POLITECNICO DI TORINO

I Faculty of Engineering

Master of Science in Aerospace Engineering

Master Thesis

Implementation of level set methods for moving interfaces on adaptive grids

Supervisor:
prof. Francesco Larocca

Co-supervisor:
prof. Sergio Ricci

Candidate:
Matteo Loss

Company supervisor
Optimad Engineering Srl
PhD. Haysam Telib

March 2018

Acknowledgements

Un grazie sentito spetta a chi, con il suo supporto, mi ha permesso di lavorare serenamente a questa monografia, aiutandomi ad affrontare tutti gli ostacoli che si sono presentati durante il suo svolgimento.

Innanzitutto, la mia gratitudine e riconoscenza va alla *Optimad Engineering srl*, in particolare al dott. Haysam Telib, il quale mi ha dato l'opportunità di mettermi in gioco e guidato nella stesura di questa tesi. La sua pazienza ed i suoi preziosi consigli, hanno contribuito al raggiungimento dei risultati sperati. Desidero, inoltre, ringraziare tutti i collaboratori dell'azienda, che mi hanno accolto calorosamente tra di loro e dimostrato disponibilità gratuita in ogni situazione.

Alla mia famiglia riservo un ringraziamento particolare per il sostegno regalatomi durante tutti i miei anni di studio, anche e soprattutto nei momenti di difficoltà.

Ringrazio mia madre per l'amore e la pazienza con cui mi ha fatto comprendere e condividere il valore più profondo dell'istruzione, della cultura e della fatica, strumenti indispensabili per "sognare" i propri obiettivi e realizzarli. Ringrazio mio padre per avermi sempre spinto a diventare la versione migliore di me stesso, insegnandomi a sfidare i miei limiti ma ricordandomi anche che, nonostante i meriti personali, non sempre tutto va come auspicato.

Infine, ma non per ultima, vorrei ringraziare Giulia, che mi ha accompagnato, tenendomi per mano, durante ogni giorno di questo viaggio. La sua energia, sempre pronta a rivitalizzarmi dopo ogni momento di sconforto, la sua intelligenza, capace di stimolarmi e rallegrare le mie giornate, e la sua dolcezza sono i "pilastri" che mi sorreggono e rafforzano quotidianamente.

Abstract

In this thesis, different level set techniques were developed to propagate the signed distance function values, from the cells adjacent to the body, in the rest of the computational domain. Furthermore, also an high-order transport scheme was implemented in order to modify the level set field through time.

Regarding propagation, two method were designed, both based on the solution of the heat equation to obtain the distance field from an object. The former is an iterative fixed-point approach where the final recovery of the distance field was achieved by solving an elliptic problem. In this case, the main dilemma was to find suitable boundary conditions for the final elliptic equation since the distance calculation, being an hyperbolic problem, should not be influenced by them. Two different type of boundary conditions were considered. The latter procedure starts from the solution of an initial heat problem followed by some Newton's method iterations to refine the distance field accordingly to the Eikonal equation.

The high-order implementation of the transport equation is based on a Finite Volume approach using a Compact WENO reconstruction procedure for Octree grids.

In conclusion, a final test case was performed for showing the potentiality of the proposed methodologies combined with a computational fluid dynamics solver.

Contents

List of Figures

Chapter 1

Introduction

1.1 Computational fluid dynamics

The computational fluid dynamics (CFD) is a discipline within the fluid mechanics subject that has the purpose to study and solve problems related to fluid flows [2]. Due to the complexity of the equations involved, namely the Navier-Stokes equations, an analitical solution is still not found nowdays, except for quite simple cases. Therefore, a numerical approach to obtain an accurate approximation is needed and CFD is the branch of computational physics that aims to provide the methodoligies to get this estimation.

This latter is made using a discretization method, which transform the partial differential equations involved into a set of algebraic mathematic expressions. Intuitively, the precision of the results is directly dependent on the quality of the discretization employed. The final output of a simulation are numbers that represent the value of flow field properties in discrete points in time and space. Since to have an accurate flow representation it is necessary to evaluate fluid's variables on the highest number of points possible, these calculations have to be performed with the aid of a computer. An example of the results are depicted in figure 1.1.

The main elements of a numerical solution method in a CFD solver are [8]:

- Mathematical model: it is the system of equations with its boundary and initial conditions. It may differs from one application to an other since, in particular cases, some hypothesis can be done in order to simplify the resolution.

- Discretization Method: it represents the method used to transform the partial differential equations into algebraic expressions. The main approaches are: finite difference, finite volume and finite element methods.

- Coordianate and basis vector system: they influence the form of the equations involved. Their choice depends on the selcted application and on the geometry of the domain. They could influence the discretization method and the grid used.

- Numerical grid: it is the gathering of all the discrete point in space where the solution of equations is calculated. It will be explained in details in the next section.

- Finite approximation: depending on the discretization method, different approximation can be done. They influence the accuracy of the solution, the ease of implementation and the computational efficiency.

- Solution method: the discretization phase produces large system of algebraic equation which have to be solved. Usually, some iterative techniques are used and the solver choice depends on the kind of grid and its nodes' number.

- Convergence criteria: the iterative method needs to be stopped when convergence has been reached. Therefore, an important part of the whole procedure is to set a suitable criteria to establish whether the problem is solved or not.

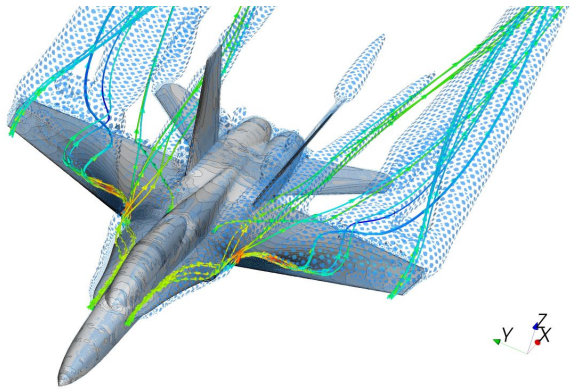


Figure 1.1: CFD simulation of the air flowing on an F18 aircraft at high angle of attack.

Following this short preamble, the complete Navier-Stokes equations should be introduced. In the differential form, they read

$$\begin{cases} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho\, \mathbf{q}) = 0 \\ \\ \frac{\partial}{\partial t}(\rho\, \mathbf{q}) + \nabla \cdot (\rho \mathbf{q} \otimes \mathbf{q}) = \nabla \cdot \overline{\overline{\Pi}} + \rho \mathbf{f}_V \\ \\ \frac{\partial}{\partial t}(\rho E) + \nabla \cdot (\rho E \mathbf{q}) = \nabla \cdot \left(\overline{\overline{\Pi}} \cdot \mathbf{q}\right) - \nabla \cdot \mathbf{q}_t + \rho \mathbf{f}_V \cdot \mathbf{q} \end{cases} \tag{1.1}$$

where $\mathbf{q}$ is the vector velocity, ρ is the density, $\overline{\overline{\Pi}}$ is the tensor of the stress tensor,$\mathbf{q}_t$ is the vector containing the heat fluexes on the three axis, $\mathbf{f}_V$ is a vector with the volume forces, E is the internal energy. The first of the three equation express the mass conservation, the second the momentum conservation and the third the energy conservation. This is a system of 5 equations (the momentum conservation is a vectorial equation) while the unknowns are 14. In order to have a well determined sistem of equations, some assumption needs to be done.

First, the fluid is considered to be perfect, in this way the perfect gas laws can be applied. They read

$$\begin{cases} e = C_v T \\ \\ \frac{p}{\rho} = RT \end{cases} \tag{1.2}$$

where e is the internal energy of the fluid, C_v is the specific heat at constat volume, R is the specific gas constant, p is the pressure and T is the temperature. This add 2 equations to our system.

Secondly, for a Newtonian fluid the so called kinematic relations hold. The $\overline{\overline{\Pi}}$ can be written as the sum of the normal and shear stresses like

$$\overline{\overline{\Pi}} = -p\overline{\overline{I}} \cdot \mathbf{n} + \overline{\overline{\tau}} \cdot \mathbf{n}$$

where $\overline{\overline{I}}$ is the identity matrix. The kinematic relations, written with the indicial notation, can be expressed as

$$\tau_{ij} = \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) - \frac{2}{3}\mu \frac{\partial u_i}{\partial x_i} \delta_{ij} \tag{1.3}$$

where u represents the components of the velocity vector, μ is the dynamic viscosity fo the fluid and δ_{ij} is the Kroneker delta operator.

With this relationships, 6 new equations are added to the previous 7 but at the same time also 6 unknown (the ones in the $\overline{\overline{\Pi}}$ symmetric tensor) are replaced by 7 unknown (the components of $\overline{\overline{\tau}}$ tensor and the p). The last passage for obtain a well determined system of equations is given by the Fourier's law which state that

$$\mathbf{q}_t = -k\nabla T \tag{1.4}$$

which express the heat flux as a function of the constant k and a new unknonw, namely the temperature T. However, being a vectorial equation this law add 3 equations to our system and 1 new unknown. The result is a sytem of 16 equation with 16 unknowns which is well determined and can be solved numerically.

The equations written until now are the differential Navier-Stokes equations with conservative variables. Therefore, when they are integrated and discretized on the appropriate control volumes, the total variation in time of the conservative variables is only due to the entering or exiting fluxes on domain's borders. Moreover, a convenient form to write the equations is the vectorial form and, in the most general situation this reads as

$$\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} + \frac{\partial H}{\partial z} = J \tag{1.5}$$

where U, F, G, H and J are columns vectors containing, respectively, the unknowns, the fluxes of the quantities on x, y and z axis and the right hand side vector. These vectors

can be explicitly written as

$$
\begin{aligned}
U &= \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{bmatrix} \\
F &= \begin{bmatrix} \rho u \\ \rho u^2 + p - \tau_{xx} \\ \rho v u - \tau_{xy} \\ \rho w u - \tau_{xz} \\ \rho E u + p u - k\dfrac{\partial T}{\partial x} - u\tau_{xx} - v\tau_{xy} - w\tau_{xz} \end{bmatrix} \\
G &= \begin{bmatrix} \rho v \\ \rho u v - \tau_{yx} \\ \rho v^2 + p - \tau_{yy} \\ \rho w v - \tau_{yz} \\ \rho E v + p v - k\dfrac{\partial T}{\partial y} - u\tau_{yx} - v\tau_{yy} - w\tau_{yz} \end{bmatrix} \\
H &= \begin{bmatrix} \rho w \\ \rho u w - \tau_{zx} \\ \rho v w - \tau_{zy} \\ \rho w^2 + p - \tau_{zz} \\ \rho E w + p w - k\dfrac{\partial T}{\partial z} - u\tau_{zx} - v\tau_{zy} - w\tau_{zz} \end{bmatrix} \\
J &= \begin{bmatrix} 0 \\ \rho f_x \\ \rho f_y \\ \rho f_z \\ \rho \left(u f_x + v f_y + w f_z \right) \end{bmatrix}
\end{aligned}
\tag{1.6}
$$

The numerical solution to these equations gives back, for each temporal steps, the physical properties of the flow in each discrete point of the considered domain. Moreover, these latter system could be simplified if some assumption are made like, for example, inviscous or incompressible flow. Obviously, being PDEs, in order to be solved, they need to have an initial condition and the boundary condition at the borders of the domain. Thanks to these inputs, the initial flow field with its properties can evolve through time. Then, if the considered problem has a stationary solution, the evolution stops and the system reaches its steady state, otherwise the evolution continues indefinetively, until it is stopped at the desired time.

There are many methods to solve these equations. The models devoloped have always been more accurate and reliable over time. One of the hardest phenomena to be correctly resolved with CFD is the flow's turbulence. In particular, three main approaches can be found for predicting turbulent fluid structure:

- *Direct Numerical Simulation (DNS)*: conceptually it is the easiest way to solve the equations, in the sense that once the grid choice has been done then the Navier-Stokes equations are solved on every node. The difficulty of this procedure relies in the complexity of flow phenomena that need to be calculated over a huge number of cells to be correctly resolved.

- *Reynolds Averaged Navier-Stokes (RANS)*: The assumption behind this procedure is that the turbulence of the flow can be decompose into a mean flow field and perturbations to it. The equation's variables are not istantaneous values but they are averaged over a certain time, in a way that turbulent phenomena take place in a lower temporal scale but the whole evolution of the mean field scales with an higher time. This solution leaded to drastically lower the time needed for a simulation with respect to the DNS simulations, at the same time the results can not be as much accurate as the other approach.

- *Large Eddy Simulation (LES)*: it is based on the numerical solution of the larger turbulent scale while the little scale are just modelled. The physical principle behind this approach is that the bigger turbulent scale, after their formation, transfer energy to the little ones, therefore these latter can be just represented as a flow's energy sink. Due to the forementioned features, this kind of simulation are more accurate than the RANS but much cheaper, from the computational's cost point of view, than the DNS.

1.2 Discretized computational domain

A fundamental part for simulating the fluid is the choice of the space's discrete representation where the flow evolves. In fact, the solution of the previous equations is calculated in specific discrete points called nodes and their ensable essentially represents the geometric domain on which the problem is to be solved. Therefore the solution is divided into a finite number of subdomains called cells (or elements, ecc...). This section is intended to introduce the different discretizations that can be adopted in order to represent the space where the solution to PDEs is desired.

A first classification of the kinds of grid is the following:

- *Structured grid*: they are made by meshes where grid lines (lines that divide the different cells) of a particular family do not cross each other and encounter each member of other families only once like in figure 1.2. This allows the lines to be numbered consecutively and any grid point or cell can be uniquely identified with two or three indexes. The main advantage is that this kind of grids are very simple to handle but, on the other hand, they can be applied only to simple geometries and the points distribution is difficult to be controlled.

- *Block-structured grid*: in this type of mesh there are, at least, two subdivision of the solution domain. They can present different grid type allowing a better resolution near important flow's region or improving the fitting close to the body (figure 1.3).

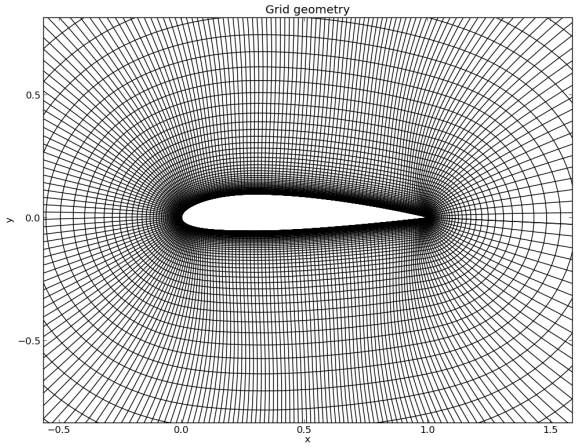


Figure 1.2: Structured grid created around a wing profile.

A crutial point of this grid's class is at block interfaces where particular attention needs to be taken. Two popular examples of these meshes are the so called *Chimera grid* and the *Boundary-Fitted Non-Orthogonal grid* that are going to be analyzed later.

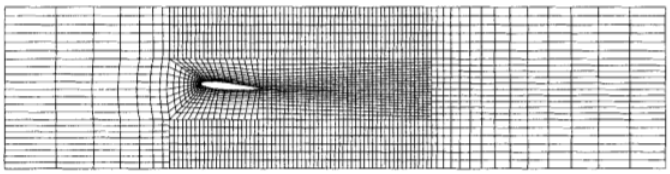


Figure 1.3: Block grid created around a wing profile.

- *Unstructured grid*: they are the most flexible type of grid and the control's volumes may have any shape and there is no restriction on the connetivity between cells. In order to generate them it is necessary to develop specific algorithms and another negative is that the resulting data structure is irregular. Therefore, matrixes are more complex to be solved and in the end the overall time needed for the simulation is higher.

In addiction, an important feature which influences the properties of a mesh is the matching at cells' interfaces of the nodes. If the neighbour cells match all their nodes on the shared interface, then the grid is called *conformal*, while if there is only a partial or no matching it is said to be *non-conformal* (figure 1.5). The positives of the first are that usually results are more accurate and, since no interpolation is needed, they are obtained

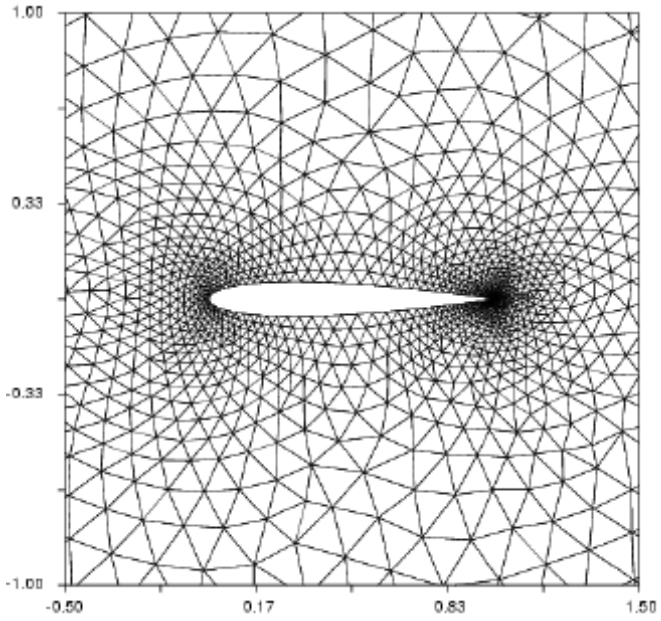


Figure 1.4: Unstructured grid created around a wing profile.

faster. In the other case, an interpolation at the so called "hanging node"[1] is required but the mesh is much more flexible. All the forementioned grid's categories can be conformal or not and, in particular, the block-structured meshes can be non-conformal even if all its blocks are conformal since at blocks interface there could be hanging nodes.

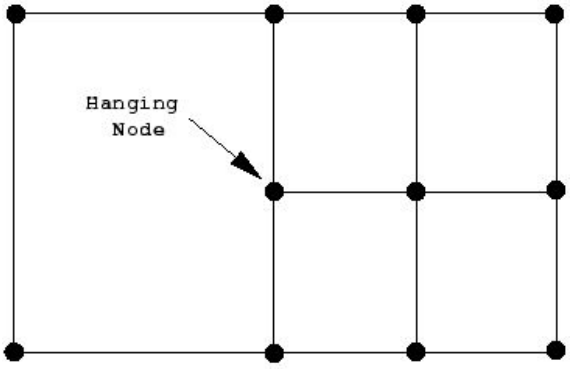


Figure 1.5: An example of hanging node in a non-conformal cartesian mesh.

It is easy to imagine that the most suitable kind of meshes to fit complicated geometries are the unstructured ones and a useful feature for this purpose is the non-conformity.

[1]Nodes that are not shared between two neighbour cells

However, there are also other existing solutions that were developed and they will be presented in the next paragraphs [8].

One of the simplest choice is to use the so called *stepwise regular grids* which employ regular cartesian grids but when the border of the geometry is curved or inclined the boundaries are approximated by staircase-like steps as in figure 1.6.

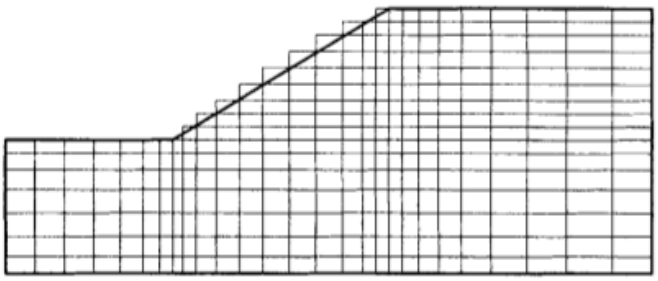


Figure 1.6: Stepwise grid applied to inclined boundary.

Of course this choice presents some clear disadvantages. The main one is the unsatisfactory approximation done when the geometry has a boundary not aligned with the reference system. Moreover, this approach creates two main problems that are the variable number of cells per line, which increases the complexity of the computer coding, and also the errors introduced in the solution by the steps itself, particularly when the grid is coarse. The only advatage of this solution is that it is quite easy to implement with respect to other possible approach.

Another solution is to adoperate the *Chimera grids* [11], which cover irregular domain with a combination of many regular grids like in figure 1.7. The main disadvantage relies in the difficulty of programming the coupling of different grid. In addiction, passing data from a grid to an other, within an overlapping region, requires an interpolation. This could introduce errors and decresing the accuracy of the solution. On the other hand, the positive is that, once the programming difficulties are overcomed, the method is suitable for handling moving bodies in the computational domain. In fact, the only additional difficulty is to calculate, for each time step, which regions of the various grids overlap and then apply the interpolation procedure to exchange data between meshes.

A further popular choice is the *Boundary-Fitted Non-Orthogonal grids* as shown in figure 1.8. These meshes can be both conformal or not and they are generated from the boundaries of the geometry using mathemathical calculation. In this way, the nodes are positioned in order to have a better resolution in regions where the flow could present strong variations. Furthermore, the lines follow the boundaries making easier to implement the boundary conditions. One disadvantage of this approch is that the orthogonality, and important feauture of a grid, is not always preserved, leading to a higher difficulty of programming and increasing the computational cost for solve the equations. Furthermore, the time required to create these meshes is high and their use is limited to the specific application that they were created for. Carefulness is needed in the developing phase

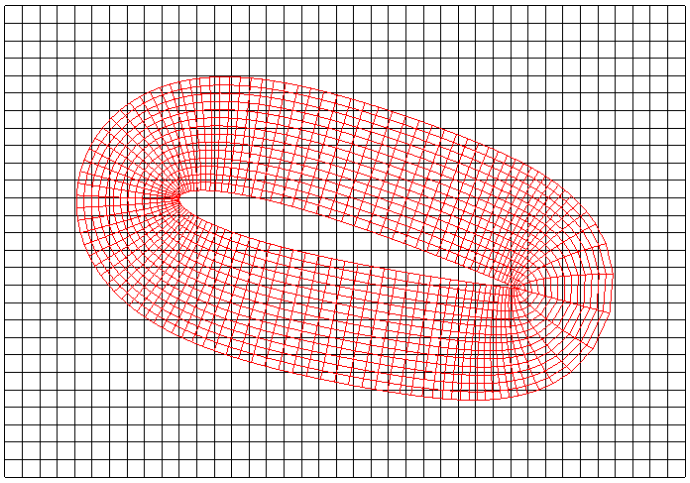


Figure 1.7: Chimera grid applied to an airfoil.

and no automatization of the process is possible. The industrial sectors where these grids raise interest are the ones that have many resources to invest for obtaining accurate results without strict time constrains. An example is the aeronautical sector, where the wing always have almost the same geometry and, since projects last long time, the main concern is to get precise outcomes.

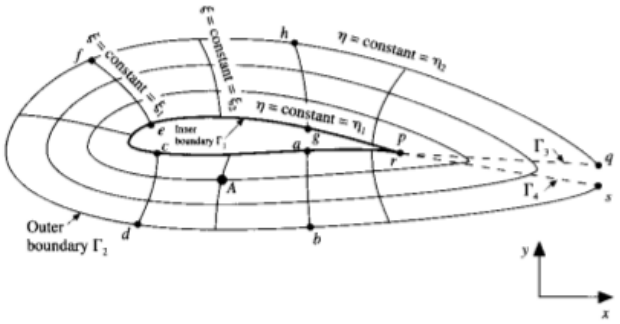


Figure 1.8: Boundary-Fitted non-orthogonal grid applied to an airfoil.

The last kind of mesh, which could be considered as the "antithesis" of the previous grid type, are the so called *Octree grids* (in three dimensions while in two are called *Quadtree mesh*) depicted in figure 1.9. They can be classified as a Cartesian non-uniform and non-conformal mesh obtained from the refinement of an initial grid where certain cells are divided in eight (or four) identical children. The generation of these grids can be obtained recursively using devoted algorithm which refine the mesh, also during run-time, in regions where the solution presents strong gradients or where an higher accuracy is desired. Since these meshes are so simple, their creation can be easily automated and being cartesian, although not conformal, they assure good results. In principle, they could not fit complex geometries. However, using a particular technique that will be introduced

in the next section, they can be used for any type of body and their feature to be arbitrary refined is particularly useful. For these reasons they can be considered as the opposite side of the coin of the Boundary-Fitted Non-Orthogonal grids. In fact, they do not have to conform to the body and their creation is trivial, thus requiring no-effort and their range of applications is much larger, even if the accuracy could not be as high as in the other case.

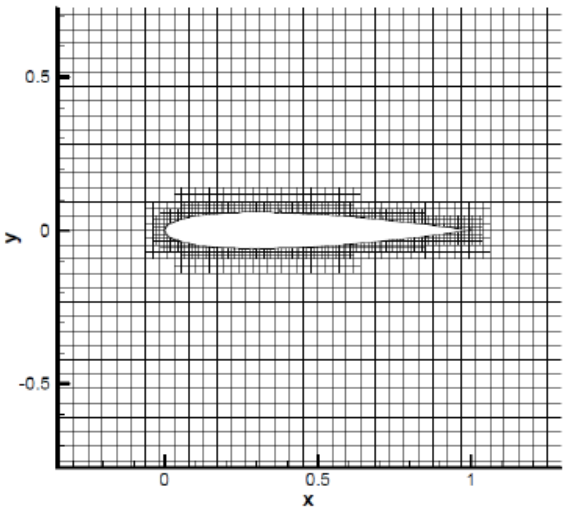


Figure 1.9: Quadtree grid applied to an airfoil.

To sum up, a suitable grid for a real application's geometry can be very difficult to be generated and the time needed to do it could exceed the one for calculate the flow solution. In addiction, having moving bodies inside the calculus loop is quite difficult and for these simulation the computational cost could become prohibitive. In fact, having to create a new mesh while the body is moving, or worst when it change its shape, takes an high ammount of time. For this reasons, techniques to deform the mesh following the body's distorions have been created. However, their range of application is restricted to little modifications of the object's shape and the computational cost is large. Consequently, when moving or deforming bodies are involved in the simulation, or when the time available to generate a mesh is short, a good compromise for obtaining accurate results is to employ the Octree grid in conjunction with the technique explained in the next section.

1.3 Immersed boundaries techniques

The *Immersed boundary method* is a procedure that was introduced by Peskin in 1972 [19] to simulate the blood flow during the cardiac beating. The interesting feature of this novelty was that the mesh employed did not conform to the geometry of the heart but instead some of its nodes were placed inside the solid boundaries of the body. Therefore,

a new technique was developed in order to impose the boundary conditions at nodes immersed in the flow but close to a border of the fluid region placed inside the grid.

The main innovation of this technique is that, taking a generic body, the mesh do not have to perfectly fit the geometry, thus being easy to create. A common choice is to use a regular cartesian volume grid, which is the easiest grid that can be applied, without having any regards on the position of the body's boundary. Naturally, this end up with the solid boundary cut through the choosen grid as depicted in figure 1.10. For this characteristic two problems arise: the interface between the body and the sorrounding space, if it is moving, must be tracked through time; the boundary conditions have to be applied to the correct nodes in the proximity of the body's border.

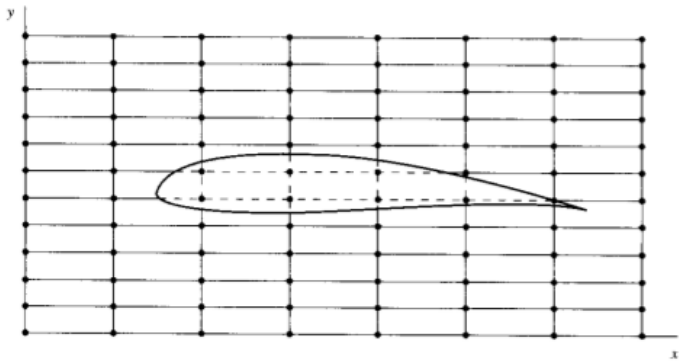


Figure 1.10: Cartesian grid applied to an airfoil with nodes inside the body itself.

Assuming that the boundary nodes are known, the application of the boundary condition is generally done by introducing a fictious distribution of forces in the governing equation thus obtaining a flow behaviour consistents with the presence of a boundary[6]. The application of these forcing terms can be done mainly in two ways, which define two distinct approaches[16]: the continous forcing method and the discrete forcing one.

The former technique introduces the fictious forces directly as a right hand side term in the governing equations of the entire domain. Then these equations are discretized over the cartesian grid and solved in all the nodes. The proposed approach is particularly well suited for elastic body's boundaries where their motion is tracked using a Lagrangian procedure employing massless particle attached to the border. However, when a problem with rigid boundary is faced, this method becomes complicated since the laws for elastic boundaries are not well posed for the rigid ones and thus some correction have to be done. The positives of the presented methodology is the easier by which the discretization of elastic borders motion is done while the main negatives relyies in the fact that the system of equations is solved in all grid's points even if they are located inside the solid boundary, leading to an higher computational time.

The latter approach starts from the complete discretization of the governing equation regardless of the immersed boundaries. Later, in the nodes near the IB the discretized equations are modified to take into account borders and then the solution of the equations' system takes place. In this case, the imposition of BCs can happen indirectly or directly.

In the first situation the force is applied from an a priori estimation of the solution without considering the presence of IB. In the second one, the stencil near the borders are modifyied thus imposing the correct condition to that nodes. The main advantages of this technique are that the immersed boundaries are represented sharply, that there are not additional stability constrains when rigid bodies are represented and that the equations for solid and fluid cells are decoupled, leading to a lower number of equations to solve. At the same time the negatives are the difficulty to include boundary motion and the higher number of conditions that are needed on the boundaries.

After having briefly introduced the main characteristics of Immersed Boundaries methodology, it is useful to see what has been done, up to day, regarding flows with moving boundaries employing the IB technique. Many studies regarding fluid flow with moving solid-liquid interface have been carried out using an Eulerian-Lagrangian procedure, in the sense that flow region have been solved with an Eulerian approach while the interface has been tracked using Lagrangian techniques. The use of Eulerian flow's governing equation, combined with a fix mesh, largely simplify the introduction of IB methods. Moreover, an important aspect is how the effect of the interface is considered on the fluid itself. A disctintion can be done between methods that introduce a diffusion in the flow while others incorporate the IB effect sharply.

It is clear that the second technique is preferable, however an additional issue has to be overcomed in this case. When the IB moves across the fixed Cartesian grid, some cells that were previously inside the body become "fluid" while for others the opposite occours. The problem is that, when a cell that was previously solid become fluid, there are no reasonable fluid's properties values that can be given to that nodes because no time integration can be performed. These are the so called "freshly-created" cells and different solution have been developed for them. One of them is to merge the new fluid cell, just for the first time step, with the adjacents one. This approach has been shown to not affect the spatial accuracy of the numerical scheme adopted [26]. Another possibility is to use an interpolation procedure to actually recover flow variables value in these cells [25], as reported in figure 1.11. As mentioned above, the problem does not occurs when the continous forcing approach is used since the fictious force distribution acts on a band of cells around the IB therefore giving temporal continuity for cells emerging from the solid.

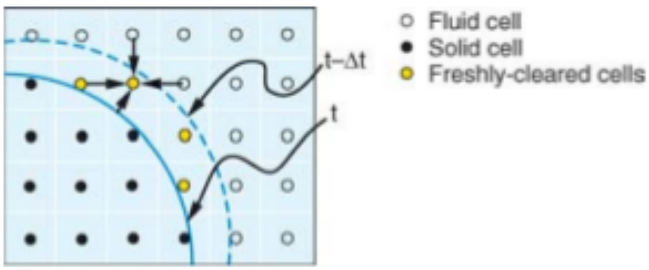


Figure 1.11: Showing of a "freshsly-created" cell after the boundary is moving and suggestion of interpolation procedure to assign its value.

In conclusion, the Immersed Boundaries technique is a powerful tool that has been developing in the last decades since the body-fitting meshes generation can become time-consuming when geometries are complicated. The ability to use a simple cartesian grid, which is not conformed to the body, basically erases the "mesh problem" and it also simplifies the handling of moving objects during the simulations. However, other challenged arises, like the tracking of the solid-fluid interface and the imposition of the bounday conditions. This work will tackling the former problem as it is going to be explained in the next section.

1.4 Contribution of the present work

The objective of the present work is to develop a coherent and comprehensive tool to handle moving interfaces (both solid-fluid or multiphase fluid interfaces) on adaptive cartesian grids in order to perform CFD simulations involving morphing bodies.

The means used for achieving this goal are the so called *Level set methods* coupled with the Immersed boundaries technique. The former sets of precedures are used, as it will deeply clarified in the next chapter, to trak moving interfaces while the latter perform the imposition of the fluid boundary conditions on the borders identified thank to the previous method. The thesis is mainly focused on the precedures involved to create and evolve through time the interfaces of the objects intented to be simulated applying the IB technique. Furthermore, the forementioned theories are then applied on adaptive cartesian meshes which have raisen a strong interest since IB's potentialities have been understood. The ability to employ the level set function on these kind of grid, combined with the application of BCs using IB, gives the opportunity to simulate quite easily and rapidly a range of physical phenomena and industrial application involving fluid flow. In fact, using a refined grid where there is an object's interface it helps to improve the accuracy of the simulation and it makes easier to track the interface during its motion. However, the strength of these techniques is that it can be applied in a wide varieties of problems related to Computer-Aided Engineering (CAE) simulations. In fact, whenever a moving objects is involved and the time availbale to gather the results is restricted, then this solutions could be employed.

The thesis will start from an accurate analysis of the theory behind the level set function, its evolution during time and the problems linked with these precedures. Then, the methodologies developed for calculate the global signed distance field will be presented in the thrid chapter while the fourth is going to explain the technique adopted to transport the level set values. In conclusion, a test case able to use the developed techniques in conjunction with the IB methodologies will be presented.

Chapter 2

Levelset Method

2.1 Signed distance function

The first step that is needed to properly define a signed distance function is to understand what is an implict interface representation. An interface is defined as the point, or collection of them, that delimits a certain region of space from another. In the monodimensional case the interface is just a single point while in two dimension is a line and in 3D is a surface. This interface could be described explicitly by all the points coordinates that belong to it or implicitly having a function which is equal to zero in all points of the interface itself so $\phi(x, y, z) = 0$.

The implicit interface representation has another advantage that consists in being able to give a signed value to points in the outer or inner part of the domain separated by the zero isocontour of the function. Using this property, it is possible to simply define which region is outside the domain delimited by the interface, having $phi > 0$, and which is the inner one, with $\phi < 0$. Therefore, there are three separated regions: Ω^+ the outer domain, $\partial\Omega$ the interface and Ω^- the internal region. A simple example in two dimensions is depicted in figure 2.1.

However, the example reported in figure 2.1 is a particular case where the interface is represented by an analitical function. Usually this does not happen and for general shape it could become hard to find a function to describe the interface. This is the reason why many times a discretization and an approximation are needed.

Often, in case of an explicitly given interface, it is convenient to parameterized the set of points that describe the interface. A common choice is to discretize the parameter into a finite set of points and their number determine the resolution of the curve.

In the implicitly interface definition the discretization it is sligtly more complicated. In fact, not only the zero isocontour needs to be approximated but all the point of the considered domain have to be discretized. However, since the interface is the main thing of interest, the only points which counts are the one close to it. Therefore, their resolution in that part of the domain is increased. The collection of points where the implicit function ϕ is defined is called grid and in the present work only Cartesian grid (also not uniform) will be used.

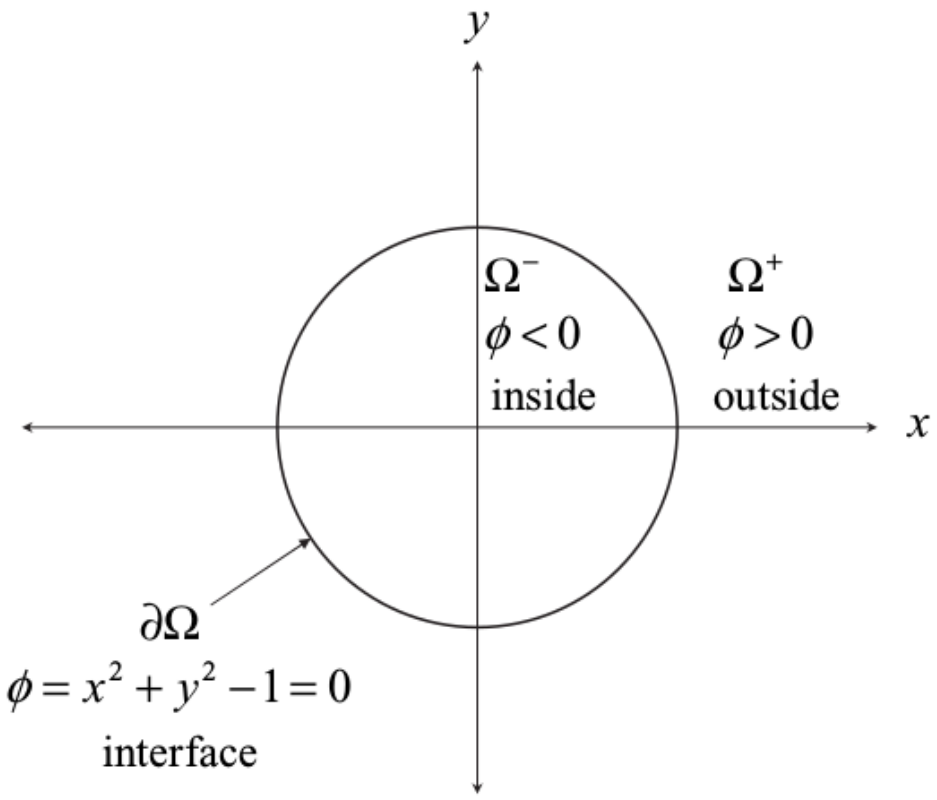


Figure 2.1: Implicit function $x^2 + y^2 - 1 = 0$ [7].

The advantages of having an implicit function for the interface greately exceed the ones of an explicit definition. In fact, this method have some powerful geometric tool availble. A main positive is to be able to simply determine whether a point is inside or outside with respect to the interface by simply looking at the corresponding value (and sign) of the ϕ. In case the point to be evaluated doesn't rely on a discretized grid point, interpolation with adjacent nodes is made. Of course, this numerical procedure could lead to wrong approximations perturbing the interface position. However, if these perturbations are small, the estimations can still give acceptable results. Clearly, augmenting the sample points (e.g. the mesh nodes) reduces the evaluation's errors.

Another gain of using implicit representation is to easier boolean operation and other complex constructing solid geometry operations [7] (for example to intersect two different implict functions, ecc...). Moreover, the derivatives and the normal to the interface can be obtained. The gradient of the implict function is defined as

$$\nabla\phi = \left(\frac{\partial\phi}{\partial x}, \frac{\partial\phi}{\partial y}, \frac{\partial\phi}{\partial z}\right) \tag{2.1}$$

and this vector is perpendicular to the isocontours of ϕ and it points in the direction of the increase of the function. Therefore, $\nabla\phi$ has the same direction of the outward normal

from the isocontours lines of the function and this normal can be defined as

$$\mathbf{N} = \frac{\nabla\phi}{|\nabla\phi|} \tag{2.2}$$

where $|\nabla\phi| = \sqrt{\phi_{,x}^2 + \phi_{,y}^2 + \phi_{,z}^2}$. Furthermore, the mean curvature of the interface, in a certain point, is defined as the divergence of the normal vector $\mathbf{N}$, therefore reading as

$$k = \nabla \cdot \mathbf{N} = \frac{\partial n_1}{\partial x} + \frac{\partial n_2}{\partial y} + \frac{\partial n_3}{\partial z} = \nabla \cdot \left(\frac{\nabla\phi}{|\nabla\phi|} \right) \tag{2.3}$$

where n_1, n_2 and n_3 are the normal components. The $k > 0$ indicates that the there is a convex interface region, the $k < 0$ a concave region and $k = 0$ a plane (see figure 2.2).

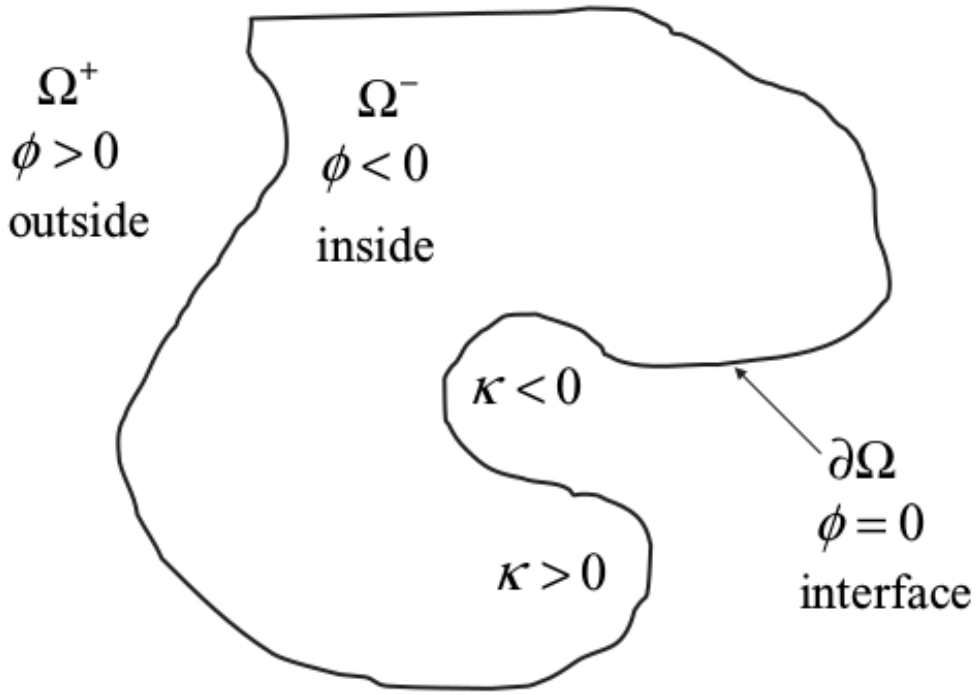


Figure 2.2: Convex and concave regions on a generic two dimensional figure.

An example of the normal vector $\mathbf{N}$ for the previous implicit function of figure 2.1 is depicted in figure 2.3.

After having introduced the implicit functions to describe an interface, it is possible to define the *distance functions*. These latter functions have a further condition which is the following

$$|\nabla\phi| = 1 \tag{2.4}$$

this means that if the function is measured in a certain point, its growth is equal to the minimum distance from the point to the interface.

The distance function is defined as

$$d(\mathbf{x}) = min(|\mathbf{x} - \mathbf{x}_I|) \tag{2.5}$$

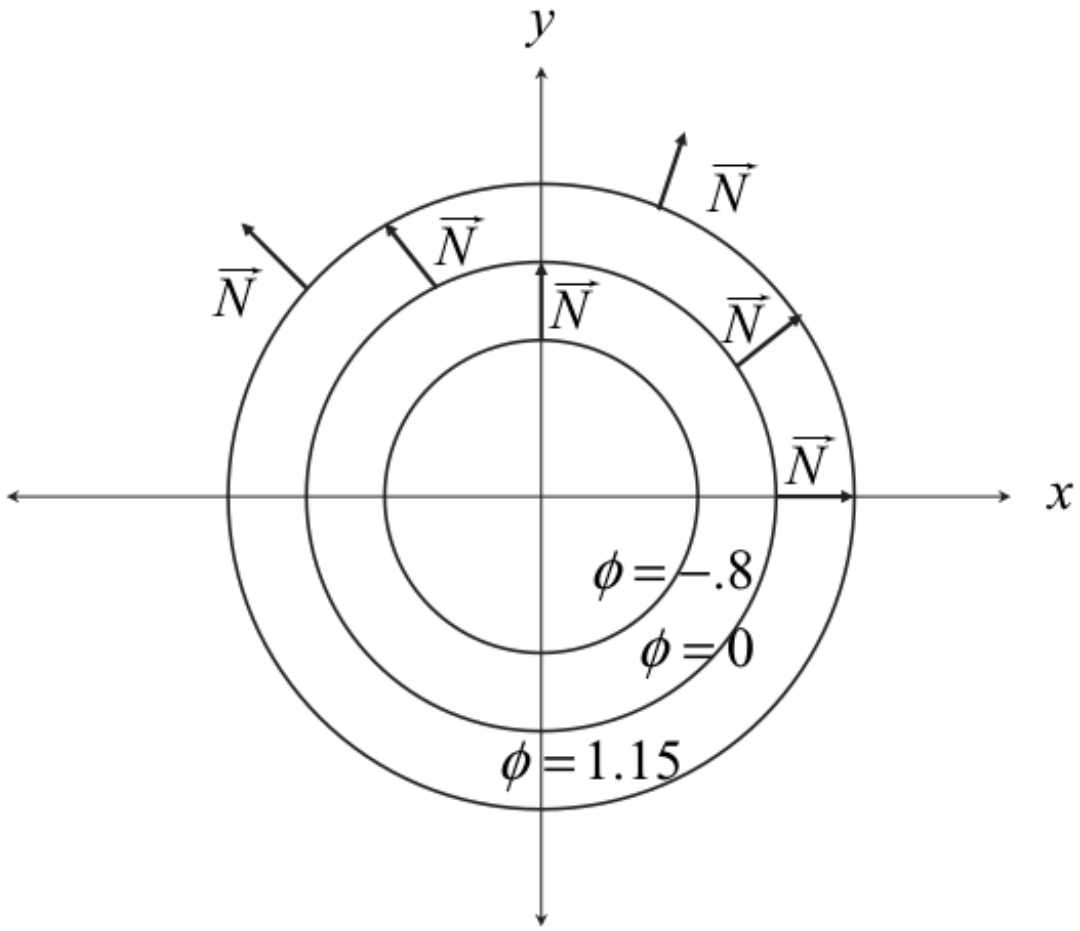


Figure 2.3: Normal vector at different points of some isocontours.

where $\mathbf{x}_I$ are the points which belong to the interface and $\mathbf{x}$ is a generic point of the domain. From the definition of points $\mathbf{x}_I$ it is obvious that $d(\mathbf{x}) = 0$ on points of $\partial\Omega$.

Then, using distance function, one can finally get the definition of a *signed distance function* which reads

$$|\phi(\mathbf{x})| = d(\mathbf{x}) \tag{2.6}$$

for all the point of the domain $\mathbf{x}$. Therefore, this leads to

$$\begin{cases} \phi(\mathbf{x}) = -d(\mathbf{x}) & \text{over} \quad \Omega^- \\ \phi(\mathbf{x}) = 0 & \text{over} \quad \partial\Omega \\ \phi(\mathbf{x}) = d(\mathbf{x}) & \text{over} \quad \Omega^+ \end{cases}$$

This new set of functions shares all the properties of the *distance functions*, also the one expressed in equation 2.4. However, this property is true for points that are equidistant

only from one point of the interface. When there are two or more points of zero isocontour that have the same distance from a point of the domain, that property doesn't hold anymore. This is problematic when the derivatives of the signed distance function are approximated by a discretized technique. On the other hand, being monotonic across the interface makes the function ϕ differentiable with a high conficence.

Furthermore, in the worst case, the level set function ϕ belongs to class C^0. In fact, the signed distance function is always continous and differentiable but its first derivatives could show discontinuities in their values, therefore not being differentiable.

The level set methods only add dynamics to signed distance functions; therefore, an implicit interface becomes a moving object which changes through the time. This first technique was introduced by Osher and Sethian in [17]. In the next section the so called "level set equation", which is the evolving equation of the procedure, will be explained in details.

2.2 Transport of level set function

Transport the level set function means to correctly evolve through time the interface of interest. Supposing to know, at each temporal step, the velocity $\mathbf{V} = (u, v, w)$ at which every points on the interface moves, the simplest way to track the position of the ϕ zero-isocontour is to solve the ordinary differential equation

$$\frac{d\mathbf{x}}{dt} = \mathbf{V}(\mathbf{x}) \tag{2.7}$$

for all interface's points. This represents the *Lagrangian* approach for the interface evolution. Since the points that belong to the surface are infinite, a discretization of the front into a finite number of segments (in 2D) or triangle (3D) is needed. Then the end-points of the discretized interface are moved following the previous law. However, large distorsion of boundary elements is likely to occur, leading to a quickly deterioration of the results. This problem can be avoided implementing a periodic modification of the interface's discretization in order to mantain it smooth and regular, but the choice of the new discretization is not trivial, thus resulting difficult to implement.

Therefore, the alternative to avoid such problems is to use the implicit function ϕ not only to represent the interface but also to evolve it through time. This approach, of evolving the whole signed distance function field through an equation is called *Eulerian* view and the law used is the so called transport equation. Thus, the level set equation is an Eulerian transport equation of the scalar field ϕ which, integrated through time, gives the motion of the surface. The equation reads

$$\frac{\partial \phi}{\partial t} + \mathbf{V} \cdot \nabla \phi = 0 \tag{2.8}$$

where the first term is partial derivatives of the implicit function in time, while the second term reads

$$\mathbf{V} \cdot \nabla \phi = u\frac{\partial \phi}{\partial x} + v\frac{\partial \phi}{\partial y} + w\frac{\partial \phi}{\partial z}$$

As already mentioned, the previous equation is an Eulerian formulation of a transport equation, this means that ϕ is evolving over all the discretized domain and the interface is numerically captured by the function itself. Since it could be complicated to have a velocity just defined over the interface, what is done is to have a complete field of velocity in every nodes of the grid. However, for computational purposes, it could be sufficient to just define a speed in a band of nodes around the zero isocontour of the implicit function.

2.3 Reinitialization of level set function

Level set function can develop steep gradient and, due to numerical viscosity, it can drift away. For this reason, the *reinitialization* technique was developed. It was first introduced by Chopp in [4] where, to adjust level set through calculation, he proposed to periodically reinitialize the function in order to always deal with a signed distance function.

Reinitialization is a powerful tool since fixes the error the can be developed from the evolution of the level set function. This is a great advantage with respect to classical approach where the behavior of the numerical solution is just assumed, from the initial data, to remain good through the time steps. In this case, the hypothesis needed is that only the $\phi = 0$ isocontour remains well behaved. However, the computational cost of the procedure is quite high. Therefore, in order to get an acceptable run time, it was proposed a modification of the technique able to correct the implicit function values only in nodes adjacent to the interface itself. In fact, when zero isocontour is repaired, then the other points of the domain can be restored to be a signed distance function using an effcient propagation technique.

One of the first attempt to create an efficient method to implement reinitialization of the level set function was made by Sussman *et. al.* [23] with the introduction of the reinitialization equation, which reads

$$\frac{\partial \phi}{\partial t} + S\left(\phi_0\right)\left(\left|\nabla \phi_0\right| - 1\right) \tag{2.9}$$

where ϕ_0 denotes the initial value of implicit function, while the term $S\left(\phi_0\right)$ is a sign function equal to ± 1, respectively, in Ω^+ and Ω^- and 0 in $\partial\Omega$. The final result of this equation is ϕ field which corresponds to the desired signed distance function. Since it is an hyperbolic problem, only boundary condition at interface are needed. For this reason, a certain number of cells near the zero isocontour are initialized with the exact value of ϕ and then used as Dirichlet boundary conditions. In particular, the points in Ω^- use values defined in Ω^+ as BCs and vice versa. Due to problems related with the boundary conditions, some errors in the motion of interface during this procedure could be detected. Therefore, some identification and correction method were developed to resolve these conditions. Furthermore, the hyperbolic term $S\left(\phi_0\right)$ is crucial to get correct result. Using numerical tests, Sussman [23] modelled this part of the equation as

$$S\left(\phi_0\right) = \frac{\phi_0}{\sqrt{\phi_0^2 + \left(\Delta x\right)^2}}$$

Also other way to get $S(\phi_0)$ were proposed. However, the previous one has the advantage to be constant through the iterations.

Chapter 3

Levelset propagation methods

3.1 Introduction to propagation of levelset function

Level set propagation method is an essential technique used for create a complete signed distance field, referred to a body, starting from the ϕ values close to the body's borders.

This technique is often used together with the reinitialization one. In fact, with the latter the zero isocontour is "repaired" and the body shape is maintained; while with propagation the values of distance from the body's interface are restored in all the computational domain.

3.1.1 The need of propagate the levelset function

Levelset function is a strong tool when the problem has an interface that evolves over time. This interface could be between two phase flow, body and fluid or whatever. In particular, using equation 2.7 the desired body's movement can be achieved by integration in time of the distance field ϕ. This latter is a scalar field and it represents the interfaces or geometries changing in time inside the domain.

Having to solve a transport equation means to handle a partial differential equation applyed in all the discrete nodes of the mesh. Therefore, the signed distance function field is needed in all the cells and not just close to the interface.

3.1.2 Traditional approach and possible new solutions

In the classical fashion, the problem of calculating the distance of a point from a body is governed by the so called *Eikonal equation* which reads

$$\nabla\phi \cdot \nabla\phi = \frac{1}{c^2} \tag{3.1}$$

where ϕ represent the unknown signed distance function and c is the characteristic velocity of the "distance fronts" and for this case is equal to 1. This is an hyperbolic PDE and the traditional approach takes advantage of this feature. In fact, the so called *Fast marching method* starts from the exact distance values of the cells adjacent to the zero isocontour and it marches outward assigning to each node the right ϕ.

In order to correctly stream information from this band around the body, it is necessary to decide which cell (not initialized) needs to be updated first. Clearly, it has to be the one that would have the smallest distance because it is the closer to the body, thus the characteristics lines reach it faster. So, all the grid points close to the band are updated using a tentive value of signed distance function. The point which has the lower tentative value is accepted and included in the band. Then the process is repeated and, in case the tentative value was not good enough, it can be modified and improved with the newly available information. When all the cell values are updated, the algorithm is concluded and the signed distance function is known at each point.

The procedure for obtaining the distance field inside the body is exactly the same, recovering also a positive distance. Then, the ϕ found internally is changed of sign.

The slowest part of the algorithm is to search through all tentative grid points to find the one with lower value. However, since this method was introduced by Tsitsiklis in [24] and [20], many modification to the original procedure have been done and the convergence speed have been largely increased. The problem that is suffered from this method relies in its intrinsic serial nature. It is difficult to parallelize even more when a good scalability is desired on a large number of processor. Therefore, even with the best algorithm it will always remains a bottle neck during a parallel procedure.

This is the the reason why a new parallelizable procedure has been researched in the present work. In the next sections two possibilities will be explained in details and both are easily implemented and could be used, with little effort, on multiple processors in order to increase their convergence velocity.

3.2 Heat method for geodesic distance

The first of the two techniques is the *heat method*. As the name suggests it is based on heat and its properties that enable to draw a direct link with the signed distance function.

In fact, takig advantage of the way the heat spreads out from a source, it is possible to construct a function, called *heat kernel*, which measures the quantity of heat transferred from the source, positioned in x, to a generic point y after time t. Moreover, this function can be simply transformed to establish a clear relationship between heat at y and its distance from x. This formula is known as *Varadhan's formula* [27] and it can be stated as:

$$\phi(x, y) = \lim_{t \to 0} \sqrt{-4t \log k_{t,x}(y)} \tag{3.2}$$

where ϕ is the distance, t is the time and $k_{t,x}(y)$ is the heat kernel function. This approach finds its physical base on the fact that heat diffusion can be seen as a great ensamble of high temperature particle taking random walks starting from point x. Therefore, every particle that reached the destination point at the time of the measure had few possibilities to not undertake the shortest path.

However, this tecnique has not been exploited until now, likely because it is difficult to exactly recover the right heat kernel function and using approximations do not give correct results. The solution proposed by this new methodology is to use function that

have the gradient parallel to the geodesic and later use this information to recover the right distance [5]. In comparison with the existing algorithm, this approch offers two main advanteges: first, it can be used on every kind of grid; second, it is based on the solution of linear sparse systems of equations which is a rapid operation.

In this section this procedure will be explored, starting from its physical fundations to the possible implementations and, at the end, its results.

3.2.1 From Eikonal to Heat equation

Using the Eikonal equation 3.1 it is possible to derive the heat equation on which the current method is based.

First, with an arbitrary change of variables the temperature field and the distance field can be related as follows

$$T = \exp^{-\frac{\phi}{k}} \tag{3.3}$$

where T is the temperature, ϕ the distance in each point of the domain and k a scaling parameter that is going to be described later.

Using the definition in equation 3.3 it is possible to apply gradient operator to this expression getting

$$\nabla T = -\frac{1}{k}\nabla\phi \exp^{-\frac{\phi}{k}} = -\frac{T}{k}\nabla\phi$$

then applying the nabla operator to the previous equation, it becomes

$$\triangle T = -\left(\frac{1}{k}\nabla T \cdot \nabla\phi + \frac{T}{k}\triangle\phi\right)$$

and substituting the definition of ∇T

$$\triangle T = -\left[\frac{1}{k}\left(-\frac{T}{k}\nabla\phi\right)\nabla\phi + \frac{T}{k}\triangle\phi\right]$$

finally, using the Eikonal equation, the seeked relationship reads

$$\frac{T}{k^2c^2} - \frac{T}{k}\triangle\phi - \triangle T = 0 \tag{3.4}$$

with the term $\frac{T}{k}\triangle\phi$ involving the second order derivatives of the distance function, thus being responsible for the correction of the curvature's solution.

The equation 3.4 is analogous to the unsteady heat equation which reads

$$\frac{\partial T}{\partial t} - \triangle T = 0 \tag{3.5}$$

under the following conditions:

$$k \to 0 \quad , \quad k = \sqrt{dt} \quad , \quad c^2 = 1$$

In fact, when k is low, the curvature term is negligible in comparison with the others and, if initial temperature is considered to be zero in the domain, the temporal derivation term reads

$$\frac{\partial T}{\partial t} \approx \frac{T^{n+1} - T^n}{dt} = \frac{T}{dt}$$

with $T = T^{n+1}$ as the temperature at the next iteration and the initial $T^n = 0$. Therefore, under the forementioned conditions, equation 3.4 becomes

$$\frac{T}{dt} - \triangle T = 0$$

which is the same as 3.5.

3.2.2 Heat method algorithm

After having exploited the relationship between temperature and distance, the procedure used to recover the ϕ field can be explained. Looking at [5], the heat method can be described by a three steps algorithm:

1. Integrate the heat flow over a fix time t;
2. Evaluate the direction of the vector field;
3. Solve a Poisson problem over the domain;

The first step it is done to obtain the right gradient direction and not the correct temperature field. For this reason, a good choice for the boundary condition is to use an homogenous Dirichlet equal to 0, thus implying an infite distance. The equation to integrate is the unsteady heat equation, which reads:

$$\frac{\partial T}{\partial t} - \triangle T = 0 \tag{3.6}$$

where T is the temperature. An important point to underline is the initial condition to impose for the temperature field. In fact, an initial temperature needs to be applyed in order to diffuse the heat in the rest of the field through the time integration. This quantity can be arbitrary and it could be the generalized Dirac function δ_0 over the entire submanifold from which we are interested to calculate the distance. Furthermore, the time step choice is critical for the accuracy of the method itself. However, as suggested from [5], a simple approximation that works increadibily well in practice is to use $t = h^2$ with h equal to the average spacing between cells. In a uniform grid, the average h is simply the real distance between cells (which is constant). However, in a not uniform grid, this parameter changes between cells and the choice of the time step becomes ambiguous if the average spacing is not employed.

After having obtained this temperature intermediate field, the normalization of its gradient, which is equal to 1 at stationariness, is done by applying the following formula in each point of the domain

$$X = -\frac{\nabla T}{|\nabla T|} \tag{3.7}$$

with $|\nabla T|$ equal to the norm of the gradient.

At the end, in order to recover the right distance function, it is necessary to solve the Poisson problem

$$\triangle\phi = \nabla \cdot X \tag{3.8}$$

having ϕ as distance field. This problem is equivalent to find the distance function that minimize $\int_M |\nabla\phi - X|^2$. One thing to note is that the solution to this step of the algorithm is unique only up to an additive constat.

The procedure described above is slightly different from the one employed in the present work. In fact, the equation integrated in the first step is the 3.4 which also includes the curvature term that was obtained during the derivation of the Heat equation using the Eikonal one.

3.2.3 Discretization

The discretizing schemes employed to implement the proposed methodology are two: a finite difference approximation and a finite volume one. In this section they are going to be explained in details.

Finite difference approach

In this approach a finite different approximation scheme was adopted on a vertex centered grid. Time integration was coded using a simple forward Euler discretization. The domain used was a square with regular spacing between grids point and with $\Delta x = \Delta y = h$ having the indices going as depicted in figure 4.2.

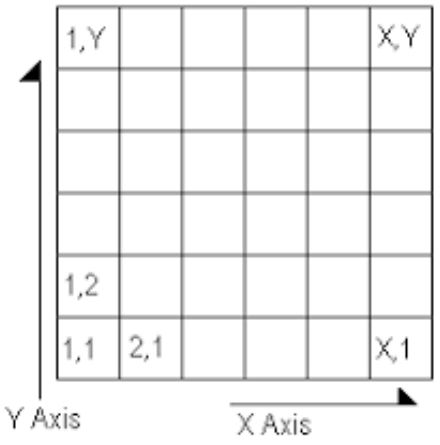


Figure 3.1: Vertex centered grid employed.

Therefore, the discretized equation solved in the first step of the method reads

$$\frac{T^{n+1}}{dt} - \triangle T^{n+1} - \frac{T^{n+1}}{\sqrt{dt}}\triangle\phi^n = \frac{T^n}{dt}$$

however, as it was said before, the temperature at initial iteration is $T^n = 0$ leading to

$$\frac{T^{n+1}}{dt} - \triangle T^{n+1} - \frac{T^{n+1}}{\sqrt{dt}} \triangle \phi^n = 0 \tag{3.9}$$

where the term $\frac{T^{n+1}}{dt}$ is a matrix having just the main diagonal different from zero and equal to $\frac{1}{dt}$, while $\triangle T^{n+1}$ is the laplacian discretized matrix and $\frac{T^{n+1}}{\sqrt{dt}} \triangle \phi^n$ is a matrix with the non zeros entries on the main diagonal equal to $\frac{\triangle \phi^n}{\sqrt{dt}}$.

The most complex part to discretize is the laplacian term, which can be written as follows

$$\triangle \phi = \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2}$$

and discretizing the derivatives operators, in a internal point of the domain, it can be written as

$$\triangle \phi \approx \frac{1}{h^2} \left(\phi_{i-1,j} + \phi_{i,j-1} - 4\phi_{i,j} + \phi_{i,j+1} + \phi_{i+1,j} \right)$$

Finite volume approach

In addition to the finite difference scheme, also a cell centered finite volume method was applied for the spacial discretization. In this case the opensource library *Bitpit* was used to handle the mesh, while the opensource library *PETSc* was used to save and solve sparse system of equations. In this case, the laplacian term reads

$$\triangle \phi = \frac{1}{V} \int_V \triangle \phi \, dV = \frac{1}{V} \int_V \nabla \cdot (\nabla \phi) \, dV$$

then using Gauss theorem, the discretized form becomes

$$\frac{1}{V} \int_V \nabla \cdot (\nabla \phi) \, dV = \frac{1}{V} \int_{\partial V} (\nabla \phi) \cdot \mathbf{n} \, dV \approx \frac{1}{V_c} \sum_{f=0}^{N_f - 1} \left[(\nabla \phi)_f \cdot \mathbf{n}_f A_f \right]$$

where the subscript f is indicating the face of the cell, $\mathbf{n}$ is the vector of the normal of the face (which is point outwards from the cell), V_c is the cell's volume and A_f is the area of the cell's face.

Modified equation

The previous schemes were applyed on Cartesian grids, both uniform and not-uniform. In this latter case, a quadtree mesh (in 2D) and an octree grid (in 3D) were employed. However, the algebric equation derived from the discretization of the original PDE, namely 3.5, could be different when the grid presents irregularities. To verify that the equation solved by the numerical scheme corresponds to the initial differential law, the *Modified equation analysis* is performed.

A modified equation is a model differential equation that is solved by a difference equation [1]. In numerical method, an algebric approximation of a PDE is derived in

order to get an accurate estimation of its solution. However, there are multiple partial differential equation for each difference equation. Therefore, in order to understand which is the real differential equation that is solved by the numerical procedure adopted, it is important to derive the related modified equation.

In particular, the derivation was done for the initial integration problem of the heat method (governed by equation 3.6). The analysis was carried out only for one dimensional case in order to avoid complicated algebric passages. In case of the cell centered finite volume approach, the 1D algebric equation that was implemented, for uniform grid (figure 3.2), in each cell is the following

$$\frac{T_j^{n+1} - T_j^n}{\Delta t} - \frac{1}{V}\left[\frac{T_{j+1}^{n+1} - T_j^{n+1}}{\Delta h} + \frac{T_{j-1}^{n+1} - T_j^{n+1}}{\Delta h}\right] = 0$$

where Δt is the discrete time step, Δh is the cell dimension, V is the volume (in 1D equal to Δh), subscript indicates the cell index while the superscripts indicate the time step at which terms are evaluated.

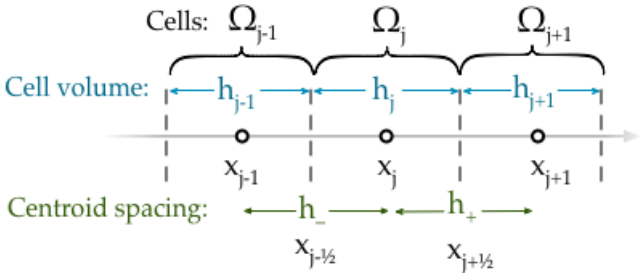


Figure 3.2: One dimensional uniform grid.

In order to get the modified equation, the different terms need to be expanded with Taylor's series. In particular the spatial expansion of T_{j+1} and T_{j-1} reads

$$T_{j+1} \approx T_j + \Delta h\, T_{,x} + \frac{\Delta h^2}{2} T_{,xx} + \frac{\Delta h^3}{6} T_{,xxx} + O\left(\Delta h^4\right)$$
$$T_{j-1} \approx T_j - \Delta h\, T_{,x} + \frac{\Delta h^2}{2} T_{,xx} - \frac{\Delta h^3}{6} T_{,xxx} + O\left(\Delta h^4\right)$$

while the one for the temporal term is the following

$$T^{k+1} \approx T^k + \Delta t\, T_{,t} + \frac{\Delta t^2}{2} T_{,tt} + O\left(\Delta h^3\right)$$

Substituting these expressions in the initial equation it leads to this final modified equation

$$T_{,t} - T_{,xx} = \frac{\Delta h^3}{12} T_{,xxx} - \frac{\Delta t}{2} T_{,tt}$$

which is the real equation solved in each node of the grid. As it can be seen, some terms appear at right hand side. They are due to the numerical error of the discretization.

However, when the Δ used (both in space and time) become infinitesimaly small then the initial PDE is recovered. This make the numerical scheme consistent.

Now, recalling figure 3.2, when the grid is not uniform things gets more complicated. In fact, using the same equation but having $h_{j-1} = \Delta h$ and $h_j = h_{j+1} = \Delta H$ with $\Delta H = 2\Delta h$, this leads to a discretized equation which reads

$$\frac{T_j^{n+1} - T_j^n}{\Delta t} - \frac{1}{V}\left[\frac{T_{j+1}^{n+1} - T_j^{n+1}}{\Delta H} + \frac{T_{j-1}^{n+1} - T_j^{n+1}}{\frac{1}{2}(\Delta h + \Delta H)}\right] = 0$$

while the Taylor's expansions of different terms can be written as

$$\begin{aligned}
T_{j+1} &\approx T_j + \Delta H\, T_{,x} + \frac{\Delta H^2}{2} T_{,xx} + \frac{\Delta H^3}{6} T_{,xxx} + O\left(\Delta h^4\right) \\
T_{j-1} &\approx T_j - \frac{1}{2}(\Delta H + \Delta h)\, T_{,x} + \frac{\frac{1}{4}(\Delta H + \Delta h)^2}{2} T_{,xx} - \frac{\frac{1}{8}(\Delta H + \Delta h)^3}{6} T_{,xxx} + O\left(\Delta h^4\right) \\
T^{k+1} &\approx T^k + \Delta t\, T_{,t} + \frac{\Delta t^2}{2} T_{,tt} + O\left(\Delta h^3\right)
\end{aligned}$$

Substituting in the difference equation and perfoming the calculation, a final expression of the following form can be obtained

$$T_t - \frac{3}{4} T_{,xx} - \frac{\Delta h}{4\Delta H} T_{,xx} = -\frac{3\Delta H}{24} T_{,xxx} + \frac{\Delta h^2 + 2\Delta h \Delta H}{24} T_{,xxx} - \frac{\Delta t}{2} T_{,tt}$$

In an analogous way it can be derived the modified equation when $h_{j-1} = h_j = \Delta h$ and $h_{j+1} = \Delta H$ and it reads

$$T_t - \frac{3}{4} T_{,xx} - \frac{\Delta H}{4\Delta h} T_{,xx} = -\frac{3\Delta h}{24} T_{,xxx} + \frac{\Delta H^2 + 2\Delta h \Delta H}{24} T_{,xxx} - \frac{\Delta t}{2} T_{,tt}$$

These two equations implies that, when a non-uniform grid si employed, the first integration problem becomes non consistent since the term $\frac{\Delta H}{4\Delta h} T_{,xx}$ doesn't tend to zero when grid spacing tends towards zero.

However, in this case the calculation of the gradient at cell's interface was performed assuming a linear distribution of the values of the temperature and simply subtracting the two values stored in the center of the cell. A more accurate approximation can be done if a second order function is used to approximate the value of the variable inside a cell itself. Considering a big cell close to a small one (as in the two cases explained above) it is possible to reconstruct the value of the temperature inside the big cell in a point specular to the center of small cell.

Calling this point with subscript $j + \frac{1}{2}$ (figure 3.3) and having $h_{j-1} = h_j = \Delta H$ and $h_{j+1} = \Delta h$ the algebric equation to solve becomes

$$\frac{T_j^{n+1} - T_j^n}{\Delta t} - \frac{1}{V}\left[\frac{T_{j-1}^{n+1} - T_j^{n+1}}{\Delta H} + \frac{T_{j+1}^{n+1} - T_{j+\frac{1}{2}}^{n+1}}{\Delta h}\right] = 0$$

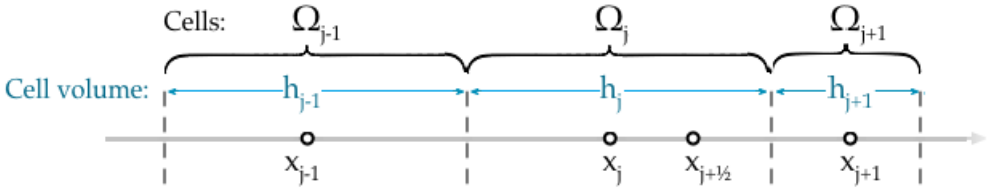


Figure 3.3: One dimensional non-uniform grid.

Using a quadratic interpolant polynomial (a parabola or a lagrange polynomial) the value of $T_{j+\frac{1}{2}}$ can be expressed using the values of T_j, T_{j+1} and T_{j-1} as

$$T_{j+\frac{1}{2}} = \frac{5}{6}T_j + \frac{5}{21}T_{j+1} - \frac{1}{14}T_{j-1}$$

and then, expanding and substituting the final expression of modified equation reads

$$T_{,t} - T_{,xx} = -\frac{\Delta t}{2}T_{,tt} - \frac{\Delta h}{6}T_{,xxx}$$

which is a consistent equation since when the temporal and spatial spacing go to zero the initial PDE is recovered and error tends to zero. Therefore, when dealing with not uniform grids, it is convenient to employ a second order polynomial to reconstruct the temperature for calculating the gradient at cell's interfaces.

3.2.4 Boundary conditions

The critical point of this methodology is the application of the boundary conditions, the dilemma arises when the Eikonal equation is used to derive the Heat equation. In fact, even if they can be related, their nature is radically different. The distance calculation problem is hyperbolic, therefore starting from a boundary condition, in this case applied on body's interface, it propagates this information in the rest of the domain marching outwards from it and no further impositions have to be done. For example, no BCs are required at the border of the mesh. On the other hand, the heat propagation is an helliptic problem which requires the boundary conditions both on the interface and on the outer grid's borders. In this case the information do not travel just in one direction but there is an interaction of the borders with interior domain's region. Therefore, it has been conducted a deep study regarding the boundary conditions that can be applyed to the helliptic problem solved, in order to not alter the final distance function.

As mentioned before, for the integration problem, only Dirichlet BCs were applyed. The true obstacle is to find a good solution for the BCs in the Poisson problem. Two possible alternatives were implemented to solve this issue:

- applying Neumann BCs having as a flux the normalized gradient of the temperature field;
- applying Dirichlet BCs and using a *BFGS* optimization method toghether with the *Discrete Adjoint* gradient's calculation to minimize the functional of the Eikonal equation.

These two approaches will be explained in details in the next paragraphs.

Neumann BCs

The former strategy is to use Neumann BCs with a border's flux given by

$$\frac{\partial \phi}{\partial n} = X_n$$

where X_n is the perpendicular component of the normalized temperature gradient at boudaries, obtained during the second step of the heat algorithm using 3.7. In the case of vertex-centered finite difference scheme, it was seen that the best way to achieve the right gradient of the solution at boundary is to use a ghost points method. The equations on a random ghost point of the west boundary of the domain reads

$$\begin{cases} \triangle\phi = \frac{1}{h^2}\left(\phi_{i-1,j} + \phi_{i,j-1} - 4\phi_{i,j} + \phi_{i+1,j} + \phi_{i,j+1}\right) \\ \\ \frac{\phi_{i+1,j} - \phi_{i-1,j}}{2h} = X_x \end{cases}$$

where X_x is the x component of the normalized temperature gradient (which is the one normal to the domain's border), h is the spacing of the grid (uniform in this case) and $\phi_{i,j}$ is the distance at point i, j of the grid. Since the equation is referred to a border's node, the $\phi_{i-1,j}$ value is not know and its a ghost point. Therefore, the final equation to be applyed to the boundary is the following

$$\frac{1}{h^2}\left(\phi_{i,j-1} - 4\phi_{i,j} + \phi_{i,j+1} + \phi_{i+1,j}\right) = \nabla \cdot X_{i,j} + \frac{2}{h} X_{x_{i,j}}$$

where $\nabla \cdot X_{i,j}$ is the right hand side of the Poisson problem and the term $\frac{2}{h} X_{x_{i,j}}$ comes from the calculation, and then substitution, of the unknown term $\phi_{i-1,j}$ in the discretized laplacian equation.

In the cell-centered finite volume method, since the borders of the domain coincide with a cell's interface, there are no stored values on them. Therefore, the normalized gradient is not available there and it needs to be extrapolated using a second order Lagrange polynomial. Then it is applyed as Neumann boundary condition for the final Poisson's problem. The normal component of the gradient at border cell's faces is known since it is the one extrapolated, thus it goes to the right hand side of the cell's equation.

Dirichlet BCs

The latter strategy consists in applying Dirichlet BCs to the Poisson problem in the heat algorithm. This approach was only developed for the vertex-centered finite difference discretization. In this case, the *Discrete Adjoint method*, combined with a *BFGS approach*, was used to optimize the value of the distance to impose on the border's domain.

The adjoint method is a way to obtain the gradient of a function, with respect to the choosen parameters, when the number of variables is large. The BFGS methods is a quasi-Newton optimization algorithm that uses the gradient of the function to evaluate the

optimization's variables in order to find the minimum of the function itself. Therefore, it is clear how the two techniques are combined together: the gradient obtained by the Adjoint method is used by the BFGS approach. In this case, the functional that is needed to be minimize is the Eikonal equation and when its optimum has been found the procedure stops.

Let deepen the procedure used for calculate the gradient with the Discrete adjoint method. In this case, the problem reads

$$\begin{cases} \triangle\phi - f = 0 & \text{on} \quad \Gamma \\ \\ \phi - \gamma = 0 & \text{on} \quad \Omega \end{cases} \tag{3.10}$$

where γ is the parameter used to minimize the functional I (the Dirichlet BC to be imposed at boundaries nodes), Γ is the internal domain and Ω represents its border. The functional, which coincides with the Eikonal equation, reads

$$I(\phi, \gamma) = \int_{\Omega} (\nabla\phi \cdot \nabla\phi - 1)\, d\Omega$$

and its gradient with respect to γ is

$$\frac{dI}{d\gamma} = \frac{\partial I}{\partial \gamma} + \frac{\partial I}{\partial \phi}\frac{\partial \phi}{\partial \gamma}$$

The desired condition could be written as

$$\frac{\partial I}{\partial \gamma} = 0 \tag{3.11}$$

that is the condition that assures the minimization of the functional.

Starting from the problem stated in 3.10, it is possible to calculate the residual of this equation as

$$R(\phi, \gamma) = M\phi - b$$

where M represents the discretized laplacian matrix. The residual becomes equal to zero when ϕ and γ are the solution of the system. Perturbing the initial ϕ_0 and γ_0 it is possible to get

$$\begin{cases} \phi = \phi_0 + d\phi \\ \\ \gamma = \gamma_0 + d\gamma \end{cases}$$

and the new residual to these new solutions can be expanded using a Taylor's series as

$$R(\phi, \gamma) = R(\phi_0, \gamma_0) + \frac{\partial R}{\partial \phi} d\phi + \frac{\partial R}{\partial \gamma} d\gamma + O\left(d^2\right)$$

where $R(\phi_0, \gamma_0) = 0$ because ϕ_0 and γ_0 are solutions of the problem 3.10. Neglecting higher order terms and supposing that the new ϕ and γ are solution to the problem, it could be written that

$$\frac{\partial R}{\partial \phi} d\phi = -\frac{\partial R}{\partial \gamma} d\gamma \quad \Rightarrow \quad \frac{\partial R}{\partial \phi}\frac{d\phi}{d\gamma} = -\frac{\partial R}{\partial \gamma} \tag{3.12}$$

Now, knowing that

$$R = M\phi - b \quad \Rightarrow \quad \frac{\partial R}{\partial \phi} = M$$

the equation 3.12 becomes

$$M\frac{du}{d\gamma} = -\frac{\partial R}{\partial \gamma}$$

which leads to

$$\frac{du}{d\gamma} - M^{-1}\frac{\partial R}{\partial \gamma}$$

and substituting this result in the optimum condition (eq. 3.11)

$$\frac{dI}{d\gamma} = \frac{\partial I}{\partial \gamma} + \frac{\partial I}{\partial \phi}\frac{\partial \phi}{\partial \gamma} = \frac{\partial I}{\partial \gamma} + \frac{\partial I}{\partial \phi}\left(-M^{-1}\frac{\partial R}{\partial \gamma}\right) = 0$$

Then, calling $\frac{\partial I}{\partial \phi}M^{-1} = \lambda^T$ the adjoint equation reads

$$\left(\frac{\partial I}{\partial \phi}\right)^T = M^T\lambda \tag{3.13}$$

and the final optimum condition to be solved in order to calculate the functional gradient with respect to γ parameter is

$$\frac{\partial I}{\partial \gamma} = \lambda^T\frac{\partial R}{\partial \gamma} \tag{3.14}$$

Therefore, knowing the matrix $\frac{\partial I}{\partial \phi}$, the adjoint equation can be solve, getting λ. Then, having the matrix $\frac{\partial R}{\partial \gamma}$ it is possible to get the variation of the functional with respect to the parameter γ.

After having calculated this gradient, the BFGS approach was used to determine γ and consequently the ϕ to be imposed as Dirichlet BCs in the Poisson's problem.

3.2.5 Fixed-point iterations with Dirichlet-Neumann update

The first implemented solution to solve the BCs dilemma for the elliptic Poisson problem is to apply a Neumann BC equal to the normalized gradient at mesh borders. Moreover, an iterative fixed-point procedure, depicted in figure 3.4, was created to optimize the distance field obtained. In particular, the objective of the iterations is to minimize the functional, represented by the Eikonal equation, until it converges to a low value.

First of all, the functional was defined as follows

$$I = \frac{1}{2}\left[(\nabla\phi)^2 - \mathbf{e}\right]^T\left[(\nabla\phi)^2 - \mathbf{e}\right] \tag{3.15}$$

where $\mathbf{e}$ is a vector of the same dimension of the grid points' number and the superscript T indicates the transposition of the vector. Then, defining the term $(\nabla\phi)^2$ as

$$(\nabla\phi)^2 = \mathbf{g} = \begin{pmatrix} \nabla\phi_1 \cdot \nabla\phi_1 \\ . \\ . \\ . \\ \nabla\phi_n \cdot \nabla\phi_n \end{pmatrix}$$

where $\nabla\phi_n$ is the gradient of the distance calculated at the n-th grid's point. After that, it is possible to obtain

$$I = \frac{1}{2}\left(\mathbf{g}-\mathbf{e}\right)^T\left(\mathbf{g}-\mathbf{e}\right) = \frac{1}{2}\left(\mathbf{g}^T\mathbf{g} - 2\mathbf{e}^T\mathbf{g} + \mathbf{e}^T\mathbf{e}\right) = I\left(\mathbf{g}\left(\phi\right)\right) \tag{3.16}$$

After that a complete Heat method is done, the functional of the distance field can be calculated. Then the iterative algorithm imposes to use the boudary ϕ values (converted into temperature) as Dirichlet BCs for the first integration problem (equation 3.4). In addiction, the temperature values imposed are relaxed using a parameter α.The conversion from distance to temperature is done using the inverse relation of 3.3,which reads

$$\phi = -k\ln\left(T\right) \tag{3.17}$$

For the first iteration, since there no availble value of temperature at boundaries, the values are set to zero, which correspond to apply an infinite distance from the body itself.

Particular attention need to be taken when the distance field interior to the body is calculated. In fact, using the cells in the narrowband as border with Dirichlet BC, the distance value is propagated also inside the body. However, if the equation are not properly modified, the sign of the ϕ will be also positive. This goes againist the convention which states that inside the body the level set has to be negative. Therefore, there are two possible path to follow: modifing the equation in a way that results will become automatically negative or using the same procedures and then change sign in the inner domain.

The result with Neumann's boundary conditions are depicted in figure 3.5 where a cartesian 50×50 nodes mesh was used over a square domain with an edge equal to 1. The black circle is the object that was used to test this method, its radius is equal to 0.25.

As it is visible, the solution seems to accurately reproduce the signed distance function from this figure. However, the quality of the solution needs to be proved by looking at the error with respect to the exact ϕ. The deviation from the exact solution, at the end of the iterative procedure, and the plot of the functional during the process, can be found in figure 3.5. From this result, as expected, the maximum error is at the farhest point of the grid so at the corners where its absolute value is equal to $15 \cdot 10^{-3}$. The convergence of this algorithm, which actually reduces the functional from a value of 12 to less than 2, is reached before 20 iterations even if the relaxation factor is $\alpha = 0.8$ which is quite high. Using different values of α leads to obtain the same result but with a different number of iterations. In particular, the oscillation of the functional are higher but the iterations required are lower. The optimal trade-off can be found when the procedure doesn't diverge and the number of steps is the minimum.

In the framework of cell centered finite volume discretization, only the Neumann boundary conditions were employed. In this case, *bitpit*'s library was used to handle the mesh and, eventually, bodies with complex shape.

The first test was conducted using a circle of radius equal to 1. The results are depicted in figure 3.6. Different grid, over the same domain were tested. In particular, the domain

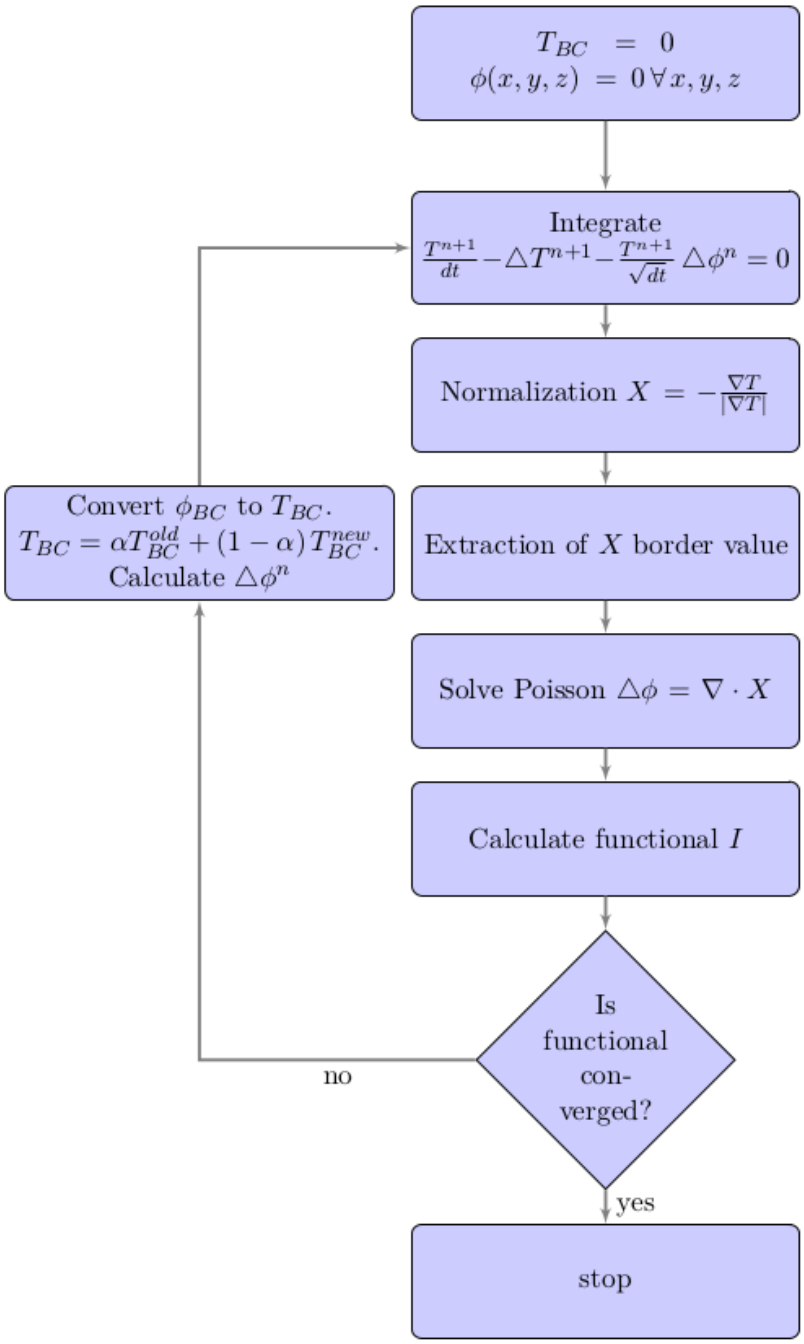
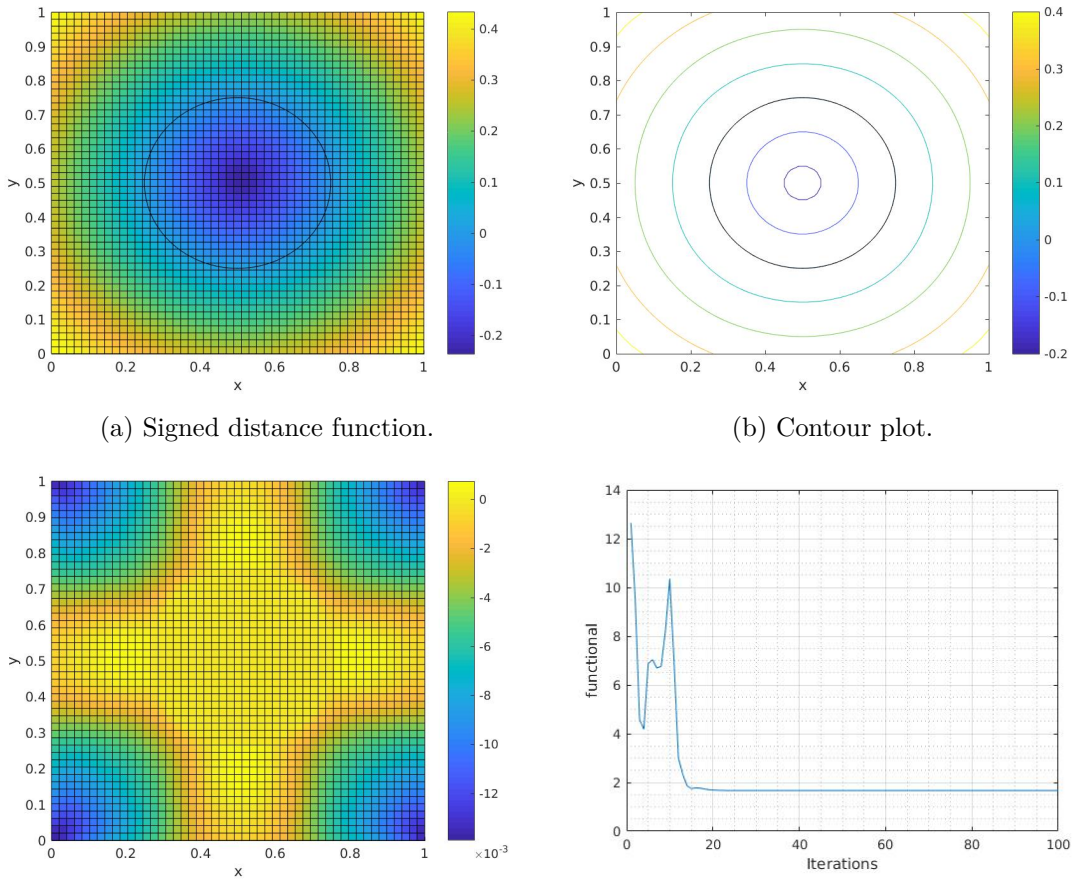


Figure 3.4: Flow chart of the algorthm developed for applying the Neumann BCs to Poisson problem.

was a two dimensional square with a side of length 2 while the figure was positioned in its center. The error, like in the previous case, is higher in the points far from the figure and



(a) Signed distance function.

(b) Contour plot.

(c) Final error.

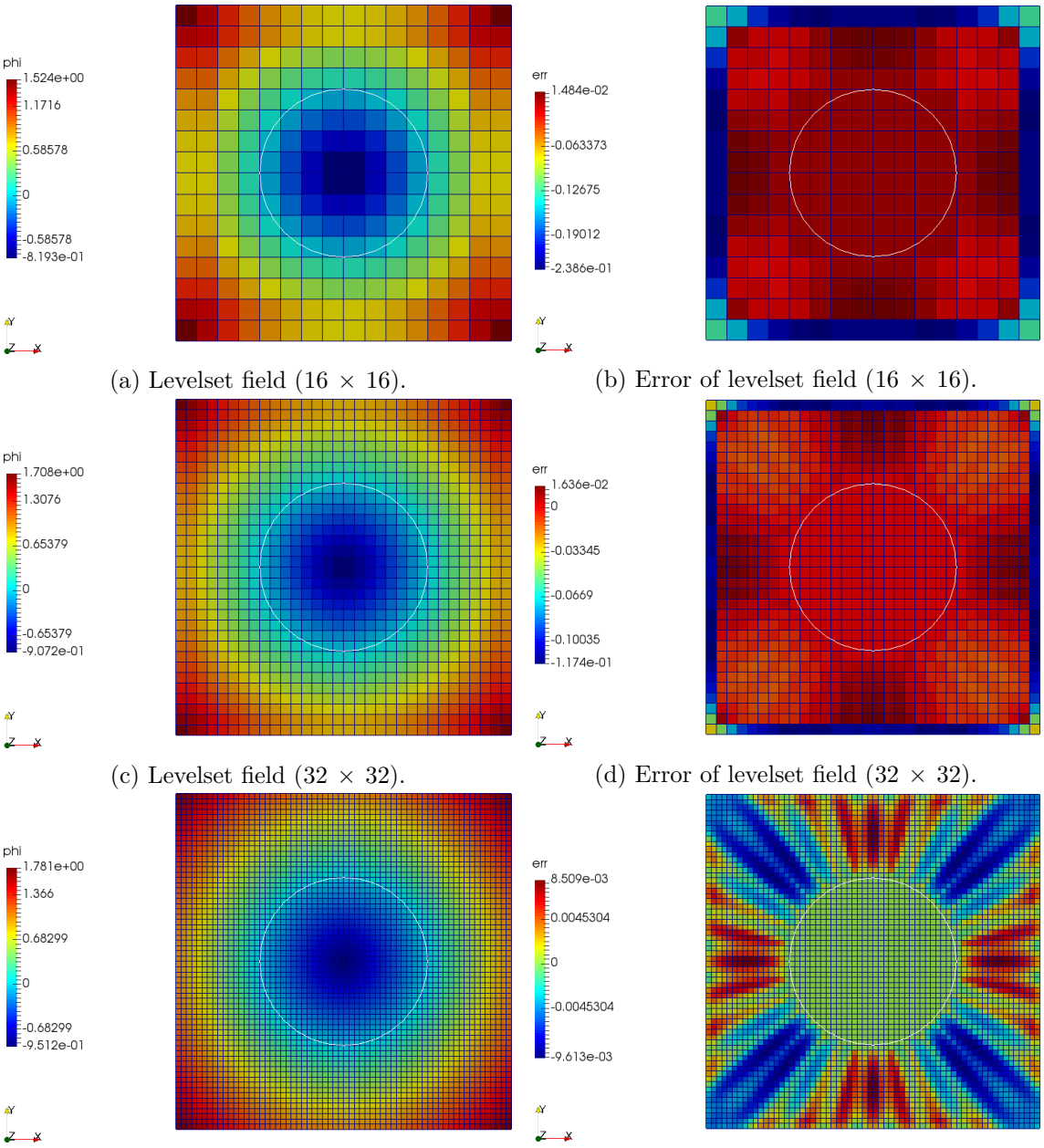
(d) Functional I.

Figure 3.5: Levelset function calculated over the domain for the Dirichlet-Neumann iterations.

when the mesh gets finner it decreases.

With this setting no relaxation factor was needed to apply the Dirichlet boundary conditions for the first integration problem. The value at interface border, where the distance value is not know, was reconstructed using a second order extrapolation and then converted into temperature.

Since *Bitpit* can easily manage complex figures, also an L-shape body was tested. This case was intended to show the ability of the heat method to recover the exact distance even with the presence of kinks and sharp edge in the body. The levelset fields show, for the different grid, a good behavior and the signed distance function seems to be correct. This is supported by the low error committed and, being the scheme consistent, it decreases as the cells become smaller. However, during the propagation there is some kind of interaction with the border. This can be spotted from the error field with the 16×16 and 32×32 grids while with the 64×64 it is not present. Usually the interaction is created from the



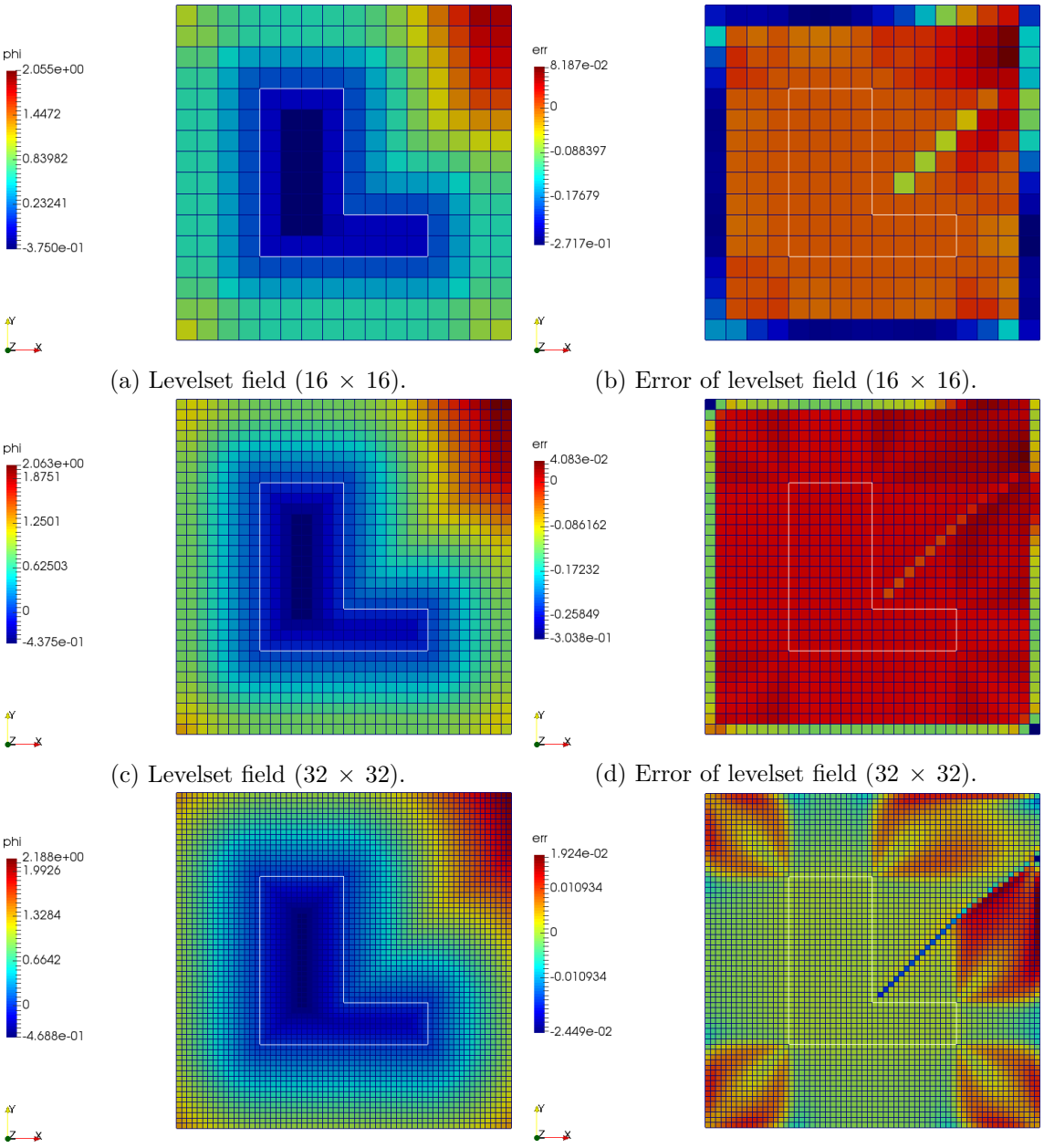
(a) Levelset field (16 × 16). (b) Error of levelset field (16 × 16).

(c) Levelset field (32 × 32). (d) Error of levelset field (32 × 32).

(e) Levelset field (64 × 64). (f) Error of levelset field (64 × 64).

Figure 3.6: Solution of the Heat method with a cell-centered FV discretization on different grid for a cirle.

cells' border that are interested by the shock. Therefore, it could be inferred that a large cell-dimension lead to a result influenced by the shock.



(a) Levelset field (16 × 16). (b) Error of levelset field (16 × 16).

(c) Levelset field (32 × 32). (d) Error of levelset field (32 × 32).

(e) Levelset field (64 × 64). (f) Error of levelset field (64 × 64).

Figure 3.7: Solution of the Heat method with a cell-centered FV discretization on different grid for an L figure.

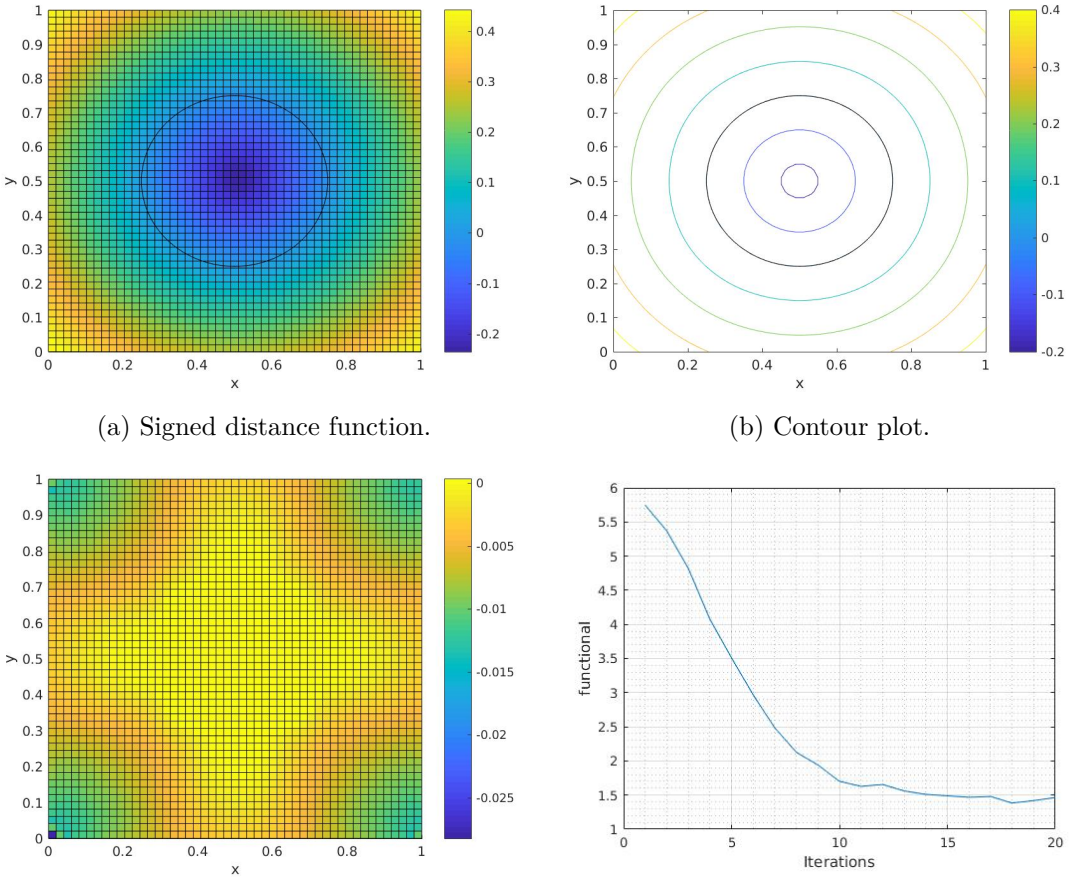
3.2.6 Fixed-point iterations with BCs through optimization

In this case the algorithm has a similar structure of the previous case but the steps taken inside the iterations are different. Like before, the goal is to reduce the functional until it converges to its minimum value. However, the BCs applyed to the Poisson problem

are of Dirichlet kind and they are obtained through an optimization loop handled by a *Matlab*'s optimization tool. The BFGS iterative procedure was choose and, for each step, an optimization cycle was performed. The gradient of the function 3.16 is calculated using the adjoint method and then supplied to the BFGS optimization function.

In the end, when the ϕ values at borders are found, in a way that I is minimum, the optimization stops and a new iteration starts taking this BCs as temperature condition for the initial integration problem. Again the temperature value are relaxed using an α parameter. The whole algorithm is depicted in the flow chart of figure 3.9.

The results are similar, even if not identical, to the Dirchlet-Neumann iterations. The grid, as well as the body, was not changed from the previous case. The picture 3.8 reports the signed distance function over the domain and the resulting iso-distance line from the circle as well as the error and functional value through the iterations.



(a) Signed distance function.

(b) Contour plot.

(c) Final error.

(d) Functional I.

Figure 3.8: Levelset function calculated over the domain for the Dirichlet-Dirichlet iterations.

Even in this case the solution seems to be the exact one. The error committed has

the same order of magnitude of the one with Neumann's BC except for one point in the south-west corner where it is higher in absolute value. Likely, this deviation was due to the value assigned in that corner from the optimization cycle that controls each single value of borders's nodes.

Even in this case, the functional starts from an higher value and then it decreases until it converges to a value almost equal to the one obtained with the Dirchlet-Neumman procedure. However, with respect to the previous case, the initial value of functional is lower and, using again $\alpha = 0.8$, there are not abrupt oscillations. This should be due to the optimization method that finds the best border condition at each step while in the other case there wasn't any optimization function.

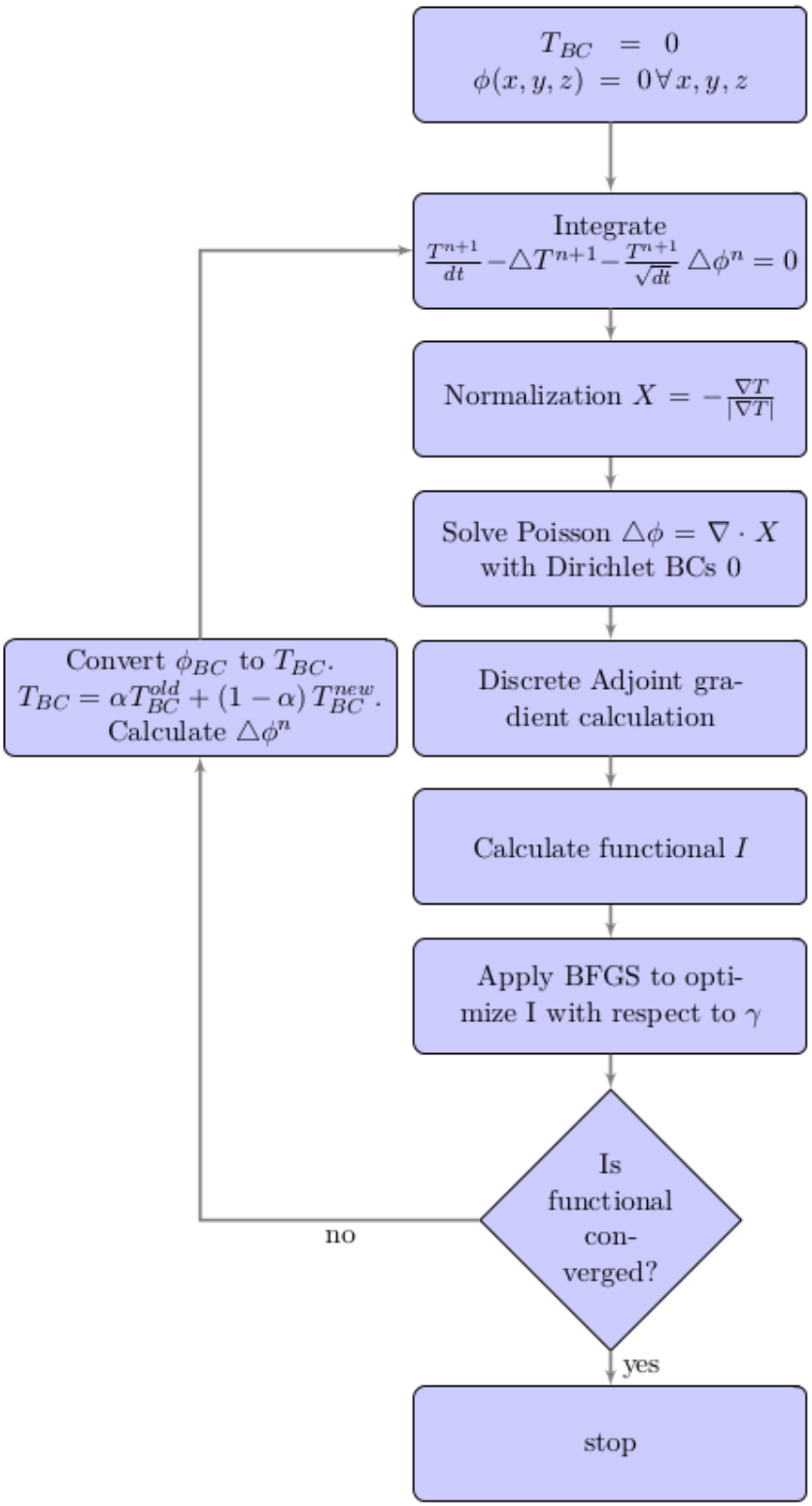


Figure 3.9: Flow chart of the algorthm developed for applying the Dirichlet BCs to Poisson problem.

3.3 Newton's method applyed to eikonal equation

The Newton's method is an iterative procedure to obtain the roots of a real-valued function [3]. A simple example of the technique can be showed in the case of one dimensional equation.

Starting from a function f defined over real numbers x, the methodology is based on using a first estimation x_0 as root of the problem and then compute a new x value to approximate the solution of the problem:

$$f(x) = 0$$

This new root's value is calcutating involving the derivative of the function in the initial guessing point. The update of the solution is obtained using the so called *Newton-Raphson formula* which reads

$$x^{n+1} = x^n - \frac{f\left(x^n\right)}{f'\left(x^n\right)}$$

where f' indicates the derivative of the function. Clearly, the initial value x_0 is used in the formula to get new approximation and in the subsequent step this new value is used as the old one and so on. A graphical explanation of the method is given in figure 3.10.

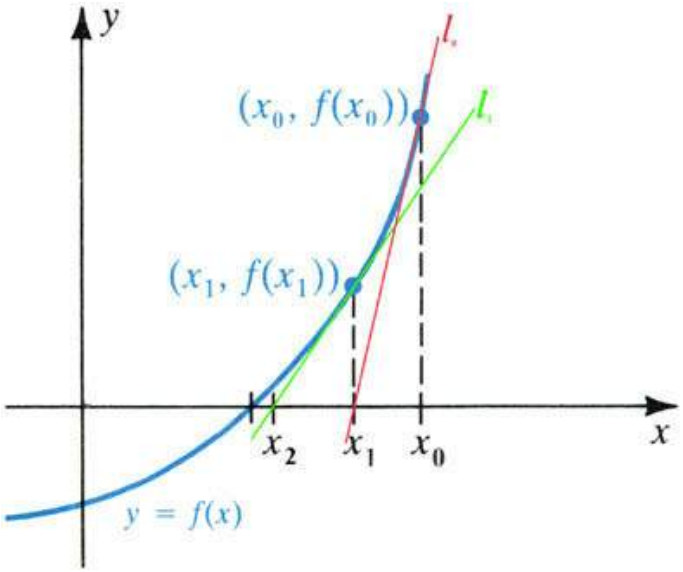


Figure 3.10: Newton's method for a one dimensional case.

The procedure stops when an adeguate approximation of root's value is reached, thus obtaining

$$f(x^{n+1}) \approx 0$$

Usually, the criterion tollerance is set by the user and higher precision is gained when the tollerance on the solution is low.

3.3.1 Newton Method for a system of non-linear equations

In case of a system of equations the concept involved in the procedure are the same; however, there are some slight changes in the formalism involved. Let the system be rapresented by

$$F(\mathbf{x}) = 0$$

where $\mathbf{x}$ is the vector of the variables while F is a vector-values function.

The algorithm for finding the $\mathbf{x}$ that makes F equal to zero is identical to the one used for the mono-dimensional situation. The only differences are that the initial estimation $\mathbf{x}_0$ is a vector containing the initial guessing for each variable and that the partial derivatives of the function are calculated with respect to all components of $\mathbf{x}$ leading to a matrix called Jacobian and defined as

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ . & \cdots & . \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \tag{3.18}$$

Then, for the system, the Newton-Raphson formula becomes

$$\mathbf{x}_{n+1} = \mathbf{x}_n + J^{-1} F(\mathbf{x}_n)$$

while the stopping criteria are similar to the one adopted for a single-variable function.

3.3.2 Derivation of the Jacobian

Starting from the unsteady Eikonal equation written as

$$\frac{\partial \phi}{\partial t} + \nabla\phi \cdot \nabla\phi = \frac{1}{c^2} \tag{3.19}$$

it is know that if ϕ is a signed distance function, then $c^2 = 1$. The residual of this equation, at a certain step, reads

$$R(\phi) = \nabla\phi \cdot \nabla\phi - \frac{1}{c^2}$$

and when ϕ satisfy the equation the residual becomes equal to 0.

At new iteration step, using Taylor' series, the residual becomes

$$R(\phi + \Delta\phi) \approx R(\phi) + \frac{\partial R}{\partial \phi}\Delta\phi + O\left(\Delta\phi^2\right)$$

where the term $\frac{\partial R}{\partial \phi}$ is equal to J, which stands for the jacobian matrix.

By discretizing the temporal term with an implicit forward euler it is possible to get

$$\frac{\partial \phi}{\partial t} \approx \frac{\phi^{n+1} - \phi^n}{\Delta t} = \frac{\Delta\phi}{\Delta t}$$

where the last equality comes from the fact that, for definition, $\phi^{n+1} = \phi^n + \Delta\phi$.

Now, the equation to update the distance field reads

$$\frac{\Delta\phi}{\Delta t} + R\left(\phi + \Delta\phi\right) = 0 \tag{3.20}$$

which can be written as

$$\frac{\Delta\phi}{\Delta t} + R\left(\phi\right) + J\Delta\phi = 0$$

and the final system of equations is

$$\left(\frac{1}{\Delta t}\left[I\right] + \left[J\right]\right)\left\{\Delta\phi\right\} = -\left\{R\left(\phi\right)\right\} \tag{3.21}$$

where the only unknown is $\Delta\phi$. Solve this linear system of equations gives the increment that the distance function has to achieve in each cell of the discretized domain.

3.3.3 Discretization of the Gradient operator

The iterative procedure that was developed in the present framework is the following:

1. choosing an initialization distance field;
2. calculate the residual of the Eikonal equation using ϕ^n;
3. obtain the jacobian matrix of the levelset function;
4. solve equation 3.21 to get the increment of previous solution $\Delta\phi$;
5. evaluate the new ϕ^{n+1} field and see if convergence has been reached.

The final convergence it is obtained when the vector of the $\Delta\phi$ is almost zero for each cells. In fact, this means that no modification to the ϕ value is needed in order to satisfy the Eikonal equation (eq. 3.19) and the temporal derivative becomes zero.

However, even if the procedure seems to be quite straighforward it hides a not-trivial step: the jacobian calculation at each iteration. There are different ways to obtain that matrix, numerical or analitical procedures. Of course the easier, conceptually speaking, are the numerical techniques which, using approximations of the derivatives, gives the complete matrix. Unfortunately, this procedure requires a huge calculation time when the cells' number is high. Moreover the accuracy of the solution will be affected by the approximations made in this step. Therefore, the best way to get the solution is to obtain the analytical Jacobian for this equation.

Jacobian matrix

The Jacobian matrix is defined by the following relationship

$$J = \frac{\partial R\left(\phi\right)}{\partial\phi}$$

It could be written that

$$\frac{\partial R(\phi)}{\partial \phi} = \frac{\partial}{\partial \phi}\left(\nabla\phi \cdot \nabla\phi - \frac{1}{c^2}\right)$$

and using the tensorial notation, the gradient operator becomes a tensor that, applyed to a scalar field, gives back the gradient components of the field in each point. In the discretized form, the tensor was represented using a blocks matrix which have the different direction derivative, for each point of the grid, in a different block of the matrix itself.

After this clarfication, the analitical derivation continues as follow

$$\frac{\partial}{\partial \phi}\left(\nabla\phi \cdot \nabla\phi - \frac{1}{c^2}\right) = \frac{\partial}{\partial \phi_i}\left(G^l_{ij}\phi_j G^l_{ik}\phi_k\right)$$

where the subscripts i, j, k indicate the element of the matrix or vector, while the superscript l denotes the matrix's block discretizing a certain directional derivative. Written in this form the tensor are linear, therefore these further passages are done

$$\begin{aligned}
\frac{\partial}{\partial \phi_i}\left(G^l_{ij}\phi_j G^l_{ik}\phi_k\right) &= G^l_{ij} G^l_{ik} \frac{\partial}{\partial \phi_i}\left(\phi_j \phi_k\right) = \\
&= G^l_{ij} G^l_{ik}\left(\frac{\partial \phi_j}{\partial \phi_i}\phi_k + \frac{\partial \phi_k}{\partial \phi_i}\phi_j\right) = \\
&= G^l_{ij} G^l_{ik}\left(\delta_{ij}\phi_k + \delta_{ik}\phi_j\right) = \\
&= 2G^l_{ij} G^l_{ik}\phi_k
\end{aligned}$$

In the end the jacobian is defined as

$$J = 2G^l_{ij} G^l_{ik}\phi_k \tag{3.22}$$

where, in the second product, after that the gradient of ϕ has been calculated, there is no summation.

Another important point for the definition of the discrete gradient matrix is how to treat the boundaries. The best numerical scheme was established by making different tries and it was inferred that an upwind scheme is the most suitable to get stability. This is certainly due to the fact that the Eikonal equation is hyperbolic and information travel on the characteristic lines emanated from the body.

Instead, the discretization technique used was the cell centered finite difference approach. At borders, where a cell does not have neighbours in all the directions, a first order discretization was employed. Since the information stream from inner cells of the domain to outer cells then the border value is not important for derivative determination (like it was pointed out before). Therefore, no boundary condition influences the gradient's matrix.

Initialization of the domain

One of the most important point to achieve the convergence of the Newton's method is the initial condition. It is well known that when the first guess for initialize the field is

far from the wanted solution, then the procedure could diverge or, in the worst scenario, converge to another solution. For this reason, a crutial point is to find a "good" initial signed distance function.

The solution adopted for this problem was to initialize the ϕ field using the first integration problem of the heat method, exploited in equation . Moreover, since only a good first guess was needed, the boundary condition used was an homogenous Dirichlet boundary condition equal to

$$T_{BC} = 0$$

3.3.4 Results

In the case of heat method, for the initialization of the field, a cell centered finite volume discretization was applied. A first order finite difference upwind scheme was used for the gradient's stencil involved in the assembly of the jacobian matrix.

A test case of a simple geometry, where the exact distance function could be obtained analitically, was performed. After that this validation phase was concluded a more complex geometry, where levelset would present kinks was employed. Moreover, the code was also extended to work in 3D space. The results of the two cases, and a three dimensionals case, will be presented in the next two sections.

Circle geometry

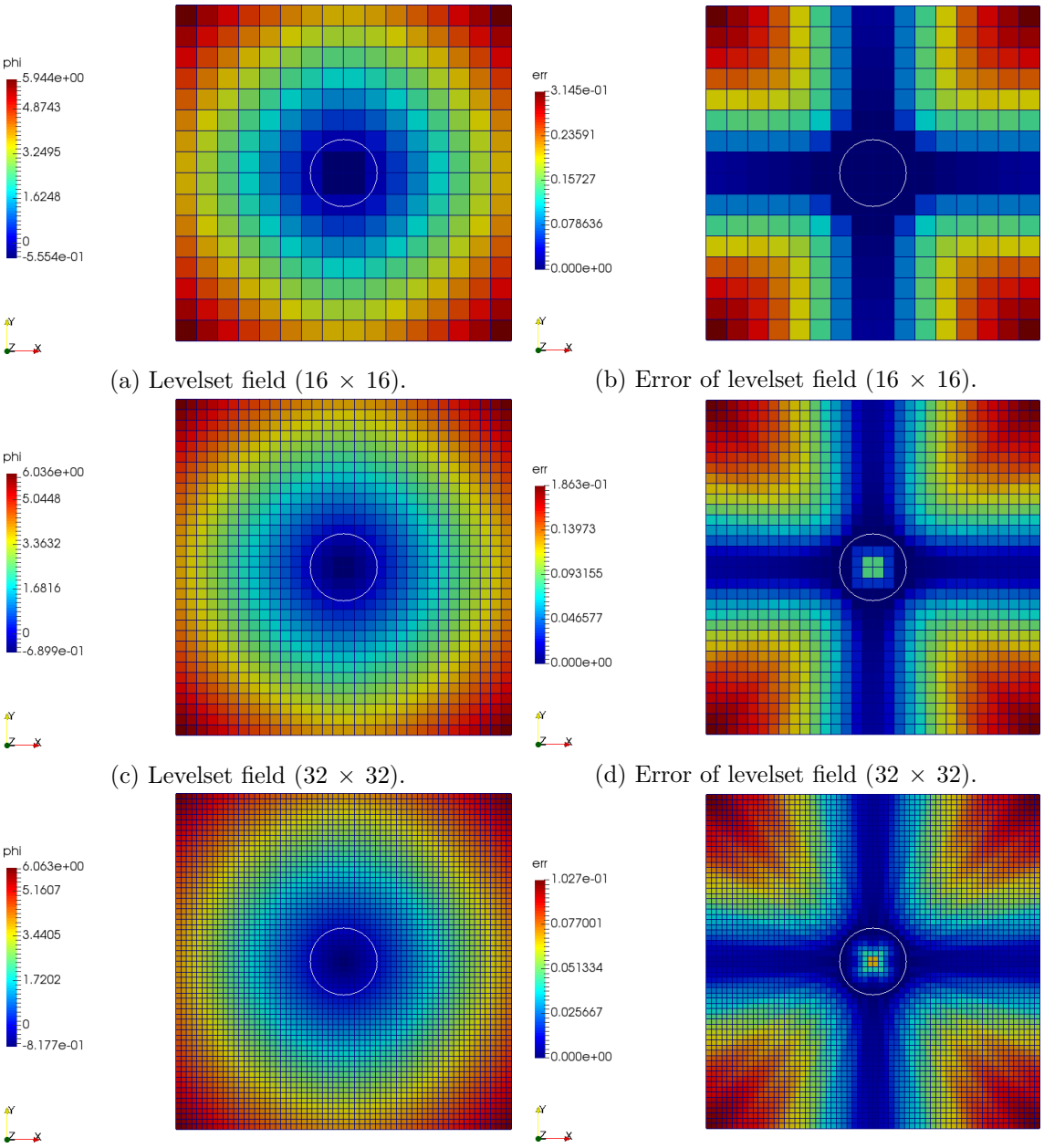
The test case for this methodology of levelset propagation was again a simple two dimensional circle. This simple case was used in order to simply reconstruct the solution with an analitical function and then find the error committed by the method. Results are depicted in figure 3.11.

With this smooth shape no kink or expantion of the levelset function can take place and, since the results were good enough, more complex geometries were tested.

L-shape geometry

For the L-shape two dimensional figure, different meshes were employed in the simulation. The variuos grid adoperated were used to see how the error and solution shape would change.

This geometry permitted to test if the code is able to handle "shoks" and "expantions" of the levelset function. These phenomena have a compleately different meaning in case they happen in the distance field. In fact, the discontinuity that occurs in the signed distance function happens in the first derivative and not in the value of the function itself. The physical sense of the shock is that, in points where it is present, the derivative suddenly changes and this implies that the closest point of the body, from which we are calculating the distance, relies in a different region of the figure. Instead, while talking of expantion, this occurs when there are a set of points which are equidistant from the body (like in the case of a sharp edge). Since in the latter case there is no discontinuity, the error is lower then in the former situation.



(a) Levelset field (16 × 16).

(b) Error of levelset field (16 × 16).

(c) Levelset field (32 × 32).

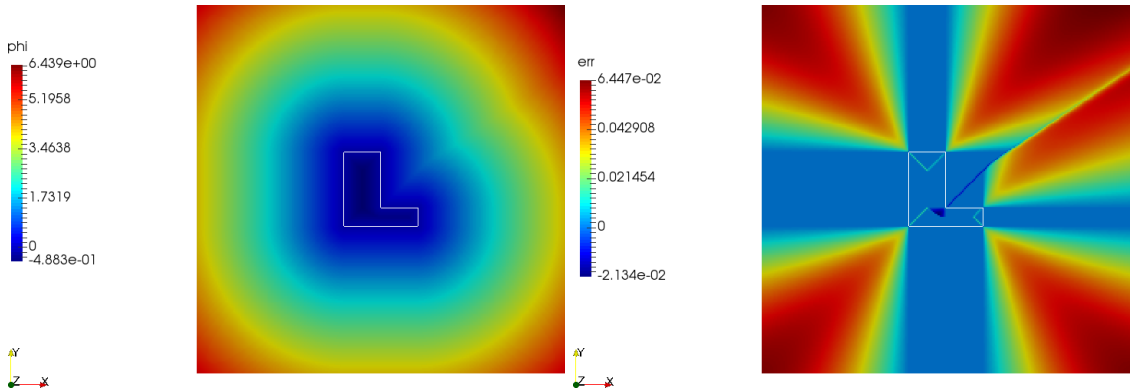
(d) Error of levelset field (32 × 32).

(e) Levelset field (64 × 64).

(f) Error of levelset field (64 × 64).

Figure 3.11: Solution of the newton's method with different grid for the circle.

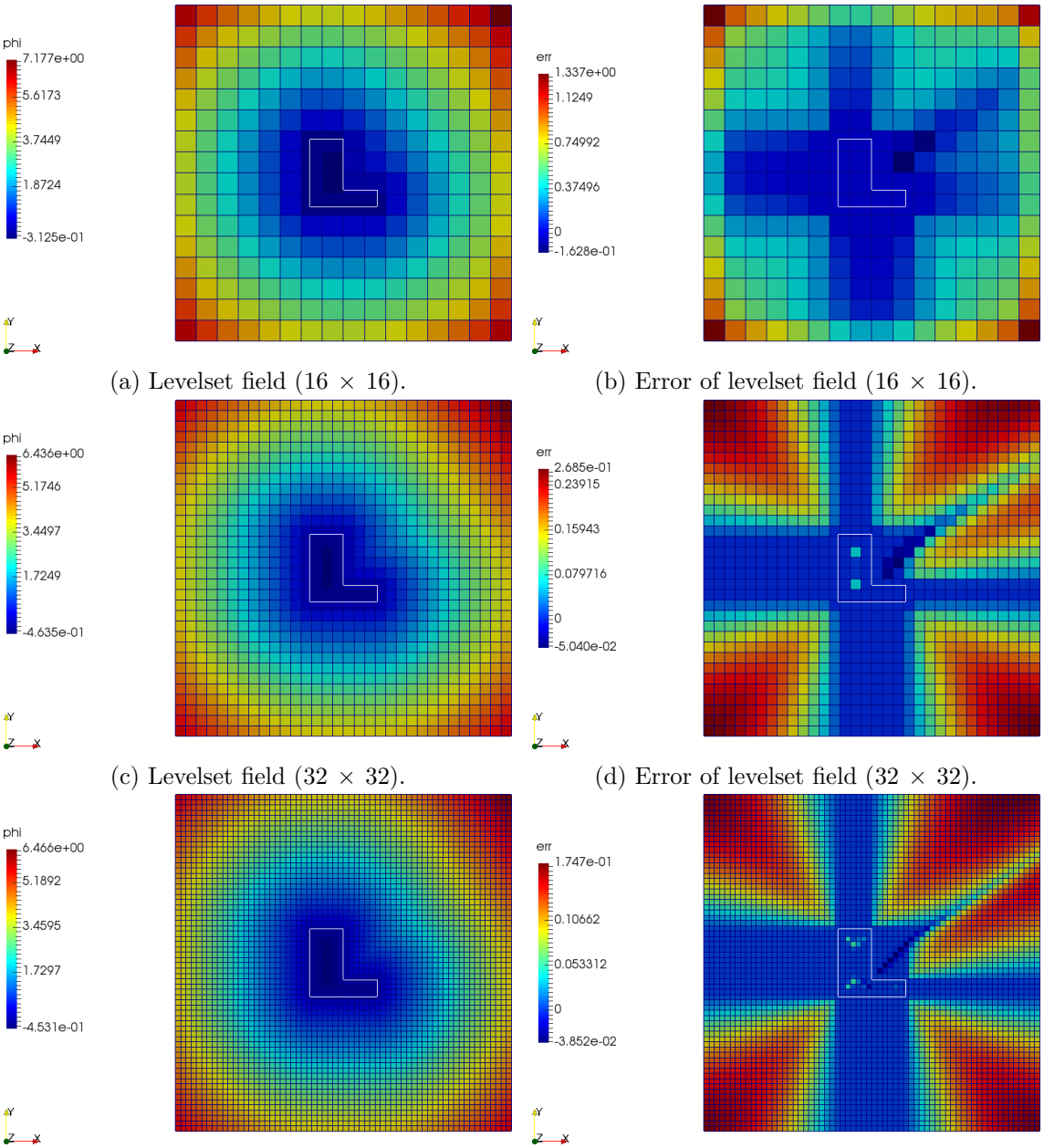
The finnest mesh employed had 256x256 nodes and the solution, like the error, is reported in figure 3.12. In this case the result is well defined and it has propagated correctly in the domain. The cells with higher error are the ones which are crossed by the shock and the expansions.



(a) Levelset field. (b) Error of levelset field.

Figure 3.12: Solution of the newton's method with 256×256 nodes for L-shape figure.

The solution shown in picture 3.13 trivially show that as much as the grid is refined then the solution becomes smoother and come closer to the correct levelset function. In fact the error decreases while the spacing between cells becomes smaller. As it was pointed out above, the higher error (in absolute value) is located in the cells that are interested by the shock and even if the grid is finner it remains quite high. However, the number of cells interested by the phenomena is lower.



(a) Levelset field (16 × 16). (b) Error of levelset field (16 × 16).

(c) Levelset field (32 × 32). (d) Error of levelset field (32 × 32).

(e) Levelset field (64 × 64). (f) Error of levelset field (64 × 64).

Figure 3.13: Solution of the newton's method with different grid for L-shape figure.

Sphere geometry

The generalization of the code was done on a simple geometry, a sphere, in order to be able to recover the exact distance field and get the error committed.

The discretized domain was a simple cube and for each direction 16 cells were emplyed

for the calculation. Figure 3.14 represent a cut of the domain in order to visualize the geometry inside it. As it can be seen the sphere is located at the center of the domain.

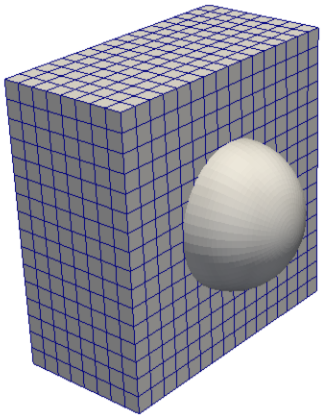
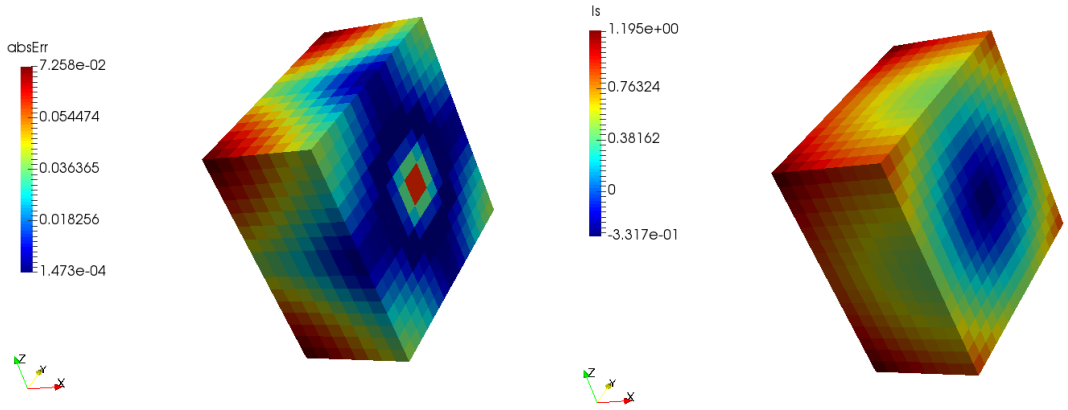


Figure 3.14: Body and 3D domain.

The cutted domain, with the calculated solution, is visible in figure 3.15, where also the absolute value of the error is shown. Even with this low number of cells the solution seems to be fine and it is well behaved in all the computational domain, internally and externally to the body.



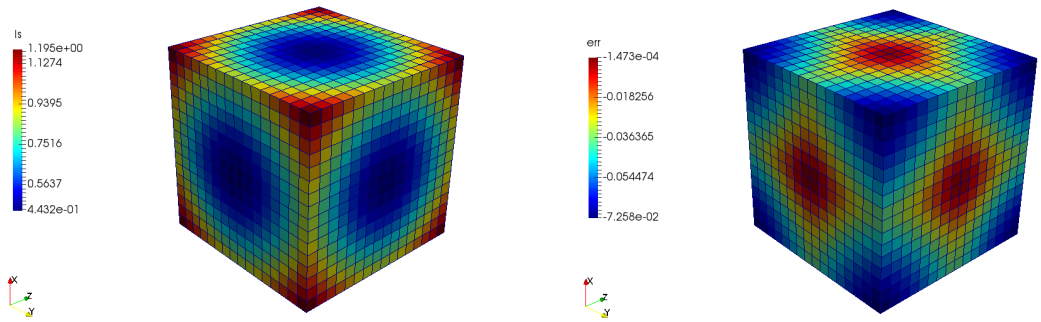
(a) Absolut error in levelset field.

(b) Internal levelset field.

Figure 3.15: Cutted 3D domain and the solution from this "internal" point of view.

In conclusion, the levelset calculated using an initialization based on heat method and then exactly determined by the newton's method can be seen in figure 3.16. From this picture, especially looking at the error, it is possible to understand that the levelset shape is correct but to get a better solution the number of cells should be increased. However, also the error behaves as expected being higher (in absolute value) at corners of the cube,

which are the points positioned further in the domain.



(a) Signed distance function.

(b) Error of levelset field.

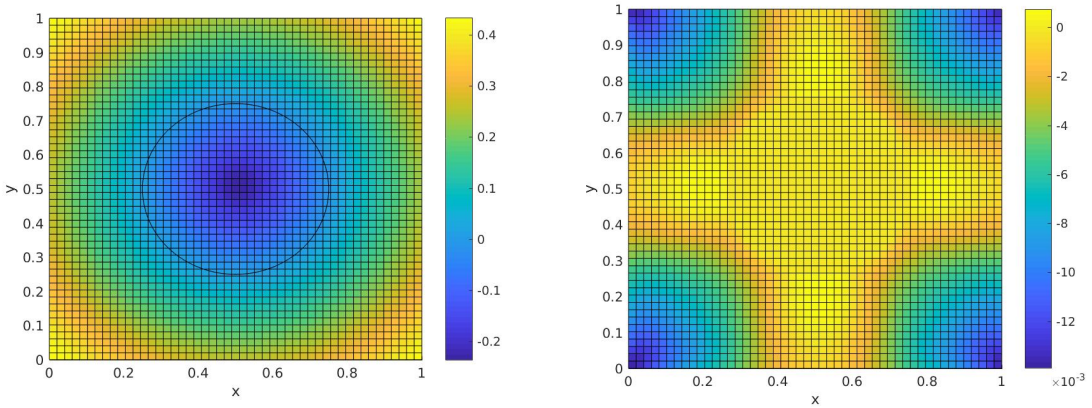
Figure 3.16: Levelset function, and its error, calculated over the 3D domain.

3.4 Comparison of the results

In the end, a comparison between the two different methodologies that were exploited in this chapter can be done. The test case that was choose to use is the simple circle of radius 0.25 in a square domain of edge equal to 1.

The results are depicted in figure 3.17, 3.18 and 3.19. As it can be seen, the error's plots show almost the same order of magnitude. However, the Newton methods presents the lowest deviation from the correct solution. Moreover, in this technique, the highest pick of error magnitude occurs at the center of the figure, where there is a singular point. In the corners the deviation form the real value is around 7×10^{-3} which is almost the half of the one commited with the other procedures. The optimization algorithm, together with the Newton iterations, posses an error very low in the middle of the domain and only near the corners the results become less reliable. On the other hand, for the Dirichlet-Neumann case, the error starts to increase close to the interface of the body. All the methodologies present a "cross" shape error with a negligible value inside this region. This should be due to the simple configuration that was tested.

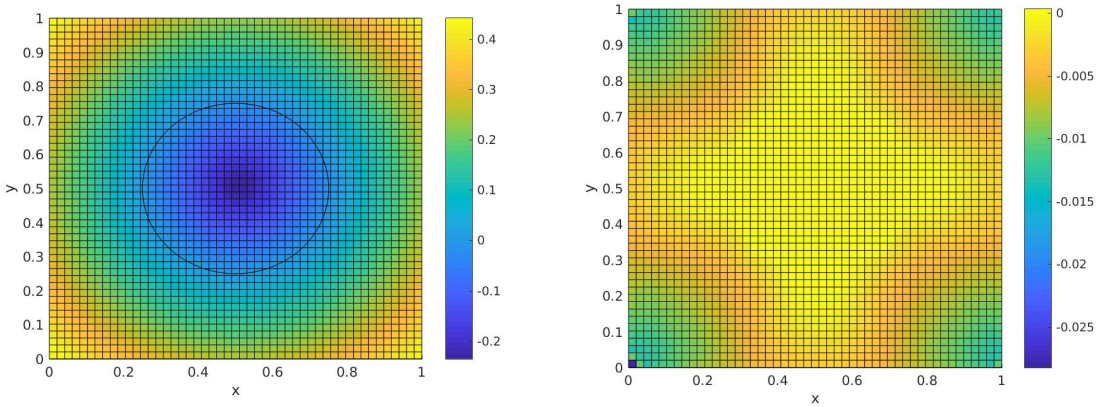
Summing-up, the most suitable procedure seems to be the Netwon one. It produces the lower error and the number of iteration to converge is the minimum. In fact only 6 iterations were needed while, for the other cases the procedure had to be repeated 20 times to converge. Therefore, both the accuracy and the computational costs gets better in the Newton framework.



(a) Levelset field.

(b) Error of level set field.

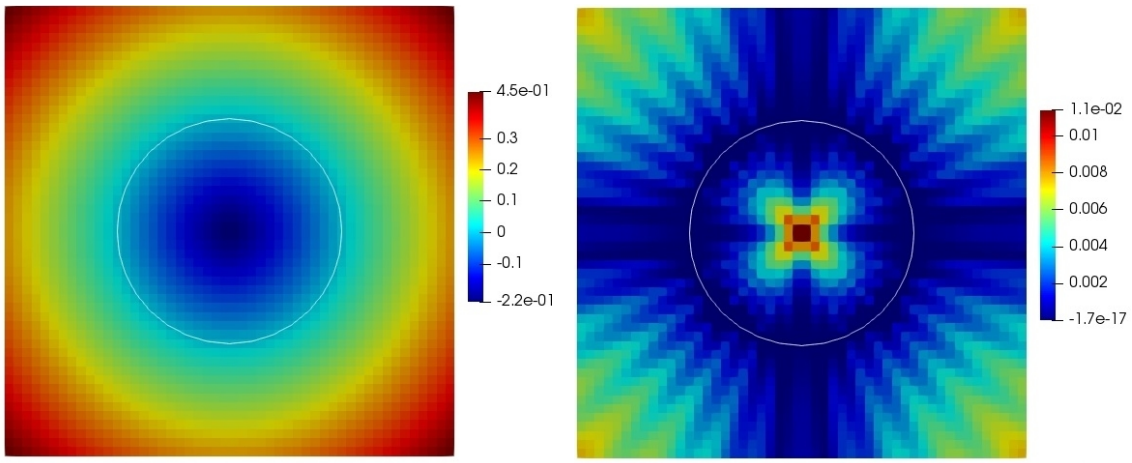
Figure 3.17: Results of the Dirichlet-Neumann iterations.



(a) Levelset field..

(b) Error of level set field.

Figure 3.18: Results of iterations with BCs obtained through iterations.



(a) Levelset field.

(b) Error of level set field.

Figure 3.19: Results of the Newton method.

Chapter 4

Transport of levelset function

Transporting the level set function in time means to evolve the signed distance function field accordingly to the convection equation, which reads

$$\frac{\partial \phi}{\partial t} + \mathbf{V} \cdot \nabla \phi = 0$$

which can be identified as an Hamilton-Jacobi equation because it is a first-order, non-linear partial differential equation for the so called Hamilton's principal function, in this case ϕ. In general this equation reads

$$\frac{\partial \phi}{\partial t} + H\left(\nabla \phi\right) = 0 \tag{4.1}$$

where ϕ is the variable while $H\left(\nabla \phi\right)$ is the Hamiltonian function and it can be dependent from both time and space.

4.1 Hamilton-Jacobi equations and conservation laws

Usually, in order to apply a finite volume discretization scheme to an equation it is easier to write a partial differential equation in its conservative form. Taking as an example the *Burger's equation* in one dimension, this reads as

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0$$

and it is a non-linear PDE which can develop shocks in the solution even if initial data are smooth and also non-physical exapansion shocks if entropy condition is not considered. Writing this equation in the conservation form means to show that a certain quantity associated with function u does not change through time. Therefore, this is obtained by writing the previous equation as

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}\left(\frac{u^2}{2}\right) = 0$$

this implies that the changes through time of u are equal to the spatial variation of the quantity $\frac{u^2}{2}$ which is conserved.

Now considering the generic one dimensional Hamilton-Jacobi equation

$$\frac{\partial \phi}{\partial t} + H\left(\phi_x\right) = 0$$

and deriving in space all terms it becomes

$$\left(\phi_t\right)_x + H_x\left(\phi_x\right) = 0$$

now commuting the derivarivation order of the first term and changing variable from ϕ to a u in a way that

$$u = \phi_x$$

the final equation reads

$$u_t + H\left(u\right)_x = 0 \tag{4.2}$$

which is a scalar conservation law. This direct correspondence can be draw only in one dimension while in cother cases the equation becomes vectorial and needs to be splitted over the singular dimension. However, the important conclusion that is inferred from this model is that the solution to the scalar conservation law represents the derivative of the Hamilton-Jacobi solution ϕ. Therefore, the Hamilton-Jacobi equation shares many properties with the conservation laws like developing kinks in the solution from smooth initial data and the possible non-physiscal expansion shocks if not corrected. However, an important difference is that the solution to Hamilton-Jacobi equations are usually continuous and only the first derivative can present discontinuities. One of the lastes work in the field of numerical solution to Hamilon-Jacobi equation was carried out by Osher and Shu in [18] with the introduction of essentially non-oscillatory methods for this class of problems.

4.2 WENO procedure

Starting from the generic hyerbolic conservation law, which reads

$$u_{,t} + f\left(\nabla\left(u\right)\right) = 0 \tag{4.3}$$

it is possible to derive a fully discrete scheme both in space and time. The starting point is to discretize the computational domain into cells and introduce the variable cell's average value (in cell's center), defined as

$$\bar{u}_i = \frac{1}{\Delta x_i} \int_{V_i} u\left(x, y, z, t\right)\, dV$$

where u is the variable, the subscript i denotes the i-th cell of the domain, V is the cell's volume.

Applying a finite volume discretization scheme the equation 4.3 becomes

$$\frac{d}{dt}\bar{u}_i + \frac{1}{V_i}\sum_{face} f(u_{face}) \cdot \overline{n}\, A_{face} = 0 \tag{4.4}$$

where the first term is a derivative in time of the cell's average value while the second is a summation over the faces of the cell of a function f (function of the variable) evaluated in that points and scalarly multiplyed by the normal and the area of that face.

It is straightfoward to infer that some kind of reconstruction is needed in order to being able to evaluate f at face of the cells. In fact, only cell's averages are avalilable. The approximation can be done in many different ways, and it is crucial to determine both the consistency, stability and accurancy of the numerical method implemented.

A particular procedure to accurately calculate numerical fluxes, even up to 5-th order of accurancy, it is the so called *WENO schemes* which stands for Weighted Essentially Non-Oscillatory schemes. These latter were developed from the *ENO schemes* (Essentially Non-Oscillatory schemes) firstly introduced by Harten, Osher, Engquist and Chakravarthy [10]. The key idea behind this method is to evaluate the smoothness of the stencils (subset of cells) used to approximate the value of the fluxes in equation 4.3 in order to use only the ones that do not contain variable's jump, therefore avoiding spurious oscillations where the solution present shocks. A one dimensional example of the stencils that could be used in the reconstruction is reported in figure 4.1. Moreover, the scheme is designed to have an high order of accuracy and only near discontinuities it is lowered to stabilizing the scheme. In fact, the concept is to pick up only one between the stencils proposed (obvoiusly the smoothest) and using it to calculate f.

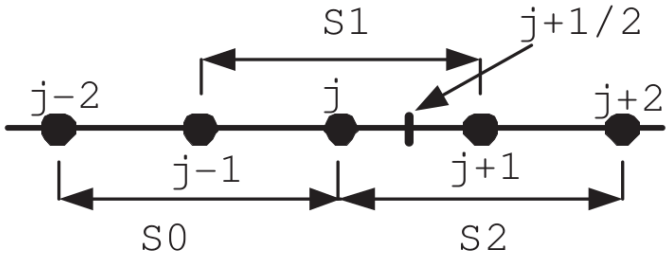


Figure 4.1: Example of 1D stencils.

Later Liu, Osher and Chan [15] introduced WENO shemes to improve the accuracy of the ENO schemes while maintaining the same robustness. In fact it was proved that the ENO philosophy is overkilling in smooth regions where no jump are encountered. Therefore, a weighted average of all the stencils is taken to evaluate the fluxes. In this way, WENO actually use all datas from surroundings cells and then, changing weights, it priviledges the information coming from stencils which do not contains discontinous data. In fact, these weights are designed in such a fashion that in "well-behaved" data regions they are equal to optimum values, enabling the scheme to be the most accurate possible, while near discontinuities the order of accuracy is lowered and only stencils not

crossed from kinks are considered. Broadly speaking, this scheme can be considered as "centered" where solution is smooth and it changes near discontinuities trying to wiping out instability that could arise in that case [12].

In conclusion, the main advantages of ENO and WENO scheme could be summed up as being able to achieve high order of accuracy throughout the entire domain, having sharp and non.oscillatory shock transitions, being robust even when strong shocks are formed and being suitable for the simulation of complex flow phenomena like interactions between shocks and vortices. Then the advantages of WENO over ENO are the higher accuracy reached using the same stencils, it is easier to be coded and implemented and the numerical flux function is smoother [22].

4.2.1 WENO on Octree grids

The WENO procedure, for an adaptive grid, it is not straighforward since the reconstruction polynomial it is based on variable stencils configuration. In fact, every cell having different neighbour's configurations have also different directional stencils. The following method was develpoed by Semplice et. al in [21].

First, it is necessary to define the cell's neighbours which can be taken, for the generic j-th cell as

$$N_j = \{k \neq j \ : \ \partial\Omega_k \cap \partial\Omega_j \neq \oslash\}$$

where Ω_j is the cell's domain and $\Omega_k$is the domain of one of the neighbours (see figure 4.2).

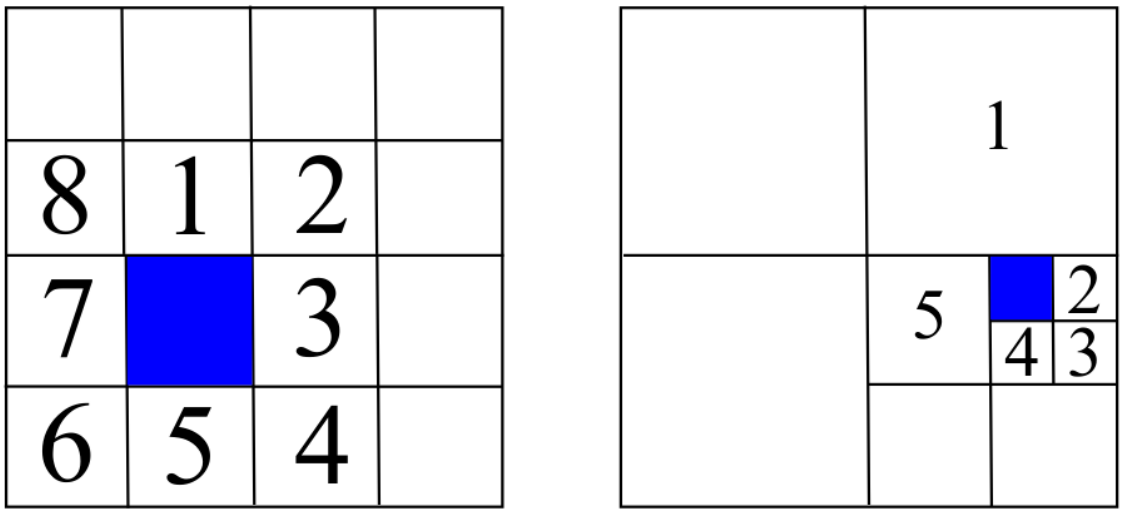


Figure 4.2: Two grid configuration: on the left a uniform cartesian mesh while on the right a particular case for the adpative grid. The j-th cell is color filled while the neighbours are numbered.

Now considering Ω_j and using a compact stencil, the reconstruction can in that specific cell can be done by means of the mean values of the function u in the cell, U_j, and its

first vertices' neighbours (the ones numbered in figure 4.2). Having a locally smooth function $u(x)$ a third order accurate polynomial, namely a parabola, can be choose as a reconstruction polynomial. The parabola reads

$$\begin{aligned} P =& U_j + p_x\,(x - x_j) + p_y\,(y - y_j) + \\ &+ \frac{1}{2} p_{xx} \left[(x - x_j)^2 - \frac{h_j^2}{12} \right] + p_{xy}\,(x - x_j)\,(y - y_j) + \frac{1}{2} p_{yy} \left[(y - y_j)^2 - \frac{h_j^2}{12} \right] \end{aligned} \tag{4.5}$$

and it is obtained from the interpolation of the cell-averages values of cells in the stencil, thus enforcing conservation. The polynomial 4.5 is called *Optimal* polynomial $P_{opt}(x)$ and the conditions that are imposed to obtain its coefficient can be written as follow

$$\frac{1}{h_k^2} \int_{\Omega_k} P_{opt}\, dx = U_k, \quad k \in N_j \tag{4.6}$$

where h_k is the mean dimension of the k-th neighbour cell, Ω_k is the neighbour volume and U_k is the mean value of the function in that neighbour cell. In this way, the polynomial fits the cell averages of the cells contained in N_j in a least-square sense.

Therefore, for each neighbour there is an equation to be satisifed and the solution of the system of equations gives the coefficients of the P_{opt}. This equations' system reads

$$Ac = r$$

where A is a $|N_j| \times 5$ matrix dependent on the geometrical position of the neighbours, $c = \{p_x, p_y, p_{xx}, p_{xy}, p_{yy}\}^T$ is the polynomial coefficients vector (the unknown) and $r = U_k - U_j$ is the vector containg the cell-average difference between the cells and its neighbours. The k-th row of the A matrix (corresponding to the k-th neighbour) is

$$\begin{pmatrix} x_k - x_j \\ y_k - y_j \\ \frac{1}{2}\left[(x_k - x_j)^2 + \frac{1}{12}\left(h_k^2 - h_j^2 \right) \right] \\ (x_k - x_j)\,(y_k - y_j) \\ \frac{1}{2}\left[(y_k - y_j)^2 + \frac{1}{12}\left(h_k^2 - h_j^2 \right) \right] \end{pmatrix}^T$$

the element of the matrix's rows come from the solution of the integral 4.6 where the polynomial 4.5 has been introduced.

After having found the coefficient of the optimal parabola, it is possible to reconstruct the value of the function $u(x)$ in every point of the cell. However, this optimal reconstruction doesn't take into account possible discontinuity in the function itself. Therefore, the result could lead to instabilities and this procedure doesn't guarantee stability anymore. For this reason, in the case of kinks in the data, other polynomial have to be used to reconstruct the value. In the two dimensional case, four linear function P_γ with $\gamma = 1, ...,4$ are introduced in a way that these polynomials match exactly the U_j cell-average and the U_k neighbours' value (of a suitalbe subset of N_j) in the least-square sense. They read

$$P_\gamma = U_j + p_x^\gamma\,(x - x_j) + p_y^\gamma\,(y - y_j) \tag{4.7}$$

The four suitable subset of neighbours used to create these linear functions are choosen along the vertices direction of the cell considered. Starting from the following sets

$$\begin{aligned} N_j^E &= \left\{ k \in N_j \,:\, x_k + \frac{h_k}{2} \geq x_j \right\}, \quad N_j^W = \left\{ k \in N_j \,:\, x_k - \frac{h_k}{2} \leq x_j \right\}, \\ N_j^N &= \left\{ k \in N_j \,:\, y_k + \frac{h_k}{2} \geq y_j \right\}, \quad N_j^S = \left\{ k \in N_j \,:\, y_k - \frac{h_k}{2} \leq y_j \right\}. \end{aligned} \tag{4.8}$$

then the four stencils reads

$$N_j^{\alpha\beta} = N_j^\alpha \cap N_j^\beta, \text{with}\, \alpha \in \{N, S\},\, \beta \in \{W, E\} \tag{4.9}$$

The definition 4.8 permits to avoid stencils with only one neighbour inside which would make impossibile to determine a linear function for that stencil. For example, in figure 4.2 the stencil in the east direction becomes

$$N_j^E = \{\Omega_k \,:\, k = 1,2,3,4\}\,,\; N_j^N = \{\Omega_k \,:\, k = 1,2,5\} \;\rightarrow\; N_j^{NE} = \{\Omega_k \,:\, k = 1,2\}$$

Renaming the stencil of the four corner direction (NE, NW, SW, SE) as N_j^γ the coefficient of the four linear function p_x^γ and p_y^γ are determined solving, in the least-square sense, a system of equations for each stencil. In particular, using the expression 4.7 together with a condition close to 4.6, the resulting system can be written in matricial form as

$$A^\gamma c^\gamma = r^\gamma$$

with $c^\gamma = \left(p_x^\gamma, p_y^\gamma\right)^T$ the vector of the unknown, $r^\gamma = (U_k - U_j)$ the vector of the difference of cell-average of the cell and the neighbour in the considered stencil and A^γ being a $|N_j^\gamma| \times 2$ matrix with it k-th row equal to

$$\begin{pmatrix} x_k - x_j \\ y_k - y_j \end{pmatrix}^T$$

After having obtained the four linear polynomial the optimal parabola can be expressed as

$$P_{opt} = \alpha_0 P_0 + \sum_{i=1}^{4} \alpha_\gamma P_\gamma \tag{4.10}$$

where P_0 is a constant parabola, P_γ are the four plane determined by the neighbours in the cell's vertices directions and α are arbitrary coefficients. In this case the following choice was made

$$\alpha_0 = \frac{1}{2}, \quad \alpha_\gamma = \frac{1}{8},\, \gamma = 1, ...,4.$$

In the end, the final reconstruction polynomial reads

$$P = \tilde{\alpha}_0 P_0 + \sum_{\gamma=1}^{4} \tilde{\alpha}_\gamma P_\gamma \tag{4.11}$$

with

$$\tilde{\alpha}_\gamma = \frac{\omega_\gamma}{\sum_{\delta=0}^{4} \omega_\delta}, \quad \omega_\gamma = \frac{\alpha_\gamma}{(\epsilon + \beta_\gamma)^2}, \quad \gamma = 0, ...,4 \tag{4.12}$$

where the ω_γ are the weights obtained from the optimal weights defined above and the smoothness indicators β_γ which will be explained in details in the next section.

The ϵ is a constant introduced to avoid zero denominators; however its role is far more important than this. In fact, its choice can influence the order of accuracy of the whole method and it is important to use the right value. Namely, low value of ϵ gives an order of convergence equal to the theoretical one but only asymptotically and for fine grids; on the other hand, an high value could lead to oscillatory results. Moreover, when an adaptive grid is employed, a fixed value it is not suitable but it needs to be adjusted considering the cell dimension. Therefore, following the work of Kolb [13], the ϵ was choosen to be equal to the h of the cell itself.

Boundary conditions

Having a compact stencil made only by the first neighbours of the cell considered implies that, at border, it is not possible to evaluate the second derivatives of the polynomial in the normal direction of the border itself.

Therefore, what it was done is to erase that coefficient from the polynomial thus lowering the order of the reconstruction. In particular this means to have an A matrix, for example on a border perpendicular to x axis, with k-th row equal to

$$\begin{pmatrix} x_k - x_j \\ y_k - y_j \\ (x_k - x_j)(y_k - y_j) \\ \frac{1}{2}\left[(y_k - y_j)^2 + \frac{1}{12}\left(h_k^2 - h_j^2\right)\right] \end{pmatrix}^T$$

where the column responsible for the calculation of p_{xx} coefficients was cancelled out. The same approach was used in case of a corner cell, where all the second derivative coefficients of the polynomial were not calculated.

4.2.2 Smoothness indicators for Hamilton-Jacobi equations

A crucial part of the WENO schemes is the recognition of the smoothness of the stencils used to create the reconstruction polynomial. For the classical conservation laws the stencil smoothness indicators can be written as

$$\beta = \sum_{|\underline{\alpha}|\geq 1} h^{l(\underline{\alpha})} \int_V \left(\partial_{\underline{\alpha}} P\right)^2 \, dV \tag{4.13}$$

where $\underline{\alpha} = (\alpha_x, \alpha_y, \alpha_z)$ is a multindex notation for the order of the derivative along the three spatial dimensions, $|\underline{\alpha}| = \sum_{i=1}^{3} \alpha_i$ is the sum of the three "directional" indexes, $\partial_{\underline{\alpha}} = \frac{\partial^{|\underline{\alpha}|}}{\partial x^{\alpha_x} \partial y^{\alpha_y} \partial z^{\alpha_z}}$, $l\left(\underline{\alpha}\right)$ is the exponent of h which is the charactetistic dimension of the cell, V is the volume considered (being a squared cell's grid $V = h^d$ where d is the dimension of the problem) and P is the function to be analyzed. In this case, P is the interpolant polynomial obtained from the sampling data of the function $u : \Re^d \to \Re$; if

u is discontinuos then its derivatives, across the discontinuity, are approximatively of the order

$$\partial_{\underline{\alpha}} \approx \frac{1}{h^{|\underline{\alpha}|}}$$

This implies that

$$\beta = \sum_{|\underline{\alpha}|\geq 1} h^{l(\underline{\alpha})} \int_V \left(\partial_{\underline{\alpha}} P\right)^2 \, dV \approx \sum_{|\underline{\alpha}|\geq 1} h^{l(\underline{\alpha})} \frac{h^d}{h^{2|\underline{\alpha}|}}$$

therefore, since it is desired to have $\beta \to 1$ when a stencil contains a discontinuity[1], it is straighforward to infer that

$$l\left(\underline{\alpha}\right) = 2|\underline{\alpha}| - d$$

and substituting in the definition it is possible to obtain

$$\beta = \sum_{|\underline{\alpha}|\geq 1} h^{2|\underline{\alpha}|-d} \int_V \left(\partial_{\underline{\alpha}} P\right)^2 \, dV \tag{4.14}$$

However, this procedure doesn't hold anymore for the case of an Hamilton-Jacobi equation since the discontinuity is not in the function itself but in its derivatives. This means that, in the worst case

$$\partial_{\underline{\alpha}} P \approx \frac{1}{h^{|\underline{\alpha}|-1}}$$

this leads to

$$\beta := \sum_{|\underline{\alpha}|\geq 2} h^{l(\underline{\alpha})} \int_V \left(\partial_{\underline{\alpha}} P\right)^2 \, dV$$

and following the same reasoning of the previous case it is possible to infer that

$$\beta \approx \sum_{|\underline{\alpha}|\geq 2} h^{l(\underline{\alpha})} \frac{h^d}{h^{2|\underline{\alpha}|-2}}$$

leading to

$$l\left(\underline{\alpha}\right) = 2|\underline{\alpha}| - d - 2$$

and finally the smoothness indicators read

$$\beta := \sum_{|\underline{\alpha}|\geq 2} h^{2|\underline{\alpha}|-d-2} \int_V \left(\partial_{\underline{\alpha}} P\right)^2 \, dV \tag{4.15}$$

Using an interpolant polynomial of the general form

$$P(x) = U_j + p_x(x - x_j) + p_y(y - y_j) + \frac{1}{2}p_{xx}(x - x_j)^2 + p_{xy}(x - x_j)(y - y_j) + \frac{1}{2}p_{yy}(y - y_j)^2$$

[1] $\beta \to 1$ implies to have the optimum weight for that stencil; a stencil which is not interested by a discontinuity have $\beta \to h^2$ leading to get a weight higher than the optimum

and having $|\underline{\alpha}| \geq 2$ means to use only the following derivatives to evaluate the smoothness of the polynomial

$$\frac{\partial^2 P}{\partial x \partial y} = p_{xy}\,,\ \frac{\partial^2 P}{\partial x^2} = p_{xx}\,,\ \frac{\partial^2 P}{\partial y^2} = p_{yy}$$

therefore, a direct computation of 4.15 yelds to

$$\begin{aligned}\beta &= h^0 \int_V (p_{xy})^2\, dV + h^0 \int_V (p_{xx})^2\, dV + h^0 \int_V (p_{yy})^2\, dV = \\ &= h^d \left(p_{xy}^2 + p_{xx}^2 + p_{yy}^2\right)\end{aligned}$$

where d is the dimension of the problem.

Since the final reconstruction polynomial is obtained from the combination of a quadratic polynomial of form

$$P(x) = U_j + p_x(x - x_j) + p_y(y - y_j) + \frac{1}{2}p_{xx}(x - x_j)^2 + p_{xy}(x - x_j)(y - y_j) + \frac{1}{2}p_{yy}(y - y_j)^2$$

and four linear polynomial like

$$P(x) = U_j + p_x(x - x_j) + p_y(y - y_j)$$

the smoothness indicator for these latter are implicitly always zero, while the β for the quadratic polynomial depends on the smoothness of the data.

In conclusion when a discontinuity is encountered, the weight of the quadratic polynomial decreases, while the ones of the linear polynomials remain unalterated.

4.3 Numerical scheme

In the general form the Hamilton-Jacobi equation reads as

$$\phi_{,t} + H\left(\nabla\phi\right) = 0 \tag{4.16}$$

where ϕ is the variable while $H\left(\nabla\phi\right)$ is the hamiltonian term which depends on the gradient of the varible itself.

The discrete domain considered is a 2D Quadtree mesh, therefore having squared cells with different areas. The cells are defined as $V_{i,j} = \left[x_{i+\frac{1}{2}}, x_{i-\frac{1}{2}}\right] \cdot \left[y_{j+\frac{1}{2}}, y_{j-\frac{1}{2}}\right]$, $I_i = \left[x_{i+\frac{1}{2}}, x_{i-\frac{1}{2}}\right]$,$J_j = \left[y_{j+\frac{1}{2}}, y_{j-\frac{1}{2}}\right]$, $I_{i+\frac{1}{2}} = \left[x_i, x_{i+\frac{1}{2}}\right]$, $J_{j+\frac{1}{2}} = \left[y_j, y_{j+\frac{1}{2}}\right]$ while $I_{i-\frac{1}{2}}$ and $J_{i-\frac{1}{2}}$ are defined specularly. The dimension of the cell's edge are considered to be equal to h both on x and y directions. Using the cell's average value of the levelset, defined as

$$\overline{\phi}_{i,j}(t) = \frac{1}{V_{i,j}} \int_{V_{i,j}} \phi\left(x, y, t\right)\, dxdy$$

and the numerical hamiltonian written as

$$H\left(\nabla\phi\right) = \hat{H}_{i,j}$$

the equation 4.16 can be discretized obtaining, for each cells, the following semi-discrete equation

$$\frac{d}{dt}\bar{\phi}_{i,j} = -\frac{1}{V_{i,j}}\hat{H}_{i,j} \tag{4.17}$$

Using the generalized Roe scheme, as reported in [28], it can be approximated by

$$\begin{aligned}
\hat{H}_{i,j} = &\int_{V_{i,j}} H(\phi_{,x}, \phi_{,y})\, dxdy \\
&+ \frac{1}{2}\int_{J_j} (\min_{x\in I_{i+\frac{1}{2}}} H_1 - |\min_{x\in I_{i+\frac{1}{2}}} H_1|)\,[\phi]\left(x_{i+\frac{1}{2}}, y\right)\, dy \\
&- \frac{1}{2}\int_{J_j} (\max_{x\in I_{i-\frac{1}{2}}} H_1 + |\max_{x\in I_{i-\frac{1}{2}}} H_1|)\,[\phi]\left(x_{i-\frac{1}{2}}, y\right)\, dy \\
&+ \frac{1}{2}\int_{I_i} (\min_{y\in J_{j+\frac{1}{2}}} H_2 - |\min_{y\in J_{j+\frac{1}{2}}} H_2|)\,[\phi]\left(x, y_{j+\frac{1}{2}}\right)\, dx \\
&- \frac{1}{2}\int_{I_i} (\max_{y\in J_{j-\frac{1}{2}}} H_2 + |\max_{y\in J_{j-\frac{1}{2}}} H_2|)\,[\phi]\left(x, y_{j-\frac{1}{2}}\right)\, dx
\end{aligned} \tag{4.18}$$

where $\int_{V_{i,j}} H(\phi_{,x}, \phi_{,x})\, dxdy$ is calculated using a four point Gauss quadrature formula as reported here

$$\begin{aligned}
\int_{V_{i,j}} H(\phi_{,x}, \phi_{,x})\, dxdy = \frac{h^2}{4}\Big[&H\left(\frac{h}{2\sqrt{3}}, \frac{h}{2\sqrt{3}}\right) + H\left(-\frac{h}{2\sqrt{3}}, \frac{h}{2\sqrt{3}}\right) + \\
&H\left(-\frac{h}{2\sqrt{3}}, -\frac{h}{2\sqrt{3}}\right) + H\left(\frac{h}{2\sqrt{3}}, -\frac{h}{2\sqrt{3}}\right)\Big]
\end{aligned}$$

where h is the cell dimension and, for example, $H\left(\frac{h}{2\sqrt{3}}, \frac{h}{2\sqrt{3}}\right)$ is the hamiltonian calculated using the reconstruction polynomial in point with coordinate $\left(\frac{h}{2\sqrt{3}}, \frac{h}{2\sqrt{3}}\right)$ taking as reference system the center of the cell. The other integrals are approximated by using the midpoint integration rule, $[\phi]\left(x_{i\pm\frac{1}{2}}, y\right) = \phi^+_{x_{i\pm\frac{1}{2}},y} - \phi^-_{x_{i\pm\frac{1}{2}},y}$ and $[\phi]\left(x, y_{j\pm\frac{1}{2}}\right) = \phi^+_{x,y_{j\pm\frac{1}{2}}} - \phi^-_{x,y_{j\pm\frac{1}{2}}}$ are the jump of ϕ at cell's interfaces where the superscript "+" indicates the reconstructed value at cell from the neighbour cell polynomial while the "−" superscript is the evaluation from the cell's reconstruction polynomial, H_1 and H_2 are given by

$$H_1 = \frac{\partial H}{\partial \phi_{,x}} = u, \quad H_2 = \frac{\partial H}{\partial \phi_{,y}} = v$$

and they are the partial derivatives of the hamiltonian with u and v being the x and y components of the velocity field, respectively. Then their max and min are calculated by considering H_1 and H_2 as linear functions from the cell center and its neighbour center; therefore, one of the two value is the minimum or the maximum.

The complete discrete equation is obtained by applying a third order Runge Kutta temporal discretization.

4.3.1 Analysis of upwinding terms

Since the Hamilton-Jacobi equation is an hyperbolic problem a source of stability is needed inside the numerical scheme. Different solution could be adopted, like using a centered scheme and add artificial viscosity. However, the most natural choice is to employ an upwinding tecnique. In particular, a generalized Roe scheme was used (equation 4.18).

By looking at the numerical $\hat{H}$ hamiltonian it is possible to infer that the fist integral, being calculated using a Gauss quadrature formula, can be considered as a centered term proportional to the volume dimension $\int_{V_{i,j}} H\, dV \approx O(h^d)$ where d is the dimension of the cell's volume.

Analyzing the consistency of the scheme, a fundamental role is played by the upwinding terms which must tend toward zero when the grid size $h \to 0$. Therefore, knowing the order of the centered term, the whole upwinding part of the equation have to scale with an order higher than $O(h^d)$. Evaluating each part of a generic upwinding term like following

$$\frac{1}{2}\int_S (\min_{x\in I_{i+\frac{1}{2}}} H_1 - |\min_{x\in I_{i+\frac{1}{2}}} H_1|)\,[\phi]\left(x_{i+\frac{1}{2}}, y\right)\, dS$$

it is possible to see that H_1 or H_2, being derivatives of the hamiltonian, when $h \to 0$, tend to a finite value, the jump of the variable $[\phi]$ depends on the reconstruction done while the integration itself makes the term being proportional to $O(h^{d-1})$. This implies that, in order to satisfy the consistency condition, it is necessary to have a reconstruction of the jump at least of the following order:

$$[\phi] \approx O(h^2)$$

in this way the overall upwinding term scales with $O(h^{d+1})$ and the scheme is consistent.

4.3.2 Temporal discretization

In order to having a fully discrete scheme of the transport equation (eq. 4.1) the temporal derivative needs to be approximated. Since the spatial term of the equation 4.1 is discretized using an high order method, in particular a third order scheme, than also in time there is the need to apply an high order discretization scheme to achieve stability and accurancy of the results.

Therefore, a third order Runge-Kutta was used. The Runge-Kutta methods are a family of implicit and explixit iterative method used to approximate solutions of Ordinary Differential Equations. After having discretized the hamiltonian term, the fully discrete equation 4.1 reads

$$U_t = L\,(U) \tag{4.19}$$

where $L\,(U)$ is the numerical hamiltonian taken to the right hand side of the equation.

The classical third order Runge-Kutta method is defined with the following Butcher

tableaux

$$\begin{array}{c|ccc} 0 & & & \\ \frac{1}{2} & \frac{1}{2} & & \\ 1 & -1 & 2 & \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}$$

Therefore the intermediate steps to advance of a complete temporal step are

$$\begin{cases} U^1 = U^n + \frac{\Delta t}{2} L\left(U^n\right) \\ U^2 = U^n - \Delta t L\left(U^n\right) + 2\Delta t L\left(U^1\right) \\ U^{n+1} = U^n + \frac{\Delta t}{6} L\left(U^n\right) + \frac{2\Delta t}{3} L\left(U^1\right) + \frac{\Delta t}{6} L\left(U^2\right) \end{cases} \tag{4.20}$$

where the superscripts n stands for the initial data of that time step (so $L\left(U^n\right)$ is the numerical hamiltonian with the data at time n), superscripts 1 is the first intermediate step and it corresponds to the guessing at time $n + \frac{1}{2}$, superscripts 2 is the second guessing for time $n+1$ and then the final step of the algorithm is the correction for time $n+1$ that gives the final result. Obviuosly, the time step Δt is choosen accordingly to the CFL condition since the one adopted is an explicit time discretization and therefore conditionally stable.

4.4 Results

In this section the results of the reconstruction procedure alone and applyed together with the transport of the levelset function will be presented. Also the order of convergence was taken into account since it was crutial to develop an high order methodology thus lowering the dissipation that usually occours when low order method are employed.

The calculation of the order of convergence was performed with different norms of the error. The first used is the infinity norm which is defined as

$$L_\infty = \max|\mathbf{e}| \tag{4.21}$$

where $\mathbf{e}$ is the vector containing the absolute value of the error for each cell. Then also the norm 1 ws used, which is written as

$$L_1 = \sum_{i=1}^{N} e_i * V_i \tag{4.22}$$

where e_i is the i-th component of the error vector, V_i is the volume of the i-th cell and N is the cell's number. The last norm used for this calculation is the norm 2 which reads

$$L_2 = \sqrt{\sum_{i=1}^{N} \left(e_i * V_i\right)^2} \tag{4.23}$$

which is the classical euclidean norm of a vector but with every components multiplyed by the value of its cell's volume.

It is important to highlight that, for the test cases that will be presented here, the boundary condition applyed are of Dirichlet type. In fact, knowing at each time step the cell-average value of the levelset function, the exact signed distance function was used for the first border cells.

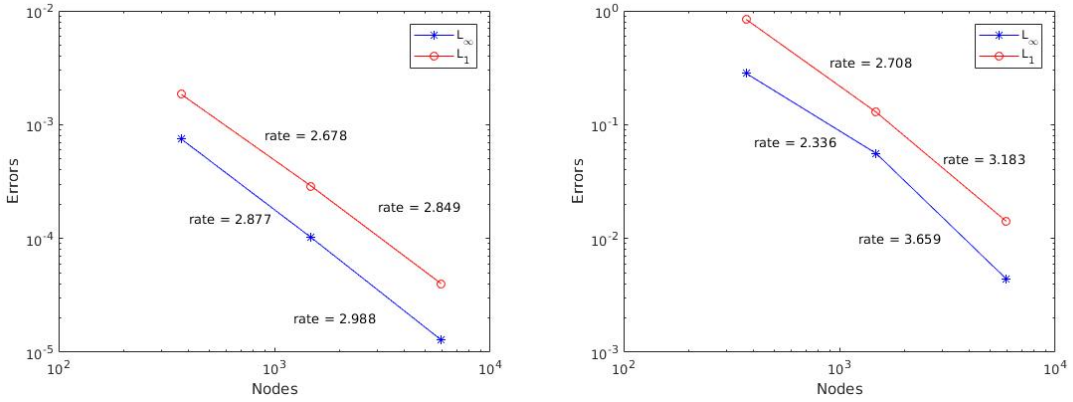
4.4.1 Test of the reconstruction procedure

In order to test the reconstruction procedure and the resulting polynomial for approximating the value of the function in every point of the cell, an analytical function was adopted. In particular, the following function was used

$$\sin(x \cdot y)$$

therefore, on a cartesian (uniform and adaptive) grid, knowing the coordinates of the cell's center, the function value was assigned at each node.

Then, the procedure described in the previous section was applied and the value of the function was reconstructed at each cell's interface. The order of convergence of the method for both uniform and adaptive grid is depicted in figure 4.4.



(a) Uniform grid.

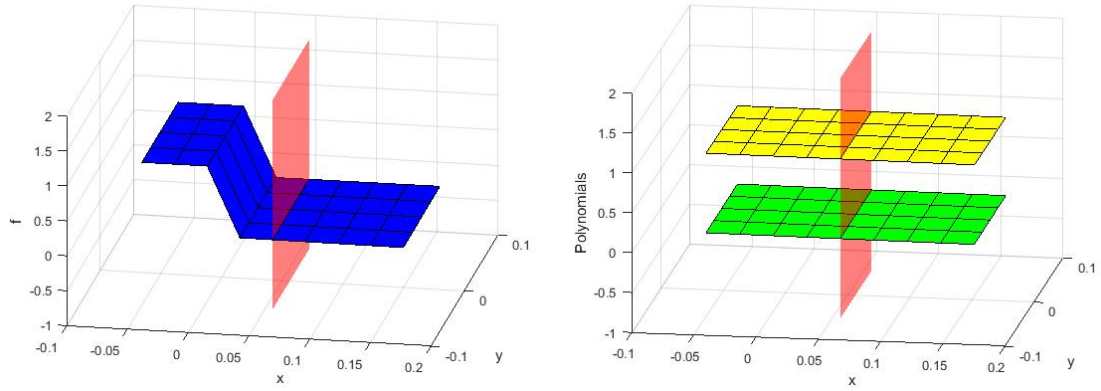
(b) Adaptive grid.

Figure 4.3: Order of convergence of the reconstruction procedure.

It is important to notice that, in order to calculate the order of convergence, the border cells were not considered since, for construction, their order is lower. From picture 4.4 it is clear that the reconstruction procedure works as expected giving an order of convergency which, asymptotically, tends to 3 both for the case of uniform and adaptive grid.

Moreover, also the smoothness indicators were tested by introducing a different analytical function, the $cos(x \cdot y)$, in a part of the domain that acted as a discontinuity. The weighting part of the procedure worked as expected lowering the global order of convergence but minimizing the constribute from the stencil that were interested by the kink.

In two subfigures, the red plane demarks the border of two adjacent cells. In subfigure 4.4a the two jointed analytical functions used to create a discontinuity in the discrete domain are depicted in blue. In subfigure 4.4b, the polynomials of the two cells calculated with the reconstruction procedure are shown; the yellow plane is from the cell containing the discontinuity, while the green one is from the other. As it is clear, no oscillations are visible in the reconstruction polynomials and the order of the method is lowered as expected (polynomials are planes and not parables).



(a) Function with a discontinuity.

(b) Approximant polynomials.

Figure 4.4: Discontinous function and reconstruction polynomials.

4.4.2 Test of the transport procedure

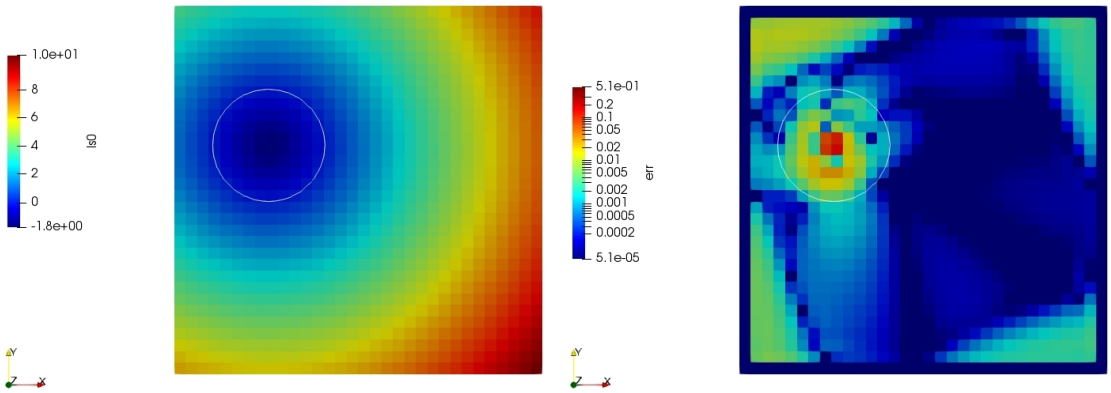
After having tested the reconstruction procedure and being sure that that part work properly, it was possible to test the discrete model for transporting the levelset value in the domain. The simulation performed was the rigid rotation of a cirle, both with its center inside and outside the computational domain. This chioce was done since the center of the circumference is a singular point, therefore the WENO procedure correctly reduce the order of the scheme in the cells interested by the phenomena and to calculate the right order of the model it was necessary to use a function which do not present kinks.

Rigid rotation of a circle on uniform grid

The first test case perfomed is the rigid rotation of a cirle, and the distance field related to it, positioned in a square domain. The center of rotation is located in the middle of the computational domain and not in the center of the moving circumference.

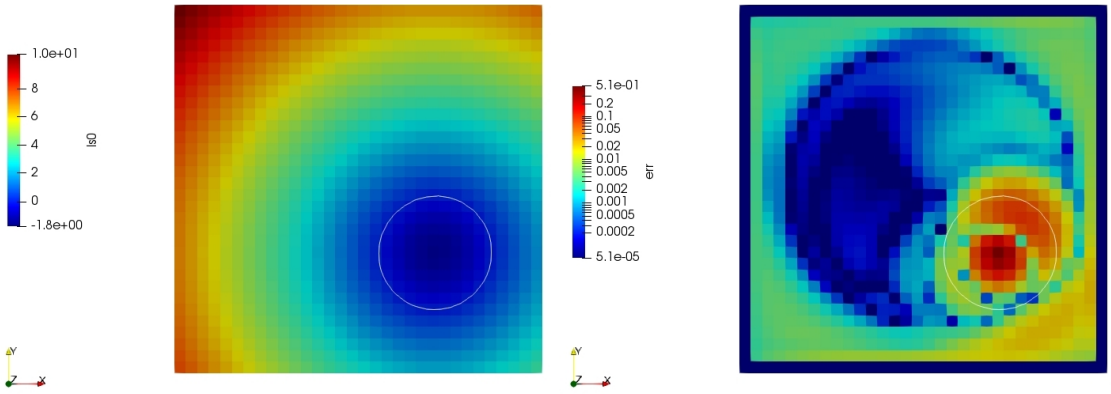
The test was performed with cartesian meshes with different number cells, in particular 32×32, 64×64 and 128×128 grid points along the x and y direction, respectively. In figure 4.13, 4.14 and 4.15 the results are reported. In particular a white circle represents the zero isocontour ate the different time step plotted. On the right part of the images the error is reported while on the left the corresponding level set field is depicted.

The global error lowers when the mesh is refined, this is clear from the color map along the error figures for different grid. However, the center of the circumference doesn't show the correct order of convergency and even if it diminuishes, the order of magnitude remains almost the same for that point. Moreover, having an high order method, the diffusivity is minimum, making the error not to spread throughout the domain in few steps. In fact, after a complete revolution of the body, there still are region of domain with an error almost equal to the one at the beginning of the transport procedure.



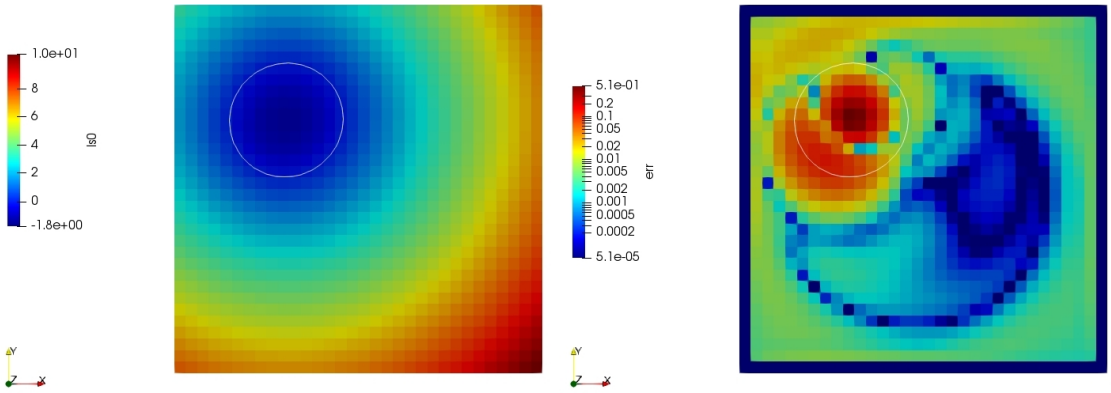
(a) 10 time step.

(b) 10 time step.



(c) Half revolution.

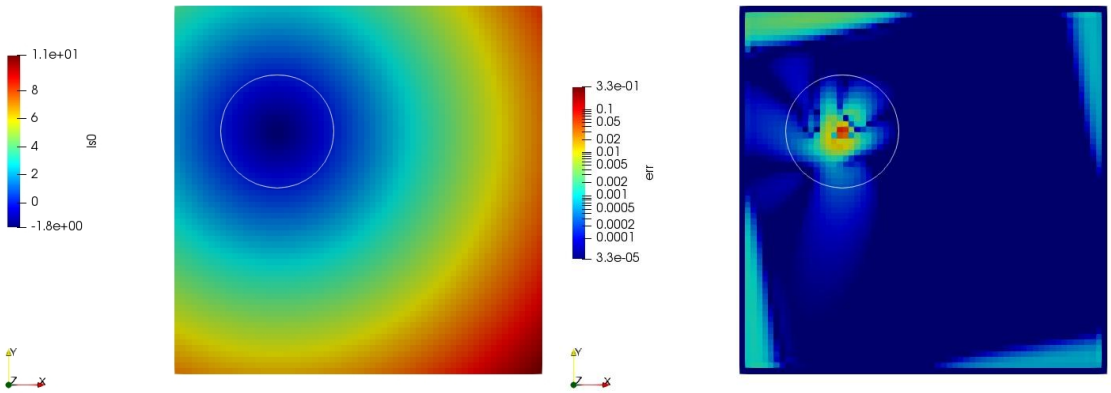
(d) Half revolution.



(e) Full revolution.

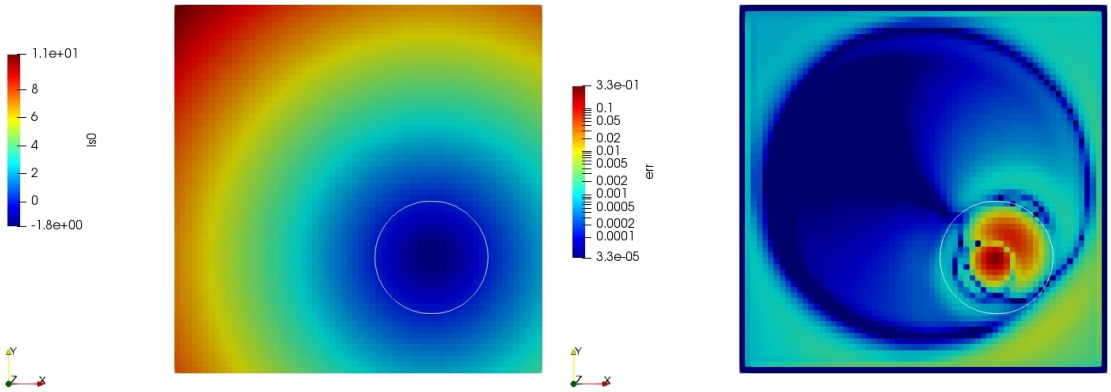
(f) Full revolution.

Figure 4.5: Evolution through time of a rotating cirle level set (left pictures) and its error (right pictures) with 32×32 grid.



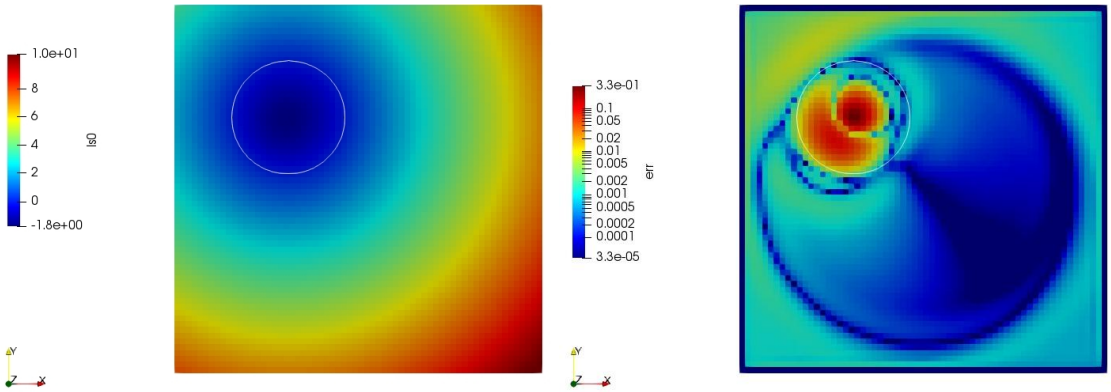
(a) 10 time step.

(b) 10 time step.



(c) Half revolution.

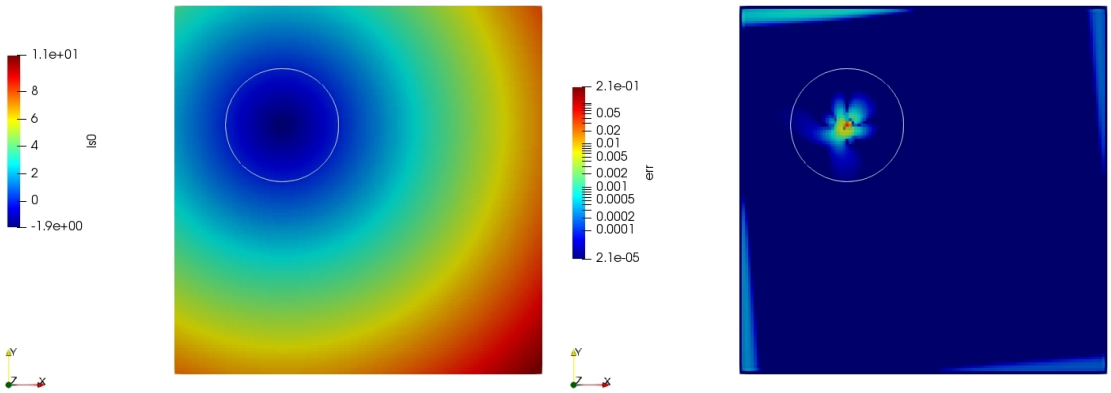
(d) Half revolution.



(e) Full revolution.

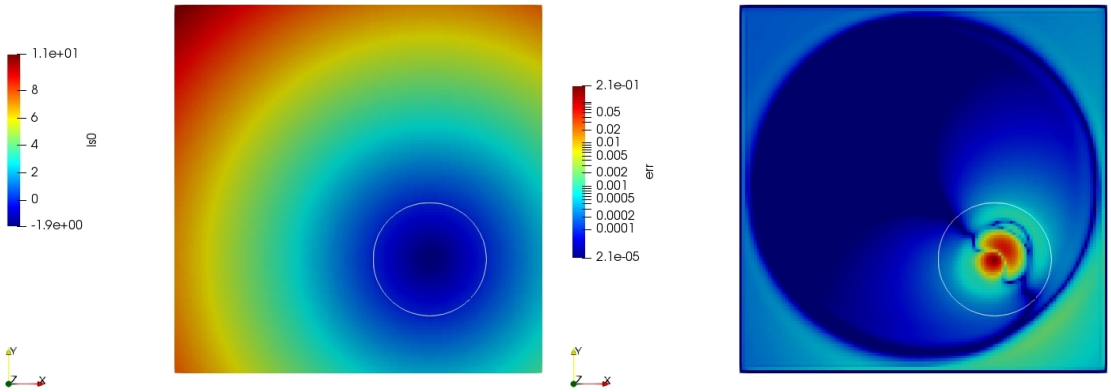
(f) Full revolution.

Figure 4.6: Evolution through time of a rotating cirle level set (left pictures) and its error (right pictures) with 64×64 grid.



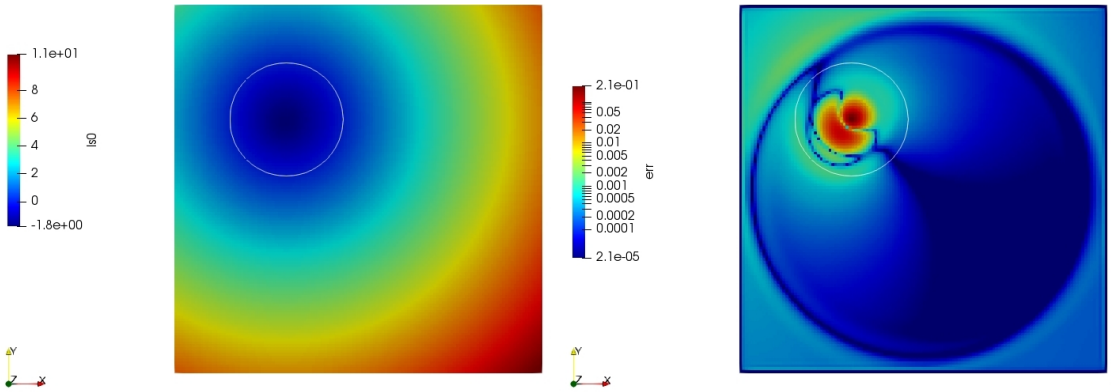
(a) 10 time step.

(b) 10 time step.



(c) Half revolution.

(d) Half revolution.



(e) Full revolution.

(f) Full revolution.

Figure 4.7: Evolution through time of a rotating cirle level set (left pictures) and its error (right pictures) with 128×128 grid.

In figure 4.8 the order of the proposed methodology is shown. The global order of convergency, calculated using the euclidean norm of the error's vector, is above the second order while the local order of convergency, calculated using the infinitive norm of the error, is lower than one. This is due to the fact that inside the domain a singular point, namely

the center of the circular level set, is included. In this point, as previously explained, the methodology lower its order thus avoiding oscillation that could lead to instability in the transport scheme.

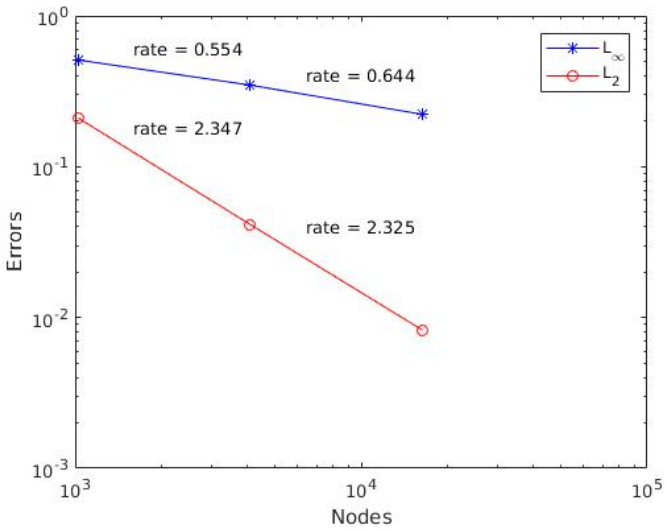
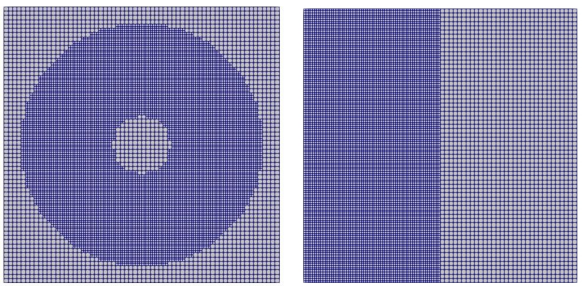


Figure 4.8: Order of convergency of the transport procedure for a circle with its center inside the computational domain.

Rigid rotation of a circle on adaptive grid

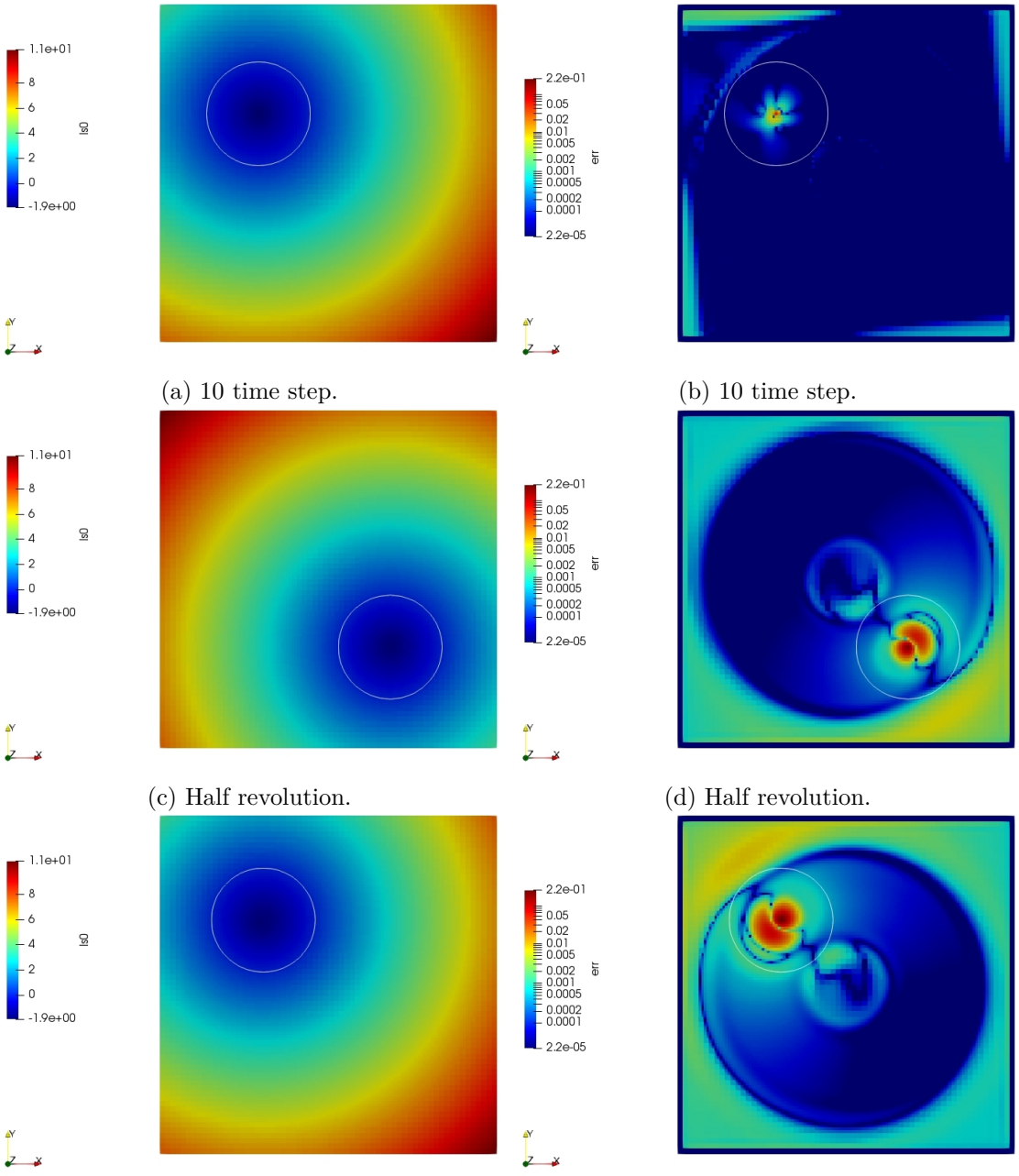
The proposed methodology works also on cartesian adaptive grid. Two test cases were performed using the meshes portraied in picture 4.9, where there is only a level of refinement which means that the little cells are obtained from splitting the mother cell into four equal parts. In the first case, the region that was choosen to be refined is the one where the cirlce is moving, therefore having an higher precision and resolution near the interface of the body. In the second case, half of the mesh is refined in order to see the functionality of the method when the interface cross a jump in the grid.



(a) (b)

Figure 4.9: Kinds of adaptive grids employed.

As in the previous case, the higher error is located in the center of the cirlce and in the other parts of the mesh the coarser cells present a larger error.



(a) 10 time step.

(b) 10 time step.

(c) Half revolution.

(d) Half revolution.

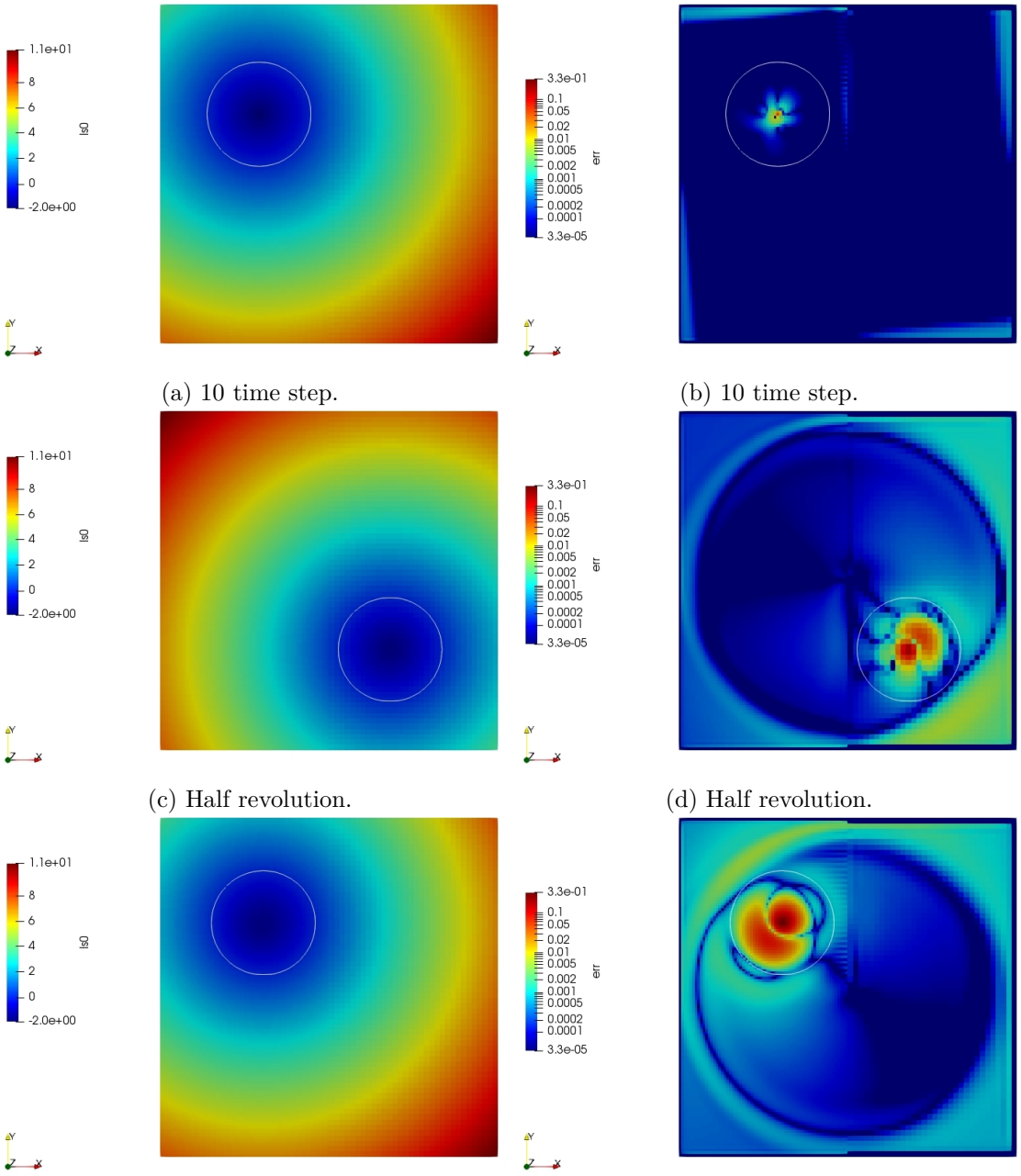
(e) Full revolution.

(f) Full revolution.

Figure 4.10: Evolution through time of a rotating cirle level set (left pictures) and its error (right pictures) with the (a) adaptive grid in figure 4.9.

Results are shown in figure 4.10 for the first case and in picture 4.11 for the second

case. In the latter image it is possible to see that the method handles well the crossing of a jump in the mesh, having the zero isocontour not particularly influenced by it. However, the error is slightly higher due to the fact that the center of the cirle was also transported in a coarser mesh, thus accumuling a major mistake.



(a) 10 time step.

(b) 10 time step.

(c) Half revolution.

(d) Half revolution.

(e) Full revolution.

(f) Full revolution.

Figure 4.11: Evolution through time of a rotating cirle level set (left pictures) and its error (right pictures) with the (b) adaptive grid in figure 4.9.

Rigid rotation of a circle on uniform mesh with center outside the domain

The last test that was performed is a rotation of a cirle whose center is outside the computational domain. Therefore, what it can be seen is the rotation of the distance field far away from its zero isocontours. Being that distant, the curvature is low and also the error is lower in magnitude.

The order of convergency for this case are draw in figure 4.12 and for global error they behave as expected for a third-order accurate scheme.

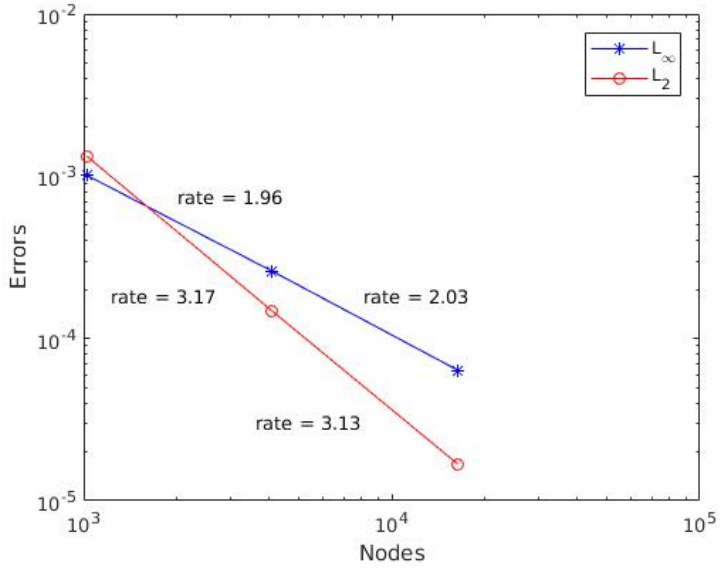
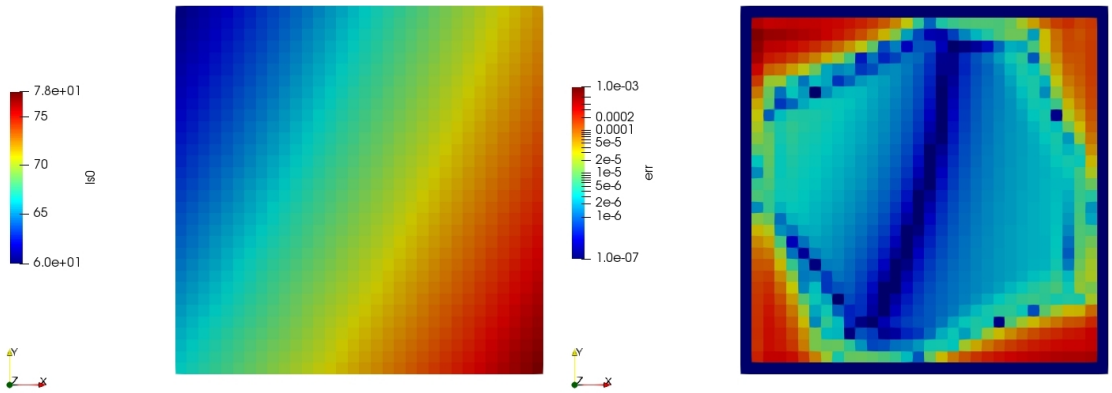


Figure 4.12: Order of convergency of the transport procedure for a circle with its center outside the computational domain.

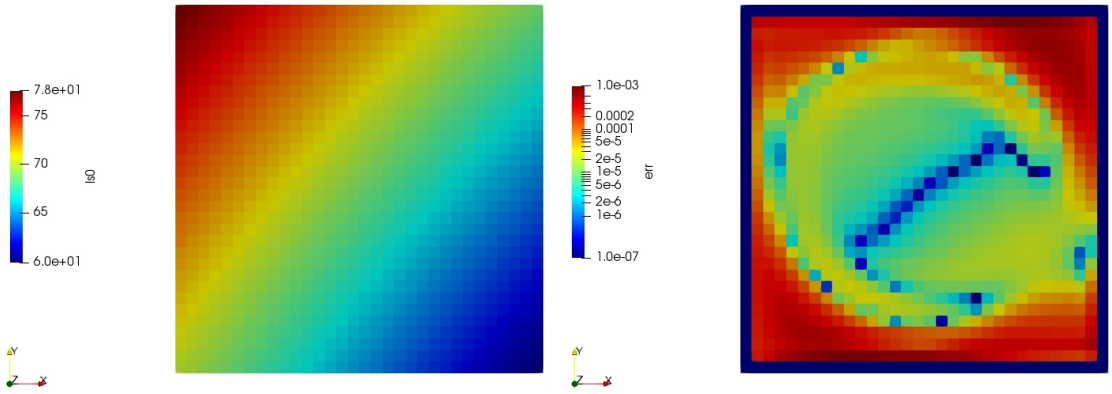
However, the local order shows that some points of the domain present a lower order of convergency, in particular equal to 2. These points are located close to the border cells and are likely due to the boundary condition adopted. In fact, using a Dirichlet boundary condition at each time step for all the boundary cell can disturb the interior domain since half of the border has a velocity which is entering (transporting correct information) and the other half the velocity transports outside information from the inner domain. Especially at the corner, one of the two edges has an entering velcoity while the other an exiting one. Therefore, the information entering from the latter edge could collide with the imposed value at other border giving an error.

This problem linked with the boundary condition seems to be confirmed by the fact that the error start close to the boundary and it is transported throught the steps, augmenting outside the centrale region. This latter part of the domain has an error that tends to increase but only in the part close to the border while in the center it stays almost constat due to the low diffusivity of the numerical scheme.



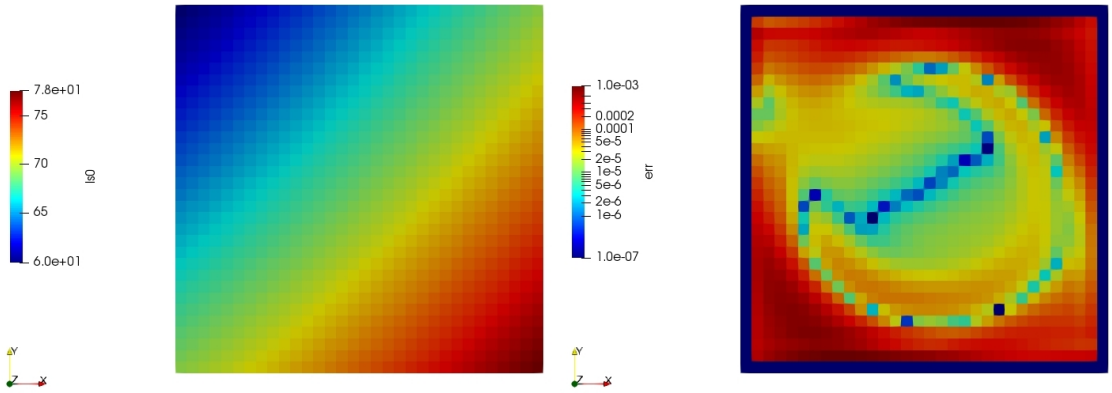
(a) 10 time step.

(b) 10 time step.



(c) Half revolution.

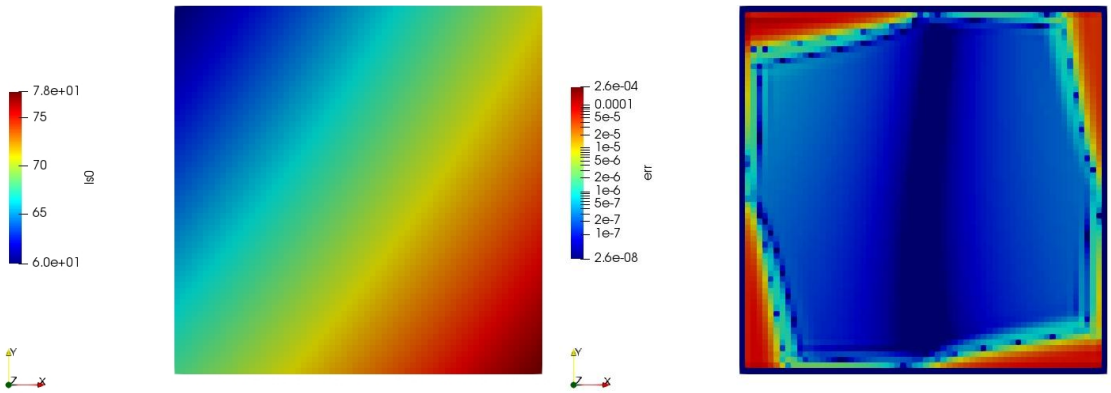
(d) Half revolution.



(e) Full revolution.

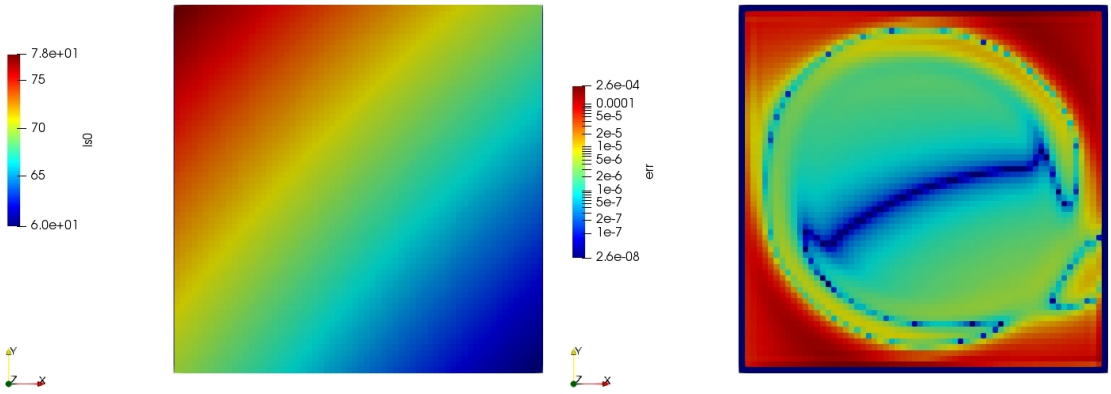
(f) Full revolution.

Figure 4.13: Evolution through time of a rotating cirle level set (left pictures) and its error (right pictures) with 32×32 grid.



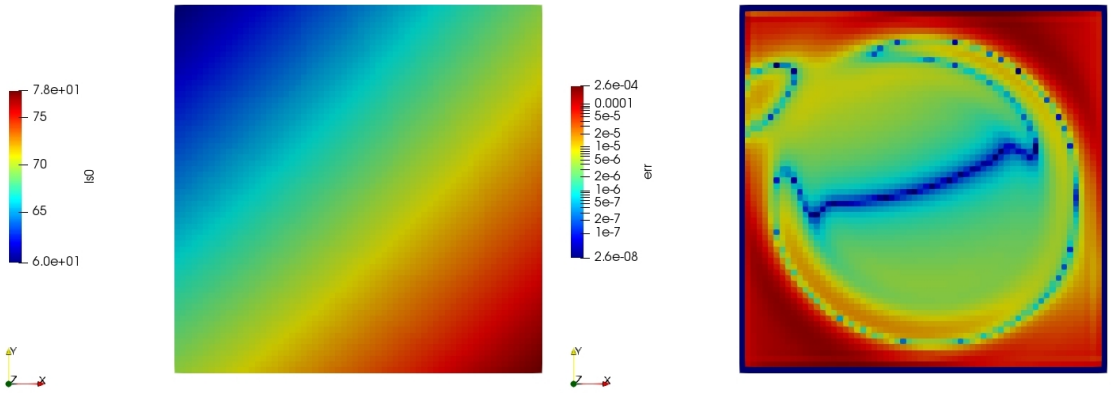
(a) 10 time step.

(b) 10 time step.



(c) Half revolution.

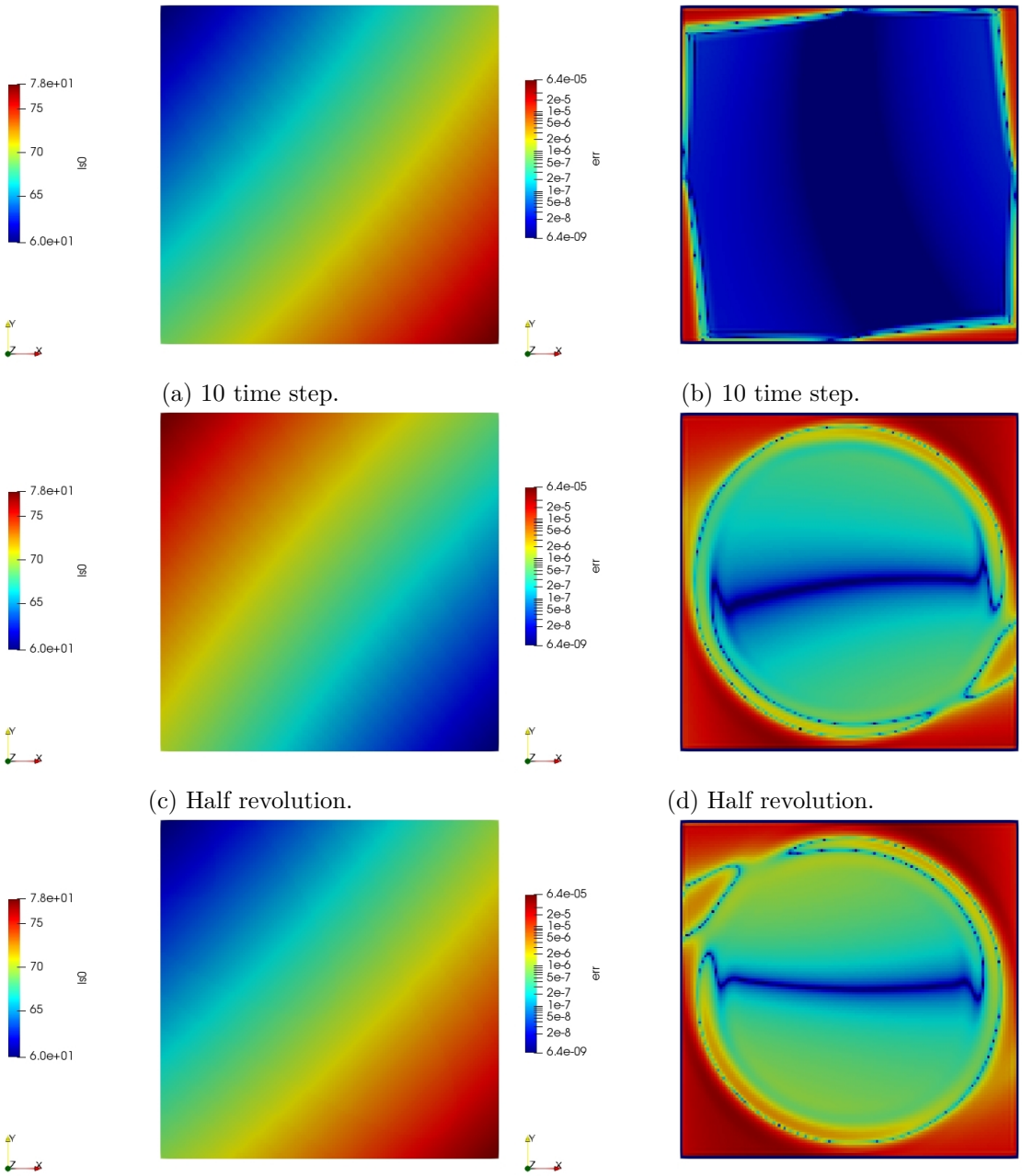
(d) Half revolution.



(e) Full revolution.

(f) Full revolution.

Figure 4.14: Evolution through time of a rotating cirle level set (left pictures) and its error (right pictures) with 64×64 grid.



(a) 10 time step.

(b) 10 time step.

(c) Half revolution.

(d) Half revolution.

(e) Full revolution.

(f) Full revolution.

Figure 4.15: Evolution through time of a rotating cirle level set (left pictures) and its error (right pictures) with 128×128 grid.

Chapter 5

Integration of the level set tecnique with CFD simulation

The Level set methods developed in the previous chapters are then coupled with the Immersed Boundary Computational Fluid Dynamic solver developed by *Optimad Engineering srl.*

This latter, it is a simple globally second order scheme for compressible inviscid flows inspired by a ghost-cell approach, where the fluid solution, away from the solid interface, is calculated using a finite-volume method on an approximate Riemann solver, while, close to the body, a specific Riemann solver is settled up to take into account the relevant boundary condition. More details can be found in [9].

5.1 Transonic airfoil test case

The test case used for showing the ability of the level set method for handling a moving body in a fluid dynamic simulation is a two-dimensional transonic flow past an oscillating NACA0012 airfoil. The oscillation follows a sinusoidal law and it takes place around a fixed point, located at 25% of the chord of the airfoil, with a frequency $f = 50\,Hz$. The flow has an infinity Mach number equal to 0.6, while the airfoil average AoA (Angle of Attack) is $\alpha = 2.89°$ and the amplitude of the angular excursion is $2.41°$.

The computational domain dimensions are based on the airfoil chord, denoted with c. The profile is positioned in order to have a distance of $10c$ both upwards and downward from the borders of the domain and $10c$ and $20c$, respectively, from the upwind and downwind borders. The simulation was conducted on 64 parallel processors and it employs $4.8^2 \times 10^6$ cells, equally distributed on x and y axis. A uniform initial condition was adopted and the simulation lasted until the hysteresis cycle of aerodynamic coefficient is periodic (after about two oscillation cycles).

5.1.1 Results of the test case

The oscillation of the airfoil was simulated successfully, giving back the different aerodynamic fields during the profile motion. Moreover, when the angle of attack of the airfoil

increases, also the velocity of the fluid on the suction side of the profile becomes higher. This transonic flow creates shocks on that region of the geometry, but the code is able to effectively solving this flow regime which can be difficult to be accurately predicted.

In pictures 5.1 and 5.2, the pressure and Mach fields of the flow around the profile are depicted for different time step, when the profile has different angle of attack.

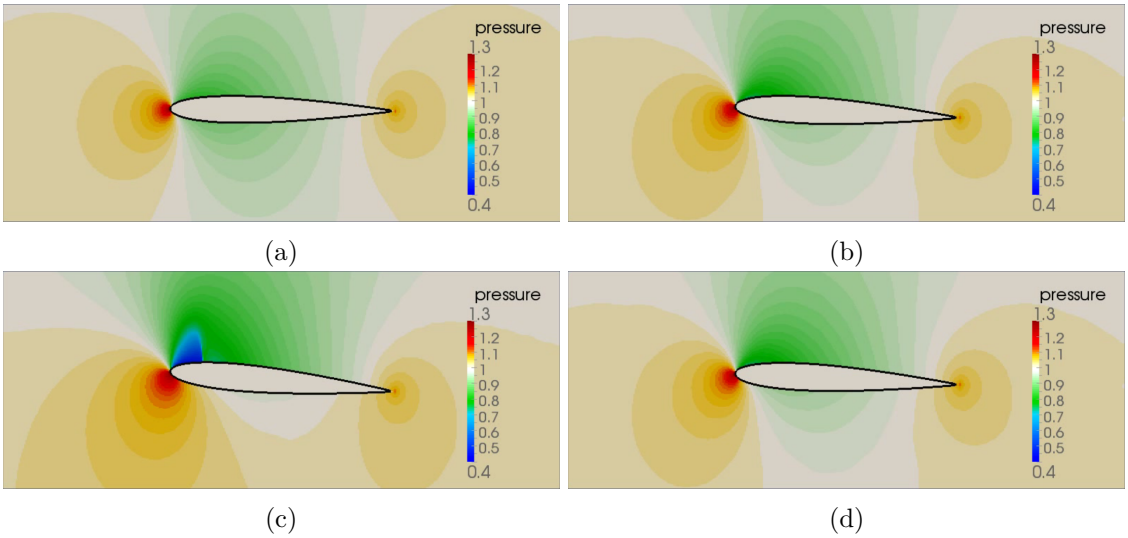


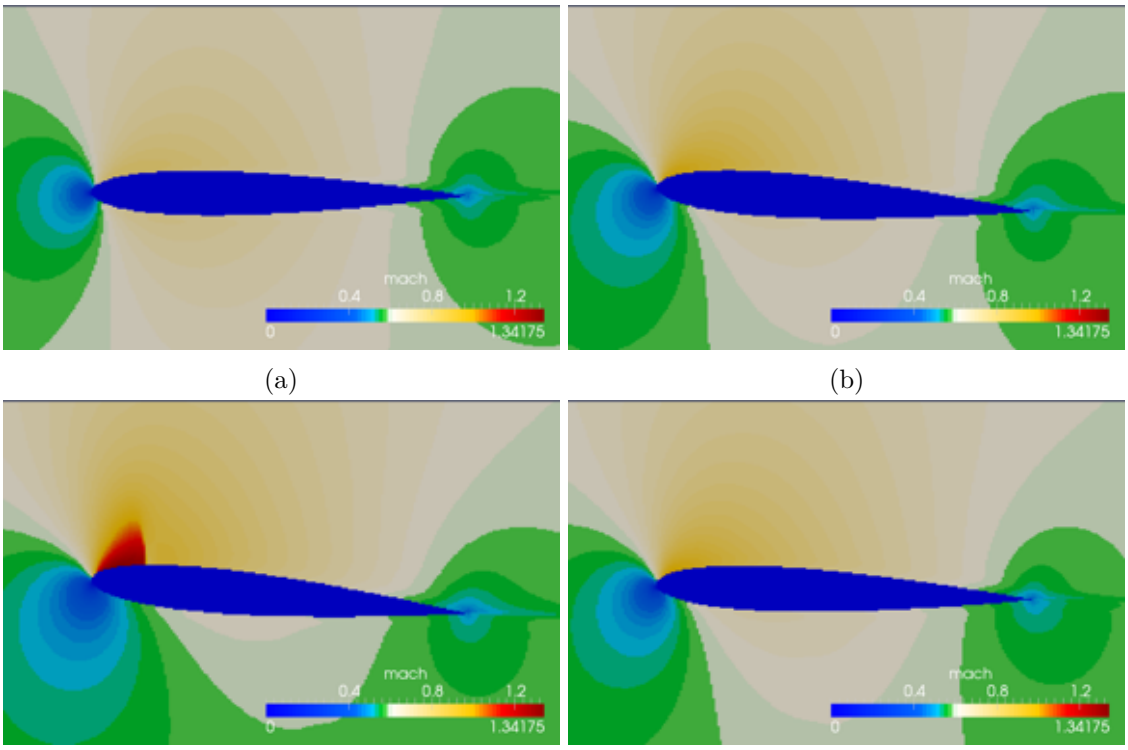
Figure 5.1: Evolution through time of the pressure field for an oscillating airfoil.

The NACA0012 profile is symmetrical, therefore at zero angle of attack it should not produce lift. However, in this simulation, the minimum AoA used was 0.5° which gives a slight difference in the pressure field between the suction and pressure side of the airfoil, as it can be seen in the 5.1a subpicture. Then, the 5.1c plot shows a shock on the suction side of the profile, which is due to the compression of the flow after that it reached a supersonic regime.

Moreover, the lift coefficient C_L is calculated for different reduced frequency ($k_0 = 0.04 \div 2.56$) at which the airfoil oscillates. The result is showed in figure 5.3, where it is clear that even these little oscillations, with a limit cycle of 2°, influence the hysteresis cycles of aerodynamic coefficients, as well as the shock position on the suction side of the airfoil. Clearly, the result are very different from the one that the linearized theory would predict and this is the reason why this kind of simulations are important. In fact, setting up a real flight test to collect these data would be extreamely expensive, while being able to use a computer simulation it reduces drastically the costs.

5.1.2 Validation of the test case

In order to validate the results obtained with the proposed methodology, a comparison with some experimental results was done. In particular, it was verified that the aerodynamic coefficient, related to the normal component of the resulting force acting on the profile, calculated with the results coming from the CFD simulation was compatible with the



(a) (b)

(c) (d)

Figure 5.2: Evolution through time of the Mach field for an oscillating airfoil.

experimental data collected in a NATO report on unsteady aerodynamics [14].

Figure 5.4 depicts this comparison and it demonstrates that the proposed methodology could potentially become an useful tool for studying, accurately, these unsteady phenomena, since the obtained C_n hysteresis cycle is actually close to the experimental one.

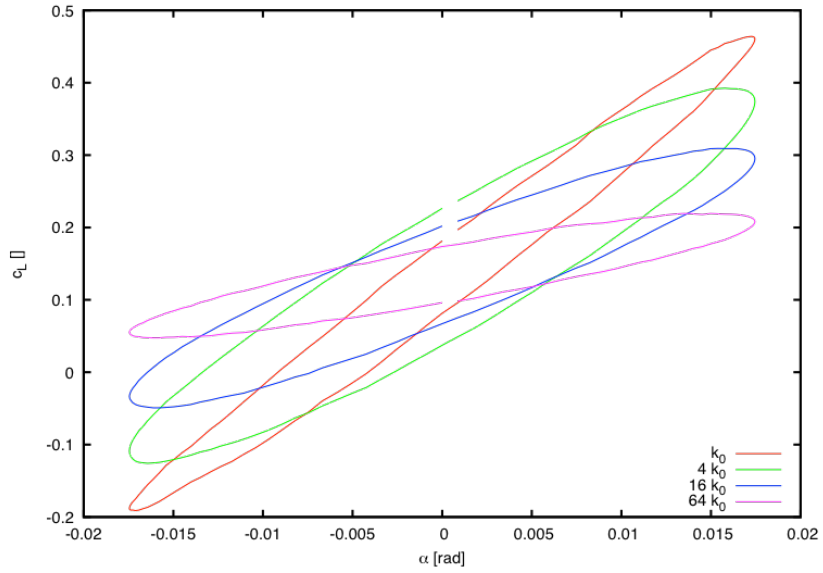


Figure 5.3: C_L-α hysteresis cycle due to airfoil oscillations.

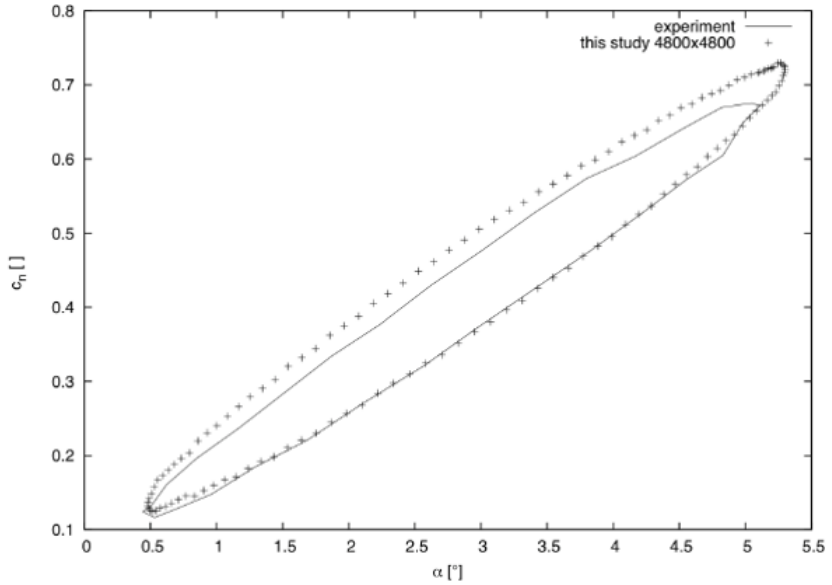


Figure 5.4: C_n-α hysteresis cycle due to airfoil oscillations for experimental data and simulation results.

Chapter 6

Conclusions

When a fluid flow needs to be simulated around a complex or even morphing geometry, one of the main issue is the discretization of space around the geometry. In the last decades, many solutions have been studied to overcome this issue, from the overlapping of multiple simple meshes to unstructured grids with element of various shapes. However, the processes to generate these meshes are time-consuming and in many situations they can not be automated. A breakthrough has been made with the introduction of the Immersed Boundary techniques, which allow to use regular cartesian grids even for bodies with complex shapes. On top of this base, methodologies able to locally refine the meshes where impleneted, resulting in the generation of the so called Octree grids. These latter are non-uniform and non-conformal cartesian meshes where the cells close to the body are recoursively splitted in sub-cells, thus permitting to have an higher accuracy in these points.

Obviously, the procedure of the IB requires to always know the position of the interface of the body. The Level set theory is one of the framework to retrieve this information. Generally speaking, a Levelset framework is made of a collection of numerical methods that allow for a correct handling of the geometry described by the Levelset. In particular, the body is described with a scalar field that express the distance of a point from the interface. Therefore, the CFD solver is able to correctly apply the BCs in the right nodes by knowing the level set at every time of the computation.

The preset thesis has studied two of the main issues when a level set framework has to be implemented: the assignment of a correct distance value in all the nodes of the grid and the implementation of an high-order numerical scheme for transport the level set through time.

The former problem appears when the level set is numerically computed through time. Substantially, errors could arise, leading to wrong body shape and wrong application of BCs for the immersed boundaries. In these situations, a restoring technique to recover the signed distance function is needed. Efficient reinitialization algorithms exists but they are applied to recover the signed distance only in the vicinity of the interface, in the so

called narrowbad[1], otherwise the computational cost would be too high. Thus, the role of the propagation technique is to spread these values in the rest of the domain, giving a complete distance field. In this work, two main methodologies have been implemented to diffuse the signed distance field from the narroband to the rest of the computational region.

The first algorithm was developed on the base of the *Heat method* introduced by Crane *et. al.* in [5]. The basic structure of the new procedure is quite close to the one adopted in the forementioned paper. However, the equations used and especially the boundary conditions employed for the helliptic problem were different. Two kind of BCs have been tested for the final Poisson problem of the Crane'scheme: a Neumann BC and a Dirichlet BC. The former involves the normalized gradient calculated in a prevoius step of the procedure, the latter uses an optimization algorithm to impose the boundary condition in a way that the resulting distance field would satisfy the Eikonal equation. Moreover, a fixed-point iterative mechanism was added to the final algorithm to improve the results.

The second solution for the propagation's issue is the develpment of a Newton iterative procedure. The level set field is initialized using the Heat equation but then, in order to get the final distance field, Newton's iterations to solve the Eikonal equation are performed. The choice of ϕ field initialization was taken on the base of the good distance approximation obtained by solving the heat equation. It has to be noted that this is a critical step for assuring the convergence of the whole method. The result of this routine were analyzed and compared with the one of the previous case. They are qualitatively better, showing a lower error, and the outcome is obtained with fewer iterations.

The latter issue exploited is the implementation of an high-order numerical scheme able to transport the level set function during time. The signed distance values at interface of the body has to remain well behaved during the whole simulation. In this way, it is possibile to repair eventual damage with a reinitialization procedure. For that reason, an high accuracy technique is needed, thus reducing the numerical dissipation of the methodology. Therefore, by using a refined grid near the zero-isocontour, combined with a low-error scheme, there is the confidence that restoring can take place. The choosen technique to transport the level set is close the one proposed by Zhu and Qiu in [28]. However, both the WENO reconstruction procedure and the transport mechanism were different. In fact, a Compact WENO reconstruction procedure, developed by Semplice in [21], for cartesian adaptive grid was employed and, in the transport model, the only variable to evolve through time was the level set itself. The test choosen to check the correctness of the implemented procedure is a rigid rotation of a simple circle. The results showed the desired order of accuracy and the scheme worked well even with Quadtree grids. Furthermore, even when the interface of the body passed across a jump in the mesh the level set field remains well behaved.

[1]The collection of cells crossed and immediately adjacent to the body's interface.

At last, a final test-case was carried out in order to match the developed level set methods with the proprietary CFD solver created by *Optimad Engineering Srl*. The coupling shows the potentiality of the level set methods applyed within the Immersed Boundary technique framework. Together they can easily allow the simulation of complex bodies changing their shape during the computation. The whole process could be automated with low effort and it could work in the most diverse applications. In any case, the possibilities opened by this approach are just being spotted and further work can be done to sharpen these features.

In particular, forthcoming research could be devoted to compleately extend the use of the proposed propagation techniques to non-uniform and non-conformal grids. In addiction, the transport algorithm could be improved to reach an higher accuracy order and the formula of the stencils smoothness indicators could be further analyzed in order to enhancing the non-oscillatory property of the the WENO procedure.

Bibliography

[1] *Numerical Methods for Conservation Laws.* Birkhauser, 1992.

[2] J. D. Anderson. *Computational Fluid Dynamics.* McGraw-Hill, 1995.

[3] S. C. Chapra and R. P. Canale. *Numerical Methods for Engineers.* McGraw-Hill, sixth edition edition, 1998.

[4] D. L. Chopp. Computing minimal surfaces via level set curvature flow. *Journal of computational physics*, 106(1):77–91, 1993.

[5] K. Crane, C. Weischedel, and M. Wardetzky. Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Transactions on Graphics (TOG)*, 32(5):152, 2013.

[6] M. D. de Tullio, P. De Palma, G. Iaccarino, G. Pascazio, and M. Napolitano. An immersed boundary method for compressible flows using local grid refinement. *Journal of Computational Physics*, 225(2):2098–2117, 2007.

[7] S. O. R. Fedkiw and S. Osher. Level set methods and dynamic implicit surfaces. *Surfaces*, 44:77, 2002.

[8] J. H. Ferziger and M. Peric. *Computational methods for Fluid Dynamics.* Springer, third edition edition, 2002.

[9] Y. Gorsse, A. Iollo, H. Telib, and L. Weynans. A simple second order cartesian scheme for compressible euler flows. *Journal of Computational Physics*, 231(23):7780–7794, 2012.

[10] A. Harten, B. Engquist, S. Osher, and S. R. Chakravarthy. Uniformly high order accurate essentially non-oscillatory schemes, iii. *Journal of computational physics*, 71(2):231–303, 1987.

[11] M. Hinatsu and J. Ferziger. Numerical computation of unsteady incompressible flow in complex geometry using a composite multigrid technique. *International Journal for Numerical Methods in Fluids*, 13(8):971–997, 1991.

[12] G.-S. Jiang and D. Peng. Weighted eno schemes for hamilton–jacobi equations. *SIAM Journal on Scientific computing*, 21(6):2126–2143, 2000.

[13] O. Kolb. On the full and global accuracy of a compact third order weno scheme. *SIAM Journal on Numerical Analysis*, 52(5):2335–2355, 2014.

[14] R. Landon and S. Davis. Compendium of unsteady aerodynamic measurements. *AGARD Report*, 702, 1982.

[15] X.-D. Liu, S. Osher, and T. Chan. Weighted essentially non-oscillatory schemes. *Journal of computational physics*, 115(1):200–212, 1994.

[16] R. Mittal and G. Iaccarino. Immersed boundary methods. *Annu. Rev. Fluid Mech.*, 37:239–261, 2005.

[17] S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. *Journal of computational physics*, 79(1):12–49, 1988.

[18] S. Osher and C.-W. Shu. High-order essentially nonoscillatory schemes for hamilton–jacobi equations. *SIAM Journal on numerical analysis*, 28(4):907–922, 1991.

[19] C. S. Peskin. Flow patterns around heart valves: a numerical method. *Journal of computational physics*, 10(2):252–271, 1972.

[20] L. C. Polymenakos, D. P. Bertsekas, and J. N. Tsitsiklis. Implementation of efficient algorithms for globally optimal trajectories. *IEEE Transactions on Automatic Control*, 43(2):278–283, 1998.

[21] M. Semplice, A. Coco, and G. Russo. Adaptive mesh refinement for hyperbolic systems based on third-order compact weno reconstruction. *Journal of Scientific Computing*, 66(2):692–724, 2016.

[22] C. W. Shu. Lecture1: Finite volume weno schemes. Division of Applied Mathematics. Brown University.

[23] M. Sussman, P. Smereka, and S. Osher. A level set approach for computing solutions to incompressible two-phase flow. *Journal of Computational physics*, 114(1):146–159, 1994.

[24] J. N. Tsitsiklis. Efficient algorithms for globally optimal trajectories. *IEEE Transactions on Automatic Control*, 40(9):1528–1538, 1995.

[25] H. Udaykumar, R. Mittal, P. Rampunggoon, and A. Khanna. A sharp interface cartesian grid method for simulating flows with complex moving boundaries. *Journal of Computational Physics*, 174(1):345–380, 2001.

[26] H. Udaykumar, R. Mittal, and W. Shyy. Computation of solid–liquid phase fronts in the sharp interface limit on fixed grids. *Journal of computational physics*, 153(2):535–574, 1999.

[27] S. R. S. Varadhan. On the behavior of the fundamental solution of the heat equation with variable coefficients. *Communications on Pure and Applied Mathematics*, 20(2):431–455, 1967.

[28] J. Zhu and J. Qiu. Finite volume hermite weno schemes for solving the hamilton-jacobi equation. *Communications in Computational Physics*, 15(4):959–980, 2014.