Development of advanced immersed-boundary methods for multiphysics

POLITECNICO DI BARI

DOTTORATO DI RICERCA IN INGEGNERIA MECCANICA E GESTIONALE

XXVIII CICLO

Curriculum: Macchine a Fluido (SSD ING-IND/08) e Fluidodinamica (SSD ING-IND/06)

Development of Advanced Immersed-Boundary Methods for Multiphysics

Dario DE MARINIS

Relatori:
Prof. **M. NAPOLITANO**
Prof. **M. D. DE TULLIO**

Controrelatori:
Prof. **G. IACCARINO**
Prof. **B. FACCHINI**

Coordinatore:
Prof. **G. P. DEMELIO**

TRIENNIO ACCADEMICO 2013-2015

a mio padre

Abstract

The purpose of this work is to develop and test an accurate and efficient tool for computing three-dimensional complex flows past fixed or moving geometries, for a wide range of Reynolds and Mach numbers.

Firstly, an accurate and efficient Immersed-Boundary (IB) method, using a state-of-the-art Unsteady Reynolds-Averaged Navier–Stokes (URANS) parallel Cartesian solver, has been improved, by means of a new IB treatment, extended to three space dimensions, and validated versus several test cases of increasing complexity.

Then, a code for solving heat conduction (HC) equation that uses the same spatial discretization and time-marching scheme as the URANS solver has been developed and coupled with it to obtain an efficient tool for solving Conjugate-Heat-Transfer (CHT) problems: the Cartesian grid presents both fluid and solid zones: the URANS equations are solved at all fluid cells and the HC equation is solved at all solid cells; the two solutions are coupled by the interface conditions requiring that both the temperature and heat-flux be the same at all (fluid-solid) boundary points.

Finally, a surface-based structural solver that simulates the dynamics of deformable geometries, discretized by triangulated Lagrangian meshes, has been coupled with the basic IB-URANS method to provide an efficient tool for solving Fluid-Structure-Interaction (FSI) problems: the forces exerted by the fluid onto the solid surface are used to determine its motion, which is fed as a kinematic boundary condition for the flow, via an iterative procedure implemented within the dual time stepping procedure of the URANS solver.

The two coupled solvers have been validated versus CHT and FSI problems, such as the flow through an air cooled gas turbine cascade and the low Reynolds shear flow past a deformable sphere, proving to be promising research and development tools for industrial and medical applications.

Keywords: *RANS equations; Immersed Boundary; Conjugate Heat Transfer; Turbine Cooling, Fluid Structure Interaction.*

Contents

List of Figures

List of Tables

Introduction

Background

In Computational Fluid Dynamics (CFD), the mathematical description of a problem requires a discretization step, which allows one to transform the Navier–Stokes equations into an algebraic set of equations. To this purpose, the entire domain of interest is discretized into small elements forming a computational grid. Usually starting from a geometrical description of the boundaries (body and external domain), a surface grid is produced and used to generate a volume grid covering the whole fluid domain. One of the main difficulties in performing calculations for very complex geometries is the grid generation process that requires many hours if not days of work by a CFD specialist.

In recent years the Immersed-Boundary (IB) method has emerged as a very appealing approach for solving flows past very complex geometries, like those occurring in most industrial applications. Its main, very significant, feature is the use of a Cartesian grid embodying the complex boundaries of the flow domain that allows one to generate the computational grid within a few minutes, as well as to use the simple and efficient numerical methods developed in that framework. The effect of a stationary or moving boundary can be accounted for by introducing a distribution of fictitious forcing terms in the governing equations, such that the correct flow boundary conditions on the solid boundaries can be assigned, or by enforcing the wall boundary conditions effect over the cells close to the surface, as done in this work.

Over the past decades, several methods have been proposed, with various degrees of accuracy and complexity. The IB method was originally developed for incompressible flows (Peskin [9], Mohd-Yusof [10], Fadlun *et al.* [11], Iaccarino and Verzicco [12]), using non-uniform Cartesian grids to take advantage of simple numerical algorithms–an interesting review of the IB method and its applications is provided by Mittal and Iaccarino [13].

The CFD group at the Department of Mechanics, Mathematics and Management of

the Polytechnic of Bari has extended the IB method to the preconditioned compressible Unsteady Reynolds-Averaged Navier–Stokes (URANS) equations for solving complex viscous flows at any value of the Mach number and equipped it with a local grid refinement procedure to resolve boundary layers and regions with high flow gradients (De Palma *et al.* [14], de Tullio *et al.* [3]).

Overview of the Immersed-Boundary technique

The first example in literature of the IB method is due to Vieceli [15] that extended the Marker And Cell (MAC) method (Welch *et al.* [16], Harlow & Welch [17]) to include boundaries of arbitrary shape. The basic idea consisted of treating the fluid-boundary interface as a free-surface and to impose there pressure boundary conditions so that particles could move only along the tangent to the boundary line. This procedure led to an iteration between pressure and velocity fields until flow incompressibility and boundary impermeability were both satisfied. The method, referred to as ABMAC (Arbitrary Boundary MAC), was generalized in a successive paper (Vieceli [18]) to handle moving walls and, in this case, in addition to the pressure, also velocity boundary conditions were imposed at the interface. This technique allowed the treatment of walls moving with a prescribed law or moving as a consequence of the forces exerted by the fluid on the surface. Peskin [9],[19] reported at the beginning of the seventies simulations of the blood flow in the hearth/mitral-valve system assuming a very low Reynolds number and two-dimensional flow. Three-dimensional heart flows that included also the contractile and elastic nature of the boundary were considered successively by Peskin [20] and McQueen and Peskin [21]-[22]. In Peskin's formulation the fluid equations (incompressible Navier–Stokes equations) are solved on a uniform Cartesian grids and the elastic fibers of the heart walls are *immersed* in the flow: fluid and fibers exert time-varying forces on each another. A Lagrangian coordinate system moving with the local fluid velocity is attached to the fibers and tracks their location in space; the information about the position of the fibers and their forcing on the fluid is transferred to the Eulerian underlying mesh where the flow solution is obtained. In this procedure the IB forcing consists of delta functions located on the first cells external to the immersed body. These forces can not be adequately represented on the computational mesh and, therefore, a smooth transition between the external fluid region and internal (body) cells is introduced. This

is equivalent to spreading the delta function over a narrow band (typically 3 or 4 nodes) across the boundary. The problem of heart modeling is complicated by the fact that the boundaries of the computational domain are moving and respond to forces (typically the pressure and viscous stresses) depending on the local flow conditions. In contrast, if the boundary configuration is fixed and known, the computation of the interaction between the fluid and the immersed surfaces is much simpler. In principle, Peskin's approach can be applied directly by decreasing the deformability of the elastic fiber; in practical terms this will result in a numerically stiff problem. The first applications of the IB approach to problems with solid, indeformable immersed surface were carried out by Basdevant and Sadourny [23], Briscolini and Santangelo [24], and Goldstein *et al.* [25]. Briscolini and Santangelo [24] used an IB approach (referred to as mask method, which was a modified version of that by Basdevant and Sadourny [23]) to compute the unsteady two-dimensional flow around circular and square cylinders at Reynolds numbers up to 1000, whereas Goldstein *et al.* [25] considered the two-dimensional start-up flow around a circular cylinder and three-dimensional plane- and ribbed-turbulent channel flow. In these works, the IB approach is used in conjunction with spectral methods and the forcing is applied in a band (consisting of $3-4$ computational nodes) around the interface. This was required to reduce spurious oscillations appearing in the solutions. On the other hand, Saiki and Biringen [26] used the forcing of Goldstein *et al.* [25] to compute the flow around steady and rotating circular cylinders using fourth order central finite-difference approximations. The use of finite-differences avoided the appearance of spurious flow oscillations at the boundary even if, in that case, the forcing was spread across the boundary using a procedure that the authors refer to as 'first-order accurate similar to the delta-function of Peskin'. The main drawback of the forcing introduced in Goldstein *et al.* [25] is that it contains two free parameters that require a problem-dependent tuning; in particular, for unsteady flows this forcing introduces a time-step limitation that reduces the efficiency and the applicability of the method. Another disadvantage of the described methods is that, in order to reduce the stiffness of the solution algorithm and to avoid unphysical flow oscillations, the boundary forcing terms are spread across the boundary which is therefore smeared over the grid and reduces the solution accuracy. Mohd-Yusof [10] derived an alternative formulation of the forcing that does not affect the stability of the discrete-time equations and does not require forcing smoothing. In addition, no user-defined parameters were used, making the approach flow independent. In Mohd-Yusof [10] the new forcing

was combined with B-splines to compute the laminar flow over a three-dimensional ribbed channel, showing substantial improvements with respect to the previous formulations. This discrete-time forcing scheme was originally developed in a spectral context and has also been successfully applied to flows around cylinders and spheres, at moderate Reynolds numbers. The same idea of forcing has been used by Fadlun *et al.* [11], Balaras [27], Kang *et al.* [28], Kim *et al.* [29], Majumdar *et al.* [30], Tzeng and Ferziger [31], Dadone and Grossman [32]. It is important to note that in this case, the forces are not specified in the continuous space by means of some physical arguments, but rather in the discrete space by directly requiring the solution to respect the desired boundary conditions. This process is equivalent to a local reconstruction of the solution near the immersed boundary. In Fadlun *et al.* [11] and Balaras [27], the solution is reconstructed at the fluid nodes closest to the immersed boundary with one-dimensional interpolation scheme along an arbitrary grid line and along the line normal to the interface, respectively. A revised version of the linear interpolation method along with quadratic and 'quadratic + momentum' interpolation schemes have been presented by Kang *et al.* [28], to reduce the error accumulated in the pressure field in the time-marching scheme. Differently, in Kim *et al.* [29], Majumdar *et al.* [30], Tzeng and Ferziger [31] the solution is reconstructed at points inside the solid phase, called 'ghost-points', using linear, quadratic and distance-weighted interpolation methods. In the ghost-cell framework, Dadone and Grossman [32] have combined the IB method with the curvature-corrected symmetry technique (CCST) to accurately impose boundary conditions for inviscid compressible flows. The above reconstruction techniques have been used in the framework of finite-difference and finite-volume methods, and were shown to preserve second-order space accuracy. The applications included the flow around simple and complicated geometries in a wide range of Reynolds numbers for incompressible and inviscid compressible flows calculations.

The first compressible viscous flows calculations have been presented by De Palma *et al.* [14] and de Tullio *et al.* [3]. A finite-volume numerical method has been employed for the solution of compressible Navier–Stokes equations, relying on high-order-upwind discretization of the convective fluxes and TVD limiter, in order to accurately describe transonic and supersonic flows. To make the methodology suitable for flow fields presenting both quasi-incompressible and supersonic regions, the solver has been equipped with a preconditioning strategy. Since industrial and aerospace applications involve very high Reynolds number flows, the Reynolds-Averaged equations have been implemented,

in conjunction with the $k - \omega$ turbulence model. A dual-time stepping approach is employed to cope with both steady and unsteady flows. Concerning the IB conditions, the direct forcing method by Mohd-Yusof [10] has been generalized to compressible flows, with several reconstruction strategies, namely the one-dimensional and distance-weighted reconstruction schemes. These works are limited to several two-dimensional and simple three-dimensional scalar calculations.

Present contribution

The purpose of this work is to develop and test an accurate and efficient tool for computing three-dimensional complex flows past fixed or moving geometries, for a wide range of Reynolds and Mach numbers.

Firstly, the scalar IB-URANS solver developed by De Palma *et al.* [14] and de Tullio *et al.* [3] for Cartesian grid is extended to three-dimensional computations, using a new IB least-squares reconstruction and a parallel decomposition of the computational domain, so as to perform simulations within reasonable computational times. The new reconstruction scheme, has been validated performing several test-cases. In order to efficiently compute turbulent flows, an adaptive version of wall functions has been introduced in combination with the IB, providing good results for attached flows.

Then, a code for solving heat conduction (HC) equation that uses the same spatial discretization and time-marching scheme as the URANS solver has been developed and coupled with it to obtain an efficient tool for solving Conjugate-Heat-Transfer (CHT) problems: the Cartesian grid presents both fluid and solid zones: the URANS equations are solved at all fluid cells and the HC equation is solved at all solid cells; the two solutions are coupled by the interface conditions requiring that both the temperature and heat-flux be the same at all (fluid-solid) boundary points. More in detail, at each iteration: i) the IB grid generator detects the position of each cell of the Cartesian grid with respect to the Lagrangian mesh, and divides the cells into four types: solid and fluid cells–whose centers lie within the body and within the fluid, respectively; fluid- and solid-interface cells, that have at least one of their neighbors inside and outside the body, respectively; ii) the URANS equations are solved at all internal fluid cells, whereas the heat conduction equation is solved at all internal solid cells using the same spatial discretization and time-marching scheme; iii) the boundary conditions, which account for the presence of the body

are imposed at the fluid-/solid-interface cells, using a local interpolation procedure; iv) the interface boundary conditions requiring that both the temperature and heat-flux are the same for the fluid and the solid at all boundary points are imposed by the CHT coupling approach. Extensive validation of the overall procedure for several test-cases concerning CHT problems in a wide range of both Reynolds and Mach numbers is carried out to demonstrate the accuracy of the proposed methodology: the flow past a heated cylinder in cross-flow, for which both experimental and numerical results using a body fitted CHT-RANS solver are available (Laskowski *et al.* [36]); compressible turbulent flow past the air cooled C3X turbine guide vane (Hylton *et al.* [37]), for which detailed experimental results are also available. The interested reader is referred to the works by Andrei *et al.* [38], Yoshiara *et al.* [39], Luo and Razinsky [40].

Finally, a surface-based structural solver that simulates the dynamics of deformable geometries, discretized by triangulated Lagrangian meshes, has been coupled with the basic IB-URANS method to provide an efficient tool for solving Fluid-Structure-Interaction (FSI) problems: the forces exerted by the fluid onto the solid surface are used to determine its motion, which is fed as a kinematic boundary condition for the flow, via an iterative procedure implemented within the dual time stepping procedure of the URANS solver. More in detail, at each physical time-step: i) the IB grid generator detects the position of each cell of the Cartesian grid with respect to the Lagrangian mesh, and divides the cells into three types: solid and fluid cells–whose centers lie within the body and within the fluid, respectively; fluid-interface cells, that have at least one of their neighbors inside the body; ii) the URANS equations are solved at all internal fluid cells, and the solid cells have no influence of the fluid domain; iii) the boundary conditions, which account for the presence of the body are imposed at the fluid-interface cells, using a local interpolation procedure; iv) the forces exterted by the fluid upon the structure are fed to the structural solver which updates the position and the velocity of the surface mesh points. The proposed methodology emerges as a promising approach for aeroelasticity problems: a first computation with a moving geometry has been performed using the least squares reconstruction scheme with very little additional effort compared to the non-moving boundary case, namely, the oscillating circular cylinder in a cross-flow; a three-dimensional FSI computation of the incompressible low Reynolds number flow past a deformable sphere is presented, showing very good agreement with the analytical data.

Outline

This thesis is organized as follows.

In Chapter 1 the automatic generation process of the Cartesian grid and the Lagrangian meshes is presented, with a particular focus on the specific procedure used to determine the position of the grid cells with respect to the body and on the local grid refinement procedure.

In Chapter 2 the governing equations for fluid flows, the heat conduction equation in the solid and the structure dynamics equation are described.

In Chapter 3 the IB procedure, focusing on the different reconstruction schemes developed and implemented in the numerical tool is described in detail.

In Chapter 4 the numerical solvers are described in detail. Firstly, the numerical method for the solution of the preconditioned compressible Navier-Stokes equations and the heat conduction equation in the Cartesian domain, is described. Then, the finite element surface-based solver implemented for the solution of the structure dynamics is presented. Finally, the proposed coupling methodologies for both CHT problems and FSI problems are described.

In Chapter 5 several numerical results are provided. In the first section the IB-URANS solver is validated versus several test-cases, covering a wide range of Mach and Reynolds numbers for two- and three- dimensional flows past rigid geometries. Since the IB-URANS solver is validated, in the second and in the third section, both CHT computations and the FSI computations are presented in order to validate the coupling methodologies proposed.

Chapter 1

Automatic Mesh Generation

In contrast to body-fitted approaches, IB methods use a regular underlying Cartesian grid, not conforming to the boundaries in the computational domain. Hexahedral cells may extend through solid wall boundaries, therefore the volume mesh structure is independent of the geometry surface discretization and topology. This allows the treatment of any complex geometry without the need of tedious and time-consuming surface preparation processes. The surface description may focus uniquely on resolving the geometry, while the volume mesh should correctly describe the flow. On the other hand, Cartesian grids suffer from a certain "lack of resolution". To resolve thin boundary layers and geometrical details of various sizes and scales, the mesh is required to be very fine. This resolution is inevitably extended to the entire computational domain using a structured environment, and the resulting mesh may exceed the storage capacity of computers. Therefore, local adaptive mesh refinement could significantly alleviate this shortcoming. As demonstrated by Berger and Oliger [41], De Zeeuw and Powell [42], Quirk [43], Melton *et al.* [44], Karman [45], Welterlen and Karman [46], Melton *et al.* [47], Melton [48], the technique is very amenable to automation. Since solid wall boundaries may cut arbitrarily the volume cells, an important component of these methods is the specific procedure to determine the position of the cells with respect to the body. Fortunately, the fundamental issues related to this problem have been thoroughly studied, and robust algorithms are available in computational geometry and computer graphics literature (Preparata and Shamos [49], Voorhies [50], O'Rourke [51], Foley *et al.* [52]).

It is important to note that the approach is component-based: the first step of the mesh generation process is the positioning of the geometries in the computational domain. Therefore, for complex configurations, one needs to create separate components that are

not required to have commensurate length scales. Once created, the components can be easily translated or rotated as necessary to quickly obtain new configurations.

1.1 Surface Geometry

Three-dimensional geometries can be specified in a variety of formats, like proprietary CAD formats, trimmed Non Uniform Rational B-Splines (NURBS), Stereo-Lithography (STL) formats, network of grid patches, and others.
The Cartesian methods essentially rely on the identification of the relative position between the body surface and the computational nodes, and this is efficiently performed in three dimensions if the body surface is described by triangles. For this reason, the STL format, which is the standard for the *Rapid Prototyping* community, has been employed in this work.
The STL representation of a surface is a collection of unconnected triangles of sizes inversely proportional to the local curvature of the original surface (Figure 1.1), and the *only* requirement for the object description is that the given surface must be a closed manifold. This restriction is enforced by rapid prototyping tools and guarantees that the final objects can be machined (produced).

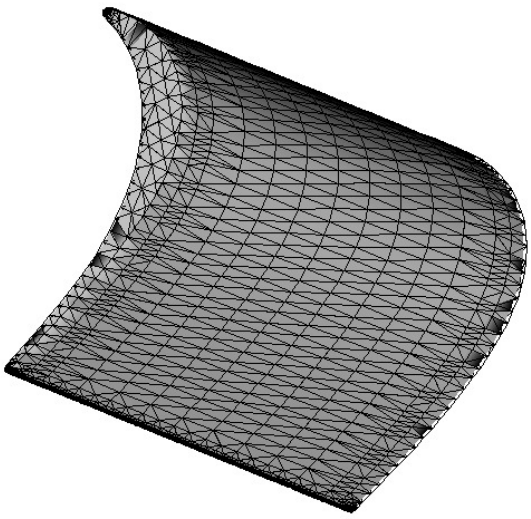


Figure 1.1: Surface triangle distribution in a STL model.

1.2 Cartesian Volume Mesh Generation

The Cartesian volume mesh generation task is simple, where the only complications arise from the presence of the immersed geometries. Since generation of uniform Cartesian cells is extremely fast, the performance of the overall algorithm depends directly on the scheme used for adaptive refinement.

The mesh generation process begins with an initial coarse mesh, with a resolution given by the user, covering the domain of interest. A notional description of the domain in integer triplets is used. The use of integer coordinates makes it possible to unambiguously compare vertex locations and leads to compact storage schemes, making these methods particularly attractive for Cartesian meshes (Aftosmis *et al.* [53]). The user specifies the minimum X_d^{min} and maximum X_d^{max} coordinates for each Cartesian direction $d = 1, 2, 3$. Each direction can be split in a possible number N_d of cells, and each node i in the mesh can be specified exactly by an integer triplet P_d^i. The Cartesian coordinates of the nodes are reconstructed, when needed, from

$$x_d^i = X_d^{min} + \frac{P_d^i}{N_d}\left(X_d^{max} - X_d^{min}\right). \tag{1.1}$$

Following the geometric criteria described in Subsection 1.2.2, the mesh sizes are then recursively halved to increase the resolution in particular areas. To this purpose, the relative position of each grid node with respect to the body has to be determined.

1.2.1 Ray Tracing

The easiest and computationally efficient way to determine the position of the geometry with respect to the grid nodes is by using a ray tracing algorithm. A ray is cast between the query point Q and a control point C the latter being definitely outside the body. The number of intersections N of the ray with the surface are counted, if N is even the point is external while if N is odd the point is internal. This method is extremely general and it works for any surface provided it is closed (it must enclose an inner volume or it must be 'watertight'). The ray tracing works for non-convex objects and even if the body contains cavities; the only delicate point of the procedure is that the intersections must be counted properly. Consider first the two-dimensional example of Figure 1.2 in which the surface triangles are segments: for the inner points Q_1 and Q_2 and for the outer points Q_3 and Q_4 the intersection counting is straightforward but the same is not true for the internal point Q_5 and the external Q_6. Following O'Rourke, however, the problem can be

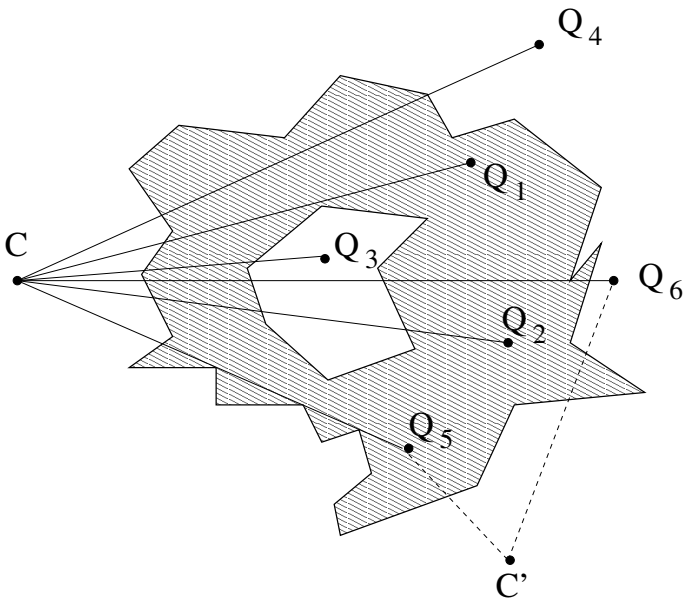


Figure 1.2: Generic 2D non convex object crossed by several rays.

easily solved by defining a criterion for an intersection to be considered valid. If S_1 and S_2 are the extrema of a segment there is a valid intersection when one of the endpoints is strictly above the ray and the other at the same level or below the ray. Without loss of generality all possible cases can be reduced to those of Figure 1.3 by properly rotating the ray and/or the segment. According to this criterion points Q_5 and Q_6 are

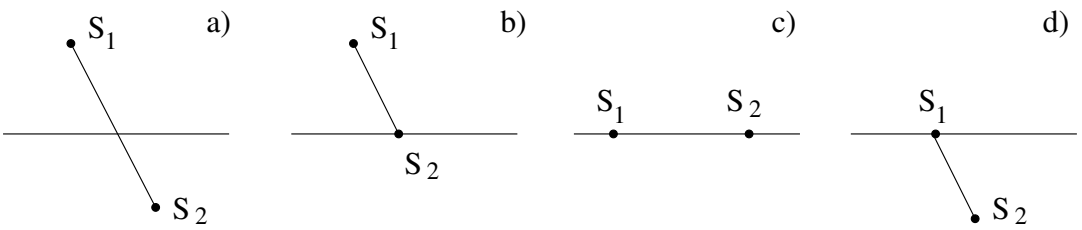


Figure 1.3: Different configurations for the intersection between two segments in a plane: cases a) and b) have a valid intersection, cases c) and d) no valid intersections.

correctly identified, respectively, as internal and external since the ray of the former has only one valid intersection with the body while the latter has six intersections. It is worth mentioning that since the intersection is computed by floating point arithmetic the terms 'intersecting' or 'non intersecting' must be considered within some tolerance. In other words cases b), c) and d) of Figure 1.3 could be correctly identified or not depending on the set tolerance. The ray tracing procedure, however, must work for *any* ray independently

of the particular control point, therefore if the amplitude of the tolerance parameter is an issue a different control point can easily used to answer the question. For example, the control point C' of Figure 1.2 does not cause any ambiguity for the intersection counting of the query points Q_5 and Q_6. Although a rigorous proof for this point can not be given, O'Rourke [51] reports that for a generic non convex polyhedron out of one million of rays generated in 0.8% of cases a second ray was needed, in 0.01% a third one and only in one case a further attempt was necessary. All the arguments previously illustrated

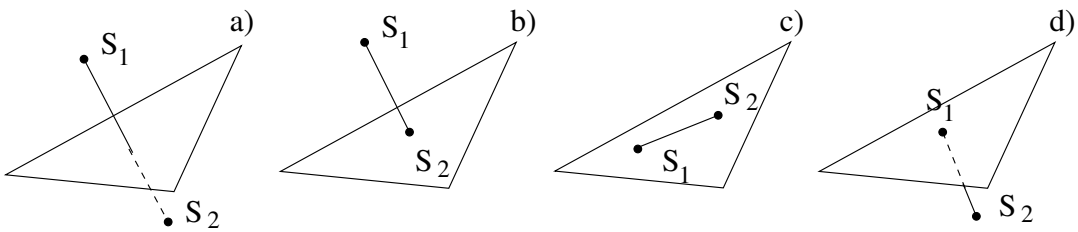


Figure 1.4: Different configurations for the intersection between a segment and a triangle in space: cases a) and b) have a valid intersection, cases c) and d) have no valid intersections.

for a two-dimensional example can be easily extended to three-dimensional configurations provided that the intersection between a ray and a segment is replaced by the intersection between a ray and a triangle (Figure 1.4). An important advantage of describing a surface in space by triangles is that the intersection between a ray and a triangle can be found parametrically by a computationally efficient procedure illustrated in Appendix A.

1.2.2 Refinement Criteria

The ray-tracing procedure allows one to generate a locally refined grid very easily, so as to use a fine grid in the high-flow-gradient regions and a coarser one where the flow is smooth. Now that the cells are identified to be 'inside' or 'outside' the body, a *tag* function is generated to mark the cells assigning the value ± 1 to 'fluid' and 'solid' cells, respectively. The gradient of this function is different from zero only at the immersed boundary and is used to select the rows of cells to be refined. A smoothing function can be applied on the ± 1 *tag* function to obtain a smeared interface that will allow a smoother transition between coarse and refined regions. In fact, anytime a cell is tagged for division, the refinement must be propagated several layers into the mesh to avoid corruption of the difference stencils in the immediate vicinity of the body. The automatic refinement strategy is based on the curvature information. The volume cells tagged for

refinement which intersect the body surface will include a number N_T of surface triangles and the local surface normal within the cell is computed averaging the N_T surface normals. Given a desired normal and tangential resolution $(\Delta n, \Delta t)$ on the surface, the cell splitting procedure is performed iteratively in each Cartesian direction independently until the cell size reaches a target defined as

$$\Delta x_i = \min\left(\frac{\Delta n}{|\tilde{n}_i|}, \Delta t\right), \tag{1.2}$$

where i represents each Cartesian direction and $\tilde{n}_i$ are the direction cosines of the local normal.

In addition to the geometric automatic refinement, it is possible to define additional regions of the computational domains to be refined. The aim is to enhance resolutions in regions of the flow field with large gradients. Figure 1.5(a) shows a 'Box window', which enforces a prescribed grid resolution inside a box; Figure 1.5(b) shows a 'Segment window', which enforces a prescribed grid resolution in the region within a distance from a line segment; Figure 1.5(c) shows a 'Surface refinement', which enforces a prescribed grid resolution along the surface of a solid body; and Figure 1.5(d) shows a 'Layer window' which enforces a prescribed grid resolution within a distance from a solid body. 'Surface resolution' is useful to achieve a desired grid resolution near the wall and the 'Layer window' can be used to achieve a smooth mesh transition from the boundary to the far field.

1.3 Parallel decomposition

A Cartesian grid with hundreds of millions cells is required to compute three dimensional flows of industrial interest. Therefore, the solver, in addition of being capable of refining the grid locally, e. g., within the boundary layers, wakes and cooling channels, has been parallelized, so that when running on a multiprocessor computer, may require an acceptable computer time for preliminary design. The parallel strategy employed here is a ghost-cell one. The computational domain is decomposed into blocks, and, at block interfaces, rows of ghost cells are created in order to exchange information between neighboring blocks. Each processor has its own grid that advances the solution and, after each solver iteration, the solution in the ghost cells is transferred from one processor to the neighbor one. The parallel communication between blocks is based on MPI libraries. Multilevel Partitioning Algorithm (PARMETIS) and node balancing are used to minimize communications between CPUs.

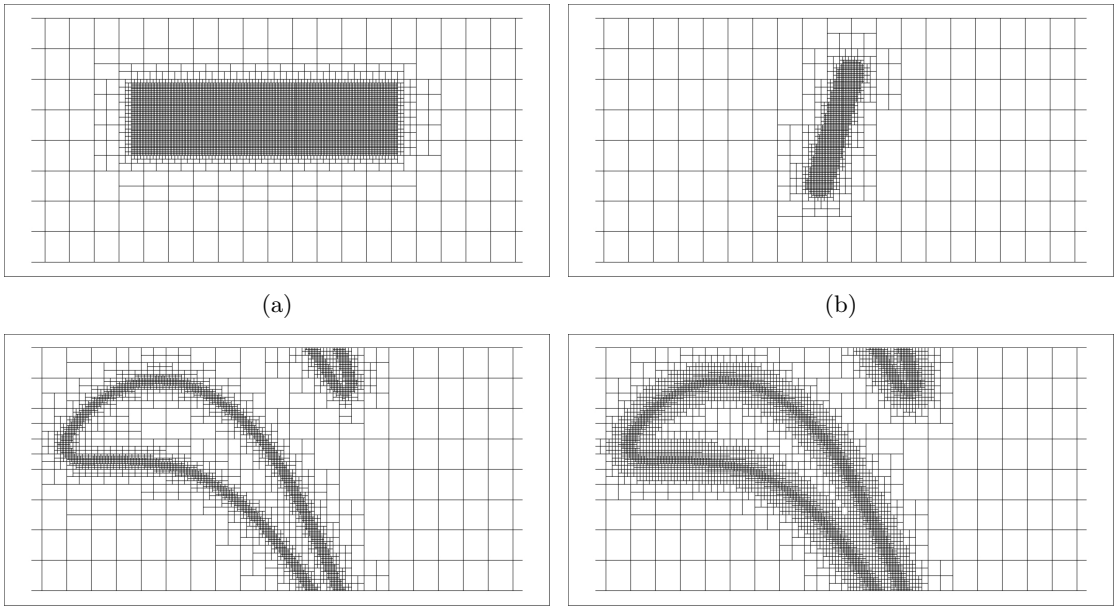


Figure 1.5: Examples of user-input keywords for local grid refinement.

Chapter 2

Governing Equations

2.1 Navier–Stokes equations for turbulent flows

The equations governing the motion of an unsteady, compressible viscous flow of a Newtonian fluid are the Navier–Stokes equations. They represent the conservation of mass, momentum and energy. These equations can be written in Cartesian coordinates (neglecting the body forces) as:

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_j}(\rho\, u_j) = 0, \tag{2.1}$$

$$\frac{\partial(\rho\, u_i)}{\partial t} + \frac{\partial}{\partial x_j}(\rho\, u_j\, u_i) = -\frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ji}}{\partial x_j}, \tag{2.2}$$

$$\frac{\partial(\rho\, e)}{\partial t} + \frac{\partial}{\partial x_j}(\rho\, u_j\, h) = \frac{\partial}{\partial x_j}\left[u_i\, \tau_{ij} - q_j\right], \tag{2.3}$$

where ρ is the density, u_i is the component of the velocity in the i-direction and e is the total energy per unit mass,

$$e = e_{int} + \frac{V^2}{2}\ , \tag{2.4}$$

e_{int} being the internal energy per unit mass and V being the magnitude of the velocity, h is the total enthalpy per unit mass,

$$h = e_{int} + \frac{p}{\rho}\ , \tag{2.5}$$

and q_j are the heat flux vector components, assumed following the Fourier's law for heat transfer by conduction

$$q_j = -K\frac{\partial T}{\partial x_j}\ . \tag{2.6}$$

The viscous stress tensor $\tau_{i,j}$ is given as follows:

$$\tau_{i,j} = \mu\left[\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\frac{\partial u_k}{\partial x_k}\delta_{i,j}\right]\ . \tag{2.7}$$

In order to close the system of fluid dynamic equations the thermodynamic variables are related by the equations of state. The problems of interest in this work involve a perfect gas at relatively low temperatures, so that it is possible to assume a *perfect gas* with constant specific heats. The equations of state are the following:

$$p = \rho(\gamma - 1)e_i = \rho(\gamma - 1)\left(e - \frac{V^2}{2}\right) , \tag{2.8}$$

$$h = \frac{\gamma p}{(\gamma - 1)\rho} + \frac{V^2}{2} . \tag{2.9}$$

Finally, the coefficient of viscosity can be related to the thermodynamic variables using the Sutherland's formula:

$$\mu = C_1 \frac{T^{3/2}}{T + C_2} , \tag{2.10}$$

where C_1 and C_2 are constants for a given gas. For air at moderate temperatures, $C_1 = 1.458 \cdot 10^{-6}\ kg/(m\ s\ \sqrt{K})$, $C_2 = 110.4\ K$. The Prandtl number

$$Pr = \frac{c_p \mu}{K} \tag{2.11}$$

is used to determine the coefficient of thermal conductivity K once μ is known since the ratio (c_p/Pr) is approximately constant for most gases. For air at standard conditions, $Pr = 0.72$.

Turbulent flows are governed by Equations (2.1) - (2.3) but they are characterized by fluctuations in space and time that can be characterized by very small space scale and high frequency. To solve a turbulent flow by *direct numerical simulation* (DNS) all relevant length scales have to be resolved, from the smallest eddies to scales on the order of the physical dimensions of the domain. Moreover, the time steps must be small enough so that the small-scale motion can be resolved in a time-accurate manner even if the flow is steady in a time-mean sense. This results in extremely high resolution requirements in space and time, demanding huge computer resources for high Reynolds number flow configurations.

Instead, the instantaneous and exact governing equations are typically averaged (either in time or in space) to remove the smallest scales, resulting in a modified set of equations that are computationally less expensive to solve. When the averaging procedure is applied to the unknown quantities in the Navier-Stokes equations, a new set of equations can be derived, the so-called Reynolds-Averaged Navier-Stokes (RANS) equations. The RANS equations now contain a term that cannot be expressed in terms of averages, usually

referred to as the *Reynolds stress tensor*. In order to *close* the equations, it is necessary to introduce additional assumptions: a turbulence model.

2.1.1 $k-\omega$ turbulence model

The simplest approach is to relate the large-scale effects of turbulence to modified thermodynamic properties of the fluids, namely to an effective viscosity obtained as the sum of the conventional molecular term and a turbulent or *eddy* viscosity, μ_t. In this case, the Reynolds stress tensor can be written as:

$$\tilde{\tau}_{ij} = \mu_t \left[\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\frac{\partial u_k}{\partial x_k}\delta_{i,j} \right] - \frac{2}{3}\rho k \delta_{ij}. \tag{2.12}$$

Equation (2.12) is normally referred to as Boussinesq hypothesis. In Equation (2.12) k is the turbulent kinetic energy defined as:

$$k = \frac{1}{2}\overline{u_i' u_i'}, \tag{2.13}$$

where u_i' are the fluctuations of the velocity components u_i. The solution of the RANS equations is now possible once the eddy viscosity is defined. There is a very large amount of literature devoted to different definition of μ_t but in general the models are derived on the basis of *(i)* dimensional arguments (i.e. the units are $[m^2/s]$), *(ii)* physical insights (i.e. consistency with experimental observation) and *(iii)* practical aspects of the formulation such as simplicity and generality. The $k-\omega$ turbulence model developed by Wilcox (1988a) [54] has been employed in this work. The transport equations for the turbulent kinetic energy k and for the specific dissipation rate ω are:

$$\frac{\partial(\rho\, k)}{\partial t} + \frac{\partial}{\partial x_j}(\rho\, u_j\, k) = \tilde{\tau}_{ij}\frac{\partial u_i}{\partial x_j} - \beta^* \rho\, \omega\, k + \frac{\partial}{\partial x_j}\left[(\mu + \sigma^* \mu_t)\frac{\partial k}{\partial x_j}\right], \tag{2.14}$$

$$\frac{\partial(\rho\, \omega)}{\partial t} + \frac{\partial}{\partial x_j}(\rho\, u_j\, \omega) = \alpha\frac{\omega}{k}\tilde{\tau}_{ij}\frac{\partial u_i}{\partial x_j} - \beta\, \rho\, \omega^2 + \frac{\partial}{\partial x_j}\left[(\mu + \sigma \mu_t)\frac{\partial \omega}{\partial x_j}\right], \tag{2.15}$$

where the eddy viscosity is

$$\mu_t = \alpha^* \rho \frac{k}{\omega}. \tag{2.16}$$

The model coefficients are:

$$\alpha^* = 1, \qquad \alpha = \frac{5}{9}, \tag{2.17}$$

$$\beta = \frac{3}{40}, \qquad \beta^* = \frac{9}{100}, \qquad \sigma = \frac{1}{2}, \qquad \sigma^* = \frac{1}{2}. \tag{2.18}$$

Low Reynolds Correction

The model presented in the previous subsection is restricted to high–Reynolds number applications since it does not take in account low–Reynolds number effects. In fact, for example, the model fails to predict the sharp peak in turbulent kinetic energy close to surface for pipe and channel flows. In order to adapt the model to low-Reynolds number flows, a correction is which renders the model coefficients α, α^*, and β^* functions of the *turbulent* Reynolds number defined as:

$$Re_t = \frac{\rho k}{\omega \mu}. \tag{2.19}$$

The coefficients maintain the value of Equations (2.17) - (2.18) as $Re_t \to \infty$, that is the fully turbulent case (herein called α_{HR} and β_{HR}), whereas they assume a lower value as the turbulent Reynolds number decreases (e.g. in the viscous sub-layer). The relations introduced by Wilcox are the followings:

$$\alpha^* = \frac{\alpha_o^* + Re_t/R_k}{1 + Re_t/R_k}, \tag{2.20}$$

$$\alpha = \alpha_{HR} \frac{\alpha_0 + Re_t/R_\omega}{1 + Re_t/R\omega} \left(\alpha^*\right)^{-1}, \tag{2.21}$$

$$\beta^* = \beta_{HR}^* \frac{\beta_0 + \left(Re_t/R_\beta\right)^4}{1 + \left(Re_t/R_\beta\right)^4}. \tag{2.22}$$

Table 2.1 provides two set of the coefficients value, corresponding to the 1988a and 1998 version of the model: The coefficients R_k, R_ω, and R_β control the rate as the closure

	R_k	R_ω	R_β	α_0	α_0^*	β_0
Wilcox[88a]	6	2.7	8	1/10	$\beta/3$	5/18
Wilcox[98]	6	2.95	8	1/9	$\beta/3$	4/15

Table 2.1: Coefficients for the transitional $k - \omega$ turbulence model.

coefficients approach the fully–turbulent case ones, and their value has been tuned with experimental and numerical data (DNS).

2.1.2 $k - \omega$ Unsteady Reynolds-Averaged Navier–Stokes equations

The Reynolds Averaged Navier–Stokes (RANS) equations, in terms of Favre mass-averaged quantities, using the $k - \omega$ turbulence model, in a Cartesian coordinate system can be rewritten:

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_j} \left(\rho\, u_j\right) = 0, \tag{2.23}$$

$$\frac{\partial(\rho\, u_i)}{\partial t} + \frac{\partial}{\partial x_j}\left(\rho\, u_j\, u_i\right) = -\frac{\partial p_t}{\partial x_i} + \frac{\partial \hat{\tau}_{ji}}{\partial x_j}, \tag{2.24}$$

$$\frac{\partial(\rho\, \tilde{H} - p_t)}{\partial t} + \frac{\partial}{\partial x_j}\left(\rho\, u_j\, \tilde{H}\right) = \frac{\partial}{\partial x_j}\left[u_i\, \hat{\tau}_{ij} + (\mu + \sigma^* \mu_t)\frac{\partial k}{\partial x_j} - q_j\right], \tag{2.25}$$

$$\frac{\partial(\rho\, k)}{\partial t} + \frac{\partial}{\partial x_j}\left(\rho\, u_j\, k\right) = S_k + \frac{\partial}{\partial x_j}\left[(\mu + \sigma^* \mu_t)\frac{\partial k}{\partial x_j}\right], \tag{2.26}$$

$$\frac{\partial(\rho\, \omega)}{\partial t} + \frac{\partial}{\partial x_j}\left(\rho\, u_j\, \omega\right) = S_\omega + \frac{\partial}{\partial x_j}\left[(\mu + \sigma \mu_t)\frac{\partial \omega}{\partial x_j}\right], \tag{2.27}$$

where

$$\tilde{H} = h + \frac{1}{2}\left(u^2 + v^2 + w^2\right) + \frac{5}{3}k, \tag{2.28}$$

$$p_t = p + \frac{2}{3}\rho k = \rho R T + \frac{2}{3}\rho k = \rho\left(R + \frac{2}{3}\frac{k}{T}\right) = \rho \tilde{R} T, \tag{2.29}$$

$$S_k = \tilde{\tau}_{ij}\frac{\partial u_i}{\partial x_j} - \beta^* \rho\, \omega\, k, \tag{2.30}$$

$$S_\omega = \alpha \frac{\omega}{k}\tilde{\tau}_{ij}\frac{\partial u_i}{\partial x_j} - \beta\, \rho\, \omega^2, \tag{2.31}$$

$$\hat{\tau}_{ij} = (\mu + \mu_t)\left[\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\frac{\partial u_k}{\partial x_k}\delta_{i,j}\right]. \tag{2.32}$$

The heat flux vector components q_j are rewritten as:

$$q_j = -\left(\frac{\mu}{Pr} + \frac{\mu_t}{Pr_t}\right)\frac{\partial h}{\partial x_j}, \tag{2.33}$$

where Pr_t is the turbulent Prandtl number.

2.2 Heat conduction equation

Heat transfer can occur for convection, conduction or radiation. The conduction mode occurs either because of an exchange of energy from one molecule to another, with no motion, or because of the motion of the free electrons. On the other hand, molecules present in liquids and gases have freedom of motion and they carry energy. The transfer of heat from one region to another due to such a motion, added to conduction within the fluid, builds heat transfer by convection. When fluid motion occurs by density variation, it is called free convection. When fluid motion is driven by external forces, forced convection occurs. Finally, heat transfer can occur via thermal radiation. According to the particular physical case, the three heat transfer modes are involved in a different amount and some

of them can be neglected. It is usually important to quantify the amount of energy being transferred per unit time. The temperature distribution in a medium is the main objective of a conduction analysis and it involves the evaluation of the temperature in the medium as a function of space at steady state and as a function of time during the transient state. Once a temperature distribution is known, for heat conduction the rate equation is known as Fourier's law and expressed as:

$$\boldsymbol{q} = -K_s \nabla T_s \tag{2.34}$$

where K_s is the thermal conductivity and $\boldsymbol{q}$ is the thermal flux.

The energy balance law considered at the beginning of this chapter, can be summarized for a generic control volume of Figure 2.1 as:

$$energy\ stored\ = Inlet\ energy\ -\ exit\ energy +\ energy\ generated\ . \tag{2.35}$$

This relation can be re-written as:

$$\rho \Delta x \Delta y \Delta z \frac{\partial T}{\partial t} = (Q_x + Q_y + Q_z) - (Q_{x+dx} + Q_{y+dy} + Q_{z+dz}) + G\Delta x \Delta y \Delta z. \tag{2.36}$$

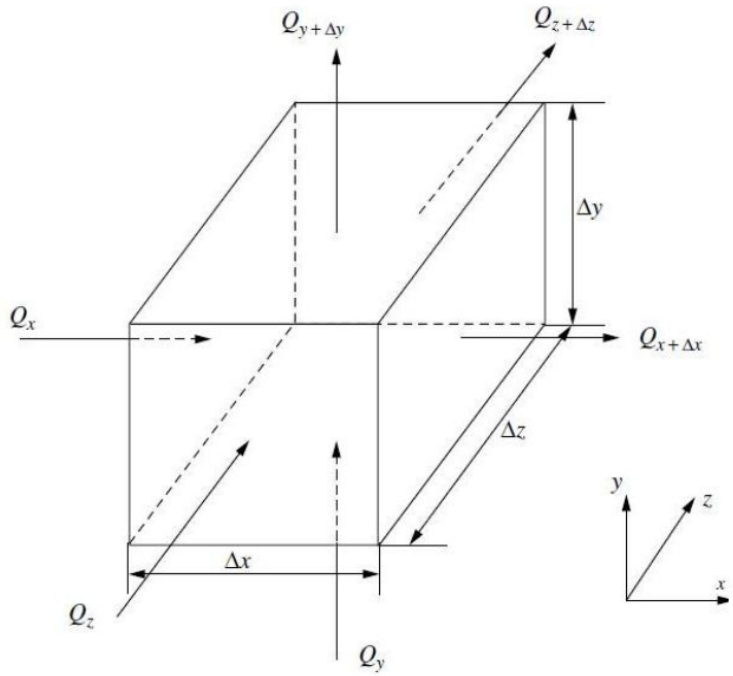


Figure 2.1: Control volume for heat transfer equation.

From Equation (2.36), the conduction heat equation can be written as:

$$\rho_s c_s \frac{\partial T_s}{\partial t} = \nabla \cdot (\boldsymbol{K}_s \nabla T_s) + S_s \tag{2.37}$$

where T is the temperature, ρ_s is the density and c_s is the volumetric specific heat. The last term on the right hand side is any source term defined by the problem. The thermal conductivity in the most general case is a tensor defined as:

$$\boldsymbol{K}_s = \begin{bmatrix} K_{xx} & K_{xy} & K_{xz} \\ K_{yx} & K_{yy} & K_{yz} \\ K_{zx} & K_{zy} & K_{zz} \end{bmatrix}. \tag{2.38}$$

Formulation (2.38) is valid for solving for heat conduction problems in anisotropic material, with a directional variation in the thermal conductivity. In this study, an isotropic material behavior is assumed and the thermal conductivity reduces to the scalar quantity, K_s.

2.3 Structure dynamics equation

The governing equation for the structure dynamics read

$$\rho_s \frac{d^2\boldsymbol{r}}{dt^2} + \eta_d \frac{d\boldsymbol{r}}{dt} = \nabla \cdot \boldsymbol{\sigma} + \rho_s \boldsymbol{t_s} + \rho_s \boldsymbol{g_s} \tag{2.39}$$

where ρ_s is the material density, $\boldsymbol{r}$ is the displacement vector, η_d is a damping coefficient, $\boldsymbol{\sigma}$ is the stress tensor, $\boldsymbol{t_s}$ is the vector of external surface forces per unit mass and $\boldsymbol{g_s}$ is the vector of the body forces per unit mass. Note that the equation is given in the Lagrangian reference frame, therefore the time derivatives of the displacements are material (or total) derivatives.

The stress tensor $\boldsymbol{\sigma}$ is related to the strain tensor

$$\boldsymbol{E} = \frac{1}{2}\left(\nabla\boldsymbol{r} + (\nabla\boldsymbol{r})^T + (\nabla\boldsymbol{r})^T \cdot \nabla\boldsymbol{r}\right), \tag{2.40}$$

by a constitutive law $\boldsymbol{\sigma} = f(\boldsymbol{E})$ describing the material behavior.

Chapter 3

Immersed Boundary technique

The IB method allows one to greatly simplify the grid generation process and even to automate it completely. The governing equations are solved directly on a Cartesian grid in their simplest form by means of very efficient numerical schemes that can be also parallelized in a relatively simple way. On the other hand, the effect of the body has to be accounted for and the relative position of each grid cell with respect to it has to be determined. This is accomplished by a tagging algorithm; the grid generator detects the cell faces that are cut by the body surface and divides the cells into four types: solid and fluid cells, whose centers lie within the body and within the fluid, respectively; and fluid-/solid-interface cells, which have at least one of their neighbor cells inside the body/fluid, respectively. Then, the centers of the fluid- and solid-interface cells are projected onto the body surface along its normal direction, so as to obtain fluid-cell and solid-cell projection points (FCPPs, SCPPs), respectively, see Fig. 3.1. The boundary conditions that

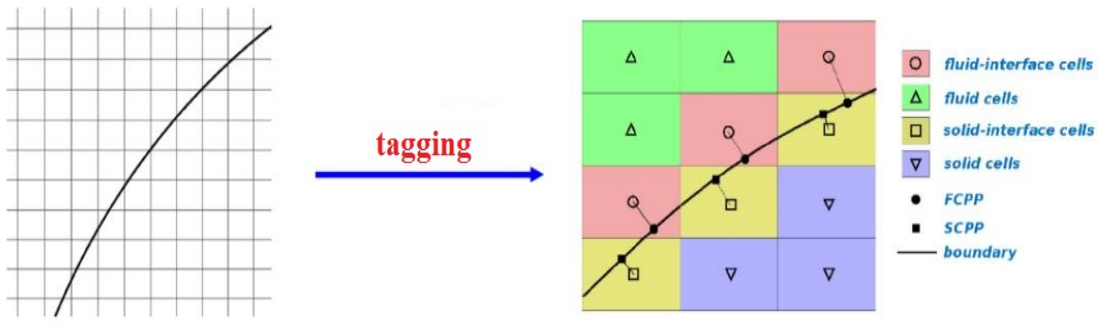


Figure 3.1: Tagging technique.

account for the presence of the body are mimicked by a special treatment of the interface cells, according to the direct forcing procedure proposed by Mohd-Yusof [10]. A local

interpolation procedure is used to transfer the boundary conditions onto the centers of the fluid-/solid-interface cells, whereas the governing equations are solved at all of the fluid/solid cell centers and the solid/fluid cells have no direct influence on the fluid/solid domain. Different methods of imposing the IB conditions are described in detail in the following.

3.1 One-dimensional reconstruction

In general, the position of the center of each cell, where the flow unknowns are located, does not lie on the immersed boundary, so that the solution is to be interpolated to account for the wall boundary conditions. Following the idea of Fadlun [11], the simplest interpolation is the one-dimensional linear one. For each fluid-interface cell, the shortest Cartesian distance between the cell center and the wall is evaluated and the relative FCPP (W) is determined along the corresponding Cartesian direction. As shown in Fig. 3.2 (for the two-dimensional case), the flow variable at the center of the interface cell (point I) is linearly interpolated between the imposed value at the FCPP and the computed value at the neighboring fluid-cell center (point A). For the pressure, a first-order accurate Neumann boundary condition is imposed forcing the value at point I to be equal to that at point A.

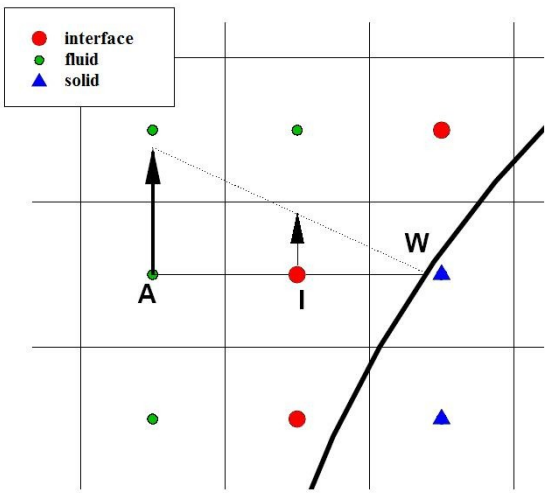


Figure 3.2: Linear reconstruction scheme.

3.2 Inverse distance weighted reconstruction

For each fluid-interface cell, the (normal) distance between the cell center and the wall is evaluated and the relative FCPP (W) is determined. At the fluid-interface cell I the flow variables are then reconstructed using an inverse-distance-based interpolation of the computed values of N_{nbr} surrounding fluid cells and the imposed one at W, see Fig. 3.3.

Thus, the interpolation formula used to reconstruct the variable Φ_I when the Dirichlet condition Φ_W is to be imposed at the immersed boundary point, W, is:

$$\Phi_I = \sum_i^{N_{nbr}} \frac{\alpha_i}{q}\Phi_i + \frac{\beta}{q}\Phi_W, \tag{3.1}$$

where $q = \sum_i^{N_{nbr}} \alpha_i + \beta$, $\alpha_i = 1/d_i$, $\beta = 1/d$. d_i are the distances between the surrounding cell centers and the fluid-interface cell center and d is its distance from the wall.

On the other hand, when a Neumann boundary condition is to be imposed, its first order discretization reads: $\Phi_W = \Phi_I - \frac{1}{\beta}\left(\frac{\partial\Phi}{\partial n}\right)_W$; and for the special case of the pressure, the homogeneous Neumann condition, $\left(\frac{\partial\Phi}{\partial n}\right)_W = 0$, leads to the following interpolation formula for Φ_I:

$$\Phi_I = \frac{\sum_i^{N_{nbr}} \frac{\alpha_i}{q}\Phi_i}{1 - \frac{\beta}{q}}. \tag{3.2}$$

In conclusion, at all fluid-interface points, the flow and turbulence variables are evaluated using Equation (3.1), whereas the pressure is computed using Equation (3.2).

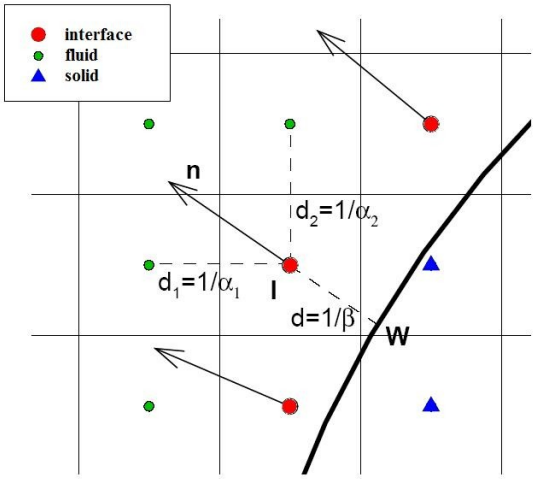


Figure 3.3: Distance-weighted reconstruction scheme.

3.3 Least squares reconstruction

For each fluid-interface cell, the (normal) distance between the cell center and the wall is evaluated and the relative FCPP (W) is determined. The value of each variable of interest at the center of each fluid-interface cell is computed by linear interpolation of the values at W–as prescribed by the boundary condition–and at the location of a probe L placed at a distance δ from the wall along its local normal direction; this does not coincide with a fluid-cell center and is to be evaluated in terms of the values at ne first neighboring fluid-cell centers. The choice of δ, which is critical for the application of this procedure, has been chosen equal to twice the largest distance between the interface cells and the wall.

The interpolation procedure is that proposed by Vanella and Balaras [55]: a support-domain is defined around each probe and a shape function is constructed in it, based on a moving least-squares approach. In this work the support-domain is a box of half-size $1.5H_x \times 1.5H_y \times 1.5H_z$ centered at the probe location. H_x, H_y and H_z are different for each probe and are proportional to the local grid. In particular

$$H_x = \max(|x_I - x_i|), \quad H_y = \max(|y_I - y_i|), \quad H_z = \max(|z_I - z_i|), \\ \text{with i} \ = \ 1,2,...,\text{ne}. \tag{3.3}$$

The transfer operator that evaluates the unknown variable U at each probe, L, from the known values at the ne neighboring fluid cells, u_i, is finally obtained as:

$$U^L(\mathbf{x}) = \sum_{j=1}^{m} p_j(\mathbf{x})\, a_j(\mathbf{x}) = \mathbf{p}^T(\mathbf{x})\, \mathbf{a}(\mathbf{x}). \tag{3.4}$$

In Equation (3.4), $\mathbf{p}^T(\mathbf{x})$ is the basis functions vector of length m, $\mathbf{a}(\mathbf{x})$ is the interpolation-coefficients vector and $\mathbf{x}$ is the position of the probe. Vanella and Balaras [55] show that a linear basis, $\mathbf{p}^T(\mathbf{x}) = [1\ x\ y\ z]$, is cost-efficient and second order accurate. The vector, $\mathbf{a}(\mathbf{x})$, is computed as follows. A weighted L2-norm is defined as:

$$J = \sum_{k=1}^{ne} W(\mathbf{x} - \mathbf{x}^k)[\mathbf{p}^T(\mathbf{x}^k)\mathbf{a}(\mathbf{x}) - u_i^k]^2, \tag{3.5}$$

where $\mathbf{x}^k$ is the position vector of the grid point k in the interpolation stencil, u_i^k is the variable value at grid point k, and $\mathrm{W}(\mathbf{x} - \mathbf{x}^k)$ is a cubic spline weight function that will be defined below. J is minimized with respect to $\mathbf{a}(\mathbf{x})$, leading to the following system:

$$
\begin{aligned}
\mathbf{A}\left(\mathbf{x}\right)\mathbf{a}\left(\mathbf{x}\right) &= \mathbf{B}\left(\mathbf{x}\right)\mathbf{u}_i^k, \\
\mathbf{u}_i^k &= [u_i^1 \; . \; . \; . \; u_i^{ne}]^T, \\
\mathbf{A}\left(\mathbf{x}\right) &= \sum_{k=1}^{ne} W\left(\mathbf{x}-\mathbf{x}^k\right)\mathbf{p}\left(\mathbf{x}^k\right)\mathbf{p}^T\left(\mathbf{x}^k\right), \\
\mathbf{B}\left(\mathbf{x}\right) &= \left[W\left(\mathbf{x}-\mathbf{x}^1\right)\mathbf{p}\left(\mathbf{x}^1\right) . \; . \; . \; W\left(\mathbf{x}-\mathbf{x}^{ne}\right)\mathbf{p}\left(\mathbf{x}^{ne}\right)\right]
\end{aligned}
\tag{3.6}
$$

Here, $\mathbf{A}(\mathbf{x})$ is a 4×4 matrix, while $\mathbf{B}(\mathbf{x})$ is of size $4 \times ne$. Combining Eqs. (3.4) and (3.6) one can write U as follows:

$$
U^L(\mathbf{x}) = \sum_{k=1}^{ne} \Phi_k^L(\mathbf{x}) u_i^k = \mathbf{\Phi}^T(\mathbf{x})\mathbf{u}_i^k \tag{3.7}
$$

where $\mathbf{\Phi}(\mathbf{x}) = \mathbf{p}(\mathbf{x})\mathbf{A}(\mathbf{x})^{-1}\mathbf{B}(\mathbf{x})$ is a column vector with length ne, containing the shape function values for the probe L. Cubic splines are finally used for the weight functions:

$$
W(\mathbf{x}-\mathbf{x}^k) = \begin{cases} \frac{2}{3} - 4\overline{r}_k^2 + 4\overline{r}_k^3 & \text{for} \;\; \overline{r}_k \leq 0.5 \\ \frac{4}{3} - 4\overline{r}_k + 4\overline{r}_k^2 - \frac{4}{3}\overline{r}_k^3 & \text{for} \;\; 0.5 \leq \overline{r}_k \leq 1.0 \\ 0 & \text{for} \;\; \overline{r}_k \geq 1.0 \end{cases} \tag{3.8}
$$

where $\overline{r}_k = |\mathbf{x}-\mathbf{x}^k|/H_i$.

After evaluating the values of all unknowns at the probe location, the zero pressure gradient in the wall normal direction is imposed by setting $p_I = p_L$ and all of the other variables are linearly interpolated using the prescribed values at the wall and the computed ones at the probe, see Fig. 3.4.

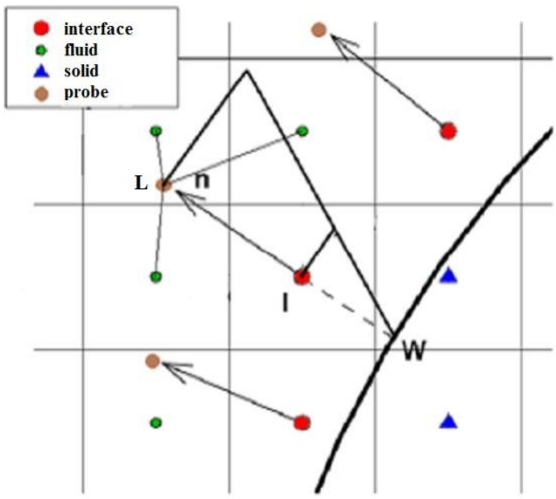


Figure 3.4: Least squares reconstruction scheme.

3.4 Adaptive wall functions for Immersed Boundary

High Reynolds number wall-bounded flows are particularly challenging, as thin turbulent boundary layers develop near wall regions, with large gradients of the flow field variables normal to the surface. For such flows, the representation of the wall boundary has a large impact on the accuracy of the computation, and is also crucial for the robustness and convergence of the flow solver. The local grid refinement approach allows the enhancement of resolution near solid boundaries, and in principle this procedure can be sufficient. In the framework of Cartesian methods, however, there are still many applications for which the correct integration of the flow variables at the wall would require a large number of grid points and thus prohibitive computer resources. In these cases, it would be suitable to employ accurate wall functions. The approach used in this work follows the work by Kalitzin *et al.* [1].

Near-wall behavior

Consider an incompressible flow with constant molecular viscosity. For turbulent flow conditions, the velocity profile can be split into three distinguished regions: the viscous sub-layer, the logarithmic layer and the defect layer. The location of the outer edge of the logarithmic layer depends on the far-field Reynolds number Re_θ as shown in Fig. 3.5 and the extension of the logarithmic layer grows with increasing Reynolds number.

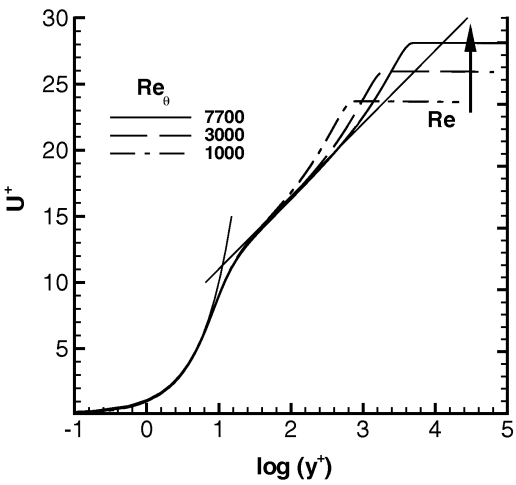


Figure 3.5: Numerical solution for U^+ computed using κ-ω turbulence model with wall integration, for different values of Re_θ, [1].

In a quasi-equilibrium boundary layer (e.g., flow over a flat plate at zero-pressure gradient), the region between the wall and the outer edge of the logarithmic layer has a characteristic behavior, not influenced by the Reynolds number. Defining the friction velocity:

$$u_\tau = \sqrt{\frac{\tau_w}{\rho}}, \tag{3.9}$$

where $\tau_w = \mu(\partial U/\partial y)_w$ is the wall shear stress, and

$$y^+ = \frac{y u_\tau}{\nu} \tag{3.10}$$

the profiles of the flow variables as function of y^+ collapse. This universal law justifies the derivation of wall functions. Near the wall, derivatives in the stream-wise direction can be neglected and the flow and turbulence variables depend only on the coordinate y, which is directed normal to the wall. To derive solutions for the viscous sub-layer and logarithmic layer, the equations are recast in non-dimensional form. The velocity and various turbulence variables in wall units are:

$$U^+ = \frac{U}{u_\tau}, \quad \nu_\tau^+ = \frac{\nu_t}{\nu}, \quad \kappa = \frac{\kappa}{u_\tau^2}, \quad \omega^+ = \frac{\omega\nu}{u_\tau^2}. \tag{3.11}$$

For the flow over a flat plate at zero pressure gradients, the RANS equations simplify in the region between the wall and the outer edge of the logarithmic layer to

$$\frac{d}{dy}\left((\mu + \mu_\tau)\frac{dU}{dy}\right) = 0. \tag{3.12}$$

Integration along the wall normal coordinate y yields

$$(\mu + \mu_\tau)\frac{dU}{dy} = \mu\left(\frac{\partial U}{\partial y}\right)_w = \rho u_\tau^2. \tag{3.13}$$

Introducing the dimensionless variables of Eq. (3.11) in the Eq. (3.13), the last becomes

$$(1 + \nu_t^+)\frac{dU^+}{dy^+} = 1. \tag{3.14}$$

The linear law, $U^+ = y^+$, follows for the viscous sub-layer where $\nu^+ \ll 1$. In the logarithmic layer, ν_t^+ is large and Eq. (3.14) is usually approximated by

$$\nu_t^+\frac{dU^+}{dy^+} = 1. \tag{3.15}$$

Using Prandtls assumption for the turbulent viscosity,

$$\nu_t^+ = A dy^+, \tag{3.16}$$

the logarithmic law $U^+ = (1/A)log(y^+) + B$ follows with the experimentally fitted constants $A = 0.41$ and $B = 5.0$. Suppose that ν_t^+ were known. Then Eq. (3.14) could be integrated to find the universal function $U^+(y^+)$. An early approach was to assume a form for that function. However, it is more consistent to determine the function by solving the wall layer equations numerically. Then, knowing the universal function, the friction velocity can be computed from the velocity at the first grid point if this lies in the wall layer. In fact,

$$Re_y \equiv yU/\nu = y^+U^+(y^+) = F(y^+), \tag{3.17}$$

where Re_y is the Reynolds number based on y and U. The right-hand-side of Eq. (3.17) is a universal function. Inverting this function for the first computed cell close to the wall with $y = y_1$ and $U = U_1$ one has:

$$y_1^+ = y_1 u_\tau/\nu = F^{-1}(Re_y)_1, \tag{3.18}$$

from which u_τ can be found and the characteristics of the boundary layer can be extracted from the universal profile. Given $F(y^+)$, the inversion can be done iteratively by Newtons method. However, this can be done once and for all, and the inverse function stored in a table. This is the method used in the present computations. In the same way, look-up tables for each dimensionless turbulence variable for a well resolved numerical solution are generated.

Numerical implementation

The look-up tables have been created from an accurate numerical solution of a flow over a flat plate at zero-pressure gradient. For variables with large variations in the boundary layer the tables are obtained for their logarithm (e.g., $\log\omega^+$ is reported in place of ω^+), thus reducing the interpolation error. An off-wall boundary condition is assumed for the velocity components as well as for the turbulence variables, enforcing the interface cell center values. For each interface cell (see Fig. 3.6), the tangential component of the velocity, U_2, is evaluated at a certain distance δ from the immersed surface along the local normal direction (this $de\ facto$ corresponds to the creation of a $body - fitted$ grid line).

Knowing the value of $y_2 = \delta$, ν_2 and U_2, the local Reynolds number $Re_{y,2} = y_2U_2/\nu_2$ is computed. A value of u_τ is then obtained from the corresponding look-up tables. From the definition of u_τ (Eq. (3.9)), knowing the interface cell values y_1 and ν_1, the tangential component of the velocity, U_1 can be computed and imposed. The same procedure is

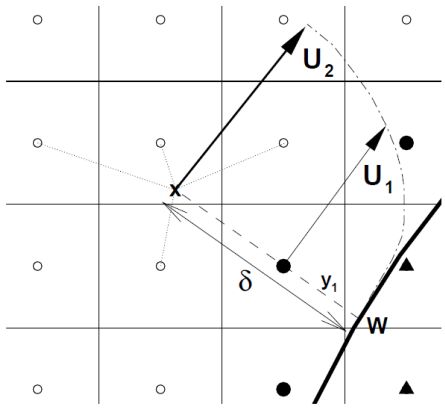


Figure 3.6: Imposition of U_1 in the interface cell using the wall function approach.

used for the turbulence variables, while both the normal velocity and the temperature are linearly interpolated using the value at the wall and at the point 2. Finally a zero pressure gradient in the normal direction is imposed.

It is worth mentioning that for a general geometry, an interpolation procedure is needed for the calculation of the tangential velocity in the point 2 since the wall normal passing through the interface cell does not contain any computational node. So the velocity at this point is computed using the least-squared formula based on the grid points surrounding the interpolation node. Finally, the choice of distance δ is obviously critical for the application of this procedure. The results in this work have been obtained using δ equal to twice the largest distance from the wall in all the interface cells.

3.5 Parallelization procedure

Since the parallel strategy employed for the Cartesian grids provides a row of ghost cells for the communication between two neighbor processors, for the one-dimensional and for the inverse distance weighted reconstruction schemes no additional information are required from the neighbor processors. Instead of the first two reconstruction, in the case of the least squares reconstruction, and more in general for all the procedures involving the probe points, additional information are required because the probe points can lie inside a computational cell belonging to a different processor-domain. Thus each processor reconstructs the variables of each probe that lie in its domain and then sends these information to the processor that contains the interface cell related to the probe.

Chapter 4

Numerical Solvers

4.1 Cartesian grid Solver

As shown in the Chapter 3, the Cartesian grid is divided in fluid zones and solid zones, where the equations (2.23), (2.24), (2.25) and the equation (2.37) are to be solved, respectively. Following the idea of Iaccarino *et al.* [56] the solid heat transfer equation (2.37) is included in the equation (2.25), deleting the convective and turbulent terms only while solving the solid zones. The numerical advantage of this approach is the reduced time for each iteration, because basically the overall system has one less equation, while more difficulties in setting a time step only for solid may be encountered. Finally in this way the same spatial discretization and time-marching numerical algorithm can be used for both fluid and solid zones.

For numerical solutions, it is convenient to recast the equations (2.23), (2.24), (2.25) into a compact form:

$$\frac{\partial Q}{\partial t} + \frac{\partial (E - E_v)}{\partial x} + \frac{\partial (F - F_v)}{\partial y} + \frac{\partial (G - G_v)}{\partial z} = H_r, \tag{4.1}$$

where Q is the vector of primary conservative variables:

$$Q = (\rho, \rho u, \rho v, \rho w, \rho \tilde{H} - p_t, \rho k, \rho \omega)^T, \tag{4.2}$$

E,F and G are the inviscid flux vectors, H_r is the source terms vector:

$$E = \begin{pmatrix} \rho u \\ \rho u^2 + p_t \\ \rho uv \\ \rho uw \\ \rho \tilde{H} u \\ \rho uk \\ \rho u\omega \end{pmatrix}, \quad F = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p_t \\ \rho vw \\ \rho \tilde{H} v \\ \rho vk \\ \rho v\omega \end{pmatrix}, \quad G = \begin{pmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p_t \\ \rho \tilde{H} w \\ \rho wk \\ \rho w\omega \end{pmatrix}, \quad H_r = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ S_k \\ S_\omega \end{pmatrix}, \tag{4.3}$$

and the viscous flux vectors E_v,F_v and G_v are written in terms of a viscous set of dependent variables

$$Q_v = (p_t, u, v, w, T, k, \omega)^T, \tag{4.4}$$

and coefficient matrices such that (Schwer [57]):

$$E_v = \mathbf{R}_{xx}\frac{\partial Q_v}{\partial x} + \mathbf{R}_{xy}\frac{\partial Q_v}{\partial y} + \mathbf{R}_{xz}\frac{\partial Q_v}{\partial z}, \tag{4.5}$$

$$F_v = \mathbf{R}_{yx}\frac{\partial Q_v}{\partial x} + \mathbf{R}_{yy}\frac{\partial Q_v}{\partial y} + \mathbf{R}_{yz}\frac{\partial Q_v}{\partial z}, \tag{4.6}$$

$$G_v = \mathbf{R}_{zx}\frac{\partial Q_v}{\partial x} + \mathbf{R}_{zy}\frac{\partial Q_v}{\partial y} + \mathbf{R}_{zz}\frac{\partial Q_v}{\partial z}. \tag{4.7}$$

The viscous stress tensors $\mathbf{R}_{ij}$ are included in Appendix B.

From the definition of p_t (Equation (2.29)):

$$dp_t = dp + \frac{2}{3}kd\rho + \frac{2}{3}\rho dk. \tag{4.8}$$

Now we can write the following functional forms:

$$\rho = \rho(p, T), \qquad h = h(p, T), \tag{4.9}$$

whose differential read:

$$d\rho = \rho_p dp + \rho_T dT, \qquad dh = h_p dp + h_T dT \tag{4.10}$$

where the subscripts indicate partial derivative operator. Substituting the equation for $d\rho$ Equation (4.10) into Equation (4.8) we get:

$$dp_t = \left(1 + \frac{2}{3}k\rho_p\right) dp + \frac{2}{3}\rho dk + \frac{2}{3}k\rho_T dT. \tag{4.11}$$

From the definition of derivative,

$$\tilde{\rho}_{p_t} = \lim_{\substack{dp_t \to 0 \\ T, k = const}} \left\{ \frac{d\rho}{dp_t} = \rho_p \frac{dp}{dp_t} + \rho_T \frac{dT}{dp_t} \right\}, \tag{4.12}$$

but $dT/dp_t = 0$, so that:

$$\tilde{\rho}_{p_t} = \rho_p \left(\frac{dp}{dp_t}\right)_{T,k}. \tag{4.13}$$

Notice that the derivatives of the flow variables are designated with $\tilde{()}$, meaning that they are functions of the set of primitive variables p_t, T, k. From Equation (4.11):

$$\left(\frac{dp}{dp_t}\right)_{T,k} = \frac{1}{1 + \frac{2}{3}k\rho_p}, \tag{4.14}$$

resulting in the expression:

$$\tilde{\rho}_{p_t} = \frac{\rho_p}{1 + \frac{2}{3}k\rho_p}. \tag{4.15}$$

The remaining derivatives are evaluated in a similar manner and are given by the following expressions:

$$\tilde{\rho}_T = \frac{\rho_T}{1 + \frac{2}{3}k\rho_p}, \qquad \tilde{\rho}_k = \frac{-\frac{2}{3}\rho\rho_p}{1 + \frac{2}{3}k\rho_p}, \tag{4.16}$$

$$\tilde{h}_{p_t} = \frac{h_p}{1 + \frac{2}{3}k\rho_p}, \qquad \tilde{h}_T = h_T - \frac{\frac{2}{3}k\rho_T h_p}{1 + \frac{2}{3}k\rho_p}, \qquad \tilde{h}_k = \frac{-\frac{2}{3}\rho h_p}{1 + \frac{2}{3}k\rho_p}. \tag{4.17}$$

Now we introduce the variable $\tilde{c}^2$ which is equal to the speed of sound squared c^2 plus a contribution from the turbulent kinetic energy:

$$\tilde{c}^2 = \frac{\rho\tilde{h}_T}{\rho\tilde{\rho_{p_t}}\tilde{h}_T + \tilde{\rho}_T(1 - \rho\tilde{h}_{p_t})} = \frac{\rho h_T}{\rho\rho_p h_T + \rho_T(1 - \rho h_p)} + \frac{\frac{2}{3}k\rho(\rho_p h_T - \rho_T h_p)}{\rho\rho_p h_T + \rho_T(1 - \rho h_p)}. \tag{4.18}$$

For a perfect gas, Equation (4.18) reduces to:

$$\tilde{c}^2 = \gamma RT + \frac{2}{3}\gamma k = \gamma\left(R + \frac{2}{3}\frac{k}{T}\right)T = \gamma\tilde{R}T. \tag{4.19}$$

Finally, Equation (4.15) can be rewritten as follows:

$$\tilde{\rho}_{p_t} = \frac{1}{\tilde{c}^2} - \frac{\tilde{\rho}_T(1 - \rho\tilde{h}_{p_t})}{\rho\tilde{h}_T}, \tag{4.20}$$

Now we consider the inviscid form of the equations, to study some characteristics of the conservation equations. Setting the viscous variables Q_v as our primary variables and using chain rule, equation (4.1) is written as:

$$\mathbf{P}\frac{\partial Q_v}{\partial t} + \mathbf{A}_v\frac{\partial Q_v}{\partial x} + \mathbf{B}_v\frac{\partial Q_v}{\partial y} + \mathbf{C}_v\frac{\partial Q_v}{\partial z} = 0, \tag{4.21}$$

where $\mathbf{P} = \partial Q/\partial Q_v$, $\mathbf{A}_v = \partial E/\partial Q_v$, $\mathbf{B}_v = \partial F/\partial Q_v$ and $\mathbf{C}_v = \partial G/\partial Q_v$.

The form of the the matrix $\mathbf{P}$ is the following:

$$\mathbf{P} = \begin{bmatrix} \tilde{\rho}_{p_t} & 0 & 0 & 0 & \tilde{\rho}_T & \tilde{\rho}_k & 0 \\ \tilde{\rho}_{p_t}u & \rho & 0 & 0 & \tilde{\rho}_T u & \tilde{\rho}_k u & 0 \\ \tilde{\rho}_{p_t}v & 0 & \rho & 0 & \tilde{\rho}_T v & \tilde{\rho}_k v & 0 \\ \tilde{\rho}_{p_t}w & 0 & 0 & \rho & \tilde{\rho}_T w & \tilde{\rho}_k w & 0 \\ \tilde{H}\tilde{\rho}_{p_t} + \rho\tilde{h}_{p_t} - 1 & \rho u & \rho v & \rho w & \tilde{H}\tilde{\rho}_T + \rho\tilde{h}_T & \tilde{H}\tilde{\rho}_k + \rho\tilde{h}_k + \frac{5}{3}\rho & 0 \\ \tilde{\rho}_{p_t}k & 0 & 0 & 0 & \tilde{\rho}_T k & \rho + k\tilde{\rho}_k & 0 \\ \tilde{\rho}_{p_t}\omega & 0 & 0 & 0 & \tilde{\rho}_T\omega & \tilde{\rho}_k\omega & \rho \end{bmatrix}. \tag{4.22}$$

We can individually diagonalize the Jacobians $\mathbf{P}^{-1}\mathbf{A}_v$ for the x-direction, $\mathbf{P}^{-1}\mathbf{B}_v$ for the y-direction, or $\mathbf{P}^{-1}\mathbf{C}_v$ for the z-direction with the eigenvectors matrix $\mathbf{M}_x$,$\mathbf{M}_y$, or $\mathbf{M}_z$, respectively, but we can not simultaneously diagonalize any pair of them. Because the Euler equations are hyperbolic in nature, the eigenvectors and eigenvalues associated with this system are real. This results in a diagonal matrix $\mathbf{\Lambda}_x$,$\mathbf{\Lambda}_y$, or $\mathbf{\Lambda}_z$, representing the propagation speeds of the characteristics variables $\mathbf{M}_x^{-1}dQ_v$, $\mathbf{M}_y^{-1}dQ_v$, or $\mathbf{M}_z^{-1}dQ_v$. For instance, diagonalizing in the x-direction results in the transformed equation:

$$\mathbf{M}_x^{-1}\frac{\partial Q_v}{\partial t} + \mathbf{\Lambda}_x\mathbf{M}_x^{-1}\frac{\partial Q_v}{\partial x} + \mathbf{M}_x^{-1}\mathbf{P}^{-1}\mathbf{B}_v\frac{\partial Q_v}{\partial y} + \mathbf{M}_x^{-1}\mathbf{P}^{-1}\mathbf{C}_v\frac{\partial Q_v}{\partial z} = 0, \tag{4.23}$$

where $\mathbf{\Lambda}_x$ is

$$\mathbf{\Lambda}_x = \text{diag}\,(\lambda_1, \lambda_2, \lambda_3, ...) = \text{diag}\,(u, u, u, u + \tilde{c}, u - \tilde{c}, u, u), \tag{4.24}$$

where c is the speed of sound. For each Jacobian matrix ($\mathbf{A}_v$, $\mathbf{B}_v$ and $\mathbf{C}_v$) we have three propagation velocities: a particle velocity u and two acoustic velocities $u \pm \tilde{c}$. These characteristic velocities play an important role in the overall convergence and dissipation properties, for different schemes. The characteristic variables in the x-direction are:

$$\mathbf{M}_x^{-1}dQ_v = \begin{pmatrix} dp_t - \left(\frac{\rho\tilde{h}_T}{1-\rho\tilde{h}_p}\right)dT \\ -\frac{\sqrt{2}}{2}\left(\rho dv - \rho dw\right) \\ -\frac{\sqrt{2}}{2}\left(\rho dv + \rho dw\right) \\ dp_t + \rho\tilde{c}du \\ dp_t - \rho\tilde{c}du \\ \rho dk \\ \rho d\omega \end{pmatrix}. \tag{4.25}$$

The first three characteristic variables all travel at the particle velocity u. The first is entropy, the second and third are transverse velocities. The fourth characteristic variable, called the p wave, is an acoustic wave and travels at the velocity $u + c$. The fifth characteristic variable, called the q wave, is also an acoustic wave and travels at the velocity $u - c$. Both eigenvalues and eigenvectors are illustrated in Appendix C.

This analysis is the starting point for increasing the convergence rate through preconditioning.

4.1.1 Solution procedure for unsteady problems

Because of the complexity of the flow fields we want to capture, we apply a three-point fully-implicit time marching procedure to the Equations (4.1):

$$\frac{3Q^{n+1} - 4Q^n + Q^{n-1}}{2\Delta t} + \left[\frac{\partial(E - E_v)}{\partial x} + \frac{\partial(F - F_v)}{\partial y} + \frac{\partial(G - G_v)}{\partial z} - H_r\right]^{n+1} = 0, \quad (4.26)$$

where Δt is the time step. In this equations, the solution at time level $n+1$ is unknown, whereas that at time levels n and $n-1$ are known. The flux vectors (E, E_v, F, F_v, G, G_v) as well as the source-terms vector H_r are all specified in terms of the solution vector Q_v but are non-linear in Q_v. Because of the non-linear nature of these equations, an iterative solution procedure must be used in order to determine the solution vector Q_v at time level $n+1$ satisfying Equation (4.26). An additional problem with the above equations is that even if (E, E_v, F, F_v, G, G_v) were linear, we would end up with a large size sparse matrix which is very difficult and time consuming to invert, this requiring some sort of factorization scheme.

The iteration methods to solve the above equations have become popular, and are covered in detail in the works by Chakravarthy and Osher [58] and Pulliam [59].

Dual Time Stepping

The approach chosen in this work for iterating to the correct solution Q_v at time level $n+1$ is the *Dual-Time Stepping* (DTS) scheme. This scheme is similar to the approximate-Newton method, but has an additional sink term that can be controlled to optimize convergence. In fact, the iterative method is treated as a time-marching method based on a pseudo-time, and in this formulation, the physical-time derivative becomes a sink in the pseudo-time frame. The total system can be viewed as a steady-state calculation with a sink term dependent on the physical-time step. For large physical-time steps, the sink term is small, the problem behaves like a steady-state problem and the Courant-Friedrichs-Lewy number based on the pseudo-time step (pseudo-CFL) should be set corresponding to the steady-state optimal values. For small time steps, the problem is sink dominated and the pseudo-CFL can be increased. Unlike the physical-time step, the pseudo-time step can be set locally depending on the local flow conditions and physical-time steps. This is especially useful for grids that include sparse regions where the CFL based on the physical time step is fairly low and stretched regions where the physical CFL is fairly high.

Since the Euler method works well for time-marching to a steady-state solution, because

of its good dissipation properties as the time step is increased (it typically damps out any unsteadiness for large values of the time step), we apply the DTS to the equations (4.26), using an Euler implicit procedure in the pseudo-time:

$$\begin{aligned}&\frac{Q_v^{n+1,m+1}-Q_v^{n+1,m}}{\Delta\tau}+\frac{3Q^{n+1,m+1}-4Q^n+Q^{n-1}}{2\Delta t}+\\&+\left[\frac{\partial(E-E_v)}{\partial x}+\frac{\partial(F-F_v)}{\partial y}+\frac{\partial(G-G_v)}{\partial z}-H_r\right]^{n+1,m+1}=0,\end{aligned}\tag{4.27}$$

where $\Delta\tau$ is the pseudo-time step, and n and m indicate the physical and pseudo-time levels respectively.

A linearization procedure is used in the pseudo-time to express the fluxes and the other quantities in terms of the primary set of dependent variables Q_v at step $n+1, m+1$:

$$E^{n+1,m+1}=E^{n+1,m}+\left(\frac{\partial E}{\partial Q_v}\right)^{n+1,m}\Delta Q_v=E^{n+1,m}+\mathbf{A}_v^{n+1,m}\Delta Q_v,\tag{4.28}$$

$$\begin{aligned}&E_v^{n+1,m+1}=E_v^{n+1,m}+\left(\frac{\partial E_v}{\partial Q_v}\right)^{n+1,m}\Delta Q_v=\\&E_v^{n+1,m}+\left(\mathbf{R_{xx}}^{n+1,m}\frac{\partial}{\partial x}+\mathbf{R_{xy}}^{n+1,m}\frac{\partial}{\partial y}+\mathbf{R_{xz}}^{n+1,m}\frac{\partial}{\partial z}\right)\Delta Q_v,\end{aligned}\tag{4.29}$$

$$Q^{n+1,m+1}=Q^{n+1,m}+\left(\frac{\partial Q}{\partial Q_v}\right)^{n+1,m}\Delta Q_v=Q^{n+1,m}+\mathbf{P}^{n+1,m}\Delta Q_v,\tag{4.30}$$

$$H_r^{n+1,m+1}=H_r^{n+1,m}+\left(\frac{\partial H_r}{\partial Q_v}\right)^{n+1,m}\Delta Q_v=H_r^{n+1,m},\tag{4.31}$$

where

$$\Delta Q_v=Q_v^{n+1,m+1}-Q_v^{n+1,m}.\tag{4.32}$$

The terms $F^{n+1,m+1}$, $G^{n+1,m+1}$, $F_v^{n+1,m+1}$ and $G_v^{n+1,m+1}$ are obtained in a similar way. Premultiplying the pseudo-time derivative through a preconditioning matrix $\mathbf{\Gamma}$ for optimum convergence in the inner iterations, multiplying all the terms through the pseudo-time step $\Delta\tau$, and rewriting the equations in delta-form (omitting the superscripts of the Jacobians for clearness):

$$\begin{aligned}&\left[\mathbf{\Gamma}+\frac{3}{2}\frac{\Delta\tau}{\Delta t}\mathbf{P}+\Delta\tau\frac{\partial}{\partial x}\left(\mathbf{A}_v-\mathbf{R}_{xx}\frac{\partial}{\partial x}-\mathbf{R}_{xy}\frac{\partial}{\partial y}-\mathbf{R}_{xz}\frac{\partial}{\partial z}\right)\right.\\&+\Delta\tau\frac{\partial}{\partial y}\left(\mathbf{B}_v-\mathbf{R}_{yx}\frac{\partial}{\partial x}-\mathbf{R}_{yy}\frac{\partial}{\partial y}-\mathbf{R}_{yz}\frac{\partial}{\partial z}\right)\\&\left.+\Delta\tau\frac{\partial}{\partial z}\left(\mathbf{C}_v-\mathbf{R}_{zx}\frac{\partial}{\partial x}-\mathbf{R}_{zy}\frac{\partial}{\partial y}-\mathbf{R}_{zz}\frac{\partial}{\partial z}\right)\right]\Delta Q_v=\\&-\Delta\tau\left[\frac{3Q^{n+1,m}-4Q^n+Q^{n-1}}{2\Delta t}+\mathcal{R}^{n+1,m}\right],\end{aligned}\tag{4.33}$$

where

$$\mathcal{R}^{n+1,m} = \left[\frac{\partial(E-E_v)}{\partial x} + \frac{\partial(F-F_v)}{\partial y} + \frac{\partial(G-G_v)}{\partial z} - H_r\right]^{n+1,m}. \tag{4.34}$$

This procedure maintains the fully non-linear method in the physical-time. To recover the non-dual-time steady-state form, we simply set the physical-time step to infinity and converge in one physical-time step. However, with a poor initial solution, a finite physical-time step typically helps to stabilize the numerical solution and thus enhances the robustness of the code.

Preconditioning

The form of the preconditioning matrix $\mathbf{\Gamma}$ used in the code is based on the generalized-equation-of-state preconditioner introduced by Merkle [60], which is based on the time-accurate Jacobian $\mathbf{P}$ introduced in Equation (4.22) with a modified density derivative $\tilde{\rho}'_{p_t}$:

$$\mathbf{\Gamma} = \begin{bmatrix} \tilde{\rho}'_{p_t} & 0 & 0 & 0 & \tilde{\rho}_T & \tilde{\rho}_k & 0 \\ \tilde{\rho}'_{p_t}u & \rho & 0 & 0 & \tilde{\rho}_T u & \tilde{\rho}_k u & 0 \\ \tilde{\rho}'_{p_t}v & 0 & \rho & 0 & \tilde{\rho}_T v & \tilde{\rho}_k v & 0 \\ \tilde{\rho}'_{p_t}w & 0 & 0 & \rho & \tilde{\rho}_T w & \tilde{\rho}_k w & 0 \\ \tilde{H}\tilde{\rho}'_{p_t} + \rho\tilde{h}_{p_t} - 1 & \rho u & \rho v & \rho w & \tilde{H}\tilde{\rho}_T + \rho\tilde{h}_T & \tilde{H}\tilde{\rho}_k + \rho\tilde{h}_k + \frac{5}{3}\rho & 0 \\ \tilde{\rho}'_{p_t}k & 0 & 0 & 0 & \tilde{\rho}_T k & \rho + k\tilde{\rho}_k & 0 \\ \tilde{\rho}'_{p_t}\omega & 0 & 0 & 0 & \tilde{\rho}_T \omega & \tilde{\rho}_k \omega & \rho \end{bmatrix}. \tag{4.35}$$

The effect of steady-state preconditioning is to modify the speed of sound such that all of the eigenvalues of the system $\mathbf{\Gamma}^{-1}\mathbf{A}_v$ are of a similar order. This prevents the stiffness problem common for factorized schemes at low Mach numbers. Viscous preconditioning ensures that the inviscid and viscous time scales are similar in low Reynolds number computational cells. Time-accurate flow fields present a further problem. For very small physical-time steps, the solution is sink dominated and best convergence can be obtained only if $\mathbf{\Gamma}$ is equal to $\mathbf{P}$ ($\tilde{\rho}'_{p_t} = \tilde{\rho}_{p_t}$), regardless of the Mach number. However, for very large time steps the sink becomes very weak and the steady-state preconditioner gives

the best convergence rates. Extensive research has been done for determining appropriate expressions for the preconditioning parameter ($\tilde{\rho}'_{p_t}$) for the steady-state inviscid (Choi and Merkle [61], Turkel [62], van Leer, Lee and Roe [63]), and viscous (Choi and Merkle [64]) flow fields as well as for time-accurate flow fields (Venkateswaran and Merkle [65]). Following the work of Venkateswaran [66], [65] and Merkle [60], the preconditioning parameter is given by the following formula:

$$\tilde{\rho}'_{p_t} = \frac{1}{V_r^2} - \frac{\tilde{\rho}_T(1 - \rho\tilde{h}_{p_t})}{\rho\tilde{h}_T}, \tag{4.36}$$

The term V_r in Equation (4.36) is a reference velocity that is chosen to appropriately precondition the relevant time scales. The non-preconditioned $\tilde{\rho}'_{p_t} = \tilde{\rho}_{p_t}$ can be recovered by setting $V_r^2 = \tilde{c}^2$.

One possible way of choosing V_r (Buelow et al. [67]) is:

$$V_r^2 = \min\left[\tilde{c}^2, \max\left(V_{inv}^2, V_{vis}^2, V_{unsteady}^2\right)\right]. \tag{4.37}$$

The *inviscid* velocity scale is given by $V_{inv}^2 \sim M^2c^2$, where M is the Mach number. The *viscous* velocity scale is given by $V_{vis}^2 \sim V_{inv}^2/Re_{\Delta x}^2$, where $Re_{\Delta x}$ is the cell Reynolds number. Venkateswaran and Merkle [65], by analyzing the exact stability solution of the one-dimensional dual-time stepping Euler equations, determined an expression for the *unsteady* velocity scale based on characteristic length scales and the physical-time step:

$$V_{unsteady}^2 \sim \left(\frac{L_x}{\Delta t}\right)^2 + \left(\frac{L_y}{\Delta t}\right)^2 + \left(\frac{L_z}{\Delta t}\right)^2. \tag{4.38}$$

The optimal value for the length scales L_x,L_y and L_z are dependent on the specific problem; however, they are always of the same order of the characteristic length of the domain. The definition for $\tilde{\rho}'_{p_t}$ differs slightly from definitions given by Merkle and Venkateswaran, $\tilde{\rho}'_{p_t} = 1/V_r^2$. The definition in Equation (4.36) was chosen because in the limit as the Mach number approaches one, this definition smoothly tends to the non-preconditioned equations exactly, whereas the Mach number becomes very small it approaches the same value given by Merkle and Venkateswaran. For an ideal gas, where $\tilde{h}_{p_t} = 0$, the expression becomes:

$$\tilde{\rho}'_{p_t} = \frac{1}{\tilde{c}^2}\left[\frac{\tilde{c}^2}{V_r^2} - 1\right] + \tilde{\rho}_{p_t}. \tag{4.39}$$

The bracketed term is always greater than zero because $\tilde{c}^2/V_r^2$ (from Equation (4.37)) is always greater than or equal to one, thus $\tilde{\rho}'_{p_t}$ is always greater than $\tilde{\rho}_{p_t}$. The conditions

for setting the appropriate velocity scale for inviscid and viscous equations is covered in detail in Buelow [68].

4.1.2 Finite-volume approach

A collocated cell-centered finite volume space discretization is used. The convective terms in the residual $\mathcal{R}$ in Equation (4.34) are discretized using either an upwind flux-difference-splitting scheme with first and second order of accuracy, or a second-order-accurate centered scheme. The viscous terms are discretized by second-order-accurate centered differences. The convective terms in the left-hand-side (LHS) of Equation (4.34), are always discretized using a first-order upwind scheme, according to a deferred-correction approach, in order to ensure convergence of the iterative solver.

Discretization of the residual $\mathcal{R}$ terms

In the present finite-volume methodology, for each face, the contributions of the neighbor cells in each Cartesian direction are collected to build the corresponding convective and dissipative operators for the cell.

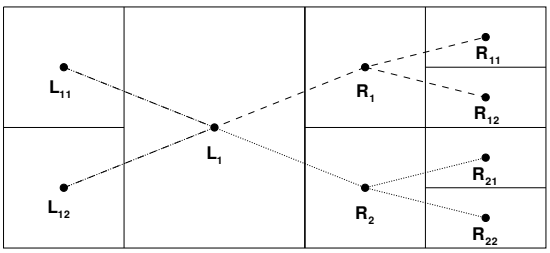


Figure 4.1: Flux balance in the x-direction (east side) of the cell L_1.

For example, considering the convective fluxes in the x direction, referring to Figure 4.1, for the cell L_1 we have:

$$\int_V \frac{\partial E}{\partial x} \delta V = (E^r_{L_1} - E^l_{L_1}) S_x, \tag{4.40}$$

where r and l indicate the right and left faces of each computational cell and the evaluation

of $E^r_{L_1}$ is carried out in the following way:

$$E^r_{L_1} = \frac{1}{N^r_{L_1}} \sum_{j=1}^{N^r_{L_1}} \left\{ \frac{1}{2}(E_{R_j} + E_{L_1}) - \phi_1 \left(\mathbf{\Gamma M}_x |\mathbf{\Lambda}_x|\right)^r_{L_1,j} \left(\mathbf{M}_x^{-1}\right)^r_{L_1,j} (Q_{v,R_j} - Q_{v,L_1}) \right.$$
$$- \frac{\phi_2}{N^l_{L_1}} \left(\mathbf{\Gamma M}_x \mathbf{\Lambda}^+_x\right)^r_{L_1,j} \left[\sum_{i=1}^{N^l_{L_1}} \left(\mathbf{M}_x^{-1}\right)^l_{L_1,i} (Q_{v,L_1} - Q_{v,L_{1i}}) \right] \tag{4.41}$$
$$\left. + \frac{\phi_2}{N^r_{R_j}} \left(\mathbf{\Gamma M}_x \mathbf{\Lambda}^-_x\right)^r_{L_1,j} \left[\sum_{i=1}^{N^r_{R_j}} \left(\mathbf{M}_x^{-1}\right)^r_{R_j,i} (Q_{v,R_{ji}} - Q_{v,R_j}) \right] \right\}.$$

In Equation (4.41), R and L indicate the cell centers at the right and left sides of the face, respectively, one and two subscripts being employed for the first and second row of cells (see Figure 4.1). N^y_X is the number of the neighboring cells of the cell X on the y side. The matrices $\mathbf{\Lambda}^\pm_x$ are defined as:

$$\mathbf{\Lambda}^+_x = \frac{1}{2}(\mathbf{\Lambda}_x + |\mathbf{\Lambda}_x|), \qquad \mathbf{\Lambda}^-_x = \frac{1}{2}(\mathbf{\Lambda}_x - |\mathbf{\Lambda}_x|). \tag{4.42}$$

Matrices $\mathbf{\Gamma}, \mathbf{M}_x, \mathbf{\Lambda}^\pm_x$ and $\mathbf{M}_x^{-1}$ are calculated using the averaged variables according to Roe's linearization and the values of ϕ_1 and ϕ_2 are provided in Table 4.1 for different orders of accuracy. A similar approach is employed to build the centered diffusive operator. When computing flows with shocks, a total variation diminishing approach, with minmod limiter function, is employed in conjunction with the second order upwind scheme.

order	ϕ_1	ϕ_2
1st-upwind	1/2	0
2nd-upwind	1/2	-1/2
2nd-centered	0	0

Table 4.1: Coefficients for different orders of accuracy.

In the same way, considering the dissipative fluxes in the x direction, we have:

$$\int_V \frac{\partial E_v}{\partial x} \delta V = (E^r_{v_{L_1}} - E^l_{v_{L_1}}) S_x, \tag{4.43}$$

where r and l indicate the right and left faces of each computational cell and the evaluation of $E^r_{v_{L_1}}$ is carried out in the following way:

$$E^r_{v_{L_1}} = \frac{1}{N^r_{L_1}} \sum_{j=1}^{N^r_{L_1}} \left(\mathbf{R_{xx_j}} \frac{\left(Q_{v_{R_j}} - Q_{v_{L_1}}\right)}{x_{R_j} - x_{L_1}} + \mathbf{R_{xy_j}} \left(\frac{\partial Q_v}{\partial y}\right)_j + \mathbf{R_{xz_j}} \left(\frac{\partial Q_v}{\partial z}\right)_j \right) \tag{4.44}$$

where $\mathbf{R_{xx_j}}$, $\mathbf{R_{xy_j}}$,$\mathbf{R_{xz_j}}$, $\left(\frac{\partial Q_v}{\partial z}\right)_j$, $\left(\frac{\partial Q_v}{\partial z}\right)_j$ are calculated using the averaged variables according to Roe's linearization.

Discretization of the LHS terms

The left-hand-side (LHS) of Equation (4.33) is modified to improve the efficiency of the method, without affecting the residual, that is, the physical solution.

A significant reduction in operation count and CPU time can be accomplished through a diagonalization procedure used in conjunction with ADI first introduced by Pulliam and Chaussee [69] and extended to dual-time stepping schemes by DeRango and Zingg [70] and Buelow *et al.* [67]. For the inviscid steady-state equations without source/sink terms, the diagonalization is straightforward. However, for the dual-time stepping formulation there is always a sink term $\frac{3}{2}\frac{\Delta\tau}{\Delta t}\mathbf{P}$. Following the work by Schwer [57], we group the preconditioning and physical-time Jacobians together into a single term $\mathbf{S} = \mathbf{\Gamma} + \frac{3}{2}\frac{\Delta\tau}{\Delta t}\mathbf{P}$ and then diagonalize the LHS with respect to this term. In order for this procedure to work efficiently, $\mathbf{S}$ must be easily invertible and must have easily obtained eigenvectors and eigenvalues. Because $\mathbf{\Gamma}$ and $\mathbf{P}$ are almost identical, $\mathbf{S}$ can be written in the form:

$$\mathbf{S} = \left(1 + \frac{3}{2}\frac{\Delta\tau}{\Delta t}\right)\begin{bmatrix} \tilde{\rho}''_{p_t} & 0 & 0 & 0 & \tilde{\rho}_T & \tilde{\rho}_k & 0 \\ \tilde{\rho}''_{p_t}u & \rho & 0 & 0 & \tilde{\rho}_T u & \tilde{\rho}_k u & 0 \\ \tilde{\rho}''_{p_t}v & 0 & \rho & 0 & \tilde{\rho}_T v & \tilde{\rho}_k v & 0 \\ \tilde{\rho}''_{p_t}w & 0 & 0 & \rho & \tilde{\rho}_T w & \tilde{\rho}_k w & 0 \\ \tilde{H}\tilde{\rho}''_{p_t} + \rho\tilde{h}_{p_t} - 1 & \rho u & \rho v & \rho w & \tilde{H}\tilde{\rho}_T + \rho\tilde{h}_T & \tilde{H}\tilde{\rho}_k + \rho\tilde{h}_k + \frac{5}{3}\rho & 0 \\ \tilde{\rho}''_{p_t}k & 0 & 0 & 0 & \tilde{\rho}_T k & \rho + k\tilde{\rho}_k & 0 \\ \tilde{\rho}''_{p_t}\omega & 0 & 0 & 0 & \tilde{\rho}_T\omega & \tilde{\rho}_k\omega & \rho \end{bmatrix}, \tag{4.45}$$

where

$$\tilde{\rho}''_{p_t} = \frac{\tilde{\rho}'_{p_t} + \frac{3}{2}\frac{\Delta\tau}{\Delta t}\tilde{\rho}_{p_t}}{1 + \frac{3}{2}\frac{\Delta\tau}{\Delta t}}. \tag{4.46}$$

Because $\mathbf{S}$ has the same form as $\mathbf{\Gamma}$ and $\mathbf{P}$, the eigenvectors and eigenvalues also take the same form, yielding an easily computed set of real eigenvalues and eigenvectors.

Now, the scheme is obtained firstly neglecting the non-orthogonal viscous coefficient matrices, $\mathbf{R}_{xy}$, $\mathbf{R}_{xz}$, $\mathbf{R}_{yx}$,$\mathbf{R}_{yz}$,$\mathbf{R}_{zx}$ and $\mathbf{R}_{zy}$, and then factoring out the matrix $\mathbf{S}$ from the

LHS:

$$\mathbf{S}\Bigg[\mathbf{I} + \Delta\tau\mathbf{S}^{-1}\frac{\partial}{\partial x}\left(\mathbf{A}_v - \mathbf{R}_{xx}\frac{\partial}{\partial x}\right) + \Delta\tau\mathbf{S}^{-1}\frac{\partial}{\partial y}\left(\mathbf{B}_v - \mathbf{R}_{yy}\frac{\partial}{\partial y}\right) + \Delta\tau\mathbf{S}^{-1}\frac{\partial}{\partial z}\left(\mathbf{C}_v - \mathbf{R}_{zz}\frac{\partial}{\partial z}\right)\Bigg]\Delta Q_v = -\Delta\tau\left[\frac{3Q^{n+1,m} - 4Q^n + Q^{n-1}}{2\Delta t} + \mathcal{R}^{n+1,m}\right]. \tag{4.47}$$

Diagonalizing following the procedure by Pulliam and Chaussee [69], the matrices $\mathbf{S}^{-1}\mathbf{A}_v$, $\mathbf{S}^{-1}\mathbf{B}_v$ and $\mathbf{S}^{-1}\mathbf{C}_v$ can be written as:

$$\mathbf{S}^{-1}\mathbf{A}_v = \mathbf{M}_x\mathbf{\Lambda}_x\mathbf{M}_x^{-1}. \tag{4.48}$$

$$\mathbf{S}^{-1}\mathbf{B}_v = \mathbf{M}_y\mathbf{\Lambda}_y\mathbf{M}_y^{-1}. \tag{4.49}$$

$$\mathbf{S}^{-1}\mathbf{C}_v = \mathbf{M}_z\mathbf{\Lambda}_z\mathbf{M}_z^{-1}. \tag{4.50}$$

For the inviscid equations, the eigenvector matrices can be factored out of the implicit operator. The viscous terms, however, present another difficulty. The linearized viscous coefficient matrix $\mathbf{R}_{ij}$ is not diagonalized by the same eigenvectors as the inviscid flux terms. The current method for incorporating viscous effects into the LHS is to replace the viscous coefficient matrix with its spectral radius R_s times the identity matrix. The spectral radius is defined as below (Schwer [57]):

$$R_s = \max\left(\frac{\mu + \mu_t}{\rho}, \frac{\tilde{\rho}''_{p_t}K}{\rho\tilde{\rho}''_{p_t}\tilde{h}_T + \tilde{\rho}_T(1 - \rho\tilde{h}_{p_t})}\right). \tag{4.51}$$

The diagonalized scheme is then written as:

$$\mathbf{S}\Bigg[\mathbf{I} + \Delta\tau\mathbf{M}_x\frac{\partial}{\partial x}\left(\mathbf{\Lambda}_x - R_s\mathbf{I}\frac{\partial}{\partial x}\right)\mathbf{M}_x^{-1} + \Delta\tau\mathbf{M}_y\frac{\partial}{\partial y}\left(\mathbf{\Lambda}_y - R_s\mathbf{I}\frac{\partial}{\partial y}\right)\mathbf{M}_y^{-1} + \Delta\tau\mathbf{M}_z\frac{\partial}{\partial z}\left(\mathbf{\Lambda}_z - R_s\mathbf{I}\frac{\partial}{\partial z}\right)\mathbf{M}_z^{-1}\Bigg]\Delta Q_v = -\Delta\tau\left[\frac{3Q^{n+1,m} - 4Q^n + Q^{n-1}}{2\Delta t} + \mathcal{R}^{n+1,m}\right]. \tag{4.52}$$

One potential drawback to the above method for diagonalizing the LHS is that the modal analysis, based on $\mathbf{S}^{-1}\mathbf{A}_v$, appears to be inconsistent with that used for the RHS dissipation terms, based on $\mathbf{\Gamma}^{-1}\mathbf{A}_v$. Due to the nature of the preconditioning, this does not appear to be a problem. Examining Equation (4.45) and Equation (4.35), the only difference between $\mathbf{S}$ and $\mathbf{\Gamma}$ is a scalar multiplier and the definition of $\tilde{\rho}''_{p_t}$ found in Equation (4.46). Since preconditioning is employed under low Mach number conditions, there are primarily two limiting cases to examine: 1) temporal resolution of acoustic waves,

where a very small physical-time step is used, and 2) temporal resolution of particle propagation, which generally employs a much larger time step. For case 1, the small time step causes the preconditioner to be shut off, thus $\tilde{\rho}''_{p_t} = \tilde{\rho}'_{p_t} = \tilde{\rho}_{p_t}$ so that leaves the system reverts to its non-preconditioned form and thus is consistent between the LHS and RHS. For case 2, the preconditioning remains active. For Euler computations, the term $\Delta\tau/\delta t$ is no greater than unity, and is typically much less than one. Also, $\tilde{\rho}'_{p_t}/\tilde{\rho}_{p_t} \approx 1/M^2$ and from examination of Equation (4.46) we see that $\tilde{\rho}''_{p_t} \simeq \tilde{\rho}'_{p_t}$ so that $\mathbf{S} \approx \mathbf{\Gamma}$ again the LHS and RHS remaining nearly consistent.

In order to have a more compact form of the matrix to be inverted, we use an ADI factorization of the LHS:

$$\begin{aligned}\mathbf{S}\mathbf{M}_x \left[\mathbf{I} + \Delta\tau \frac{\partial}{\partial x}\left(\mathbf{\Lambda}_x - R_s \mathbf{I}\frac{\partial}{\partial x}\right)\right] \mathbf{M}_x^{-1}\mathbf{M}_y \left[\mathbf{I} + \Delta\tau \frac{\partial}{\partial y}\left(\mathbf{\Lambda}_y - R_s \mathbf{I}\frac{\partial}{\partial y}\right)\right] \mathbf{M}_y^{-1} \\ \mathbf{M}_z \left[\mathbf{I} + \Delta\tau \frac{\partial}{\partial z}\left(\mathbf{\Lambda}_z - R_s \mathbf{I}\frac{\partial}{\partial z}\right)\right] \mathbf{M}_z^{-1}\Delta Q_v = -\Delta\tau \left[\frac{3Q^{n+1,m} - 4Q^n + Q^{n-1}}{2\Delta t} + \mathcal{R}^{n+1,m}\right].\end{aligned} \tag{4.53}$$

4.1.3 Overall Algorithm

The final algorithm takes the following seven steps:

$$\Delta\hat{Q}_v^{(0)} = -\Delta\tau \mathbf{S}^{-1}\left[\frac{3Q^{n+1,m} - 4Q^n + Q^{n-1}}{2\Delta t} + \mathcal{R}^{n+1,m}\right], \tag{4.54}$$

$$\left[\mathbf{I} + \Delta\tau \frac{\partial}{\partial x}\left(\mathbf{\Lambda}_x - R_s \mathbf{I}\frac{\partial}{\partial x}\right)\right] \Delta\hat{Q}_v^{(1)} = \mathbf{M}_x^{-1}\Delta\hat{Q}_v^{(0)}, \tag{4.55}$$

$$\Delta\hat{Q}_v^{(2)} = \mathbf{M}_x \Delta\hat{Q}_v^{(1)}, \tag{4.56}$$

$$\left[\mathbf{I} + \Delta\tau \frac{\partial}{\partial y}\left(\mathbf{\Lambda}_y - R_s \mathbf{I}\frac{\partial}{\partial y}\right)\right] \Delta\hat{Q}_v^{(3)} = \mathbf{M}_y^{-1}\Delta\hat{Q}_v^{(2)}, \tag{4.57}$$

$$\Delta\hat{Q}_v^{(4)} = \mathbf{M}_y \Delta\hat{Q}_v^{(3)}, \tag{4.58}$$

$$\left[\mathbf{I} + \Delta\tau \frac{\partial}{\partial z}\left(\mathbf{\Lambda}_z - R_s \mathbf{I}\frac{\partial}{\partial z}\right)\right] \Delta\hat{Q}_v^{(5)} = \mathbf{M}_z^{-1}\Delta\hat{Q}_v^{(4)}, \tag{4.59}$$

$$\Delta Q_v = \mathbf{M}_z \Delta\hat{Q}_v^{(5)}, \tag{4.60}$$

where Equations (4.55), (4.57), and (4.59) are inverted using the BiCGStab approach (van der Vorst [71]). $\mathbf{S}^{-1}$ is illustrated in Appendix C.

4.1.4 Compact matrices coefficient storage

As mentioned in Subsection 4.1.3, for each Cartesian direction we end up with a linear problem of the form:

$$[\mathbf{A}]\mathbf{X} = \mathbf{B}, \tag{4.61}$$

The matrix $[\mathbf{A}]$ is a $(N_{cells} \cdot N_{eq})$ square matrix, because the equations are coupled, and it is sparse due to the non-structured connectivity of the grid. Furthermore, the BiCGStab iterative procedure used to solve this system is based on several matrix-vector operations. It is clear that matrices storage and manipulation strategy plays a crucial role when solving complex three-dimensional configurations, where a very large number of grid points is needed, without demanding prohibitive computer memory resources. To cope with this problem, a compact matrices coefficient storage procedure is used in this work. The key idea is that thanks to the diagonalizing procedure, only N_{eq} elements per cell are non-zero and we know their position in the matrix structure. To save memory during calculations, only the non-zero coefficients of the original matrix are stored in a vector $\mathbf{A}'$ with $(N_{cells} + N_{faces,i}) \cdot N_{eq}$ elements, where i is the sweep Cartesian direction. To rebuilt the matrix structure, we need now to create two auxiliary vectors of integers, storing the column and row informations of the matrix elements. For this purpose are introduced: the vector **IND** that stores the index of the column of the elements of $\mathbf{A}'$ in the matrix $\mathbf{A}$, and has the same size of $\mathbf{A}'$; the vector $\mathbf{N_{COL}}$ that stores the ending index in $\mathbf{A}'$ of each row of the matrix $\mathbf{A}$ and has $(N_{cells} \cdot N_{eq} + 1)$ elements.
As N_{cells} increases, $N_{faces,i}$ increase proportionally, with a factor depending on the levels of refinement required (for the structured case, the factor is 2, since we are considering a single direction). For large values on N_{cells} the ratio between the size of the three vectors to store in the compact case and the size of the single matrix of the standard case, assuming that the storing of an integer takes the same memory amount of the storing of a real (this assumption penalizes our estimate), will be:

$$\frac{2\left(N_{cells} + N_{faces,i}\right) N_{eq} + \left(N_{cells} \cdot N_{eq} + 1\right)}{\left(N_{cells} \cdot N_{eq}\right)^2} \simeq \frac{3N_{cells} + 2N_{faces,i}}{N_{cells}^2 \cdot N_{eq}} \propto \frac{1}{N_{cells}}. \tag{4.62}$$

An example of the procedure, for a very simple case is given in the following.

Considering the grid shown in Figure 4.2, supposing $N_{eq} = 1$ and solving the x–

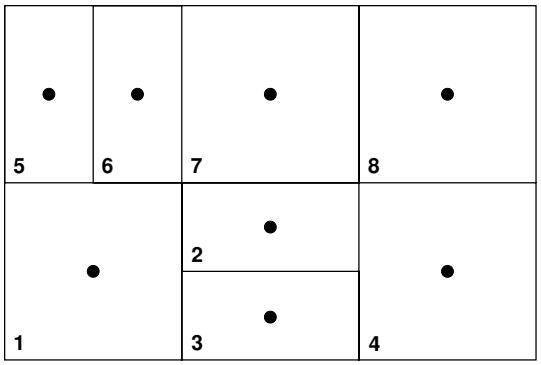


Figure 4.2: Example of grid points distribution.

direction, the matrix $[\mathbf{A}]$ of the resulting system will have the following form:

$$[\mathbf{A}] = \begin{bmatrix} \mathcal{A}_{11} & \mathcal{A}_{12} & \mathcal{A}_{13} & 0 & 0 & 0 & 0 & 0 \\ \mathcal{A}_{21} & \mathcal{A}_{22} & 0 & \mathcal{A}_{24} & 0 & 0 & 0 & 0 \\ \mathcal{A}_{31} & 0 & \mathcal{A}_{33} & \mathcal{A}_{34} & 0 & 0 & 0 & 0 \\ 0 & \mathcal{A}_{42} & \mathcal{A}_{43} & \mathcal{A}_{44} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathcal{A}_{55} & \mathcal{A}_{56} & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathcal{A}_{65} & \mathcal{A}_{66} & \mathcal{A}_{67} & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathcal{A}_{76} & \mathcal{A}_{77} & \mathcal{A}_{78} \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathcal{A}_{87} & \mathcal{A}_{88} \end{bmatrix}. \tag{4.63}$$

The stored vectors will be:

$$\mathbf{A}' = (\mathcal{A}_{11}, \mathcal{A}_{12}, \mathcal{A}_{13}, \mathcal{A}_{22}, \mathcal{A}_{21}, \mathcal{A}_{24}, \mathcal{A}_{33}, \mathcal{A}_{31}, \mathcal{A}_{34}, \mathcal{A}_{44}, \mathcal{A}_{42}, \mathcal{A}_{43}, \\ \mathcal{A}_{55}, \mathcal{A}_{56}, \mathcal{A}_{66}, \mathcal{A}_{65}, \mathcal{A}_{67}, \mathcal{A}_{77}, \mathcal{A}_{76}, \mathcal{A}_{78}, \mathcal{A}_{88}, \mathcal{A}_{87}), \tag{4.64}$$

$$\mathbf{IND} = (1, 2, 3, 2, 1, 4, 5, 1, 4, 4, 2, 3, 5, 6, 6, 5, 7, 7, 6, 8, 8, 7), \tag{4.65}$$

$$\mathbf{N}_{(\mathbf{COL})} = (0, 3, 6, 9, 12, 14, 17, 20, 22). \tag{4.66}$$

Note that for each row of the matrix $\mathbf{A}$, building the vector $\mathbf{A}'$, the first element to be stored is the diagonal element $\mathcal{A}_{ii}$.

In the general case of N_{eq}, each element $\mathcal{A}_{i,j}$ is a $N_{eq} \times N_{eq}$ diagonal matrix.

4.1.5 Boundary conditions

Because the code is a cell-centered finite-volume method, the boundary conditions at the external surfaces of the computational domain are applied using fake cells (f). Several types of boundary conditions have been applied. For inflow and outflow boundaries, the fake cell values are set so that certain quantities as stagnation or static pressure and

stagnation or static temperature are kept constant. For wall boundaries, the fake cell values are set such that there is no flux through the boundary. The fake cells are updated after the interior flow field is calculated and the boundary conditions are imposed explicitly. In the next subsections, the subscripts f, 1 and 2 indicate the fake-cell and the first and second cell inside the computational domain respectively.

Wall

The pressure gradient normal to the wall is assumed to be zero, which is approximately correct for boundary layers:

$$\left.\frac{\partial p}{\partial n}\right|_{wall} = 0. \tag{4.67}$$

A first-order application of this boundary requires that the fake cell value for pressure is equal to that of the first cell within the domain,

$$p_f = p_1. \tag{4.68}$$

Walls can either be isothermal or have no heat flux. The no heat flux case requires that the temperature gradient be zero at the wall. Similar to the pressure boundary condition above, this requires the temperature at the fake cell to be equal to the temperature of the first cell within the domain,

$$T_f = T_1. \tag{4.69}$$

The isothermal case states that the wall is held at a constant temperature T_{wall}. To ensure that the wall remains at the appropriate temperature, in the fake cells the following condition is employed:

$$T_f = 2T_{wall} - T_1. \tag{4.70}$$

Walls can also either be inviscid or viscous. Viscous walls satisfy the no slip condition such that the normal component of the relative velocity to the wall V_n and the tangential component of the relative velocity to the wall V_t are zero. This condition is easily applied by

$$u_f = 2u_{wall} - u_1, \qquad v_f = 2v_{wall} - v_1, \qquad w_f = 2w_{wall} - w_1. \tag{4.71}$$

Inviscid walls must ensure that no flow goes through them, therefore only the normal component of the velocity V_n is set to zero. Like the pressure and no heat-flux conditions, for inviscid walls the gradient of the tangent velocity normal to the wall is zero.

As for temperature (4.70) and velocity (4.71), the wall boundary condition for k and ω are

$$k_f = -k_1, \qquad \omega_f = 2\omega_{wall} - \omega_1, \tag{4.72}$$

being $k_{wall} = 0$. For the specific dissipation rate the boundary condition suggested by Menter [72] is used:

$$\omega_{wall} = a\frac{6\nu}{\beta y_1^2}, \tag{4.73}$$

where y_1 is the distance between the wall and the first cell above the wall, and the constant a is equal to 10.

Subsonic inflow

A free-stream flow condition is imposed and the magnitude of the velocity at the fake cell centers is reconstructed using a second-order discretization:

$$\begin{aligned} |V_1| &= \sqrt{u_1^2 + v_1^2 + w_1^2}, \\ |V_2| &= \sqrt{u_2^2 + v_2^2 + w_2^2}, \\ \frac{\partial |V|}{\partial n} = 0 \quad &\longrightarrow \quad |V_f| = \frac{4}{3}|V_1| - \frac{1}{3}|V_2|. \end{aligned} \tag{4.74}$$

Then, using the imposed values of the stagnation temperature T_{stag} and the stagnation pressure p_{stag}, the static temperature and the static pressure can be evaluated:

$$\begin{aligned} T_f &= T_{stag} - \frac{|V_f|^2}{2h_t}, \\ p_f &= p_{stag}\left(\frac{T_f}{T_{stag}}\right)^{\frac{h_t}{R}}. \end{aligned} \tag{4.75}$$

The direction of the inlet velocity is also imposed, and then the three components of the velocity can be calculated:

$$u_f = \sin\phi\cos\theta\,|V_f|, \qquad v_f = \sin\phi\sin\theta\,|V_f|, \qquad w_f = \cos\phi\,|V_f|, \tag{4.76}$$

where θ and ϕ are the azimuth and zenith angle, respectively. The turbulent variables are calculated imposing the inlet turbulence intensity t_i and the inlet kinematic turbulent viscosity $\nu_t = \mu_t/\rho$, in the following way:

$$k_f = \frac{3}{2}t_i^2\,|V_f|^2, \qquad \omega_f = \frac{\alpha^*}{\nu_t}k_f. \tag{4.77}$$

Supersonic inflow

In the case of supersonic inflow, the Mach number M, the static temperature T_∞ and the static pressure p_∞ are imposed. The speed of sound is evaluated as

$$c = \sqrt{\frac{\rho h_T}{\rho_T \left(1 - \rho h_p\right) + \rho \rho_p h_T}} \tag{4.78}$$

and then in the fake cells:

$$T_f = T_\infty, \qquad p_f = p_\infty, \qquad |V_f| = Mc. \tag{4.79}$$

As for the subsonic inflow case, the direction of the inlet velocity is also imposed and then the three components of the velocity can be calculated:

$$u_f = \sin\phi \cos\theta \left|V_f\right|, \qquad v_f = \sin\phi \sin\theta \left|V_f\right|, \qquad w_f = \cos\phi \left|V_f\right|. \tag{4.80}$$

The turbulent variables are calculated imposing the inlet turbulence intensity t_i and the inlet kinematic turbulent viscosity $\nu_t = \mu_t/\rho$, in the following way:

$$k_f = \frac{3}{2} t_i^2 \left|V_f\right|^2, \qquad \omega_f = \frac{\alpha^*}{\nu_t} k_f. \tag{4.81}$$

Subsonic outflow

In the case of subsonic outflow, only the static pressure p_{out} is imposed while all of the other variables are extrapolated with a second-order reconstruction. Then:

$$\begin{aligned} & & p_f &= p_{out}, & & \\ u_f &= \frac{4}{3}u_1 - \frac{1}{3}u_2, & v_f &= \frac{4}{3}v_1 - \frac{1}{3}v_2, & w_f &= \frac{4}{3}w_1 - \frac{1}{3}w_2, \\ T_f &= \frac{4}{3}T_1 - \frac{1}{3}T_2, & k_f &= \frac{4}{3}k_1 - \frac{1}{3}k_2, & \omega_f &= \frac{4}{3}\omega_1 - \frac{1}{3}\omega_2. \end{aligned} \tag{4.82}$$

Supersonic outflow

In the case of supersonic outflow, no variables can be imposed and then all the variables are extrapolated using always the same second-order reconstruction. Then:

$$
\begin{aligned}
& & p_f &= \frac{4}{3}p_1 - \frac{1}{3}p_2, & & \\
u_f &= \frac{4}{3}u_1 - \frac{1}{3}u_2, & v_f &= \frac{4}{3}v_1 - \frac{1}{3}v_2, & w_f &= \frac{4}{3}w_1 - \frac{1}{3}w_2, \\
T_f &= \frac{4}{3}T_1 - \frac{1}{3}T_2, & k_f &= \frac{4}{3}k_1 - \frac{1}{3}k_2, & \omega_f &= \frac{4}{3}\omega_1 - \frac{1}{3}\omega_2.
\end{aligned}
\tag{4.83}
$$

Periodicity

The implementation of periodic boundary conditions is employed using ghost-cells. In this case, the values of the first cells inside the flow of one surface are copied in the ghost cells of the second surface and vice-versa. In this way, periodicity of the functions and their derivative is imposed. The restriction is that the cells of the two surface must have a one-to-one correspondence.

Immersed boundary

For boundaries inside the computational domain, the wall boundary conditions are imposed using the IB technique shown in Chapter 3.

4.2 Surface-based structural solver for Lagrangian mesh

The structure is defined by a triangulated network of N_e springs (edges), forming N_t triangles. The mass of the structure is concentrated on the N_v vertices of the triangles with Cartesian coordinates $\{\boldsymbol{x}_i\}$, $i \epsilon 1 \ldots N_v$, uniformly distributed on the surface, see Fig. 4.3.

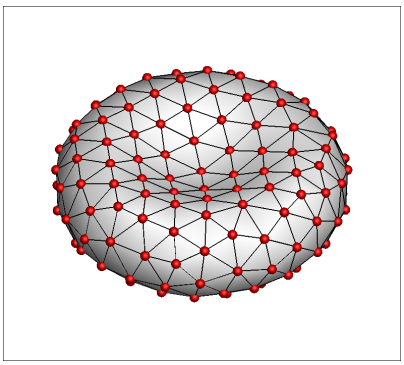


Figure 4.3: Triangulated surface mesh.

The potential energy of the structure includes in-plane elastic terms, combined with bending energy and constraints for volume conservation:

$$W\left(\{\mathbf{x}_i\}\right) = W_{in-plane}\left(\{\mathbf{x}_i\}\right) + W_{bending}\left(\{\mathbf{x}_i\}\right) + W_{volume}\left(\{\mathbf{x}_i\}\right). \tag{4.84}$$

and the corresponding nodal internal forces:

$$\begin{aligned}
\mathbf{f}_i &= \mathbf{f}_e + \mathbf{f}_b + \mathbf{f}_v, \\
\mathbf{f}_e &= -\frac{\partial W_{in-plane}\left(\{\mathbf{x}_i\}\right)}{\partial \mathbf{x}_i}, \\
\mathbf{f}_b &= -\frac{\partial W_{bending}\left(\{\mathbf{x}_i\}\right)}{\partial \mathbf{x}_i}, \\
\mathbf{f}_v &= -\frac{\partial W_{volume}\left(\{\mathbf{x}_i\}\right)}{\partial \mathbf{x}_i}.
\end{aligned} \tag{4.85}$$

The in-plane deformations are evaluated using a finite element method for large deformations of a triangulated surface mesh based on the work of Taylor *et al.* [33] and also implemented by Hammer *et al.* [34]. The model ignores the shear bending forces in the element, which will be evaluated in a separated way. Here, the method is briefly described and the reader is referred to [33] for the details and the derivations. The right Cauchy-Green deformation tensor can be computed as

$$\mathbf{C} = \mathbf{G}^T \ \mathbf{g} \ \mathbf{G} \tag{4.86}$$

where $\mathbf{G}$ and $\mathbf{g}$ are the inverse Jacobian matrices mapping the position of a point in global coordinates in the initial and current position, respectively, to the parametric representation adopted within a triangle:

$$\mathbf{G} = \begin{bmatrix} \frac{1}{|\Delta\mathbf{R}_{21}|} & -\frac{\Delta\mathbf{R}_{21}^T \cdot \Delta\mathbf{R}_{31}}{|\Delta\mathbf{R}_{21}||\mathbf{n}|} \\ 0 & \frac{|\Delta\mathbf{R}_{21}|}{|\mathbf{n}|} \end{bmatrix} \qquad \mathbf{g} = \begin{bmatrix} \Delta\mathbf{r}_{21}^T \cdot \Delta\mathbf{r}_{21} & \Delta\mathbf{r}_{21}^T \cdot \Delta\mathbf{r}_{31} \\ \Delta\mathbf{r}_{21}^T \cdot \Delta\mathbf{r}_{23} & \Delta\mathbf{r}_{31}^T \cdot \Delta\mathbf{r}_{31} \end{bmatrix} \tag{4.87}$$

where $\Delta\mathbf{r}_{ij}$ and $\Delta\mathbf{R}_{ij}$ are the position vectors from vertex j to i in the present (deformed) and original (undeformed) configuration, respectively, see Fig. 4.4. The vector $\mathbf{n}$ is defined as $\Delta\mathbf{R}_{21} \times \Delta\mathbf{R}_{31}$. The Green strain tensor can be obtained by

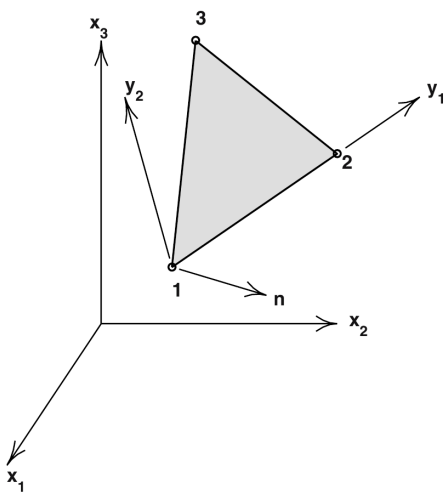


Figure 4.4: Description of coordinates for triangular element.

$$\mathbf{E} = \frac{1}{2}\left(\mathbf{C} - \mathbf{I}\right) \tag{4.88}$$

where $\mathbf{I}$ is the identity tensor. As shown in Section 2.3, the stress tensors are related to the strain tensors by the constitutive law of the material. In the case of elastic behavior, stresses are thus given by

$$\boldsymbol{\sigma} = \mathbf{DE} \qquad \text{elastic behavior} \tag{4.89}$$

where $\mathbf{D}$ are constant elastic moduli. In the case of hyperelastic behavior, stresses are given by

$$\boldsymbol{\sigma} = 2\mu\mathbf{E} + \lambda tr(\mathbf{E})\mathbf{I} \qquad \text{hyperelastic behavior} \tag{4.90}$$

where μ and λ are the Lamè coefficients and $tr(\cdot)$ is the trace function. Finally, the local component of the stress tensor $\boldsymbol{\sigma}$ are used to calculate the internal forces on the vertices

of a triangular element:

$$\mathbf{f}_e = Ah\mathbf{B}^T \begin{pmatrix} \boldsymbol{\sigma}_{11} \\ \boldsymbol{\sigma}_{22} \\ \boldsymbol{\sigma}_{12} \end{pmatrix} \tag{4.91}$$

where A and h are the area and height of the triangular element in the original configuration, respectively, and $\mathbf{B} = \mathbf{Qb}$, where $\mathbf{Q}$ is the stress tensor transformation matrix, which can expressed in terms of components $\mathbf{G}$ as:

$$\mathbf{Q} = \begin{bmatrix} G_{11}^2 & 0 & 0 \\ G_{12}^2 & G_{22}^2 & G_{12}G_{22} \\ 2G_{12}G_{11} & 0 & G_{11}G_{22} \end{bmatrix} \tag{4.92}$$

and $\mathbf{b}$ is a 3×9 strain displacement matrix given by:

$$\mathbf{Q} = \begin{bmatrix} -\Delta\mathbf{r}_{21}^T & \Delta\mathbf{r}_{21}^T & (0,0,0) \\ -\Delta\mathbf{r}_{31}^T & (0,0,0) & \Delta\mathbf{r}_{31}^T \\ -(\Delta\mathbf{r}_{21} + \Delta\mathbf{r}_{31})^T & \Delta\mathbf{r}_{31}^T & \Delta\mathbf{r}_{21}^T \end{bmatrix}. \tag{4.93}$$

The nine elements of vector $\mathbf{f}_e$ are the three components of vertex 1, followed by those of vertices 2 and 3 of the triangle. The internal force f_{e_i} for a given vertex i is obtained by summing the contributions from all elements sharing that vertex.

The out-of-plane deformation of two adjacent faces sharing an edge is modeled by means of a bending spring, as shown in Fig. 4.5. Four vertices are involved in the energy term. Here, the model adopted in [35] is employed:

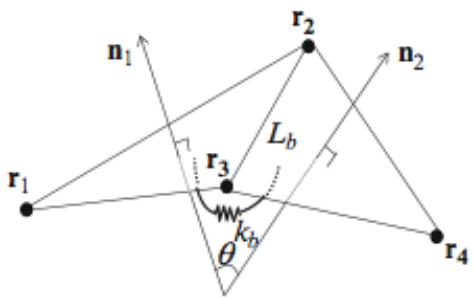


Figure 4.5: The out-of-plane deformation of two adjacent faces.

$$W_{bending}\left(\{\mathbf{x}_i\}\right) = \sum_{j=1}^{N_{e_i}} k_b \left[1 - \cos(\theta_j - \theta_j^0)\right] \tag{4.94}$$

where N_{e_i} is the number of edges starting from the i–vertex, θ and θ^0 are the angles between two faces in the actual and *stress-free* configurations, respectively, and k_b is the bending constant. The nodal forces corresponding to the bending energy are obtained as (see Fig. 4.5):

$$\begin{aligned}
\boldsymbol{f}_{b_1} =& \beta_b[b_{11}(\boldsymbol{n}_1 \times \boldsymbol{r}_{32}) + b_{12}(\boldsymbol{n}_2 \times \boldsymbol{r}_{32})] \\
\boldsymbol{f}_{b_2} =& \beta_b[b_{11}(\boldsymbol{n}_1 \times \boldsymbol{r}_{13}) + b_{12}(\boldsymbol{n}_1 \times \boldsymbol{r}_{34} + \boldsymbol{n}_2 \times \boldsymbol{r}_{13}) + b_{22}(\boldsymbol{n}_2 \times \boldsymbol{r}_{34})] \\
\boldsymbol{f}_{b_3} =& \beta_b[b_{11}(\boldsymbol{n}_1 \times \boldsymbol{r}_{21}) + b_{12}(\boldsymbol{n}_1 \times \boldsymbol{r}_{42} + \boldsymbol{n}_2 \times \boldsymbol{r}_{21}) + b_{22}(\boldsymbol{n}_2 \times \boldsymbol{r}_{42})] \\
\boldsymbol{f}_{b_4} =& \beta_b[b_{11}(\boldsymbol{n}_1 \times \boldsymbol{r}_{23}) + b_{22}(\boldsymbol{n}_2 \times \boldsymbol{r}_{23})]
\end{aligned} \tag{4.95}$$

with

$$b_{11} = -\frac{\cos(\theta)}{|\boldsymbol{n}_1|^2}; \quad b_{12} = \frac{1}{|\boldsymbol{n}_1||\boldsymbol{n}_2|}; b_{22} = -\frac{\cos(\theta)}{|\boldsymbol{n}_2|^2}; \quad \beta_b = k_b \frac{\sin(\theta)\cos(\theta_0) - \cos(\theta)\sin(\theta_0)}{\sqrt{1-\cos^2(\theta)}}; \tag{4.96}$$

and $\boldsymbol{r}_{ij} = \boldsymbol{r}_i - \boldsymbol{r}_j$, with $\boldsymbol{r}_i$ position vector of the vertex i.

The volume conservation constraint is given by

$$W_{volume}(\{\mathbf{x}_i\}) = \frac{k_v \left(V - V_0^{tot}\right)^2}{2V_0^{tot}} \tag{4.97}$$

where k_v is the volume constraint constant and V_0^{tot} is the desired total volume. The nodal forces corresponding to the volume conservation constraint are obtained as:

$$\boldsymbol{f}_{v_i} = -\frac{k_v(V - V_0^{tot})}{V_0^{tot}} \sum_{j=1}^{N_{t_i}} \frac{1}{6}\left(\frac{|\boldsymbol{n}_j|}{3} + \boldsymbol{t}_c^j \times \boldsymbol{r}_{lm}^j\right) \tag{4.98}$$

where N_{t_i} is the number of triangles containing the i–vertex and $\boldsymbol{t}_c^j = (\boldsymbol{x}_i + \boldsymbol{x}_l + \boldsymbol{x}_m)/3$ is the center-of-mass of the j-th triangle.

Considering also body forces G and external surface forces T (i.e., the gravity forces and the hydrodynamic load) the total potential energy is:

$$\Pi(\{\mathbf{x}_i\}) = W(\{\mathbf{x}_i\}) + \Phi_G(\{\mathbf{x}_i\}) + \Phi_T(\{\mathbf{x}_i\}). \tag{4.99}$$

Then, the forces applied to mass points are:

$$\mathbf{F}_i = -\frac{\partial \Pi(\{\mathbf{x}_i\})}{\partial \mathbf{x}_i} = \mathbf{f}_i + \mathbf{G}_i + \mathbf{T}_i, \tag{4.100}$$

The minimum energy problem can be solved by moving mass point in accordance with the motion equation:

$$m_i \frac{d^2\mathbf{r}_i}{dt^2} + \eta_i \frac{d\mathbf{r}_i}{dt} = \mathbf{F}_i, \tag{4.101}$$

which yields the displacement $\mathbf{r}_i$ of the i-th element of mass m_i, inner viscosity η_i subject to the force $\mathbf{F}_i$.

4.3 Conjugate-Heat-Transfer coupling

In CHT problems, the URANS equations are solved at all fluid cells and the heat conduction equation is solved at all solid cells, using the same spatial discretization and time-marching numerical algorithm presented in the Section 4.1; the two solutions are coupled by the interface conditions requiring that both the temperature and heat-flux be the same at all (fluid-solid) boundary points. Therefore, in order to enforce such interface conditions, as already done for the fluid domain, for each solid-interface cell, the appropriate SCPP, and eventually L, are determined and the temperature is interpolated just as for each fluid-interface cell, see Chapter 3.

4.3.1 Interface boundary conditions

With the scope to ensure a solution to the heat transfer equation, two conditions shall be imposed at each time step. At first, it is reasonable to say from theory that the temperature distribution on a finite plate is completely defined by its boundary temperature. The consequence of that is a boundary condition involving the wall temperature. Applying this statement to a general solid body surface and to the particular case study, it is straightforward to define that a continuity of temperature shall occur on the solid-fluid interface. As the author has experienced, this first condition it is not sufficient to define a physical solution. Heat transfer between two different materials is strongly influenced by the difference in thermal diffusivity, affecting the exchange rate. The second condition may be introduced, which forces the heat transfer rate by means of thermal fluxes at solid-fluid interface. Moreover, solid heat transfer travels on very different time scales than Navier–Stokes equations (2.1) - (2.3). In fact, because of the significant difference between the two media, the transient phenomena in the fluid usually take place at a much smaller time scale as those in the solid. Thus, a direct coupled solution of the URANS and heat conduction equations is unfeasible and an iterative procedure is to be employed.

The aforementioned interface conditions are given as:

$$T_{w,f} = T_{w,s} \tag{4.102}$$

$$K_s \nabla T_s \cdot \mathbf{n}_w = K_f \nabla T_f \cdot \mathbf{n}_w \tag{4.103}$$

where the subscripts s and f refer to solid and fluid respectively, K indicates the thermal conductivity, and $\mathbf{n}_w$ is the unit vector normal to the wall. The boundary conditions above

are to be enforced at all fluid- and solid-interface cells. Unfortunately the corresponding wall-points do not coincide, see Fig. 3.1, so that a connectivity map for all FCPPs and SCPPs on the surface is to be created and an iterative procedure to calculate the wall temperature for both FCPPs and SCPPs at each physical time-step is required.

A first-order-accurate approximation of $\nabla T_f \cdot \mathbf{n}_{w,f}$ is obtained as:

$$\nabla T_f \cdot \mathbf{n}_{w,f} = (T^m_{int,f} - T^{m-1}_{w,f})\beta_f \tag{4.104}$$

where m is the current iteration and β_f is the inverse of the distance between the fluid interface cell and the wall. Eq. (4.104) provides the heat fluxes (apart from the thermal conductivity) at all FCPPs. Such values are then used to provide those pertaining to all SCPPs by means of a distance-weighted interpolation, see fig 4.6. Then, the Neumann

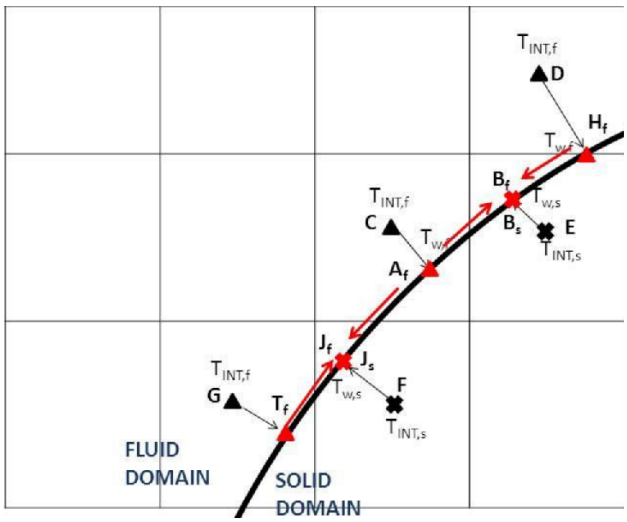


Figure 4.6: Gradient interpolation on the surface

condition at all SCPPs is written as:

$$\nabla T_s \cdot \mathbf{n}_{w,s} = \frac{K_f}{K_s}(\nabla T_f \cdot \mathbf{n}_{w,f}) = \frac{K_f}{K_s}\sum_i^{ns}\frac{\alpha_i}{q}(\nabla T_f \cdot \mathbf{n}_{w,f})|_i, \tag{4.105}$$

where ns is the number of the neighboring FCPPs, α_i is the inverse of the distance between the considered SCPP and the surrounding FCPP$_i$ and $q = \sum_i^{ns}\alpha_i$. $T^m_{w,s}$ can then be computed using again a first-order-accurate scheme:

$$T^m_{w,s} = T^m_{int,s} - \frac{\nabla T_s \cdot \mathbf{n}_{w,s}}{\beta_s}, \tag{4.106}$$

and used to provide the wall fluid temperature at all SCPPs by means of the Dirichlet condition. Such values are finally interpolated onto each FCPP to provide the updated value for $T^m_{w,f}$:

$$T^m_{w,f} = \sum_i^{nf} \frac{\alpha_i}{q} \left(T^m_{w,f}\right)|_i = \sum_i^{nf} \frac{\alpha_i}{q} \left(T^m_{w,s}\right)|_i, \tag{4.107}$$

where nf is the number of the neighboring SCPPs, α_i is the inverse distance between the considered FCPP and the surrounding SCPP$_i$ and $q = \sum_i^{nf} \alpha_i$.

4.3.2 Stability considerations

The way to apply these boundary conditions to the fluid and solid zones is critical for the accuracy and stability of the computations. Several studies have been carried out about the way to force boundary conditions in a CFD solver. Giles [73] makes a very accurate analysis on the stability of the solid-fluid heat transfer coupling in 1D and in that analysis the coupling technique which is used by this work is classified as "loosely coupled". This analysis is performed on a backward implicit algorithm and explicit updating of boundary conditions. The scope is to make a correct choice as which domain shall use boundary condition on temperature and which one shall use boundary condition on thermal flux. The solid-fluid interface is identified as $x_{surface} = 0$, with C (heat capacity in J/K) and K (thermal conductivity) having uniform values C_- and K_- for $x < 0$, on one side, and C_+ and K_+ for $x > 0$, on the other side. The initial assumption is to impose the temperature B.C. as:

$$T_+ = T_w \quad \text{from} \quad T_w = T_- \tag{4.108}$$

The stability analysis shows a stability limit of

$$\frac{\Delta x_+ C_+}{\Delta x_- C_-} < 1 \tag{4.109}$$

which is satisfied only if the temperature B.C. is applied to the fluid and more precisely if "+" is the fluid side and "-" is the solid side. As it is explained later in this paragraph, this condition implies fluid temperature to lag solid temperature by one iteration. The final conclusion is that the thermal flux B.C. (4.103) should be forced to the solid and the temperature B. C. of (4.102) should be forced to the fluid. To a similar conclusion have been led Duchaine et al. in [74], who simulate CHT coupling of separate solid and fluid solvers. The main reason for unstable behaviour is the large difference between the thermal conductivity which may occur between solid and fluid, which makes the B.C.

choice critical. Moreover, the difference between the solid- and the fluid-heat-transfer time scales can lead to instability problems. In the present work, the first problem is overcome by using a proper coupling technique and some numerical test cases will show the applicability of the method to a wide range of thermal conductivities. No additional relaxation factors are needed [74]. On the other hand, the second remark is solved by using a higher time scale solely for equation (2.37). Moreover, this approach is a classical stable approach for problems having Biot number less than one. For all applications, this work falls inside this class.

4.3.3 Overall procedure

The overall procedure to be employed at each pseudo-time step m is shown in Fig. 4.7. More in detail: i) the IB grid generator detects the position of each cell of the

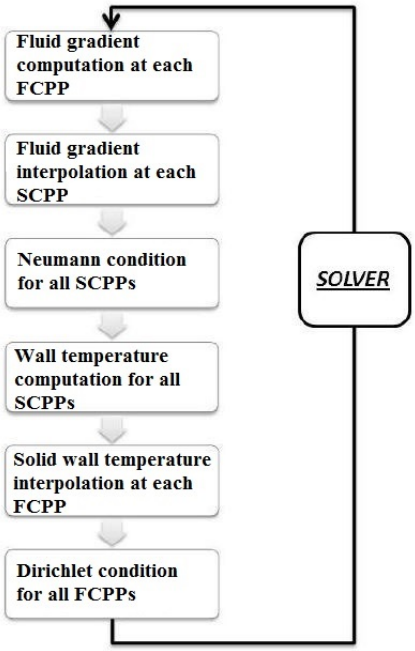


Figure 4.7: Overall CHT procedure to be employed at each pseudo-time step m.

Cartesian grid with respect to the geometry, discretized by a surface mesh consisting of triangular elements, and divides the cells into four types: solid and fluid cells—whose centers lie within the body and within the fluid, respectively; fluid- and solid-interface cells, that have at least one of their neighbors inside and outside the body, respectively; ii) the URANS equations are solved at all internal fluid cells, whereas the heat conduction

equation is solved at all internal solid cells using the same spatial discretization and time-marching scheme; iii) the boundary conditions, which account for the presence of the body are imposed at the fluid-/solid-interface cells, using a local interpolation procedure; iv) the interface boundary conditions requiring that both the temperature and heat-flux are the same for the fluid and the solid at all boundary points are imposed by the CHT approach.

4.4 Fluid Structure Interaction coupling

Aeroelasticity is one of the most important and challenging problems in the turbomachinery design and operation; and its understanding is critical to improve the performance, efficiency and reliability of a given advanced-design fluid machinery. A serious aeroelasticity analysis demands an accurate simulation of the flow field and of the blades dynamic response by means of a state-of-the-art fluid-structure-interaction (FSI) solver. Here, the finite volume URANS solver is combined with the efficient surface-based structural one describing the dynamics of deformable bodies, to provide a dual-time-stepping computational tool for predicting the flow field around deformable bodies as well as their induced motion.

For the case of not fixed boundaries, the position of the Lagrangian surface points changes in time with respect to the fixed Cartesian grid, and therefore the tagging procedure is repeated at each physical time-step, in order to recognize the new fluid/solid-interface cells and the new position of the relative FCPPs and SCPPs. It is noteworthy that the Cartesian grid and the Lagrangian mesh are not re-generated at each physical time-step and thus the ray-tracing technique cannot be used to refine the grid in the new high gradient regions. For this reason the grid is generated refining all the regions crossed by the Lagrangian mesh points in the considered FSI problem.

Two different types of structures can be solved using this approach:

- rigid structure with imposed motion: the structure is rigid and the position of each mesh point is imposed at each time step;
- deformable structure: the forces exerted by the fluid upon the structure are fed to the structural solver which updates the position and the velocity of the surface mesh points at each physical time-step.

An important issue that requires careful consideration arises when the motion of the immersed boundary exposes into the fluid a cell, which at the previous time-step was internal. The numerical problem is that such cells lack of physically realistic values of the variable at the time steps n and $n-1$ required in Equation (4.27). For the present work, in the Dual Time Stepping (DTS) framework, this represents a restriction for the physical-time step used. In fact, the physical-time step has to guarantee that the body never transverses an entire computational cell within one time step, meaning that a previ-

ous solid cell first emerges into the fluid as an interface cell (for which the variable values are iteratively interpolated in the pseudo-time step and not computed). In the first computations, we have noted that this conditions is less restrictive than the CFL condition for stability of the DTS algorithm, and, thus, any additional stability restriction on the overall algorithm is not required.

4.4.1 Surface forces calculation

The computation of the surface pressure and shear stress is a key issue for non–boundary conforming formulations. In the present study, a linear interpolation strategy has been implemented. The method starts with the geometrical description of the three-dimensional object, as a triangulated closed surface. Then, using a local search process, starting from each vertex, a probe point L on the outgoing normal is selected so that all surrounding computational cells lies inside the fluid, see Fig. 4.8.

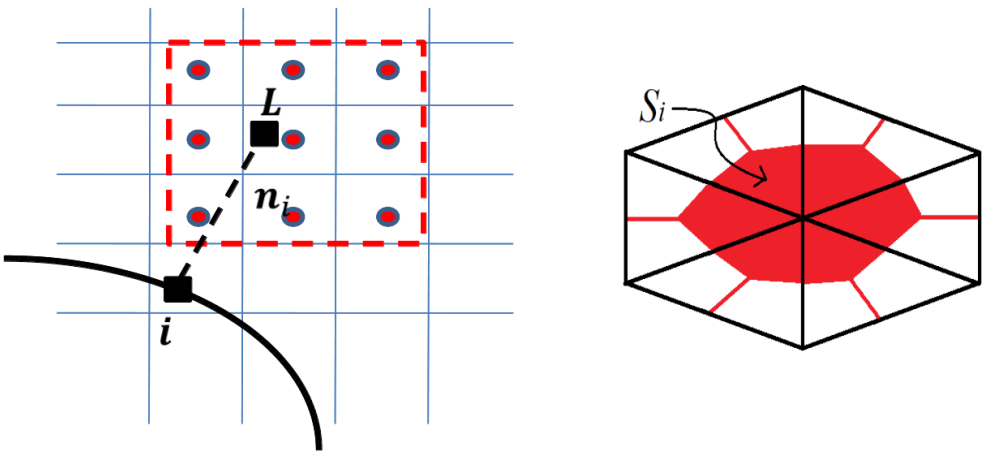


Figure 4.8: Support-domain of the probe used for the evaluation of the forces at each triangulated mesh vertex.

This step can be iterative, adjusting the distance from the boundary until the above condition is met. The value of the pressure and velocity derivatives at the probe point L are calculated by the least-squares interpolation formula (see Section 3.3) involving the surrounding cells. Velocity derivatives on the body are assumed to be equal to those at the probe location (linear velocity profile) and then the external surface force T_i can be calculated as

$$\mathbf{T}_i = (\boldsymbol{\tau}_L \cdot \mathbf{n}_i - p_L \mathbf{n}_i)\, S_i, \tag{4.110}$$

where S_i is the area related to the i–vertex, see Fig. 4.8.

4.4.2 Overall procedure

The overall procedure to be employed at each physical-time step n is shown in Fig. 4.9. In more detail, at each physical time-step: i) the IB grid generator detects the position

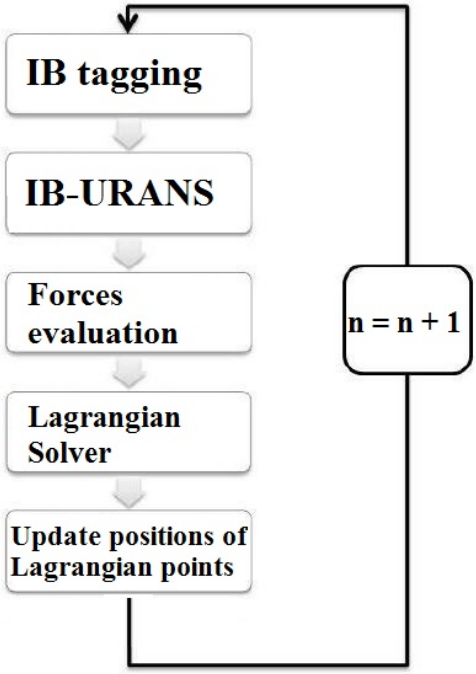


Figure 4.9: Overall FSI procedure to be employed at each physical-time step n.

of each cell of the Cartesian grid with respect to the geometry, discretized by a surface mesh consisting of triangular elements, and divides the cells into three types: solid and fluid cells–whose centers lie within the body and within the fluid, respectively; and fluid interface cells, which have at least one of their neighbors inside the body; ii) the flow variables at the centers of the fluid cells are computed by the URANS solver and the boundary conditions, which account for the presence of the structure, are imposed at the interface fluid cells, using a local interpolation procedure, so that the solid cells have no influence on the flow field; iii) the forces exerted by the fluid upon the structure are fed to the structural solver which updates the position and the velocity of the surface mesh points; iv) the coupling is converged by iterating within a standard dual time stepping procedure, until the unsteady residuals are reduced to a prescribed level.

Chapter 5

Validation

In order to test the proposed methodology several test-cases have been considered, involving two- and three-dimensional steady and unsteady flows in an extended range of Mach and Reynolds numbers past both rigid and deformable geometries.

Firstly, in Section 5.1, the accuracy of the new LS reconstruction (Section 3.3) is compared with the 1D reconstruction (Section 3.1) already validated by de Tullio [75] using the following test-cases: the laminar incompressible flow past a sphere, the laminar supersonic flow past an NACA-0012 airfoil and the compressible supersonic flow past a cylinder. Then, adaptive wall functions (Section 3.4) have been employed for: both subsonic and supersonic flows through the VKI-LS59 turbine-rotor cascade, the transonic flow past the Unmanned Space Vehicle, and the transonic flow past the Agard Wing.

Then, in Section 5.2, the CHT interface conditions coupled with the 1D, IDW and LS reconstructions have been tested considering the following test case: the heat transfer in a rotating flow inside a tube, the flow past a heated cylinder in cross-flow and the internally cooled C3X vane.

Finally, in Section 5.3, a first computation with a moving geometry has been performed using the least squares reconstruction scheme with very little additional effort compared to the non-moving boundary case, namely, the oscillating circular cylinder in a cross-flow. A three-dimensional FSI computation of the incompressible low-Reynolds number flow past a deformable sphere is presented, showing very good agreement with analytical data even in the three-dimensional case.

5.1 IB-URANS validation

5.1.1 Incompressible flow past a sphere

The flow past a sphere has been computed to test the flow solver and the IB method versus a first three dimensional application. A single value of the free-stream Mach number, $M_\infty = 0.03$, and four values of the Reynolds number (based on the sphere diameter, D, the free-stream velocity, U_∞, and kinematic viscosity, ν_∞), namely, 40, 60, 80, and 100, have been considered. The computational domain is a box; the inlet and outlet boundary planes are located at $x_i = -40D$ and $x_o = 80D$ and the far-field boundaries are located at $y = \pm 40D$ and $z = \pm 40D$, the origin of the box coinciding with the center of the sphere. Computations have been performed using a grid, with a total number of cells equal to 1219822. The grid has been locally refined on the sphere surface in order to have a good resolution of the boundary layer, and in a box surrounding the sphere and the wake, in order to describe accurately the separation region, see Fig. 5.1.

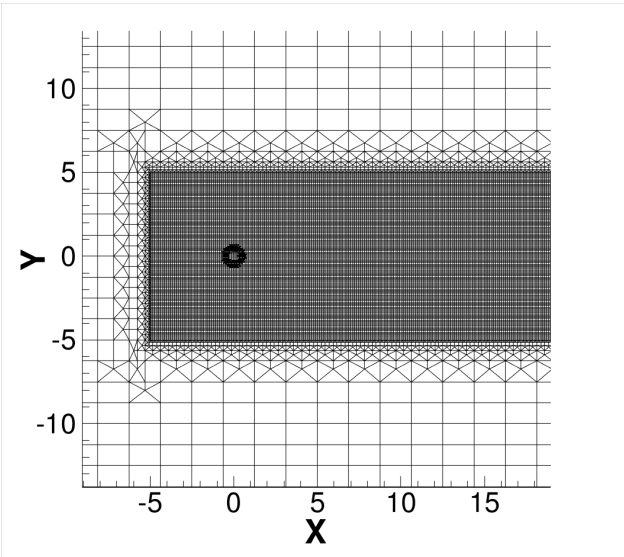


Figure 5.1: Incompressible flow past a sphere: local view of the refined grid for $z = 0$.

According to Batchelor, the flow around a sphere does not separate up to $Re \simeq 24$ and for increasing Reynolds number the axial length of the separation bubble grows linearly up to $Re \simeq 100$. The same results are obtained by the present numerical simulations. Figures 5.2 and 5.3 show the length of the separation bubble and the drag coefficient compared to the experimental data by Batchelor [2] and Clift *et al.* [4] and the numerical results obtained by de Tullio *et al.* [3] using the inverse distance weighted reconstruction.

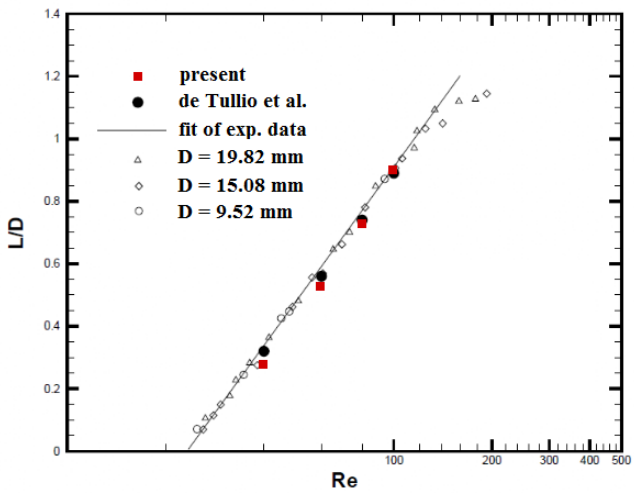


Figure 5.2: Incompressible flow past a sphere: length of the separation bubble compared with the experimental data [2] and the numerical ones obtained by de Tullio *et al.* [3].

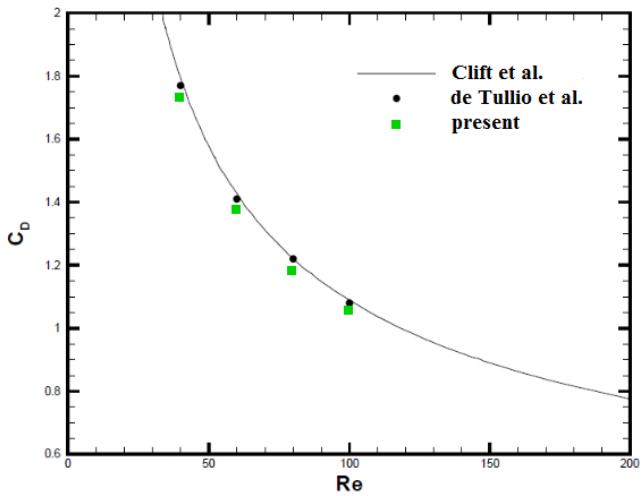


Figure 5.3: Incompressible flow past a sphere: drag coefficient are compared with the experimental data obtained by Clift *et al.* [4] and the numerical ones obtained by de Tullio *et al.* [3].

5.1.2 Supersonic flow past an NACA0012 airfoil

In order to test the proposed methodology versus a well documented viscous flow at high Mach number, the laminar supersonic flow past a NACA0012 airfoil with $M_\infty = 2$, $\alpha = 10°$ and $Re_\infty = 1000$ has been considered (Bristeau *et al.* [76]). De Palma *et al.* [14] used three structured grids with 125^2, 250^2, and 500^2 cells to discretize the computational domain $[-8\,c; 9\,c] \times [-8\,c; 8\,c]$, c being the chord-length of the airfoil, whose leading edge is located at the origin. Standard characteristic boundary conditions have been imposed at inlet and outlet surfaces. Numerical results are obtained using the TVD second-order-accurate upwind scheme and the finest grid used also by De Palma *et al.* [14].

The distribution along the profile of the pressure coefficient $c_p = (p - p_\infty)/(0.5\rho_\infty U_\infty^2)$ is given in Figure 5.4 where the solution obtained with the least squares reconstruction, is in a very good agreement with those obtained by De Palma *et al.* [14] using the one dimensional reconstruction.

Finally, the Mach number contours computed are provided in Figure 5.5 showing that the shock is well captured.

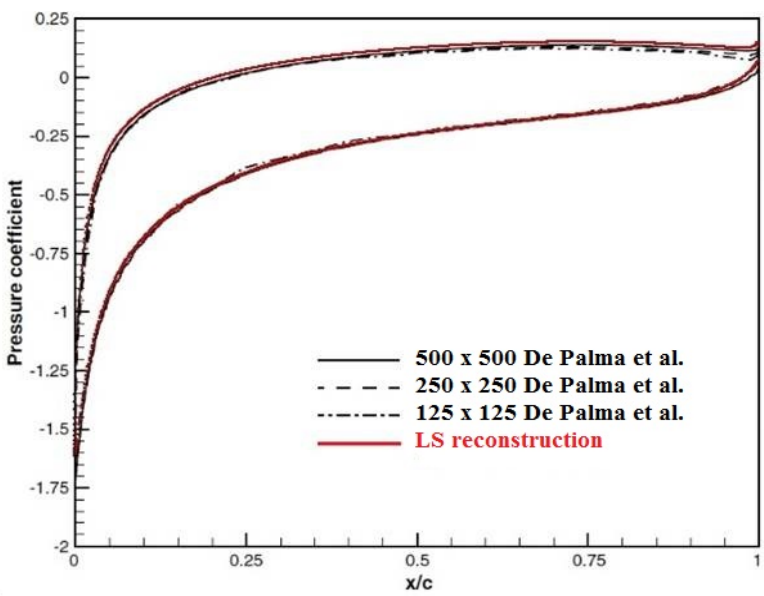


Figure 5.4: Supersonic laminar flow past an NACA0012 airfoil: pressure coefficient distributions along the profile.

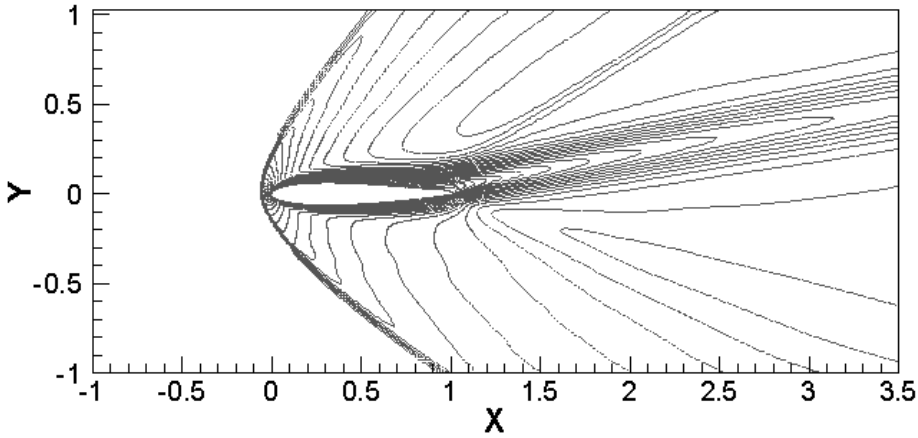


Figure 5.5: Supersonic laminar flow past an NACA0012 airfoil: Mach number contours on the finest grid, $\Delta M = 0.1$.

5.1.3 Supersonic flow past a circular cylinder

The steady turbulent supersonic flow past a circular cylinder has been considered as a suitable test case to validate the method for turbulent compressible flows, involving shocks. The case with $M_\infty = 1.7$, $Re_\infty = 2 \times 10^5$, inlet values of the turbulence kinetic energy and specific dissipation rate $k/U_\infty^2 = 0.0009$ and $\omega D/U_\infty = 40$, respectively, has been computed. For the considered value of M_∞, a bow shock is obtained upstream of the cylinder; the subsonic flow at the front part close to the wall accelerates along the surface forming a supersonic-flow region, which envelopes the subsonic recirculation region behind the cylinder, with two symmetric tail shocks at the end of the separation region. Results have been obtained using a rectangular computational domain with dimensions $[-8\,D; 9\,D] \times [-8\,D; 8\,D]$, D being the diameter of the cylinder centered at the origin. Periodic conditions have been applied in the third direction in order to simulate a two dimensional case using the three dimensional solver. Standard characteristic boundary conditions have been imposed at inlet and outlet surfaces and free-shear wall boundary conditions are imposed at the far-field boundaries. Simulations have been performed using two grids having 139777 cells. The grid is locally refined as shown in Figure 5.6.

The Mach number contours are given in Fig. 5.7, showing that a clear description of the shocks and of the wake are obtained, thanks to the resolution of the grid in those regions. The computed separation angle, measured clockwise from the leading edge, is

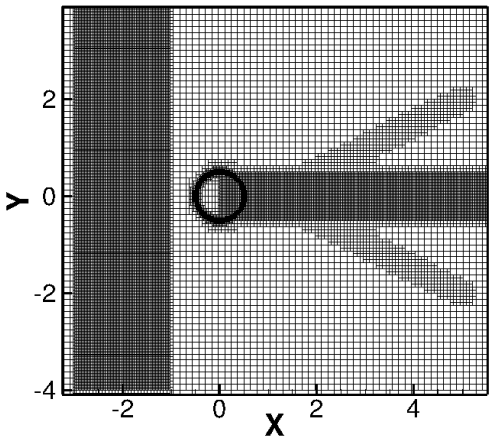


Figure 5.6: Supersonic turbulent flow past a circular cylinder: local view of the mesh.

equal to 116° which reasonably agree well with the corresponding experimental data, namely, 112°, provided in Bashkin *et al.* [77].

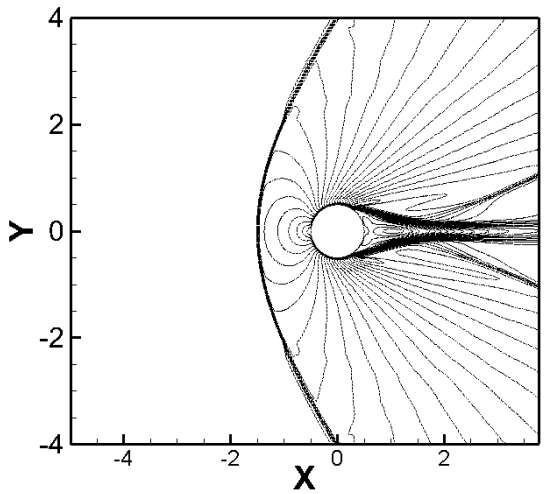


Figure 5.7: Supersonic turbulent flow past a circular cylinder: Mach number contours.

.

Finally, the computed pressure coefficient distributions along the surface of the cylinder are provided in Fig. 5.8 together with the experimental data obtained by Bashkin *et al.* [77] and the numerical data obtained by De Palma *et al.* [14]. All numerical results agree reasonably well with the literature data.

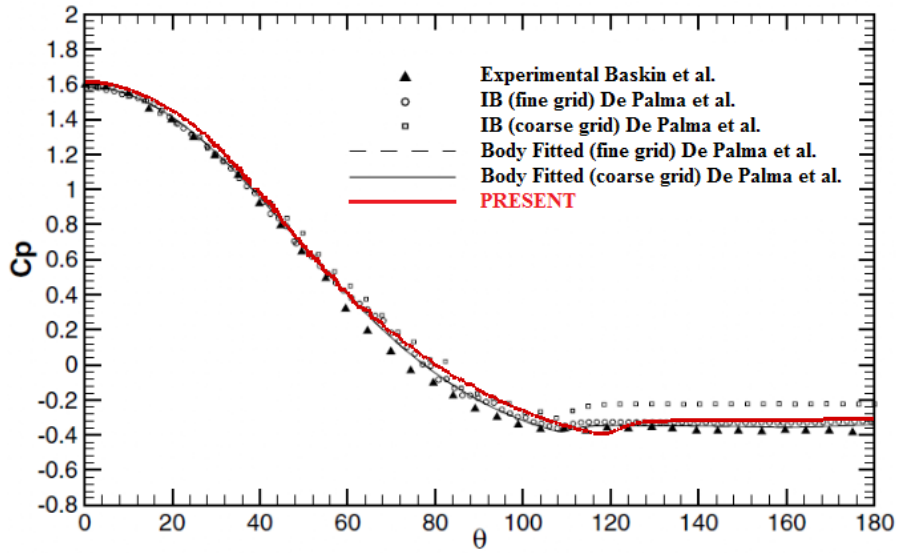


Figure 5.8: Supersonic turbulent flow past a circular cylinder: pressure coefficient distribution along the surface of the cylinder. Comparison between experimental and numerical results for $M_\infty = 1.7$.

A final comment on the experimental results is warranted. The discrepancies between the numerical solutions and the experimental results are believed to be due to three-dimensional or wall effects in the experiments, rather than to inadequate turbulence modeling, insofar as they are equally important in both the attached and separated flow regions.

5.1.4 Flow through VKI-LS59 turbine-rotor cascade

The IB method has been employed to compute subsonic and transonic flows through the high turning VKI-LS59 turbine-rotor cascade.

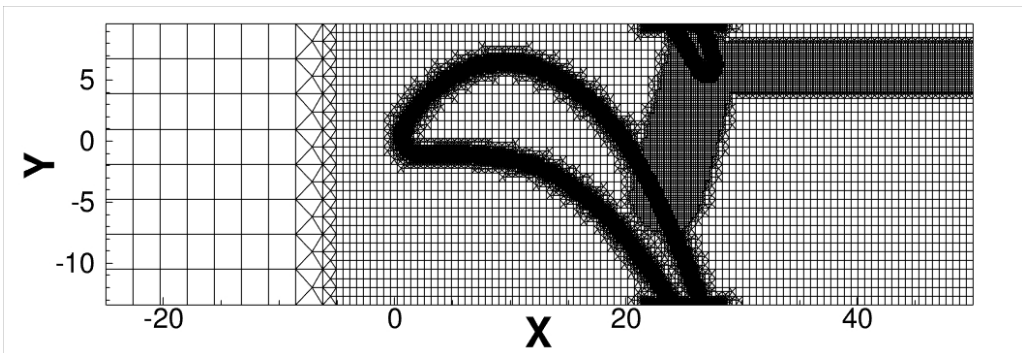


Figure 5.9: Flow through VKI-LS59 turbine-rotor cascade: local view of the grid.

Experimental data available in the literature (Sieverding [78], Kiock *et al.* [79]), indicate that the flow is nearly two-dimensional. End-wall effects and aspect ratio influence are practically negligible, so that a flow computation in two-dimensions is adequate.

Four flow conditions have been considered, with isentropic exit Mach number, $M_{2,is}$ equal to 0.810, 1.00, 1.11, and 1.20. The corresponding Reynolds numbers, based on the blade chord c and exit conditions, are 8.22×10^5, 7.44×10^5, 7.00×10^5, 6.63×10^5, whereas the inlet flow angle with respect to the axial direction is always 30°. Total conditions have been imposed at the inlet surface, static pressure is fixed at the outlet surface whereas periodic conditions are imposed in the y direction. Adaptive wall functions (Section 3.4) have been used. Computations have been performed using a 61669 cells grid refined on the immersed boundary and in the shocks regions, see Fig. 5.9.

Figure 5.10 show the Mach number contours for the four exit Mach number flow cases. In all cases, the complex structure of the flow is well predicted and the shocks are captured in the correct position.

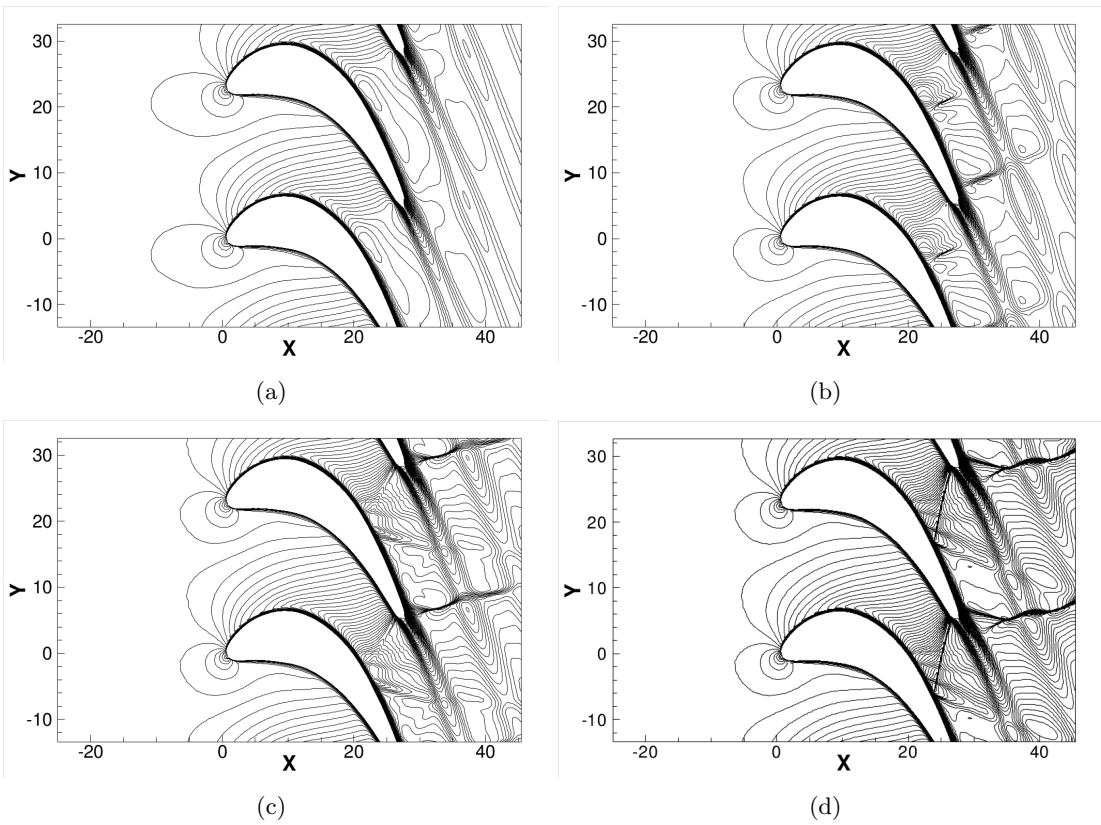


Figure 5.10: Flow through VKI-LS59 turbine-rotor cascade: Mach number contours, $\Delta M = 0.03$. (a) $M_{2,is} = 0.810$; (b) $M_{2,is} = 1.00$; (c) $M_{2,is} = 1.11$; (d) $M_{2,is} = 1.20$.

The computed isentropic Mach number distributions along the blade are shown in Figure 5.11. All the solutions agree reasonably well with the experimental data provided by Sieverding [78], and the numerical ones obtained by de Tullio [75] using the 1D reconstruction scheme.

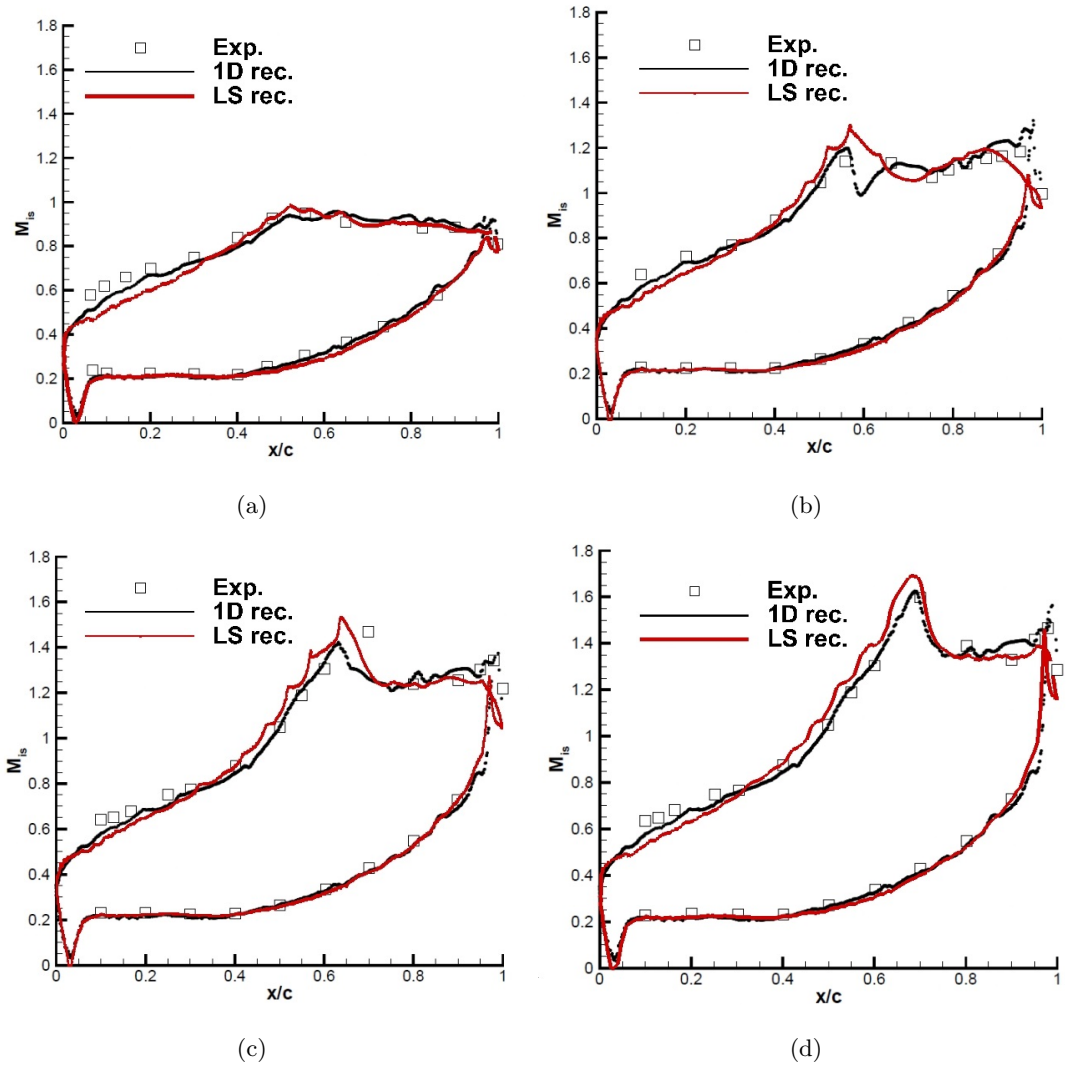
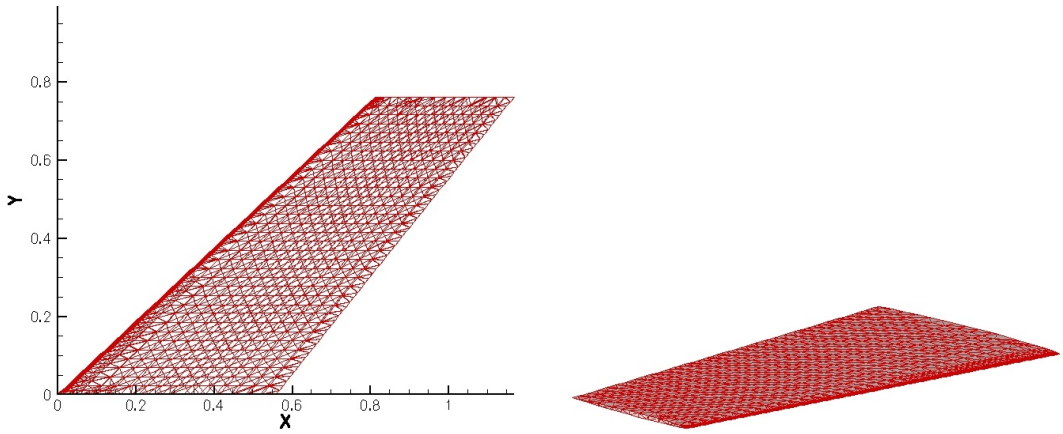


Figure 5.11: Flow through VKI-LS59 turbine-rotor cascade: isentropic Mach number distributions along the blade. (a) $M_{2,is} = 0.810$; (b) $M_{2,is} = 1.00$; (c) $M_{2,is} = 1.11$; (d) $M_{2,is} = 1.20$.

5.1.5 Transonic flow past the AGARD Wing

The IB method has been then employed to compute the transonic flow past the AGARD wing. The geometry complexity of the AGARD wing can be appreciated in Fig. 5.12, that clearly shows the very little aspect ratio of the wing thickness to the other dimensions. This is a fully three-dimensional turbulent flow well documented in literature for both steady cases [5] and dynamic response configurations, e.g., flutter configurations [80].



(a) (b)

Figure 5.12: Agard wing: 2D (a) and 3D (b) view of the triangulated mesh.

In this work, the investigation is limited to the steady case. Computations have been performed using a grid with 3833720 cells and 11596105 faces and refined on the wing surface in order to have a good resolution of the boundary layer, and in a box surrounding the wing. The imposed inlet conditions ensure $Re_\infty = 451000$, $Ma_\infty = 0.96$, $P^o_\infty = 4662.34Pa$ and $T^o_\infty = 257.81K$; the outlet static pressure is imposed equal to $2579.23Pa$; finally, the angle of attack is equal to $0°$. To reduce the computational cost of the simulation, adaptive wall functons have been imposed to the IB. The pressure coefficient for the mean aerodynamic chord is shown in Fig. 5.13 and compared with the numerical results obtained by Lee-Rausch and Batina [5]. The oscillations in the pressure coefficient behavior are probably due to the Menter condition on ω, see Eq. (4.73), when it is applied to interface cells having $y^+ < 1$.

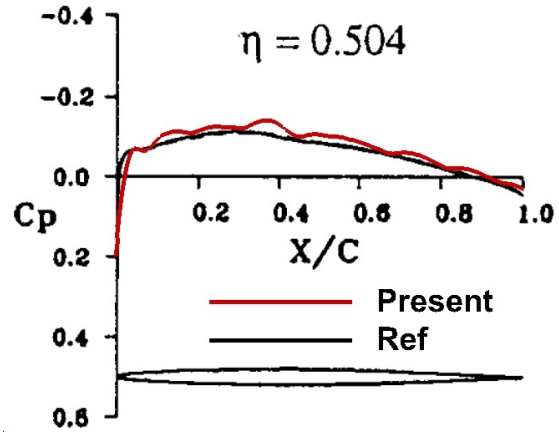


Figure 5.13: Agard wing: pressure coefficient on the mean aerodynamic chord η. Results obtained by Lee-Raush and Batina [5] are shown for comparison.

5.1.6 The transonic flow past the Unmanned Space Vehicle

The CIRA USV (Unmanned Space Vehicle) is a multi-mission, re-usable vehicle developed at CIRA, the Italian Aerospace Research Center. It is a not-propelled, winged vehicle able to perform experiments on aerodynamics. The geometrical complexity of the winged body can be appreciated in the Fig. 5.14 where the well-refined triangular mesh, used as immersed boundary in the set of simulations, is shown.

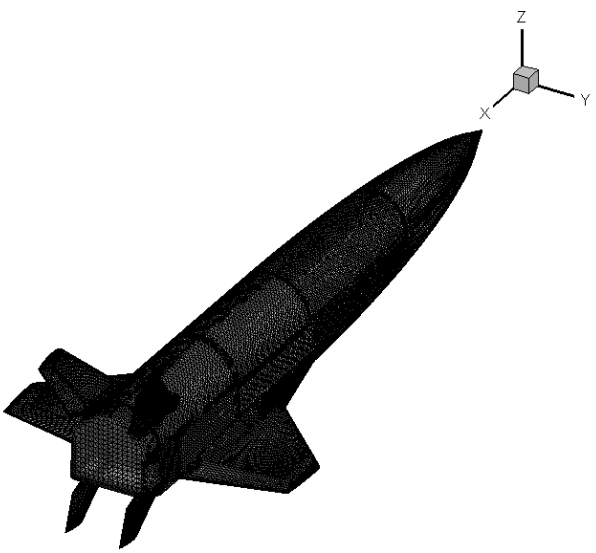
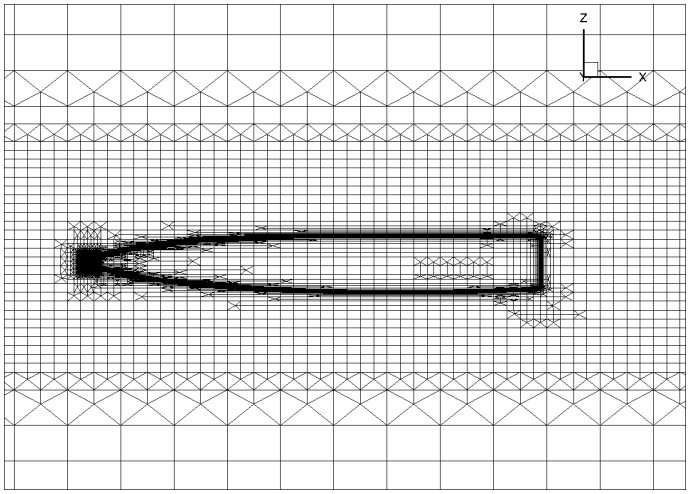
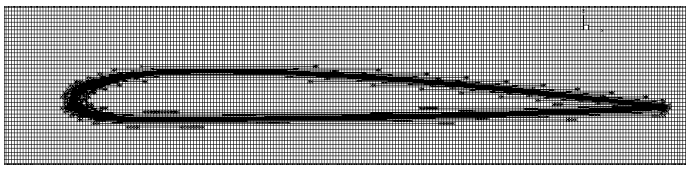


Figure 5.14: Triangular mesh of the Unmanned Space Vehicle.

Here, a numerical rebuilding result of the in-flight aerodynamic experiment carried out during the first mission DTFT (Dropped Transonic Flight Test) is presented. The aerodynamic characterization of the transonic phase of a re-entry space vehicle trajectory is critical due to the strong variability of the aerodynamic coefficients, the reason being the non-linear behaviour of the flow field. From CFD computations both global aerodynamic coefficients and local pressure distributions can be extracted, thus allowing for the comparison of CFD results with the in-flight acquired global forces and static pressure measurements [81].



(a)



(b)

Figure 5.15: Local views of the 3D mesh used for the USV simulations: USV symmetry plane $y = 0$ (a) and the right wing at $y = 1m$.

All the simulations have been performed with a clean configuration having side-slip angle $\beta = 0$, elevon deflection angles $\delta_E^{r,l} = 0$ and rudder deflection angles $\delta_R^{r,l} = 0$, where r and l are the right wing and the left wing, respectively. Calculations have been performed using 240 CPUs since the mesh contains 12341210 cells and 39875070 faces. In Fig. 5.15, local grid refinements for the body of the USV (a) and for the right wing (b) are shown.

Different values of the angle of attack, namely, 0°, 5°, 7.24°, 10°, 15°, 20°, have been considered. The analysis of the results presented here allows the understanding of the

global aerodynamics coefficients behaviour as function of the angle of attack. In Fig. 5.16 the lift coefficients at $M = 0.94$ are plotted in function of the angle of attack α and are compared to the data available in [81]. Fig. 5.17(a) provides the contours of the pressure

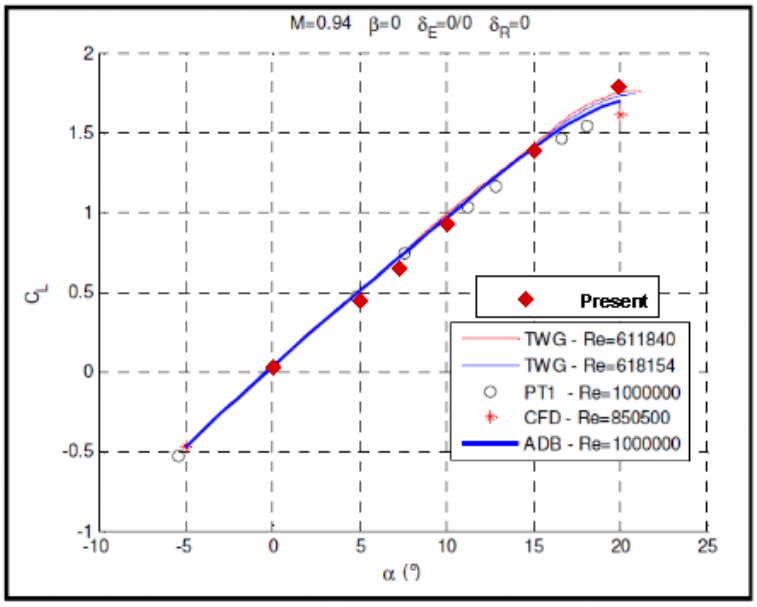
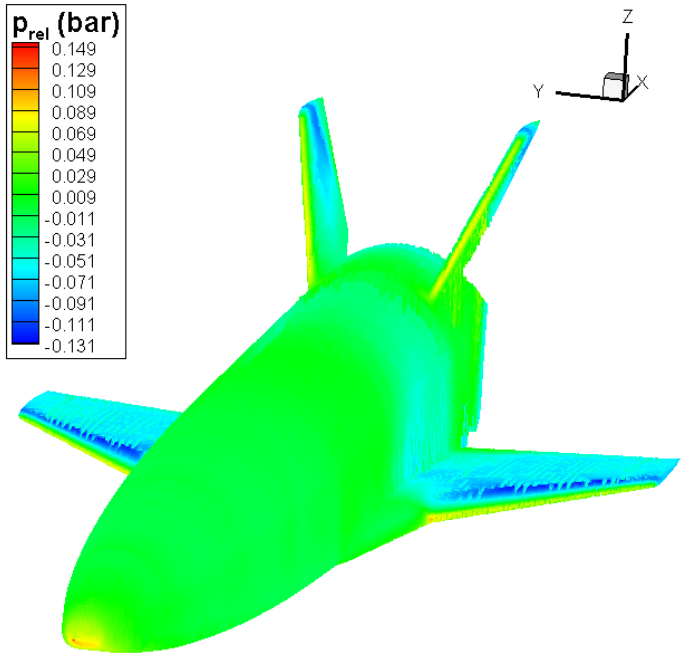
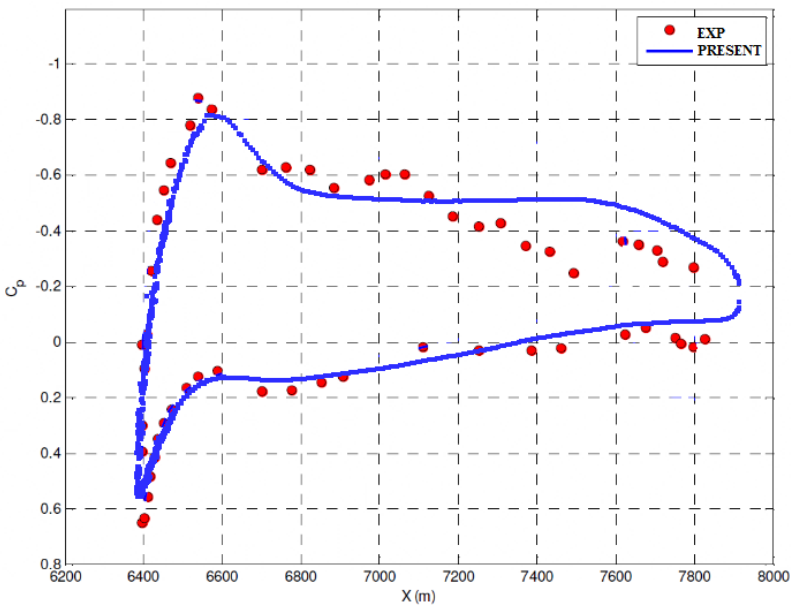


Figure 5.16: C_L vs. α at $M = 0.94$ for the clean configuration of the USV.

distribution on the USV obtained for $M = 0.94$ and $\alpha = 7.24^o$. More in detail, the comparisons between predicted and in-flight measured pressure coefficients at one wing section ($y = 1.0m$ from the vehicle's symmetry plane) are reported in Fig. 5.17(b). Even if in the experimental results carried out during the mission DTFT the elevon and rudder deflection angles had not been set equal to zero, in the numerical simulation the USV present a clean configuration with all these parameters equal to zero. Thus, the behaviour of the pressure coefficient agrees with the experimental results as less as far from the stagnation point.



(a)



(b)

Figure 5.17: a): pressure distribution at the USV surface at $M = 0.94$ and $\alpha = 7.24^o$. b) pressure coefficient at wing section $y = 1.0m$ ($M = 0.94$, $\alpha = 7.24$), obtained with the present IB-URANS solver are compared to the experimental results.

5.2 CHT-IB-URANS validation

5.2.1 Conjugate-heat-transfer in a rotating heated fluid

The heat transfer problem in a rotating flow inside a tube–for which an analytical solution of the 2D Navier–Stokes equations is available [28]–has been considered at first to validate and compare the three different IB wall treatments presented in the previous sections within a CHT approach. The fluid is contained between a stationary hollow inner cylinder and a rotating outer one. The outer cylinder has a radius $R_o = 1.8m$, moves with tangential velocity and is kept at $T_o = 700K$. The inner hollow cylinder has radii equal to $R_m = 0.9m$ and $R_i = 0.45m$, respectively, the inner surface is kept at $T_i = 500K$ and $K_s/K_f = 9$. The geometry is shown in Figure 5.18.

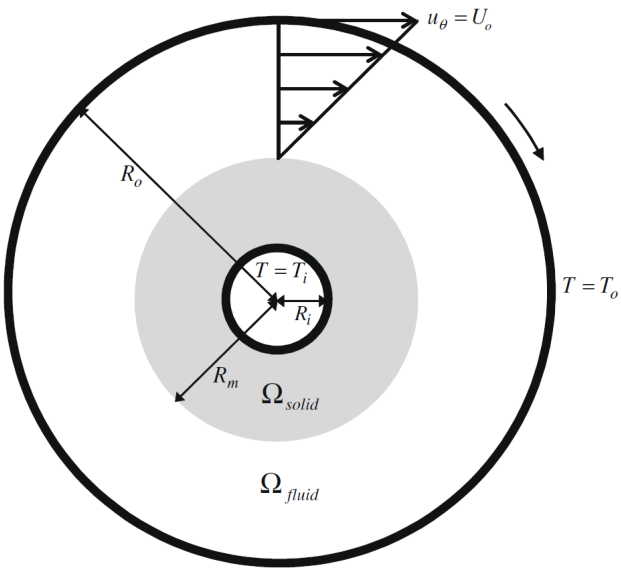


Figure 5.18: Rotating tube geometry setup.

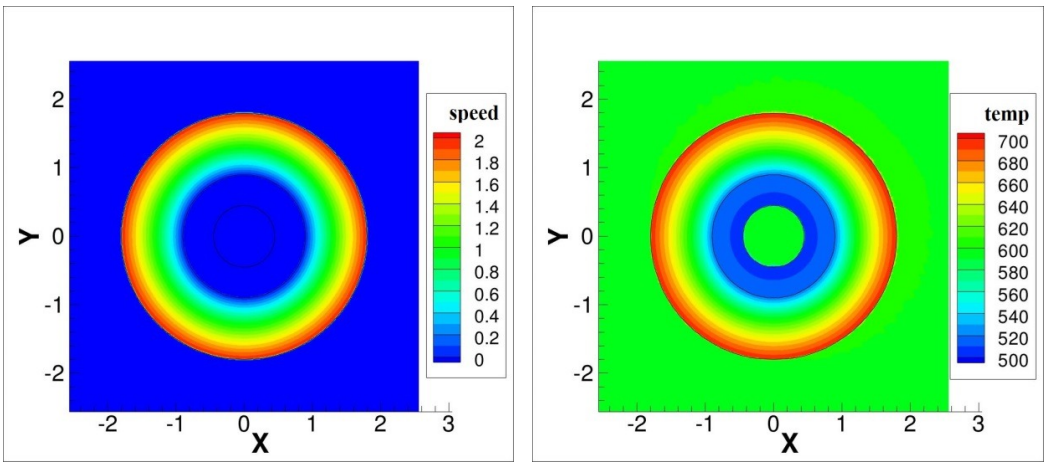
The velocity distribution is given as

$$u_r = 0, \qquad u_\theta(r) = \begin{cases} 0 & \text{for } R_i \leq r \leq R_m \text{ (solid)}, \\ -\frac{R_o R_m^2 U_o}{(R_o^2 - R_m^2) r} - \frac{R_o U_o}{(R_o^2 - R_m^2)} r, & \text{for } R_m \leq r \leq R_o \text{ (fluid)}, \end{cases} \tag{5.1}$$

whereas the temperature distribution is given as

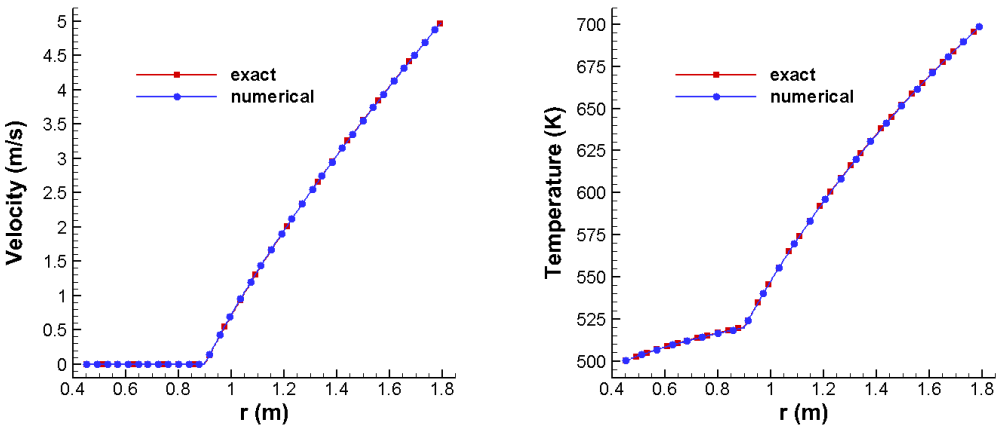
$$T(r) = \begin{cases} T_i + \frac{T_o - T_i}{\log\left(\frac{R_m}{R_i}\right) + \left(\frac{K_s}{K_f}\right)\log\left(\frac{R_o}{R_m}\right)} \log\left(\frac{r}{R_i}\right), & \text{for } R_i \leq r \leq R_m \text{ (solid)}, \\ T_o - \frac{T_o - T_i}{\log\left(\frac{R_o}{R_m}\right) + \left(\frac{K_f}{K_s}\right)\log\left(\frac{R_m}{R_i}\right)} \log\left(\frac{R_o}{r}\right), & \text{for } R_m \leq r \leq R_o \text{ (fluid)}. \end{cases} \tag{5.2}$$

Computations have been performed using periodic boundary conditions in the axial direction—to solve a 2D flow by a 3D code—and a uniform Cartesian grid with $\Delta x = \Delta y = 0.16, 0.08, 0.04, 0.02, 0.01$. Figure 5.19 show the finest-grid (tangential) speed- and temperature-contours using the least squares reconstruction scheme. Fig. 5.20 shows the speed (a) and temperature (b) radial distributions obtained with the finest grid. The exact analytical solutions are also reported for comparison. It is worth remarking that the IB method provides very smooth contours, also using the other IB reconstruction schemes.



(a) (b)

Figure 5.19: contours of the (tangential) speed (a) and temperature (b) obtained using the least squares reconstruction scheme.



(a) (b)

Figure 5.20: radial velocity distribution (a) and radial temperature distribution (b).

Using the analytical solution, for each numerical solution, the mean square error (mse)

and the maximum error (err_{max}) have been computed:

$$mse = \frac{1}{n_{cells}} \sqrt{\sum_{i}^{n_{cells}} err_i^2}, \qquad err_{max} = \max(err_i). \tag{5.3}$$

Firstly, a set of simulations have been provided imposing the analytical value of the wall temperature also for the fluid and solid interface ($r = R_m$), and using one of the three IB reconstructions (Sec. 3.1, 3.2, 3.3) indicated as 1D, IDW and LS, respectively. All of the solutions well predict the tangential-speed and temperature fields within the flow and the hollow tube. The corresponding computed errors are reported in Figs. 5.21 and 5.22 and show that the IB reconstructions do not affect the second order of the global spatial accuracy, while the maximum errors are first-order-accurate.

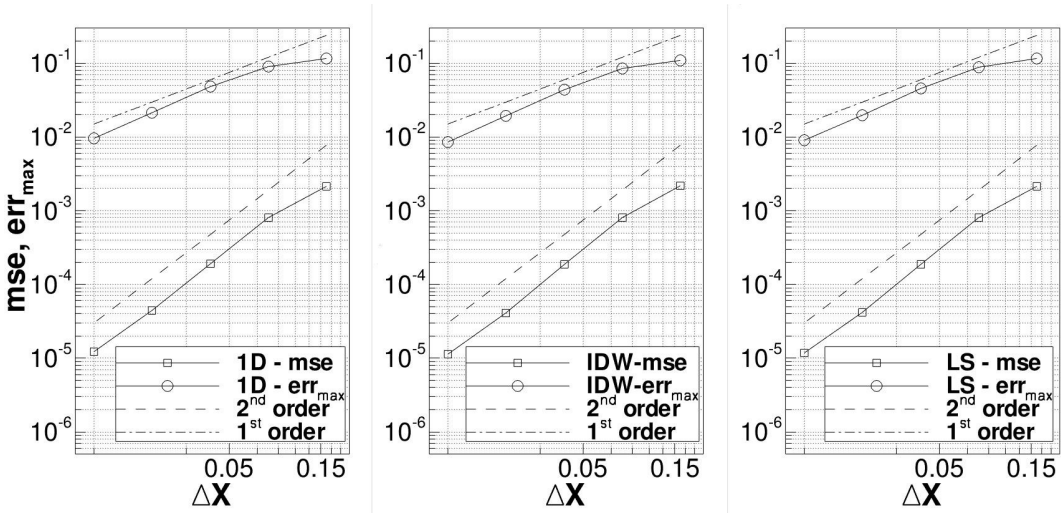


Figure 5.21: computed mse and err_{max} of the velocity obtained imposing the wall temperature at $r = R_m$ and using the different IB reconstructions.

Then, the same computations have been performed using the CHT conditions at $r = R_m$ and ns and nf (see Section 4.3) both equal to 7. The order of the spatial accuracy is not modified, see Figs. 5.23 and 5.24, and the temperature errors are slightly higher, as anticipated. Such results clearly indicate the validity of the three CHT-IB interface reconstructions.

A more detailed analysis has been conducted in order to better understand the merits and limitations of the three different reconstruction schemes. Because of the Cartesian grid that uses FCPPs and SCPPs at different distances from the corresponding interface

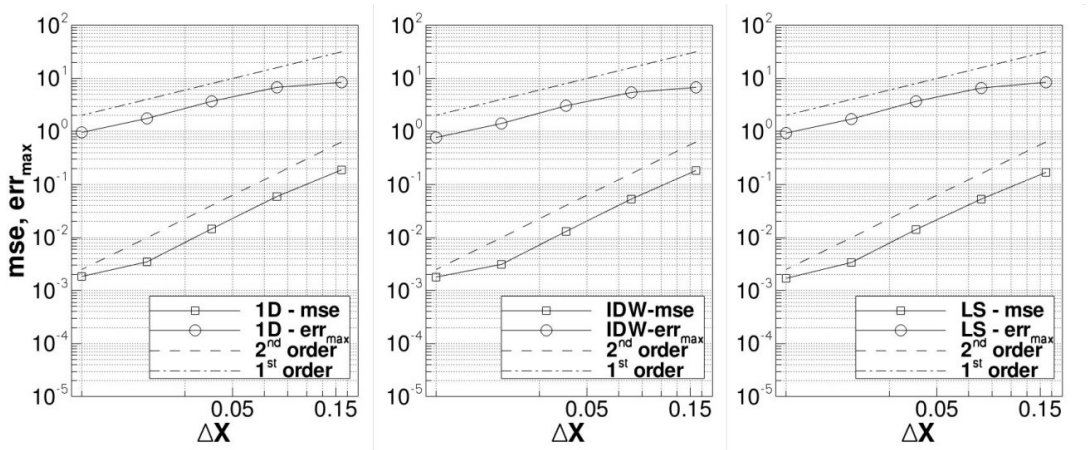


Figure 5.22: computed mse and err_{max} of the temperature obtained imposing the wall temperature at $r = R_m$ and using the different IB reconstructions.

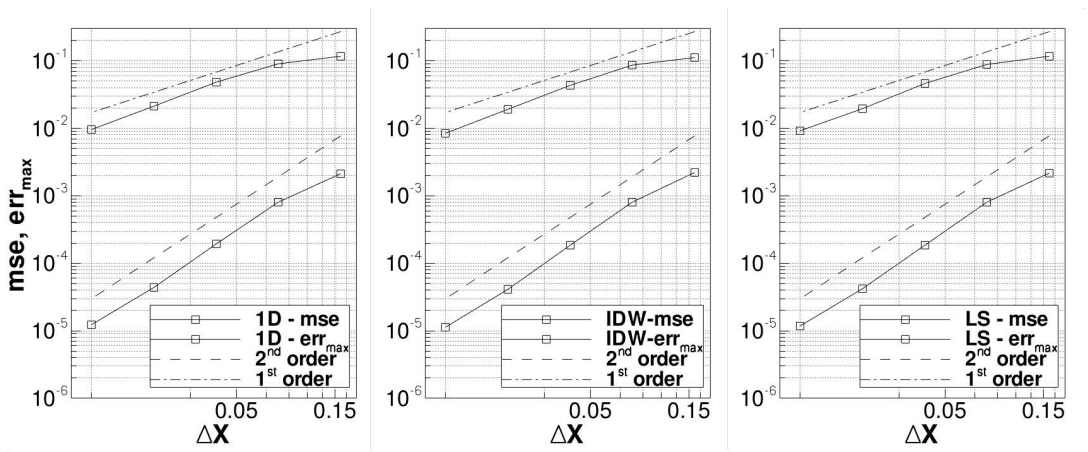


Figure 5.23: computed mse and err_{max} of the velocity obtained using the different CHT-IB reconstructions.

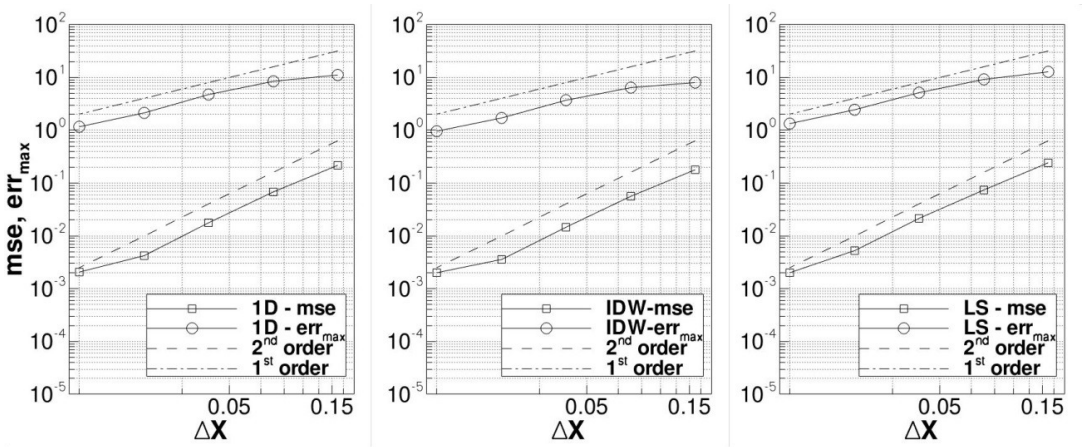


Figure 5.24: computed mse and err_{max} of the temperature obtained using the different CHT-IB reconstructions.

cells, the numerical solutions are not uniform along the cylinders contours, but experience a periodic behaviour, characterized by both low- and high-frequency errors, see Fig. 5.25. It appears that the 1D and IDW reconstructions experience the largest high and low frequency errors, respectively, whereas the LS reconstruction dramatically reduces the high frequency error but provides the least accurate mean temperature. It is somewhat surprising that the 1D reconstruction provides the most accurate mean temperature, probably due to the fact that it employs a computational node value in the interpolation procedure.

5.2.2 Flow past a heated cylinder in cross-flow

A more challenging test case, involving unsteady flow, natural convection, transition to turbulence and heat transfer from and to an internally heated cylinder in water cross-flow has been used to further validate the present IB-CHT method, namely, the flow past a heated cylinder in cross-flow, for which both experimental and numerical results using a body fitted CHT-RANS solver are available [36]. The tube has a length equal to 61 cm and a height equal to 7.62 cm, as shown in figure 5.26. The center of the tube has coordinates 42.7 cm and 1.43 cm in the streamwise and wall-normal directions, respectively; the inner and outer diameters of the tube are equal to 0.635 cm and 1.587 cm, respectively, and the Reynolds number, based on the streamwise velocity and the tube diameter, is 189. The inlet velocity has been imposed equal to 1.09 cm/s and the inlet temperature of the water is 284 K; the heated bottom surface is kept at a constant temperature equal to 318 K

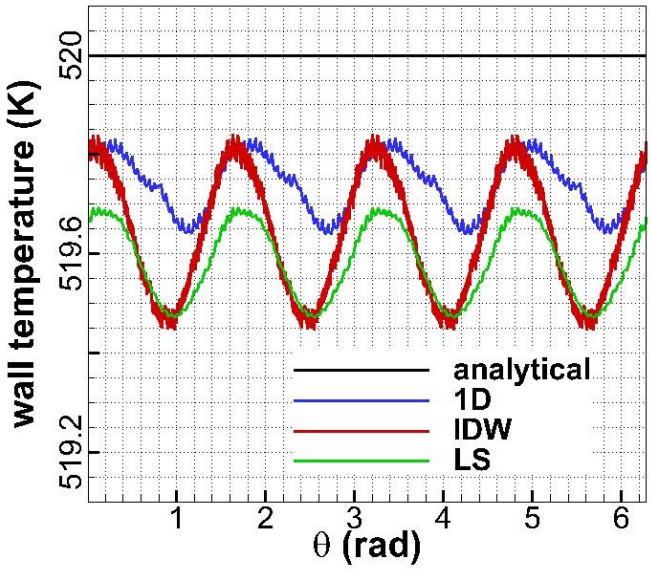


Figure 5.25: computed wall temperature at $r = R_m$ using the different CHT-IB methods and $\Delta x = \Delta y = 0.01$

and the top wall is considered adiabatic; the core temperature of the tube is considered constant and equal to 318 K; the outlet pressure has been imposed equal to 1 bar. The ratio between the solid and fluid thermal conductivities, $K_s/K_f = 35$.

Again, in the present calculations, two internal arrays of cells and two arrays of ghost cells have been used in the third direction to impose periodic conditions so as to compute a 2D CHT problem using a 3D code. The finest locally refined computational grid, used in this work and verified to provide grid-converged solutions has a total of 152375 cells in the x-y plane, see Fig. 5.27.

A physical time step of 0.005 s has been used and a local pseudo-time step corresponding to = 0.5. The CPU time for each physical time step and a fixed number of inner iterations equal to 80, using 2 Xeon 10-core E5-2660v3 (2.6 Ghz) is equal to 7.8 s. As far as the temperature is concerned, Laskowski *et al.* [36] provide only the mean value of the measured temperature which is equal to 311.3 K, whereas the present 1D, IDW and LS computations provide mean values of 314.5 K, 312.1 K and 312.0 K, respectively and time-averaged temperature profiles of the outer surface of the cylinder, as shown in Fig. 5.28(a). The IDW and LS schemes are clearly superior insofar as they provide a more accurate mean value and considerably smoother time averaged circumferential profiles. As far as the more challenging heat flux computations, the present time-averaged heat

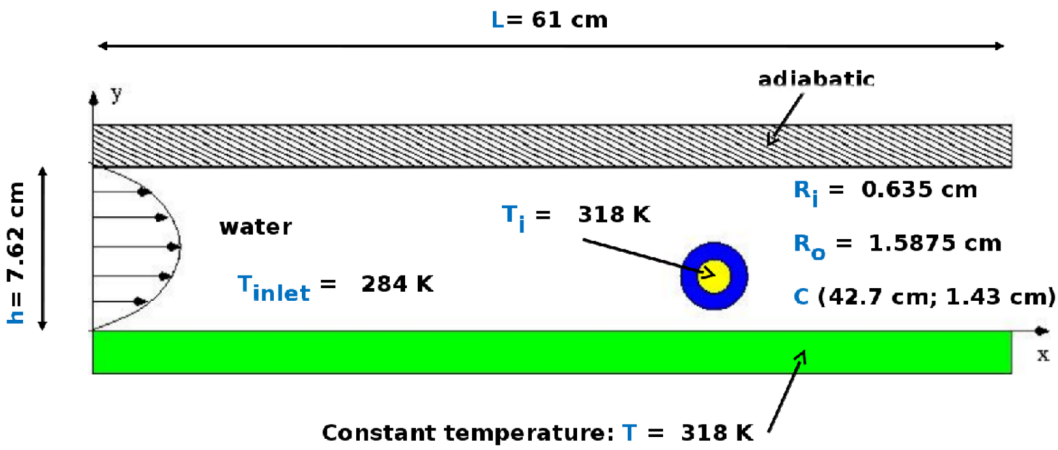


Figure 5.26: Data-set of the numerical simulation of the flow past a heated cylinder in a channel flow.

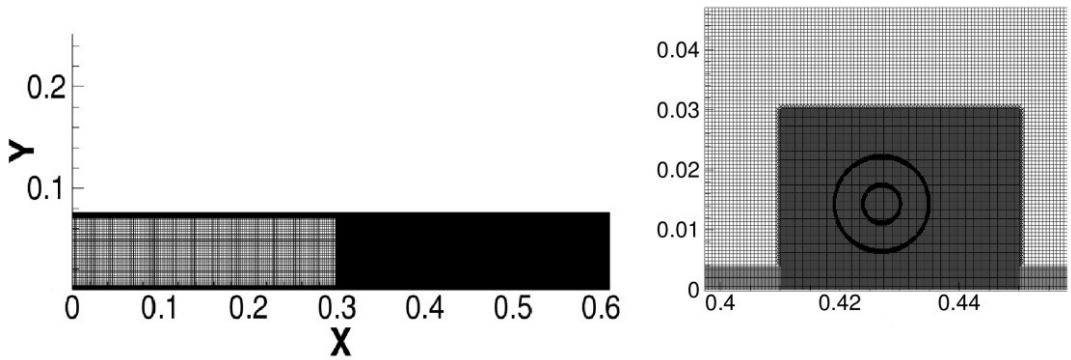
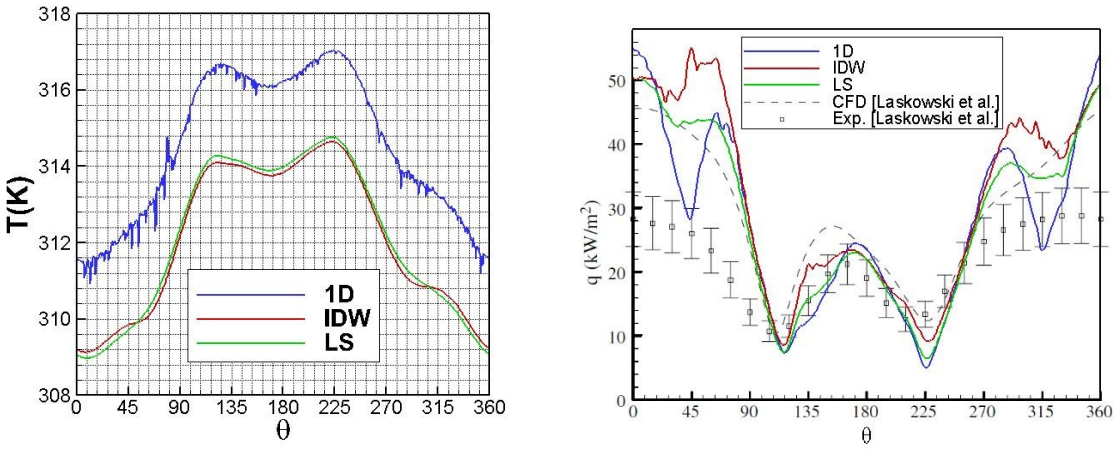


Figure 5.27: locally refined computational grid.

fluxes q are presented in Fig. 5.28(b), where both the measured and the numerical results by Laskowski *et al.* [36] are also given for comparison. It clearly appears that only the results obtained with the LS reconstruction are acceptable: it is noteworthy that also the body-fitted-RANS solutions do not agree with the experimental ones by the same authors.



(a) (b)

Figure 5.28: contours of the (tangential) speed (a) and temperature (b) obtained using the least squares reconstruction scheme.

5.2.3 Conjugate-heat-transfer in an internally cooled C3X vane

The flow past the C3X turbine guide vane of Hylton *et al.* [37]–for which detailed experimental results are available–was finally selected to test the method versus a CHT problem of industrial interest. The three-dimensional computational domain included one vane, with periodic boundary conditions employed to simulate the cascade test condition, as well as 10 cooling channels. The domain extended from the hub to the shroud, the computational inlet/outlet being located at about one chord length upstream/downstream of the vane leading/trailing edges, respectively.

The operating conditions, listed in Tab. 5.1, are the ones of the test-case R112 documented in [37]. The hot gas total pressure $P^h_{t,in}$ and total temperature $T^h_{t,in}$ were specified at the inlet where the turbulence intensity was imposed to be the experimental value of 8.3%. The static pressure $P^h_{s,out}$ (derived using $P^h_{t,in}$ and $T^h_{t,in}$) was specified at the exit. The coolant flows in the ten channels are independent of each other and fully developed

at the channel inlet (i.e., the hub of the vane), being fed by long tubes. The mass flow rates (FR) for each channel, as reported by Hylton *et al.* [37] are listed in Tab. 5.2, which also provides the hub total pressure $P^c_{t,in}$ and temperatures $T^c_{t,in}$, being specified to match the corresponding flow rate, see [38]. A static pressure $P^c_{s,out}$ equal to 1 bar was imposed at the exit of the channels. Finally, the vane material is stainless steel (ASTM Type 310), with thermal conductivity $K_s = 0.0182T + 6.13$. It is noteworthy that in addition to the complexity of the flow, the test-case emerges as a very challenging CHT problem being the ratio between the solid and fluid thermal conductivities, K_s/K_f almost equal to 500.

$P^h_{t,in}(bar)$	$T^h_{t,in}(K)$	Ma^h_{out}	$P^h_{s,out}(bar)$	$Tu_{in}(\%)$
3.217	783.0	0.9	1.925	8.3

Table 5.1: Flow conditions of the R112 test-case

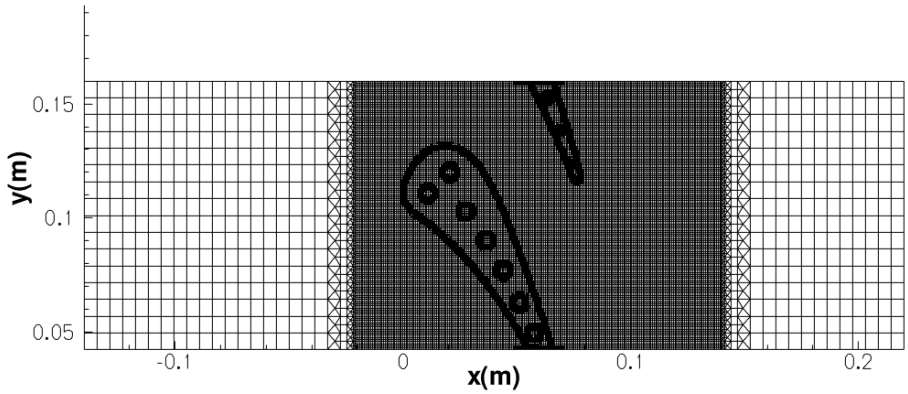
Coolant (#)	Diameter (mm)	$P^c_{t,in}(bar)$	$T^c_{t,in}(K)$	$P^c_{s,out}(bar)$	FR (g/s)
1	6.30	1.539	412.2	1.0	7.79
2	6.30	1.386	408.7	1.0	6.58
3	6.30	1.336	389.0	1.0	6.34
4	6.30	1.386	396.0	1.0	6.66
5	6.30	1.351	372.1	1.0	6.50
6	6.30	1.420	435.4	1.0	6.72
7	6.30	1.344	384.2	1.0	6.33
8	3.10	2.015	372.3	1.0	2.26
9	3.10	1.449	420.0	1.0	1.38
10	1.98	1.872	433.2	1.0	0.68

Table 5.2: Coolant flow conditions of the R112 test-case.

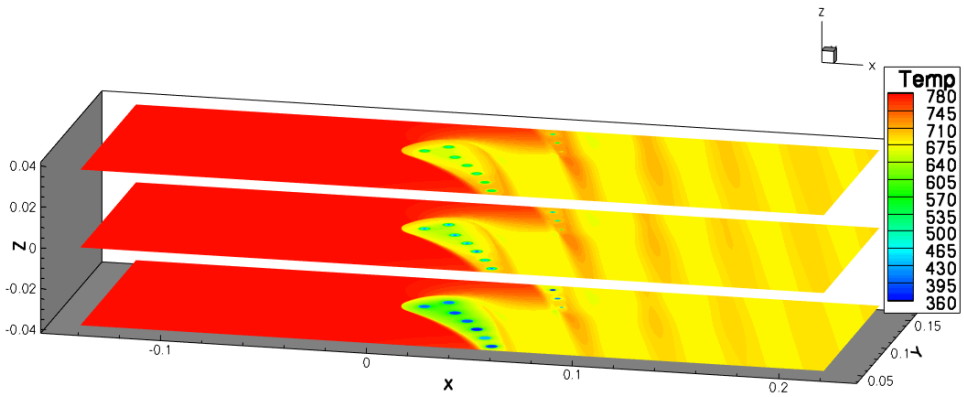
Results have been obtained with four meshes having a total number of cells equal to 69624×8^n cells, n being equal to 0, 1, 2, 3, respectively; namely, each finer grid was obtained by doubling the number of cells of the coarser one in each space dimension. The grids are smaller within the holes and the vanes and are locally refined, as shown in Fig. 5.29(a), where the coarsest grid at mid-span is shown. It is noteworthy that the periodic boundary conditions simulating the cascade test condition can be imposed within the IB method even if the vane height is larger than the vane spacing. Fig. 5.29(b) shows the temperature contours for three different z-planes, using the finest grid, to give an overall idea of the cooling process. Finally, Fig. 5.29(c) and 5.29(d) provide the computed and measured mid-span pressure and temperature distributions on both sides of the vane, respectively. It is noteworthy that in Fig. 5.29(d) the pressure side of the blade is indicated

by negative values of x.

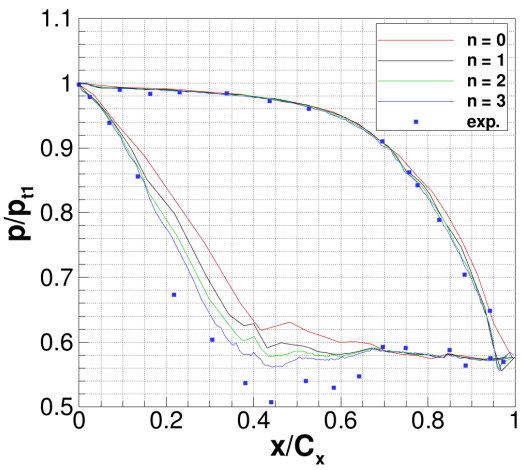
The results show that the pressure distribution on the pressure side of the vane is well captured even by the coarsest grid, whereas even the finest grid with about 32 million cells does not provide a grid converged pressure distribution on the suction side of the vane. As far as the temperature distribution is concerned, the coarsest grid is seen to be quite inadequate, but the finer grids show a consistent behavior tending towards grid convergence, but some of the flow features near the holes are not yet completely captured. Moreover, the differences between the computed values and the experimental ones can be considered acceptable for such a very complex turbulent flow and it is in the authors opinion very difficult to decide which are the more correct ones. For this reason, current work aims at implementing velocity and temperature wall functions, which are anticipated to further improve the accuracy of the method. At the moment, fully converged solutions on the 4 and 32 million-cells-grids require about 100 and 300 CPU hours using 16 and 32 Xeon 10-core E5-2660v3 (2.6 Ghz).



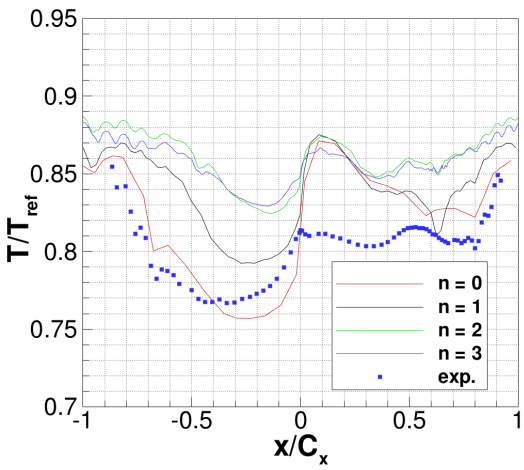
(a)



(b)



(c)



(d)

Figure 5.29: C3X vane: locally refined grid (a); temperature contours for three different z-planes (b); mid-span pressure (c) and temperature (d) distribution on the external surface of the vane; C_x is the axial chord of the vane.

5.3 FSI validation

5.3.1 Oscillating circular cylinder in a cross-flow

The case of a circular cylinder transversely oscillating in a cross-flow is considered in order to validate the method in case of a moving geometry with a prescribed motion. As showed by Uhlmann [82], the direct forcing approach of Fadlun et *al.* [11], can lead to large fluctuations of the hydrodynamic forces, when it is coupled with a fractional time-step solver. This suggests us to use an IB reconstruction as smooth as possible also in the present URANS approach, to reduce the high frequency errors.

The Reynolds number, based on the cylinder diameter D and the free-stream velocity U, is equal to 185. The ratio of the forcing frequency, f_e, to the natural shedding frequency, f_0, is equal to 1, with $f_0 D/U = 0.195$. The motion of the cylinder follows a sinusoidal law, $y(t) = A_0 \sin(2\pi f_e t)$, with $A_0 = 0.2D$. The constant physical time-step used in the computation is $\Delta t = 0.001 D/U$.

The time histories of the computed drag and lift coefficients, C_D and C_L are given in Fig. 5.30 showing a very smooth behavior.

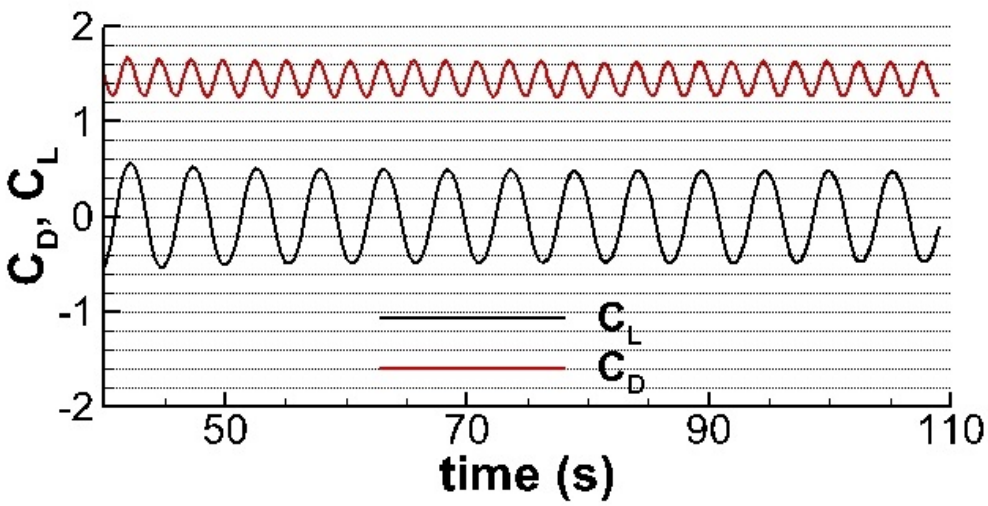


Figure 5.30: Oscillating circular cylinder in a cross-flow: drag and lift coefficients as a function of time, $f_e = f_0$.

The different behaviour of the force coefficients is captured accurately, and the results are in very good agreement with those obtained by Guilmineau and Queutey [6] obtained using a body-fitted approach, and the ones obtained by de Tullio *et al.*(2012) [7] using a

fractional time-step incompressible solver. The distribution of pressure and skin-friction coefficients, C_p and C_f, respectively, on the cylinder surface are shown in Figure 5.31, at the time instant corresponding to the extreme upper position. The results are in a very good agreement.

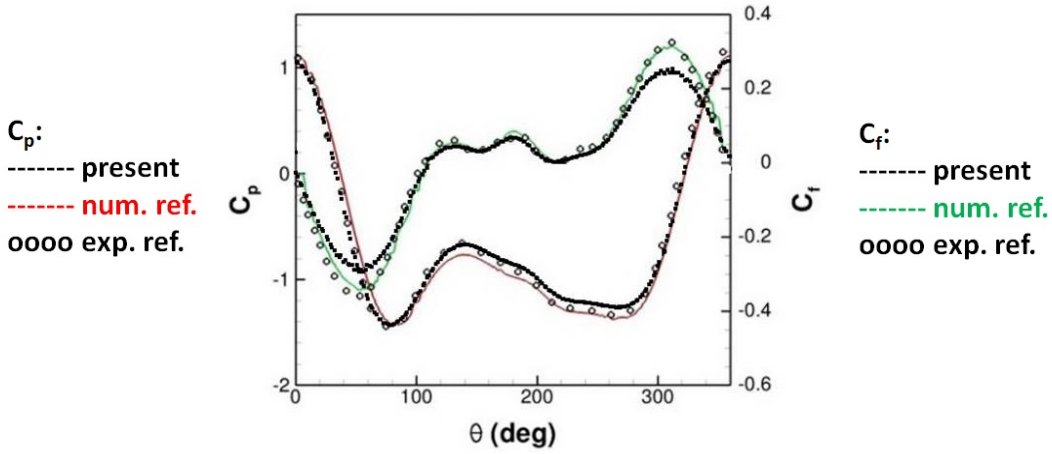
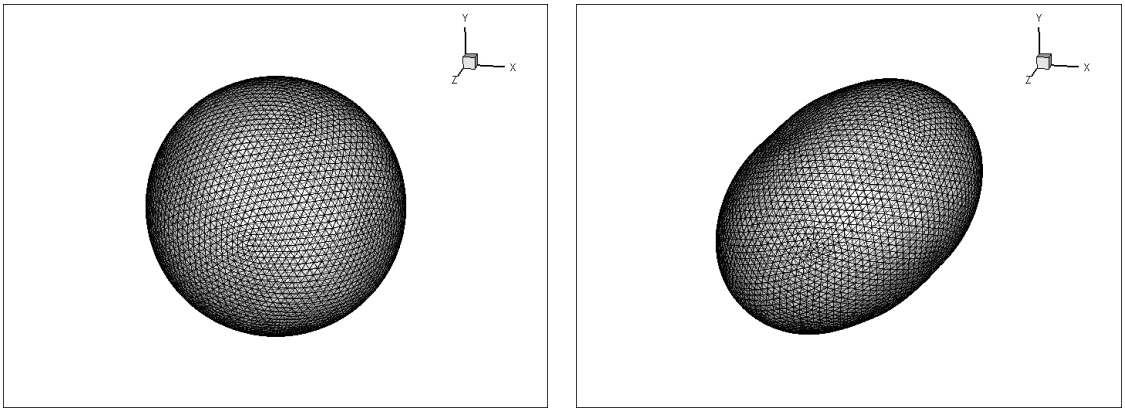


Figure 5.31: Oscillating circular cylinder in a cross-flow: pressure and skin friction coefficients, C_p and C_f, when the cylinder is located at the extreme upper position, compared with the experimental data provided by Guilmineau and Queutey [6] and the numerical ones obtained by de Tullio *et al.*(2012) [7].

5.3.2 Motion of a spherical microcapsule freely suspended in linear shear flow

Here, the motion of a spherical microcapsule made of neo-Hookean material in a shear flow is studied. Barthès-Biesel [8] provides a perturbation solution valid for small values of the ratio ϵ of viscous deforming forces to elastic shape-restoring forces ensuring that the deformation of the sphere remains small. The flow between two parallel surfaces is driven by the shear rate $\dot{\gamma} = |U|/(h)$ where U is the velocity of the moving plate and h is distance between the two parallel plates. Here the provided results have been obtained for $\dot{\gamma} = 0.015$ and $\epsilon = 0.03$. Since ϵ is very small, the material can be modeled as a neo-Hookean (hyperelastic) material using a value of the Lamè parameters μ and λ equal to 1/30 and 1/15, respectively. The bending constant k_b has been fixed equal to 0.1 and the volume constraint constant k_v equal to 1.0. Starting from its initial condition (Fig. 5.32(a)), the microcapsule reaches a steady deformed configuration (Fig. 5.32(b)).

The converged solution has been reached after $25s$ using a physical time-step equal to



(a) (b)

Figure 5.32: Spherical microcapsule freely suspended in linear shear flow: initial condition (a) and converged solution (b)

0.005s and 100 inner iterations for each physical time-step. The forces exerted upon the surface by the fluid are calculated at the end of the inner iterations and are fed to the structural solver which updates the new position of the surface mesh points. The comparison of the converged final solution with the analytical solution proposed by Barthès-Biesel [8] is shown in Fig. 5.33. The results obtained using the proposed methodology are in a good agreement with the analytical data.

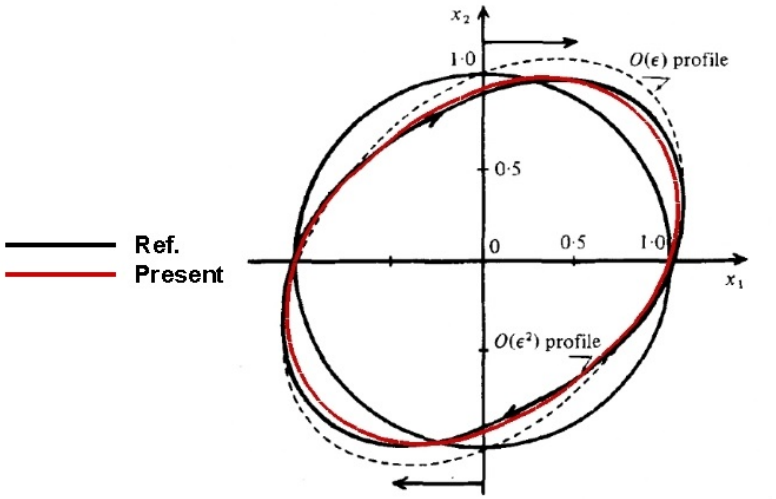


Figure 5.33: Spherical microcapsule freely suspended in linear shear flow: comparison with the analytical solution proposed by Barthès-Biesel [8].

Conclusions

The purpose of this work has been to develop and test an accurate and efficient tool for computing three-dimensional complex flows past fixed or moving geometries, for a wide range of Reynolds and Mach numbers.

Firstly, an accurate and efficient IB method, using a state-of-the-art URANS parallel Cartesian solver, has been improved, by means of a new IB treatment, extended to three space dimensions, and validated versus several test cases of increasing complexity. The new IB treatment based on a least-squares approach, performs best in all applications.

Then, a code for solving the HC equation that uses the same spatial discretization and time-marching scheme as the URANS solver has been developed and coupled with it to obtain an efficient tool for solving CHT problems. The CHT-IB solver has proven to compute the flow and temperature fields within a rotating heated fluid, a heated channel flow past a cylinder and in and around an internally cooled C3X vane.

Finally, a surface-based structural solver that simulates the dynamics of deformable geometries, discretized by triangulated Lagrangian meshes, has been coupled with the basic IB-URANS method to provide an efficient tool for solving FSI problems. The FSI-IB solver has proven to compute the flow past an oscillating circular cylinder and the three-dimensional incompressible flow past a deformable sphere.

Current and future work aim at:

- developing a CHT boundary treatment based on wall functions so as to render the proposed method a formidable tool for computing very challenging industrial flows, such as the transonic high Reynolds number flows within an entire turbine;
- extending the FSI methodology to aeroelasticity problems in turbomachinery;
- extending the method to hypersonic flows characterized by ionization and dissociation.

Appendix A

Segment triangle intersection

In the followings an efficient method for the computation of the intersection in space between a segment and a triangle is described.
Assume that $\mathbf{Q} = (x_Q, y_Q, z_Q)$ is the query point and $\mathbf{C} = (x_C, y_C, z_C)$ is an (external) control point. The segment $\mathbf{QC}$ is the ray $\mathbf{r} = (x_Q - x_C, y_Q - y_C, z_Q - z_C)$ and any point on the ray can be spanned by the parameter $u \in [0, 1]$ through

$$x_U = x_C + u(x_Q - x_C), \qquad y_U = y_C + u(y_Q - y_C) \qquad \text{and} \qquad z_U = z_C + u(z_Q - z_C) \quad \text{(A.1)}$$

or in vector form $\mathbf{U} = \mathbf{C} + u\mathbf{r}$. In order to check if the ray $\mathbf{r}$ intersects a triangle (given by the three vertices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$) it is convenient to verify first if the ray intersects the plane containing the triangle. A plane in space is represented by the equation $ax + by + cz = d$ or equivalently by the inner product between the plane normal vector $\mathbf{n}$ and a vector $\mathbf{x} = (x, y, z)$, namely $\mathbf{n} \cdot \mathbf{x} = d$ (recall that $\mathbf{n} = (a, b, c)$ is not a unit vector). This has the simple geometrical interpretation that any point $\mathbf{x}$ belonging to the plane has the same projection length d on $\mathbf{n}$.
It should be noted that given the coordinate of the triangle vertices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ the coefficients a, b, c and d of the plane equation are not known immediately. Of course the fact that $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ all belong to the plane could be sufficient to determine, by a linear system, the ratios a/d, b/d and c/d [1] even if the procedure can be made simpler. In fact, we know that $\mathbf{n}$ is a vector orthogonal to the plane while $\mathbf{B} - \mathbf{A}$ and $\mathbf{C} - \mathbf{A}$ both lie on the plane. The cross product $(\mathbf{B} - \mathbf{A}) \times (\mathbf{C} - \mathbf{A})$ is in fact $\mathbf{n}$ from which a, b and c are immediately available. The remaining coefficient d can be simply found by substituting the coordinates of one of the points $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ in the the equation for the plane. With

[1] This can be done provided $d \neq 0$. Should be $d = 0$, however, the plane equation contains only a, b and c and the passage of the plane through $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ is sufficient for the computation of the coefficients.

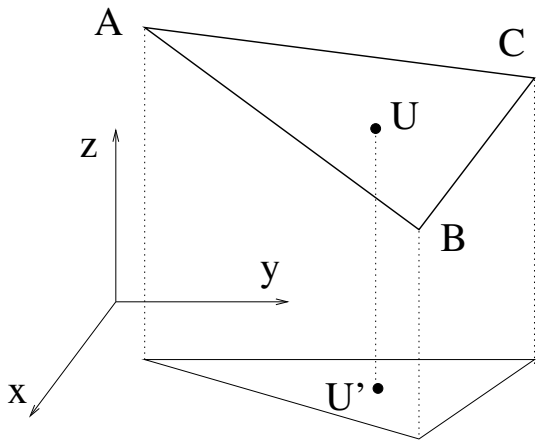


Figure A.1: Projection of a triangle and its intersection point on a coordinate plane.

the equation for the plane at hand the computation of the ray/plane intersection is very easy upon considering that the condition of a point on the ray $\mathbf{U}$ belonging to the plane is $\mathbf{U} \cdot \mathbf{n} = d$; by substitution this yields

$$(\mathbf{C} + u\mathbf{r}) \cdot \mathbf{n} = d \qquad u = \frac{d - \mathbf{C} \cdot \mathbf{n}}{\mathbf{r} \cdot \mathbf{n}}, \tag{A.2}$$

which plugged into Equation (A.1) yields the coordinates of the intersection point $\mathbf{U}$.
The final task is to check if the intersection $\mathbf{U}$ between the ray and the plane is inside or outside the triangle. Although both elements are in three-dimensions the intersection problem can in fact be reduced to a two-dimensional problem by introducing a reference frame on the plane of the triangle and considering only the intersection point. Even this rotation, however is unnecessary when considering that $\mathbf{U}$ is within the triangle only if the projection $\mathbf{U}'$ on one plane is inside the projection of the triangle over the same plane. The projection over the plane orthogonal to the largest component of $\mathbf{n}$ guarantees that degenerate cases like a triangle nearly (or exactly) orthogonal to the plane do not occur.
As a preliminary step consider the problem of computing the area of a triangle. The most usual definition as one half the product of the lengths of the base times the altitude is not practical if the triangle is assigned by the coordinates of the three vertices. An alternated definition, however, can be obtained by the cross product of two vectors. Let $\mathbf{B} - \mathbf{A}$ and $\mathbf{C} - \mathbf{A}$ be the vectors, their cross product is a third vector orthogonal to the plane containing the original vectors with a magnitude equal to the area of the parallelogram formed by $\mathbf{B} - \mathbf{A}$ and $\mathbf{C} - \mathbf{A}$. Being $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ the vertices of the triangle its area then

simply reads

$$\mathcal{A}'_{ABC} = \frac{1}{2}|(\mathbf{B} - \mathbf{A}) \times (\mathbf{C} - \mathbf{A})|, \tag{A.3}$$

which can be explicitly computed from the coordinates of the vertices. Of course since in Equation (A.3) we take the absolute value of the cross product the area $\mathcal{A}'_{ABC}$ will be positive regardless of the relative position of $\mathbf{B} - \mathbf{A}$ and $\mathbf{C} - \mathbf{A}$. Considering the above expression with its own sign $\mathcal{A}_{ABC} = (\mathbf{B} - \mathbf{A}) \times (\mathbf{C} - \mathbf{A})/2$, however, evidences an interesting property of the area of a triangle which is positive or negative depending if the orientation of the path ABC is counterclockwise or clockwise. This property can be efficiently used for the determination of the relative position of a point with respect to a triangle. Consider in fact the configuration of Figure A.2a with the point $\mathbf{U}$ inside the triangle. The three triangles ABU, BCU and CAU all have positive areas, respectively, $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_3$. If the point $\mathbf{U}$ is outside the triangle one or two of these triangles will have negative areas (Figures A.2b-c). If the point $\mathbf{U}$ is on one of the edges of the triangle ABC one of the areas will be zero (Figure A.2d). If instead two areas are zero the $\mathbf{U}$ will be on one of the vertices (Figure A.2e). All the above arguments hold only if the path

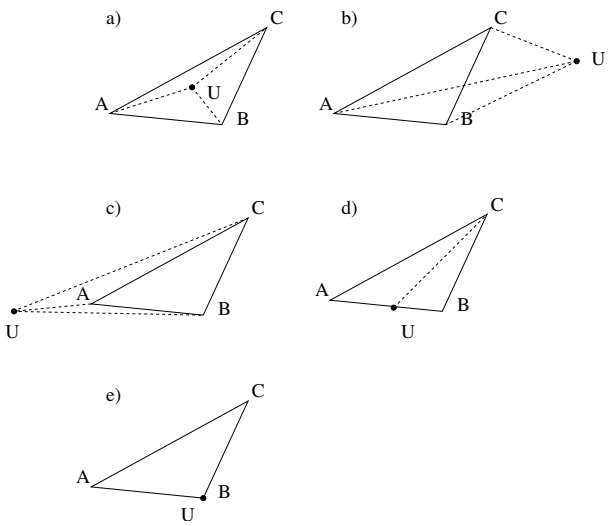


Figure A.2: Different configurations for the relative position of a point and a triangle.

ABC is oriented counterclockwise, it is however easy to show that if the sequence ABC is oriented clockwise the area of the triangle is negative and if $\mathbf{U}$ is internal triangles ABU, BCU and CAU will all have negative areas.

The criterion can then be easily generalized as follows: if the three sub-triangles all have areas with the same sign then $\mathbf{U}$ is inside the triangle, while if the sign of one of the areas

is different from the other two $\mathbf{U}$ is outside the triangle. If $\mathcal{A}_1$ is zero $\mathbf{U}$ is on the edge $\mathbf{AB}$ while if $\mathcal{A}_1$ and $\mathcal{A}_2$ are both zero $\mathbf{U} \equiv \mathbf{B}$. All the other possibilities can be deduced by induction.

Appendix B

Viscous coefficient matrices

$$\mathbf{R}_{xx} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{4}{3}(\mu+\mu_t) & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & (\mu+\mu_t) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & (\mu+\mu_t) & 0 & 0 & 0 \\ 0 & \frac{4}{3}(\mu+\mu_t)u & (\mu+\mu_t)v & (\mu+\mu_t)w & C_p\left(\frac{\mu}{Pr}+\frac{\mu_t}{Pr_t}\right) & (\mu+\sigma^*\mu_t) & 0 \\ 0 & 0 & 0 & 0 & 0 & (\mu+\sigma^*\mu_t) & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & (\mu+\sigma\mu_t) \end{bmatrix};$$

$$\mathbf{R}_{xy} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{2}{3}(\mu+\mu_t) & 0 & 0 & 0 & 0 \\ 0 & (\mu+\mu_t) & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & (\mu+\mu_t)v & -\frac{2}{3}(\mu+\mu_t)u & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix};$$

$$\mathbf{R}_{xz} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{2}{3}(\mu+\mu_t) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & (\mu+\mu_t) & 0 & 0 & 0 & 0 & 0 \\ 0 & (\mu+\mu_t)w & 0 & -\frac{2}{3}(\mu+\mu_t)u & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix};$$

$$\mathbf{R}_{yx} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & (\mu+\mu_t) & 0 & 0 & 0 & 0 \\ 0 & -\frac{2}{3}(\mu+\mu_t) & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{2}{3}(\mu+\mu_t)v & (\mu+\mu_t)u & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix};$$

$$\mathbf{R}_{yy} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & (\mu+\mu_t) & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{4}{3}(\mu+\mu_t) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & (\mu+\mu_t) & 0 & 0 & 0 \\ 0 & (\mu+\mu_t)u & \frac{4}{3}(\mu+\mu_t)v & (\mu+\mu_t)w & C_p\left(\frac{\mu}{Pr}+\frac{\mu_t}{Pr_t}\right) & (\mu+\sigma^*\mu_t) & 0 \\ 0 & 0 & 0 & 0 & 0 & (\mu+\sigma^*\mu_t) & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & (\mu+\sigma\mu_t) \end{bmatrix};$$

$$\mathbf{R}_{yz} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{2}{3}(\mu+\mu_t) & 0 & 0 & 0 \\ 0 & 0 & (\mu+\mu_t) & 0 & 0 & 0 & 0 \\ 0 & 0 & (\mu+\mu_t)w & -\frac{2}{3}(\mu+\mu_t)v & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix};$$

$$\mathbf{R}_{zx} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & (\mu+\mu_t) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{2}{3}(\mu+\mu_t) & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{2}{3}(\mu+\mu_t)w & 0 & (\mu+\mu_t)u & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix};$$

$$\mathbf{R}_{zy} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & (\mu+\mu_t) & 0 & 0 & 0 \\ 0 & 0 & -\frac{2}{3}(\mu+\mu_t) & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{2}{3}(\mu+\mu_t)w & (\mu+\mu_t)v & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix};$$

$$\mathbf{R}_{zz} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & (\mu+\mu_t) & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & (\mu+\mu_t) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{4}{3}(\mu+\mu_t) & 0 & 0 & 0 \\ 0 & (\mu+\mu_t)u & (\mu+\mu_t)v & \frac{4}{3}(\mu+\mu_t)w & C_p\left(\frac{\mu}{Pr}+\frac{\mu_t}{Pr_t}\right) & (\mu+\sigma^*\mu_t) & 0 \\ 0 & 0 & 0 & 0 & 0 & (\mu+\sigma^*\mu_t) & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & (\mu+\sigma\mu_t) \end{bmatrix};$$

Appendix C

Right and Left Eigenvectors

The inverse of the matrix $\mathbf{S}$ is given by:

$$\mathbf{S}^{-1} = b \begin{bmatrix} sinv_{11} & \frac{\tilde{\rho}_T u}{d'} & \frac{\tilde{\rho}_T v}{d'} & \frac{\tilde{\rho}_T w}{d'} & -\frac{\tilde{\rho}_T}{d'} & sinv_{16} & 0 \\ -\frac{u}{\rho} & \frac{1}{\rho} & 0 & 0 & 0 & 0 & 0 \\ -\frac{v}{\rho} & 0 & \frac{1}{\rho} & 0 & 0 & 0 & 0 \\ -\frac{w}{\rho} & 0 & 0 & \frac{1}{\rho} & 0 & 0 & 0 \\ sinv_{51} & -\frac{u\tilde{\rho}''_{p_t}}{d'} & -\frac{v\tilde{\rho}''_{p_t}}{d'} & -\frac{w\tilde{\rho}''_{p_t}}{d'} & \frac{\tilde{\rho}''_{p_t}}{d'} & sinv_{56} & 0 \\ -\frac{k}{\rho} & 0 & 0 & 0 & 0 & \frac{1}{\rho} & 0 \\ -\frac{\omega}{\rho} & 0 & 0 & 0 & 0 & 0 & \frac{1}{\rho} \end{bmatrix}, \tag{C.1}$$

where

$$\begin{aligned} h &= \tilde{H} - \frac{1}{2}\left(u^2 + v^2 + w^2\right) \\ d' &= \rho\tilde{\rho}''_{p_t}\tilde{h}_T + \tilde{\rho}_T\left(1 - \rho\tilde{h}_{p_t}\right) \\ b &= \frac{1}{1 + \frac{3\Delta\tau}{2\Delta t}}. \\ sinv_{11} &= \frac{\tilde{h}_T\left(\rho + \tilde{\rho}_k k\right) + \tilde{\rho}_T\left(h - \tilde{h}_k k - \frac{5}{3}k\right)}{d'} \\ sinv_{16} &= \frac{\tilde{\rho}_T\left(\frac{5}{3} + \tilde{h}_k\right) - \tilde{\rho}_k\tilde{h}_T}{d'} \\ sinv_{51} &= \frac{\left(1 + \frac{k}{\rho}\tilde{\rho}_k\right)\left(1 - \rho\tilde{h}_{p_t}\right) - \tilde{\rho}''_{p_t}\left(h - \frac{5}{3}k - \tilde{h}_k k\right)}{d'} \\ sinv_{56} &= -\frac{\frac{\tilde{\rho}_k}{\rho}\left(1 - \rho\tilde{h}_{p_t}\right) + \tilde{\rho}''_{p_t}\left(\frac{5}{3} + \tilde{h}_k\right)}{d'} \end{aligned} \tag{C.2}$$

The eigenvalues are:

$$\begin{aligned}
\lambda_1 = \lambda_2 = \lambda_3 &= \hat{U} b \\
\lambda_{4,5} &= \frac{1}{2} b \left[\hat{U} \left(1 + \epsilon^{'} \right) \pm \sqrt{\hat{U}^2 \left(1 - \epsilon^{'} \right)^2 + \frac{4 \rho \tilde{h}_T \left| \nabla l \right|^2}{d^{'}}} \right]
\end{aligned} \tag{C.3}$$

where

$$\begin{aligned}
\hat{U} &= u l_x + v l_y + w l_z \\
\nabla l &= (l_x, l_y, l_z) \\
\left| \nabla l \right|^2 &= l_x^2 + l_y^2 + l_z^2 \\
\epsilon^{'} &= \frac{\rho \tilde{\rho}_{p_t} \tilde{h}_T + \tilde{\rho}_T \left(1 - \rho \tilde{h}_{p_t} \right)}{\rho \tilde{\rho}_{p_t}^{''} \tilde{h}_T + \tilde{\rho}_T \left(1 - \rho \tilde{h}_{p_t} \right)}
\end{aligned} \tag{C.4}$$

and l represents generalized coordinates.

The corresponding right and left eigenvectors are given respectively by:

$$\mathbf{M} = \begin{bmatrix}
0 & 0 & 0 & \frac{\tilde{\lambda_5} - \epsilon^{'} \tilde{U} b}{\tilde{\lambda_5} - \tilde{\lambda_4}} & \frac{\tilde{\lambda_4} - \epsilon^{'} \tilde{U} b}{\tilde{\lambda_4} - \tilde{\lambda_5}} & 0 & 0 \\
0 & \frac{m_1}{\rho} & \frac{n_1}{\rho} & \frac{\tilde{l}_x b}{\rho \left(\tilde{\lambda_4} - \tilde{\lambda_5} \right)} & -\frac{\tilde{l}_x b}{\rho \left(\tilde{\lambda_4} - \tilde{\lambda_5} \right)} & 0 & 0 \\
0 & \frac{m_2}{\rho} & \frac{n_2}{\rho} & \frac{\tilde{l}_y b}{\rho \left(\tilde{\lambda_4} - \tilde{\lambda_5} \right)} & -\frac{\tilde{l}_y b}{\rho \left(\tilde{\lambda_4} - \tilde{\lambda_5} \right)} & 0 & 0 \\
0 & \frac{m_3}{\rho} & \frac{n_3}{\rho} & \frac{\tilde{l}_z b}{\rho \left(\tilde{\lambda_4} - \tilde{\lambda_5} \right)} & -\frac{\tilde{l}_z b}{\rho \left(\tilde{\lambda_4} - \tilde{\lambda_5} \right)} & 0 & 0 \\
-\frac{\left(1 - \rho \tilde{h}_{p_t} \right)}{\rho \tilde{h}_T} & 0 & 0 & A_4 & A_5 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{1}{\rho} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\rho}
\end{bmatrix}, \tag{C.5}$$

and

$$\mathbf{M}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & -\frac{\rho\tilde{h}_T}{\left(1-\rho\tilde{h}_{p_t}\right)} & 0 & 0 \\ 0 & \rho m_1 & \rho m_2 & \rho m_3 & 0 & 0 & 0 \\ 0 & \rho n_1 & \rho n_2 & \rho n_3 & 0 & 0 & 0 \\ 1 & B_4\tilde{l}_x & B_4\tilde{l}_y & B_4\tilde{l}_z & 0 & 0 & 0 \\ 1 & B_5\tilde{l}_x & B_5\tilde{l}_y & B_5\tilde{l}_z & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \rho & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \rho \end{bmatrix}, \tag{C.6}$$

where A_4 and A_5 are given by

$$A_4 = \frac{\left(1-\rho\tilde{h}_{p_t}\right)}{\rho\tilde{h}_T}\frac{\tilde{\lambda}_5 - \epsilon'\tilde{U}b}{\tilde{\lambda}_5 - \tilde{\lambda}_4} \qquad \text{and} \qquad A_5 = \frac{\left(1-\rho\tilde{h}_{p_t}\right)}{\rho\tilde{h}_T}\frac{\tilde{\lambda}_4 - \epsilon'\tilde{U}b}{\tilde{\lambda}_4 - \tilde{\lambda}_5}; \tag{C.7}$$

B_4 and B_5 are given by

$$B_4 = \rho\left(\frac{\tilde{\lambda}_4}{b} - \epsilon'\tilde{U}\right) \qquad \text{and} \qquad B_5 = \rho\left(\frac{\tilde{\lambda}_5}{b} - \epsilon'\tilde{U}\right); \tag{C.8}$$

$\nabla\tilde{l}$, m, n represent three mutually orthogonal unit vectors,

$$\begin{aligned} \left(\tilde{l}_x, \tilde{l}_y, \tilde{l}_z\right) &= \frac{\nabla l}{|\nabla l|}, \\ (m_1, m_2, m_3) &= \frac{(l_y - l_z, l_z - l_x, l_x - l_y)}{\sqrt{2\left(|\nabla l|^2 - l_x l_y - l_y l_z - l_x l_z\right)}}, \\ (n_1, n_2, n_3) &= \frac{\left(l_x\left(l_y + l_z\right) - l_y^2 - l_z^2, l_y\left(l_x + l_z\right) - l_x^2 - l_z^2, l_z\left(l_x + l_y\right) - l_x^2 - l_y^2\right)}{|\nabla l|\sqrt{2\left(|\nabla l|^2 - l_x l_y - l_y l_z - l_x l_z\right)}}, \end{aligned} \tag{C.9}$$

and, finally,

$$\tilde{U} = \tilde{l}_x u + \tilde{l}_y v + \tilde{l}_z w \qquad \text{and} \qquad \tilde{\lambda}_{4,5} = \frac{\lambda_{4,5}}{|\nabla l|}. \tag{C.10}$$

Bibliography

[1] G. Kalitzin, G. Medic, G. Iaccarino, and P. Durbin. Near-wall behavior of RANS turbulence models and implications for wall functions. *Journal of Computational Physics*, 204:265–291, 2005.

[2] G. K. Batchelor. *An Introduction to Fluid Mechanics.* Cambridge Univ. Press, 1967.

[3] 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*, 225:2098–2117, 2007.

[4] R. Clift, J. R. Grace, and M. E. Weber. *Bubbles, Drops and Particles.* Academic Press, 1978.

[5] M. Lee-Rausch and J. T. Batina. Calculation of AGARD wing 445.6 flutter using Navier–Stokes aerodynamics. Technical report, 1993.

[6] E. Guilmineau and P. Queutey. A numerical simulation of vortex shedding from an oscillating circular cylinder. *Journal of Fluid and Structures*, 16:773–794, 2002.

[7] M. D. de Tullio, G. Pascazio, and M. Napolitano. Arbitrarily shaped particles in shear flow. *Proceedings Seventh International Conference on Computational Fluid Dynamics (ICCFD7), Big Island, HI, US*, 2012.

[8] D. Barthès-Biesel. Motion of a spherical microcapsule freely suspended in a linear shear flow. *Journal of fluid mechanics*, 100:831–853, 1980.

[9] C. S. Peskin. *Flow Patterns Around Heart Valves: A Digital Computer Method for Solving the Equations of Motion.* PhD thesis, Physiology, Albert Einstein College of Medicine. University Microfilms 72-30, 378, 1972.

[10] J. Mohd-Yusof. Combined immersed boundaries/b-splines methods for simulations of flows in complex geometries. Technical report, NASA Ames / Stanford University, 1997. CTR Annual Research Briefs.

[11] E. A. Fadlun, R. Verzicco, P. Orlandi, and J. Mohd-Yosuf. Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations. *Jorunal of Computational Physics*, 161:35–60, 2000.

[12] G. Iaccarino and R. Verzicco. Immersed boundary technique for turbulent flow simulations. *Appl. Mech. Rev.*, 56:331–347, 2003.

[13] R. Mittal and G. Iaccarino. Immersed boundary methods. *Annual Review of Fluid Mechanics*, 37:239–261, 2005.

[14] P. De Palma, M. D. de Tullio, G. Pascazio, and M. Napolitano. An immersed boundary method for compressible viscous flows. *Computers & Fluids*, 35:693–702, 2006.

[15] J. A. Vieceli. A method for including arbitrary external boundaries in the mac incompressible fluid computing technique. *Journal of Computational Physics*, 4:543–551, 1969.

[16] J. E. Welch, F. H. Harlow, J. P. Shannon, and B. J. Daly. A computing technique for solving viscous incompressible transient fluid flow problems involving free-surfaces. *Report LA–3425, Los Alamos Scientific Laboratory*, 1966.

[17] F. H. Harlow and J. E. Welch. Numerical calculation of time-dependent viscous incompressible flows of fluid with free surface. *Phys. Fluids*, 8:2182–2189, 1965.

[18] J. A. Vieceli. A computing method for incompressible flows bounded by moving walls. *Journal of Computational Physics*, 8:119–143, 1971.

[19] C. S. Peskin. Numerical analysis of blood flow in the heart. *Journal of Computational Physics*, 25:220–252, 1977.

[20] C. S. Peskin. The fluid dynamics of heart valves: Experimental, theoretical and computational methods. *Annual Review of Fluid Mechanics*, 14:235–259, 1982.

[21] C. S. Peskin and D. M. McQueen. A three-dimensional computational method for blood flow in the heart I. immersed elastic fibers in a viscous incompressible fluid. *Journal of Computational Physics*, 81:372–405, 1989.

[22] D. M. McQueen and C. S. Peskin. A three-dimensional computational method for blood flow in the heart: (II) contractile fibers. *Journal of Computational Physics*, 82:289–297, 1989.

[23] C. Basdevant and R. Sadourny. Numerical solution of incompressible flow: the mask method. *Laboratoire di Meteorologie Dynamique, Ecole Normale Superieure, Paris (unpublished)*, 1984.

[24] M. Briscolini and P. Santangelo. Development of the mask method for incompressible unsteady flows. *Journal of Computational Physics*, 84:57–75, 1989.

[25] D. Goldstein, R. Handler, and L. Sirovich. Modeling no-slip flow boundary with an external force field. *Journal of Computational Physics*, 105:354–366, 1993.

[26] E. M. Saiki and S. Biringen. Numerical simulation of a cylinder in uniform flow: Application of a virtual boundary method. *Journal of Computational Physics*, 123:450–465, 1996.

[27] E. Balaras. Modeling complex boundaries using an external force field on fixed Cartesian grids in large-eddy simulations. *Computers & Fluids.*, 33:375–404, 2004.

[28] S. Kang. *An improved immersed boundary method for computation of turbulent flow with heat transfer.* PhD thesis, Stanford University, 2008.

[29] J. Kim, D. Kim, and H. Choi. An immersed-boundary finite volume method for simulations of flow in complex geometries. *Journal of Computational Physics*, 171:132–150, 2001.

[30] S. Majumdar, G. Iaccarino, and P. Durbin. RANS solvers with adaptive structured boundary non-conforming grid. *Center for Turbulence Research, Annual Research Briefs, Stanford University*, pages 353–366, 2001.

[31] Y. H. Tseng and J. H. Ferziger. A ghost-cell immersed boundary method for flow in complex geometry. *Journal of Computational Physics*, 153:535–574, 1999.

[32] A. Dadone and B. Grossman. Ghost-cell method for inviscid three-dimensional flows on Cartesian grids. *43rd AIAA Aerospace Sciences Meeting & Exhibit, Reno, Nevada*, 10-13 Jan 2005.

[33] R. L. Taylor, P. Onate, and P. Ubach. Finite element analysis of membrane structures. *In book: Textile Composites and Inflatable Structures*, pages 47–68, 2005.

[34] P. E. Hammer, M. S. Sacks, P. J. del Nido, and R. D. Howe. Mass-spring model for simulation of heart valve tissue mechanical behavior. *Annals of Biomedical Engineering*, 39:1–12, 2011.

[35] D. A. Fedosov. *Multiscale Modeling of Blood Flow and Soft Matter.* PhD thesis, Division of Applied Mathematics at Brown University, USA., 2010.

[36] G. M. Laskowski, S. P. Kearney, G. Evans, and R. Greif. Mixed convection heat tranfer to and from a horizontal cylinder in cross-flow with heating from below. *International Journal of Heat and Fluid Flow*, 28:454–468, 2007.

[37] L. Hylton, M. Mihelc, E. Turner, D. Nealy, and R. York. Analytical and experimental evaluation of the heat transfer distribution over the surfaces of turbine vanes. Technical report, National Aeronautics and Space Administration, NASA Lewis Research Center, 1983.

[38] L. Andrei, A. Andreini, B. Facchini, and L. Winchler. A decoupled CHT procedure: application and validation on a gas turbine vane with different cooling configurations. *Energy Procedia*, 45:1087–1096, 2014.

[39] T. Yoshiara, D. Sasaki, and K. Nakahashi. Conjugate heat transfer simulation of cooled turbine blades using unstructured-mesh cfd solver. In *49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition in Orlando, Florida*, page 498, 2011.

[40] J. Luo and E. H. Razinsky. Conjugate heat transfer analysis of a cooled turbine vane using the V2F turbulence model. *Journal of turbomachinery*, 129:773–781, 2007.

[41] M. J. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Computational Physics*, 53:482–512, 1984.

[42] D. De Zeeuw and K.M. Powell. An adaptively-refined Cartesian mesh solver for the Euler equations. *AIAA Paper 91-1542*, 1991.

[43] J. Quirk. An alternative to unstructured grids for computing gas dynamic flows around arbitrarily complex two dimensional bodies. *ICASE Report 92-7*, 1992.

[44] J. E. Melton, F. Y. Enomoto, and M. J. Berger. Automatic Cartesian grid generation for Euler flows. *AIAA Paper 93-3386-CP*, 1993.

[45] S. L. Jr Karman. Splitflow: A 3D unstructured Cartesian/prismatic grid CFD code for complex geometries. *AIAA 95-0343*, 1995.

[46] T. J. Welterlen and S. L. Jr. Karman. Rapid assessment of F-16 store trajectories using unstructured CFD. *AIAA 95-0354*, 1995.

[47] J. E. Melton, M. J. Berger, M. J. Aftosmis, and M. D. Wong. 3D applications of a Cartesian grid Euler method. *AIAA Paper 95-0853*, 1995.

[48] J.E. Melton. *Automated Three-Dimensional Cartesian Grid Generation and Euler Flow Solutions for Arbitrary Geometries.* PhD thesis, Univ. CA., Davis CA, 1996.

[49] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction.* SpringerVerlag, 1985.

[50] D. Voorhies. *Graphics Gems II: TriangleCube Intersections.* Academic Press, Inc., 1992.

[51] J. O'Rourke. *Computational Geometry in C.* Cambridge Univ. Press, NY, 1993.

[52] J. Foley, A. van Dam, S. Feiner, and J. Hughes. *Computer Graphics: Principles and Practice.* AddisonWesley, Reading, MA, 1995.

[53] M. J. Aftosmis, M. J. Berger, and J.E Melton. *Handbook of Mesh Generation.* Contributed Chapter. CRC Press., 1998.

[54] D. C. Wilcox. *Turbulence Modelling for CFD.* DCW Industries, Inc., second edition, 1998.

[55] M. Vanella and E. Balaras. A moving-least-squares reconstruction for embedded-boundary formulations. *Journal of Computational Physics*, 228:6617–6628, 2009.

[56] G. Iaccarino and S. Moreau. Natural and forced conjugate heat transfer in complex geometries on cartesian adapted grids. *Transactions of ASME*, 128:838–846, 2006.

[57] D. A. Schwer. *Numerical study of unsteadiness in non-reacting and reacting mixing layers.* PhD thesis, Department of Mechanical Engineering, The Pennsylvania State University, 1999.

[58] S. Chakravarthy and S. Osher. Numerical experiments with the Osher upwind scheme for the Euler equations. *AIAA Journal*, 24:1241–1248, 1986.

[59] T. H. Pulliam. Time accuracy and the use of implicit methods. *AIAA Paper 93-3360 CP*, 1993.

[60] C. Merkle. Preconditioning methods for viscous flow calculations. *Computational Fluid Dynamics*, pages 419–436, 1995.

[61] D. Choi and C. Merkle. Application of time–iterative schemes to incompressible flows. *AIAA Journal*, 23:1518–1524, 1985.

[62] E. Turkel. Preconditioning method for solving the incompressible and low speed compressible equations. *Journal of Computational Physics*, 72:277–298, 1987.

[63] B. van Leer, W. Lee, and P. Roe. Characteristic time-stepping or local preconditioning of the Euler equations. *AIAA paper 91-1552 CP*, 1992.

[64] D. Choi and C. L. Merkle. The application of preconditioning to viscous flows. *Journal of Computational Physics*, 105:207–223, 1993.

[65] S. Venkateswaran and C. L. Merkle. Dual time stepping and preconditioning for unsteady computations. *AIAA paper 95-0078*, 1995.

[66] S. Venkateswaran, J. M. Weiss, Merkle C. L., and Y. H. Choi. Propulsion–related flowfields using the preconditioned Navier–Stokes equations. *AIAA paper 92-3437*, 1992.

[67] P. E. O. Buelow, D. A. Schwer, J. Z. Feng, C. L. Merkle, and D. Choi. A preconditioned dual-time, diagonalized ADI scheme for unsteady computations. *AIAA Paper 97-2101 CP*, 1997.

[68] P. E. O. Buelow. *Convergence Enhancement of Euler and Navier–Stokes algorithms.* PhD thesis, Department of Mechanical Engineering, The Pennsylvania State University, 1995.

[69] T. H. Pulliam and D. S. Chaussee. A diagonal form of an implicit approximate factorization algorithm. *Journal of Computational Physics*, 39:347–363, 1981.

[70] S. De Rango and D. Zingg. Improvements to a dual-time stepping method for computing unsteady flows. *AIAA Journal*, 35:1548–1551, 1997.

[71] van der Vorst. Bi-CGStab: a fast and smoothly converging variant of the bi-CG for the solution of non-symmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 13:361, 1992.

[72] F. R. Menter. Two-equation eddy-viscosity turbulence models for engineering applications. *AIAA Journal*, 32-8:1598–1605, 1994.

[73] M. B. Giles. Stability analysis of numerical interface conditions in fluid-structure thermal analysis. *International Journal of Numerical methods in fluid*, 25:421–436, 1997.

[74] F. Duchaine, A. Corpron, L. Pons, V. Moureau, F. Nicoud, and T. Poinsot. Development and assessment of a coupled strategy for conjugate heat transfer with large eddy simulation: application to a cooled turbine blade. *International Journal of Heat and Fluid Flow*, 30:1129–1141, 2009.

[75] M. D. de Tullio. *Development of an Immersed Boundary method for the solution of the preconditioned Navier-Stokes equations.* PhD thesis, Politecnico di Bari, 2006.

[76] M. O. Bristeau, R. Glowinski, J. Periaux, and A. Viviand. *Numerical simulation of compressible Navier-Stokes flow*, volume 18. Vieweg, 1987.

[77] V. A. Bashkin, A. V. Vaganov, I. V. Egorov, D. V. Ivanov, and G. A. Ignatova. Comparison of calculated and experimental data on supersonic flow past a circular cylinder. *Fluid Dynamics*, 37:473–483, 2002.

[78] C. H. Sieverding. Experimental data on two transonic turbine blade sections and comparisons with various theoretical methods. *VKI Report No. LS59, von Karman Institute, Belgium*, 1973.

[79] R. Kiock, F. Lethaus, N. C. Baines, and C. H. Sieverding. The transonic flow through a plane turbine cascade as measured in four European winnd tunnels. *ASME J. Eng. Gas Turbines Power*, 108:277–285, 1986.

[80] F. Liu, J. Cai, and Y. Zhu. Calculation of wing flutter by a coupled fluid-structure method. *Journal of aircraft*, 38:334–342, 2001.

[81] G. C. Rufolo, M. Marini, P. Roncioni, and S. Borrelli. In-flight aerodynamic experiment for the unmanned space vehicle ftb-1. *First CEAS European Air and Space Conference, Berlin, Germany, Septemper 10-13*, 2007.

[82] M. Uhlmann. An immersed boundary method with direct forcing for the simulation of particulate flows. *Journal of Computational Physics*, 209:448–476, 2005.