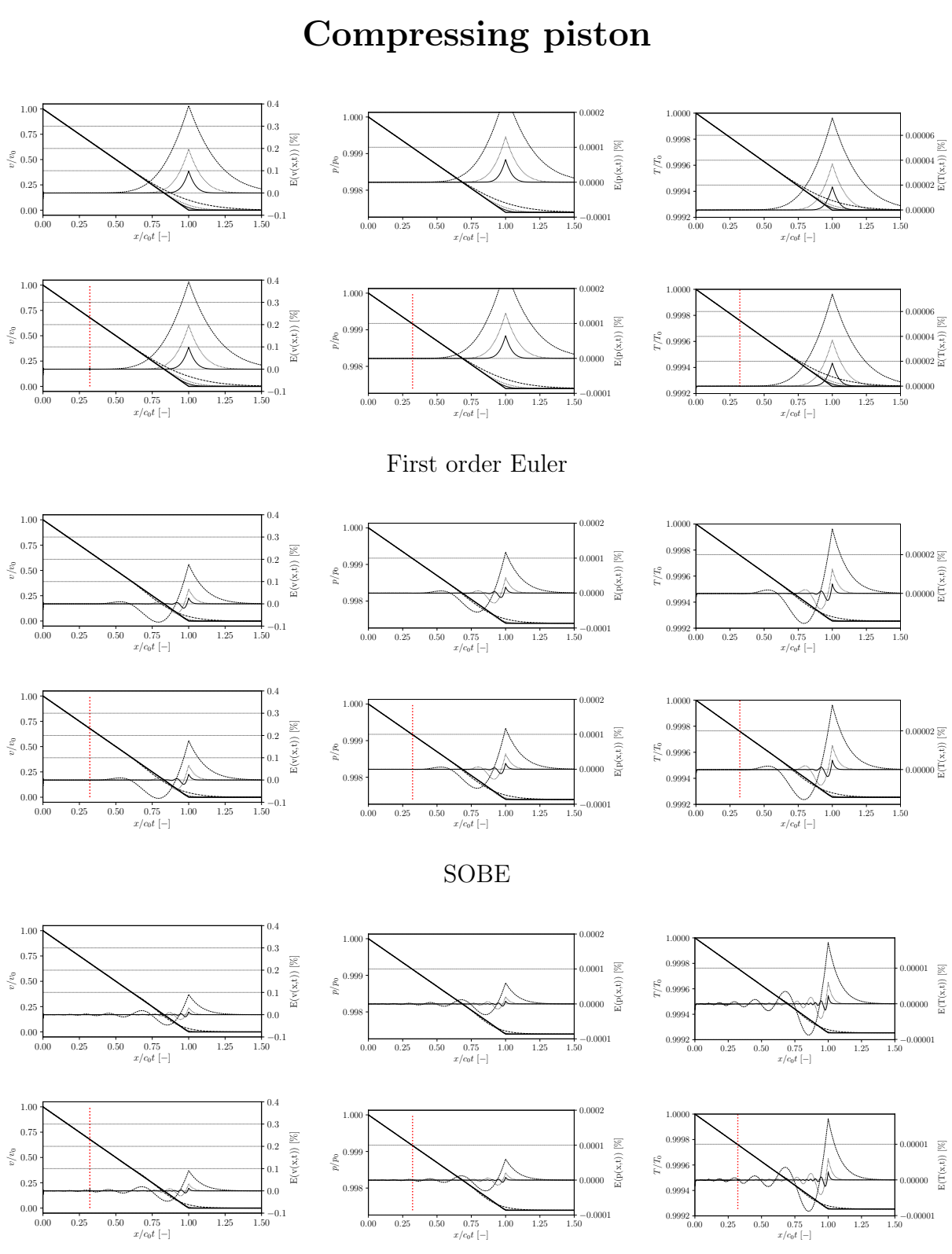


Figure 5.5: Relative error [%] with cell stretching (upper row) and dynamic layering (bottom row) with different time steps: --- $2\mu s$, ····· $0.5\mu s$ and —— $0.125\mu s$; for the Euler, SOBE and CN time schemes. ····· represents the position of the dynamic layer. For each subfigure, from left to right, errors on the velocity, pressure and temperature fields, respectively.

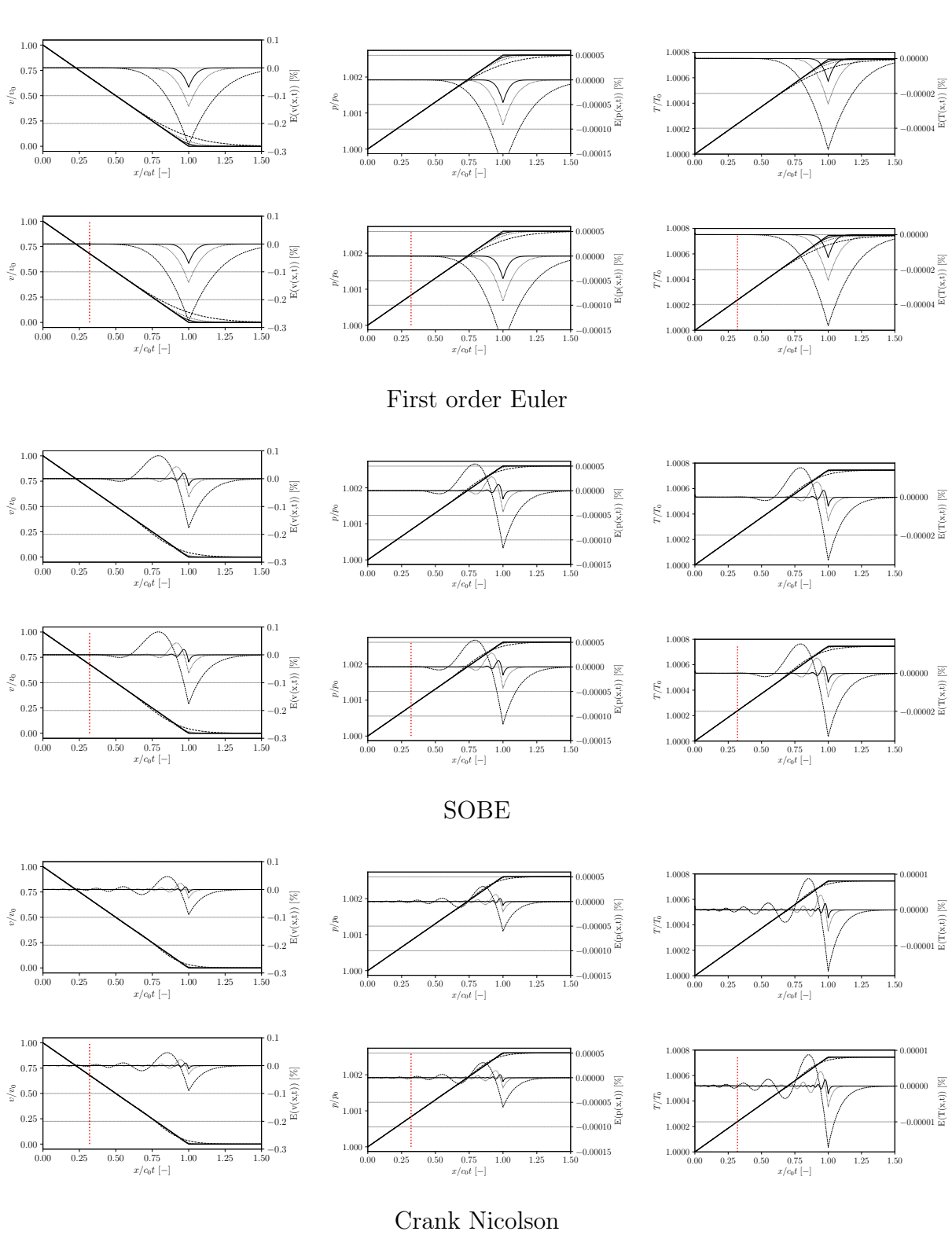


Figure 5.6: Relative error [%] with cell stretching (upper row) and dynamic layering (bottom row) with different time steps: --- $2\mu s$, ····· $0.5\mu s$ and —— $0.125\mu s$; for the Euler, SOBE and CN time schemes. ····· represents the position of the dynamic layer. For each subfigure, from left to right, errors on the velocity, pressure and temperature fields, respectively.

	Layer removal			Layer addition		
Δt	Euler	SOBE	CN	Euler	SOBE	CN
1.25e-7	3.19e-4	3.75e-4	2.10e-4	2.27e-4	3.76e-4	4.49e-4
2.50e-7	5.07e-4	7.37e-5	3.12e-4	8.81e-5	7.70e-5	3.18e-4
5.00e-7	8.16e-4	3.50e-5	3.24e-3	2.24e-4	4.26e-5	6.03e-4
1.00e-06	2.35e-3	6.55e-5	8.85e-4	8.88e-4	6.85e-5	8.21e-4
2.00e-06	1.15e-3	6.22e-4	3.11e-3	2.34e-3	7.56e-4	1.60e-3

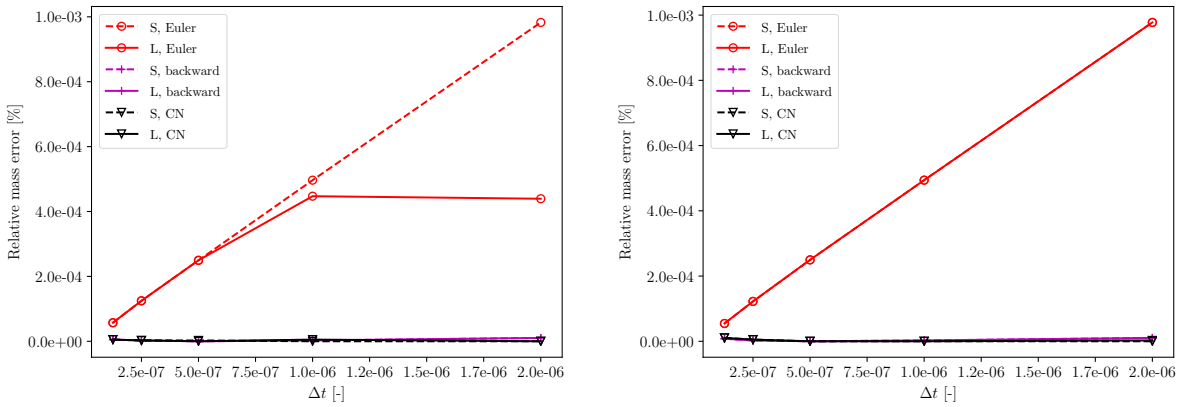
Table 5.2: Maximum relative error introduced in the velocity field by the topology change during cell removal (left) and cell addition (right)

Mass Conservation

The global mass conservation error can be estimated by computing the mass imbalance between start and end of the simulation:

$$E_{\text{mass}} = \left| \frac{\int_\Omega \rho \, \mathrm{d}V \big|_{t=t_N} - \int_\Omega \rho \, \mathrm{d}V \big|_{t=t_0}}{\int_\Omega \rho \, \mathrm{d}V \big|_{t=t_0}} \right| \tag{5.8}$$

and it is reported in figures 5.7a and 5.7b for the compressing and expanding piston, respectively. To compute this value, longer simulations have been performed where the number of cells in the mesh changed in at least 100, both in compression and expansion. In all cases the mass conservation error is very low, always with a relative error below 0.001%. For the Euler scheme, the error decreases linearly whereas it is always very small for the SOBE and CN schemes, both with layer A/R and with cell stretching (in the order of 1×10^{-9}). These results confirm that the proposed methodology does not create any spurious mass sources. This result was expected for the Euler and SOBE due to the fact that the DGCL has been satisfied explicitly, ensuring that the geometry is always conserved. However, the simplifications made with the CN scheme are very small and do not introduce any spurious mass source in the solution. In addition, these results indicate that the proposed methodology for conservative mapping is able to ensure mass conservation across topology changes both with first and second order temporal schemes.



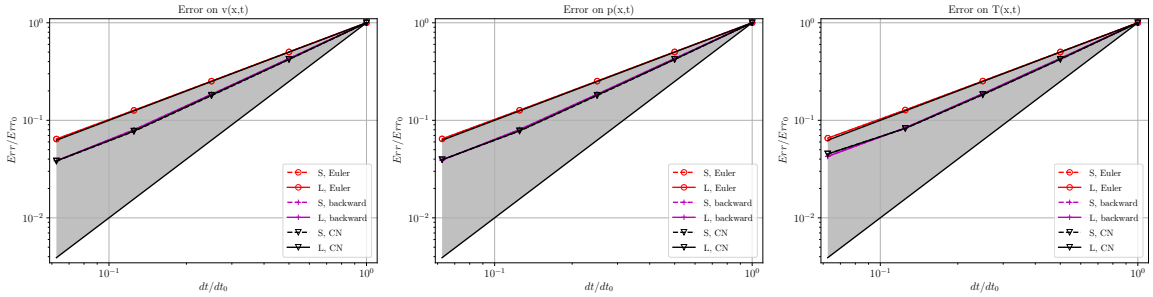
(a) Compressing piston

(b) Expanding piston

Figure 5.7: Relative mass error

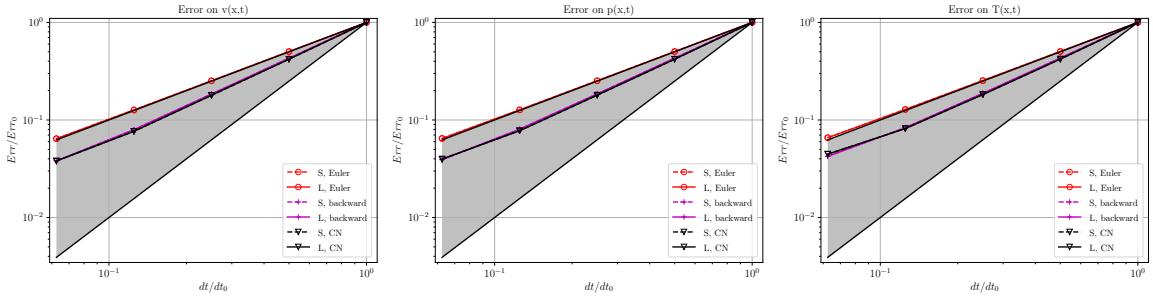
Temporal Order of Accuracy

So far, it has been shown that the error introduced in the solution when performing topological changes is very small. In addition, the effect of the methodology in the temporal order of accuracy also has to be checked. In Fig. 5.8 the order of accuracy of the three numerical schemes with dynamic layering and cell stretching strategies is shown for the velocity, pressure and temperature fields, with the piston both compressing and expanding. The first that has to be noticed is that, in all the studied cases, both strategies of mesh motion (dynamic cell layering and mesh stretching) achieved the same order of accuracy. This probes that the error introduced by the topological change does not modify the convergence rate of the solver. For the Euler scheme, an order of 0.99 is recovered for the velocity, pressure and temperature fields, while an order of 1.28 is achieved for both of the second order schemes. The observed convergence is determined by the relative importance of three error sources: the intrinsic truncation error in the time derivatives $\frac{\partial}{\partial t}$, the error introduced if the DGCL is unfulfilled and the error due to a lack of momentum conservation. The latter, like in the previous case, is not reduced by a smaller Δt, thus impairing the global accuracy order that is well below the value of 2 on all test cases.



(a) Velocity (b) Pressure (c) Temperature

Compressing piston



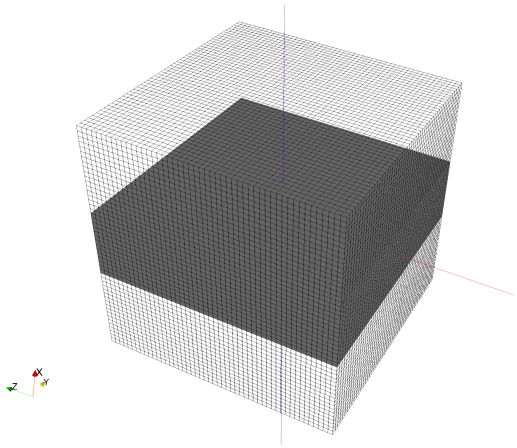
(d) Velocity (e) Pressure (f) Temperature

Expanding piston

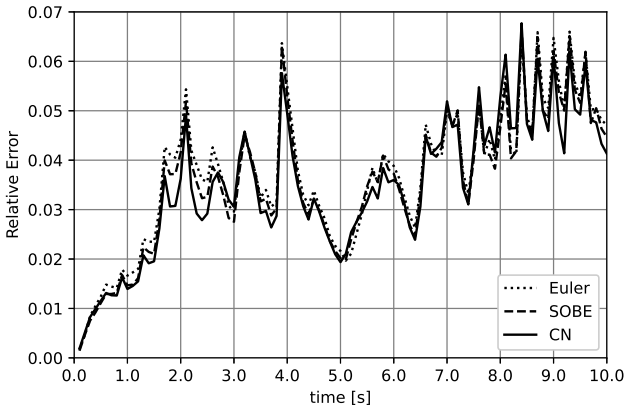
Figure 5.8: Temporal order of accuracy for Euler, SOBE and CN schemes

5.2. Lid-Driven Cavity Test Case

A three-dimensional lid driven cavity case at Re = 5000 and an orthogonal uniform mesh serves as a test to check the proposed method. No turbulence is present in the case tested and the 200x200x200 mesh is uniform and perfectly orthogonal, so no other effect could have an influence on the final solution. At $t = 10$ s, the stationary state is fully reached. A region of the grid made of six horizontal layers of cells located next to the upper part of the domain (Fig. 5.9a) oscillates periodically with a sinusoidal displacement of amplitude 0.045 m and frequency 0.05 Hz, spanning the all domain. The neighboring layer of cells, both above and below this region, will deform to accommodate the movement of the cells (Fig. 5.9a). When a threshold value of the cell volume has reached, all dynamic cell layering is applied. For the mesh flux, a first order upwind scheme has been chosen as it provides the exact solution. The case has been chosen because its solution tends to steady state. If the internal grid moves after the steady state is reached, any "unsteady effect" eventually observed is due to the numerics



(a) Test geometry



(b) Relative error

Figure 5.9: Three-dimensional lid-driven cavity: simulation setup. The dark region of cells oscillates periodically with a sinusoidal displacement of amplitude 0.045 m and frequency 0.05 Hz over the z-axis of the global reference frame. Normalized velocity profiles are computed over two center-lines over the x (U_x, blue) and the z direction (U_z, red). a) representative computational grid: the number of cells has been coarsened to improve clarity in the visualization; b) percent error computed by Eq. 5.9.

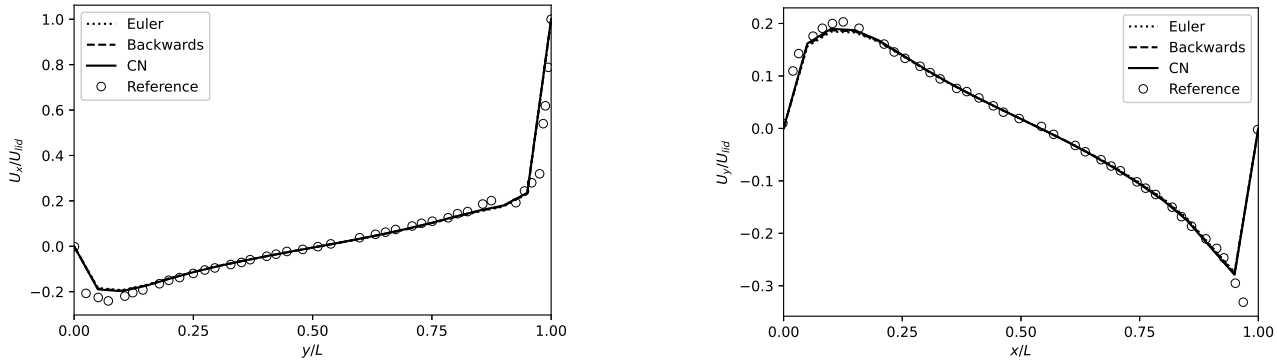
related to the grid motion (e.g. GCL). For this reason, this is a perfect case for validation.

Normalized velocity profiles have been computed over two center-lines along the x (U_x, blue) and the z direction (U_z, red). Comparisons between reference solutions on a static grid using a second-order implicit backward differencing scheme have been compared against dynamic simulations (Fig. 5.10a and 5.10b). As apparent, the error introduced by the mesh motion in the flow field is negligible.

To quantify the error introduced by the topological changes, the following (percent) relative error has been computed:

$$E_x(t) = \frac{\overline{|U_x^{dynamic}(\hat{z},t) - U_x^{static}(z,t)|}}{U_x^{static}(z,t)} \cdot 100 \tag{5.9}$$

In Eq. 5.9, the superscript dynamic shows that quantities are computed accounting the oscillating region (Fig. 5.9a). The percent relative error is shown in Fig. 5.9b. It has to be noted that the results in the dynamic grid have been evaluated at $\hat{z}(t)$, whose distribution changes with time following the mesh motion. In order to compare the results, the fields have been interpolated over the original positions of the grid, where the static solution is computed. The error of the different temporal schemes is reported in Fig. 5.9b. The evolution of the velocity magnitude field at times $1s$, $2s$, $3s$, $4s$ and $5s$ is shown in Figs. 5.11 for the temporal schemes



(a) $U_x(y)/U_{lid}$ in vertical line

(b) $U_y(x)/U_{lid}$ in horizontal line

Figure 5.10: Lid-driven cavity test case. Normalized velocity profiles computed over two centerlines over the x and the z direction (see Fig. 5.9a).

tested (Euler, SOBE and CN). In the figures, graphs on the upper rows refers to simulations on a static grid, while calculations on the dynamic grid are reported in the bottom row. The grid points representing the cell centers are superimposed to the velocity flow field and differ in number and positions between the static and the dynamic grid. For all three schemes tested, results on the static and the dynamic grid are in very good agreement, proving that the proposed methodology does not introduce any error in the solution. The generation and evolution of the vortex is correctly captured in the whole domain and, once it is formed, the mesh motion does not distort it. Both the location and intensity of the vortex are also maintained across the topology changes. Finally, mass conservation has also been checked for this test case. The relative mass error has been computed using Eq. 5.8 for the three temporal schemes. The results show that the SOBE and CN schemes introduce more error in the solution than the Euler scheme. However, the value of this error is below 2.5×10^{-5} for all the schemes tested.

5.3. Industrial Case: Internal Combustion Engine

Once the methodology has been validated in academic problems, the next step is to test the algorithmic developments to improve temporal accuracy presented in Sec. 4.4 in an industrial configuration. Among all the possible cases involving large deformations of the computational domain, an Internal Combustion Engine (ICE) has been selected as it presents several independent moving parts that undergo large displacements, namely the valves and the piston. In addition, the usage of second order temporal schemes for this problem, where the flow transport and mixing play an important role, provides a remarkable improvement in the accuracy of the solution.

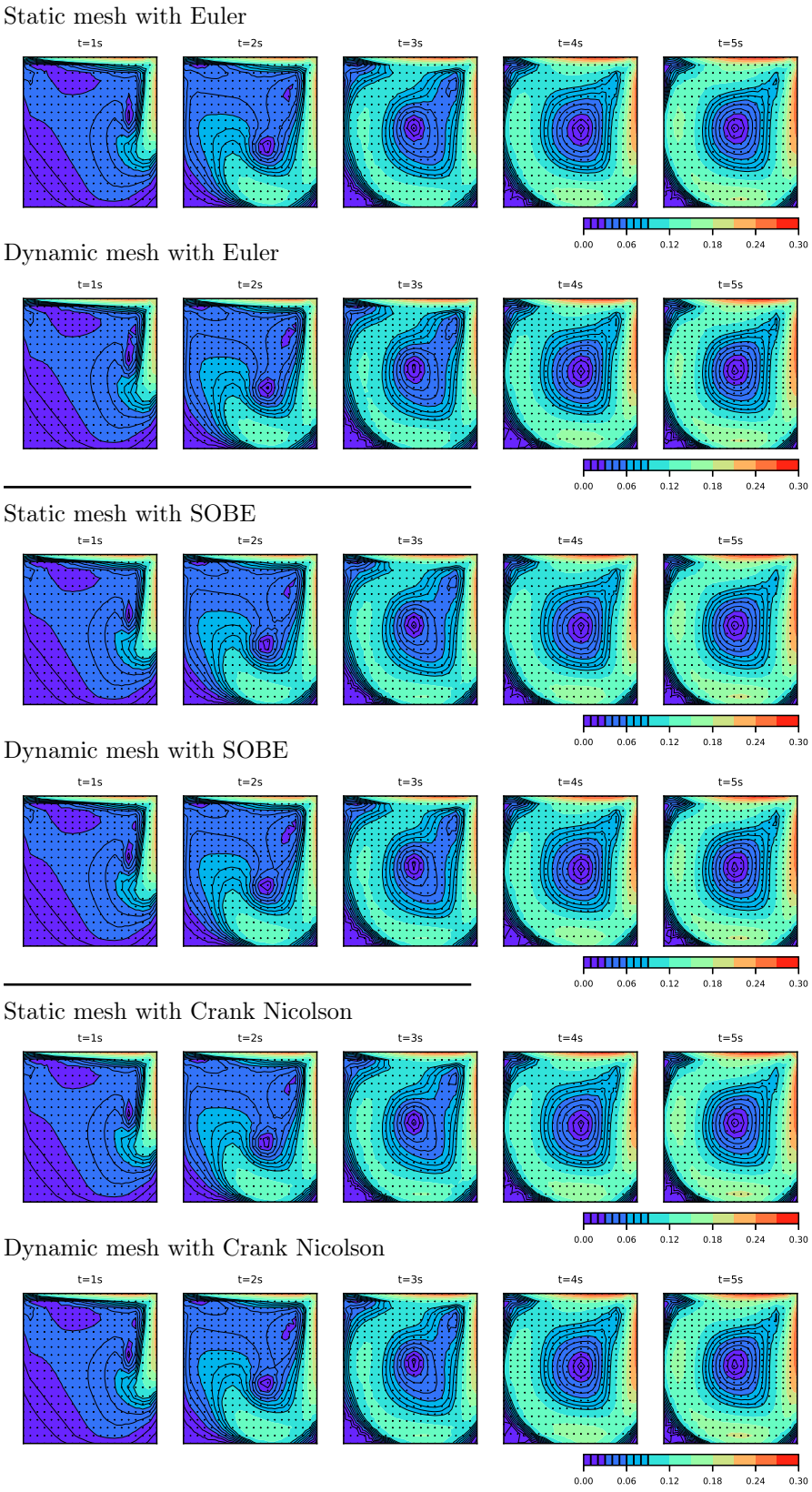


Figure 5.11: Lid-driven cavity test case. Comparison between the evolution of the velocity flow field in time over a central cutting-plane at t=1,2,3,4,5 s. Points representing the cell centers are superimposed to the velocity flow field and differ in number and positions between the static and the dynamic grid.

Due to the variation in the geometric volume of the flow domain, as well as the computational grid, dynamic layering techniques become necessary [45]. For instance, the volume of the combustion chamber of a typical commercial IC engine changes approximately in a 10 : 1 ratio. This compression ratio is achieved through the motion of the piston in only one direction between the top dead center (TDC) and bottom dead center (BDC). As most of the clearance volume is provided by the pent roof (or the bowl of the piston in diesel engines), the ratio of the grid step sizes Δ_{mesh} at different engine times could vary up to 40 times, depending on the specific engine design, if the mesh's topology remains unchanged during the entire engine cycle. However, this is not ideal for any scale resolving simulation and specially LES, as the lower limit of the filter size explicitly depends on Δ_{mesh}.

To balance the resolution, one solution is to prepare multiple predefined grids for different crank angles (CAs) and remap between them runtime during simulation. However, this requires a significant amount of meshing effort. An alternative solution is the automatic cell layering, which avoids re-meshing effort while maintaining a user-defined grid resolution. In this method, Δ_{mesh} of the entire mesh domain remains unchanged during the mesh motion, except for one layer of the mesh where the points are stretched according to the prescribed piston motion law. The advantages of this dynamic mesh strategy are that grid resolution is practically decoupled from the piston motion, location of the layer can be specified by the user, and much less meshing effort is required compared to mesh remapping.

In addition, using cell layering techniques involves that only a part of the mesh will move, while the most part will remain stationary. This is specially desirable for the Lagrangian spray modeling because the results are very dependent on the underlying grid, as it will be shown in Sec. 6. Using second order temporal schemes with dynamic cell layering techniques represents therefore an important improvement to the simulation of ICEs, taking the advantages of a mesh handling based on topology changes and the improved accuracy provided by high-order numerical schemes.

The selected geometry to illustrate the methodology is known as the TCC-III engine [58] and constitutes a benchmark case for numerical validation. It is a four stroke, spark ignited engine with two vertical valves and a flat piston head. All the details regarding the geometry and experimental data sets can be found in [60] and a summary is provided in Tab. 5.3.

Bore $[cm]$	9.20	Conrod lenght $[cm]$	23.10
Stroke $[cm]$	8.60	Geometric CR	10.00
Clearance at TDC	0.95	Effective (IVC) CR	8.00
Comb chamber $[cm^3]$	63.15	Steady-flow swirl ratio	0.4
Top land crevice $[cm^3]$	0.37	TDC volume $[cm^3]$	63.54
Sparkplug crevice $[cm^3]$	0.02	Swept volume $[cm^3]$	571.7

Table 5.3: TCC-III engine geometry

ICECube: Automatic Mesh Generator

The first step in the simulation of an ICE is the generation of the initial mesh. It needs to have the adequate resolution to capture all the appearing phenomena plus have a number of cells that is affordable for the available computational resources. In addition, it must have as high quality as possible and with special resolution in regions close to the valves where the velocity gradients are more important. Usually, this is a process that takes a person a large amount of time to generate, as there are a lot of constrains that must be satisfied, being often a bottleneck in the full simulation procedure. Therefore, generating a good quality mesh automatically represents an important salvage of engineering time, specially for industrial geometries.

In this work, the ICECube software has been used to generate the initial mesh. ICECube is an in-house multi-block hexahedral automatic mesh generator developed at the Department of Aerospace Science and Technology (DAER) at the Politecnico di Milano. This innovative tool is designed specifically for internal combustion engines and piston machines. Starting from a surface geometry provided in the stereolithography format (STL), the algorithm can accurately identify various engine components such as valves, pistons, cylinder heads, and cylinders. It then generates multiple blocks, automatically morphing their surfaces to match the original CAD geometry (each different color in Fig. 5.12). Subsequently, the hexahedral mesh is generated individually for each region, providing an accurate interface between different blocks. In just a matter of seconds, ICECube produces a high-quality hexa-block grid of the IC Engine, even including intricate features like head ducts.

Mesh Motion With Topology Changes

Once the initial mesh has been created with ICECube, mesh motion is achieved by using three different types of topology changes, namely dynamic cell layering, Arbitrary Mesh Interface (AMI) and valve Attach/Detach. The main goal of the strategy is to keep the high initial mesh quality during the whole engine cycle. To do so, the idea is to move or deform the minimum

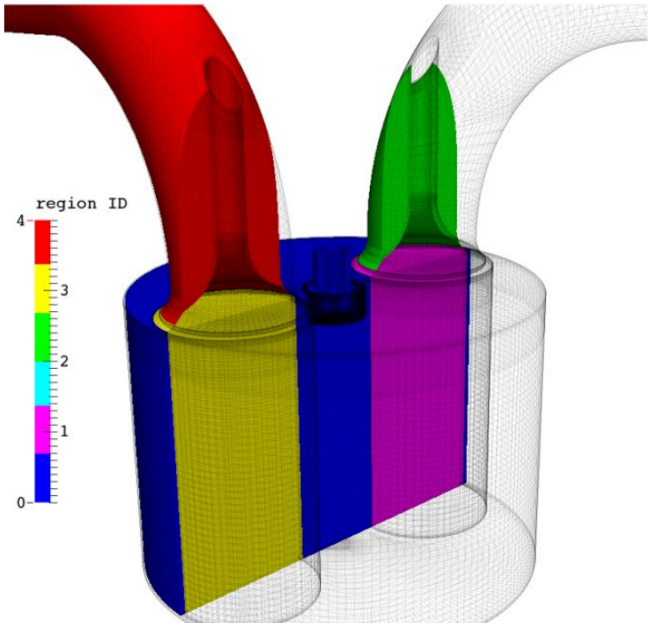
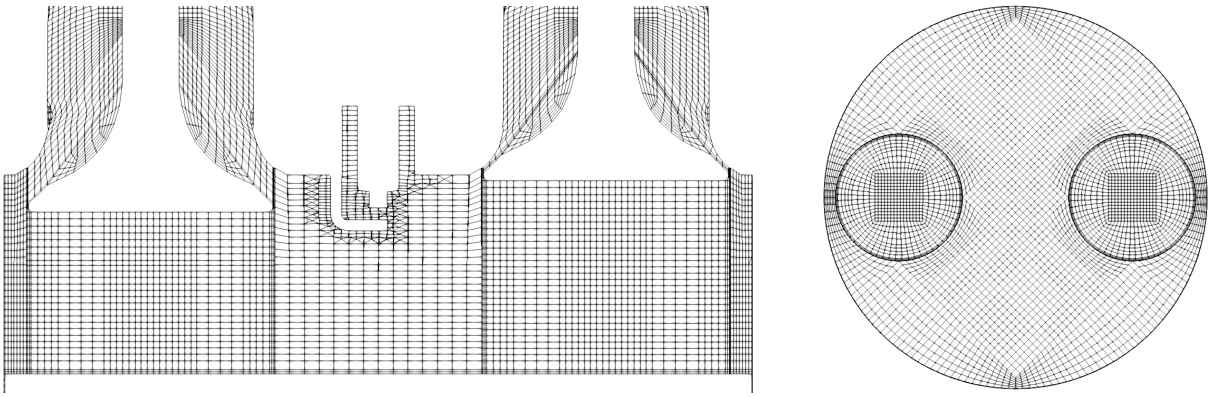


Figure 5.12: Diagram showing the different morphed hexa-block regions of the TCC-III engine

number of cells possible. In the case of the TCC-III engine, there are three moving parts: the two valves and the piston. In addition, the valves can be separated into its upper and lower surfaces, which would make a total of 5 moving surfaces, that correspond to the 5 blocks of Fig. 5.12. The blocks below the valves (yellow and pink regions in Fig. 5.12) have been created following the direction of its motion, so that the structured mesh below them is aligned with the movement and dynamic cell layering can be easily applied. For these regions, both the upper and lower extremes of the block will suffer the dynamic cell layering to account for the movement of the valves and piston, respectively.

The regions in the upper part of the valves (red and green regions in Fig. 5.12) must be handled more carefully. One logical axis of the structured grid must be aligned with the valve stem. Then, a truncated cone surface is created from the valve outer ring to the stem, like in Fig. 5.13a. The cells located between this surface and the surface of the valve, will move rigidly with it and will nor deform. Another truncated cone surface, parallel to the previous one, will also be created, but this time from the valve seat, located in the cylinder head. When the valve closes, these two surfaces will coincide and will be converted into boundary cells to separate the ducts from the cylinder. The cells located above this surface (in the ducts) will not move nor deform. Finally, dynamic cell layering will be applied to the region between these two surfaces, creating or removing cells parallel to them. Finally, the rest of the cylinder (blue region in Fig. 5.12) also has an structured mesh parallel to the cylinder head and will suffer dynamic cell layering due to the piston motion.



(a) Central cut (b) Bottom view

Figure 5.13: Initial mesh generated automatically with ICECube

All the cylindrical regions have been meshed with an internal squared zone to avoid small cells in the axis, surrounded by axisymmetric cells in a structured manner (see Fig. 5.13b). The connection between the regions below the valves and the region covering the rest of the cylinder has been performed with Arbitrary Mesh Interfaces (AMIs), which are boundary conditions that allow communication between two different mesh blocks. They permit different meshes to overlap, providing a seamless connection between them, and are useful when simulating rotating geometries or objects with irregular shapes that require different levels of mesh refinement. The AMI works by defining an interface region where the two meshes touch, and then interpolating the flow information between the meshes in this region. The interpolation is performed using a weight-based approach, where the weights are calculated based on the distances between each point in the interface region and the surrounding grid points. In addition, as the number of cells in the mesh changes, and so does the topology of the grid, the connectivity along the AMI has to be recomputed every time step, introducing some additional computational cost in the calculation. However, as the overlapping regions are very large, this method is preferred over others like sliding interfaces, that works best in smaller overlapping areas.

To simulate the full engine cycle, it is necessary to model the opening and closure of valves, which can divide the computational domain into up to three separate regions. The *attachDetach* functionality in OpenFOAM® is used to handle this, which converts internal faces into boundaries and alters the mesh topology of affected cells. OpenFOAM® also allows for simultaneous solving of disconnected domains, increasing computational efficiency and enabling solutions for closed valve ducts. These topology changes result in an automatic and efficient mesh handling process that maintains high-quality grids throughout the whole engine cycle. The computational overhead associated with these changes depends on the implementation and geometry, but the simulation preparation time is minimal, as only a few input parame-

ters are required for both the initial mesh generation and dynamic mesh handling, which are executed automatically.

Discretization

The objective of this test case is to validate the algorithmic developments introduced for second order accurate temporal schemes in an industrial configuration. The mesh motion technique based on topology changes described in the previous sections, has already been validated by *F. Piscaglia et al.* [45] and is not considered in this work. However, only the Euler scheme (first-order accurate) was considered for the temporal derivative. To validate the proposed methodology with second order temporal schemes, the results will be compared with the ones obtained with first order schemes in [45]. The goal is to verify that the solutions are similar (but not equal as different schemes are used), and that no numerical distortion is introduced in the regions where the topology changes take place, namely around the valves and in the piston. Special attention is put in the upper part of the valves because the grid lines are not perpendicular to each other.

In this work, the Top Dead Center (TDC) is considered to take place at 360CA, where intake stroke starts. To save computational time, the simulation has been computed only from 320CA to 600CA as the rest of the cycle does not add any particular challenge to the methodology. This period starts with the exhaust valve open and the piston moving up, pushing some air out of the engine. This valve will move up until eventually be closed at 392CA after TDC. The intake valve will open at 337CA and will descend following the movement provided by experimental measurements. The Bottom Dead Center (BDC) will occur at 540CA, where the piston will change its direction and start moving upwards again. Like this, only the simulation of 280CAs suffices to check all the topology changes: valve opening and closure, piston compressing and expanding, and valve moving up and down. In addition, as cold flow simulations (without combustion) are being computed, the biggest gradients will also occur during this period of time, with the air entering the engine and mixing with the existing air.

The implementation of time differencing schemes suitable for dynamic meshes with topology changes clearly works with any turbulence model; in the following, RANS simulations by the $k-\omega\ SST$ turbulence model have been employed to limit the computational time of the simulations. The only fluid present is air and has been considered as a perfect gas. Second order central differences have been used to discretize spatially the equations, including the mesh fluxes. A PIMPLE algorithm with 2 outer correctors, 3 inner correctors and 1 non-orthogonal corrector has been used to solved the algebraic equations. An adjustable time step has been used, allowing a maximum Courant number of 5 during the simulation. Due to stability issues,

the maximum Courant number is reduced to 1 when the opening and closure of the valves occurs, returning to 5 after a few time steps have been completed. The mesh has around 300000 cells in the TDC, including both ducts and the plenums, with an average mesh size of $1mm$, approximately.

Results

To validate the methodology, the same test case has been run with three different temporal schemes: first order Euler, SOBE and Crank-Nicolson. From Figure 5.14 to 5.20 the results of the velocity field are shown at different relevant times in the central plane of the engine. The image in the left corresponds to the Euler scheme, the central one to the SOBE and the one in the right to the Crank Nicolson. The color map in all the figures goes from 0 to $50m/s$.

The first conclusion one can make by looking at the results is that no discontinuity appears in the solution when using second-order temporal schemes. In the cylinder, the vertexes are smoothly convected both close to the piston and under the valves. In addition, in the valve ducts, where dynamic cell layering is applied to non-orthogonal cells, no discontinuity is present, even in the presence of very large gradients. The discontinuity appearing in the region above the valve external ring is due to the two-dimensional visualization of three-dimensional non-conformal interfaces connecting the different blocks of the mesh (see Fig. 5.13a).

In general, results with second order schemes provide sharper edges of the vortexes, capturing better the regions with zero velocity. This is also in agreement with experiments reported in the existing litterature [1].

Figure 5.14: Results of the TCC-III engine at 320CA

(a) Euler (b) SOBE (c) CN

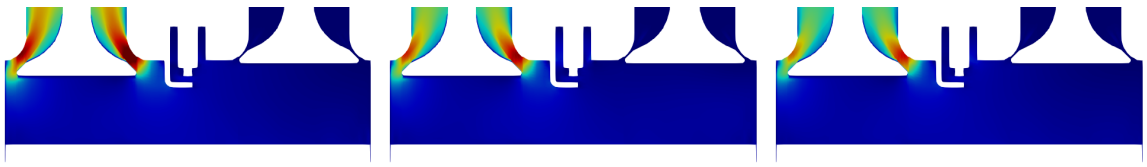


Figure 5.15: Results of the TCC-III engine at 400CA

(a) Euler (b) SOBE (c) CN

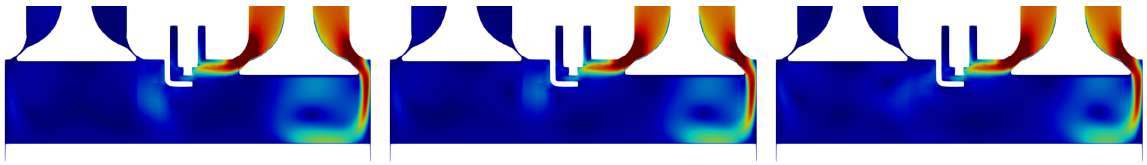


Figure 5.16: Results of the TCC-III engine at 440CA

(a) Euler (b) SOBE (c) CN

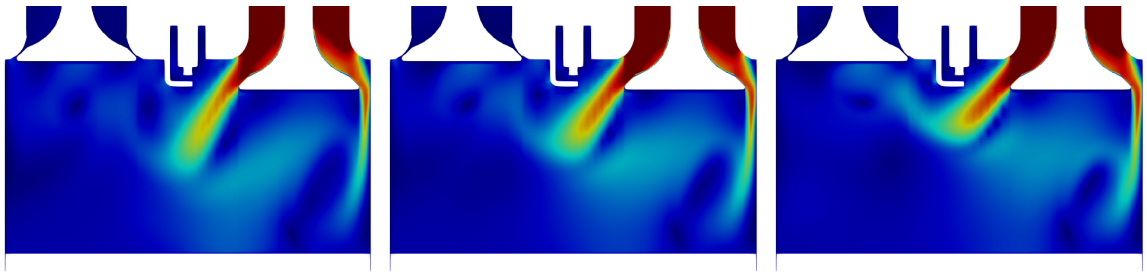


Figure 5.17: Results of the TCC-III engine at 460CA

(a) Euler (b) SOBE (c) CN

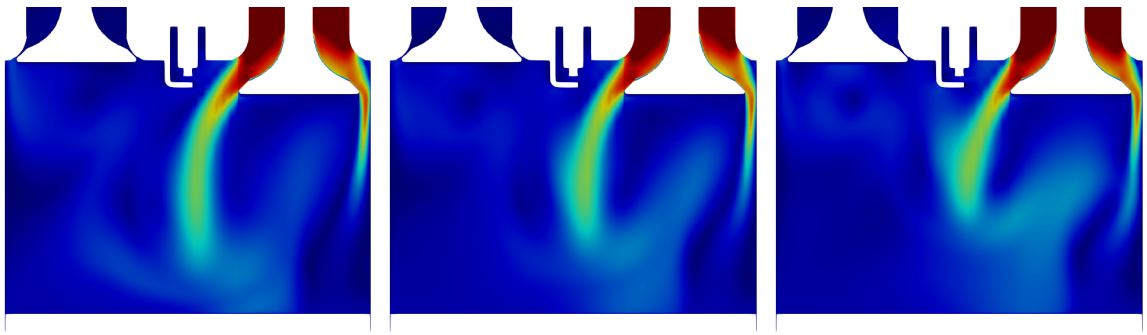


Figure 5.18: Results of the TCC-III engine at 480CA

(a) Euler (b) SOBE (c) CN

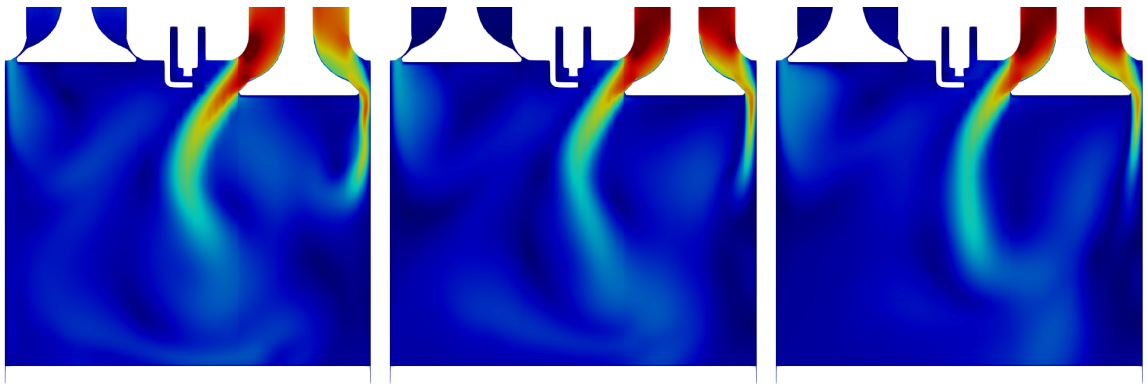


Figure 5.19: Results of the TCC-III engine at 500CA

(a) Euler (b) SOBE (c) CN

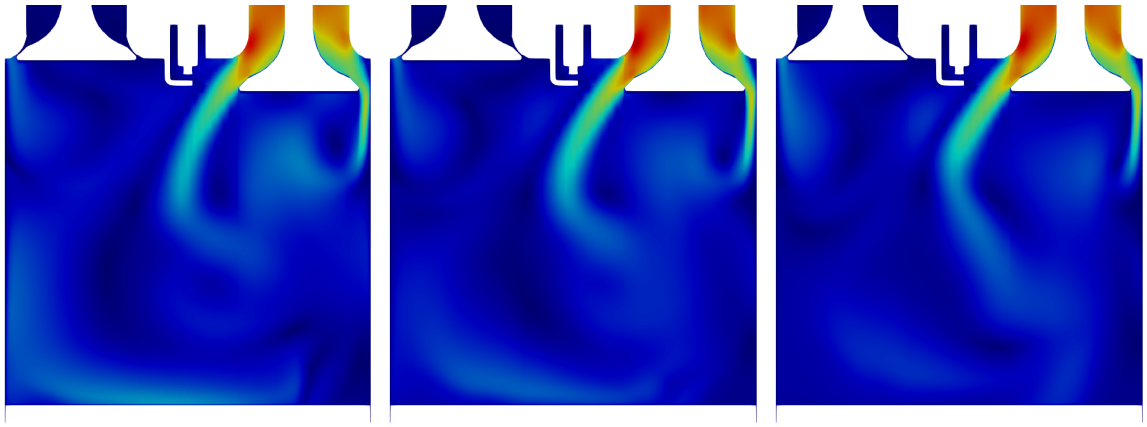
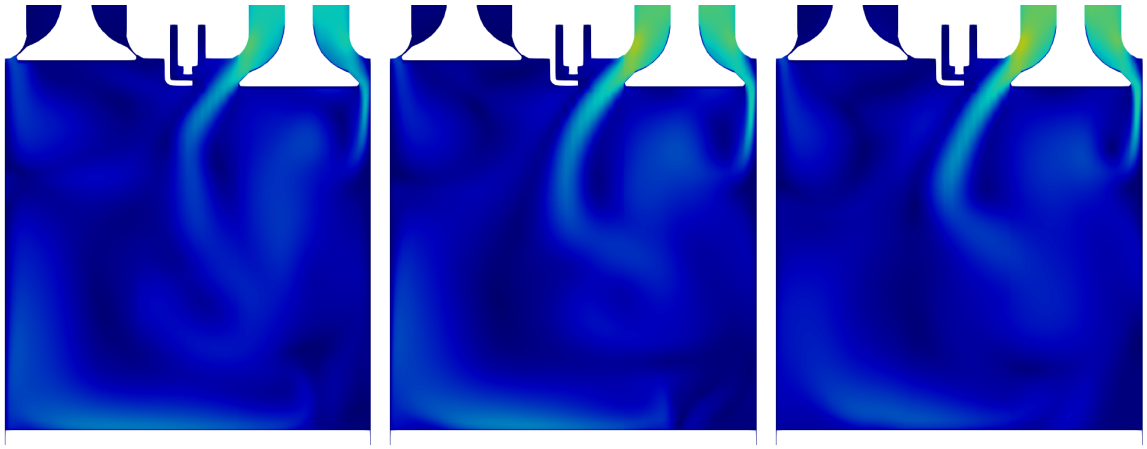


Figure 5.20: Results of the TCC-III engine at 520CA

(a) Euler (b) SOBE (c) CN



6 | Lagrangian Spray Modeling

The aim of this chapter is to couple highly-accurate mesh motion techniques with spray modeling. Aiming at improving the simulation of dual-fuel engines, a Lagrangian hybrid spray model [36] has been implemented and validated in OpenFOAM® and applied to two benchmark cases (Spray A and G), for which experimental measurements are available [1]. The spray model introduces the Huh-Gosman breakup model to improve the accuracy in the liquid core of the spray. Then, this model has been used in a simple bomb case with a moving wall, representing a similar situation to what would happen in a real IC Engine. The idea is to investigate its behavior when coupled with overset grids methodologies: the goal is to use an overlapping static mesh, where the lagrangian spray is solved, on the top of a background dynamic grid of the engine. Results with overset grids are compared with the ones obtained with different mesh motion techniques to illustrate its effectiveness.

6.1. Introduction

There is an extensive literature about atomization and sprays of liquid jets in combustion chambers [9, 41, 42, 48, 19, 1]. The most used approach to simulate disperse spray is the Discrete Droplet Method (DDM), where the liquid phase of the fuel is represented by solid parcels. The primary and secondary atomizations of the spray are represented by means of phenomenological models, using magnitudes like the characteristic turbulent length and time or aerodynamic instabilities. To predict correctly both atomization stages, usually hybrid models are employed, combining different models for each mechanism. One of the most popular hybrid models uses the Kelvin-Helmholtz instability to model how the droplets are formed in the liquid core (primary breakup) and the Rayleigh-Taylor to model how the droplets are broken into smaller ones as a consequence of the aerodynamic forces (secondary breakup).

Recently, novel dual-fuel engines, where two different fuels are burnt simultaneously, are being developed aiming at increasing the efficiency of the combustion. This technology is particularly suited for heavy-duty machinery and marine applications, where emissions are still high

compared to street cars. In this case, the mixing and combustion processes are even more complex because there are two fuels plus the air interacting with each other. Fuels are typically injected in liquid phase, getting atomized and vaporized during the interaction with the other fluids present in the engine. In addition, non conventional fuels like hydrogen, ammonia or bio-fuels are being investigated for this type of engine, as they may provide better results than traditional fuels like diesel or gasoline.

Even thought experimental results show very promising results with this technology, traditional computational models cannot reproduce accurately all the phenomena involved. In particular, spray modeling represents the biggest limitation as traditional models lack the sufficient accuracy. In dual-fuel engines, new advanced hybrid models must be utilized, considering different breakup mechanisms suited for each particular region of the spray. In this work, an advanced hybrid spray model has been implemented aiming at improving the accuracy of the spray simulation closer to the injector, where the spray is mainly in liquid form. In particular, the Huh-Gosman [30] model has been introduced as it contains information about the turbulence created inside the injector and that is present in the flow.

6.2. Hybrid Spray Model

A series of mechanisms have been proposed in the literature to simulate the breakup and atomization of a spray under the Lagrangian framework. The combined effect of cavitation, turbulence and aerodynamic forces is important for causing the primary breakup, while the secondary one is mainly caused by aerodynamic forces. Therefore, it seems natural to build separate models for each one of the breakup processes. The most used ones are explained in the following sections.

6.2.1. Huh-Gosman (HG) Model

In the Huh-Gosman model, the initial perturbation of the liquid jet surface is caused by the turbulence in the nozzle. The perturbation grows according to the Kelvin-Helmholtz instability until the primary breakup occurs, detaching some droplets. Coupling the nozzle conditions to the atomization process reduces the influence of the conditions of the jet at the exit of the nozzle, reducing the related uncertainties.

For the initial perturbation, the instantaneous length ($L_T(t)$) and time ($\tau_T(t)$) scales of the turbulence in the nozzle are calculated as:

$$L_T(t) = L_T \left(1 + C_{a1} \frac{t}{\tau_T}\right)^{C_{a2}} \tag{6.1}$$

$$\tau_T(t) = \tau_T \left(1 + C_{a1} \frac{t}{\tau_T}\right) \tag{6.2}$$

L_T and τ_T represent the initial turbulence length and time scales calculated from the turbulence energy and dissipation at the nozzle exit. The constants, tuned from experimental values, have values of: $C_{a1} = 0.92$ and $C_{a2} = 0.4565$. The initial perturbation will grow until eventually detach some droplets from the jet. The atomization length scale (L_A) and the the surface perturbation wavelength scale (L_W) are expressed as a linear combination of L_T as:

$$L_A = C_1 \cdot L_T \tag{6.3}$$

$$L_W = C_2 \cdot L_A \tag{6.4}$$

The wave growth time scale (τ_W) is computed like:

$$\tau_W = \frac{L_W}{w} \sqrt{\frac{\rho_l}{\rho_g}} \tag{6.5}$$

where ρ_l/ρ_g represents the ratio of the densities of the liquid and gas phases, and w represent the relative velocity between the liquid and the gas. The atomization time scale (τ_A) in this model is therefore computed as:

$$\tau_A = C_3 \cdot \tau_T + C_4 \cdot \tau_W \tag{6.6}$$

$C_1 = 0.1$, $C_2 = 2$, $C_3 = 1.2$ and $C_4 = 0.4$ are model constants. The variation of the mother droplet size is expressed as:

$$\frac{dr}{dt} = -\frac{L_A}{\tau_A} \tag{6.7}$$

A new parcel it created with this model once its mass exceeds 10% of the initial mass of the mother parcel. The size of the mother parcel decays following Eq. 6.7 while the size of the new parcel is set to L_A.

6.2.2. Kelvin-Helmholtz (KH) Model

This model assumes the breakup can occur if there is a large enough velocity gradient in the interface between the liquid and the gas. The wavelength (Λ_{KH}) and the frequency (Ω_{KH}) of

the wave in the surface of the jet can be calculated as:

$$\Lambda_{KH} = \frac{9.02 \cdot r_0 \left(\left(1 + 0.45\sqrt{Z} \right) (1 + 0.4 \cdot T)^{0.7} \right)}{\left(1 + 0.865 \cdot We_g^{1.67} \right)^{0.6}} \tag{6.8}$$

$$\Omega_{KH} = \frac{0.34 + 0.385 \cdot We^{1.5}}{(1 + Z)(1 + 1.4 \cdot T^{0.6})} \sqrt{\frac{\sigma}{\rho_l \cdot r_0^2}} \tag{6.9}$$

where σ is the surface tension coefficient, $We_g = \rho_g w^2 r_0/\sigma$ and $We_l = \rho_l w^2 r_0/\sigma$ are the Weber number of the gaseous and liquid phases respectively, $Z = \sqrt{We_l}/Re_l$ the Ohnesorge number of the liquid phase, Re_l the Reynolds number of the liquid phase and $T = Z\sqrt{We_g}$ is the Taylor number.

The new droplet size is computed like:

$$r_{KH} = B_0 \cdot \Lambda_{KH} \tag{6.10}$$

where $B_0 = 0.61$ is a model constant, and the breakup time scale (τ_{KH}) like:

$$\tau_{KH} = \frac{3.788 \cdot B_1 \cdot r_0}{\Omega_{KH} \cdot \Lambda_{KH}} \tag{6.11}$$

where $B_1 = 120$ is another model constant, both of which which can be used to calculate the size variation of the mother droplets like:

$$\frac{dr}{dt} = -\frac{r_0 - r_{KH}}{\tau_{KH}}, r_{KH} \leq r_0 \tag{6.12}$$

KH model can be divided in two schemes. When $r_{KH} \leq r_0$, the droplet breakup occurs according to Eq. 6.12 and a new droplet is created with a radius of r_{KH}, once its mass exceeds a given percentage of the initial parcel mass. This value is called B_2 and is another model constant that will have a value of 0.03. Otherwise, the droplets are enlarged only once.

6.2.3. Rayleigh-Taylor (RT) Model

The Rayleigh–Taylor is an instability of an interface between two fluids of different densities which occurs when the lighter fluid pushes the heavier one. In the spray atomization process, this is important in the breakup process of liquid droplets, as they decelerate rapidly as a results of the drag forces with the nozzle gas. In this model, the fastest-growing frequency Ω_{RT}, the

wave number K_{RT} and the wavelength Λ_{RT} can be calculated as:

$$\Omega_{RT} = \sqrt{\frac{2}{3\sqrt{3}\sigma} \cdot \frac{(a \cdot (\rho_l - \rho_g))^{3/2}}{\rho_l + \rho_g}} \tag{6.13}$$

$$K_{RT} = \sqrt{\frac{a\,(\rho_l - \rho_g)}{3\sigma}} \tag{6.14}$$

$$\Lambda_{RT} = C_{RT} 2\pi \sqrt{\frac{3\sigma}{a\,(\rho_l - \rho_g)}} \tag{6.15}$$

$$\tau_{RT} = \frac{C_T}{\Omega_{RT}} \tag{6.16}$$

with $a = 3C_D\rho_g w^2/(8\rho_l r)$ is the droplet acceleration in the direction of movement and C_D represents the drag coefficient. $C_T = 1$ and $C_{RT} = 0.1$ are the model constants responsible for the droplet size and breakup time. To decide if a droplet breaks up or not, the droplet diameter r_0 is compared with the fastest-growing wavelength. If $\Lambda_{RT} \leq 2r_0$, the breakup will take place and new product droplets will be generated with a size of $r_{RT} = \pi C_{RT}/K_{RT}$. Once the RT breakup takes place, no new particles are generated and only the number of droplets in the parcel and the droplet size are updated.

6.2.4. Hybrid Model

The most widely used model for the simulation of spray injection is the KH-RT model from Reitz et al. [56]. There, the primary breakup is modeled with the KH mechanism, while a competition between the KH and the RT mechanism is used for the secondary breakup. If high injection pressures are used, the KH mechanism is not capable of capturing the behavior of the liquid core close to the injector. In this regard, models like the HG [30] provide a higher accuracy of this region. The three breakup models have been combined following [3], according to the scheme from Fig. 6.1.

For the primary breakup, the HG and the KH mechanism compete to decide which one is dominant for every parcel. To do so, the characteristic breakup times are compared, selecting the smaller one as the breakup time scale of the parcel τ_{bu}:

$$\tau_{bu} = min\,(\tau_A, \tau_{KH}) \tag{6.17}$$

Once the dominating mechanism is chosen, the size of the parcels will evolve according to Eq. 6.7 or Eq. 6.12, respectively. When the parcels are located further from the injector than

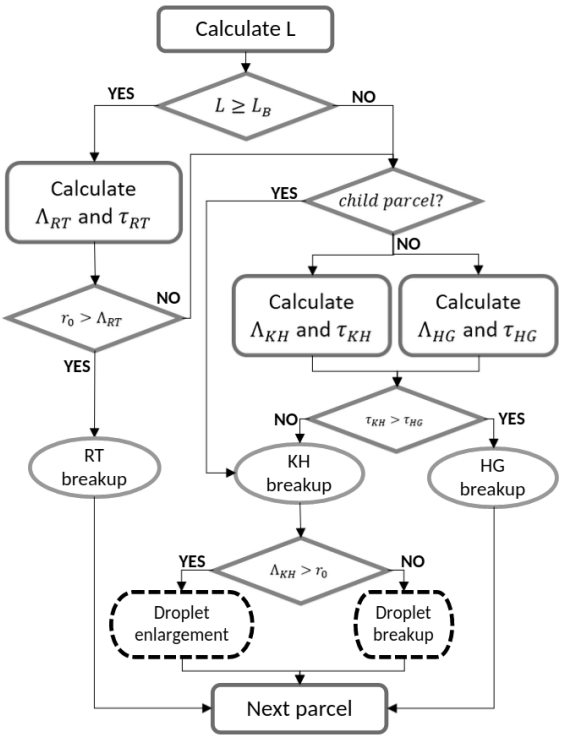


Figure 6.1: Schematic of the hybrid spray model [36].

the breakup length L_{bu}, the RT model also enters into the competition to simulate the secondary breakup. The RT mechanism has preference and, if the conditions are satisfied, the breakup will take place. Otherwise, the same competition between HG and KH will occur. The breakup length is computed as:

$$L_{bu} = \frac{1}{2} C_{bu} D \sqrt{\frac{\rho_l}{\rho_g}} \tag{6.18}$$

where C_{bu} is a model constant to be tuned for each case and D represents the nozzle size.

When breakup takes place intensely, in both the HG and the KH mechanisms the children droplets strip off from the mother droplets. These new droplets are therefore small and the effect of the nozzle turbulence on them has been neglected. That means that the secondary breakup of newly generated droplets will only be affected by aerodynamic forces.

This approach neglects the influence of cavitation, being the turbulence at the nozzle exit and the aerodynamic forces due to the interaction between fuel and surrounding air the only the main factors driving the spray formation. The Huh-Gosman model is combined with the blob injection model, where the droplets are injected at a fixed diameter (nozzle diameter). The main advantage is that the distribution of the droplet injection is not required from input and that the parameters required by the user are mostly related to the injector geometry. For this reason, the application of the model is very simple when detailed experimental data are not

available.

6.3. Validation of the Hybrid Spray Model

To validate the hybrid spray model implemented in OpenFOAM®, two benchmark cases have been selected: spray A and spray G. Numerical simulations have been performed for each case, comparing the results with experimental measurements. In addition, mass conservation has been verified for each case.

ECN Spray A

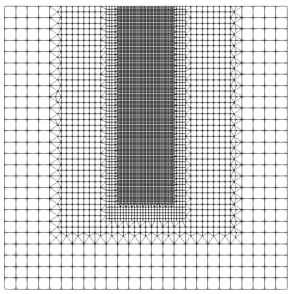
The spray A case from ECN [1] has been considered as the first benchmark case [10]. It is a diesel injector with one hole whose specifications can be found in Table 6.1.

T_{amb} $[K]$	900	Injection time $[ms]$	1.5
p_{amb} $[bar]$	60	Injection mass $[mg]$	3.56
ρ_{amb} $[kg/m^3]$	22.8	Discharge Coeff	0.89
U_{amb} $[m/s]$	< 1	Nozzle diameter $[mm]$	0.9
N holes	1	Spray cone angle $[°]$	22
Fuel p_{inj} $[bar]$	1500	Nozzle L/D	1.4
T_{fuel} $[K]$	363		

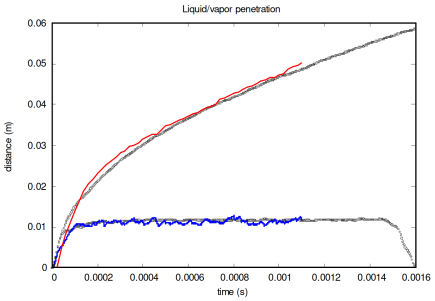
Table 6.1: ECN Spray A: injector geometry and operating conditions.

The simulation has been performed on the grid shown in Fig. 6.2a, where the smallest cell has a length of $0.125mm$, and the turbulence has been modeled with the kOmegaSST SAS model. The grid setup has been taken from the smaller possibility of [17] as it was shown to provide accurate results with RANS models. Results of the evolution of the liquid and vapor penetrations are shown in Fig. 6.2b with a red and blue line respectively. The black dots indicate the experimental results. For this case, the values of the model constants used are $B_2 = 0.2$ and $C_{bu} = 17$. It can be seen that both of them are in good agreement with the experimental results. In addition, mass is conserved during the simulation, being the mass of the system always equal to the injected mass, as shown in Fig. 6.2c.

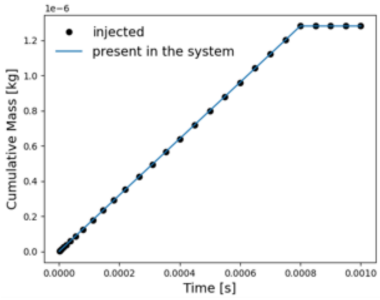
The evolution of the spray plume at five relevant time instants is shown in Fig. 6.3, where both the color and the size of the spheres represent the diameter of the liquid parcels. It can be seen how the parcels decrease their size and breakup at the breakup length approximately, until eventually evaporate completely.



(a) Spray A case: mesh



(b) Liquid (red) and vapor (blue) penetration



(c) Mass conservation

Figure 6.2: Spray A: validation.

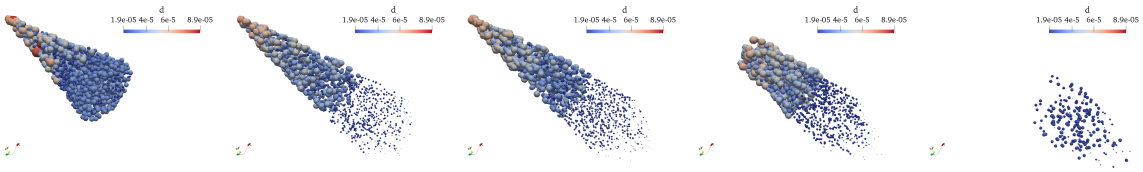


Figure 6.3: Spray A droplet size distribution

In addition, the spray plume is compared with experimental results provided by ECN [1] in Fig. 6.4, where each row represents different time instants since the spray injection. The first column represents the difference between two consecutive snapshots while the central one represents the current state of the spray. The right hand side column represents the numerical results obtained with the hybrid model implemented in this work. In all the figures, the green lines represent the limit of the gaseous spray, whereas the blue one refers only to the liquid part of it. As it is shown in the figure, the numerical results are in good agreement with the experimental ones. In particular, the liquid penetration is properly captured, as it was already shown in Fig. 6.2b. The spray shape is also captured accurately although it presents some differences. First, the gaseous region near the injector is wider in the numerical results, which is explained by the blob model introduced in the simulation, injecting parcels with the diameter of the nozzle. In addition, some perturbations appear in the surface of the spray at the break-up length, introducing a small discrepancy in the numerical solution. However, this does not affect to the simulation of the rest of the plume, which follows a smooth evolution.

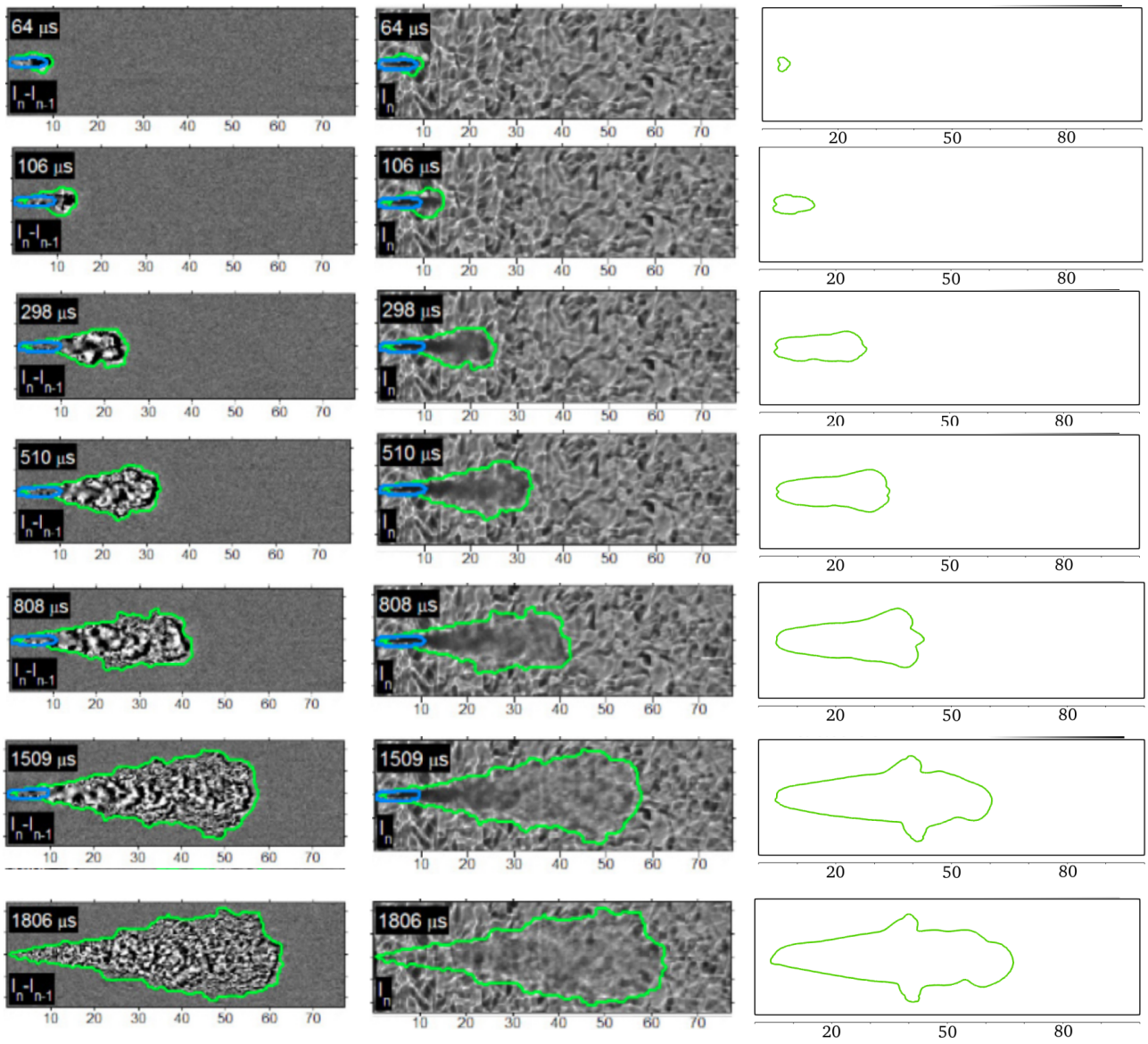


Figure 6.4: Spray A: liquid plume evolution

ECN Spray G

The second case for validation is the Spray G case [2] from ECN [1]. It is a gasoline injector with 8 holes, whose operating conditions are indicated in Table 6.2.

T_{amb}	573 K	Injection time	0.78 ms
p_{amb}	6.0 bar	Injection mass	10 mg
ρ_{amb}	3.5 kg/m^3	Discharge Coeff	179.5
U_{amb}	< 1 m/s	Nozzle diameter	0.17 mm
N holes	8	Spray cone angle	15°
Fuel p_{inj}	200 bar	Nozzle L/D	1.4
T_{fuel}	363 K		

Table 6.2: ECN Spray G: injector geometry and operating conditions.

The computational grid comprises two regions with varying resolutions, as illustrated in Fig. 6.5. In the vicinity of the injector nozzle, where the formation of eight sprays is anticipated, the cell size measures 0.5 mm. For the remainder of the domain, the average mesh size is 2 mm. This configuration has been chosen due to its successful performance in a similar case presented in [3].

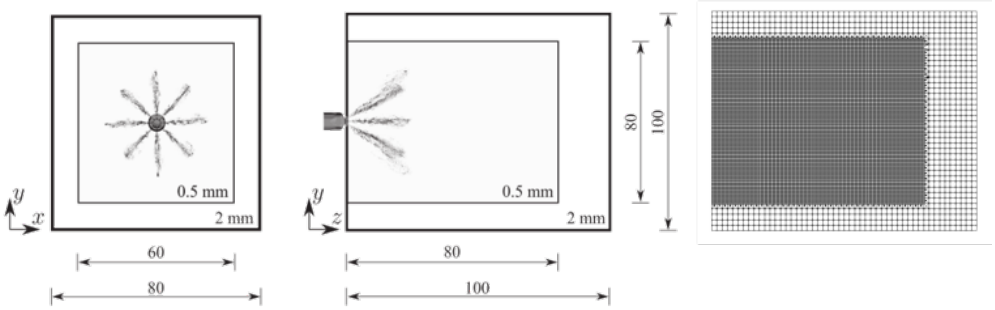
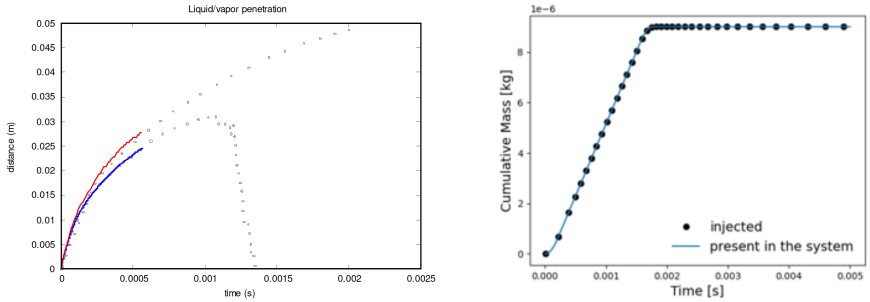


Figure 6.5: Mesh structure for the spray G case.

The results illustrating the penetration of liquid (in blue) and vapor (in red) can be seen in Fig. 6.6a, and they align well with the experimental findings. This alignment is achieved using the model constants $B_1 = 80$ and $C_{bu} = 10$. Furthermore, the mass injected by the injector matches the mass within the system, as evidenced in Fig. 6.6b, reaffirming that the implemented model effectively conserves mass throughout the simulation.



(a) Liquid (red) and vapor (blue) penetration

(b) Mass conservation

Figure 6.6: Spray G results

Finally, the evolution of the sprays is reported in Fig. 6.7, where again both the color and size of the spheres represent the diameter of the parcels. The breakup length can be clearly identified, where the RT mechanism starts competing. In addition, the mushroom shape on the tip of the sprays can be observed. Fig. 6.8 presents a comparison with experimental results provided by ECN [1], where the upper row represents the evolution of the spray measured experimentally and the bottom one the numerical results. The plots show the contour of the gaseous plume, while the red line in the experimental results represents the average value in the central plane of the spray. The results are in good agreement, both in shape and size for all the time steps. In this case, the change of breakup model taking place at the breakup length does not introduce any relevant discrepancy and the surface of the spray has a smooth evolution.

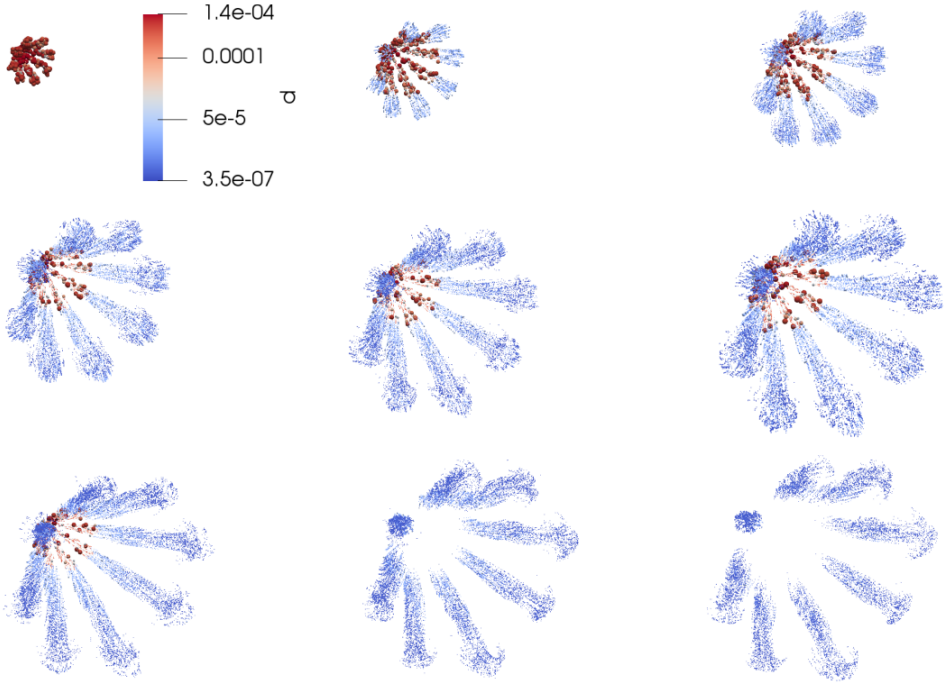


Figure 6.7: Spray G evolution

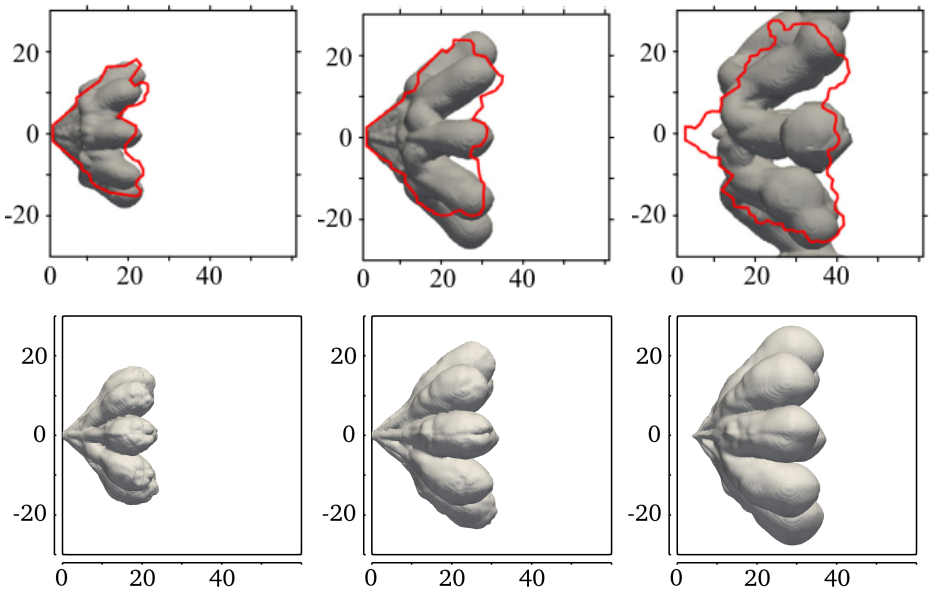


Figure 6.8: Evolution of the Spray G plume.

6.4. Overset Grid with Lagrangian Spray Modeling

The Lagrangian spray model introduced in Sec. 6.2 provides an accurate solution of the flow in a static domain. However, in most industrial applications like IC engines, the domain boundaries move and a dynamic grid is required. In these models, a Lagrangian approach is taken to simulate the liquid core and droplets of the spray, while an Eulerian approach is used for the fluid variables of the background gas in the nozzle. The coupling between both approaches is not trivial and depends largely on the chosen mesh size. The most common approach identifies all the parcels whose center lies inside of every cell at each time step. Then, the properties of all the parcels are averaged and the result is imposed in the center of the cell, allowing to couple them with the values from the Eulerian approach.

This procedure has several limitations, the principal being its mesh dependency. As the gas properties are taking from the cell centers in the Eulerian approach, all the parcels in one cell will have the same turbulent and aerodynamic interactions. However, these interactions are going to be different that the ones in the neighboring cell. Therefore, two identical parcels located infinitely close to each other but whose centers lay in different cells will have a different behavior.

In addition, most of the Lagrangian spray models present several model constants that have to be tuned for each particular case. Specifically, these model constants are dependent on the mesh in which the spray is being solved and the optimal values for one mesh resolution or configuration may not be the same than for different ones. Usually, a Cartesian mesh aligned with the injection direction is preferred as it provides more accurate results. If mesh refinement is applied locally in some regions of the domain, aiming at improving the accuracy while keeping a relatively low cell count, special techniques must be used. Reducing the cell size requires to reduce the time step proportionally to keep a low CFL number, increasing the computational time, unless special techniques, like the one in [14], are employed.

In problems with dynamic grids, additional limitations are introduced as the location of the cell centers changes with time. The value of the model constants, that were tuned for a particular grid, will no longer be valid after the mesh motion, potentially introducing a large error in the solution. Therefore, a static grid is always preferred in the region where the spray develops.

Different approaches can be used to move one part of the grid while keeping the spray region fixed. One approach is to create different mesh regions and apply the mesh motion only to some of them, while keeping others static. If a mesh motion based in topology changes is used, like in the TCC-III test case in Sec. 5.3, this condition is automatically satisfied and no further consideration is required, as long as the topology changes happen outside of the spray region.

Another approach would be to create a background mesh, independent of the spray region, and use a static overset grid for the spray. This approach allows to use a mesh optimized for each particular injector and preserves the optimum values of the model constants. In addition, a mesh optimized for the domain geometry can also be used, which is of particular relevance in complex geometries like IC engines.

As explained in Sec. 2.4, with overset grids, each grid is solved independently, using the values from the neighboring grids as boundary conditions. Usually, there is an overlapping region between both grids where the results from one grid are interpolated into the other in order to couple the whole domain. In case of having dynamic overset grids, the same procedure can be applied. However additional computational cost is required for several reasons. The first step when overset grids are used is to identify which cells correspond to each grid and, more importantly, which cells are being overlapped by the other grid. Then, some of the overlapping cells are deleted from one grid, as they are already going to be solved in the other, while leaving some overlapping cells in the boundary between grids where the solution is going to be interpolated. The weights of the interpolation are then computed based on the cell volume. Linear interpolation is widely used, even though some other techniques may be used to ensure conservation.

In the case of static grids, the identification of the cells and the calculation of the weights is only done once at the beginning of the simulation and they will not change. However, if dynamic grids are considered, this procedure has to be done each time step, identifying the overlapping cells and recomputing the interpolation weights for the new grid distribution. This procedure requires a large computational time, specially if the number of grid cells is large.

To evaluate the overset grid methodology, a new solver has been implemented using the OpenFOAM® technology. Although a dynamic overset grid solver already exists in the legacy code, it cannot handle Lagrangian spray simulations. In this study, the existing dynamic solver has been coupled with the hybrid spray model discussed in Sec. 6.2. The goal was to test scenarios where a background grid deforms over time while the overset grid remains stationary in the spray region.

For the preliminary test case, a square bomb with a vertically moving bottom face was considered, as depicted in Fig. 6.9. The deformation of the background mesh was achieved using a cell stretching technique, where all cells deformed while maintaining constant topology throughout the simulation. The spray was injected into the overset grid's center, with the background grid and overset grid dimensions of $20x20x100mm$ and $10x10x50mm$, respectively, resulting in a total of $160,000$ cells. Initially, cells in the overset region were eight times smaller than those in the background grid and decreased in size as the boundary moved. The boundary moved at

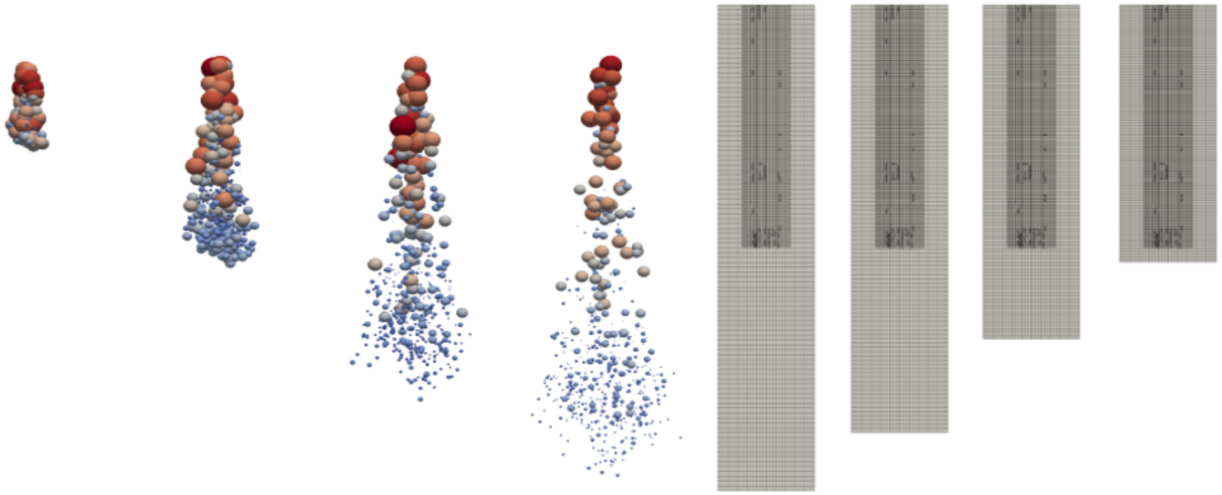


Figure 6.9: In-cylinder flow simulation: the flow transport is solved in the background deforming grids, while lagrangian spray calculations are employed on the overset mesh.

a velocity of $4m/s$, and the time step was set to $2.5\mu s$ to accurately compute spray formation.

The simulation results for the spray formation are shown in Fig. 6.9. The computational time required for this simulation was prohibitive. For each time step, approximately 90% of the time was spent recomputing interpolation weights, with only the remaining 10% dedicated to solving the fluid equations. This percentage would likely increase with larger grids, making this methodology too computationally expensive for industrial applications. Hence, a topology-based methodology like the one explained in Sec. 5.3 for the TCC-III engine, is preferred. It introduces less computational overhead while delivering accurate results.

An additional simulation was conducted to quantify the computational overhead. In this case, the same test case with the same initial mesh and boundary velocity was considered, but all the mesh cells were stretched, with each cell's deformation being proportional to the inverse of its distance to the moving wall. In this scenario, the mesh topology remained constant throughout the entire simulation. The liquid penetration for both cases was computed, and the results are shown in Fig. 6.10.

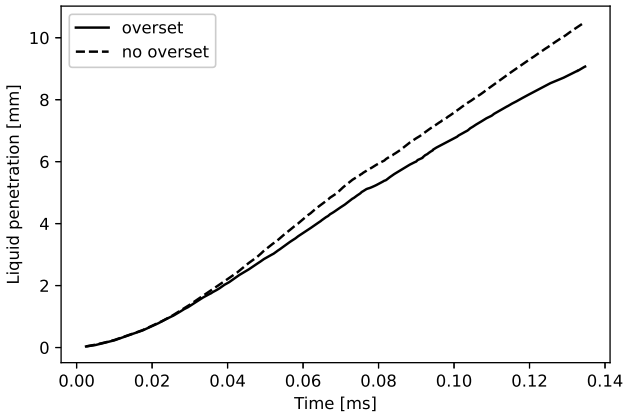


Figure 6.10: Liquid penetration: comparison between the cases with and without overset grid

As anticipated, there is a noticeable discrepancy between the outcomes of these two simulations, highlighting the impact of mesh deformation on the results. The spray with overset grid is solved on a locally static grid whereas it is solved on a deforming one in the other case. Moreover, it was found that the simulation utilizing deforming grids with constant topology was approximately 30 times faster than its counterpart employing the overset grid methodology. The overhead in computational time is unmanageable for the simulation of industrial cases, where several million cells are required. Therefore, other mesh motion techniques are preferred as they can provide results as accurate as the ones with overset grids with less computational effort.

7 | Conclusions and Future Developments

In this thesis, a series of algorithmic advancements have been introduced for the simulation of reactive flows on dynamic grids. Challenges arise when the computational domain's boundaries experience large displacements or deformations, leading to potential distortion or even invalidation of the mesh. Of the various strategies available for dynamic mesh handling, altering the grid's topology stands out as a particularly promising option. By strategically adding and removing computational cells to adapt to boundary movements, mesh quality is preserved across the majority of the domain, isolating the mesh deformation to a selected few cells.

A novel methodology has been presented to ensure second-order temporal accuracy during mesh topology modifications that result in a change in the number of cells. The absence of field data from two preceding time steps in the new topology prevents direct remapping. To navigate this issue, the proposed methodology leverages the Geometric Conservation Law (GCL) to reconstruct old-old time fields within the new topology. This reconstruction is crucial for advancing the solution in time using second-order accurate time schemes, such as the Second Order Backwards Euler (SOBE) and Crank Nicolson (CN).

The effectiveness of this methodology was tested using a one-dimensional Uniformly Accelerated Piston scenario, which provides an analytical benchmark. The results confirmed that the error introduced by cell addition or removal is negligible when compared to the spatial discretization error across all tested temporal schemes. Furthermore, the derived solution was benchmarked against another moving mesh strategy—cell stretching—which preserves grid topology. The comparison yielded a relative velocity field error below $5 \times 10^{-4}\%$ for the SOBE scheme and below $1 \times 10^{-2}\%$ for the CN scheme. The study also verified that the proposed algorithm maintains the temporal order of accuracy of the scheme, as same results were found in two different mesh motion strategies, with and without topology changes. Additionally, it was proved that the methodology conserves the mass of the system and does not create spurious mass sources in the solution, as demonstrated by the mass conservation test.

A three-dimensional lid-driven cavity test was also conducted, confirming the methodology's

validity in 3D contexts. Here, similar results have been found, proving that the relative error is independent of the ALE temporal scheme, whether it is first or second order. The algorithm's capacity to accurately capture and conserve the mass during vortex creation and convection was also demonstrated.

Finally, the algorithm developments have been tested on an industrial test case. In particular, the TCC-III Internal Combustion Engine geometry has been selected because it presents a complex fluid flow and it possesses three boundaries that undergo large, independent displacements. The initial mesh has been generated automatically with ICE Cube and the mesh motion during the whole engine cycle has been modeled through topological changes of the mesh, adding and deleting layers of cells following the motion of the boundaries, allowing for several independent topology changes to occur at the same time. This case also presents dynamic cell layering in non orthogonal cells in the valve regions. The complete setup had been validated previously for first-order accurate temporal schemes [45], which is used as benchmark. Benchmarking against first-order accurate temporal schemes validated the setup, with the results for both second-order schemes closely aligning and showing no discontinuities in regions of topology change, even with significant flow gradients.

The major limitation of the proposed methodology lies in the fact that only topology changes that involve the addition or deletion of static faces is supported. So far, the methodology has only been tested for dynamic cell layering of structured meshes. However, its application to Adaptive Mesh Refinement techniques or unstructured grids should be straight forward and future developments should explore its applicability.

Given the good results achieved in the internal combustion engine test case, and trying to take advantage of the increased accuracy in the solution, the next objective was to apply the methodology to the simulation of dual fuel engines. As traditional Lagrangian spray models can not capture all the complex phenomena appearing in the interaction of the two fuels, an advanced hybrid spray model has been implemented and validated in OpenFOAM®, considering the Huh-Gosman break model for the liquid core next to the injector. The model has been validated with the benchmark cases from ECN Spray A and G. The results are in good agreement with the experimental observations and the mass has been conserved during the simulation.

Finally, as the spray is to be injected inside an ICE (which presents a dynamic mesh), the hybrid spray model has been simulated on a simple square bomb case with a moving wall, simplifying the situation of a real industrial configuration. As it is recommended to solve the spray on a static grid, an overlapping grid that does not move has been introduced in the region where the spray plume will be formed. A new solver coupling dynamic overset grids with a Lagrangian

spray solver has been implemented in OpenFOAM® and used to simulate this case. The results showed that the time spent to compute the interpolation coefficients required by the dynamic overset grid takes around 90% of the computational time, while only the 10% is spent on solving the fluid equations. This behavior is expected to be even worse if more cells are added in the domain, making it unpractical for industrial configurations. In addition, an additional test has been performed to prove the mesh dependency of the hybrid spray, highlighting the difference between using static and dynamic grids to simulate Lagrangian sprays. The impracticality of overset grids and the accuracy loss introduced if the spray is solved on a moving grid indicate that a mesh motion technique based in topology changes is a superior option for the simulation of internal combustion engines.

As future work, it remains to validate the complete methodology, simulating an ICE geometry with LES turbulence model and second order accurate temporal schemes, adding and deleting cells to account for the boundary motion, and using the hybrid model to simulate the spray injection. This would provide state of the art results with an improved accuracy and reduced computational cost compared with the results present in the literature.

Bibliography

[1] Engine Combustion Network (ECN). https://ecn.sandia.gov.

[2] Spray G configuration. https://ecn.sandia.gov/gasoline-spray-combustion/target-condition/spray-g-operating-condition/.

[3] Horacio J. Aguerre and Norberto M. Nigro. Implementation and validation of a lagrangian spray model using experimental data of the ecn spray g injector. *Computers & Fluids*, 190:30–48, 2019.

[4] F. Alauzet. A changing-topology moving mesh technique for large displacements. *Engineering with Computers*, 30:175–200, 2014.

[5] A. Amsden, P. J. O'Rourke, and T. D. Butler. *KIVA-II: A Computer Program for Chemically Reactive Flows with Sprays*. LA 11560-MS, Los Alamos National Laboratory, 1989.

[6] Anthony Anderson, Xiaoming Zheng, and Vittorio Cristini. Adaptive unstructured volume remeshing – i: The method. *Journal of Computational Physics*, 208(2):616–625, 2005.

[7] A. J. Barlow. A compatible finite element multi-material ale hydrodynamics algorithm. *International Journal for Numerical Methods in Fluids*, 56(8):953–964, 2008.

[8] Joseph D. Baum, H. Luo, and Rainald Löhner. A new ALE adaptive unstructured methodology for the simulation of moving bodies. *Archives of Computational Methods in Engineering. AIAA Paper 1994-0414, Reno, NV, USA*, 1994.

[9] Prasad Boggavarapu and R. Ravikrishna. A review on atomization and sprays of biofuels for ic engine applications. *International Journal of Spray and Combustion Dynamics*, 5:85–122, 06 2013.

[10] Gilles Bruneaux, Louis-Marie Malbec, Laurent Hermant, Caspar Christiansen, Jesper Schramm, Lyle M. Pickett, and Caroline L. Genzale. Comparison of diesel spray combustion in different high-temperature, high-pressure facilities. *SAE International Journal of Engines*, 3(2):156–181, oct 2010.

[11] William M. Chan. *Best Practices on Overset Structured Mesh Generation for the High-Lift CRM Geometry*.

[12] Gaëtan Compère, Jean-François Remacle, Johan Jansson, and Johan Hoffman. A mesh adaptation framework for dealing with large deforming meshes. *International Journal for Numerical Methods in Engineering*, 82(7):843–867, 2010.

[13] J. Crank and P. Nicolson. A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. *Mathematical Proceedings of the Cambridge Philosophical Society*, 43(1):50–67, 1947.

[14] F. Dabonneville, N. Hecht, J. Reveillon, G. Pinon, and F.X. Demoulin. A zonal grid method for incompressible two-phase flows. *Computers & Fluids*, 180:22–40, 2019.

[15] I. Demirdžić and M. Perić. Space conservation law in finite volume calculations of fluid flow. *International Journal for Numerical Methods in Fluids*, 8(9):1037–1050, 1988.

[16] I. Demirdžić and M. Perić. Finite volume method for prediction of fluid flow in arbitrarily shaped domains with moving boundaries. *International Journal for Numerical Methods in Fluids*, 10(7):771–790, 1990.

[17] Giovanni Di Ilio, Vesselin Krastev, Federico Piscaglia, and Gino Bella. Hybrid urans/les turbulence modeling for spray simulation: A computational study. 04 2019.

[18] Jean Donea, Antonio Huerta, Jean-Philippe Ponthot, and Antonio Rodríguez-Ferran. Arbitrary lagrangian–eulerian methods. 2004.

[19] B. Duret, R. Canu, J. Reveillon, and F.X. Demoulin. A pressure based method for vaporizing compressible two-phase flows with interface capturing approach. *International Journal of Multiphase Flow*, 108:42 – 50, 2018.

[20] Charbel Farhat and Philippe Geuzaine. Design and analysis of robust ale time-integrators for the solution of unsteady flow problems on moving grids. *Computer Methods in Applied Mechanics and Engineering*, 193(39):4073–4095, 2004. The Arbitrary Lagrangian-Eulerian Formulation.

[21] Charbel Farhat, Philippe Geuzaine, and Céline Grandmont. The discrete geometric conservation law and the nonlinear stability of ale schemes for the solution of flow problems on moving grids. *Journal of Computational Physics*, 174(2):669 – 694, 2001.

[22] J.H. Ferziger, M. Perić, and R. L. Street. *Computational Methods for Fluid Dynamics*. Springer, 4th edition edition, 2020.

[23] Rao Garimella, Milan Kucharik, and Mikhail Shashkov. An efficient linearity and bound preserving conservative interpolation (remapping) on polyhedral meshes. *Computers & Fluids*, 36(2):224–237, 2007.

[24] Philippe Geuzaine, Céline Grandmont, and Charbel Farhat. Design and analysis of ale schemes with provable second-order time-accuracy for inviscid and viscous flow simulations. *Journal of Computational Physics*, 191(1):206–227, 2003.

[25] H. Guillard and C. Farhat. On the significance of the geometric conservation law for flow computations on moving meshes. *Comput. Methods Appl. Mech. Engrg.*, 190:1467–1482, 2000.

[26] O. Hassan, K. A. Sørensen, K. Morgan, and N. P. Weatherill. A method for time accurate turbulent compressible fluid flow simulation with moving boundary components employing local remeshing. *International Journal for Numerical Methods in Fluids*, 53(8):1243–1266, 2007.

[27] Charles Hirsch. *Numerical Computation of Internal and External Flows*. Elsevier, 2007.

[28] C. Hirt. *An arbitrary Lagrangian-Eulerian computing technique*, volume 8, chapter 1, pages 350–355. Springer, 01 1971.

[29] C.W Hirt, A.A Amsden, and J.L Cook. An arbitrary lagrangian-eulerian computing method for all flow speeds. *Journal of Computational Physics*, 14(3):227–253, 1974.

[30] KY Huh. A phenomenological model of diesel spray atomization. In *Proc. the International Conf. on Multiphase Flows, 1991*, 1991.

[31] H. Jasak. *Error analysis and estimation in the Finite Volume Method with applications to fluid flows*. PhD thesis, Imperial College, University of London, 1996.

[32] H. Jasak and Z Tukovic. Automatic Mesh Motion for the Unstructured Finite Volume Method. *Transactions of FAMENA*, 30(2):1–18, 2007.

[33] Hrvoje Jasak, Henry G Weller, and Niklas Nordin. In-cylinder CFD simulation using a C++ object-oriented toolkit. In *SAE Technical Paper 2004-01-0110*, page 20. SAE International, Apr 2004.

[34] Milan Kucharik, Mikhail Shashkov, and Burton Wendroff. An efficient linearity-and-bound-preserving remapping method. *Journal of Computational Physics*, 188(2):462–471, 2003.

[35] L. D. Landau and E. M. Lifshitz. *Physique Théorique – Tome VI Mécanique del Fluides*. Éditions de Moscou, 1971.

[36] Zhi-Hua Li, Bq He, and Hua Zhao. Application of a hybrid breakup model for the spray simulation of a multi-hole injector used for a disi gasoline engine. *Applied Thermal Engineering*, 65:282–292, 04 2014.

[37] Raphaël Loubère, Pierre-Henri Maire, and Mikhail Shashkov. ReALE: A Reconnection Arbitrary-Lagrangian–Eulerian method in cylindrical geometry. *Computers & Fluids*, 46(1):59–69, 2011. 10th ICFD Conference Series on Numerical Methods for Fluid Dynamics (ICFD 2010).

[38] Raphaël Loubère, Pierre-Henri Maire, Mikhail Shashkov, Jérôme Breil, and Stéphane Galera. Reale: A reconnection-based arbitrary-lagrangian–eulerian method. *Journal of Computational Physics*, 229(12):4724–4761, 2010.

[39] R. Löhner. Three-dimensional fluid-structure interaction using a finite element solver and adaptive remeshing. *Computing Systems in Engineering*, 1(2):257–272, 1990. Computational Technology for Flight Vehicles.

[40] L.G. Margolin and Mikhail Shashkov. Second-order sign-preserving conservative interpolation (remapping) on general grids. *Journal of Computational Physics*, 184(1):266–298, 2003.

[41] L. Germes Martinez, B. Duret, J. Reveillon, and F.X. Demoulin. A new dns formalism dedicated to turbulent two-phase flows with phase change. *International Journal of Multiphase Flow*, 143, 2021.

[42] L. Germes Martinez, B. Duret, J. Reveillon, and F.X. Demoulin. Vapor mixing in turbulent vaporizing flows. 161, 2023.

[43] Diogo M.C. Martins, Duarte M.S. Albuquerque, and José C.F. Pereira. On the use of polyhedral unstructured grids with a moving immersed boundary method. *Computers & Fluids*, 174:78–88, 2018.

[44] Rajat Mittal and Gianluca Iaccarino. Immersed boundary method. *Annu. Rev. Fluid Mech.*, 37:239–261, 01 2005.

[45] Andrea Montorfano, Federico Piscaglia, and Angelo Onorati. An Extension of the Dynamic Mesh Handling with Topological Changes for LES of ICE in OpenFOAM. In *SAE Technical Paper 2015-01-0384*, page 20. SAE International, Apr 2015.

[46] Scott Morton, David Mcdaniel, and Russell Cummings. F- 16xl unsteady simulations for the cawapi facet of rto task group avt-113. *Aerospace Engineering*, 9, 01 2007.

[47] Fabio Nobile. *Numerical approximation of fluid-structure interaction problems with application to haemodynamics*. PhD thesis, Ecole Polytechnique Federale de Lausanne (EPFL), 2001.

[48] L. Palanti, S. Puggelli, L. Langone, A. Andreini, J. Reveillon, B. Duret, and F.X. Demoulin. An attempt to predict spray characteristics at early stage of the atomization process by

using surface density and curvature distribution. *International Journal of Multiphase Flow*, 147, 2022.

[49] S.V. Patankar. *Numerical Heat Transfer and Fluid Flow*. Hemisphere, New York, 1980.

[50] J. Peinado, J. Ibáñez, E. Arias, and V. Hernández. Adams–Bashforth and Adams–Moulton methods for solving differential Riccati equations. *Computers & Mathematics with Applications*, 60(11):3032 – 3045, 2010.

[51] Charles S Peskin. Flow patterns around heart valves: A numerical method. *Journal of Computational Physics*, 10(2):252 – 271, 1972.

[52] F. Piscaglia. *Developments in Transient Modeling, Moving Mesh, Turbulence and Multiphase Methodologies in OpenFOAM*. Keynote Lecture at The 4th Annual OpenFOAM User Conference, October 11-13, 2016, October 11th-13th 2016.

[53] F. Piscaglia, A. Montorfano, and A. Onorati. Development of Fully-Automatic Parallel Algorithms for Mesh Handling in the OpenFOAM-2.2.x Technology. *SAE Technical Paper 2013-24-0027*, 2013.

[54] F. Piscaglia, A. Montorfano, and A. Onorati. A Moving Mesh Strategy to Perform Adaptive Large Eddy Simulation of IC Engines in OpenFOAM. In *International Multidimensional Engine Modeling User's Group Meeting 2014, The Detroit Downtown Courtyard by Marriott Hotel, Detroit, MI (USA)*, page 20, April 7th 2014. [download].

[55] B. Re, C. Dobrzynski, and A. Guardone. An interpolation-free ale scheme for unsteady inviscid flows computations with large boundary displacements over three-dimensional adaptive grids. *Journal of Computational Physics*, 340:26–54, 2017.

[56] Rolf Reitz. Modeling atomization processes in high-pressure vaporizing sprays. *Atomisation Spray Technology*, 3:309–337, 01 1987.

[57] C.M. Rhie and W.L. Chow. A numerical study of the turbulent flow past an isolated airfoil with trailing edge separation. *AIAA J.*, 21:1525–1532, 1983.

[58] Philipp Schiffmann, Saurabh Gupta, David Reuss, V. Sick, Xiaofeng Yang, and Tang-wei Kuo. Tcc-iii engine benchmark for large-eddy simulation of ic engine flows. *Oil & Gas Science and Technology*, 71, 01 2015.

[59] Matthew L. Staten, Steven J. Owen, Suzanne M. Shontz, Andrew G. Salinger, and Todd S. Coffey. *Proceedings of the 20th International Meshing Roundtable*, chapter A Comparison of Mesh Morphing Methods for 3D Shape Optimization, pages 293–311. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[60] TCC-III Input Dataset, University of Michigan. [link].

[61] The OpenFOAM® Foundation. [link].

[62] J. G. Trulio. Report No. AFWL- TR-66-19. Technical report, Air Force Weapons Laboratory, Kirtland Air Force Base, 1961.

[63] J. G. Trulio and K. R. Trigger. Numerical solution of the one-dimensional hydrodynamic equations in an arbitrary time-dependent coordinate system. Technical report, University of California Lawrence Radiation Laboratory, Report UCLR-6522, 1961.

[64] Z. Tukovic and H. Jasak. A moving mesh finite volume interface tracking method for surface tension dominated interfacial fluid flow. *Computers & Fluids*, 55:70–84, 2012.

[65] Zeljko Tukovic, Aleksandar Karač, Philip Cardiff, Hrvoje Jasak, and Alojz Ivankovic. Openfoam finite volume solver for fluid-solid interaction. *Transactions of FAMENA*, 42:1–31, 10 2018.

[66] Rong Wang, Patrick Keast, and Paul Muir. A comparison of adaptive software for 1d parabolic pdes. *Journal of Computational and Applied Mathematics*, 169(1):127–150, 2004.

List of Figures

List of Tables

Abbreviations

ALE	Arbitrary Lagrangian-Eulerian
AMI	Arbitrary Mesh Interface
AMR	Adaptive Mesh Refinement
BDC	Bottom Dead Center
CA	Crank Angle
CFD	Computational Fluid Dynamics
CFL	Courant–Friedrichs–Lewy
CN	Crank-Nicolson
CPU	Central Processing Unit
CV	Control Volume
DFICE	Dual-Fuel Internal Combustion Engine
DGCL	Discrete Geometric Conservation Law
ECN	Engine Combustion Network
FV, FVM	Finite Volume, Finite Volume Method
GCL	Geometric Conservation Law
HCCI	Homogeneous Charge Compression Ignition
IBM	Immersed Boundary Method
IC, ICE	Internal Combustion, Internal Combustion Engine
KHRT	Kelvin–Helmholtz Rayleigh–Taylor spray model
Layer A/R	Layer Addition Removal
LES	Large-Eddy Simulations
MPI	Message Passing Interface
NASA	National Aeronautics and Space Administration
PDE	Partial Differential Equation
PIMPLE	Transient SIMPLE

PISO	Pressure-Implicit with Splitting of Operators
RANS	Reynolds Average Navier-Stokes
SIMPLE	Semi-Implicit Method for Pressure Linked Equations
SOBE	Second Order Backward Euler
TCC	Transparent Combustion Chamber
TDC	Top Dead Center
UAP	Uniformly Accelerated Piston