The overset grid method, applied to the solution of the incompressible Navier-Stokes equations in two and three spatial dimensions.

A thesis submitted to the University of Manchester for the degree of Doctor of Philosophy in the Faculty of Engineering and Physical Sciences.

2011

Alex Skillen

School of Mechanical, Aerospace and Civil Engineering

Contents

Final word count: $\sim$ 70,000

List of Figures

Abstract

- The University of Manchester
- Alex Skillen
- Doctor of Philosophy
- The overset grid method, applied to the solution of the incompressible Navier-Stokes equations in two and three spatial dimensions.
- 2011

The generation of structured grids around complex geometries is generally a difficult task. This task is typically a major bottleneck in the overall solution procedure; however, the overset grid method can be used to relieve much of this burden. An overset grid consists of a set of simple component grids, which can overlap arbitrarily (provided there is sufficient overlap to interpolate from). The union of all simple grids should then delineate the global domain. This allows complex domains to meshed using a series of simple meshes. Interpolation boundary conditions are enforced at internal boundaries to ensure a continuous solution. Standard tri-linear interpolation is typically used for this purpose, although there are alternative methods that attempt to enforce global conservation.

A new CFD code has been developed that incorporates the overset grid method in three spatial dimensions. This code uses the steady state, finite volume discretisation method. SIMPLE pressure velocity coupling has been used on a colocated grid with Rhie-Chow interpolation for face velocities. Different interpolation methods have been compared for the information transfer at internal boundaries from one grid to the next. It has been shown that for a variety of test cases, continuous and accurate solutions are obtained from one grid to another, which are comparable to those of the single-block or block-structured solutions, or to experimental data (where available). A new hole cutting algorithm and bulk correction outlet condition are presented. Improvements to existing digital tree data structures are also described.

Lid driven cavity flow, the flow around rotating cylinders, and flow impingement onto a concave surface are considered in order to demonstrate the method. The flow over a backward facing step, over a multi-element airfoil, through a bifurcating artery and over a wing-body junction are then considered (with experimental comparison). This demonstrates the range of applicability of the method. In all cases, the overset method offers significant advantages over block-structured solutions that are available in the literature. It is shown that greater numerical efficiency is generally achievable via the use of an overset simulation: Since the gridding is flexible, high aspect ratio cells need not propogate into the domain (as is often the case for a block-structured arrangement). Also, much of the domain away from localised regions of geometrical complexity can be meshed with efficient Cartesian grids.

Declaration

No portion of the work referred to in this report has been submitted in support of an application for another degree or qualification of this or any other university or other institution of learning.

Copyright Statement

Nomenclature

Symbol	Usage
a	coefficients of $P^{'}$ in the pressure correction equation
A	Coefficients of u or v in the discretised momentum equations
C	Mass flux through a cell's face $\left(kg \cdot s^{-1}\right)$
g_1, g_2	Coefficients used in the QUICK scheme
i	Grid cell index in the ξ direction.
j	Grid cell index in the η direction
J	Jacobian
k	Grid number
k	Specific turbulent kinetic energy $= \frac{1}{2}\overline{u_i'u_i'}$
l	Iteration number
N_i	Number of nodes in the i direction
N_j	Number of nodes in the j direction
P	Pressure $\left(N \cdot m^{-2}\right)$
Re_t	$\equiv \frac{\rho k^2}{\mu\epsilon}$
$\bar{u}_i$	Component of mean velocity in i^{th} direction $\left(m \cdot s^{-1}\right)$
U^i	contravariant velocity components
x	A component of the global coordinate system (Cartesian reference frame). Perpendicular to both the y and z directions.

y	A component of the global coordinate system (Cartesian reference frame). Perpendicular to both the x and z directions.
z	A component of the global coordinate system (Cartesian reference frame). Perpendicular to both the x and y directions.

Greek Symbols

α	Under-relaxation factor
Γ	Diffusivity or boundary of a domain
Δx	Grid spacing in the x direction
Δy	Grid spacing in the y direction
$\Delta \eta$	Grid spacing in the η direction
$\Delta \xi$	Grid spacing in the ξ direction
ϵ	Turbulent dissipation rate
ζ	A component of the local coordinate system. Perpendicular to the ξ and η directions.
η	A component of the local coordinate system. Perpendicular to the ξ and ζ directions.
λ	Linear interpolation factor
μ	Fluid's dynamic viscosity $\left(N \cdot s \cdot m^{-2}\right)$
μ_t	Turbulent viscosity $\left(N \cdot s \cdot m^{-2}\right)$
ξ	A component of the local coordinate system. Perpendicular to the η and ζ directions.
ρ	Fluid's density $\left(kg \cdot m^{-3}\right)$
σ_k, σ_ϵ	Turbulent Prandtl numbers for k and ϵ transport equations.
ϕ	Conserved scalar quantity.
φ	An interpolation coefficient
χ	An interpolation coefficient

ψ	An interpolation coefficient
Ω	The overall domain (Ω_A, Ω_B, etc. are subdomains)

Subscripts & Superscripts

b	The bottom face of cell P
B	The node to the bottom of node P, and adjacent to P.
e	The east face of cell P
E	The node to the east of node P, and adjacent to P.
EE	The node to the east of node E, and adjacent to E.
n	The north face of cell P
N	The node to the north of node P, and adjacent to P.
NN	The node to the north of node N, and adjacent to N.
P	The cell that is currently being updated.
s	The south face of cell P
S	The node to the south of node P, and adjacent to P.
SS	The node to the south of node S, and adjacent to S.
V	Cell's volume $\left(m^3\right)$
w	The west face of cell P
W	The node to the west of node P, and adjacent to P.
WW	The node to the west of node W, and adjacent to W.

Acronyms

ADI	Alternating Direction Implicit
CDS	Central Difference Scheme
CFD	Computational Fluid Dynamics
CV	Control Volume
LES	Large Eddy Simulation
MFBI	Mass-Flux Based Interpolation

QUICK	Quadratic Upwind Interpolation for Convective Kinematics
RMS	Root Mean Squared
STREAM	Simulation of Turbulent Reynolds-averaged Equations for All Mach numbers, (an in-house CFD code).
TDMA	Tri-Diagonal Matrix Algorithm
UMIST	Upstream Monotonic Interpolation for Scalar Transport
UDS	Upwind Difference Scheme
ZIG	Zonal Interface Generation

Acknowledgements

- Thank you Tim and Hector, for your time, help and support with this project.
- Thank you Rob, for your constructive feedback on earlier progress reports.
- Thank you Adam, for being the perfect inspiration.
- Thank you Vicky, for your love and support (and for doing the dishes recently!)
- Thank you Mum and Dad, for everything else!

A most sincere thanks to all.

Chapter 1

Introduction

The majority of flows that are of engineering interest involve complex geometrical domains. Examples of such complex domains range from the flow over an entire aircraft, to internal biomedical flows such as the flow induced by a beating heart. When simulating such problems within a CFD framework, it is first necessary to generate a suitable grid over which the discretised governing equations can be solved. However, the generation of grids around complex geometries poses many challenges. To overcome these challenges, the overset grid method can be used.

In the overset grid method, it is noted that it is not a necessary condition to have grid-block interfaces aligned with one another (in contrast to a block-structured grid arrangement). Figure 1.1 illustrates an overset grid that has been generated with a Cartesian background mesh and a body-fitted curvilinear mesh around the obstacle area. An equivalent block-structured arrangement is also illustrated. It is evident from Figure 1.1 that the overset meshing technique has the potential to be far simpler at the pre-processing stage. This is even more apparent in three spatial dimensions, or for more complex or time-varying geometries.

Structured grids in general (including the overset method) also have many advantages over unstructured formulations. An unstructured mesh is characterised by irregular connectivity, with cell shapes that may vary, dependent on the solver. While the latter does offer the ability to mesh very complex domains with relative ease, this comes at a cost. Firstly, the hexahedral cells that are typically used in a structured formulation are particularly well suited for resolution of near wall regions. High aspect ratio cells can be employed which are aligned with the flow direction for a wall bounded flow. In this case, a perturbation to the main streamwise velocity component will induce only a small mass imbalance in the cell (since the face area normal to the velocity vector is small). On the other hand, other cell shapes that are of high aspect ratio may suffer from stability issues since relatively large face areas (present due to the high aspect ratio) will generally have a large projection normal to the streamwise direction. Any perturbation to the main streamwise flow in this case will cause a large mass imbalance in the cell which may produce instabilities when attempting to set the cell's pressure. In order to overcome this, one typically requires the near-wall cells of a mesh that employs triangular cells to be close to equilateral triangles (a great burden on the mesh size). Alternatively, a hybrid mesh could be used which uses hexahedral cells close to a wall, and triangular cells away from the wall. However, in this case the complexity of the underlying flow solver would be increased since more than one cell shape would need to be accounted for. In addition to the aforementioned disadvantage, unstructured grids are also typically very expensive to use. Cell connectivity information needs to be computed and stored. Expensive gradient reconstruction procedures are also generally required. There are also issues in implementing higher order convection schemes. Finally, the generation of a high quality unstructured mesh is not as easy as one may expect. While it is very easy to mesh an arbitrary domain with unstructured cells, the control of the cell spacing can be challenging. In the overset method however, it is straightforward to cluster the grid close to solid surfaces without distorting the global mesh density; a feature that is highly desirable, particurly ar high Reynolds number. The overset method does not suffer from these disadvantages

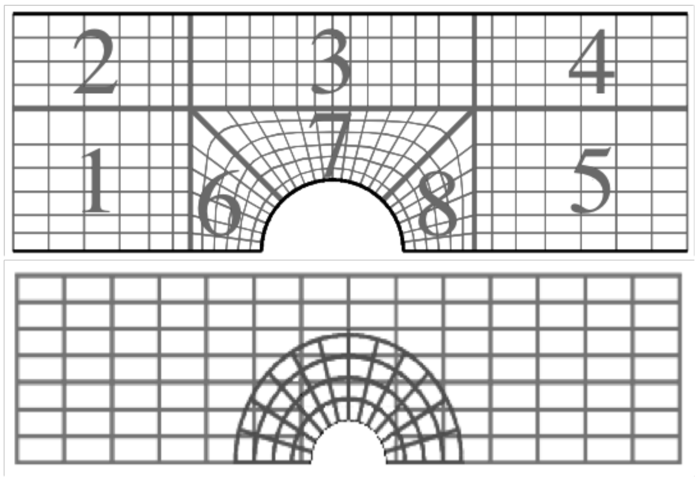


Figure 1.1: A duct and obstacle problem meshed with block-structured grids (above) and overlapping grids (below).

The overset method's solution procedure commences by solving the governing equations over one mesh with boundary conditions at internal boundaries obtained by interpolation from a suitable donor mesh. This is then repeated in an iterative manner on all other meshes until convergence is achieved. The main features of an overset CFD code that are unique to the overset formulation are: the interpolation algorithm; and an algorithm that constructs the overall grid from the set of component grids. The former algorithm interpolates information from a donor grid, providing boundary condition data at internal boundaries for the grid that is being solved on. The latter algorithm tags each node on a component grid with an integer code that dictates what type of node is being dealt with. Each node on a component grid must be one of the following;

- *A standard discretisation node.* If the given node can be discretised to the required order using nodes on its own grid only, and is not an unused node (see later), then it is a standard discretisation node.

- *An interpolation node.* If the given node can be interpolated from discretisation nodes on another grid, and is not a discretisation node or unused node itself, then the node is an interpolation node.

- *An unused node.* If the node lies within an overlap region where there is more than one mesh covering the same physical space, and it is not required by interpolation nodes from other grids, or by discretisation nodes on the same grid, then the node may be tagged as an unused node. The node is also tagged unused if it lies outside the solution domain

It is possible to generate highly efficient algorithms that automatically tag all nodes on each component mesh accordingly. Of key significance is the fact that if part of the mesh is outside of the domain, the unused mesh nodes are still present, but are tagged such that they are not solved on. If such unused nodes then come into the domain at a later time (as may well be the case for sliding meshes), the grid simply gets re-tagged (using some efficient algorithm), rather than regenerating the whole component mesh from scratch, which would be costly. This makes the overset method particularly attractive for sliding mesh simulations.

While there are clearly many benefits of using overset grids in a CFD code, the method is unfortunately not without challenges. Arguably the main issue with the overset method is enforcing conservation at grid interfaces. CFD codes are based on conservation laws, hence global conservation is of critical importance. However, standard linear interpolation for inter-grid data transfer is generally non-conservative. This lack of conservation can often cause stability and/or accuracy problems. Alternatives to linear interpolation have been developed for the overset method, that attempt to preserve conservation from one grid to the next. However, fully conservative interface treatments either involve costly regeneration of grids at each time-step, or are very difficult (if not impossible) to implement in 3D. Simpler semi-conservative methods may be used as an alternative (see Ref. [1]). Through the use of such methods, the overset grid method offers a powerful tool for the simulation of flows involving complex geometries.

1.1 Numerical Techniques

A new computational code has been developed in this project, from the ground up. This code is based on the steady-state, incompressible RANS equations, closed with a two-equation $k-\epsilon$ model (the low-Reynolds number $k-\epsilon$ formulation of Launder and Sharma has been used for this purpose [2]). A cell-centred, finite volume discretisation is used with SIMPLE pressure-velocity coupling over a colocated grid arrangement. Intra-grid interpolations are carried out via a choice of a first-order upwind, the second-order central-difference, or the third-order QUICK schemes. The RANS governing equations are employed, as will be discussed in Chapter 3. Implicit under-relaxation is employed in order to increase the diagonal dominance of the equations, which are solved iteratively using the Alternating Direction Implicit (ADI) technique, via Thomas' algorithm (ref. [3]). Internal boundary conditions are dealt with via an inter-grid interpolation. Linear interpolation, or the semi-conservative method of Tang *et. al.*, [1], have been coded for this purpose.

1.2 Objectives of this Research

As outlined earlier, conservation at grid interfaces is one area of the overset method that is potentially a cause for concern. It is therefore an important objective of this research to check if this concern is justified, or if the non-conservative interface treatment of linear interpolation is sufficient. Semi-conservative interface treatment will also be investigated.

The various overset algorithms available in the literature shall be evaluated and incorporated, improved, or replaced into the present code so as to provide a complete overset formulation.

Another objective of the present research is to simulate practical engineering problems involving complex geometries with overset meshes. This not only demonstrates the ability of the method in dealing with such geometrical complexity, but also provides some understanding into the complex flow physics such cases present.

1.3 Layout of this Thesis

In Chapter 2, a review of the current state of knowledge with respect to the overset method shall be presented. Chapters 3 and 4 consider, respectively, the implementation details of a single-block structured CFD code and the modelling of turbulence on this block. Chapter 5 then outlines the extension of this single block formulation to the overset formulation developed in the present work. Chapter 6 demonstrates the overset method applied to fairly simple test cases, which have been highlighted in order to demonstrate specific features of the method. In Chapters 7 to 10 complex geometrical cases are investigated, for which it is shown that the overset method gives solutions that are typically in good agreement with experimental data, or where any discrepancies can be attributed to shortcomings of the turbulence model. Test cases include the flow through a human artery bifurcation, the flow over a multi-element airfoil and the flow around a wing-body junction to name a few. Finally, in Chapter 11, some future work is suggested, and some concluding remarks are made.

Chapter 2

Literature Review

The overset grid method was pioneered by Benek, Steger and Dougherty [4] in 1983, who were also the first to coin the phrase 'Chimera grid'. This name was given in reference to the monster of Greek mythology, whose form is made up from parts of multiple animals. Since the original conception, significant improvements have been made to the Chimera methodology, specifically in the areas of automated grid tagging, and in conservative interface treatment. Different researchers have favoured different approaches, often varying radically in their implementation. This section shall first outline selected non-conservative interface treatments, including the original method of Benek *et al.*. We then move on to some of the main schools of thought that have branched off from the original Chimera grid method, that attempt to enforce conservation, highlighting both their advantages and disadvantages. Finally, an alternative to the overset method that is equally well equipped in dealing with complex geometries is investigated (namely the immersed boundary method).

2.1 Non-Conservative Interface Treatment

The original Chimera method used *non-conservative* linear interpolation of all variables to provide boundary conditions on an overset grid. Benek *et. al.* [4, 5] demonstrated the method by solving the discretised Euler equations for flow over an airfoil with a flap. Figure 2.1 shows the mesh used for this problem, while Figure 2.2 shows a close up of the flap region.

Unfortunately, the results of Benek *et. al.* were not particularly satisfactory. Figure 2.3 shows the pressure coefficient (C_P) variation along both the airfoil and flap's surface, with an angle of

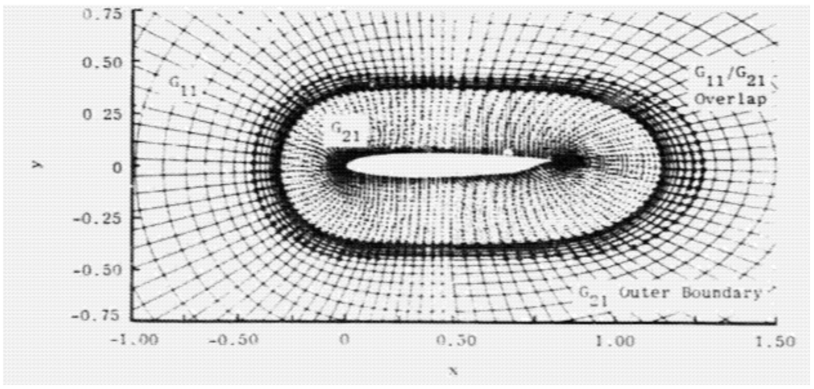


Figure 2.1: The grid used by Benek *et. al.* to develop the Chimera grid scheme. Image from [5].

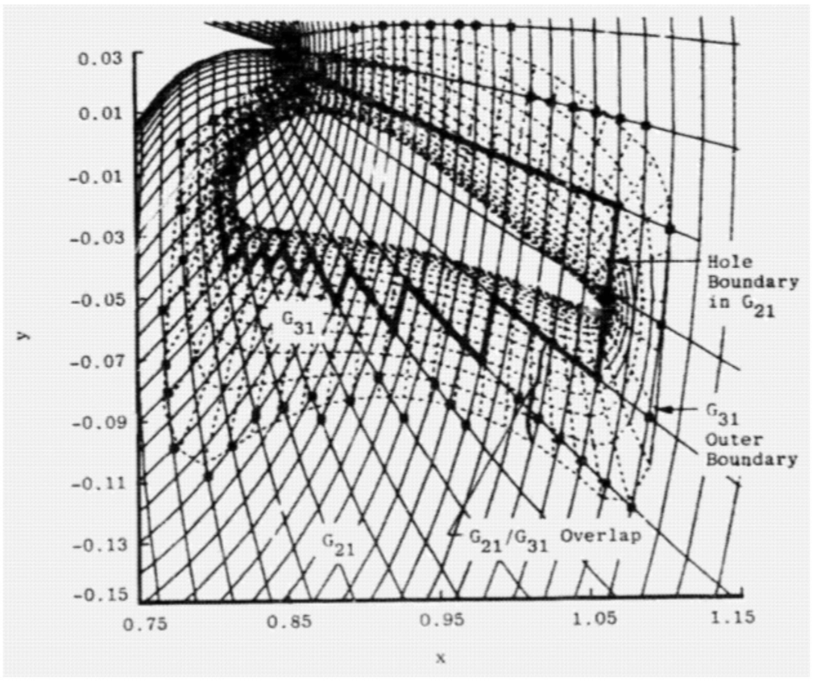


Figure 2.2: A closeup of the flap region. Image from [5].

attack equal to 3°, a flap deflection equal to 10°, and a free stream Mach number equal to 0.6.

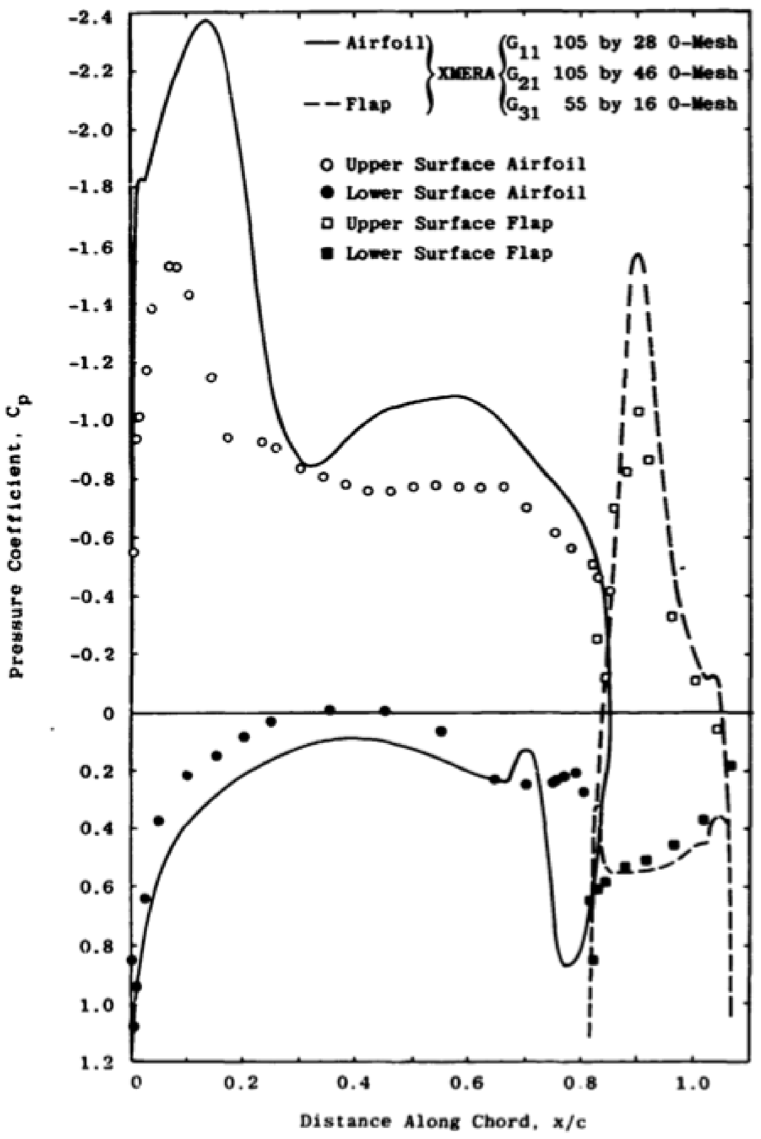


Figure 2.3: Pressure coefficient variation over an airfoil with flap. $\alpha = 3°$, $\beta = 10°$ **and** $M_\infty = 0.6$. **Image from [5].**

In Figure 2.3, it can be seen that, by comparing the computational results (the solid and dashed lines) with the experimental results (the symbols '•' and '○'), there is a significant overprediction of pressure coefficient on the suction surface of both the airfoil and the flap. However, it is not clear as to what extent the discrepancy is due to the Chimera grid's non-conservative interpolation scheme. It is expected that the majority of the discrepancy is down to the CFD code's use of the *inviscid* Euler equations as the governing equations, while the experimental results used air as the working fluid which clearly has a non-zero viscosity.

Henshaw [6] developed a fourth order method for the solution to the Navier-Stokes equations on overlapping grids. Henshaw used an interpolation that was based on Lagrange polynomials. The method was applied to the 2D flow over a cylinder, and the 3D flow over a sphere, which both gave continuous solutions from one grid to the next. Figure 2.4 illustrates the grids used in the flow over the cylinder, and also plots contours of the horizontal velocity.

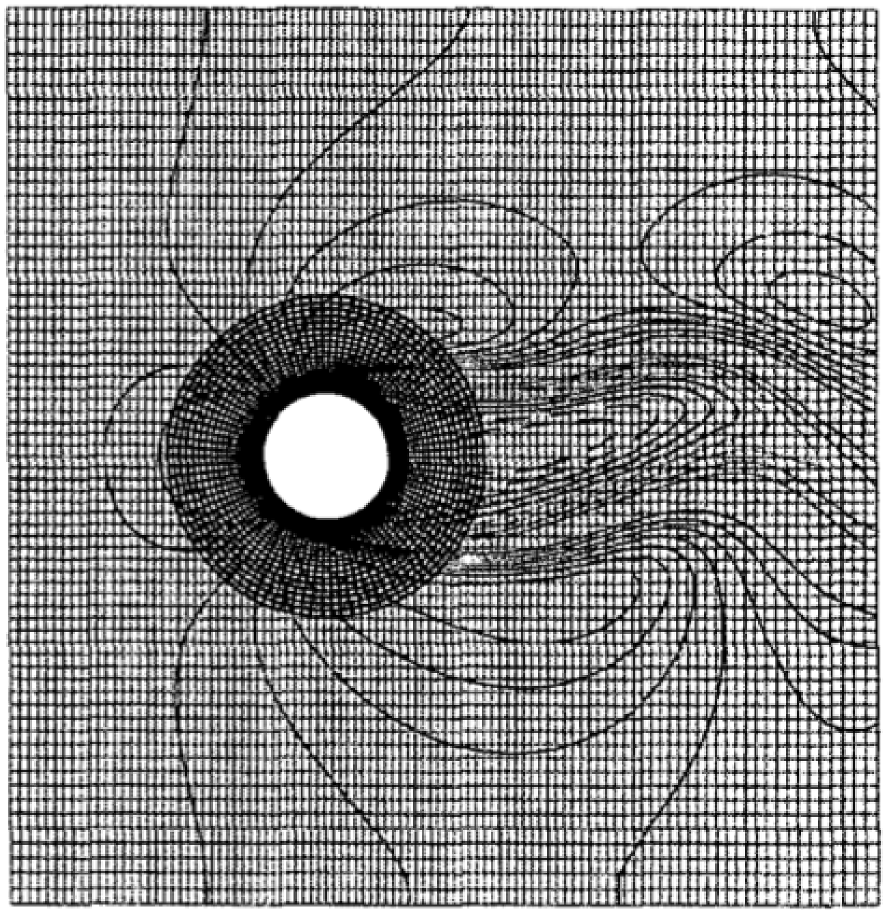


Figure 2.4: 2D flow over a cylinder simulated with overset grids and Lagrangian polynomial interpolation. Image from [6].

Tang *et. al.* [1] show that the residual error of the pressure and velocities when using linear interpolation fails to fall to as low a level as that of block-structured grids for a general overset problem. Furthermore, the convergence rate when using overset grids with linear interpolation is inferior (relative to an alternative interpolation method that Tang *et. al.* have developed, discussed later). It has also been noted that the pressure 'checkerboarding' phenomenon of colocated grids is more pronounced for linear interpolation, relative to that in single block or block-structured grids (when pressure-smoothing, such as Rhie-Chow interpolation, is not used). The reason for this has been identified as originating from the over-prescription of boundary conditions [1]. This in itself would not be a problem if the truncation error were zero everywhere, for then the boundary conditions for all the variables would be consistent with one another, and equal to the exact solution to the continuous governing equations (excluding numerical round-off error). However, interpolation does not guarantee consistent values at the boundaries, since truncation error will generally be non-zero. As a result, conservation at grid interfaces cannot be assured and is generally not achieved. It has been recognised that, conservation (amongst other criteria: e.g. an increase in net entropy of a closed system, stability, consistency, etc.) is a necessary condition for a physically relevant solution to be obtained (Lax, [7]). It is this non-conservation (particularly of the mass flux [1]) that causes the error.

Several other non-conservative interpolation methods have been used by different authors, details of which shall not be presented here. These alternatives are typically higher order interpolation methods. However, since the underlying flow solver used here is second order, there is little to be gained from the use of higher order interpolation methods. If fact, it will be shown presently that the increased region of overlap that is required in order to implement higher order methods (due to the increased interpolation stencil size) can actually be detrimental to solution continuity.

2.2 Conservative Interface Treatment

Despite the potential problems of non-conservative interface treatment, there have been several authors who have reported that non-conservative interpolation methods give satisfactory results for complex flows (e.g. Freitas *et. al.* [8], Fast *et. al.* [9], Basso *et. al* [10] and Chung *et. al.* [11]). However, other authors (e.g. Tang *et al.* [1], Wang [12], Ge *et. al.* [13, 14]) conclude that such a procedure is unsatisfactory, as it often results in a spurious, odd-even de-coupling to the pressure field.

It is expected that the disparities between reported successful and unsuccessful simulations between different authors arises from the fact that in the limit of zero grid spacing (i.e. a continuous field) any interpolation algorithm will be conservative (since in this limit, there is no interpolation to be done). Therefore, when aspiring toward a grid independent solution, non-conservative interpolation methods will give satisfactory results if using a sufficiently fine mesh. This is as was reported by Freitas and Runnels [8], who used standard linear interpolation for all variables and noted that it is possible to circumvent the problems associated with a non-conservative interface treatment by refining their grid. However, it is not desirable to refine the mesh unnecessarily as the additional computational costs can quickly become prohibitive. Alternative interpolation algorithms have therefore been developed that attempt to preserve the conservative properties of the underlying governing equations at the interface, without being prohibitively expensive. Some of these methods shall now be addressed.

2.2.1 DRAGON Grids

Conceptually, the simplest method of ensuring conservation at internal grid interfaces is perhaps the 'Dragon Grid' method. This method was originally developed by Liou & Kao [15], then later extended into three dimensions by Liou *et. al.* [16]. In their method, conservation is ensured at grid interfaces by eliminating the need for inter-grid interpolation altogether. The geometry is first meshed in much the same way as for the overset scheme. Body-fitted grids depict the complex nature of the geometry, while an efficient Cartesian grid fills the majority of the domain. A hole is then cut into the overlap region of the background grid, which extends a few cells further out than for an overset simulation, such that the grids are no longer overlapping (see Figure 2.5 (a)). The gap between grids is then filled with an unstructured mesh that has faces aligned at the interface, thereby ensuring the flux leaving the structured grid is equal to the flux entering the unstructured grid (and vice versa). The resulting mesh is a hybrid mesh. Figure 2.5 illustrates the two step generation procedure, with Figure 2.5 (a) showing the two user generated grids with the hole cut into the background grid (automatically), and 2.5 (b) showing the final DRAGON grid.

The developers of DRAGON grid technology argue that, as the costly unstructured mesh is only used for a small portion of the overall domain, the overall increase in computational costs attributable to expensive unstructured meshes is minimal. The method also clearly preserves the ease of grid generation that was found in the original Chimera scheme.

To demonstrate the effectiveness of DRAGON grid technology, Liou *et al.* [15] present the complex DRAGON grid of Figure 2.6. Each letter is wrapped with a structured body-conforming mesh, which cuts a hole into the Cartesian background mesh. The unstructured mesh that connects

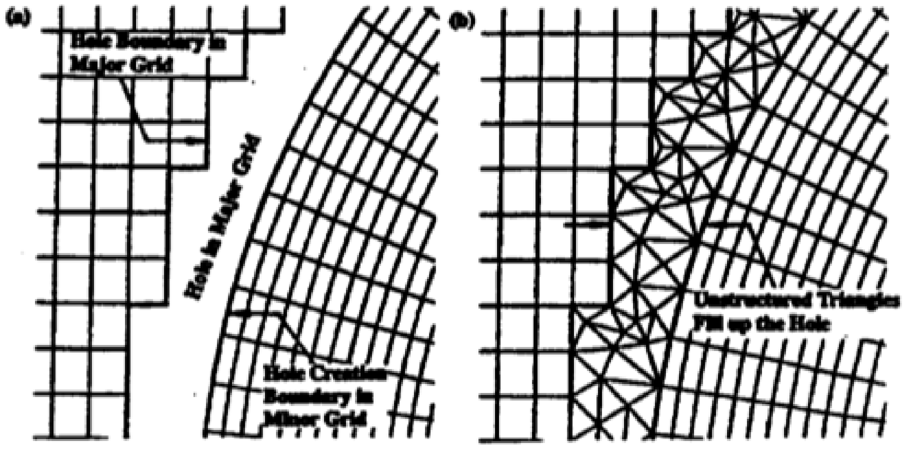

(a)
Hole Boundary in Major Grid
Hole in Major Grid
Hole Creation Boundary in Minor Grid
(b)
Unstructured Triangles Fill up the Hole

Figure 2.5: 2D DRAGON grid generation. Image from [15].

the body-conforming mesh with the background mesh can be seen to occupy only a very small portion of the overall domain. Figure 2.6 clearly demonstrates the potential of DRAGON grids to mesh complex geometries.

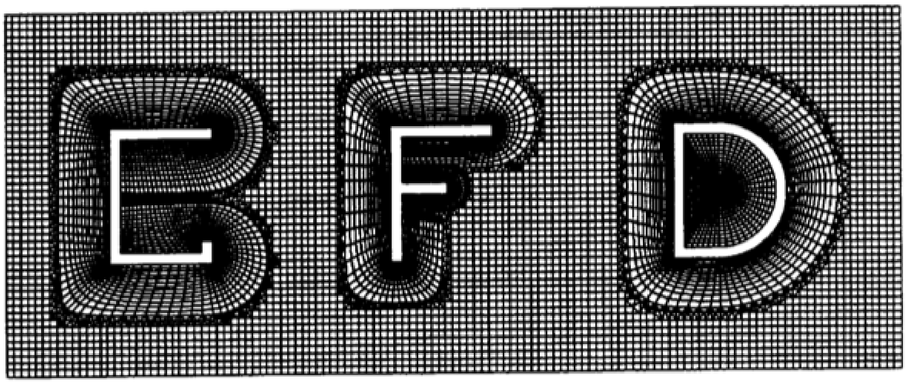


Figure 2.6: DRAGON grid demonstrating the mesh over the letters CFD. Image from [15].

The authors then go on to solve the discretised Euler equations applied over the first letter 'C' from the DRAGON grid (i.e. the 'F' and 'D' grids are removed, presumably for reasons of computational efficiency). The results they present show the time evolution of a shock moving from left to right, toward the 'C'. Pressure contours close up around the 'C' are shown in Figure 2.7. From Figure 2.7 it can be seen that a continuous solution across the interface could be expected. However, the authors fail to plot contours across the unstructured portion of the mesh so this is hard to confirm.

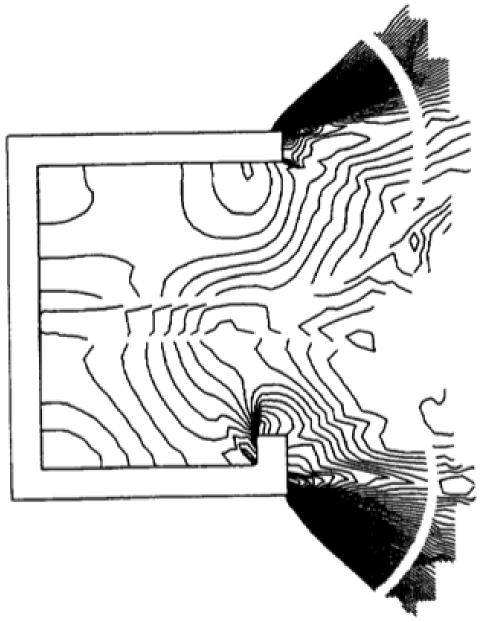


Figure 2.7: Pressure contours around the letter 'C' using DRAGON grid technology. Image from [15].

The extension of 2D DRAGON grids to three dimensions is a relatively straightforward step. It is conceptually the same idea as that of the 2D grids; hence details shall not be repeated here. For further information, the reader is directed toward Reference [16]. The grid shown in Figure 2.8 shows a close up of a mesh used in the simulation of turbine blade film cooling. This figure demonstrates the ability to mesh very complex 3D domains.

The main drawback of DRAGON grid technology is that, for sliding meshes, the unstructured portion of the grid would have to be re-meshed repeatedly as time is marched on, as well as a retagging of holes into the background mesh (as opposed to the Chimera scheme where only a simple re-tagging is required). This could add up to a significant increase in computational costs. Further disadvantages are found due to fact the underlying flow solver (and grid generator) must be able to handle both structured and unstructured meshes. Not all codes have this flexibility built into them already and to add such flexibility is not a trivial task. Hence adding DRAGON grid capabilities to an existing code would likely be a major task.

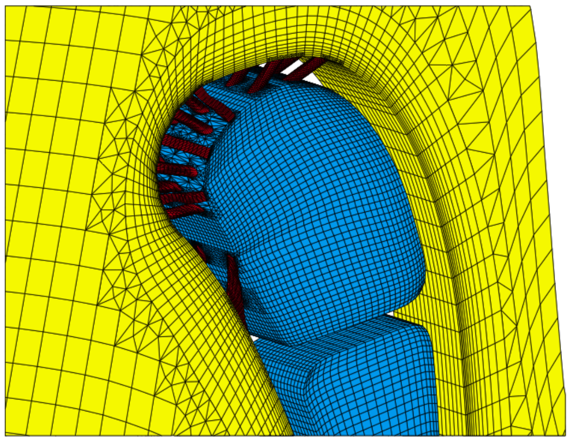


Figure 2.8: A 3D DRAGON grid demonstrating the ability to mesh highly complex domains as encountered in the film cooling of turbine blades. Image from [16].

2.2.2 Zonal Interface Generation

A similar methodology to that of DRAGON grids has been adopted by Wang [17], and was later extended to allow for moving bodies (Wang, [12]). Wang's method is named 'Zonal Interface Generation' (ZIG). Rather than converting the overlapped grids into a hybrid mesh (as was done for DRAGON grids), Wang converts the overlapped grids into a block-structured mesh. Wang eliminates the non-conservative properties of the original Chimera scheme by eliminating the need for interpolation from grid to grid (as was also achieved for DRAGON grids). Figure 2.9 illustrates the grids Wang used to demonstrate the ZIG method. A 'major' Cartesian grid (A) and a 'minor' curvilinear grid (B) are used. To ensure conservation, Wang generates an internal boundary, Γ_{BO} common to both grid A and grid B.

The first step of the ZIG algorithm consists of tagging each node of the major grid as a standard discretisation node (marked '$\times$' in Figure 2.9), a cut cell (marked with a '+'), or an unused node (marked with a '$\bullet$'). Standard discretisation nodes ('$\times$') are solved in the usual manner, while unused nodes ('$\bullet$') are not solved for at all, hence no further action on the part of the interface generator is required at either of these types of point. For cut cells ('+'), the cells are reconstructed such that the cell faces of the major grid cut the cell faces of the minor grid where they intersect (and vice versa). Consider the new cells 'b-6-7-e' and '5-6-7-8-9-f-g' (as seen in Figure 2.9) generated by this process. The flow variables immediately to the left and immediately to the right of the face (Q^L and Q^R respectively) are then evaluated using either a constant reconstruction or a linear

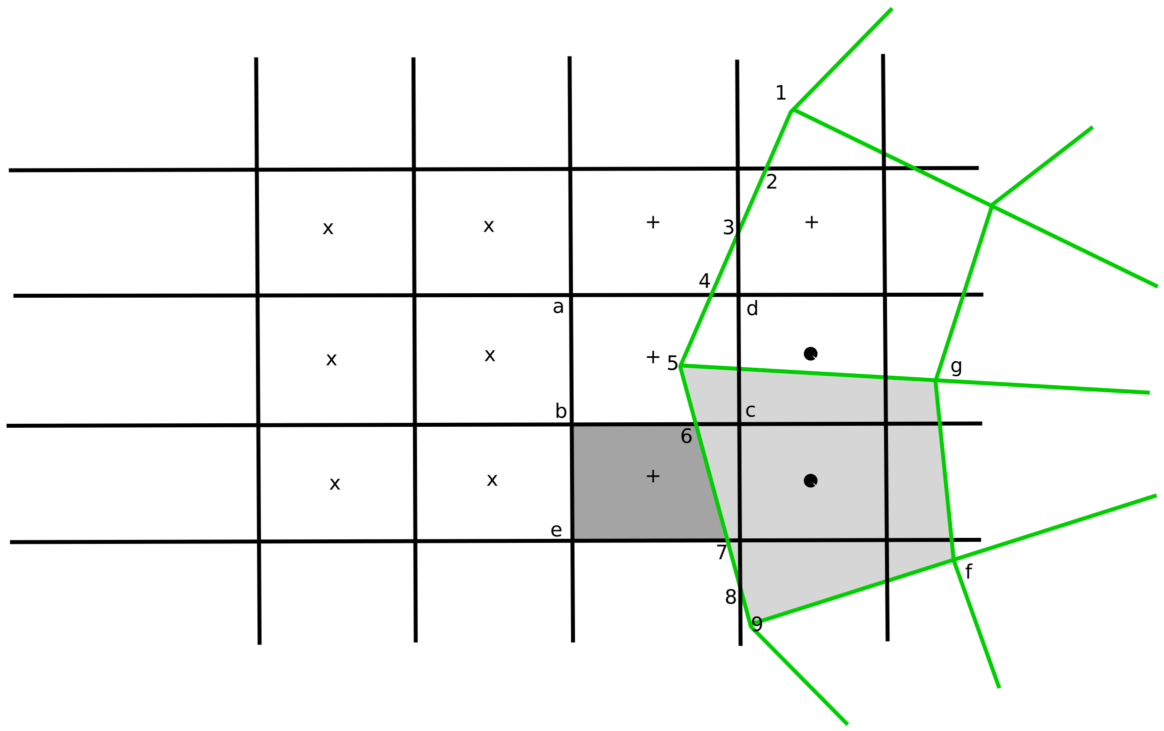


Figure 2.9: 'Zonal Interface Generation'.

reconstruction. For the constant reconstruction (a first-order reconstruction), Q^L is simply taken to be the primitive variable values at the centre of 'b-6-7-e', and Q^R is taken as the primitive variable values at the centre of '5-6-7-8-9-f-g'. For the linear reconstruction, information from neighbouring cells is brought in to determine the local gradient of Q which is used to determine Q^L & Q^R to second order accuracy with respect to grid spacing. Having found Q^L and Q^R, Wang employs the approximate Riemann solver of Roe to find the numerical flux through each cut face (for details of the Riemann solver see Ref. [18]).

Wang demonstrated the ability of the ZIG algorithm by simulating a slow moving shock moving through grid interfaces, a particularly challenging simulation that non-conservative interface schemes would have major difficulties with. The grids used in Wang's simulation are shown in Figure 2.10.

Density contours for the moving shock problem are shown in Figure 2.11, where the shock is moving from left to right. Comparisons are made between the original, non-conservative Chimera (b), and the ZIG method (a). It can be seen from Figure 2.11 that the original Chimera scheme develops instabilities that cause the erroneous dissipation of the shock. The ZIG method however, resolves the moving shock through the interface reasonably well.

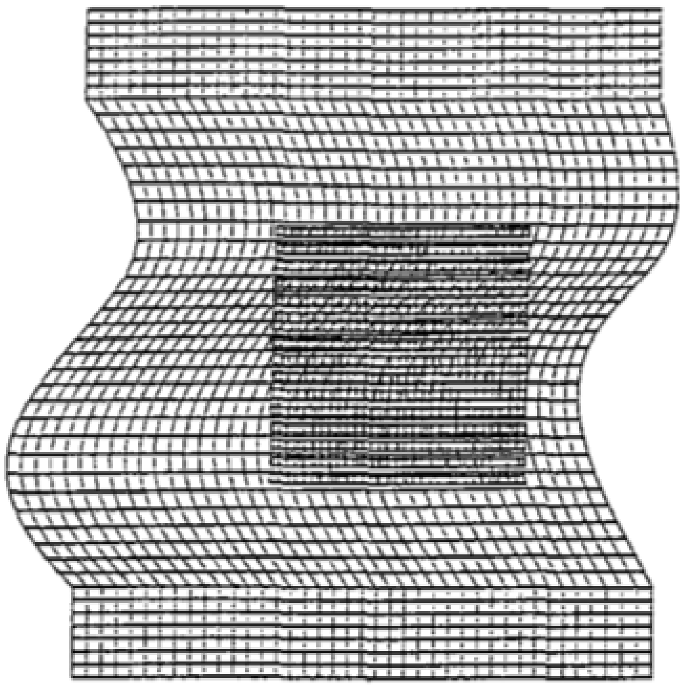


Figure 2.10: The grids used by Wang to simulate a slow moving shock problem. Image from [17].

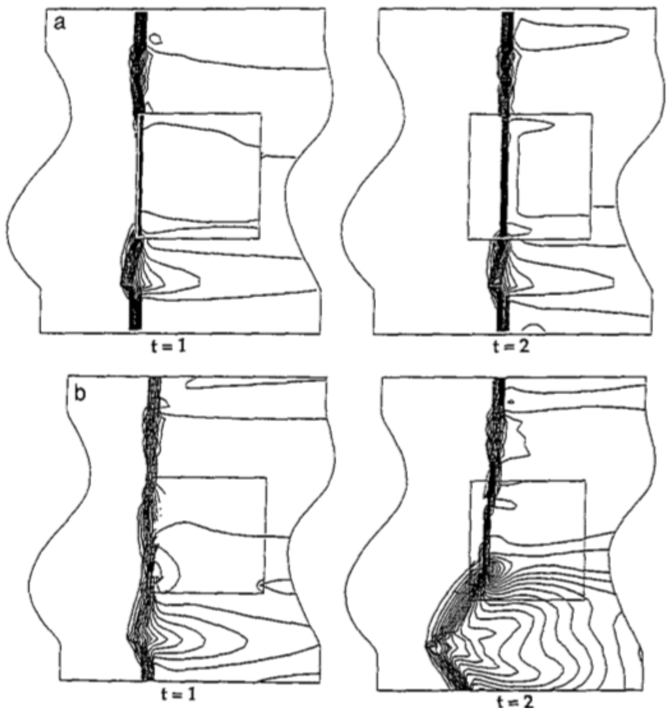


Figure 2.11: Density contours for moving shock problem at times $t = 1$ and $t = 2$ for the ZIG method (a) and non-conservative Chimera (b). Image from [17].

The main problem with the ZIG method is that the CPU time associated with the interface generation in 2D, is of the same order of magnitude as that of one time-step of the flow solver [17]. In 3D, Wang expects the interface generation to be *much more* resource intensive [17]. For non-sliding meshes, this is not particularly problematic, however if the meshes were sliding, the interface would need to be regenerated repeatedly and this could become prohibitively expensive (especially in 3D). The extension of ZIG to three dimensions was apparently never carried out.

2.2.3 Conservative Interpolations

The previous conservative interface treatments mentioned (namely; DRAGON grids and the ZIG method) required partial re-generation or reconstruction of the grids from one time-step to the next (if sliding meshes are to be used). This is a costly procedure that would be best avoided if possible. If a conservative interpolation method could be developed, the regeneration of grids could be avoided.

Berger presents a discussion of conservative interpolation in one spatial dimension and for specific cases in two spatial dimensions, where the grids are aligned in specific ways. Her paper (Ref. [19]) discusses conservation issues for overlapping grids. The basic idea presented in [19] involves assuming the interpolation coefficients are free parameters, and then deriving constraints on these parameters that ensure conservation.

Berger notes that, for a conservative interpolation, it is necessary to interpolate the *flux* through the internal boundary rather than the value of the variable directly. In one dimension (see Figure 2.12) Berger shows that a *linear* interpolation of the flux is conservative.

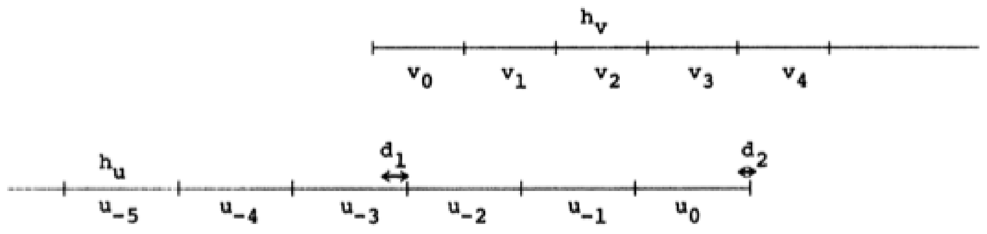


Figure 2.12: 1D grids used by Berger to demonstrate conservative interpolation. Image from [19].

In two dimensions, the result is not so straightforward. Berger develops conservative interpolation methods where the grids are aligned in specific ways, such that the problem is simplified to the extent where a conservative interpolation may be performed. Figures 2.13 and 2.14 illustrate the problems considered by Berger which are of increasing complexity.

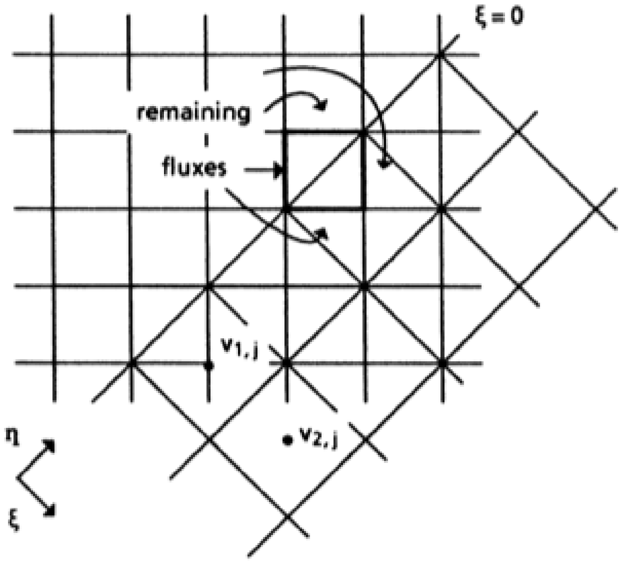


Figure 2.13: 2D grid used by Berger to develop conservative interpolation. Image from [19]

Figure 2.13 considers the case where the grids are rotated by $\pi/4$ with respect to one and other, and where the grid lines of one grid go through the corners of the other. In this case, it is shown that the conservative interpolation method actually reduces to linear interpolation. In other words, linear interpolation is conservative for this specific case. This follows from the fact that the interpolation points lie at the cell centers of the donor cells.

Figure 2.14 considers a second, somewhat more complex situation, where the grids are rotated, but the gridlines need not exactly go through the corners of the other grid. For this case, Berger describes the restrictions to the interpolation weights that ensure global conservation. It is found that, for conservative interpolation, it is necessary to use a 6-point interpolation stencil, which does not fit into the finite volume method in a natural way.

Finally, Berger goes on to describe a general algorithm for 2D, but the details are lacking, and it was apparently never implemented. Furthermore, numerical experiments were not conducted by Berger in [19], so the success of the interpolation scheme is difficult to confirm. The extension of Berger's interpolation method to three dimensions would likely be prohibitively complex.

Chesshire and Henshaw extend on the preliminary work of Berger by generalising the 2D algorithm to make it applicable to any number of arbitrarily arranged grids (Ref. [20]). Numerical

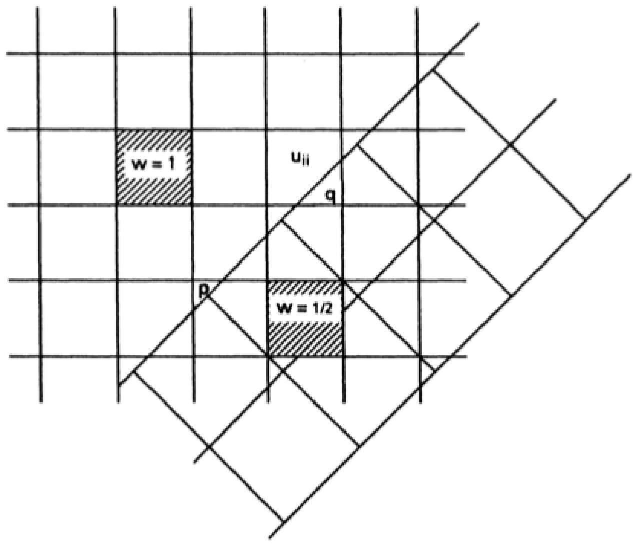


Figure 2.14: A more complex 2D interpolation problem, considered by Berger. Image from [19]

experiments were conducted by simulating the slow moving shock problem on overlapping grids. The governing equation they use is the viscous Burger's equation rather than the complete Navier-Stokes equations. Essentially they are making a simplification by solving a 1D governing equation, but on 2D grids. Burger's equation is given in Equation 2.1.

$$\frac{\partial u}{\partial t} + \frac{1}{2} u \frac{\partial u}{\partial x} = \mu \frac{\partial^2 u}{\partial x^2} \tag{2.1}$$

The results they obtained from applying the conservative interpolation to the above-mentioned problem are shown in Figure 2.15.

Chesshire and Henshaw computed the difference between the conserved discrete integral of u, and the analytical integral of the exact solution, both integrated over the entire domain,. The difference between these two quantities is representative of the error. The mean value of the error was subtracted out, as this portion of the error is attributable to truncation error and computational round off error. What remains is the conservation error. A plot of this conservation error, as a function of time was presented by Chesshire and Henshaw (Ref. [20]), and is reproduced here as Figure 2.16.

It can be seen from Figure 2.16 that the conservation error is below 3×10^{-11} at all times. It is therefore of negligible magnitude relative to the mean truncation error (3.26×10^{-6}). It can also be

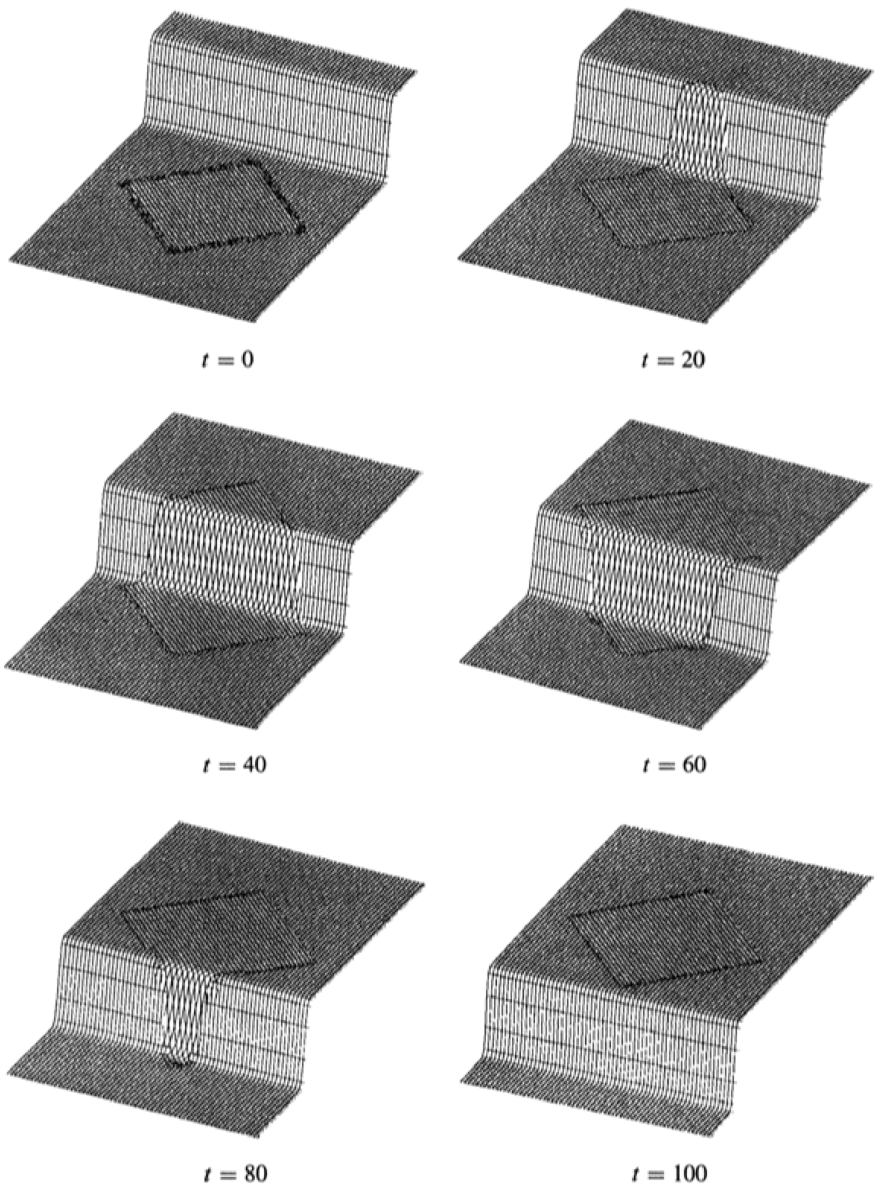


Figure 2.15: Results of simulating the slow moving shock problem, using the viscous Burger's equation as a governing equation, and the conservative interpolation of Chesshire and Henshaw. Image from [20].

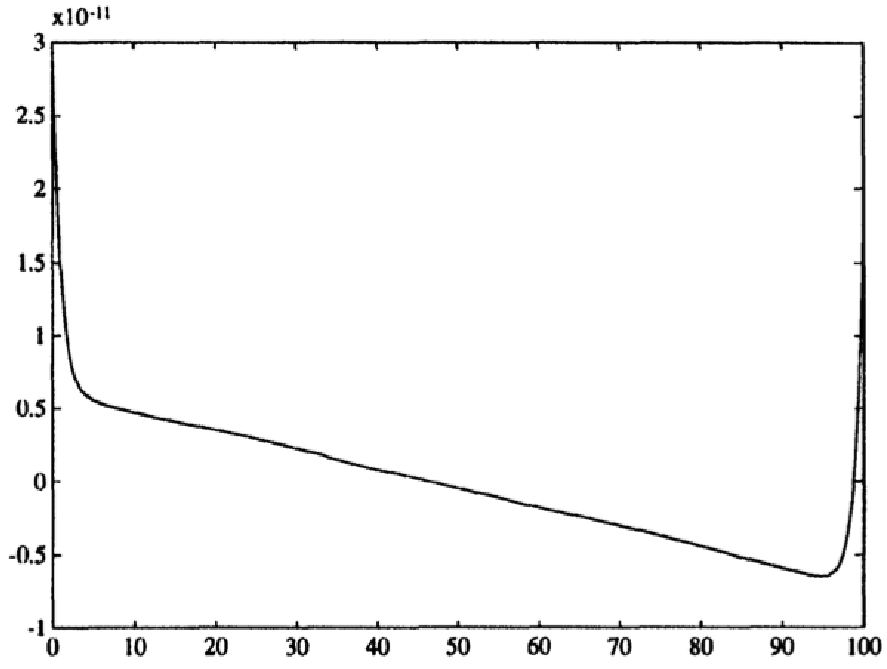


Figure 2.16: Conservation error as a function of time. Image from [20].

seen that the error is largest at times $t = 0$ and $t = 100$. This is due to the fact that, at these times, the shock is closest to the computational boundary; the error is higher in this case due to the large truncation error in the discretisation of the boundary conditions, rather than due to conservation of the interpolation scheme. At time $t = 20$, when the shock starts to cross the internal interface, there is no notable jump in the conservation error. The conservation properties of the interpolation scheme are therefore confirmed to be fully conservative.

That Chesshire and Henshaw chose to use the viscous Burger's equation as the governing equation, rather than the full Navier-Stokes equations, is due to the fact the former equations offer a significantly simpler problem to be solved. Extending the conservative interpolation considered by Chesshire and Henshaw to the general case of solutions to the complete Navier-Stokes equations would likely involve difficulties, particularly in the over-prescription of boundary conditions. Furthermore, the extension of this interpolation method to 3D would involve significant additional complications.

2.2.4 Semi-Conservative Interpolations

The work of Berger [19], and Chesshire *et. al.* [20] has shown that a fully conservative interpolation scheme is extremely difficult to implement, even for simple governing equations over 2D domains. The development of a fully conservative interface treatment for the solution of the full Navier-Stokes equations over 3D domains is prohibitively complex at present (and is likely to remain so). An alternative *semi*-conservative interface treatment has been developed by Tang *et. al.* [1].

In their method, Tang *et. al.* derive a condition that must be satisfied for the exact conservation of mass. However, they note that the enforcing of the condition is *extremely* difficult (if not impossible) to implement in 3D (as was also found by Chesshire and Henshaw). The method is based on enforcing a second order accurate approximation to the condition, rather than the exact condition. Since they enforce a discrete approximation to the condition, and not the exact condition, the method is said to be *semi* conservative. Tang *et. al.* refer to their method as Mass-Flux Based Interpolation, or MFBI.

Despite not being fully conservative, the MFBI method does prove to be very useful. Tang *et. al.* show that the MFBI reduces the odd-even de-coupling found on colocated overset grids, relative to linear interpolation (where both methods use *no* pressure smoothing in order to exagerate the problem for demonstration purposes). Figure 2.17 shows the grids used by Tang *et. al* for the simulation of a 3D unsteady flow in a cube cavity. The sub-domain, Ω_A oscillates vertically with time ($W = \cos(2\pi t)$). The domain is closed, with the fluid inside being driven entirely by the moving lid.

Figure 2.18 shows contours of pressure and two velocity components for a cross section through the cube at the $y = 0$ plane, with a comparison between the MFBI and linear interpolation methods. It can be seen from Figure 2.18 that the velocity field is very close to continuous across both grids. However the pressure experiences strong odd-even decoupling in the case of linear interpolation, but is significantly reduced by using the MFBI method.

Figure 2.19 shows the convergence histories of the pressure and velocities for the oscillating cavity problem.

From Figure 2.19, it is evident that the residual falls to a lower level at each timestep for the MFBI method relative to that of linear interpolation. This is the case despite using the same grids for each case, and is due to the fact the conservation error is lower for the MFBI method, compared to the

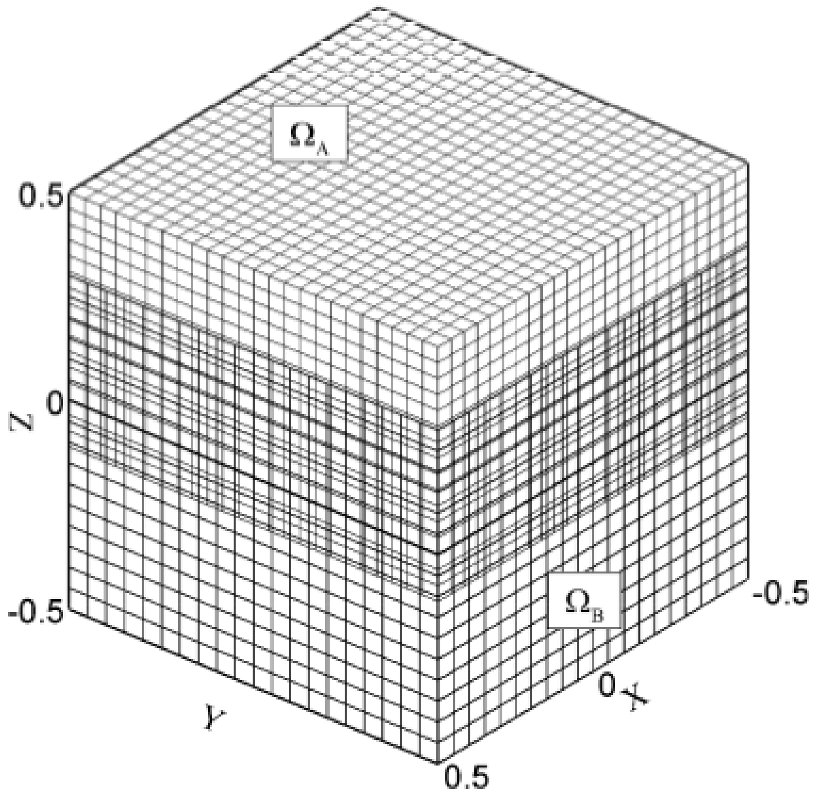


Figure 2.17: The grids used for the simulation of 3D oscillating flow.

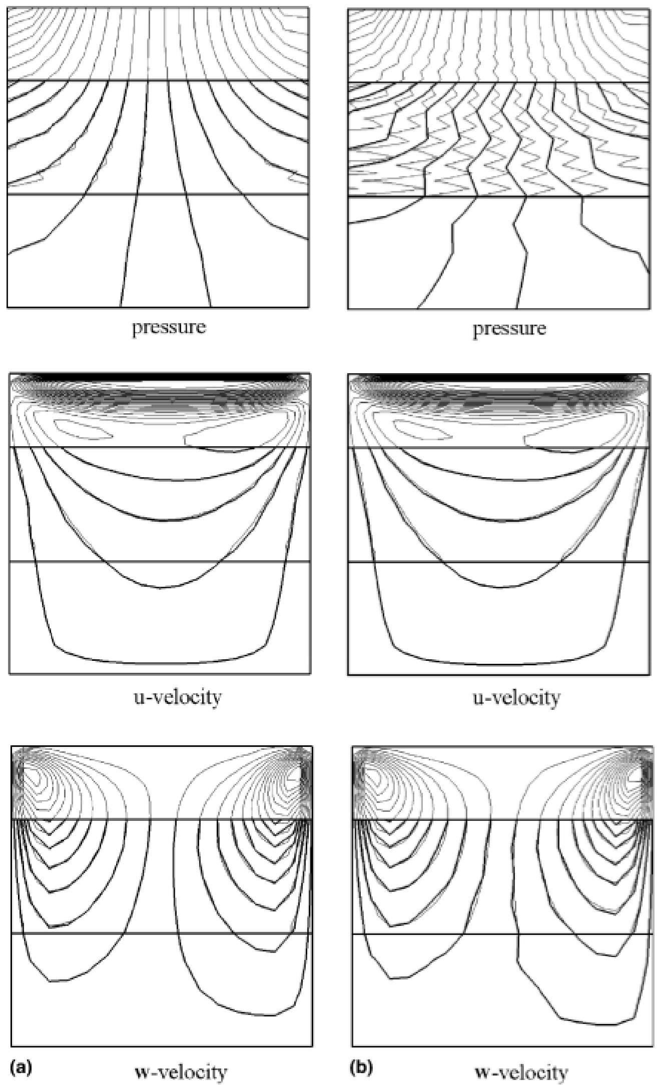


Figure 2.18: Pressure and velocity contours at time $t = 0.25$ and $Re = 100$. (a)MFBI; (b)linear interpolation. Image from [1].

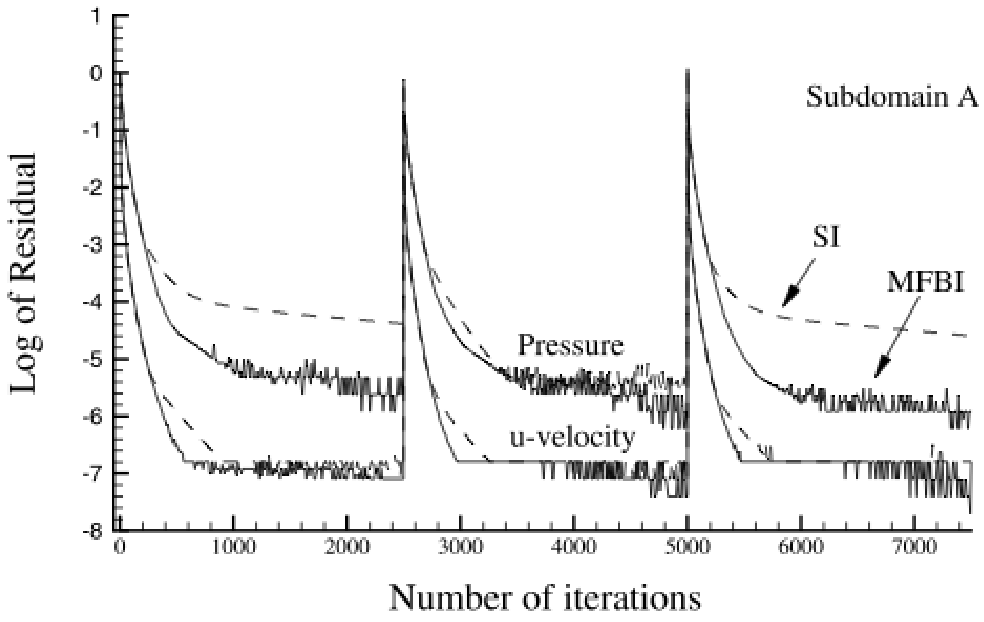
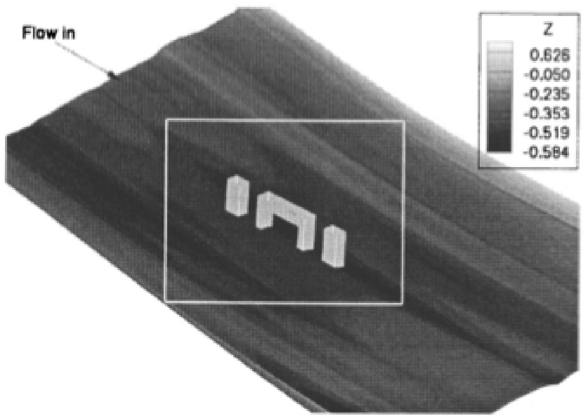


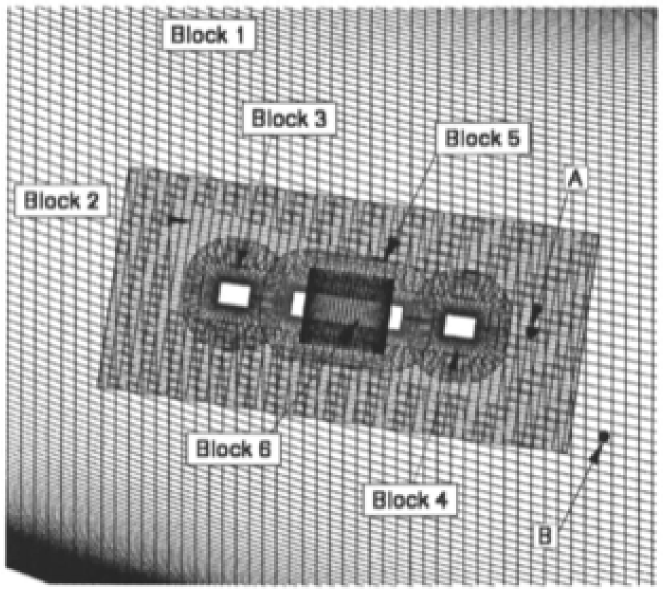
Figure 2.19: Convergence histories for the pressure and U-velocity residuals for three timesteps. (SI=linear interpolation). Image from [1].

linear interpolation method.

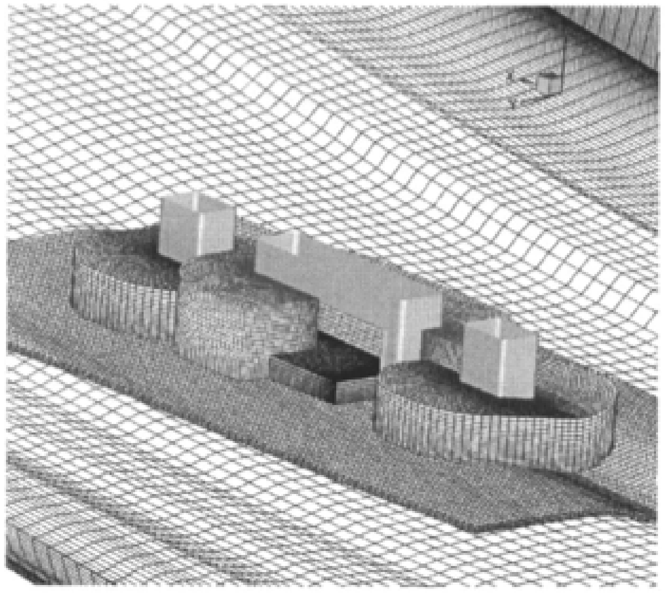
The MFBI method has been applied by Ge and Sotiropoulos [13, 14] to a river flow around a bridge foundation. Figure 2.20 shows the geometry and grids used in this case, from which it can be seen that the former is complex, while the latter is not. The resulting flow stream traces are shown in Figure 2.21.

Kangle and Gang, [21], use the MFBI method to simulate the flow over multi-element airfoils. It is found that the method again offers improved convergence and slightly improved results (suggesting that grid independence was not achieved in their study). Figure 2.22 shows the predicted pressure coefficient over the surface of the multi-element configuration. It can be seen that the MFBI results are closer to experiment data than the results obtained from using linear interpolation.





(a)



(b)

Figure 2.20: Geometry (above), and overset grids used in [13] for the flow around a bridge foundation.

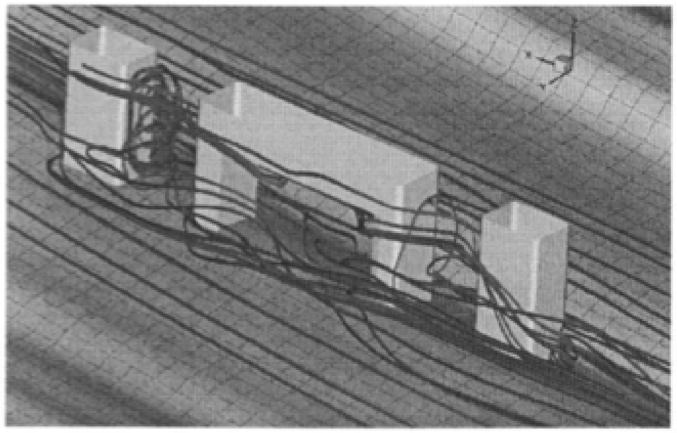


Figure 2.21: Flow stream traces obtained in [13] for the flow around a bridge foundation.

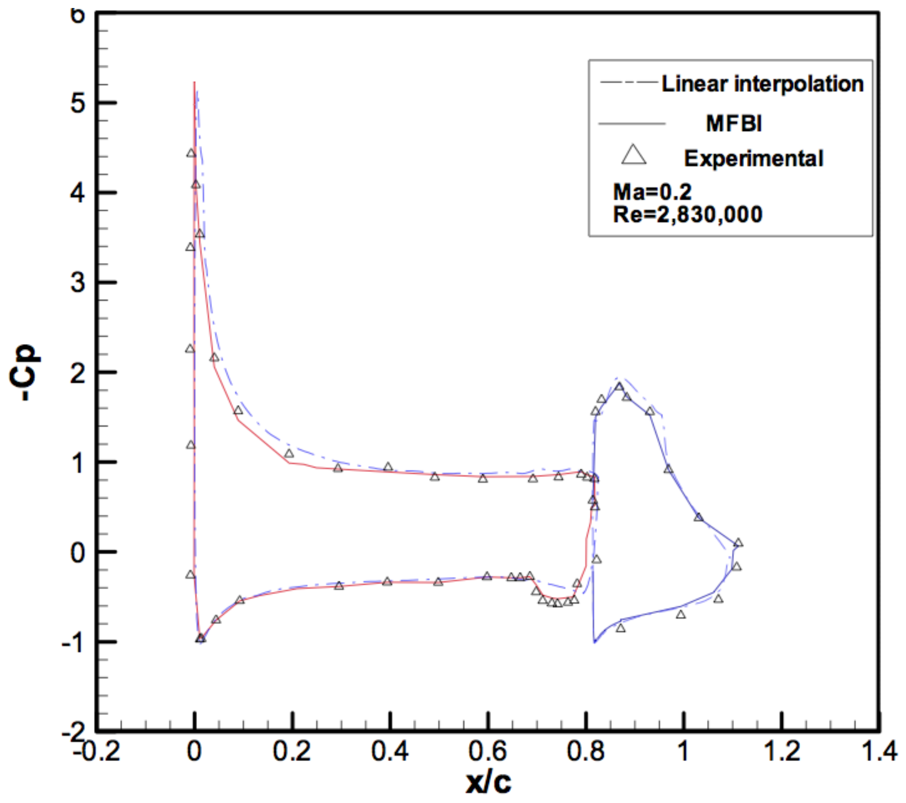


Figure 2.22: Surface pressure coefficient over a multi-element airfoil. Image from [21].

2.3 Alternative methods to the overset method in dealing with complex geometries

An attractive alternative to the overset grid method is the immersed boundary method as proposed by Peskin [22, 23]. In this method, a simple and efficient Cartesian mesh is used throughout the entire domain (see Figure 2.23). The Cartesian mesh is generally non-conformal to the boundaries of the geometry being considered (i.e. boundary Γ_b). Instead, the effects of the boundary Γ_b are accounted for through modification of the governing equations. Since there is no requirement for grids to be aligned to the geometry, the pre-processing step is simplified for arbitrarily complex geometries.

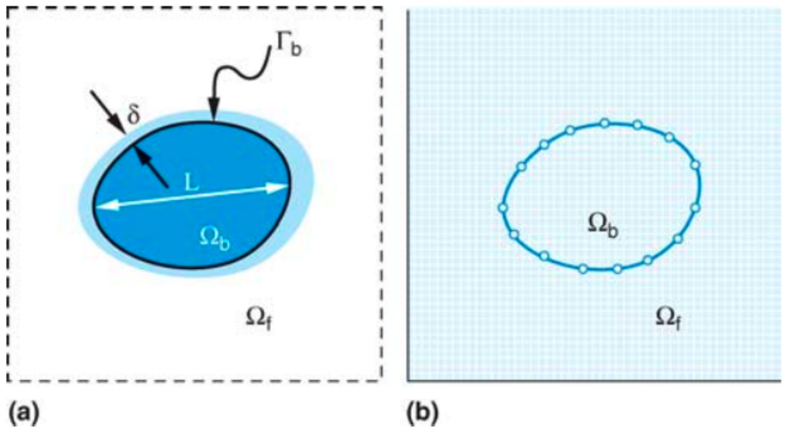


Figure 2.23: Immersed boundary method example problem and grid. Image from [24].

To deal with the wall boundary conditions presented by Γ_b, modifications are made to the governing equations. These modifications take on the form of additional source terms in the momentum and turbulence equations, with the aim of reproducing the effect of the boundary.

There are two distinct methods of dealing with immersed boundary conditions. The first implementation involves applying a continuous forcing function to the governing equations, and then discretising the result. The second option involves discretising the governing equations with no regard to the immersed boundary, and then applying modifications in the vicinity of the immersed boundary in an *ad hoc* manner, to account for its presence.

There are fundamental differences in these two methods. The former method has a strong physical basis, whereas the latter method arguably demonstrates less rigour. Despite this, both methods feature prominently, and both have associated advantages and disadvantages with their use.

In the former, continuous approach, the immersed boundary can be modelled as a set of massless points, connected to one another via 'elastic fibres', and tethered to a fixed anchor point (see Figure 2.24). A mixed Euler-Lagrangian approach is formulated, where the Navier-Stokes equations are solved in the Eulerian domain, Ω, and the fluid-boundary interactions are tracked on a Lagrangian domain.

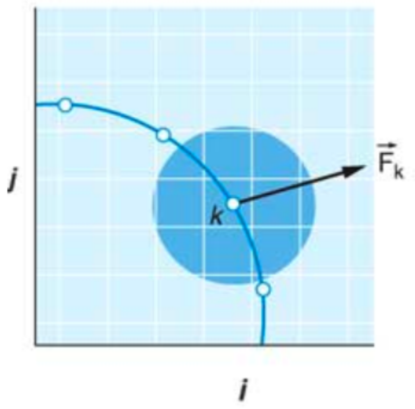


Figure 2.24: The transfer of force to surrounding fluid nodes. Massless points are represented by the white circles, and are connected by the blue 'elastic fibres'. Shaded region shows the region of influence of a point. Image from [24].

A forcing function, $\vec{f}$, is added to the right hand side of the Navier-Stokes momentum equations and is given by

$$f_i(\vec{x},t) = \sum_k F_{i_k}(t)\delta(\vec{x} - \vec{X}_k) \tag{2.2}$$

where $\vec{F}_k$ is the force density acting at point k (due to the tension in the elastic fibres, and due to the tethers) and $\vec{X}_k$ is the position vector of the k^{th} massless point. Since the location of the massless points do not generally coincide with the grid nodes, the force is distributed to the fluid via a smoothed-out discrete version of the Dirac δ-function. Some of the possible distributions of the δ-function that have been used are illustrated in Figure 2.25.

Clearly a method of specifying $\vec{F}$ at each point k is required. To model the elastic behaviour of

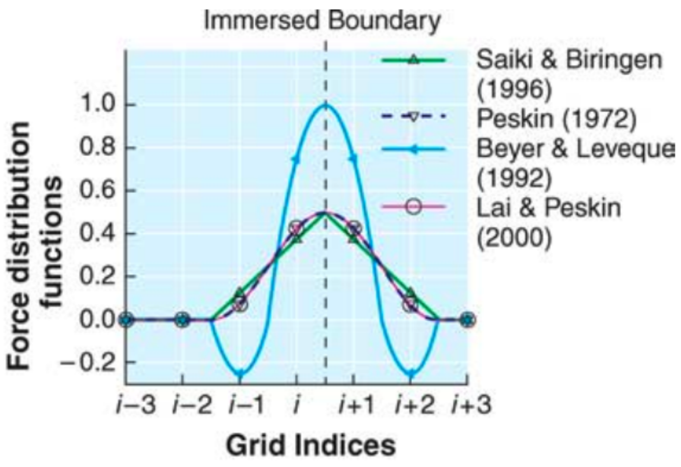


Figure 2.25: Different approximations of the Dirac δ-function. Image from [24].

the immersed boundary, we have;

$$F_i(s,t)^{ELASTIC} = \frac{\partial}{\partial s}(T\tau_i) \tag{2.3}$$

where T is the tension in the elastic fibres, and $\vec{\tau}$ is the unit tangent. The tension in the elastic fibres is obtained by relation to the deformation of the fibres through constitutive relations such as the generalised Hooke's law.

The restoring tethers can be modelled as springs. Each massless point is then thought to be attached to its equilibrium position via this spring. The equilibrium position is usually taken as the initial starting position of the massless points at time $t = 0$. The tether force is then given by

$$F_i(s,t)^{TETHER} = -\kappa\,[X_i(s,t) - X_i(s,0)] \tag{2.4}$$

where κ is the spring constant. The force $\vec{F}^{TETHER}$ is the restoring force, driving the immersed wall back to its original position as the fluid pushes it away. The final force $\vec{F}$, for use in Equation 2.2, is then given by;

$$F_{i_k} = \frac{\partial}{\partial s}(T\tau_i) - \kappa\left[X_{i_k}(s,t) - X_{i_k}(s,0)\right] \qquad (2.5)$$

The modulus of elasticity of the fibres (used to relate the tension in the fibres to their displacement), and the spring constant, κ, are user defined parameters that can be selected to yield the desired wall properties.

The solution procedure commences by solving for the flow-field at time $t = 0$. There will initially be no forces $\vec{F}$ acting on the massless points since the immersed boundary is in its equilibrium position. However, from the no-slip condition we know the immersed boundary wall must move with the local fluid velocity. This fact can be used to update the position of the massless points from one time-step to the next via

$$\frac{\partial}{\partial t}X_i(s,t) = u_i(\vec{X},t) = \int u_i(\vec{x},t)\delta(\vec{x} - \vec{X}(s,t))d\vec{x} \qquad (2.6)$$

At the end of the first time-step, the immersed boundary will therefore move downstream by distance $U_\infty\Delta_t$, where U_∞ and Δ_t are the freestream velocity and time-step respectively. This displacement will result in a restoring force, calculated via Equation 2.5. Equation 2.2 is then used to distribute this force to the fluid, resulting in a source term in the Navier-Stokes equations. In an iterative manner, this will lead to the desired result of a simulation of the flow around the immersed boundary.

While this method is well suited for simulating flow over elastic boundaries, using the method to simulate the flow around rigid bodies poses many challenges. In the rigid limit, the constitutive laws used for the elastic boundaries are ill-posed. While it is theoretically possible to treat the system as strictly elastic, but with a very high κ value, this can often lead to stability issues [25]. There are alternative immersed boundary formulations that are better suited to rigid body simulations, but they will not be considered here. For further information, see References [24, 26].

To demonstrate the potential of the immersed boundary method, Yun *et. al.* [27] performed an LES simulation of the turbulent flow past a sphere at $Re = 10^4$. Figure 2.26 shows computed particle tracks, with comparison to an experimental flow visualisation.

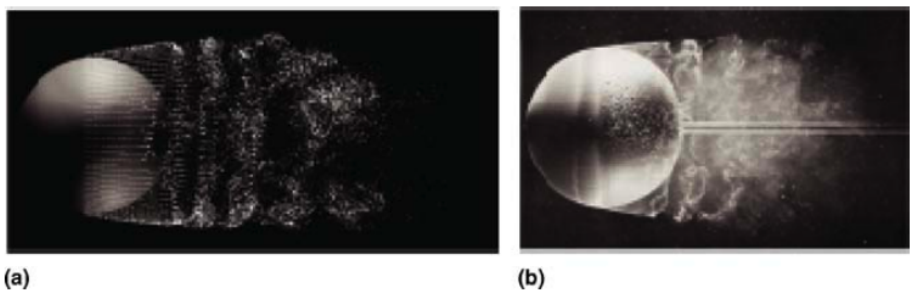


Figure 2.26: Turbulent flow over a sphere. (a) Immersed boundary LES particle tracks. (b) Experimental flow visualisation. Image from [24].

From Figure 2.26, it is apparent that the computed flow field is qualitatively similar to that of the experiment. The vortex structures are similar in form from computation to experiment. Table 2.3 shows comparisons of average base pressure coefficient $\overline{C_P}$, average drag coefficient $\overline{C_D}$ and Strouhal number St from different computations or experiments. From Table 2.3 it is apparent that the LES immersed boundary method seems to perform just as well as a detached eddy simulation with body-conformal grids for this case. It is also evident that the computed vortex shedding frequency ($f \propto St$) largely agrees with that of experimental data, to within a few per cent.

	$\overline{C_P}$	$\overline{C_D}$	St
LES with Immersed Boundary [27]	-0.274	0.393	0.167
Experiment. Kim *et. al.* [28]			0.16
Experiment. Sakamoto *et. al.* [29]			0.18
Detached eddy simulation with body conformal grids. Constantinescu *et. al.* [30]	-0.275	0.393	0.195

Table 2.1: Predicted and experimental values of average base pressure coefficient ($\overline{C_P}$), average drag coefficient($\overline{C_D}$) and Strouhal number St for the flow over a sphere at $Re = 10^4$ from different computations/experiments.

Mittal *et. al.*, [31], simulated the free-fall of five blocks through a fluid, with arbitrary initial arrangement, using the immersed boundary method. Figure 2.27 shows the position of the blocks and computed pressure contours at three different time-steps. This simulation would be very time consuming to conduct using an unstructured grid simulation since the grid would have to be regenerated at each time-step.

Finally, Fauci and McDonald used the immersed boundary method to perform a study on the motility of sperm cells [32]. They noted that despite the simplistic two-dimensional simulations

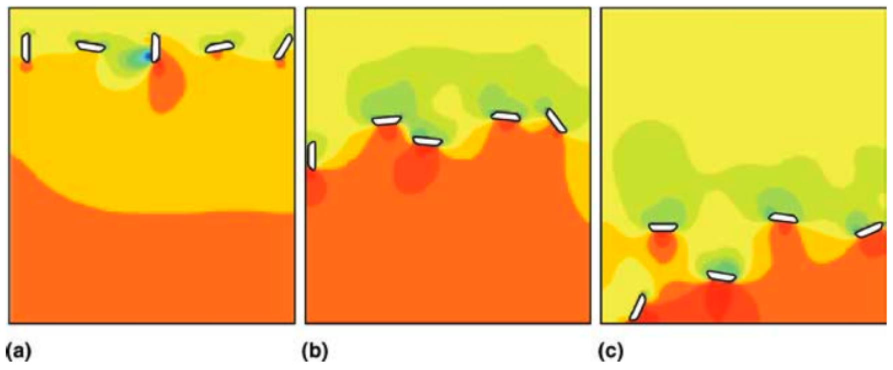


Figure 2.27: The free-fall of 5 blocks through a fluid. (a) Initial stage. (b) Intermediate stage. (c) final stage. Image from [31].

they performed, the qualitative features of their simulations (such as the cell accumulation around walls and the phase locking of neighbouring cells flagellum) matched experimental observations. Figure 2.28 illustrates the velocity vectors induced by two cells swimming in a channel, bounded by elastic walls. This figure demonstrates the immersed boundary method's ability to model complex geometrical shapes involving fluid-structure interaction.

The three different immersed boundary simulations outlined above (the flow over a sphere, the free-fall of five blocks and the swimming of sperm cells) demonstrates the potential of the immersed boundary method. The method clearly has a wide range of applicability and also has the advantage of being easy to implement (Mittal *et. al.* note that it would be possible to implement the method into an existing CFD solver, within a matter of weeks [24]).

Clearly there are many advantages associated with the immersed boundary method. However, one major disadvantage of the method is that, relative to the overset method, the number of grid nodes required to reach a grid independent solution is often significantly higher. With reference to Figure 2.23a, for a simulation with characteristic length L, and boundary layer thickness δ, it may be required to provide an average grid spacing of at least Δ_n and Δ_t in the normal and tangential directions to the body respectively. In this case, for a body-fitted grid, the number of grid nodes required for a grid independent solution will scale as $(L/\Delta_t)(\delta/\Delta_n)$, whereas for a Cartesian grid, the number of nodes required scales as $(L/\Delta_n)^2$. Assuming that $\Delta_t \propto L$ and $\Delta_n \propto \delta$, we can note that the *ratio* of the number of nodes required in Cartesian grids to body-fitted grids will scale as $(L/\delta)^2$. According to Schlichting and Gersten, [33], the ratio (L^2/δ^2) increases linearly with Reynolds number for laminar flows. Hence, as the Reynolds number is increased,the number of

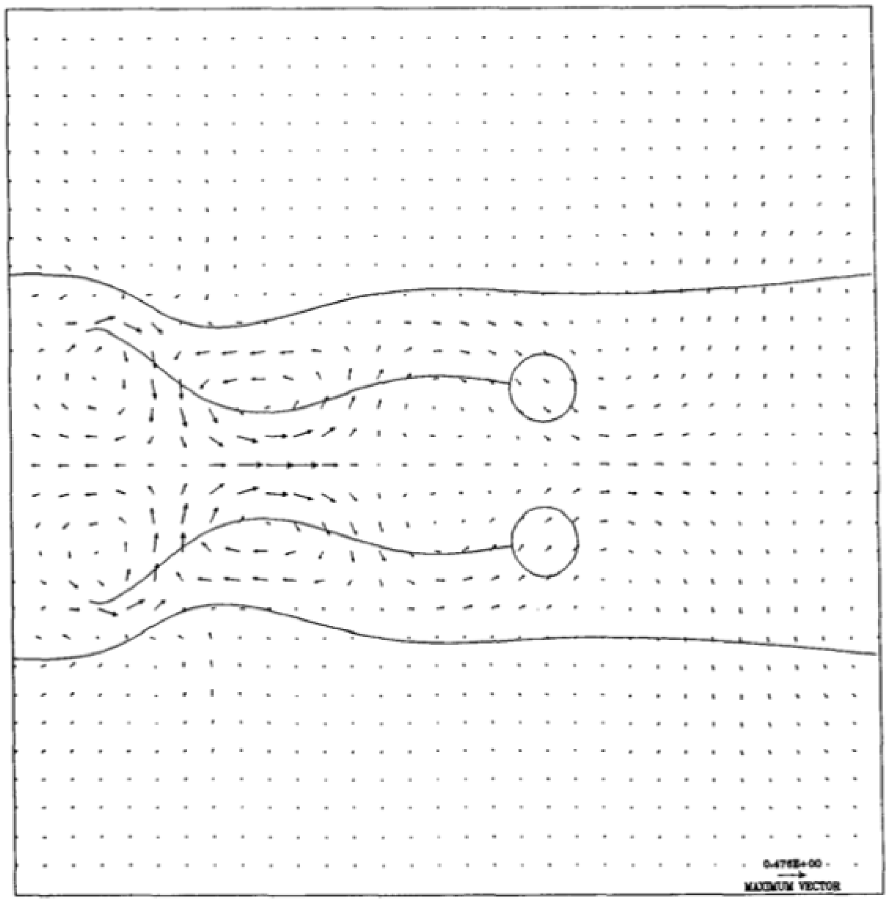


Figure 2.28: The simulation of sperm cells swimming in a channel with elastic walls. Image from [32].

grid nodes required will increase at a greater rate for the immersed boundary method than for the overset method. This result is also intuitively obvious since body-fitted grids usually employ high aspect-ratio cells to efficiently deal with near-wall regions. In the immersed boundary method this is not possible since the grid is generated with little or no regard to the location or orientation of any immersed boundaries.

It is clear that the immersed boundary method does offer an attractive way of dealing with complex geometries. However, due to the larger computational grids that are often required (relative to the overset method), and due to issues surrounding stability in the rigid limit, the immersed boundary method is not particularly well suited to all classes of problems. However, a hybrid immersed-overset methodology would seem to be a powerful tool in dealing with a wide range of complex problems.

2.4 Conclusion

Several different methods for dealing with overset grids have been outlined in this literature review. It has been noted that fully conservative interface treatment is either too costly when using sliding meshes, or is very difficult to implement. Semi-conservative methods offer an attractive compromise between accuracy and simplicity. The MFBI method is one such semi-conservative interpolation method that is very easy to implement in two or three spatial dimensions. It has been shown that, for a variety of cases, the MFBI method performs better than using linear interpolation [1]. It seems to be that the MFBI offers the best available method for dealing with the interface conservation issues at present. Numerical details of the MFBI method shall be considered in Chapter 4.

Chapter 3

Numerical Aspects of a Structured CFD Code

3.1 Introduction

The equations that govern the physics of a fluid flow are the Navier-Stokes equations, and are presented below (using compact tensor notation, with implied summation over repeated indices), for a Cartesian reference frame:

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_j}\left[\rho u_j\right] = 0 \tag{3.1a}$$

$$\frac{\partial}{\partial t}\left[\rho u_i\right] + \frac{\partial}{\partial x_j}\left[\rho u_i u_j + p\delta_{ij} - \tau_{ji}\right] = 0 \qquad i = 1, 2, 3 \tag{3.1b}$$

$$\frac{\partial}{\partial t}\left[\rho e_0\right] + \frac{\partial}{\partial x_j}\left[\rho u_j e_0 + u_j P + q_j - u_i \tau_{ij}\right] = 0 \tag{3.1c}$$

where δ_{ij} is the Kronecker delta ($\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ otherwise), τ_{ij} is the viscous shear stress tensor, q_i is the local heat-flux vector, and e_0 is the total energy ($e_0 = C_v T + \frac{1}{2} u_k u_k$).

Equations 3.1 are simply statements of conservation. They can be derived by considering the conservation of mass over an infinitesimal fluid element (giving Equation 3.1a), and by applying Newton's second law of motion and the first law of thermodynamics over the same fluid element (giving Equations 3.1b and 3.1c respectively). Equations 3.1a, 3.1b and 3.1c are referred to as the continuity condition, the momentum equations (with the i index being cycled through to generate a separate equations for the x, y and z directions), and the energy equation respectively.

The Navier-Stokes equations are coupled, non-linear partial differential equations which have no general analytical solution. Traditionally the only way of gaining useful analytical information from the Navier-Stokes equations has been to reduce the equations to a linear form by making several simplifying assumptions. For example, by neglecting viscosity, the Euler equations are obtained; the Euler equations can then be further simplified by neglecting terms describing vorticity, yielding the full potential equations. Finally, the potential equations can be linearised by assuming small perturbations to the flow field (i.e. by neglecting second order and higher terms). The linearised equations can then be solved analytically. This procedure has indeed provided great insights into the underlying physics of fluid flow. However, the simplifying assumptions do not fully depict the physics of real-world flows, and in most cases the application of linearised equations to engineering problems is inappropriate. For example, the application of the potential equations to a body (of arbitrary shape) in an inviscid flow field leads to the erroneous prediction of zero drag. This is known as d'Alembert's paradox.

The desire for detailed, quantitative information on complex, turbulent, three-dimensional flows, as is typically encountered in real-world circumstances, cannot be satisfied through analytical solutions. Before the advent of the modern computer, the only option available to the fluid-dynamicist was to gather flow data through experiments. In more recent times, as the computer has become increasingly commonplace, the field of Computational Fluid Dynamics (CFD) has evolved, and is now frequently used in conjunction with experiment.

CFD involves the replacement of the integrals and partial derivatives found in the governing equations with discretised algebraic forms. These algebraic equations are then solved numerically on a computer to obtain *numbers* at each discrete point. The numbers are representative of the three velocity components, the pressure, the temperature and the density at each node, as well as any other problem dependent variables that may be solved for. In order to close the set of equations 3.1, it is generally necessary to relate the local pressure to the local fluid density and temperature. Empirical relations may be used for this purpose, such as the ideal gas law, with the local fluid temperature and density being found via the energy equation and continuity condition respectively.

It is worth briefly noting that, while the Navier-Stokes equations are applicable to a *very wide* variety of fluid flows, the equations themselves are not universally applicable. The momentum equations are based on Newton's second law, which has been conclusively demonstrated to be an incomplete picture of our physical universe through Einstein's relativity theories. That said, relativistic effects seldom become even close to significant for nearly all conceivable flows that may be of interest. However, particularly for astrophysical flows, the effects are sometimes important.

A further physical approximation that is built into the Navier-Stokes equations is that the fluid behaves as a continuum. Hence, where discrete molecular effects are important, real fluids (made out of discrete molecules) cannot be modelled through the use of the Navier-Stokes equations. At microscopic and mesoscopic scales, these molecular effects are always important and need to be modelled via the use of alternative governing equations (e.g. the Boltzmann equation). Even at macroscopic scales, molecular effects are often important. For example, tracking the interface between a liquid and a gas (a molecular phenomena) is particularly challenging for a Navier-Stokes solver. One alternative to a Navier-Stokes solver is to use the lattice Boltzmann method, for example[1]. Through the use of the lattice Boltzmann method, the interface tracking is natural, and is particularly easy to achieve.

Despite these inadequacies in the Navier-Stokes equations, the equations are applicable to such a wide variety of different flow situations that they can safely be used as the governing equations of a general purpose CFD code.

The main attraction of CFD is that it can deal with the full non-linear form of the governing equations without resorting to any additional geometrical or physical approximations (other than those already built into the Navier-Stokes equations). In practice however, for turbulent flows physical approximations are (almost) invariably made. Using the full Navier-Stokes equations to fully resolve all turbulent scales (known as a direct numerical simulation (DNS)) is usually prohibitively expensive by todays computational standards. However, in theory, no physical approximations are necessary. Only numerical discretisation approximations need be made, which can be controlled to within the desired accuracy by using a sufficiently fine mesh (provided sufficient computational resources are available). CFD therefore enables the simulation of highly complex flows that could not be handled analytically.

CFD should not be thought of as a replacement for experiment, but rather as an accompaniment to experiment. It is still necessary to conduct some experiments to verify the CFD results. As

[1]The lattice Boltzmann method is a relatively new, and rapidly expanding branch of CFD which simulates macroscopic flow physics via the effects of multiple mesoscopic collisions between fictive particles. For further details of the method, the reader is referred to the very interesting review paper by Chen and Doolen, [34]

is encapsulated by Charles Babbage in the quote that follows, if one puts erroneous input into a computer (in this case, a CFD code), the output may well be far from physical. Experiments are therefore often used to provide boundary conditions at the edge of the computational domain that may otherwise be unknown.

> On two occasions I have been asked, - "Pray, Mr. Babbage, if you put into the machine wrong figures, will the right answers come out?"... I am not able rightly to apprehend the kind of confusion of ideas that could provoke such a question.
>
> **Charles Babbage**

In the sections that make up the remainder of this chapter, details of the finite volume discretisation technique shall be presented. The finite volume method is frequently used in CFD codes as it is arguably better suited than other discretisation methods within a CFD framework, primarily due to the rigorous enforcing of conservation over each grid cell. For this reason, the finite volume method was used in the present study. A solution algorithm that allows for solutions to the complete Navier-Stokes equations shall also be presented in the later sections of this chapter (namely the SIMPLE algorithm).

3.2 Physical approximations employed in the present code

While it is true that no physical approximations of the fluid flow are strictly necessary (other than those already built into the governing equations), some physical approximations are usually made in the interests of computational efficiency. For example, for a wide variety of flows, compressibility effects are negligible (i.e. the density of the fluid is near uniform throughout). Water, for example, has a compressibility of $4.6 \times 10^{-10} Pa^{-1}$ at a temperature of $299^{\circ}K$ and a pressure of $1bar$,[35]. This compressibility *decreases further* with increasing pressure, hence, for an increase in density of just 1%, the water would have to be pressurised to in excess of $20MPa$. To put this into perspective, this pressure is encountered at a water depth of roughly $2km$. For nearly all practical applications, the compressibility of water can therefore be neglected. The compressibility of other liquids is also generally so small that is can be neglected for practical flow simulations.

For gases, where compressibility is orders of magnitude higher than that of liquids, this is not necessarily the case. However, even for gases, if the Mach number is low, compressibility effects will be small. As a general rule of thumb, for Mach $M <\sim 0.3$, compressibility effects can usually be considered negligible.

For flows where the effects of compressibility are deemed negligible, the density of the fluid can be treated as a constant, and the pressure field is determined from the velocity field, via the use of a pressure-velocity coupling algorithm (as will be discussed further in Section 3.5), rather than being linked to the density and temperature fields. The use of the energy equation therefore becomes redundant (except in cases involving heat transfer, in which case the energy equation is solved for the temperature field, and the energy equation is decoupled from the other flow equations, or is only loosely coupled via a temperature dependent density). This incompressibility assumption therefore significantly reduces the complexity of the problem being dealt with. This assumption has been made in the present study, hence the code presented here is applicable only for incompressible flows, of which there are many.

A further physical assumption made in the present study is that the working fluid is a Newtonian fluid (that is, the viscous shear stress is proportional to the strain rate). For a Newtonian fluid, the viscous shear stress, τ_{ij}, is given by Equation 3.2. Note that, owing to the continuity condition for incompressible flows, the second term in Equation 3.2 is identically zero.

$$\tau_{ij} = \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) - \frac{2}{3}\mu \frac{\partial u_k}{\partial x_k}\delta_{ij} \tag{3.2}$$

Physical approximations to turbulence have also been made in the present study, as to fully resolve all turbulent scales in both space and time would be prohibitively expensive. The details of the physical modelling of turbulence that has been employed are sufficiently lengthy to warrant the use of a separate chapter, and are therefore postponed until Chapter 4.

One final physical assumption has been employed in the present code, which is that the velocity and pressure fields do not vary with time. Time derivatives are therefore zero, and the code is hence applicable only to steady-state problems.

3.3 The Finite Volume Method

The finite volume method commences by splitting up the overall domain into a number of smaller cells through the use of a grid. To demonstrate the method, a simple Cartesian grid shall initially be considered. The extension from a Cartesian grid to a general, non-orthogonal grid is straightforward in concept (but detailed in the workings) and shall be considered in a later section. The method shall be presented using the steady state version of Equation 3.1b as the governing equation, with the index i set equal to 1 (i.e. the u-momentum equation with time derivatives neglected). It is assumed, for the time being, that the pressure field p, and the v and w velocity components are known at each grid node in this analysis, and that one is solving for the unknown variable u (i.e. the governing equation is decoupled). Methods for coupling the momentum equations and the continuity equation will be dealt with in a later section. Throughout this thesis, the standard CFD nomenclature for referencing variable values at faces and nodes shall be used, as adopted by most CFD textbooks (see for example Reference [36]). Figure 3.1 illustrates the use of this nomenclature. Values at cell faces are denoted with lowercase 'n', 's', 'e', 'w', 't' and 'b' for the north, south, east, west, top and bottom faces respectively. Uppercase letters are used to denote values at surrounding cell centres, and the centre of the current cell is denoted as point P.

3.3.1 Basic Discretisation

In the finite volume method, the governing equation(s) is (are) integrated over each control volume. For the momentum equations, this gives;

$$\iiint_V \frac{\partial}{\partial x_j}\left[\rho u_i u_j\right] \cdot dxdydz = \iiint_V \frac{\partial}{\partial x_j}\left[-p\delta_{ij} + \tau_{ij}\right] \cdot dxdydz \tag{3.3}$$

In Equation 3.3, the left hand side represents convective fluxes, the first term of the right hand side is a source term arising from the pressure gradient, and the remainder of the right hand side represents the diffusive fluxes.

The volume integrals of the convective and diffusive flux terms are converted into surface integrals through the use of the divergence theorem, leading to the following (after substitution of expression 3.2 for the viscous shear stress, and summing over the repeated index);

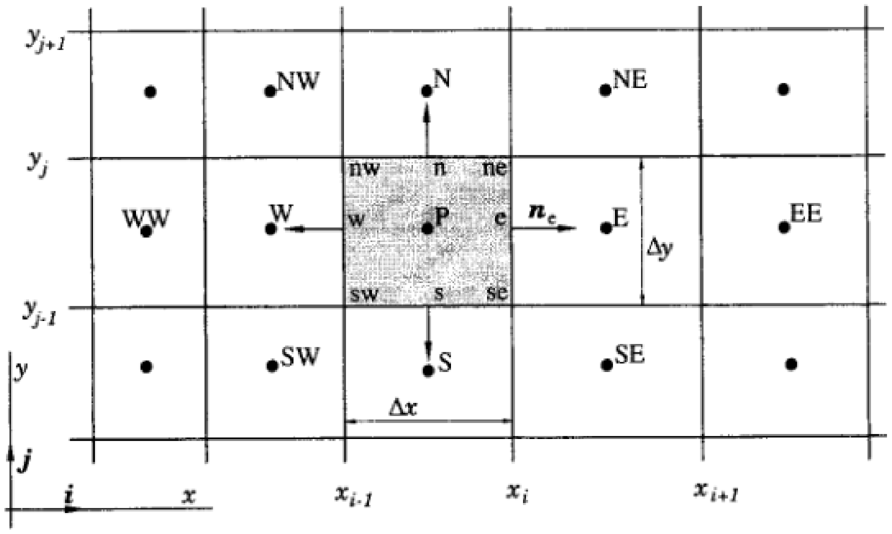


Figure 3.1: A typical three-dimensional control volume illustrating the standard CFD nomenclature used for referencing surrounding variables.

$$\left[\iint (\rho uu)\,dydz\right]_w^e + \left[\iint (\rho uv)\,dxdz\right]_s^n + \left[\iint (\rho uw)\,dxdy\right]_b^t = -\iiint_V \frac{\partial p}{\partial x} dxdydz + \\ \left[\iint \mu\frac{\partial u}{\partial x} dydz\right]_w^e + \left[\iint \mu\frac{\partial u}{\partial y} dxdz\right]_s^n + \left[\iint \mu\frac{\partial u}{\partial z} dxdy\right]_b^t + \\ \left[\iint \mu\frac{\partial u}{\partial x} dydz\right]_w^e + \left[\iint \mu\frac{\partial v}{\partial x} dxdz\right]_s^n + \left[\iint \mu\frac{\partial w}{\partial x} dxdy\right]_b^t \qquad (3.4)$$

To evaluate the surface integrals in Equation 3.4 exactly, it would be necessary to know the value of the integrand at all locations on the faces. However, this information is not available as only nodal values of variables are computed, hence discrete approximations have to be made. The surface integrals are first approximated in terms of values at a point or points on the cell's face, and these face values are interpolated from nodal values.

The midpoint rule provides a simple second order approximation to the surface integrals. In the mid-point rule, the mean value of the integrand is assumed to be equal to its value at the face centre. The integral is therefore taken as the integrand's value evaluated at the centre of the face, multiplied by the face area. Higher order integration schemes, such as the fourth-order Simpson's rule can be used but this complicates the code as the integrand (i.e. the flux) would need to be evaluated at multiple locations on the face. There is also little advantage to be gained from using higher-order schemes unless higher-order schemes are also used elsewhere in the code, since the overall order of the discritisation will be equal to the lowest order of each of the component schemes used. Due to the significant complexity involved in implementing these higher order schemes, the midpoint rule was used.

Using the midpoint rule, the convective flux terms (i.e. the first three terms of Equation 3.4) are approximated as follows:

$$\begin{aligned}
\left[\iint (\rho uu)\, dydz\right]_w^e + \left[\iint (\rho uv)\, dxdz\right]_s^n + \left[\iint (\rho uw)\, dxdy\right]_b^t& \\
\approx [\rho uu\Delta y\Delta z]_w^e + [\rho uv\Delta x\Delta z]_s^n + [\rho uw\Delta x\Delta y]_b^t& \\
= (\rho u_e\Delta y\Delta z)\, u_e - (\rho u_w\Delta y\Delta z)\, u_w +& \\
(\rho v_n\Delta x\Delta z)\, u_n - (\rho v_s\Delta x\Delta z)\, u_s +& \\
(\rho w_t\Delta x\Delta y)\, u_t - (\rho w_b\Delta x\Delta y)\, u_b& \\
= C_e u_e - C_w u_w + C_n u_n - C_s u_s + C_t u_t - C_b u_b& \qquad (3.5)
\end{aligned}$$

where the C's are the mass fluxes through the relevant faces. The values of u at the cell face centres are obtained through interpolation (details of suitable interpolation methods are presented in section 3.3.2). The mass fluxes (C) at the cell faces are typically evaluated through a central difference interpolation (linear averaging) from the two nodal values either side of the face (e.g. nodes P and E for the east face, and similarly for other faces).

Again using the midpoint rule, but this time applied to the diffusive flux terms, the following is obtained:

$$\begin{aligned}
&\left[\iint \mu\frac{\partial u}{\partial x}dydz\right]_w^e + \left[\iint \mu\frac{\partial u}{\partial y}dxdz\right]_s^n + \left[\iint \mu\frac{\partial u}{\partial z}dxdy\right]_b^t + \\
&\left[\iint \mu\frac{\partial u}{\partial x}dydz\right]_w^e + \left[\iint \mu\frac{\partial v}{\partial x}dxdz\right]_s^n + \left[\iint \mu\frac{\partial w}{\partial x}dxdy\right]_b^t \approx \\
&\left[\mu\frac{\partial u}{\partial x}\Delta y\Delta z\right]_w^e + \left[\mu\frac{\partial u}{\partial y}\Delta x\Delta z\right]_s^n + \left[\mu\frac{\partial u}{\partial z}\Delta x\Delta y\right]_b^t + \\
&\left[\mu\frac{\partial u}{\partial x}\Delta y\Delta z\right]_w^e + \left[\mu\frac{\partial v}{\partial x}\Delta x\Delta z\right]_s^n + \left[\mu\frac{\partial w}{\partial x}\Delta x\Delta y\right]_b^t \qquad (3.6)
\end{aligned}$$

In order to fully evaluate the diffusive flux terms on the right hand side of Equation 3.6, it is necessary to express the derivatives in terms of the values of the variables at the nodal locations. By assuming a *linear* variation of u between nodes P and E, the gradient on the faces can be evaluated using a central difference. This is a suitable approximation for diffusive fluxes as it is consistent with diffusion acting homogeneously in all directions since there is equal weighting applied to both the upstream and downstream nodes. Using central differences to approximate the gradients, the following expression for the diffusive fluxes is obtained:

$$
\begin{aligned}
&\left[\mu\frac{\partial u}{\partial x}\Delta y\Delta z\right]_w^e + \left[\mu\frac{\partial u}{\partial y}\Delta x\Delta z\right]_s^n + \left[\mu\frac{\partial u}{\partial z}\Delta x\Delta y\right]_b^t + \left[\mu\frac{\partial u}{\partial x}\Delta y\Delta z\right]_w^e + \left[\mu\frac{\partial v}{\partial x}\Delta x\Delta z\right]_s^n + \left[\mu\frac{\partial w}{\partial x}\Delta x\Delta y\right]_b^t \approx \\
&\qquad (2\mu\Delta y\Delta z)_e \left.\frac{u_E-u_P}{\Delta x}\right._e - (2\mu\Delta y\Delta z)_w \left.\frac{u_P-u_W}{\Delta x}\right._w + \\
&\qquad (\mu\Delta x\Delta z)_n \left.\frac{u_N-u_P}{\Delta y}\right._n - (\mu\Delta x\Delta z)_s \left.\frac{u_P-u_S}{\Delta y}\right._s + \\
&\qquad (\mu\Delta x\Delta y)_t \left.\frac{u_T-u_P}{\Delta z}\right._t - (\mu\Delta x\Delta y)_b \left.\frac{u_P-u_B}{\Delta z}\right._b + \\
&\qquad (\mu\Delta x\Delta z)_n \left.\frac{v_N-v_P}{\Delta y}\right._n - (\mu\Delta x\Delta z)_s \left.\frac{v_P-v_S}{\Delta y}\right._s + \\
&\qquad (\mu\Delta x\Delta y)_t \left.\frac{w_T-w_P}{\Delta z}\right._t - (\mu\Delta x\Delta y)_b \left.\frac{w_P-w_B}{\Delta z}\right._b
\end{aligned}
\tag{3.7}
$$

Up to this point, the convective and diffusive flux terms of Equation 3.3 have been discretised. What remains is the source term which requires a volume integration over the control volume. The volume integral is equal to the mean value of the integrand throughout the volume, multiplied by the cell's volume. However, the mean value of the integrand throughout the volume needs to be approximated since an exact evaluation would require the integrand to be known at all locations throughout the volume. The simplest method of achieving this is to assume the mean value of the integrand to be its value at the cell centre. This enables the source term to be approximated by the following;

$$
\begin{aligned}
-\iiint_V \frac{\partial p}{\partial x}dxdydz &\approx -\left(\frac{\partial p}{\partial x}\right)_P \Delta x\Delta y\Delta z \\
&\approx -\left(p_e - p_w\right)\Delta y\Delta z
\end{aligned}
\tag{3.8}
$$

Gathering the results of Equations 3.5, 3.7 and 3.8, the final form of the discretised u momentum equation is obtained, and is presented below as Equation 3.9.

$$
\begin{aligned}
C_e u_e - C_w u_w + C_n u_n - C_s u_s + C_t u_t - C_b u_b = &-\left(p_e - p_w\right)\Delta y \Delta z + \\
&\left(2\mu\Delta y\Delta z\right)_e \left.\frac{u_E - u_P}{\Delta x}\right._e - \left(2\mu\Delta y\Delta z\right)_w \left.\frac{u_P - u_W}{\Delta x}\right._w + \\
&\left(\mu\Delta x\Delta z\right)_n \left.\frac{u_N - u_P}{\Delta y}\right._n - \left(\mu\Delta x\Delta z\right)_s \left.\frac{u_P - u_S}{\Delta y}\right._s + \\
&\left(\mu\Delta x\Delta y\right)_t \left.\frac{u_T - u_P}{\Delta z}\right._t - \left(\mu\Delta x\Delta y\right)_b \left.\frac{u_P - u_B}{\Delta z}\right._b + \\
&\left(\mu\Delta x\Delta z\right)_n \left.\frac{v_N - v_P}{\Delta y}\right._n - \left(\mu\Delta x\Delta z\right)_s \left.\frac{v_P - v_S}{\Delta y}\right._s + \\
&\left(\mu\Delta x\Delta y\right)_t \left.\frac{w_T - w_P}{\Delta z}\right._t - \left(\mu\Delta x\Delta y\right)_b \left.\frac{w_P - w_B}{\Delta z}\right._b
\end{aligned}
\tag{3.9}
$$

3.3.2 Intra-Grid Interpolation Practices

Values of u at the cell face centres are required for the convection terms in equation 3.9. These are obtained through interpolation from nodal values. Numerous different interpolation methods are available, some of which shall be presented in this section.

Linear Interpolation: The Central Difference Scheme

To find the value of u on the east face (u_e), a linear variation of u from node E to node P can be assumed. This is equivalent to using a central difference scheme (CDS). At the east face, for a Cartesian grid, the CDS gives;

$$u_e = u_E \lambda + u_P \left(1 - \lambda\right)$$

where λ is the linear interpolation factor, defined as;

$$\lambda = \frac{x_e - x_P}{x_E - x_P}$$

To assess the accuracy of the CDS, a Taylor's series expansions can be conducted. By conducting a Taylor's series expansion for u_P about the point e, the following is obtained;

$$u_P = u_e + \left(\frac{\partial u}{\partial x}\right)_e (x_P - x_e) + \left(\frac{\partial^2 u}{\partial x^2}\right)_e \frac{(x_P - x_e)^2}{2} + \mathcal{O}\left(\Delta x^3\right) \tag{3.10}$$

A second Taylor series expansion, this time for u_E about the point e yields;

$$u_E = u_e + \left(\frac{\partial u}{\partial x}\right)_e (x_E - x_e) + \left(\frac{\partial^2 u}{\partial x^2}\right)_e \frac{(x_E - x_e)^2}{2} + \mathcal{O}\left(\Delta x^3\right) \tag{3.11}$$

Multiplying Equation 3.10 by $(1-\lambda)$ and 3.11 by λ, then summing the results yields;

$$u_e = u_E\lambda + u_P\,(1-\lambda) + \mathcal{O}\left(\Delta x^2\right) \tag{3.12}$$

In the CDS, only the first two terms of the right side of Equation 3.12 are retained. The leading truncation error term is therefore proportional to Δx^2. The CDS is hence second order accurate with respect to the grid spacing.

The second order accuracy of the CDS may seem desirable as a more accurate solution can be obtained with a coarser mesh (relative to a first order scheme). However, when the convection terms in the governing equations are large relative to the diffusion terms (as is the case for most flows, excluding those at very low Reynolds number), the coefficients of u in the discretised governing equation (Equation 3.9) may become negative, meaning an increase in u_E could lead to a decrease in u_P, for example. This is unphysical. The source of this erroneous result originates from the fact the convection process is directional, whereas the CDS applies equal weighting to both upstream and downstream nodes rather than giving greater weighting to upstream nodes. This can manifest itself in the form of unphysical solutions where the solution at node P may lie outside the results at the surrounding nodes (i.e. the solution may be unbounded). Around steep gradients, this is evident as under-shoots and over-shoots, or 'wiggles' that can be seen in contour plots. The central difference scheme has therefore not been used within the present study.

Upwind Difference Scheme

To overcome the transportive issues surrounding the CDS, a first order Upwind Difference Scheme (UDS) can be used. This scheme is unconditionally bounded. In the first order UDS, the value of u_e is taken as;

$$u_e = \begin{cases} u_P & \text{if } C_e > 0 \\ u_E & \text{if } C_e < 0 \end{cases}$$

By doing a Taylor's series expansion of u_P or u_E (depending on the flow direction) about the point e, the order of the scheme can be determined. The Talyor's series expansions have already been conducted in the evaluation of the order of the CDS, and were presented as Equations 3.10 and 3.11 respectively. From these equations, it can be seen that only the first term of the expansion is retained. The leading truncation error is therefore of order $O\left(\Delta x\right)$. Hence the scheme is first order accurate with respect to grid spacing.

Since the leading error term is a multiple of a first derivative, it acts to augment the viscous flux, which is also a multiple of a first derivative. The error is therefore described as 'numerical diffusion'. Viscosity tends to have the effect of damping out disturbances in the flow, and is stabilising. The UDS therefore achieves its stability by being diffusive in nature.

By giving greater weighting to the upstream node (all of the weighting in fact), the UDS provides stable, bounded results that are transportive, but the accuracy is not particularly attractive (first order). The UDS scheme is often used in the first few iterations of a CFD simulation where its stable nature enables a better estimate than the initialisation to be obtained. Once a reasonable first-order approximation to the flow field is obtained (which is presumably a much better approximation than the initial guessed values were), a higher order scheme can be switched to in order to obtain more accurate results.

QUICK - Quadratic Upwind Interpolation for Convective Kinematics

Rather than assuming a linear variation of u from E to P, Leonard, [37] proposed a quadratic variation. To fit a quadratic curve, three points are required to fully define the curve. When finding u_e, we have nodes P and E which come naturally. The third point is selected as EE if $C_e < 0$, or as W if $C_e > 0$. This gives the upwind bias that is required for the scheme to be transportive. The

QUICK scheme gives for the east face;

$$u_e = \begin{cases} u_P + g_1\left(u_E - u_P\right) + g_2\left(u_P - u_W\right) & \text{if } C_e > 0 \\ u_E + g_1\left(u_P - u_E\right) + g_2\left(u_E - u_{EE}\right) & \text{if } C_e < 0 \end{cases}$$

where g_1 and g_2 are geometric quantities, and are defined by:

$$g_1 = \begin{cases} \frac{(x_e - x_P)(x_e - x_W)}{(x_E - x_P)(x_E - x_W)} & (C_e > 0) \\ \frac{(x_e - x_E)(x_e - x_{EE})}{(x_P - x_E)(x_P - x_{EE})} & (C_e < 0) \end{cases}; \qquad g_2 = \begin{cases} \frac{(x_e - x_P)(x_E - x_e)}{(x_P - x_W)(x_E - x_W)} & (C_e > 0) \\ \frac{(x_e - x_E)(x_P - x_e)}{(x_E - x_{EE})(x_P - x_{EE})} & (C_e < 0) \end{cases}$$

To assess the accuracy of the QUICK scheme, Taylor's series expansions for u_P and u_E, in addition to either u_W or u_{EE} (depending on flow direction), can be conducted about the point e. The first two of these expansions have already been conducted as Equations 3.10 and 3.11 and are repeated below. For the third expansion, we shall assume a positive flow velocity, and hence conduct an expansion for u_W;

$$u_E = u_e + \left(\frac{\partial u}{\partial x}\right)_e (x_E - x_e) + \left(\frac{\partial^2 u}{\partial x^2}\right)_e \frac{(x_E - x_e)^2}{2} + \left(\frac{\partial^3 u}{\partial x^3}\right)_e \frac{(x_E - x_e)^3}{6} + \mathcal{O}\left(\Delta x^4\right) \tag{3.13}$$

$$u_P = u_e + \left(\frac{\partial u}{\partial x}\right)_e (x_P - x_e) + \left(\frac{\partial^2 u}{\partial x^2}\right)_e \frac{(x_P - x_e)^2}{2} + \left(\frac{\partial^3 u}{\partial x^3}\right)_e \frac{(x_P - x_e)^3}{6} + \mathcal{O}\left(\Delta x^4\right) \tag{3.14}$$

$$u_W = u_e + \left(\frac{\partial u}{\partial x}\right)_e (x_W - x_e) + \left(\frac{\partial^2 u}{\partial x^2}\right)_e \frac{(x_W - x_e)^2}{2} + \left(\frac{\partial^3 u}{\partial x^3}\right)_e \frac{(x_W - x_e)^3}{6} + \mathcal{O}\left(\Delta x^4\right) \tag{3.15}$$

For a uniform grid, the coefficients of the nodal values of u happen to be $\frac{3}{8}$, $\frac{6}{8}$ and $-\frac{1}{8}$ for the downstream node, the first upstream node and the second upstream node respectively. By multiplying Equation 3.13 by $\frac{3}{8}$, Equation 3.14 by $\frac{6}{8}$ and Equation 3.15 by $-\frac{1}{8}$, then summing the results, the following is obtained;

$$u_e = \frac{6}{8}u_P + \frac{3}{8}u_E - \frac{1}{8}u_W - \frac{3\left(\Delta x\right)^3}{48}\left(\frac{\partial^3 u}{\partial x^3}\right)_e + \mathcal{O}\left(\Delta x^4\right) \tag{3.16}$$

In the QUICK scheme, only the first three terms to the right of Equation 3.16 are retained. The leading error term is therefore proportional to $(\Delta x)^3$. The scheme is hence said to be third order accurate with respect to grid spacing.

Third order accuracy is sufficient for the majority of engineering problems. In fact, it makes little sense to use higher order schemes if the second order midpoint rule is used to discretise the integrals. Unless a higher order integration method is used, the overall accuracy will be limited to the accuracy of the integration scheme. As already mentioned, higher order integration schemes are complex to implement. Also, higher order interpolation schemes have large computational stencils. Hence the QUICK scheme, coupled with the midpoint rule, offers an attractive compromise between complexity and accuracy. It is also transportive as it has an upwind bias, and hence is more suitable than using a central difference scheme for general fluid flow problems.

UMIST Scheme

The main problem with the QUICK scheme is that the value of u_e may lie outside of the values at u_E and u_P. The value is said to be unbounded in such circumstances. This is particularly problematic when modelling turbulence, where the turbulent parameters (e.g. k and ϵ) are required to be positive. A negative value for such parameters would have no physical meaning.

If $u_{EE} > u_E > u_P$, or if $u_W < u_P < u_E$, u is said to be locally monotonic and QUICK interpolation is guaranteed to be bounded. The UMIST scheme therefore uses a variant of QUICK interpolation in this case. However, if u is not monotonic (e.g. if $u_W > u_P < u_E$), boundedness can not be assured. In such cases, the UMIST scheme defaults to the guaranteed bounded first-order upwind difference scheme.

Where the flow is monotonic, the third order QUICK scheme is used. The first-order scheme is only used where it is necessary to achieve boundedness. Hence, the UMIST scheme is mostly of third order accuracy, and it is always bounded. Lien and Leschziner (Ref. [38]) present further details of the scheme.

3.4 Solution Procedures

Now that the governing equation has been discretised, and that suitable intra-grid interpolation methods have been identified, the next problem is to actually solve the resulting set of algebraic equations (i.e. Equation 3.9). To assist with this matter, after approximating the cell face values of u via one of the above methods, Equation 3.9 can be rewritten as follows:

$$A_P u_P = \sum_m A_m u_m + S \tag{3.17}$$

where the A's are the coefficients associated with the nodal values of u, and S is the source term which contains all terms that cannot be absorbed into the A coefficients. The summation is conducted over all nodes that make up the computational stencil (e.g. nodes N, S, E, W, T and B).

In practice, the source term is usually linearised by splitting it into solution dependent and independent terms, denoted S_P and S_u respectively. S_P is then subtracted from the leading diagonal A_P in order to linearise the system and enhance convergence. To ensure diagonal dominance of the system is preserved, only *negative* contributions to S_P will be permitted, with any positive contributions being considered a part of S_u. This gives, for a general discritised transport equation;

$$\left(A_P - S_P\right) u_P = \sum_m A_m u_m + S_u \tag{3.18}$$

where S_P is the solution dependent source term ($S_P \leq 0$), and S_u is the solution independent source term. As an example, the values of the A's, S_P and S_u for the discretised u momentum equation, using first order upwind differencing, are presented below;

$$A_E = \max(0, -C_e) + \left(\frac{2\mu\Delta y\Delta z}{\Delta x}\right)_e \qquad A_W = \max(0, C_w) + \left(\frac{2\mu\Delta y\Delta z}{\Delta x}\right)_w$$
$$A_N = \max(0, -C_n) + \left(\frac{\mu\Delta x\Delta z}{\Delta y}\right)_n \qquad A_S = \max(0, C_s) + \left(\frac{\mu\Delta x\Delta z}{\Delta y}\right)_s$$
$$A_T = \max(0, -C_t) + \left(\frac{\mu\Delta x\Delta y}{\Delta z}\right)_t \qquad A_B = \max(0, C_b) + \left(\frac{\mu\Delta x\Delta y}{\Delta z}\right)_b$$

$$\begin{aligned}
A_P &= A_E + A_W + A_N + A_S + A_T + A_B \\
S_P &= 0 \\
S_u &= -(p_e - p_w)\Delta y\Delta z + \\
&\quad (\mu\Delta x\Delta z)_n \frac{v_N - v_P}{\Delta y} - (\mu\Delta x\Delta z)_s \frac{v_P - v_S}{\Delta y} + \\
&\quad (\mu\Delta x\Delta y)_t \frac{w_T - w_P}{\Delta z} - (\mu\Delta x\Delta y)_b \frac{w_P - w_B}{\Delta z}
\end{aligned}$$

Equation 3.18 is solved at each node that makes up the computational domain. Clearly, the values of u at surrounding nodes will not be known before the solution is obtained, so we are dealing with a linked system of equations. The system of equations can be written in matrix form which results in a large (number of nodes $\times$ number of nodes) but sparse matrix. The inversion of this matrix would yield the desired result, however, the coefficients, A, contain in them the unknown, u (e.g. the mass fluxes, C depend on the flow velocity, u). Also, the inversion of such a large matrix may be prohibitively expensive by todays computational standards. Hence iterative solution methods are used.

3.4.1 Gauss-Seidel Iterative Method

In the Gauss Seidel iterative method, values are updated one at a time using the values at previous iterations where necessary (or the initial guess values if at the first iteration). This is shown in Equation 3.19 for a seven point stencil:

$$u_P^{l+1} = \frac{A_E^l u_E^l + A_W^l u_W^l + A_S^l u_S^l + A_N^l u_N^l + A_T^l u_T^l + A_B^l u_B^l + S_u^l}{A_P^l - S_P^l} \tag{3.19}$$

where l is the iteration number. It can be shown that the number of iterations required to achieve convergence is proportional to the square of the number of nodes in one direction. This is not particularly efficient and can take a lot of computational effort for a large system.

3.4.2 Alternating Direction Implicit method

The Gauss-Seidel method presented above is very easy to implement but unfortunately converges rather slowly. One alternative, that is generally very efficient for structured grids, is the 'Alternating Direction Implicit' method (ADI). In the ADI method, a whole row or column is updated at once. A set of sweeps is first conducted along each row in the i direction (from west to east) giving a preliminary update, denoted as iteration number $l + \frac{1}{3}$. Following this, sweeps along each column in the j direction are conducted, giving the update at iteration $l + \frac{2}{3}$. Finally, a sweep in the k direction (from top to bottom) is performed, yielding the final updated values at iteration number $l+1$. Since a whole line is updated at once, and since there are alternating sweeps in each direction, information propagation throughout the domain can be achieved much quicker than when using the Gauss-Seidel method.

In the ADI method, algebraic equations of the following form are solved:

$$\left(A_P^l - S_P^l\right) u_P^{l+\frac{1}{3}} - A_E^l u_E^{l+\frac{1}{3}} - A_W^l u_W^{l+\frac{1}{3}} = A_S^l u_S^l + A_N^l u_N^l + A_T^l u_T^l + A_B^l u_B^l + S_u^l \tag{3.20}$$

where the terms on the left hand side are treated implicitly for the current sweep, while the terms on the right are treated explicitly. This gives the following tri-diagonal system of equations;

$$\begin{pmatrix} A^l_{P(1,j,k)} - S_P & A^l_{E(1,j,k)} & \cdots & 0 \\ A^l_{W(2,j,k)} & A^l_{P(2,j,k)} - S_P & \ddots & \vdots \\ \vdots & \ddots & \ddots & A^l_{E(Ni-1,j,k)} \\ 0 & \cdots & A^l_{W(Ni,j,k)} & A^l_{P(Ni,j,k)} - S_P \end{pmatrix} \cdot \begin{pmatrix} u^{l+\frac{1}{3}}_{(1,j,k)} \\ u^{l+\frac{1}{3}}_{(2,j,k)} \\ \vdots \\ u^{l+\frac{1}{3}}_{(Ni,j,k)} \end{pmatrix} = \begin{pmatrix} Q^l_{(1,j,k)} \\ Q^l_{(2,j,k)} \\ \vdots \\ Q^l_{(Ni,j,k)} \end{pmatrix} \tag{3.21}$$

where $Q = (S_u + A_N u_N + A_S u_S + A_T u_T + A_B u_B)$. Tri-diagonal systems of equations, such as that of Equation 3.21 can be solved using the Tri-Diagonal Matrix Algorithm (TDMA). The TDMA is a simplified version of Gaussian Elimination that removes generality in favour of high efficiency. The result is a highly efficient algorithm that is only applicable to tri-diagonal systems. This is particularly well suited to the solution of the systems of equations that are found in CFD when using a structured grid. In Section 3.4.4, implementation details of the TDMA method are presented.

Once the TDMA has been used to find all values of $u^{l+\frac{1}{3}}$ along the rows via 3.21, the sweep direction is switched to the north-south direction say. For this sweep, the discretised equation at any point is written as:

$$\left(A^l_P - S^l_P\right) u_P^{l+\frac{2}{3}} - A^l_N u_N^{l+\frac{2}{3}} - A^l_S u_S^{l+\frac{2}{3}} = A^l_E u_E^{l+\frac{1}{3}} + A^l_W u_W^{l+\frac{1}{3}} + A^l_T u_T^{l+\frac{1}{3}} + A^l_B u_B^{l+\frac{1}{3}} + S^l_u \tag{3.22}$$

which gives a system similar to that of Equation 3.21. The system is again solved using the TDMA and the procedure is repeated for each column. Finally, a sweep from top to bottom is performed by treating u_P, u_T and u_B implicitly, with all other stencil contributions being treated explicitly. Once all rows and columns have been updated, the updated value of u^{l+1} will have been obtained at each node. The whole procedure is then repeated until convergence, which can be assessed via Equation 3.23.

It can be shown that the number of iterations required to achieve convergence is proportional to the number of nodes in one direction. This is significantly faster than the Gauss-Seidel method for large systems.

3.4.3 Convergence

Convergence of an iterative method can be assessed by examining the root-mean-squared (RMS) value of the residuals at each node in the domain. The RMS of the residuals is given by:

$$\text{RMS} = \left[\sum_{\text{all nodes}} \left((A_P - S_P)u_P - \sum_m A_m u_m - S_u \right)^2 \right]^{0.5} \tag{3.23}$$

Once the RMS residual error value has fallen to a sufficiently low level, the iterative procedure is stopped and convergence may be assumed. The specific RMS level at which convergence can be assumed is problem dependent, but generally one should aim to reduce the residuals by several orders of magnitude from their initial values at $t = 0$. Performing further iterations on a converged solution should have negligible effect on the solution. The final solution should therefore be unique, and independent of the number of iterations conducted.

3.4.4 The Tri-Diagonal Matrix Algorithm

For the tri-diagonal system

$$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i \qquad (i = 1, 2, \ldots, n)$$

the first step is to compute modified coefficients c'_i and d'_i, which are based on the original coefficients as follows;

$$c'_i = \begin{cases} \frac{c_i}{b_i} & i = 1 \\ \frac{c_i}{b_i - c'_{i-1} a_i} & i = 2, 3, \ldots, n \end{cases}$$

$$d'_i = \begin{cases} \frac{d_i}{b_i} & i = 1 \\ \frac{d_i - d'_{i-1} a_i}{b_i - c'_{i-1} a_i} & i = 2, 3, \ldots, n \end{cases}$$

The solution is then obtained from

$$\begin{aligned} x_n &= d'_n \\ x_i &= d'_i - c'_i x_{i+1} \qquad\qquad i = n-1, n-2, \ldots, 1 \end{aligned} \tag{3.24}$$

For derivation details of the algorithm, the reader is referred to the original article by Thomas [3], or to the more modern textbook by Versteeg *et. al.* [39].

3.4.5 Coupled Systems of Equations

Up to this point, a single, de-coupled equation has been used as the governing equation. It has been assumed that all variables in the discretised u momentum equation, excluding u itself, are somehow known. In reality, the Navier-Stokes equations that govern the flow are coupled, and clearly all flow variables need be computed. When solving the coupled Navier-Stokes equations using one of the iterative methods discussed above, there is a choice of solving the system simultaneously or in a segregated manner. In the former method, all equations are considered as part of a single system. Since all equations are solved at the same time, there is a strong coupling between the momentum and continuity equations which can increase convergence rates. However, the side effect of solving all equations simultaneously is that there is a significant increase in required computational memory (RAM), owing to the fact that all equation coefficients are required at the same time, so a single array cannot simply be overwritten. For this reason, such simultaneous solution methods shall not be considered here. Stability is also often impaired.

In the segregated method, each discretised governing equation is solved sequentially. Each equation is treated as having only one unknown variable, with other variables taking their value from the last iteration. The first variable is updated, separate from the rest. This is then followed by updating the second variable, and so on. The variables are updated iteratively using the ADI method, for example, in what is known as an *inner*-iteration. Once *all* variables have been updated in sequence, one *outer*-iteration is said to have been conducted. As many outer-iterations are carried out as are necessary to achieve convergence. Convergence is assumed when the RMS values of all the residuals fall below a pre-defined tolerance for all variables (see Equation 3.23). The tolerance may be different for each variable, if appropriate. Typically, *one* inner-iteration of each momentum equation is conducted, followed by 3-8 (say) inner-iterations of the pressure correction equation, for each outer iteration. However, the specific number of inner-iterations for each equation may depart from these typical values if desired (and convergence rates can sometimes be improved by doing so).

3.4.6 Under-Relaxation

When considering a system of equations that are coupled, a change in a variable through an inner-iteration will yield a change in the coefficients of other variables. If this change is large, instabilities can occur. It is therefore usually necessary to limit the magnitude of the changes through the use of under-relaxation.

The method of under-relaxation that is used is not dependent on the iterative scheme. By generalising the iterative schemes discussed above, one can arrive at the general equation given below, applicable to any of the iterative schemes mentioned, and for any of the discritisation practices discussed;

$$\left(A_P^l - S_P^l\right)\phi_P^{\mathrm{NEW}} - \sum_m A_m^l \phi_m^{l+1} = \sum_{b \neq m} A_b^l \phi_b^l + S_u^l \tag{3.25}$$

where the summation over m is conducted over each node that is being treated implicitly, and the summation over b is conducted over each remaining node that makes up the computational stencil that is being treated explicitly.

The value ϕ^{NEW} is the result of performing an inner-iteration on the discretised governing equation with no under-relaxation, but this value is not used to update to ϕ^{l+1}. Instead, the updated value of ϕ is given by;

$$\begin{aligned}\phi^{l+1} &= \phi^l + \alpha\left(\phi^{\mathrm{NEW}} - \phi^l\right) \\ &= \alpha\phi^{\mathrm{NEW}} + (1-\alpha)\,\phi^l \end{aligned} \tag{3.26}$$

where α is the under-relaxation factor, and has a value $0 < \alpha < 1$.

Substituting ϕ^{NEW} from Equation 3.25 into Equation 3.26 yields;

$$\phi_P^{l+1} = \alpha \left[\frac{\sum_{b \neq m} A_b^l \phi_b^l + S_u^l + \sum_m A_m^l \phi_m^{l+1}}{A_P^l - S_P^l} \right] + (1-\alpha)\,\phi_P^l \tag{3.27}$$

which upon rearranging gives us the following:

$$\underbrace{\frac{A_P^l - S_P^l}{\alpha}}_{A_P^*} \phi_P^{l+1} - \sum_m A_m^l \phi_m^{l+1} = \underbrace{\sum_{b \neq m} A_b^l \phi_b^l + S_u^l + \frac{(1-\alpha)}{\alpha} A_P^l \phi_P^l}_{S_u^*} \tag{3.28}$$

In Equation 3.28, the modified coefficient and source terms are denoted A_P^* and S_u^* respectively. It can be seen that, since α is set to a value $0 < \alpha < 1$, the modified coefficient A_P^* is greater than the original coefficient A_P. The effect of under-relaxation is therefore to increase the diagonal dominance of the iterative scheme, which increases the rate of convergence (relative to solving without this implicit under-relaxation and then updating ϕ explicitly through Equation 3.26). Once a sufficient number of iterations have been conducted, and convergence is reached, the value of ϕ_P^{l+1} will equal ϕ_P^l, hence α in equation 3.28 will cancel, and the scheme will converge to the solution of the original problem, irrespective of the value of α chosen (assuming convergence is actually achieved).

The optimum value of the under-relaxation factor is often different for different governing equations and for different problems. Typically, the momentum equations may be solved using a relatively high value of α, (~ 0.7 for example), while the particularly sensitive pressure correction equation requires more significant under-relaxation (e.g. ~ 0.2). However, specific values of α are problem dependent. To speed up the rate of convergence, it is desirable to set the under-relaxation factor to as high a value as is possible, but not so high that instabilities occur. The selection of suitable values of α is more an art form than an exact science. Successful selections often come as a result of experience, or through trial and error, rather than through analytic analysis. One strategy that works well is to use low values of α for early iterations, where the solution generally changes rapidly, then to gradually increase α, a little at a time, until convergence.

3.4.7 Deferred Correction Approach

A sufficient (but not necessary) condition for a *bounded* solution is that the coefficients A in Equation 3.18 are all positive, [40]. This is also a sufficient condition for convergence of the tri-diagonal matrix solver (see Section 'refsec:TDMA). However, the only convection scheme that guarantees these positive coefficients is the first order upwind scheme (UDS), described in Section 3.3.2.

It has already been stated that the first order accuracy of the UDS is unattractive. A method of implementing higher order schemes, such as QUICK, while also avoiding these convergence issues is therefore sought. A solution is the deferred correction approach.

In the deferred correction approach, the coefficients resulting from application of the first order upwind scheme for convective fluxes are computed, and will be referred to as A_P^{UDS}, A_E^{UDS}, etc. Similarly, the coefficients resulting from the application of the desired higher order scheme, such as QUICK, are also computed, and will be referred to as A_P^{HOS}, A_E^{HOS}, etc. (where the acronym HOS stands for Higher Order Scheme).

The UDS contributions are then treated implicitly, with the difference between the UDS and HOS contributions being treated explicitly via addition to the source term, giving ;

$$\left(A_P^{UDS} - S_P\right) u_P - \sum_m A_m^{UDS} u_m = \left(A_P^{UDS} - A_P^{HOS}\right) u_P - \sum_m \left(A_m^{UDS} - A_m^{HOS}\right) u_m + S_u \quad (3.29)$$

where the terms on the left hand side are treated implicitly, while terms on the right are treated explicitly. Here, the term, S_u represents the portion of source that excludes the difference between the UDS and HOS contributions. In practice, all the terms on the right are absorbed into the source term S_u, however they are typed out fully here to demonstrate the deferred correction approach.

Note that in practice, some of the terms on the left may be transferred to the right (i.e. treated explicitly) if the specific iterative algorithm dictates this. For example, for the Gauss-Seidel iterative method, the terms $\sum_m A_m^{UDS} u_m$ are always treated explicitly.

Since only the UDS coefficients are passed on to the tri-diagonal matrix algorithm, or other iterative solver, stability issues are avoided. Also, at convergence, the UDS terms will cancel out, and the resulting solution will be equal to that obtained via the use of the HOS without any deferred correction.

3.5 The SIMPLE Algorithm

Fluid flows must satisfy the conservation of momentum *and* mass. For incompressible flows, the momentum equations provide a link between velocity and pressure, and the continuity equation dictates a restriction on the velocity field that ensures mass conservation. There is however, no direct link between pressure and mass conservation in the incompressible Navier-Stokes equations. It is this link that is required to couple the system. By coupling the system, the conservation of momentum and mass can simultaneously be achieved, thereby yielding physical results.

Logical thinking shows there clearly is a link between pressure and the conservation of mass for incompressible flows, as follows:

- An increase in a cell's pressure, relative to that of surrounding cells, will 'push' fluid out of the cell, resulting in a decrease in the net mass flux into the cell.

- Conversely, a decrease in a cell's pressure, relative to surrounding cells, will 'pull' fluid into the cell, resulting in an increase in the net mass flux into the cell.

There is a need to quantify the link expressed above for the use in practical calculations. This can be achieved through a number of different iterative schemes, each of which enable the solution to the complete incompressible Navier-Stokes equations to be obtained. This section shall outline one such method; namely the SIMPLE algorithm, which was developed by Patankar and Spalding [41]. In this project, the SIMPLE algorithm has been applied to *colocated* grids (all variables are stored at the same nodes) rather than staggered grids (where different grids are used for u, v, w and p), as the former method simplifies the gridding, particularly of complex geometries.

To commence, the momentum equations are initially solved using a guessed value of the pressure field (or the value from the previous iteration) to yield a preliminary velocity field. However, in general the resulting velocity field will not satisfy the conservation of mass. In an attempt to conserve mass *and* momentum simultaneously, it is supposed that there are corrections, u', v', w' and p' that can be added to the original values obtained, such that the corrected variables satisfy both momentum and mass. The corrected variables are then given by;

$$p^{l+1} = p^l + p' \qquad u^{l+1} = u^* + u' \qquad v^{l+1} = v^* + v' \qquad w^{l+1} = w^* + w' \tag{3.30}$$

where the superscript $*$ denotes the preliminary velocity field found from an inner-iteration of the momentum equations, with a guess of the pressure field. The values at iteration $l+1$ are the updated, mass conserving values at the end of an outer-iteration. A prime superscript denotes the correction that is to be applied.

To illustrate how the values of the corrections are obtained, consider the steady state continuity equation;

$$\frac{\partial \rho u}{\partial x} + \frac{\partial \rho v}{\partial y} + \frac{\partial \rho w}{\partial z} = 0 \tag{3.31}$$

Integrating Equation 3.31 over the control volume, then discretising using the finite volume method (as discussed in Section 3.3.1) yields the following;

$$\rho \left(u_e^{l+1} - u_w^{l+1}\right) \Delta y \Delta z + \rho \left(v_n^{l+1} - v_s^{l+1}\right) \Delta x \Delta z + \rho \left(w_t^{l+1} - w_b^{l+1}\right) \Delta x \Delta y = 0 \tag{3.32}$$

Equation 3.32 imposes a restriction on the velocity field that must be satisfied by the end of each outer-iteration. By substituting the corrections (i.e. Equations 3.30) into the continuity equation, the following is obtained;

$$\rho \left(u_e' - u_w'\right) \Delta y \Delta z + \rho \left(v_n' - v_s'\right) \Delta x \Delta z + \rho \left(w_t' - w_b'\right) \Delta x \Delta y = -\dot{m}^* \tag{3.33}$$

where $\dot{m}^*$ is the net mass imbalance arising from the preliminary velocity field that we wish to drive to zero, and is defined by the following;

$$\dot{m}^* \equiv \rho \left[\left(u_e^* - u_w^*\right) \Delta y \Delta z + \left(v_n^* - v_s^*\right) \Delta x \Delta z + \left(w_t^* - w_b^*\right) \Delta x \Delta y\right] \tag{3.34}$$

The objective of the SIMPLE algorithm is to express the discrete continuity condition (Equation 3.33) in terms of a restriction to the pressure field (rather than on the velocity field), so that the pressure can be adjusted to yield zero mass imbalance. To achieve this, it is first imagined that the momentum equations are solved at the faces of the cell. The discretised form of these equations is presented below;

$$u_f^{l+1} = \frac{1}{A_{P_f}} \left[\sum_m A_m u_m^{l+1} - \left(p_E^{l+1} - p_W^{l+1} \right) \Delta y \Delta z + S_u^l \right] \tag{3.35a}$$

$$v_f^{l+1} = \frac{1}{A_{P_f}} \left[\sum_m A_m v_m^{l+1} - \left(p_N^{l+1} - p_S^{l+1} \right) \Delta x \Delta z + S_u^l \right] \tag{3.35b}$$

$$w_f^{l+1} = \frac{1}{A_{P_f}} \left[\sum_m A_m w_m^{l+1} - \left(p_T^{l+1} - p_B^{l+1} \right) \Delta x \Delta y + S_u^l \right] \tag{3.35c}$$

where the subscript f denotes a face, and the summation is conducted over the imaginary nodes surrounding the node under consideration. In Equations 3.35, S_u is the source term with the pressure source contributions removed.

By substituting the corrections (i.e. Equations 3.30) into the momentum equations (Equations 3.35), and then subtracting the original momentum equations (evaluated with the guessed pressure field) from the result, the following expressions are obtained;

$$u'_f = \frac{1}{A_{P_f}} \left[\sum_m A_m u'_m - \left(p'_E - p'_W \right) \Delta y \Delta z \right] \tag{3.36a}$$

$$v'_f = \frac{1}{A_{P_f}} \left[\sum_m A_m v'_m - \left(p'_N - p'_S \right) \Delta x \Delta z \right] \tag{3.36b}$$

$$w'_f = \frac{1}{A_{P_f}} \left[\sum_m A_m w'_m - \left(p'_T - p'_B \right) \Delta x \Delta y \right] \tag{3.36c}$$

Equations 3.36 provide a linkage between the velocity corrections at the cell faces, and the pressure corrections. These links can be substituted into the continuity equation. However, in Equations 3.36, the summations over surrounding faces would bring in to the continuity equation values of u', v' and w' at surrounding locations. These values at the surrounding locations would hence need to be evaluated. This would also involve evaluating u', v' and w' at locations that surround the original surrounding locations, and so on. The solution of the continuity equation at node P would hence require knowledge of u', v', and w' at each face throughout the domain. This is clearly impractical so the summations in the above equations are neglected, in what is referred to as the SIMPLE approximation. This simplification is justifiable, since at convergence, the corrections will be zero anyway (i.e. $u' = 0$, $v' = 0$ and $w' = 0$ at convergence, at all locations), so the simplification does not alter the final converged result.

By making the SIMPLE approximation, and substituting Equations 3.36 into Equation 3.33 yields;

$$\begin{aligned}
&\rho\left[\frac{1}{A_{P_e}}\left(p'_P - p'_E\right) - \frac{1}{A_{P_w}}\left(p'_W - p'_P\right)\right]\left(\Delta y \Delta z\right)^2 \\
&+\rho\left[\frac{1}{A_{P_n}}\left(p'_P - p'_N\right) - \frac{1}{A_{P_s}}\left(p'_S - p'_P\right)\right]\left(\Delta x \Delta z\right)^2 \\
&+\rho\left[\frac{1}{A_{P_t}}\left(p'_P - p'_T\right) - \frac{1}{A_{P_b}}\left(p'_B - p'_P\right)\right]\left(\Delta x \Delta y\right)^2 = -\dot{m}^*
\end{aligned} \tag{3.37}$$

which is an equation for the pressure corrections. Equation 3.37 can be recast as follows:

$$a_P p'_P - a_E p'_E - a_W p'_W - a_N p'_N - a_S p'_S - a_T p'_T - a_B p'_B = -\dot{m}^* \tag{3.38}$$

where the a's are the coefficients of $p^{'}$ in the pressure correction equation, and are defined below;

$$\begin{aligned}
a_E &= \rho\frac{1}{A_{P_e}}\left(\Delta y \Delta z\right)^2 & a_W &= \rho\frac{1}{A_{P_w}}\left(\Delta y \Delta z\right)^2 \\
a_N &= \rho\frac{1}{A_{P_n}}\left(\Delta x \Delta z\right)^2 & a_S &= \rho\frac{1}{A_{P_s}}\left(\Delta x \Delta z\right)^2 \\
a_T &= \rho\frac{1}{A_{P_t}}\left(\Delta x \Delta y\right)^2 & a_B &= \rho\frac{1}{A_{P_b}}\left(\Delta x \Delta y\right)^2
\end{aligned}$$

$$a_P = a_E + a_W + a_N + a_S + a_T + a_B$$

Note that uppercase A denotes the diagonal coefficients of the discretised momentum equations, while lowercase a are the coefficients of the pressure correction equation. The coefficients A_{P_f} which are contained within the a's, are evaluated through linear interpolation to obtain their face values from surrounding nodal values. It can be seen that Equation 3.38 is of the same form as that of the discretised momentum equations. The same iterative solver can therefore be used for both sets of equations. Typically the pressure correction benefits from performing several inner-iterations (e.g. 3-8) for each outer iteration. This is due to the fact that the pressure correction equation is reinitialised to zero at the beginning of each outer iteration.

Once the correction to the pressure has been computed, the pressure and the velocity components are updated via the following;

$$p^{l+1}_{(i,j,k)} = p^{l}_{(i,j,k)} + \alpha_p p'_{(i,j,k)} \tag{3.39a}$$

$$u^{l+1}_P = u^*_P + \frac{1}{A_P}\left(p'_w - p'_e\right)\Delta y \Delta z \tag{3.39b}$$

$$v^{l+1}_P = v^*_P + \frac{1}{A_P}\left(p'_s - p'_n\right)\Delta x \Delta z \tag{3.39c}$$

$$w^{l+1}_P = w^*_P + \frac{1}{A_P}\left(p'_b - p'_t\right)\Delta x \Delta y \tag{3.39d}$$

3.5.1 Pressure ‘Checkerboarding’

In the SIMPLE algorithm, the current mass imbalance ($\dot{m}^*$) needs to be evaluated. It has been shown that the mass imbalance is derived from the continuity equation, and is defined by the following;

$$\dot{m}^* \equiv \rho\left(u^*_e - u^*_w\right)\Delta y \Delta z + \rho\left(v^*_n - v^*_s\right)\Delta x \Delta z + \rho\left(w^*_t - w^*_b\right)\Delta x \Delta y \tag{3.40}$$

The question is, how are the face velocities in 3.40 evaluated? Using liner interpolation may seem the obvious answer, which for a uniform Cartesian grid, yields;

$$u^*_e = \frac{u^*_E + u^*_P}{2} \qquad u^*_w = \frac{u^*_W + u^*_P}{2}$$
$$v^*_n = \frac{v^*_N + v^*_P}{2} \qquad v^*_s = \frac{v^*_S + v^*_P}{2}$$
$$w^*_t = \frac{w^*_T + w^*_P}{2} \qquad w^*_b = \frac{w^*_B + w^*_P}{2}$$

Substituting the face velocities into the expression for the mass imbalance (Equation 3.40) yields;

$$\dot{m}^* = \rho\left(\frac{u_E^* - u_W^*}{2}\right)\Delta y\Delta z + \rho\left(\frac{v_N^* - v_S^*}{2}\right)\Delta x\Delta z + \rho\left(\frac{w_T^* - w_B^*}{2}\right)\Delta x\Delta y \qquad (3.41)$$

Equation 3.41 can then be used in the SIMPLE algorithm to complete as many outer-iterations as are required for convergence. However, from the discretised u momentum equation, it can be noted that the velocity u_E^* is dependent on the pressure at nodes P and EE (with the contribution at node E cancelling after linear interpolation for the pressure). Similarly, the velocity u_W^* is dependent on the pressure at nodes P and WW. Hence, the term $u_E^* - u_W^*$ in Equation 3.41 involves pressure at nodes WW, P and EE, but not at nodes E or W. A similar result is also found for the terms $v_N^* - v_S^*$ and $w_T^* - w_B^*$. This leads to a situation where the pressure at even nodes are coupled with each other, but not with the odd nodes, and vice versa. The resulting pressure field, in one-dimension, may take a variation such as that of Figure 3.2, for example, and still satisfy the discretised Navier-Stokes equations. In two dimensions, a 'checkerboard' style pressure variation may be obtained, where the black squares of a checkboard are analogous to the coupled 'odd' nodes, which are decoupled from the white squares, which are analogous to the 'even' nodes.

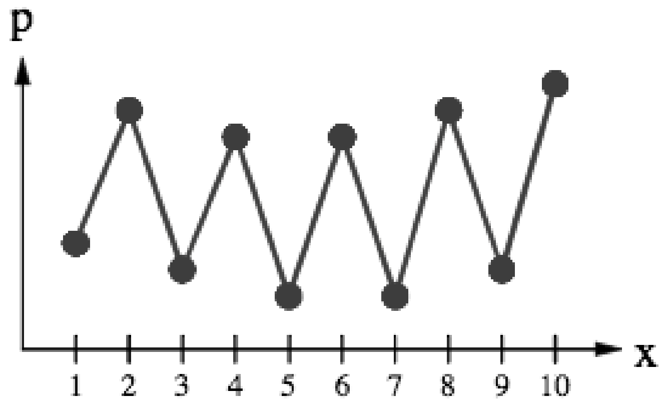


Figure 3.2: Odd-Even decoupling of the pressure field which may be found when using colocated variables, with linear interpolation to determine the mass imbalance.

Such a pressure variation is clearly unphysical; it is an artifact of the discretisation process rather than a feature of the Navier-Stokes equations. The following sections outline two different remedies to solve this 'checkerboarding' problem.

A solution: The Staggered Grid

To rectify the checkerboarding problem, a staggered grid can be used. In a staggered grid, velocity components are stored half way between the pressure nodes that drive them. This leads to three different meshes being required in 2D flows, or four meshes for 3D flow. Such a staggered grid arrangement is illustrated in Figure 3.3.

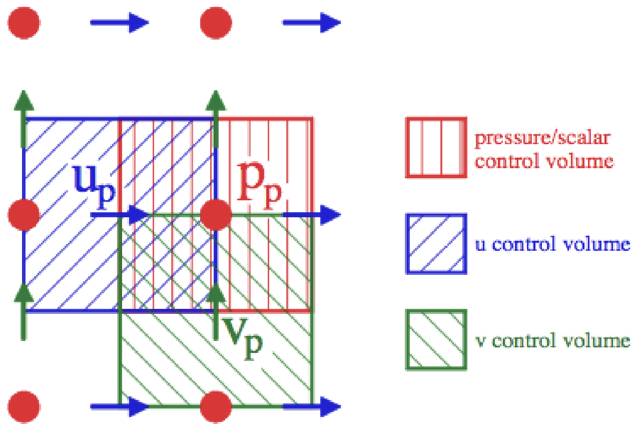


Figure 3.3: A staggered grid arrangement.

The advantages of using a staggered grid arrangement are;

- The pressure is now stored at the location required by the momentum equations. No interpolation is required.
- The velocities are now stored at the locations required to compute the mass imbalance. Again, without interpolation.

Because of this, when computing the mass imbalance, there is no odd-even de-coupling to the pressure, hence no checkerboarding is encountered. However, the generation of separate meshes for the different variables requires significant effort, particularly when using non-orthogonal grids. For this reason, the staggered grid is not always a practical solution, especially in 3D and for complex geometries.

Rhie-Chow Interpolation

Rather than using a staggered grid, it is desirable to use a colocated grid to simplify the grid generation problem. To overcome checkerboarding issues, an alternative interpolation method can be used to evaluate the face velocities when computing the mass imbalance. Rhie and Chow proposed a non-linear interpolation scheme that ensures the velocities at the faces are linked to the pressure at surrounding nodes, [42]. Their idea involves taking a central difference of the velocity, and then subtracting a central pressure difference, discretised at the nodal locations and averaged (linearly interpolated) onto the face. Finally the central pressure difference, discretised at the face is added to the result. As an example, the Rhie-Chow interpolation for u_e^* is given by

$$u_e^* = \overline{u^* - \frac{\Delta y \Delta z}{A_P}\left(p^l_{i-1/2,j,k} - p^l_{i+1/2,j,k}\right)}_e + \overline{\frac{\Delta y \Delta z}{A_P}}_e \left(p^l_P - p^l_E\right) \tag{3.42}$$

where an overbar indicates linear interpolation from the neighbouring nodes to the face. For a uniform, Cartesian, grid this gives;

$$\begin{aligned} u_e^* &= 0.5\left[u_P^* + u_E^*\right] \\ &+ 0.5\left[\left(\frac{\Delta y \Delta z}{A_P}\right)_{(i,j,k)} \frac{\left(p^l_W - p^l_E\right)}{2} + \left(\frac{\Delta y \Delta z}{A_P}\right)_{(i+1,j,k)} \frac{\left(p^l_P - p^l_{EE}\right)}{2}\right] \\ &+ 0.5\left[\left(\frac{\Delta y \Delta z}{A_P}\right)_{(i,j,k)} + \left(\frac{\Delta y \Delta z}{A_P}\right)_{(i+1,j,k)}\right]\left(p^l_P - p^l_E\right) \end{aligned} \tag{3.43}$$

As the grid size tends to zero, the second and third lines of Equation 3.43 cancel, hence the discretised approximation is equivalent to the continuum value of u in this limit (i.e. the interpolation scheme is consistent). Lines two and three of Equation 3.43 are a pressure smoothing term, which eliminates the checkerboarding that may be found when not using any pressure smoothing. The checkerboarding is avoided due to the fact that the face velocities now incorporate pressure from both odd *and* even nodes, so there is no decoupling. It can be shown that the pressure smoothing term is equivalent to an artificial dissipation that is of third order.

3.6 Boundary Conditions

The same governing equations can be used to simulate a wide variety of *different* flow conditions, even if the same computational mesh is used. For a steady state solution, one of the driving factors that makes the solution unique is the *boundary conditions.* It is hence of critical importance to treat the boundary conditions in an appropriate numerical way.

The most common types of boundary conditions are;

- Dirichlet boundary conditions (i.e. ϕ specified)
- Neumann boundary conditions (i.e. $\frac{\partial \phi}{\partial n}$ specified)

Both types of boundary conditions can be implemented by transferring the boundary flux to the source term. Hence, the coefficient A in the boundary direction is set to zero (in order to 'cut' the boundary node from the solution), and then the boundary flux is added to the source. Is is usual to store the boundary values at extra nodes located at the boundary faces, so that the same methods of computing fluxes and coefficients as for internal nodes can be used, hence simplifying the computational code.

The following physical boundary conditions are frequently encountered in CFD simulations;

- At a wall, the no-slip condition applies (assuming a non-zero fluid viscosity). In other words, the velocity of the fluid is equal to the velocity of the wall (a Dirichlet boundary condition).
- At an inlet, the fluid velocity is normally specified (a Dirichlet condition). The pressure is normally extrapolated from the interior of the domain.
- At an outlet, the flow variables are usually not known. However, it is usually the case that the outlet is set far enough downstream that the flow is not changing significantly in the downstream direction. In this case, $\frac{\partial \mathbf{v}}{\partial n} \approx 0$. By discretising this relation, it can be seen that this is the equivalent of setting u_b, v_b and w_b to their values at the adjacent node. A zero gradient is not applicable for the pressure since a pressure gradient may be present no matter how far downstream the outlet is. For example, in a pipe flow, it is this pressure gradient that is driving the flow. The pressure can therefore be extrapolated from the interior. A linear extrapolation may be used for this purpose. An alternative for internal flows, where the outlet boundary values should be set such that global mass conservation is achieved will be described in Section 3.6.1.

The pressure correction equation is usually solved with the boundary condition $\frac{\partial p'}{\partial \mathbf{n}} = 0$. This is found from Equations 3.36, where the velocity at the boundary is taken as known for each inner-iteration of the pressure correction equation, hence $u' = v' = 0$. Making the SIMPLE approximation then gives $p'_f = p'_P$. Alternatively, if the pressure is known at the boundary, the correction must be set to zero there. For interpolation cells, the pressure correction is interpolated since the correction should be the same as at the point the pressure was interpolated from. This follows from the fact that if the pressure changes on one subgrid, the pressure at the same location on another overlapping subgrid should change by a corresponding amount. There cannot be separate pressure fields covering the same physical space.

3.6.1 Bulk Correction Methods

For internal flows where the flow is driven by a pressure difference between the inlet and outlet (such as the flow through a channel or pipe), it is important that the mass entering the domain through the inlet is exactly balanced by the mass exiting through the outlet. However, the combination of zero gradient for the velocity components, coupled with a linear extrapolation for the pressure, does not guarantee that this will be the case.

A simple solution can be achieved by summing the total mass flux entering the domain through the inlet (M_{in}), as well as the total mass flux exiting the domain through the outlet (M_{out}). By then multiplying the normal velocity component at each face on the outlet by $\frac{M_{in}}{M_{out}}$, the conservation of mass can be assured. However, this simple solution leads to an over-specification of the discrete problem on the outlet boundary, which can in some cases lead to convergence difficulties.

The root of the problem in this simple method is that the boundary velocity will generally be inconsistent with the boundary pressure gradient. This is due to the fact that the boundary velocity has been altered (multiplied by $\frac{M_{in}}{M_{out}}$) without any corresponding alterations to the pressure field. An alternative, which is more rigorous than the above approach, is to derive from the discrete momentum equations a *bulk pressure correction*, which (in an iterative manner) drives the solution toward global mass conservation.

To derive the iterative bulk pressure correction algorithm, consider the two-dimensional grid of Figure 3.4. Note that the method is equally applicable to three-dimensional grids, but the working is lengthy and hence will not be presented here.

The current mass flux through the outlet is given by ;

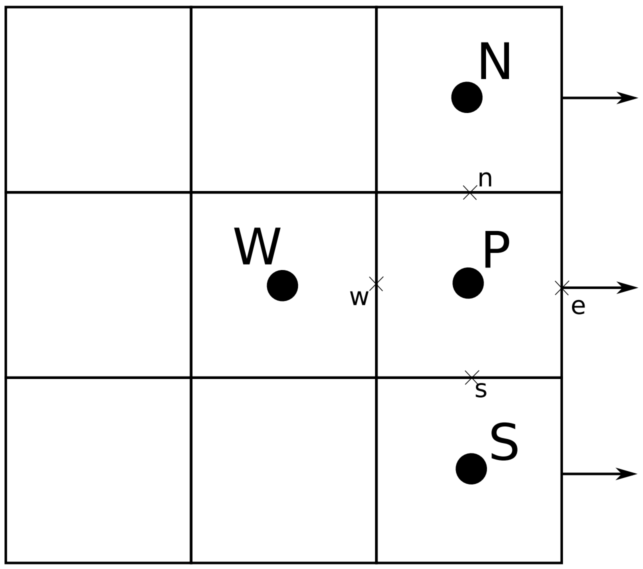


Figure 3.4: A sample two-dimensional outlet.

$$M_{out}^{l} \equiv \sum \rho u_{e}^{l} \Delta y \tag{3.44}$$

where the summation is conducted over all the faces that make up the outlet boundary, and the superscript l indicates that the flux is computed at the beginning of the current iteration.

By assuming a zero gradient boundary condition for the velocity on the outlet (which is appropriate provided the outlet is sufficiently far downstream from any flow disturbances), we can replace u_e in Equation 3.44 with u_P, giving 3.45 ;

$$M_{out}^{l} = \sum \rho u_{P}^{l} \Delta y \tag{3.45}$$

For global mass conservation, we require that the mass flux entering the domain through the inlet is equal to the mass flux leaving through the outlet. Hence we have ;

$$M_{in} = M_{out}$$
$$\therefore M_{in} = \sum \rho u_P \Delta y \tag{3.46}$$

However, the *current* mass flux M_{out}^l will, in general, not satisfy 3.46. We therefore apply a correction to the velocity component normal to the outlet, that drives the solution toward global mass conservation. The applied correction is of the form ;

$$u_P = u_P^l + u_P' \tag{3.47}$$

where u_P' is the correction, u_P^l is the current (non mass-conserving) velocity, and u_P is the final mass conserving velocity after applying the bulk correction.

Substituting 3.47 into 3.46 yields ;

$$M_{in} = \sum \rho \left(u_P^l + u_P'\right) \Delta y \tag{3.48}$$

We now wish to find the corresponding pressure correction at the boundary face that would drive the flow toward this globally mass conserving velocity. It has already been shown in Section 3.5 that the discretised u-momentum equation provides a link between the velocity correction and pressure correction as follows;

$$u_P' = \frac{1}{A_P} \left(p_w' - p_e'\right) \Delta y \tag{3.49}$$

where the SIMPLE approximation has been made in order to avoid introducing the contributions from surrounding nodes (see Section 3.5 for details).

The term p_w' in 3.49 is now set to zero, since we do not wish the bulk pressure correction algorithm to alter the solution on the interior of the domain. Only the boundary pressure should be set by the bulk correction algorithm, with the interior pressure field arising as a result of the SIMPLE algorithm. Substituting Equation 3.49 into 3.48 (with $p'_w = 0$), we have ;

$$M_{in} = \sum \rho u^l_P \Delta y - \sum \rho \frac{1}{A_P} p'_e \Delta y^2 \tag{3.50}$$

which upon rearrangement gives ;

$$p'_e = \frac{M^l_{out} - M_{in}}{\sum \rho \frac{1}{A_P} \Delta y^2} \tag{3.51}$$

where p'_e has been conveniently be removed from the summation since it is treated as a constant (i.e. the same correction is applied to all faces making up the outlet boundary).

Equation 3.51 gives a correction to the current boundary pressure that should be applied to each node on the outlet boundary in order to achieve global mass conservation.

3.7 Coordinate Transformations

Up to now, a Cartesian coordinate system has been considered. While this has been useful in demonstrating the finite-volume method, restricting ourselves to problems in a Cartesian reference frame is not particularly useful for problems involving complex geometries. This section considers the coordinate transformations required to simulate flows in a general coordinate system.

We wish to transform the global coordinate system (x, y, z) to the local coordinate system (ζ, η, ξ). From the chain rule we have;

$$\begin{aligned}
\frac{\partial \phi}{\partial \zeta} &= \frac{\partial \phi}{\partial x}\frac{\partial x}{\partial \zeta} + \frac{\partial \phi}{\partial y}\frac{\partial y}{\partial \zeta} + \frac{\partial \phi}{\partial z}\frac{\partial z}{\partial \zeta} \\
\frac{\partial \phi}{\partial \eta} &= \frac{\partial \phi}{\partial x}\frac{\partial x}{\partial \eta} + \frac{\partial \phi}{\partial y}\frac{\partial y}{\partial \eta} + \frac{\partial \phi}{\partial z}\frac{\partial z}{\partial \eta} \\
\frac{\partial \phi}{\partial \xi} &= \frac{\partial \phi}{\partial x}\frac{\partial x}{\partial \xi} + \frac{\partial \phi}{\partial y}\frac{\partial y}{\partial \xi} + \frac{\partial \phi}{\partial z}\frac{\partial z}{\partial \xi}
\end{aligned}$$

Or, in matrix form;

$$\begin{pmatrix} \frac{\partial\phi}{\partial\zeta} \\ \frac{\partial\phi}{\partial\eta} \\ \frac{\partial\phi}{\partial\xi} \end{pmatrix} = \begin{pmatrix} \frac{\partial x}{\partial\zeta} & \frac{\partial y}{\partial\zeta} & \frac{\partial z}{\partial\zeta} \\ \frac{\partial x}{\partial\eta} & \frac{\partial y}{\partial\eta} & \frac{\partial z}{\partial\eta} \\ \frac{\partial x}{\partial\xi} & \frac{\partial y}{\partial\xi} & \frac{\partial z}{\partial\xi} \end{pmatrix} \cdot \begin{pmatrix} \frac{\partial\phi}{\partial x} \\ \frac{\partial\phi}{\partial y} \\ \frac{\partial\phi}{\partial z} \end{pmatrix}$$

Inverting this system yields;

$$\begin{pmatrix} \frac{\partial\phi}{\partial x} \\ \frac{\partial\phi}{\partial y} \\ \frac{\partial\phi}{\partial z} \end{pmatrix} = \frac{1}{|J|} \begin{pmatrix} \left(\frac{\partial y}{\partial\eta}\frac{\partial z}{\partial\xi} - \frac{\partial z}{\partial\eta}\frac{\partial y}{\partial\xi}\right) & -\left(\frac{\partial y}{\partial\zeta}\frac{\partial z}{\partial\xi} - \frac{\partial z}{\partial\zeta}\frac{\partial y}{\partial\xi}\right) & \left(\frac{\partial y}{\partial\zeta}\frac{\partial z}{\partial\eta} - \frac{\partial z}{\partial\zeta}\frac{\partial y}{\partial\eta}\right) \\ -\left(\frac{\partial x}{\partial\eta}\frac{\partial z}{\partial\xi} - \frac{\partial z}{\partial\eta}\frac{\partial x}{\partial\xi}\right) & \left(\frac{\partial x}{\partial\zeta}\frac{\partial z}{\partial\xi} - \frac{\partial z}{\partial\zeta}\frac{\partial x}{\partial\xi}\right) & -\left(\frac{\partial x}{\partial\zeta}\frac{\partial z}{\partial\eta} - \frac{\partial z}{\partial\zeta}\frac{\partial x}{\partial\eta}\right) \\ \left(\frac{\partial x}{\partial\eta}\frac{\partial y}{\partial\xi} - \frac{\partial y}{\partial\eta}\frac{\partial x}{\partial\xi}\right) & -\left(\frac{\partial x}{\partial\zeta}\frac{\partial y}{\partial\xi} - \frac{\partial y}{\partial\zeta}\frac{\partial x}{\partial\xi}\right) & \left(\frac{\partial x}{\partial\zeta}\frac{\partial y}{\partial\eta} - \frac{\partial y}{\partial\zeta}\frac{\partial x}{\partial\eta}\right) \end{pmatrix} \cdot \begin{pmatrix} \frac{\partial\phi}{\partial\zeta} \\ \frac{\partial\phi}{\partial\eta} \\ \frac{\partial\phi}{\partial\xi} \end{pmatrix} \tag{3.52}$$

where the terms $\frac{\partial y}{\partial\zeta}$, $\frac{\partial y}{\partial\eta}$, $\frac{\partial y}{\partial\xi}$, $\frac{\partial x}{\partial\zeta}$ $\frac{\partial x}{\partial\eta}$, $\frac{\partial x}{\partial\xi}$ $\frac{\partial z}{\partial\zeta}$ $\frac{\partial z}{\partial\eta}$ and$\frac{\partial z}{\partial\xi}$ are geometrical quantities that can be discretised via a finite difference, and then evaluated simply by examining the mesh. The term, $|J|$ is the Jacobian of the transformation, and is given by;

$$|J| = \frac{\partial x}{\partial\zeta}\left(\frac{\partial y}{\partial\eta}\frac{\partial z}{\partial\xi} - \frac{\partial z}{\partial\eta}\frac{\partial y}{\partial\xi}\right) - \frac{\partial y}{\partial\zeta}\left(\frac{\partial x}{\partial\eta}\frac{\partial z}{\partial\xi} - \frac{\partial z}{\partial\eta}\frac{\partial x}{\partial\xi}\right) + \frac{\partial z}{\partial\zeta}\left(\frac{\partial x}{\partial\eta}\frac{\partial y}{\partial\xi} - \frac{\partial y}{\partial\eta}\frac{\partial x}{\partial\xi}\right) \tag{3.53}$$

The transformations given by 3.52 can now be substituted into the Navier-Stokes equations and the resulting equations can then be discretised via the finite volume method in the usual way. By doing this, it will be noticed that, for the diffusive flux terms on non-orthogonal grids, terms involving the velocities at the *vertices* of the cell will appear. These extra terms are referred to as "cross-diffusion" sources. The cross-diffusion terms should be evaluated via interpolation, and added to the source term, S_u.

Chapter 4

Modelling Turbulence

4.1 Introduction

The majority of flows that are of engineering interest are turbulent. Turbulent flows are fully defined by the Navier-Stokes equations. Hence by solving the time-dependent Navier-Stokes equations through typical discretisation techniques (e.g. the finite volume method), one could in principle obtain a depiction of a turbulent flow field. This is known as a Direct Numerical Simulation (DNS). However, turbulent flows are characterised by apparently random fluctuations over a wide range of length and time scales. At the largest scales, eddies of comparable scale to that of the flow field form. These large eddies are unstable and break down, transferring their energy to smaller eddies. These smaller eddies are themselves unstable and also break down, passing their energy to yet smaller eddies. This process continues until the scales of the smallest eddies are sufficiently small that viscous damping is prominent, at which point the smallest eddies are dissipated as heat. This leads to a near-fractal distribution of scales. In order to depict all the significant features of turbulence in a DNS simulation, it is necessary to solve over a computational domain that is larger than the largest eddies, and with a grid-resolution fine enough to fully resolve the smallest eddies. Failure to capture either end of this spectrum of scales would lead to an invalid simulation. In practice, this requirement almost invariably leads to a prohibitively fine grid resolution. The computational resources available (even when using modern supercomputers or large parallel clusters) are simply insufficient to perform DNS simulations in all but the most simple of flows, and at relatively low Reynolds numbers (and it is likely this will remain the case for some time). Hence a model is required to account for the effects of turbulence, whilst without fully resolving all (or any) turbulent scales. One class of models is based on the eddy-viscosity hypothesis, and these will be looked at within this chapter, since this the approach adopted in the present work.

4.2 Reynolds-Averaged Navier-Stokes Equations (RANS)

More often than not, it is the mean flow field that is of interest to an engineer rather than the precise details of turbulent fluctuations. By decomposing variables into a mean and fluctuating part, one can arrive at the Reynolds-Averaged Navier-Stokes (RANS) equations. For a steady state simulation, the following decomposition is used on a variable;

$$\phi\left(x_i,t\right) = \overline{\phi\left(x_i\right)} + \phi'\left(x_i,t\right)$$

where

$$\overline{\phi\left(x_i\right)} = \lim_{T\to\infty}\frac{1}{T}\int_0^T \phi\left(x_i,t\right)\cdot dt$$

To develop the RANS equations, we start with the incompressible, steady state Navier-Stokes equations, given below.

$$\frac{\partial u_j}{\partial x_j} = 0 \tag{4.1a}$$

$$\frac{\partial}{\partial x_j}\left[\rho u_i u_j + p\delta_{ij} - \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\right] = 0 \qquad i = 1,2,3 \tag{4.1b}$$

Substituting $u_i(x,t) = \bar{u}_i(x) + u_i'(x,t)$ and $p(x,t) = \bar{p}(x) + p'(x,t)$, and taking the time-average of the result yields:

$$\frac{\partial \bar{u}_j}{\partial x_j} = 0 \tag{4.2a}$$

$$\frac{\partial}{\partial x_j}\left[\rho\bar{u}_i\bar{u}_j - 2\mu\bar{S}_{ij}\right] = -\frac{\partial}{\partial x_j}\left(\bar{p}\delta_{ij} - \rho\overline{u_i'u_j'}\right) \tag{4.2b}$$

where $\bar{S}_{ij}$ is the mean rate of strain tensor, defined as:

$$\bar{S}_{ij} \equiv \frac{1}{2}\left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i}\right)$$

Equations 4.2 are the Reynolds-Averaged Navier-Stokes equations. The terms $\rho\overline{u_i'u_j'}$ in Equation 4.2b are the Reynolds stresses, and are unknown; they cannot be represented purely in terms of the mean flow quantities. The system of equations given by 4.2 is therefore not closed. Models are required to approximate the Reynolds stresses.

4.3 The Eddy Viscosity Hypothesis

Since turbulence leads to increased mixing of a fluid, it seems a logical assumption to model the effect of turbulence as an increased viscosity. To this end, a 'turbulent viscosity' can be computed which is added to the molecular viscosity to yield the desired level of mixing. This approach is known as the eddy-viscosity hypothesis. The Reynolds stresses are then modelled as follows:

$$-\rho\overline{u_i'u_j'} \approx \mu_t \left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i}\right) - \frac{2}{3}\rho\delta_{ij}k \tag{4.3}$$

where μ_t is the turbulent viscosity and k is the specific turbulent kinetic energy, defined as $k = \frac{1}{2}\overline{u_i'u_i'}$.

Whilst the eddy-viscosity hypothesis is incorrect in the detail, the method has been shown to provide reasonable engineering approximations to turbulence for a variety of flows. The method is also simple to implement and is computationally economical. It is therefore extensively used in both research and industry.

To close the RANS equations (Equations 4.2) it is now necessary to develop a definition of the turbulent viscosity (μ_t). Turbulence is characterised by its kinetic energy and a length scale (see Prandtl, [43]). By combining these parameters through dimensional analysis, one can arrive at the following definition of the turbulent viscosity:

$$\mu_t = C_\mu f_\mu \rho k^{\frac{1}{2}} L \tag{4.4}$$

where C_μ is a dimensionless constant, f_μ is a dimensionless damping term (to be discussed presently) and L is a lengthscale. What now remains is to determine both k and L throughout the domain. There are several different methodologies available for achieving this. The simplest method is a so called *zero-equation* model, in which both k and L are prescribed algebraically. However, algebraic relationships are derived from the experimental observation of simple flows and are simply not valid for more complex flows.

Alternatively, *one-equation* models can be used in which one variable is solved using a transport equation, while the other is obtained from algebraic relationships. While this methodology does improve the generality of the turbulence model somewhat, it is still limited to rather simple flows. As an alternative to one-equation models, *two-equation* models can be employed where two transport equations are solved, one to determine k and the other to determine a length scale L. Using this approach, it has been shown that reasonable accuracy can be achieved for a wide range of different flows. However, generality is still restricted somewhat due to limitations of the eddy-viscosity hypothesis and by some of the approximations that have to be made. Despite these limitations, two-equation models offer an attractive compromise between efficiency and accuracy, and hence are used extensively. It is this approach that shall be considered here.

4.3.1 Two-Equation Models

There are several different two-equation models available in the literature. In almost all such models, a transport equation for the specific turbulent kinetic energy, k, is used directly in Equation 4.4. It is reasonably straightforward to show that the exact transport equation for k is given by Equation 4.5. For derivation details of Equation 4.5, the reader is referred to Reference [44].

$$\rho\frac{\partial k}{\partial t} + \frac{\partial}{\partial x_j}\left[\rho\bar{u}_j k - \mu\frac{\partial k}{\partial x_j}\right] = -\rho\overline{u_i'u_j'}\frac{\partial \bar{u}_i}{\partial x_j} - \mu\overline{\frac{\partial u_i'}{\partial x_j}\frac{\partial u_i'}{\partial x_j}} - \frac{\partial}{\partial x_j}\left[\frac{1}{2}\rho\overline{(u_i'u_i'u_j')} + \overline{u_j'p'}\right] \tag{4.5}$$

The terms on the left hand side of Equation 4.5 need no modelling and are of the same form as a general scalar transport equation. Standard finite volume method techniques can therefore be used to discretise the left hand terms. The right hand terms however, are not represented solely in terms of the mean flow variables, hence approximations to these terms are required.

The first term on the right hand side of Equation 4.5 represents the rate of production of turbulent kinetic energy by the mean flow. Using the eddy-viscosity hypothesis (Equation 4.3) the production term is approximated as follows:

$$P_k = -\rho\overline{u_i'u_j'}\frac{\partial \bar{u}_i}{\partial x_j} \approx \mu_t\left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i}\right)\frac{\partial \bar{u}_i}{\partial x_j} \tag{4.6}$$

The second term on the right hand side represents the turbulent dissipation rate (i.e. $\rho\epsilon$). This is physically the rate at which turbulent kinetic energy is converted into heat:

$$\mu\overline{\frac{\partial u_i'}{\partial x_j}\frac{\partial u_i'}{\partial x_j}} \approx \rho\epsilon \tag{4.7}$$

This process is irreversible. A method of modelling the dissipation rate, ϵ, will be given in due course.

The final term on the right hand side physically represents the turbulent diffusion of turbulent kinetic energy. This can be modelled using the gradient diffusion approximation:

$$\left(\frac{1}{2}\rho\overline{(u_i'u_i'u_j')} + \overline{u_j'p'}\right) \approx \frac{\mu_t}{\sigma_k}\frac{\partial k}{\partial x_j} \tag{4.8}$$

for some (constant) σ_k. The final modelled transport equation for k is therefore given by:

$$\rho\frac{\partial k}{\partial t} + \frac{\partial}{\partial x_j}\left[\rho\bar{u}_j k - \left(\mu + \frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] = P_k - \rho\epsilon \tag{4.9}$$

This equation can be solved using standard solution procedures. For example, if using a structured grid, one can use the highly efficient tri-diagonal matrix algorithm, coupled with an alternating direction implicit solution procedure (see Chapter 3 for details).

To determine the length scale of the turbulence, a second transport equation is solved. The choice of variable that is used in this transport equation varies from one model to the next. Different models can perform better than others for different simulations. There is no overall *best* model that is universally applicable to the whole class of turbulent flows. This is testament to the fact that the eddy-viscosity hypothesis, along with the further modelling approximations that are made, are non-exact.

In the present work, the turbulent dissipation rate, ϵ, is used to determine the length scale. This choice leads to a class of models known as the $k-\epsilon$ models. The $k-\epsilon$ class was chosen due to the fact that it is extensively used, and is hence well established and arguably better suited to a wider range of flow situations than some other options.

The exact ϵ transport equation can be derived from the Navier-Stokes equations in a similar way to that of the k transport equation (see Reference [44] for details). However, the result contains a number of unknown correlations associated with the very small scale part of the turbulence spectrum. As a result, the equation has to undergo extensive modelling to represent all fluctuating quantities in terms of the known mean quantities. Consequently, there is little point in attempting to derive an equation for ϵ; instead the whole equation can be considered an empirical model without much loss of rigour. The typical modelled transport equation for ϵ is given by Equation 4.10:

$$\rho\frac{\partial \epsilon}{\partial t}+\frac{\partial}{\partial x_j}\left[\rho\bar{u}_j\epsilon-\left(\mu+\frac{\mu_t}{\sigma_\epsilon}\right)\frac{\partial \epsilon}{\partial x_j}\right]=C_{\epsilon 1}\frac{\epsilon}{k}P_k-C_{\epsilon 2}\rho\frac{\epsilon^2}{k}+S_\epsilon \tag{4.10}$$

where the constants $C_{\epsilon 1}$ and $C_{\epsilon 2}$ are closure coefficients; they assume appropriate values such that the resulting flow field for simple flows is equivalent to that from alternative data sources (often experimental or DNS data). The term S_ϵ represents any additional model dependent source terms (if present) such as the Yap correction term (see Section 4.4). Appropriate values of the closure coefficients will be given in Section 4.3.2.

Once the distribution of ϵ has been obtained throughout the domain, ϵ is related to the length scale through the empirical relation $\epsilon \approx k^{\frac{3}{2}}/L$. The result is used (along with the value of k) to determine the turbulent viscosity via Equation 4.4. This turbulent viscosity is then used to approximate the Reynolds stresses via Equation 4.3, which are used in the RANS equations (Equations 4.2) to yield the mean flow properties. This gives a complete (closed) model of turbulence.

4.3.2 Near Wall Treatments

Near to a wall, the behaviour of the $k-\epsilon$ model as described above is inaccurate. Close to solid boundaries, low Reynolds number viscous effects become prominent. If the $k-\epsilon$ equations are to be solved right up to a wall, empirical damping terms are required to ensure the turbulent predictions correlate with experimental observations (at least for simple flows). The typical forms adopted for the k and ϵ equations, with damping terms and additional source terms to account for near wall effects, are given by Equations 4.11, 4.12 and 4.13. These equations are collectively referred to as the low-Reynolds number $k-\epsilon$ turbulence models.

$$\rho\frac{\partial k}{\partial t}+\frac{\partial}{\partial x_j}\left[\rho\bar{u}_j k-\left(\mu+\frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right]=P_k-\rho\epsilon \tag{4.11}$$

$$\rho\frac{\partial\tilde{\epsilon}}{\partial t}+\frac{\partial}{\partial x_j}\left[\rho\bar{u}_j\tilde{\epsilon}-\left(\mu+\frac{\mu_t}{\sigma_\epsilon}\right)\frac{\partial\tilde{\epsilon}}{\partial x_j}\right]=C_{\epsilon 1}f_1\frac{\tilde{\epsilon}}{k}P_k-C_{\epsilon 2}\rho f_2\frac{\tilde{\epsilon}^2}{k}+\rho E \tag{4.12}$$

$$\epsilon=\tilde{\epsilon}+D \tag{4.13}$$

The damping terms and coefficients may vary from one model to the next. These terms are given in Table 4.1 for the Chien, [45], and Launder-Sharma, [2], models (the two models implemented in this project).

	Chien	Launder-Sharma
f_μ	$1-e^{-0.0115y^+}$	$\exp\frac{-3.4}{(1+Re_t/50)^2}$
f_1	1	1
f_2	$1-0.22e^{-\left(\frac{Re_t}{6}\right)^2}$	$1-0.3e^{-Re_t^2}$
D	$2\frac{\mu}{\rho}\frac{k}{y^2}$	$2\frac{\mu}{\rho}\left(\frac{\partial\sqrt{k}}{\partial x_j}\right)$
E	$-\frac{2\mu\tilde{\epsilon}}{y^2}e^{-0.5y^+}$	$2\frac{\mu}{\rho}\frac{\mu_t}{\rho}\left(\frac{\partial^2 u_j}{\partial x_j^2}\right)$
$C_{\epsilon 1}$	1.35	1.44
$C_{\epsilon 2}$	1.8	1.92
σ_k	1	1
σ_ϵ	1.3	1.3
c_μ	0.09	0.09

Table 4.1: Damping terms and coefficients for the Chien and Launder-Sharma low-Re $k-\epsilon$ models.

The damping terms and coefficients in these models have been arrived at empirically, through the observation of experimental flow data and/or DNS data. The terms are tuned to match observations for simple flows (for example, with the aim of reproducing the law of the wall for a flat plate boundary layer). The fact that there are several variants of the $k-\epsilon$ model, with no overall 'superior model', is evidence of the fact that the different models are tuned for specific applications. Their applicability to complex flows is therefore questionable. For example, the $k-\epsilon$ model performs poorly in flows with strong pressure gradients.

4.3.3 Boundary Conditions

In both the Chien and Launder-Sharma turbulence models, both k and $\tilde{\epsilon}$ are equal to zero at a wall. The former result follows from the no-slip condition. For the latter result, the turbulent dissipation rate is generally non-zero at a wall. However the extra term, D in the models is designed to provide the very near-wall dissipation rate and hence $\tilde{\epsilon}$ is equal to zero at a wall.

At an outflow a zero-gradient condition may often be assumed (if this is physically the case). By placing the outflow far enough downstream from disturbed flow, this assumption is usually valid. Outflow boundary conditions are treated in the same way as those of the mean flow variables.

When using the overset grid method, interpolation boundaries are frequently encountered. In this situation, values for both k and $\tilde{\epsilon}$ can simply be interpolated from a suitable donor grid in the same way as for the mean flow variables (see Chapter 5 for details).

At an inlet, turbulent conditions must be estimated or reproduced from experimental data (if available). It is often useful to define the turbulent inlet conditions in terms of a viscosity ratio ($\frac{\mu_t}{\mu}$) and a turbulence intensity, (I), defined by:

$$I = \frac{U'}{\bar{u}} \tag{4.14}$$

where U' is the root-mean-square of the turbulent velocity fluctuations and $\bar{u}$ is the mean Reynolds-averaged velocity.

The primitive variables, k and $\tilde{\epsilon}$ at the inlet are then related to the intensity and viscosity ratio through Equations 4.15:

$$k_{inlet} = \frac{3}{2}\left(UI\right)^2 \tag{4.15a}$$

$$\tilde{\epsilon}_{inlet} = C_\mu \frac{\rho k_{inlet}^2}{\mu}\left(\frac{\mu_t}{\mu}\right)^{-1} \tag{4.15b}$$

where U is the magnitude of the mean inlet velocity. In the absence of suitable experimental data, an experienced engineer can make engineering judgement as to suitable levels of turbulence, based on the expected conditions upstream of the inlet (i.e. outside of the computational domain). Table 4.2 gives possible values for a selection of upstream flow conditions.

Intensity	Example upstream conditions	Typical values of intensity (I)
High	High Reynolds number internal flows involving complex geometries. e.g. compressors or heat exchangers	5% - 20%
Medium	Flows in large pipes or rooms	1% - 5%
low	High quality wind-tunnels or flow that originates from stationary conditions e.g. the external flow over a car on a windless day	$< 1\%$

Table 4.2: Suitable values of turbulence intensities for different flow inlet conditions

4.4 Length scale correction

In regions of flow separation it is well understood that the standard $k - \epsilon$ turbulence model class tends to predict excessive levels of turbulence [46, 47, 48]. The root of the problem lies in an underprediction of ϵ in the separation region [44, 49]. An *ad hoc* correction to the length scale has been proposed by Yap [50], which takes the form of an additional source term in the ϵ equation. The correction that is proposed is given by Equation 4.16:

$$S_\epsilon = \max \left| 0.83\rho\frac{\epsilon^2}{k}\left(\frac{k^{1.5}}{2.55\epsilon y_n} - 1\right)\left(\frac{k^{1.5}}{2.55\epsilon y_n}\right)^2, 0 \right| \tag{4.16}$$

where y_n is the normal distance to the nearest wall.

In complex geometries, the normal distance to the wall is difficult to compute, and hence the Yap correction is poorly suited to such situations. Iacovides and Raisee, [51], propose a differential form of the length scale correction term that is independent of the distance to the wall. The additional source term of ϵ that this differential correction takes is of the following form:

$$S_\epsilon = \max \left| 0.83F\left(F+1\right)^2 \rho\epsilon^2/k, 0 \right| \tag{4.17}$$

$$F \equiv \left(D - E\right)/2.55 \tag{4.18}$$

$$D \equiv \left[\left(\frac{\partial k^{3/2}/\epsilon}{\partial x_j}\right)\left(\frac{\partial k^{3/2}/\epsilon}{\partial x_j}\right)\right]^{0.5} \tag{4.19}$$

$$E \equiv 2.55\left[1 - \exp(-0.1069Re_t)\right] + 0.1069 \times 2.55Re_t \exp(-0.1069Re_t) \tag{4.20}$$

It can be seen that this alternative form of the length scale correction is independent of wall distance.

4.5 Alternative Turbulence Models

While the two-equation eddy viscosity turbulence models described above do offer an efficient and reasonably accurate description of turbulence for a variety of flows, they are far from the only options available. Other options typically address different points on the accuracy versus computational efficiency spectrum. To gain increased accuracy, one usually has to sacrifice computational efficiency. Clearly, there is no point in blindly striving for the greatest possible accuracy at any cost, when a simple and efficient turbulence model may suffice. For this reason, simple eddy-viscosity models have their place, but so do other more accurate methods. There is no universal 'one fits all' approach to turbulence modelling, and a small selection of the alternative methods will be briefly outlined in this section.

At a greater potential accuracy than the $k - \epsilon$ model, we have the Reynolds-Stress transport models, where a transport equation for each Reynolds stress is solved. While this method does offer the potential of increased accuracy and greater generality, the success of the model has been limited. Some turbulent flows can be simulated more accurately with the Reynolds-stress approach (versus a $k - \epsilon$ model) whereas the opposite may be true for other flow categories. For 3D flows, six transport equations are solved, one for each of the Reynolds stresses, and one extra transport equation is solved to determine the dissipation rate. In total, seven transport equations are therefore solved (versus two for the $k - \epsilon$ models). This amounts to a significant increase in computational resources relative to two-equation models. As a result, the Reynolds-stress transport model has received limited acclaim.

When striving for greater accuracy over RANS approaches, or when the details of instantaneous fluctuations are required, a Large Eddy Simulation (LES) is an attractive option. In this method, the larger eddies (those large enough to be fully resolved on a given grid) are fully resolved in both space and time. The smaller, sub-grid scale, eddies are modelled. According to Kolmogorov's theory of self-similarity, [52], the larger eddies in a flow are dependent on the geometry, whereas smaller eddies are more universal. This makes the development of a suitable sub-grid scale turbulence model a relatively simple task, and also means that large-eddy simulations have the potential of a wide range of applicability.

The computational costs associated with performing a large-eddy simulation are significantly higher than for a RANS approach (especially given the simulation will always be three-dimensional and time dependent, even where the mean flow is two-dimensional and steady), but are lower than performing a full DNS simulation. When given the processing and storage capabilities of modern parallel clusters, LES is a practical option for the majority of simulations. However, since not all users of CFD have access to such resources, LES usage is somewhat restricted. This situation will no doubt improve with time as computational capabilities improve. It is this authors opinion that LES usage will surpass RANS models during the next generations of turbulence modelling.

Chapter 5

The Overset Algorithm

5.1 Introduction

In this chapter, an outline of the overset algorithms used and developed as a part of this thesis are presented. In Section 5.2, the details of the new hole cutting algorithm developed in this project are presented. This is followed by the details of an algorithm that is used to determine if two triangles intersect one another, which is a necessary test used by the hole cutting algorithm. The interpolation methods implemented in this project are then outlined in Section 5.4. The details of two alternative interpolation methods are given; Section 5.4.1 gives the implementation details of a standard tri-linear interpolation, while Section 5.4.2 introduces the semi-conservative approach of Tang *et. al.*, [1], which attempts to enforce the conservation of mass over overset grids. In Section 5.5, details of binary search trees and an alternating digital tree formulation are presented. These algorithms are used for quick searching and geometric intersection testing, for example, in order to find interpolation stencils. Following this, Section, 5.6 outlines the implementation details of a surface grid 'zipping' algorithm, based on the method of Chan and Buming, [53], which is used in the present study for applying an overall bulk correction to overset grids. Finally, the algorithm presented in Section 5.7 demonstrates how all the algorithms, described elsewhere in this chapter, come together to provide domain connectivity between the overset grids.

5.2 Hole-Cutting

The objective of the hole-cutting algorithm is to provide an overset grid from a set of structured sub-grids. This is achieved by tagging each cell on each sub-grid as a standard cell, an interpolation cell, or an unused cell. Standard cells are those that the discritised governing equations can be solved on using a computational stencil comprising of other standard cells, or interpolation cells only. Interpolation cells are cells in which the values of the primitive variables are obtained through interpolation, and are required in order to provide a complete computational stencil to other standard cells. Unused cells are cells that take no part in the computation at the current time-step (they may come into the computational domain at later time-steps however).

For simple geometries, hole cutting is not always necessary. However, in the general case, some cells may lie outside of the computational domain, in which case they must be 'cut', (i.e. tagged as unused). In order to remove cells that lie outside of the computational domain, the first step is to identify any cells that intersect the edge of the computational domain. The intersection of walls shall be considered here, but the method is equally applicable to other edges of the computational domain, such as inlets, outlets and symmetry boundary conditions. A 'wall face' is defined as the quadrilateral face of a cell that has the wall boundary condition applied to it. To test for intersection of this wall face with any given hexahedral cell, the wall face can be split into two triangles. Similarly, each face of the test cell can be split into two triangles, generating a total of twelve triangles for the test cell, in addition to the two triangles for the wall face. The wall face to cell intersection test is then decomposed into upto twenty-four triangle-triangle intersection tests (the triangle-triangle intersection test algorithm used in the present study is described in Section 5.3). If an intersection is detected, then the test cell must lie partly outside of the computational domain, and therefore cannot be used in the computation, and hence should be tagged as an unused cell. By means of an example, Figure 5.1 illustrates user generated grids that may be used for the study of flow around a circular cylinder. Figure 5.2 shows the same grids after removing any cells intersecting the cylinder's wall.

Once all cells that intersect the edge of the computational have been removed, what remains is to remove the cells that lie *fully* outside of the computational domain. This can be achieved by noting that any cells that are adjacent to an unused cell cannot be standard cells (since their computational stencils would be incomplete). They must therefore be either interpolation cells or unused cells. For each cell that is adjacent to an unused cell, a suitable donor grid from which to interpolate from is attempted to be found (this is achieved by attempting to find interpolation coefficients between 0 and 1, as described in Section 5.4). Where no suitable donor grid can be identified the cell in question must be outside the computational domain, and is therefore tagged as unused. Alternatively, if a suitable donor-grid can be identified, the cell should be tagged as an interpolation cell. By repeating this process until no further changes are made by repeating further, all cells that lie outside of the computational domain will have been tagged as unused, and a valid overset grid will have been attained. Figure 5.3 shows the outcome of this step.

While the overset grid of Figure 5.3 is valid, it corresponds to the case of the maximum permissible overlap. It is however usually desirable to minimise the overlap between grids in order to reduce duplicate computational effort. In an overlap region, where there are multiple cells from different sub-grids covering the same physical space, it is not immediately obvious which cell (if any) to mark as unused. The majority of hole-cutting algorithms available in the literature allow the user to prescribe a priority to whole sub-grids and cells are cut from sub-grids with the lowest priority, while preserving cells on sub-grids with higher priority (see for example Reference [54]). The sub-grid with the highest priority therefore remains fully intact. While this method is simple to implement, it is not particularly satisfactory since if a high priority sub-grid overlaps a near-wall region on a lower priority sub-grid, the near wall nodes of the lower priority grid will be cut. Often, near-wall nodes will have been generated with particular care and attention toward the near wall grid spacing (particularly for turbulent flows). Having these cells marked as unused by the hole-cutting algorithm would lead to poor quality solutions, or to grids that are difficult to generate (thereby undermining one of the primary benefits of the overset method). To overcome this shortfall, a new hole-cutting algorithm has been developed in which priorities are assigned on a cell by cell basis (rather than a grid by grid basis). Priorities are calculated automatically, based on a user defined criteria (for example, by preserving cells with the lowest volume). The algorithm is as follows:

- Consider overlapping sub-grids Ω_A and Ω_B. Initially, we wish to remove the cells on Ω_A that (a) fully overlap Ω_B, (b) score lower than the mean score of the overlapping cells on Ω_B in a test such as lowest cell volume test, and (c) do not form part of an interpolation stencil for any of the interpolation cells on any other sub-grid. Any cells on Ω_A that match all three of these criteria may be tagged as unused. Note that the cells on sub-grid Ω_B are not altered at this stage.

- Firstly, all the cells on Ω_A that are required for interpolation by interpolation cells on other sub-grids are flagged. This is achieved by generating interpolation coefficients for all interpolation cells on grids Ω_i, $i \neq A$, via the method described in Section 5.4, and flagging the cells on Ω_A that comprise of the resulting interpolation stencil.

- Any cells that were flagged in the previous step cannot be removed, as to do so would cause an invalid interpolation stencil (one that comprises unused donor cells) for the interpolation cells on other sub-grids. Furthermore, if a cell that is *adjacent to* a previously flagged cell were to be removed, then the previously flagged cell should be converted to an interpolation cell in order to provide a complete computational stencil to surrounding standard cells. However, the interpolation from other interpolation cells is circular, and hence is not permitted. Therefore, all cells on Ω_A that are adjacent to those flagged in the previous step also cannot be removed, and therefore should also be flagged.

- For each cell on Ω_A that has not been flagged in the previous steps, the validity of removing the cell is assessed. If the cell were to be removed, any surrounding standard cells would need to be converted to interpolation cells to maintain complete computational stencils. The viability of this conversion is assessed by attempting to find a suitable donor grid for all cells subject to conversion. Where no donor grid can be identified for any one of the candidate conversions, the original cell in question cannot be removed. Where donor stencils can be identified in all cases, the original cell in question may be removed, (but is not removed yet).

- If the cell identified in the previous step may be removed, the cell(s) on the underlying grid that overlap the present cell are identified (this step is done approximately for computational efficiency by simply assessing if any underlying cell's bounding box intersects the bounding box of the present cell, where a bounding box is defined as a Cartesian aligned hexahedron, which is defined by six numbers, indicating the smallest and largest x, y and z coordinates of the object it encloses). The present cell is removed only if its volume (or other user defined criteria, such as aspect-ratio, skew, etc.) is greater than the average of that of the cells it overlaps. This way, smaller cells are preserved in favour of larger cells, thereby minimising the chances of insufficient grid resolution, and avoiding the removal of carefully generated near wall grid regions.

- These steps are repeated for each sub-grid in turn.

This fully defines the overset grids, and they are now ready to be used to obtain an overset simulation. The resulting overset grids for the circular cylinder case are shown in Figure 5.4. It can be seen that the overlap is now minimised relative to the overset grids of Figure 5.3 (which are valid, but are also excessive).

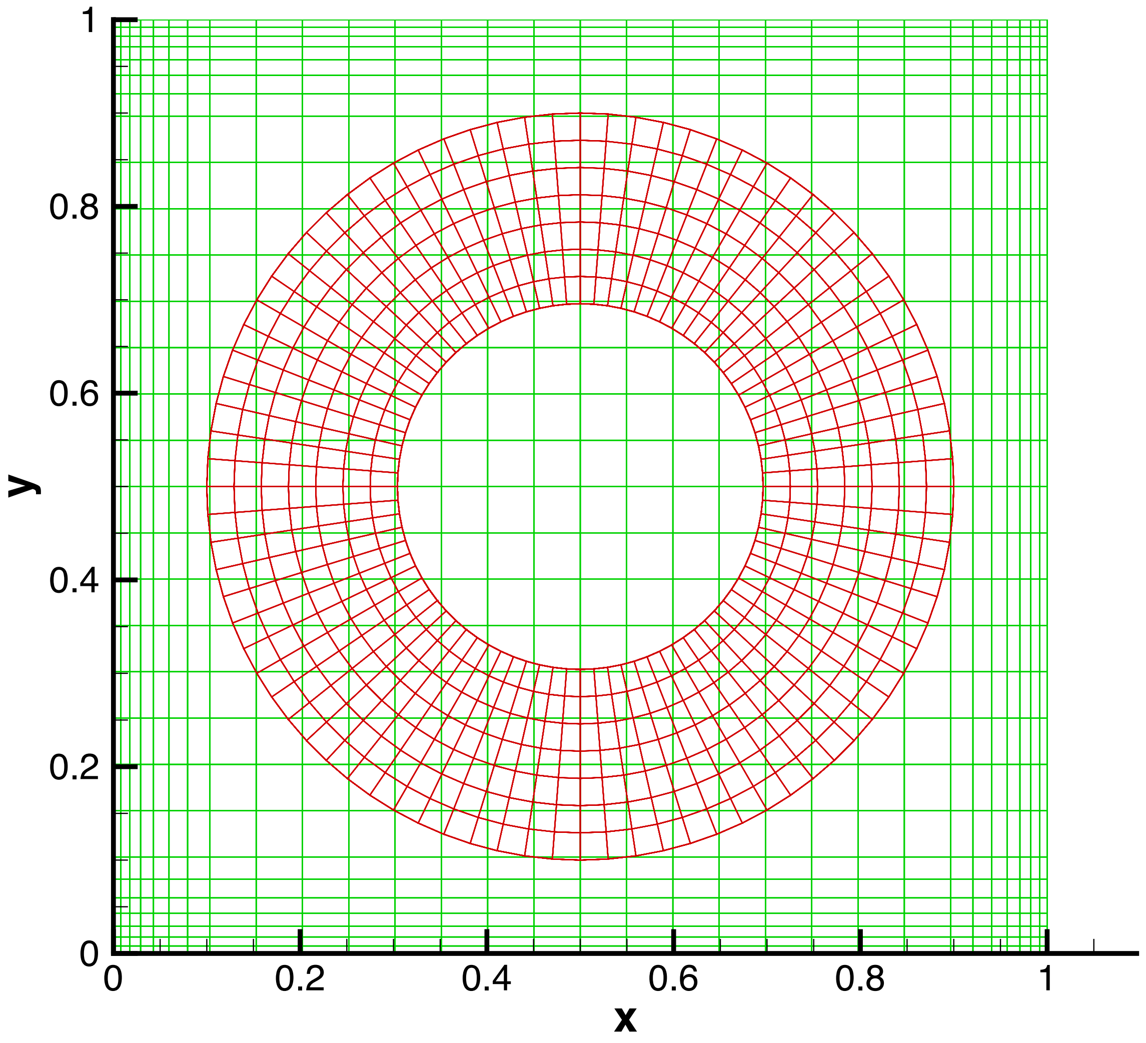


Figure 5.1: Initial user generated grids for the flow around a circular cylinder.

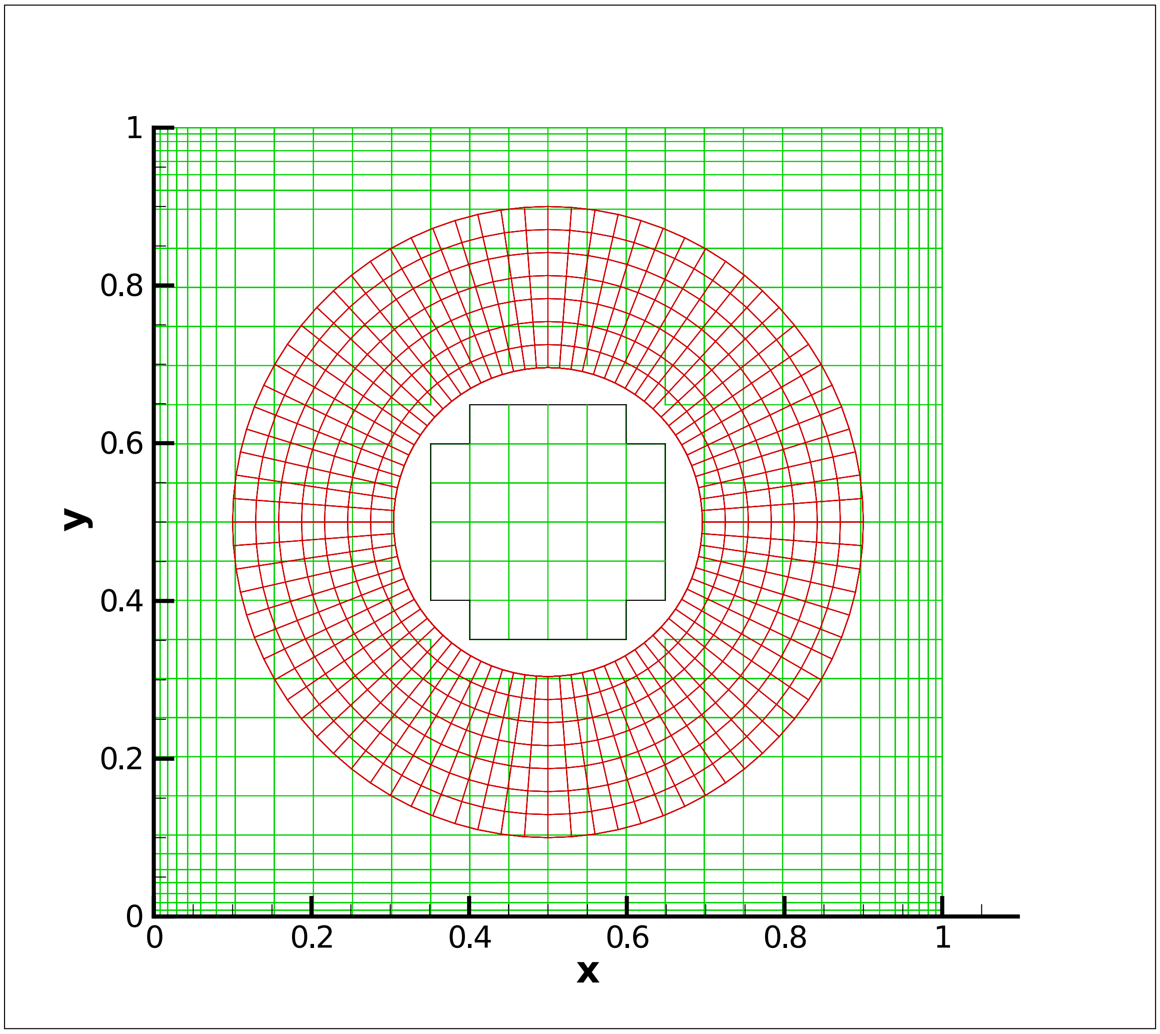


Figure 5.2: Grids for the flow over a circular cylinder, after removing all cells that intersect the cylinder's wall.

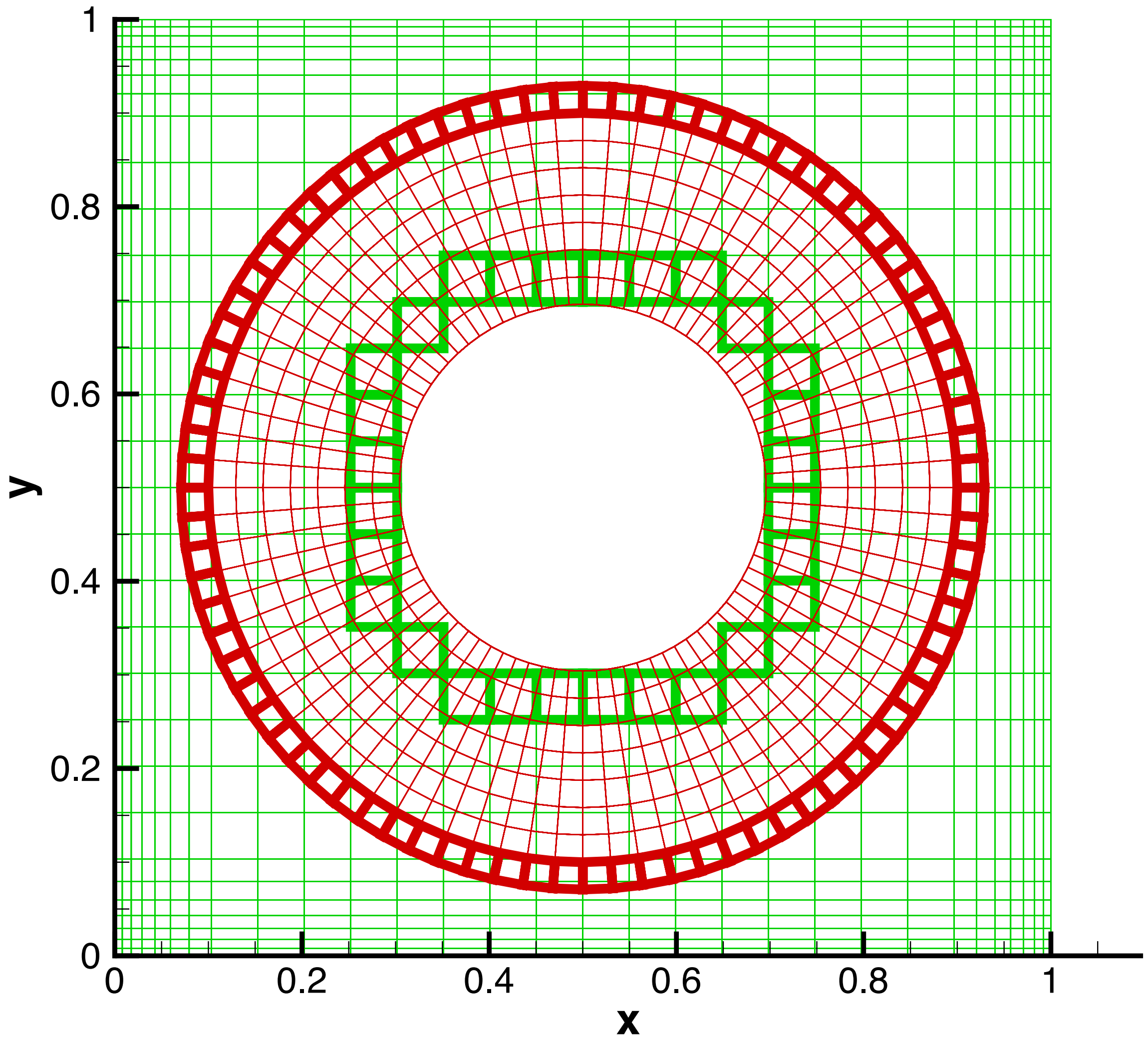


Figure 5.3: Grids for the flow over a circular cylinder, after removing any invalid cells, and tagging all (current) interpolation cells (interpolation cells are denoted through the use of heavy lines).

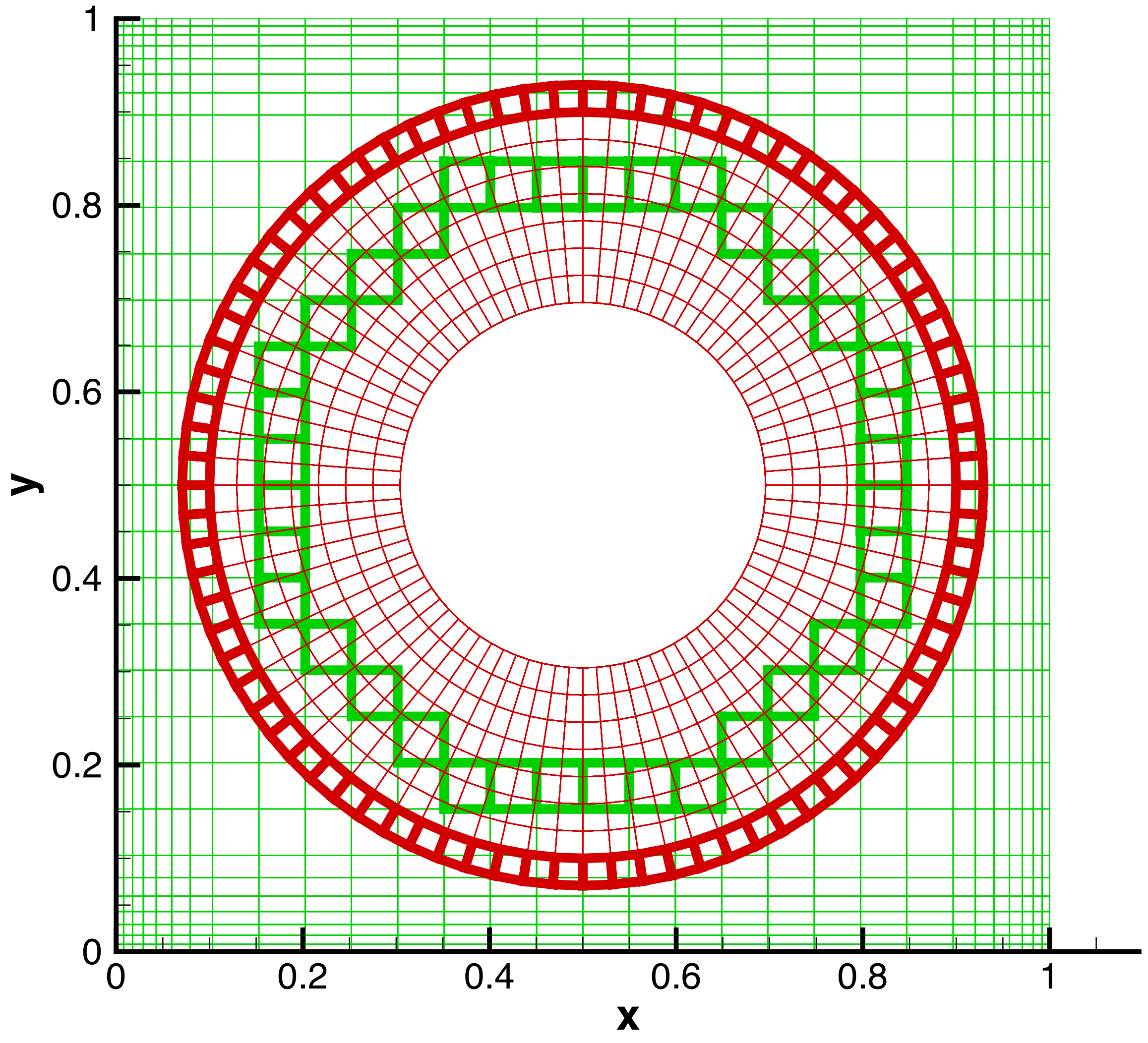


Figure 5.4: Grids for the flow over a circular cylinder, after removing any excess cells in the overlap region. These are the final grids, ready to be used in a simulation.

5.3 Triangle-Triangle Intersection test

It can be seen from Section 5.2 that the intersection of two triangles in 3D space frequently needs to be tested as part of the hole cutting procedure. This section describes the triangle-triangle intersection testing algorithm used for this purpose.

To test the intersection of triangles $T^{\mathcal{A}}$ and $T^{\mathcal{B}}$, which lie on planes $\pi^{\mathcal{A}}$ and $\pi^{\mathcal{B}}$ respectively, the equation of the plane $\pi^{\mathcal{A}}$ is first computed:

$$A^{\mathcal{A}}x + B^{\mathcal{A}}y + C^{\mathcal{A}}z + D^{\mathcal{A}} = 0 \tag{5.1a}$$

$$A^{\mathcal{A}} = y_1^{\mathcal{A}}\left(z_2^{\mathcal{A}} - z_3^{\mathcal{A}}\right) + y_2^{\mathcal{A}}\left(z_3^{\mathcal{A}} - z_1^{\mathcal{A}}\right) + y_3^{\mathcal{A}}\left(z_1^{\mathcal{A}} - z_2^{\mathcal{A}}\right) \tag{5.1b}$$

$$B^{\mathcal{A}} = z_1^{\mathcal{A}}\left(x_2^{\mathcal{A}} - x_3^{\mathcal{A}}\right) + z_2^{\mathcal{A}}\left(x_3^{\mathcal{A}} - x_1^{\mathcal{A}}\right) + z_3^{\mathcal{A}}\left(x_1^{\mathcal{A}} - x_2^{\mathcal{A}}\right) \tag{5.1c}$$

$$C^{\mathcal{A}} = x_1^{\mathcal{A}}\left(y_2^{\mathcal{A}} - y_3^{\mathcal{A}}\right) + x_2^{\mathcal{A}}\left(y_3^{\mathcal{A}} - y_1^{\mathcal{A}}\right) + x_3^{\mathcal{A}}\left(y_1^{\mathcal{A}} - y_2^{\mathcal{A}}\right) \tag{5.1d}$$

$$D^{\mathcal{A}} = -(x_1^{\mathcal{A}}\left(y_2^{\mathcal{A}}z_3^{\mathcal{A}} - y_3^{\mathcal{A}}z_2^{\mathcal{A}}\right) + x_2^{\mathcal{A}}\left(y_3^{\mathcal{A}}z_1^{\mathcal{A}} - y_1^{\mathcal{A}}z_3^{\mathcal{A}}\right) + x_3^{\mathcal{A}}\left(y_1^{\mathcal{A}}z_2^{\mathcal{A}} - y_2^{\mathcal{A}}z_1^{\mathcal{A}}\right) \tag{5.1e}$$

where $x_i^{\mathcal{A}}$ is the x-coordinate of the i'th vertex on triangle $T^{\mathcal{A}}$, and similarly for $y_i^{\mathcal{A}}$ and $z_i^{\mathcal{A}}$.

The signed distances from the vertices that make up triangle $T^{\mathcal{B}}$, to the plane $\pi^{\mathcal{A}}$ are computed by inserting the coordinates of each vertex of $T^{\mathcal{B}}$ into the equation of the plane $\pi^{\mathcal{A}}$, (Equation 5.1a). If the signed distances of all vertices that make up triangle $T^{\mathcal{B}}$, to the plane $\pi^{\mathcal{A}}$ are equal to 0, then planes $\pi^{\mathcal{A}}$ and $\pi^{\mathcal{B}}$ are coplanar, and the intersection can be discarded (co-planar triangles can be regarded as non-intersecting for the requirements of the present code).

Similarly, if some (but not all) of the signed distances computed above are equal to zero, then the triangle $T^{\mathcal{B}}$ touches the plane $\pi^{\mathcal{A}}$ at a point, or along an edge, but does not intersect the plane. If touching triangles were to count as an intersection, then further tests would be required to determine if triangle $T^{\mathcal{B}}$ touches the plane $\pi^{\mathcal{A}}$ *within* $T^{\mathcal{A}}$. However, for the purposes of hole cutting, touching triangles can be safely regarded as non-intersecting, hence if *any* of the signed distances computed above are equal to zero, the intersection is discarded.

After discarding all triangle pairs that failed the prior tests, further triangle pairs can be discarded (i.e. found to be non-intersecting) by testing if the triangle $T^{\mathcal{B}}$ intersects the plane $\pi^{\mathcal{A}}$, and vice versa. If the signed distances to $\pi^{\mathcal{A}}$ of all three vertices that make up $T^{\mathcal{B}}$ have the same sign (and are non-zero), then all points of $T^{\mathcal{B}}$ lie on the same side of the plane $\pi^{\mathcal{A}}$, hence triangle $T^{\mathcal{B}}$ does not intersect the plane $\pi^{\mathcal{A}}$, and the triangles therefore cannot intersect.

Similar tests are then performed on the vertices of triangle $T^{\mathcal{A}}$ against the plane $\pi^{\mathcal{B}}$. If any of these tests fail, the triangles do not intersect, and no further analysis is required.

For triangle pairs to have passed all prior tests, the triangles must intersect one another's plane, hence $\pi^{\mathcal{A}}$ and $\pi^{\mathcal{B}}$ must also intersect one another. The line of intersection, L, between planes $\pi^{\mathcal{A}}$ and $\pi^{\mathcal{B}}$, can be computed from:

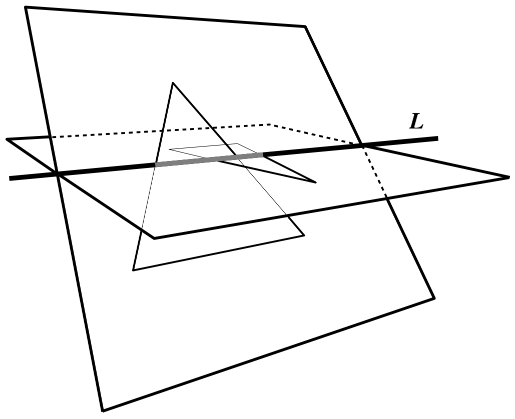
$$L_x = x_0 + t(N^{\mathcal{A}} \times N^{\mathcal{B}})_x \tag{5.2a}$$

$$L_y = y_0 + t(N^{\mathcal{A}} \times N^{\mathcal{B}})_y \tag{5.2b}$$

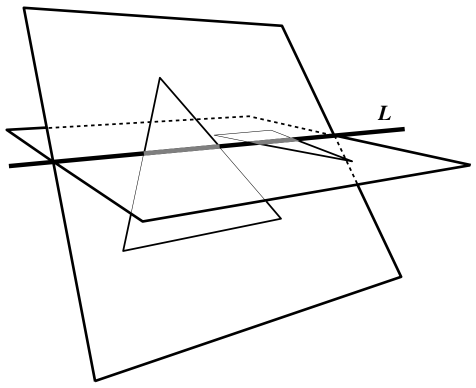
$$L_z = z_0 + t(N^{\mathcal{A}} \times N^{\mathcal{B}})_z \tag{5.2c}$$

where $N^{\mathcal{A}} = (A^{\mathcal{A}}, B^{\mathcal{A}}, C^{\mathcal{A}})$, $N^{\mathcal{B}} = (A^{\mathcal{B}}, B^{\mathcal{B}}, C^{\mathcal{B}})$ are the normals of planes $\pi^{\mathcal{A}}$ and $\pi^{\mathcal{B}}$ respectively, (x_0, y_0, z_0) is some point on L, and t is the parameter that can take any real value ($-\infty < t < \infty$).

For triangles $T^{\mathcal{A}}$ and $T^{\mathcal{B}}$ to intersect, the intervals on L that each triangle makes must overlap one another (as depicted in Figure 5.5(a)). If the intervals do not overlap, the triangles do not intersect (as depicted in Figure 5.5(b)). The objective of the remainder of the intersection test is therefore to compute the intervals on L for each triangle.



(a) Intersecting triangles



(b) Non-intersecting triangles

Figure 5.5: Figure illustrating the relationship between the interval on L that each triangle makes, and whether or not they intersect one another.

Triangle $T^{\mathcal{B}}$ comprises of vertices $V_1^{\mathcal{B}}$, $V_2^{\mathcal{B}}$ and $V_3^{\mathcal{B}}$, where vertices $V_1^{\mathcal{B}}$ and $V_2^{\mathcal{B}}$ lie on the same side of $\pi^{\mathcal{A}}$ (i.e. their signed distances to $\pi^{\mathcal{A}}$ have the same sign), and vertex $V_3^{\mathcal{B}}$ lies on the other side of $\pi^{\mathcal{A}}$. The edge connecting $V_1^{\mathcal{B}}$ to $V_3^{\mathcal{B}}$ intersects L at the point denoted E (see Figure 5.6).

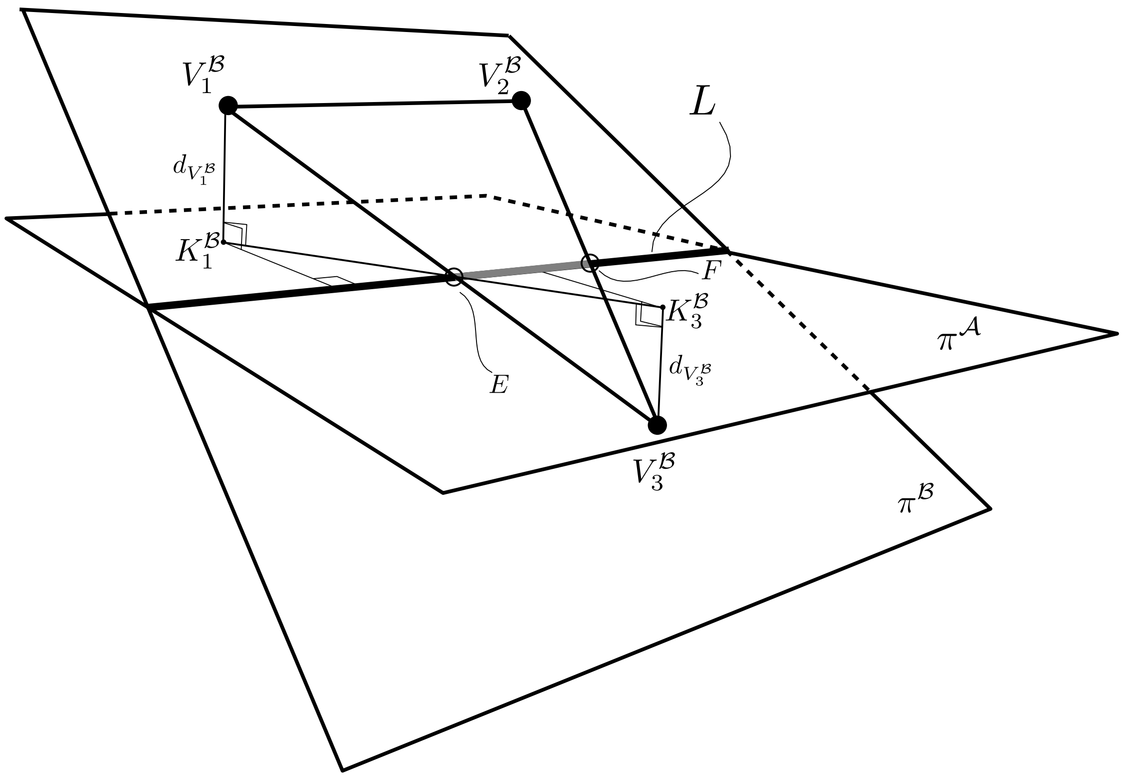


Figure 5.6: The geometry of triangle $T^{\mathcal{B}}$ three-dimensional space. $V_i^{\mathcal{B}}$ are the vertices of $T^{\mathcal{B}}$, $K_i^{\mathcal{B}}$ are the projections of these vertices onto $\pi^{\mathcal{A}}$. Points E and F are the points at which the lines $\overline{V_1^{\mathcal{B}}V_3^{\mathcal{B}}}$ and $\overline{V_2^{\mathcal{B}}V_3^{\mathcal{B}}}$ respectively intersect the plane $\pi^{\mathcal{A}}$. Finally, the line L is the (infinite) line of intersection between the planes $\pi^{\mathcal{A}}$ and $\pi^{\mathcal{B}}$.

Let $K_i^{\mathcal{B}}$ denote the projection of $V_i^{\mathcal{B}}$ onto the plane $\pi^{\mathcal{A}}$. It can be readily seen that the Euclidean distance from $V_1^{\mathcal{B}}$ to E is proportional to the Euclidean distance from $V_1^{\mathcal{B}}$ to $K_1^{\mathcal{B}}$ (denoted $d_{V_1^{\mathcal{B}}}$), and also that the Euclidean distance from $V_3^{\mathcal{B}}$ to E is proportional to the Euclidean distance from $V_3^{\mathcal{B}}$ to $K_3^{\mathcal{B}}$ (denoted $d_{V_3^{\mathcal{B}}}$). It can also be seen that the triangles $\triangle V_1^{\mathcal{B}}EK_1^{\mathcal{B}}$ and $\triangle V_3^{\mathcal{B}}EK_3^{\mathcal{B}}$ are similar, therefore the constant of proportionality is *the same* for both cases.

The distances $d_{V_i^{\mathcal{B}}}$ are already known (they had been computed previously from Equation 5.1a). Using the fact that $\triangle V_1^{\mathcal{B}}EK_1^{\mathcal{B}}$ and $\triangle V_3^{\mathcal{B}}EK_3^{\mathcal{B}}$ are similar, the coordinates of point E can be readily computed through interpolation between points $V_1^{\mathcal{B}}$ and $V_3^{\mathcal{B}}$ (linear interpolation will be exact since the line $\overline{V_1^{\mathcal{B}}V_3^{\mathcal{B}}}$ is straight):

$$E = V_1^{\mathcal{B}} + \left(V_3^{\mathcal{B}} - V_1^{\mathcal{B}}\right) \frac{d_{V_1^{\mathcal{B}}}}{d_{V_1^{\mathcal{B}}} - d_{V_3^{\mathcal{B}}}} \tag{5.3}$$

The values of x_0, y_0 and z_0 in Equation 5.2a can then be set to equal E (this is valid since (x_0, y_0, z_0) is *any* point on L, and E is a point on L). The corresponding parameter value at point E (denoted t_E), is therefore 0 ($t_E = 0$ from Equation 5.2a). All remaining parameter values at other points along L will therefore be specified relative to this point.

The coordinates of point F in Figure 5.6 are then found in the same way as that of point E (i.e. via interpolation between vertices $V_2^{\mathcal{B}}$ and $V_3^{\mathcal{B}}$). The corresponding parameter value at point F (t_F), can then found from Equation 5.2a.

The scalar interval on L that triangle $T^{\mathcal{B}}$ makes has now been identified ($t_E \leq t \leq t_F$). What remains is to find the scalar interval that triangle $T^{\mathcal{A}}$ makes on L. This is achieved via the same method as that of triangle $T^{\mathcal{B}}$. Once both scalar intervals have been computed, all that remains is to test if the two intervals overlap. If the two intervals overlap, the triangles intersect one another, and vice versa.

Note that, in order to reduce the required computational effort, the triangles $T^{\mathcal{A}}$ and $T^{\mathcal{B}}$, and the line L can be projected onto the Cartesian coordinate plane which L is most closely aligned (i.e. the plane that maximises the projected interval lengths). This projection is valid since if the intervals on L overlap, then the projected intervals will also overlap (and vice versa). The projected intervals are then computed via the method outlined above. This reduces the three-dimensional problem to a simpler two-dimensional one.

5.4 Inter-Grid Interpolations

The goal of the inter-grid interpolation algorithm is to transfer information from a donor mesh to provide the boundary conditions for a receiving grid. There are several different methods of achieving this, many of which have been briefly introduced in Chapter 2. In this section, numerical details of both linear interpolation and the semi-conservative Mass-Flux Based Interpolation (MFBI) method of Tang *et. al.* [1] shall be considered.

5.4.1 Linear Interpolation

Bilinear or trilinear interpolation can be used for the inter-grid information transfer in two or three spatial dimensions respectively. However, in general, the interpolation stencil will not be known prior to an interpolation being conducted. The following procedure is therefore carried out.

Suppose we wish to determine an interpolated value of ϕ at the point $P = (x, y, z)$. To start, a three-dimensional interpolation stencil is defined as the hexahedron that has vertices that are coincident with the *cell centres* of the underlying donor cells (the cell centres are used since we wish to interpolate from the cell centre values on the donor grid), and that point P lies within. The interpolation stencil will therefore consist of eight donor cells, each contributing an octant to the overall interpolation stencil. The definition of the interpolation stencil is further specified by stating that, if cell (i, j, k) is one donor cell contributing to the interpolation stencil, then the other donor cells will be cells $(i + 1, j, k)$, $(i + 1, j + 1, k)$, $(i + 1, j + 1, k + 1)$, $(i + 1, j, k + 1)$, $(i, j + 1, k + 1)$, $(i, j+1, k)$ and $(i, j, k+1)$. That is, interpolation stencils that are larger than necessary (for example, comprising of the cell $(i + 2, j, k)$) are not permitted.

Close to, or on computational boundaries, there may not be any interpolation stencil available. To resolve this, halo cells are used on all boundaries, which may form part of the interpolation stencil, thereby making it possible to interpolate at any point in the computational domain, even points that lie on boundaries. For some types of computational boundary, these halo cells may have a geometry that is a mirror of the adjoining boundary cell. Alternatively, they may have zero volume and be essentially a quadrilateral shape that is identical to the quadrilateral boundary face.

For a given donor grid, there are $(Ni+1) \times (Nj+1) \times (Nk+1)$ candidate interpolation stencils (where Ni, Nj and Nk are the number of cells in the donor grid, excluding halo cells, in the i, j and k directions respectively). While several of these interpolation stencils can be immediately discarded, due to them comprising of unused or interpolation donor cells (see Section 5.2), there will, in general, still be several candidate interpolation stencils remaining. The only valid interpolation stencil is the one that point P lies within. If any other stencil were to be used, then a linear extrapolation would be performed rather than a linear interpolation. Extrapolations should always be avoided since they are guaranteed to generate unbounded solutions for a one-dimensional extrapolation (assuming there is actually a variation in ϕ between the donor cells), and are hence very likely to generate unbounded solutions in two or three spatial dimensions. This is particularly problematic if negative turbulence quantities are generated through an unbounded extrapolation.

Determining which interpolation stencil that point P lies within, from all of the $(Ni+1) \times (Nj+1) \times (Nk+1)$ available stencils, can be a very costly process, especially for non-Cartesian grids. In order to identify the correct interpolation stencil, one possible algorithm would be to visit each candidate interpolation stencil in turn, splitting each quadrilateral face of the interpolation stencil into two triangles. The three vertices of one of these triangles is then combined with the point P to form a tetrahedron. By repeating this for all twelve triangles, twelve tetrahedral volumes are identified. If the sum of the volumes of all twelve tetrahedrons is equal to the volume of the hexahedral candidate interpolation stencil, then the point P is inside the interpolation stencil. Alternatively, if the sum of the volumes of the twelve tetrahedrons is greater than the volume of the interpolation stencil, the point P is outside of the candidate interpolation stencil, and the next candidate interpolation stencil should be tested.

For a donor grid comprising of millions of candidate interpolation stencils, computing the volume of twelve tetrahedrons and one hexahedron for each can quickly become prohibitively expensive. Furthermore, the above algorithm can fail if P is close to one of the candidate interpolation stencil's quadrilateral sides. This failure is due to the assumed piece-wise planar quadrilateral sides (consisting of two triangles), while the actual sides of the interpolation stencil may be non-planar. There may therefore be gaps between adjacent assumed piece-wise planar candidate interpolation stencils. If P lies within one of these gaps, there will be no available interpolation stencil. Note that, while other algorithms are available, the other algorithms considered suffered a similar fate when P lies close to one of the faces of the stencil. For this reason, as well as due to the high computational costs associated with the volume computations, the identification of the actual interpolation stencil is not attempted prior to computing interpolation coefficients. Instead, all candidate interpolation cells on the donor grid whose bounding box overlaps the point P are identified (where the bounding box axes are aligned with the Cartesian coordinate system). Details of the algorithm used to efficiently achieve this will be given in Section 5.5. For Cartesian grids, the bounding box test will return the actual interpolation stencil. However, in general non-Cartesian grids, there may be more than one candidate interpolation stencil whose bounding box overlaps the point P, but the total number of candidate stencils identified will usually be low (for stencils with an aspect ratio of around unity, a *maximum* of eight candidate stencils will be identified).

For the first of the candidate stencils identified via the bounding box test, interpolation coefficients are computed. To find the interpolation coefficients, a trilinear interpolation would give expressions of the form;

$$x = \sum_{i=1}^{8} N_i \overline{x}_i \qquad y = \sum_{i=1}^{8} N_i \overline{y}_i \qquad z = \sum_{i=1}^{8} N_i \overline{z}_i \tag{5.4}$$

where $(\overline{x}_i, \overline{y}_i, \overline{z}_i)$ are the coordinates of the vertices that make up the candidate interpolation stencil, and the N's are evaluated from the interpolation coefficients (φ, χ and ψ) by;

$$\begin{aligned}
N_1 &\equiv (1-\varphi)(1-\chi)(1-\psi) & N_2 &\equiv \varphi(1-\chi)(1-\psi) \\
N_3 &\equiv (1-\varphi)\chi(1-\psi) & N_4 &\equiv (1-\varphi)(1-\chi)\psi \\
N_5 &\equiv \varphi \cdot \chi(1-\psi) & N_6 &\equiv \varphi(1-\chi)\psi \\
N_7 &\equiv (1-\varphi)\chi \cdot \psi & N_8 &\equiv \varphi \cdot \chi \cdot \psi
\end{aligned} \tag{5.5}$$

In the definitions given by Equation 5.5, the values φ, χ and ψ are the unknown interpolation coefficients that we wish to determine. We therefore have a system of three nonlinear equations (i.e. Equations 5.4) involving the three unknown interpolation coefficients.

Letting $F_i(\varphi, \chi, \psi)$ denote the functional relations to be zeroed (i.e. the relations $\sum_{k=1}^{8} N_k \overline{x}_k - x$, and similarly for the y and z equations), a Taylor series expansion about the point P yields:

$$F_i(\varphi+\delta\varphi, \chi+\delta\chi, \psi+\delta\psi) = F_i(\varphi, \chi, \psi) + \frac{\partial F_i}{\partial \varphi}\delta\varphi + \frac{\partial F_i}{\partial \chi}\delta\chi + \frac{\partial F_i}{\partial \psi}\delta\psi + \mathcal{O}(\delta\varphi^2) + \mathcal{O}(\delta\chi^2) + \mathcal{O}(\delta\psi^2) \tag{5.6}$$

By neglecting second order and higher terms, and by setting $F_i(\varphi+\delta\varphi, \chi+\delta\chi, \psi+\delta\psi)$ to zero, a set of linear equations for the corrections ($\delta\varphi$, $\delta\chi$ and $\delta\psi$), that move the function closer to zero are obtained:

$$\delta\varphi = \frac{-F_i(\varphi,\chi,\psi) - \frac{\partial F_i}{\partial \chi}\delta\chi - \frac{\partial F_i}{\partial \psi}\delta\psi)}{\frac{\partial F_i}{\partial \varphi}} \tag{5.7}$$

$$\delta\chi = \frac{-F_i(\varphi,\chi,\psi) - \frac{\partial F_i}{\partial \varphi}\delta\varphi - \frac{\partial F_i}{\partial \psi}\delta\psi)}{\frac{\partial F_i}{\partial \chi}} \tag{5.8}$$

$$\delta\psi = \frac{-F_i(\varphi,\chi,\psi) - \frac{\partial F_i}{\partial \varphi}\delta\varphi - \frac{\partial F_i}{\partial \chi}\delta\chi)}{\frac{\partial F_i}{\partial \psi}} \tag{5.9}$$

Once the corrections have been obtained, the variables φ, χ and ψ are updated via;

$$\varphi^{NEW} = \varphi^{OLD} + \delta\varphi \tag{5.10}$$

$$\chi^{NEW} = \chi^{OLD} + \delta\chi \tag{5.11}$$

$$\psi^{NEW} = \psi^{OLD} + \delta\psi \tag{5.12}$$

and the process is repeated in an iterative manner. This process is essentially the Newton-Raphson method, applied to nonlinear systems of equations.

More often than not, the Newton-Raphson method converges remarkably quickly (tests conducted in this project have shown that typically only 4-5 iterations are required to bring the residuals of Equations 5.4 to negligible levels, with each iteration requiring only around 150 floating point operations). Convergence can be slower for interpolation stencils that are of very high aspect-ratio, since in this case the residual of one equation would have to be reduced by several orders of magnitude lower than its convergence criterion in order to further reduce the residuals of the other equations. For exceptionally high aspect ratio interpolation stencils, this could pose a problem, since if the residuals of two of the three equations reaches machine accuracy, the other equation will not change (even if not converged). However, it has been found that typical aspect ratios for which this problem arises are significantly higher than those which would be used in practical computations, so the issue is of little concern.

For the point (x, y, z) to lie within the interpolation stencil that was used above, the values of φ, χ and ψ must all lie between 0 and 1. If $(0 \leq \varphi \leq 1)$, $(0 \leq \chi \leq 1)$ and $(0 \leq \psi \leq 1)$ all hold, then the correct interpolation stencil was assumed above and we have found the required values of the interpolation coefficients. However, if any of the interpolation coefficients lie outside the values of 0 to 1, a new candidate interpolation stencil is tested. The new interpolation stencil will be the next candidate interpolation stencil identified by the bounding box test. The nonlinear system (Equations 5.4) is then re-evaluated based on the new interpolation stencil. This is repeated until the correct interpolation stencil is found.

It can be shown that the convergence rate of Newton's method is quadratic, provided the initial guess is sufficiently close to the final solution [55]. In the present code, an initial guess of 0.5 has been used for all three variables. This was selected since for the correct interpolation stencil, the values of all three variables will lie between 0 and 1. If the correct interpolation stencil was not selected above, the iterative solution may diverge (detected by the residuals exceeding a predefined value), or the number of iterations to reach convergence may be high. To avoid wasted computation time, the iterative procedure should be terminated as soon as divergence is detected, or once the number of iterations exceeds the maximum permissible number (set to 15 iterations in the present study). The maximum number of permissible iterations should be set high enough to ensure than the correct interpolation stencil is not erroneously discarded due to non-convergence, whilst also ensuring that the number is low enough to avoid excessive computation. The value of up to 15 iterations was chosen to satisfy these requirements, but may not be universally applicable (particularly for high aspect ratio grids). The value is therefore user adjustable, however the default value has been found to work for all cases considered in the present study.

Further computation time can be saved by terminating the iteration process early if values of the interpolation coefficients are outside of the range of 0 to 1 (i.e. the wrong interpolation stencil was selected), even if the system is converging. This is implemented by defining an initial convergence criteria, which is several orders of magnitude larger than the ultimate convergence criteria. Once the residuals satisfy the initial convergence criteria, further iterations are only performed if the correct interpolation stencil has been identified.

Once the correct interpolation stencil, and the associated values of the interpolation coefficients, have been identified (where each coefficient lies between 0 and 1), the interpolated value of any quantity, ϕ, at point P is given by;

$$\phi \approx \sum_{i=1}^{8} N_i \overline{\phi}_i \tag{5.13}$$

By finding the interpolation stencil and interpolation coefficients through the above procedure, rather than identifying the correct stencil beforehand, the overall algorithm has been found to be quick and robust (even when P lies on a face of the interpolation stencil).

Where more than one grid is available for interpolation, interpolation is conducted from the donor grid whose donor cells are a closest match, in terms of size, to the cell size of the receiving grid at point P. This minimises any unnecessary heterogeneity in cell size, thereby improving solution continuity between sub-grids.

5.4.2 Mass-Flux based interpolation

Linear interpolation is, in general, non-conservative (unless the grids are aligned in a very limited number of specific ways). In an attempt to enforce the conservation of mass over any arbitrary arrangement of overset grids, the Mass Flux Based Interpolation method (MFBI) can be used. The MFBI method of Tang *et. al.* [1] has already been introduced in Chapter 2, where the basic idea and selected results from the method were outlined. In this section, implementation details of the method are presented. To start, the necessary conditions required for the conservation of mass over overset grids are derived. However, it has been noted in Chapter 2, and in [1, 17], that the enforcement of the conditions is very difficult (if not impossible) to implement for a general overset grid, especially in three dimensions. A second order approximation to this condition is therefore enforced in the MFBI method, hence the MFBI method is, strictly speaking, only fully conservative in the limit of zero grid spacing (as are all other interpolation methods at this rather impractical limit). The method is hence referred to as semi-conservative. Despite the strictly non-conservative nature of the MFBI method, satisfactory results have been reported in [1], and the results of the present code seem to mirror this success.

The main issue with standard linear interpolation, with regards to conservation issues, is not due to the order of accuracy of the interpolation method (linear interpolation is second order accurate with respect to grid sizing), but rather is due to the locations where the interpolations are carried out. The requirement of treating the interpolation boundaries in a manner that is consistent with the underlying discretisation scheme for standard cells, necessitates the use of halo cells (or ghost cells) at the interpolation boundaries (see Section 5.7 for further details). Interpolating at the halo cell centres, and then constructing the mass flux through the interpolation boundary (via a weighted average of the halo and adjoining cell's velocity values), generally leads to a non-conservative formulation since there is no guarantee that the mass flux entering through one interpolation boundary face will be balanced by mass flux through other boundaries. If one were to interpolate for the mass flux at the interpolation boundary (in addition to interpolating for the primitive variables at the halo cells), the conservation of mass could be assured (to within second order accuracy), however there would be inconsistencies between the primitive variables at the cell centres, and the mass flux through the cell's face. These inconsistencies can prevent convergence. The goal of the MFBI method is to derive interpolated values for the three velocity components at the cell centre of interpolation cells, that when averaged onto the interpolation boundary, result in a conservative mass flux through the boundary.

Conditions for global mass conservation on overset grids

Consider the overset domain sketched in Figure 5.7. In this figure, the overall domain, Ω, is delineated by the union of the subdomains, Ω_A and Ω_B (that is $\Omega = \Omega_A \cup \Omega_B$). The boundary of Ω is labelled Γ, which can be written as;

$$\Gamma = \Gamma_A - \Gamma_a + \Gamma_B - \Gamma_b$$

where Γ_A is the boundary of the subdomain Ω_A, and Γ_a is the portion of Γ_A that is overlapped by Ω_B (and similarly of Γ_B and Γ_b). This is illustrated in Figure 5.7.

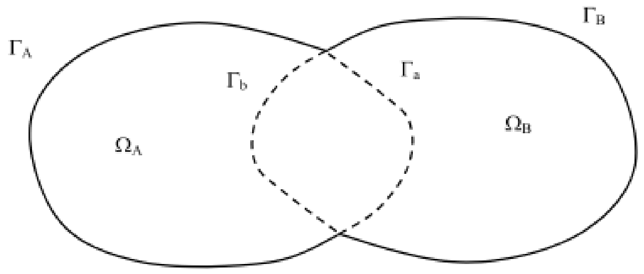


Figure 5.7: A sketch of the grids used to derive the conservative interface conditions.

Global mass conservation requires that;

$$\int_{\Gamma} \vec{\mathbf{V}} \cdot \vec{\mathbf{n}} \, \mathrm{d}S = \int_{\Gamma_A} \vec{\mathbf{V}}^A \cdot \vec{\mathbf{n}} \, \mathrm{d}S - \int_{\Gamma_a} \vec{\mathbf{V}}^A \cdot \vec{\mathbf{n}} \, \mathrm{d}S + \int_{\Gamma_B} \vec{\mathbf{V}}^B \cdot \vec{\mathbf{n}} \, \mathrm{d}S - \int_{\Gamma_b} \vec{\mathbf{V}}^B \cdot \vec{\mathbf{n}} \, \mathrm{d}S = 0 \qquad (5.14)$$

where $\vec{\mathbf{V}}$ is the velocity vector, and $\vec{\mathbf{n}}$ is the unit vector in the wall-normal direction, pointing outward. The superscripts A and B indicate which sub-domain the velocity vector is computed from.

Since conservation over each sub-domain is assured in the finite volume method, we have;

$$\int_{\Gamma_A} \vec{\mathbf{V}}^A \cdot \vec{\mathbf{n}}\, \mathrm{d}S = 0 \qquad\qquad \int_{\Gamma_B} \vec{\mathbf{V}}^B \cdot \vec{\mathbf{n}}\, \mathrm{d}S = 0 \tag{5.15}$$

Combining the results of Equations 5.14 and 5.15 one obtains;

$$\int_{\Gamma_a} \vec{\mathbf{V}}^A \cdot \vec{\mathbf{n}}\, \mathrm{d}S + \int_{\Gamma_b} \vec{\mathbf{V}}^B \cdot \vec{\mathbf{n}}\, \mathrm{d}S = 0 \tag{5.16}$$

Since the boundary $\Gamma_a + \Gamma_b$ forms a closed loop, the net mass flux through this loop must be zero for physically relevant solutions. Combining this with the fact that the overlap region belongs to both sub-domains, hence the net mass flux through $\Gamma_a + \Gamma_b$ should be zero when computing the velocity vector from either sub-domain, gives us the following conservation relations;

$$\begin{gathered}\int_{\Gamma_a} \vec{\mathbf{V}}^A \cdot \vec{\mathbf{n}}\, \mathrm{d}S + \int_{\Gamma_b} \vec{\mathbf{V}}^A \cdot \vec{\mathbf{n}}\, \mathrm{d}S = 0 \\ \int_{\Gamma_a} \vec{\mathbf{V}}^B \cdot \vec{\mathbf{n}}\, \mathrm{d}S + \int_{\Gamma_b} \vec{\mathbf{V}}^B \cdot \vec{\mathbf{n}}\, \mathrm{d}S = 0\end{gathered} \tag{5.17}$$

Combining the results of (5.16) and (5.17) yields the following conditions for the global conservation of mass;

$$\begin{gathered}\int_{\Gamma_a} \vec{\mathbf{V}}^A \cdot \vec{\mathbf{n}}\, \mathrm{d}S = \int_{\Gamma_a} \vec{\mathbf{V}}^B \cdot \vec{\mathbf{n}}\, \mathrm{d}S \\ \int_{\Gamma_b} \vec{\mathbf{V}}^A \cdot \vec{\mathbf{n}}\, \mathrm{d}S = \int_{\Gamma_b} \vec{\mathbf{V}}^B \cdot \vec{\mathbf{n}}\, \mathrm{d}S\end{gathered} \tag{5.18}$$

Equations 5.18 state that, for the global conservation of mass over a set of overset grids, the same mass flux through an internal boundary must be obtained, regardless of which domain is used to compute the velocity vector. However, the enforcement of this condition is extremely difficult for general grids, especially in 3D.

The semi-conservative approach

To demonstrate the MFBI method, consider the grids in Figure 5.8. For the purpose of this discussion, we shall assume the grids in this figure are 3D and extend in the ζ direction, into the page (with k indexing the nodes in this direction). The method can just as easily be applied in 2D if desired. In the MFBI method, a discrete approximation to Equation 5.18 is enforced on S, where S is the interpolation boundary of the grid. The cells at $i = 1$ are extra 'halo' cells that have been added to the grid in order to store the boundary variables.

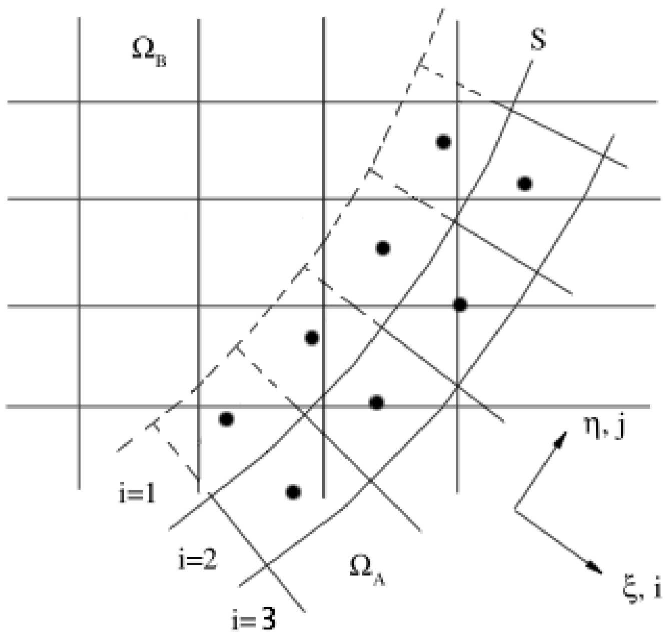


Figure 5.8: The 3D grids used to demonstrate the MFBI method. Grids are extend in the ζ, k direction into the page.

A semi-discrete (i.e. partially discretised), second-order approximation to the condition in Equation 5.18 can be made through the mid-point rule, which yields;

$$\sum_{j,k\in S} \left(\frac{U^1}{J}\right)^A_{3/2,j,k} \Delta\eta\Delta\zeta \approx \int_q \left(\frac{U^1}{J}\right)^B_q \,\mathrm{d}\eta\;\mathrm{d}\zeta \tag{5.19}$$

where U^1 is the contravariant velocity component in the direction normal to the internal boundary S. Note that the right hand side of Equation 5.19 is not discritised at this stage for reasons that will become apparent. The summation on the left hand side of Equation 5.19 is conducted over each cell face of Ω_A along the boundary S. The integral on the right hand side of Equation 5.19 is conducted over the infinite number of points (q), that make up the intersection of the surface (S) with the Ω^B subdomain.

Computing the right hand side of 5.19 would require the integrand be known at all points, q. The enforcement of this condition (or even a discretised version of the condition) is extremely difficult. Hence Tang *et. al.* [1] proposed the following approximation;

$$\sum_{j,k\in S}\left(\frac{U^1}{J}\right)^A_{3/2,j,k}\Delta\eta\Delta\zeta = \sum_{j,k\in S}\left[\frac{\Im^A_B\,(u)^1}{J}\right]_{3/2,j,k}\Delta\eta\Delta\zeta \tag{5.20}$$

where $\Im^A_B()^i$ is the operator that interpolates the Cartesian velocity components from subdomain Ω_B to subdomain Ω_A (at the centre of the faces making up S), and then computes the resulting contravariant velocity component, U^i (in this case the contravariant velocity component is in the direction 1, normal to the internal boundary). Linear interpolation was used by Tang *et. al.* to interpolate the Cartesian velocity components (i.e. the method discussed in Section 5.4.1 can be used). In Equation 5.20, rather than computing the mass flux through S from subdomain Ω_B exactly (which would require the integrand of Equation 5.19 be known at all the points q), the velocity components are interpolated from Ω_B onto the centre of the faces of Ω_A that make up S. These interpolated values are then used to construct an approximate mass flux from Ω_B.

Clearly, Equation 5.20 can be satisfied by setting the U^1 contravariant velocity component on subdomain Ω_A to the contravariant velocity computed from the interpolated Cartesian velocity components. In other words;

$$\left(\frac{U^1}{J}\right)^A_{3/2,j,k} = \left[\frac{\Im^A_B\,(u)^1}{J}\right]_{3/2,j,k} \tag{5.21}$$

In order to find the boundary values of the contravariant velocity (U^1 at $i = 1$), Equation 5.21 needs to be approximated in terms of its value at surrounding nodes. To achieve this, the term

$\left(\frac{U^1}{J}\right)^A_{3/2,j,k}$ in equation 5.21 is approximated to second order accuracy using a central difference interpolation. This is consistent with the way the net mass imbalance ($\dot{m}^*$) is evaluated in the SIMPLE algorithm, since Rhie-Chow velocity interpolation is not applied to the boundary faces. Through a central difference interpolation, the following is obtained;

$$\left(\frac{U^1}{J}\right)^A_{3/2,j,k} \approx \frac{1}{2}\left[\left(\frac{U^1}{J}\right)^A_{1,j,k} + \left(\frac{U^1}{J}\right)^A_{2,j,k}\right] \approx \left[\frac{\Im^A_B\,(u)^1}{J}\right]_{3/2,j,k} \tag{5.22}$$

or, upon rearranging;

$$\left(\frac{U^1}{J}\right)^A_{1,j,k} = 2\left[\frac{\Im^A_B\,(u)^1}{J}\right]_{3/2,j,k} - \left(\frac{U^1}{J}\right)^A_{2,j,k} \tag{5.23}$$

Equation 5.23 imposes a restriction to the contravariant velocity component, U^1 that must be satisfied for the discrete approximation of the conservation of mass to hold.

The remaining contravariant velocity components, U^2 and U^3, and the pressure, are linearly interpolated onto the boundary node as follows;

$$\left(\frac{U^2}{J}\right)^A_{1,j,k} = \left[\frac{\Im^A_B\,(u)^2}{J}\right]_{1,j,k} \tag{5.24}$$

$$\left(\frac{U^3}{J}\right)^A_{1,j,k} = \left[\frac{\Im^A_B\,(u)^3}{J}\right]_{1,j,k} \tag{5.25}$$

$$(P)^A_{1,j,k} = \left[\Im^A_B\,(P)\right]_{1,j,k} \tag{5.26}$$

Other problem specific flow variables, such as k and ϵ for example, are linearly interpolated in a manner similar to that of the pressure.

Equations 5.23, 5.24 and 5.25 are three equations involving the three contravariant velocity components. In general, the three contravariant velocities will depend on the three Cartesian velocity components. It is these Cartesian velocity components that need to be specified at the boundary. Hence, Equations 5.23, 5.24 and 5.25 can instead be interpreted as representing a system of three (linear) equations involving the three unknown Cartesian velocity components. This system is inverted, and the resulting (semi-) mass conserving velocity components are used as boundary conditions.

The MFBI method, discussed above, can be thought of as linearly interpolating two of the three Cartesian velocity components (and the pressure), and then deriving constraints on the third velocity component that conserves mass.

5.5 Binary search trees, Alternating digital trees, and geometric intersection testing

It has been seen in the preceding sections that determining which interpolation stencil point $\mathbf{P}$ lies within, out of potentially millions of candidate interpolation stencils, is a frequently encountered problem in the interpolation algorithm. A first step in solving this problem is to find all interpolation stencils that have a bounding box that intersects the point. Hence, one could simply visit each candidate interpolation stencil in turn, and generate a bounding box for the given interpolation stencil (note that each bounding box has extremities that are coincident with those of the interpolation stencil, and has axis that are Cartesian aligned). By then testing if $\mathbf{x}_{min} \leq \mathbf{P} \leq \mathbf{x}_{max}$ holds, one can determine if the current interpolation stencil has a bounding box that engulfs point $\mathbf{P}$ (where $\mathbf{x}_{min}$ and $\mathbf{x}_{max}$ are the position vectors of the minimum and maximum extremities of the bounding box respectively). In three spatial dimensions, this test requires six comparisons for each candidate bounding box, in addition to the forty-two comparisons required in constructing the bounding box (comprising of seven `min()` function calls for each of the three spatial dimensions, in addition to seven `max()` function calls for each spatial dimension, assuming each call returns the minimum, or maximum, of two numbers). For millions of candidate bounding boxes, this exhaustive algorithm would be costly.

A related algorithm, that has application in hole cutting (see Section 5.2), is to determine which bounding box or boxes, out of potentially millions of candidate boxes, a given *bounding box* intersects. In this case, the inequalities $\mathbf{x}^A_{min} \leq \mathbf{x}^B_{max}$ and $\mathbf{x}^A_{max} \geq \mathbf{x}^B_{min}$ must both be satisfied (where the superscripts, A and B, denote the current candidate box and the given test box respectively). Again, an exhaustive algorithm would be costly for large grids.

An alternative to these exhaustive algorithms is to use an Alternating Digital Tree (ADT) search algorithm, as proposed by Bonet *et. al.*, [56]), which is similar to the binary search tree method commonly used in various branches of computer science, but with the important difference that the ADT algorithm has multi-dimensional geometric searching applications. In the remainder of this section, the various search algorithms will be presented. Binary search trees will first be introduced in Section 5.5.1, as they form the basis of the ADT search algorithm. A basic ADT algorithm will then be presented in Section 5.5.2, which can be used to quickly find a given point from a list of millions of candidate points. While this ADT search algorithm has no practical use in the overset code considered here, it does form the basis of the more advanced ADT method involving geometric intersection, which *is* used in the present code. Finally, the geometric intersection ADT algorithm, which is used for determining which bounding box (or boxes) intersect a given test *bounding box*, will be presented in Section 5.5.3. Note that this latter algorithm can also be used for finding all bounding boxes that engulf a given *point*, as is frequently required for the interpolation algorithm, simply be setting the test box to a singular point (or rather, in order to avoid the singularity, a negligibly small box).

5.5.1 Binary Search Trees

A binary search tree is a non-sequential data structure that can be used to search through lists in $\mathcal{O}(\log n)$ time at best (but $\mathcal{O}(n)$ time at worst). An exhaustive algorithm, on the other hand, would search through lists in $\mathcal{O}(n)$ time for all cases. The binary tree method therefore offers significant cost savings over an exhaustive search for most cases, especially for those involving large datasets, and in any case, is no worse than an exhaustive search.

The data items in a binary search tree are stored non-sequentially in the computer's memory. That is, any given data item can be stored at any arbitrary address in computer memory, with no logical link required between the tree's topological structure and the computational memory address locations. To achieve connectivity between the data, each data item is extended by the addition of two pointers, known as the left and right links, to form a *node*. The added links can point to the address of another node (where another data item, and associated pointer links are stored), or can be null (pointing to nothing). Every node, except for the first node inserted into the tree (known as the *root node*), will have exactly one link pointing to it. The root node will have no links pointing to it. The left and right links are connected to other nodes to form a tree-like structure, as illustrated in Figure 5.9. Figure 5.10 shows the topological structure of a simple tree, and one possible arrangement of the tree in computational memory. The nodes are arranged in a hierarchical manner, with the root node (i.e. node A in Figure 5.10(b)) having a hierarchy of 0. The nodes linked to the left and right of the root node, if present, have a hierarchy of 1 (i.e. nodes B and C in Figure 5.10(b)), and nodes linked to these nodes will have hierarchy 2 (i.e. nodes D and E in Figure 5.10), and so on. This hierarchy numbering will become useful later on in this section. By starting at the root node, one can traverse to any other node on the tree, simply by following either the left or right links. That is, there is a path from the root to every other node on the tree. The opposite, however, is not true; there is no way of traversing from node C say, back to the root, as node C has no knowledge of the memory address of node A (the left and right links point to nodes D and E respectively, whereas there are no links pointing to node A). All searches of the tree must therefore start at the root node.

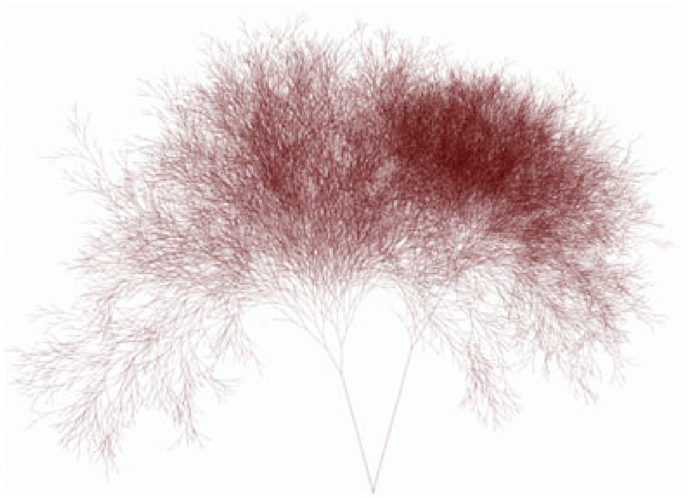


Figure 5.9: A large binary search tree.

The nodes of a binary search tree are inserted into the tree structure based on the values they contain. Any values that are part of a node's left sub-tree are always less than the node's value and any values in the right sub-tree are always greater than the node's value. It is this ordering that enables the quick searching through large datasets; if the value that is being searched for is less than the root value, one would traverse to the left sub-tree and immediately discard all nodes on the right sub-tree. One would then repeat this analysis on the left sub-tree, with the left branch of the sub-tree being traversed if the value being searched for is less than the sub-tree's root value, and vice versa. This continues until the value being searched for is *equal to* root value of the sub-tree being considered, at which point the value will have been found, and the search is complete. If a

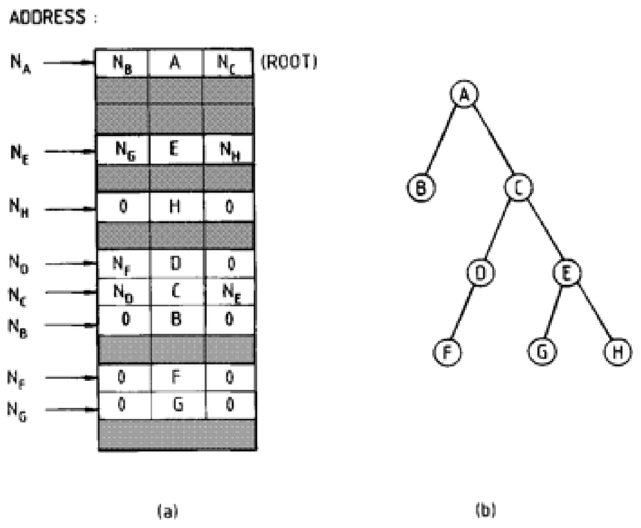


Figure 5.10: A simple binary search tree, and one possible arrangement of nodes in computer memory.

node with no links to the left or right is encountered (known as a leaf of the tree), and the value being searched for is not equal to this leaf's value, then the search fails, and one can deduce that the search item was not present anywhere in the tree (and hence was not present in the original dataset). An example follows in order to demonstrate the concept.

Tree generation, and node insertion

When creating a binary search tree, one starts with an unsorted list of data items that we wish to search through. In developing a virtual address book, for example, we may have a list of names. Lets say that this list comprises of the names `Vicky, Alex, Adam, John, Holly, Zofia, William` and `Tom`. The first node of the tree could be chosen arbitrarily, but usually the first data item of the list is chosen (in this case, `Vicky`). There is little to no advantage to be gained by selecting any other data item, so the first data item may as well be used. The first node, `Vicky`, is placed at the root of the tree.

When inserting the next data item from the list, one would insert to the left of the root if the next data item is *less than, or equal to* the root value, and one would insert to the right of the root if the next data item is *greater than* the root value. In the context of the virtual address book, 'less than' can simply be interpreted as coming alphabetically before, while 'greater than' implies alphabetically after. Hence, if the next data item is alphabetically before `Vicky`, the left branch is taken, while if the next data item is after `Vicky`, the right branch is taken. The next data item, `Alex`, would therefore be inserted to the left of `Vicky`.

When inserting the third data item, `Adam`, one would start at the root, and traverse to the left (since `Adam` is alphabetically before than `Vicky`). Since this location is already occupied (by the node `Alex`), the address of `Adam` cannot be stored here, so the process is repeated on the sub-tree where `Alex` is at the root. This time, the branch to the left of `Alex` will be traversed if the data item is less than (or equal to) `Alex`. Alternatively, the right branch will be traversed if the data item has a value that is greater than `Alex`. The node `Adam` will therefore be placed to the left of `Alex`.

This process is repeated until all data items of the list are inserted into the tree. The resulting tree for the case outlined above is shown in Figure 5.11.

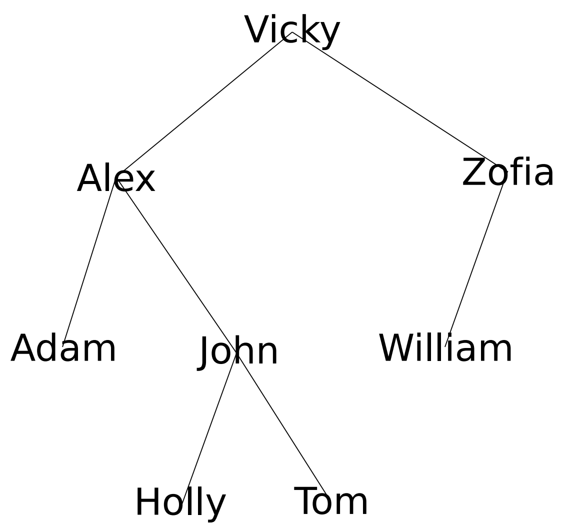


Figure 5.11: The resulting binary search tree for the virtual address book.

Binary Tree Performance

Each level down the tree (each increasing hierarchy) splits the data off into an ever decreasing fraction of the overall dataset. It can be seen that, if there are an equal number of nodes on the left and right branches of a tree, half of the data will be eliminated from the search at the first comparison. Similarly, if there is an equal number of nodes on the left and right branches of each subsequent subtree, half of the remaining data will be eliminated at each subsequent comparison. This ideal case is referred to as a perfectly balanced tree, and it is this case that leads to the optimum $\mathcal{O}(\log n)$ search time. As $n \to \infty$, most trees will approach a reasonably well balanced state, provided that there is a sufficient variation in the initial dataset, so search performances close to logarithmic time can usually be expected for large datasets.

Conversely, for trees where each node is connected to only one other node, the binary search tree offers no improvement over an exhaustive algorithm, and the tree is known as a degenerate tree. Possible layouts of such trees are shown in Figure 5.12. However, even for this worst case, the binary search performs no worse than an exhaustive algorithm, so there is nothing to lose in its use. Furthermore, these degenerate trees are thankfully very rare for large datasets, and can be easily avoided. In fact, degenerate trees will only occur if the initial dataset that is used to construct the tree is already ordered.

It can be noticed from the preceding description of the binary search tree, that the overall shape of the tree, and hence the performance of any subsequent searches, will be influenced by the order in which the nodes are inserted into the tree. Generally it is be a good idea to insert items that are expected to be searched for frequently before items that will be seldom searched for. This way, the frequently searched items will be near the root of the tree, and hence will be found quickly. However, in the search algorithms implemented in the present code, the frequency of searching for each stencil will not be known beforehand, so this optimisation method could not be attempted.

Theoretically, the order of inserting the nodes could also be tuned to provide a tree that is as balanced as is possible for a given dataset. However, the complexity and cost of optimising the tree in this way would likely outweigh any advantages, and hence was not considered in the present study.

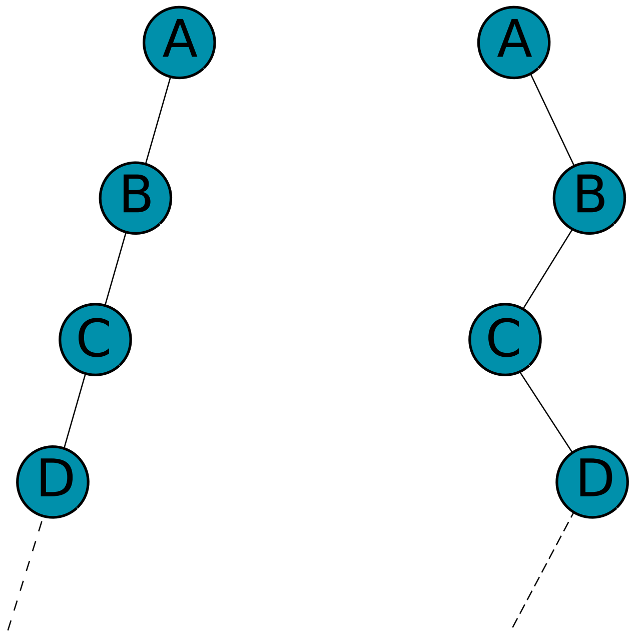


Figure 5.12: Degenerate Binary Search Trees.

Implementation of a binary search tree algorithm

The implementation of a binary search algorithm is reasonably straight forward in a programming language that supports recursion (where a function can call itself). Examples of such languages include Fortran 90, C and C++, among many others. A node insertion function, for example, would take as its arguments, the address of the node to be inserted, and the address of the root of the tree (or sub-tree). The function would then evaluate if the node can be placed at the root, or failing that, if the left or right branch should be traversed (based on the criteria discussed above). If the left branch is to be used, the insertion function would simply call itself, this time with the second argument (i.e. the new root) set to equal the address of the left sub-tree's root. A similar procedure is carried out to account for the cases where the right branches are to be traversed, thereby completing the node insertion function. Through the use of recursion, the node insertion function can be coded in only a few lines, as demonstrated by the following pseudo-code:

A search function can be coded in a manner that is very similar to that of the node insertion function, again taking advantage of recursive features of modern programming languages. In this case, one would start at the root and compare the root value's data with the data being searched

```
Function insertNode(node, root)
  begin
      if $root = NULL$ then
          root ← node's address
          return
      end
      if $node.data \leq root.data$ then
          call insertNode(node, root.left)
      else
          call insertNode(node, root.right)
      end
  end
```

for. If it is not found, one would traverse to the left or right branches (where the decision of left or right traversal is based on the criteria discussed in Section 5.5.1), and compare the data of the left or right sub-tree's root for equality with the data being searched for. The following pseudo-code demonstrates this concept:

```
Function search(data, root)
  begin
      if $data = root.data$ then
          return address of root
      end
      if $data \leq root.data$ then
          call search(data, root.left)
      else
          call search(data, root.right)
      end
  end
```

If developing a binary search algorithm in a language that does not support recursion, such as Fortran 77, a pseudo-recursive version of the algorithm could be developed. The basic concept behind the pseudo-recursive mechanism is to define two versions of the same function (e.g. `insertNode1` and `insertNode2`). The first version of the function will then call the second version, and vice versa. The code inside each function will be identical to that of the fully recursive version, except for the function calls, which would instead call the version of the function that would not result in recursion (see Reference [57] for further details of the pseudo-recursive method). While this method may seem a little messy, it is far simpler than some of the other methods available for non-recursive languages, which often require considerable bookkeeping.

5.5.2 The basic ADT method

An alternating digital tree search algorithm is similar to the binary search algorithm described in the previous section (in fact, the ADT method is a specific type of binary search tree). The unique features of an ADT algorithm, that differentiates it from the basic binary search algorithm, is in the criteria used in selecting the traversal of the left, or the right branches. In the basic ADT method, the dataset is a list of points in space. A link between the physical location of the dataset's points, and the position of the nodes on the tree is then established. Through the use of this link, a proportion of nodes can be eliminated from the search at each hierarchy down the tree, as was seen in the case of the basic binary search tree.

In setting up an ADT, we start with a list of points that we wish to search through (the dataset). In order to generate the tree, as for a binary search tree, the first data item in the dataset is extended via the addition of two pointers (the left and right links) to form a node. This node is then inserted at the root of the tree.

When inserting the remaining nodes, the left or right branches are traversed based on comparisons of the node's physical location in one coordinate direction, relative to that of the current subtree's root. The selection of the coordinate direction is based on the current hierarchy. For example, for a two-dimensional dataset, all sub-trees with a root at *even* hierarchy will branch based on the node's *x-coordinate*. The left branch will be traversed if the node's x-coordinate is less than, or equal to, that of the subtree's root, while the right branch will be traversed if the x-coordinate is greater than that of the subtree's root. Similarly, for all sub-tree's with a root at *odd* hierarchy, branches are traversed based on the node's *y-coordinate*. This is repeated, in an alternating order, with alternations between x and y coordinate direction comparisons made all the way down the tree. As an example, Figure 5.13 shows several points comprising of a sample dataset, with the resulting ADT for this particular case shown on the right (assuming that the nodes are inserted into the tree in the alphabetical order in which they are labelled).

For a three-dimensional dataset, comparisons of the x, y and z directions are made at each hierarchy in cyclic order. Comparisons of the *x-coordinate* direction are made at all hierarchies where $\frac{h}{3} - \lfloor \frac{h}{3} \rfloor$ is equal to zero (where h is the hierarchy, and $\lfloor\ \rfloor$ is the floor function[1]). Similarly, comparisons of the y and z coordinate directions are made at all hierarchies where the expression,

[1]The floor function returns the largest integer value that is not greater than its real argument. The expression $\frac{h}{3} - \lfloor \frac{h}{3} \rfloor$ is therefore equal to the remainder of the quotient $\frac{h}{3}$.

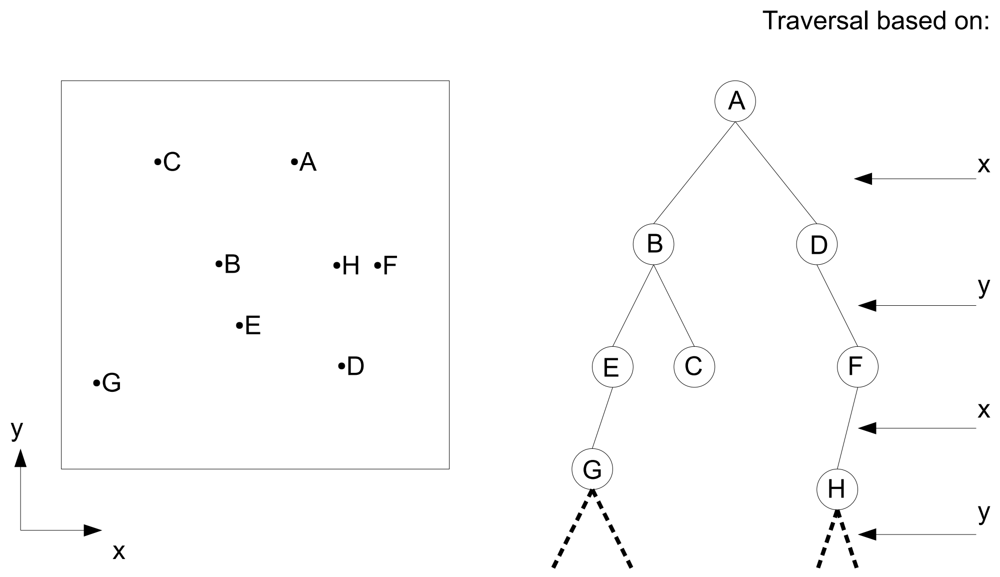


Figure 5.13: A two-dimensional sample dataset, and the resulting ADT.

$\frac{h}{3} - \lfloor \frac{h}{3} \rfloor$, is equal to $\frac{1}{3}$, or $\frac{2}{3}$ respectively. As an example, Table 5.1 shows the coordinate direction that any branching is based upon, for the first seven hierarchies.

h	$\frac{h}{3} - \lfloor \frac{h}{3} \rfloor$	Coordinate direction
0	0	x
1	$\frac{1}{3}$	y
2	$\frac{2}{3}$	z
3	0	x
4	$\frac{1}{3}$	y
5	$\frac{2}{3}$	z
6	0	x
$\vdots$	$\vdots$	$\vdots$

Table 5.1: Table showing the coordinate direction that traversal is based upon, for the first seven hierarchies.

More generally, for a n-dimensional dataset, comparisons in the i^{th}-coordinate direction are made at each hierarchy where $\frac{h}{n} - \lfloor \frac{h}{n} \rfloor = \frac{i-1}{n}$.

When implementing this ADT node insertion algorithm, each node must be aware of its own hierarchy in order facilitate the bisection process described above. One option would be to store the hierarchy of each node within the node itself. However, this would result in a waste of computational storage space. An alternative is to instead take advantage of the fact that all searches (or node insertions), originate from the root of the tree, where the hierarchy is zero, and that on each traversal, the hierarchy is increased by one. By incrementing the current hierarchy by one, and passing the result as an argument to the recursive node insertion function on each tree traversal, it is not necessary to store each node's hierarchy. The following pseudo-code demonstrates this.

```
Function ADTinsert(node, root, h)
begin
    if $root = NULL$ then
        root ← node's address
        return
    end
    i = $\left(\frac{h}{3} - \lfloor\frac{h}{3}\rfloor\right)$
    if $node.x^i \leq root.x^i$ then
        call ADTinsert(node, root.left, h+1)
    else
        call ADTinsert(node, root.right, h+1)
    end
end
```

Once the tree has been generated, searching for a point starts at the root (as was seen for the binary search tree), and follows the same traversal rules as outlined for the node insertion algorithm, until the required node is found. By following these rules, any given node can be identified in close to logarithmic time, provided the tree is reasonably well balanced.

5.5.3 Extending the basic ADT method to solve geometric intersection problems.

The basic ADT search method of Section 5.5.2 quickly returns a specific node from a list of nodes. While this basic method is useful for some applications, the method itself is not particularly useful for the overset algorithms developed for this code. It does however serve as a useful introduction to an alternative formulation outlined in this section, which *is* useful in the overset algorithms described within this chapter. This alternative ADT formulation, tests for the *intersection* of a finite sized object (the test object) with other finite sized objects comprising of the dataset.

Valid objects, for which the ADT method is formulated, are Cartesian boxes. Hence, if the intersection of arbitrary shaped objects is required, the bounding boxes of the objects can be quickly tested for intersection via an ADT search, eliminating the need for a more expensive intersection test if the bounding boxes do not intersect. If it is found that the bounding boxes intersect, then further tests may be performed to determine if the actual objects intersect, using, for example, the triangle-

triangle intersection test described in Section 5.3 if ultimately one is performing an intersection test of two triangles. If the bounding boxes do not intersect, then the objects cannot intersect, thereby eliminating the need for any further tests.

While the ADT method presented here requires a finite sized object as the test object, the intersection of a *point* and a candidate interpolation stencil is a typically required test for the interpolation algorithm (see Section 5.4.1). This is of little concern, since a point can be suitably approximated by a small bounding box. In this context, a small bounding box implies a Cartesian aligned box which is centred on the point, and has extremities that are negligibly far from the point (e.g. at a small distance such as 1×10^{-30} multiplied by a typical cell dimension, from the point). By setting up the test object in this manner, all (large) bounding boxes comprising of the dataset, that engulf the test point can effectively be found.

Note that in theory, the use of a small bounding box to approximate a point could potentially result in some false positive intersections being reported if the point is *very* close to the sides of the dataset's boxes. However when computing interpolation stencils for such a false positive result, the interpolation coefficients would lie outside their permissible range of zero to one, hence the stencil would not ultimately be used anyway.

For the purposes of the overset algorithms considered here, a suitable option for the dataset Catresian boxes is a list of boxes bounding each cell. To aid expeditious searching for cases where there are multiple subgrids (as are found in overset simulations), a separate tree is generated for each subgrid, with each tree's dataset comprising of all the cells in the given subgrid.

Tree generation

To generate a tree for geometric intersection testing, the first data item of the first grid is used to form a node (with the addition of the left and right pointers, as described previously), and inserted at the root of the tree. Remaining nodes are added in the same manner as for the basic ADT algorithm described in Section 5.5.2, with the exception that traversal is based on comparisons against the x_{min}, y_{min}, z_{min}, x_{max}, y_{max} and then z_{max} values of the box, in cyclic order at each hierarchy (where x_{min}, for example, is the minimum x-coordinate of the bounding box that is being inserted.).

Formally, for a three-dimensional dataset, tree traversal is based on the x_{min}, y_{min} or z_{min} values of the bounding box when $\frac{h}{6} - \lfloor \frac{h}{6} \rfloor$ is equal to zero, $\frac{1}{6}$, or $\frac{2}{6}$ respectively; similarly, traversal is based on the x_{max}, y_{max} and z_{max} values when $\frac{h}{6} - \lfloor \frac{h}{6} \rfloor$ is equal to $\frac{3}{6}$, $\frac{4}{6}$, or $\frac{5}{6}$ respectively. The left branch is traversed if the respective coordinate value is less than, or equal to, that of the current subtree's root, while the right branch is traversed if the opposite is true.

Intersection testing

When performing a geometric intersection search, we wish to find *all* bounding boxes that intersect the test box. Generally, there will be more than one such box (but not necessarily so). Since the test box is Cartesian aligned, it is fully defined by just two points in space, $(\mathbf{a}, \mathbf{b})$, corresponding to the minimum and maximum extremities of the box respectively. Starting at the root, the root node is first tested for intersection with the test box. If the inequalities $\mathbf{a}^{test} \leq \mathbf{b}^{root}$ and $\mathbf{b}^{test} \geq \mathbf{a}^{root}$ are both satisfied, then the box at the root intersects the test box, and is therefore added to a list of intersecting boxes. If, on the other hand, the above inequalities are not both satisfied, the box at the root does not intersect, it is therefore not added to the list.

Following from the inequalities above, it can readily be seen that if $\mathbf{b}_x^{test} < \mathbf{a}_x^{root}$ *or* $\mathbf{a}_x^{test} > \mathbf{b}^{root}$ hold, then the boxes *cannot* intersect. Furthermore, at the first branching, where traversal of the left and right branches is based on the minimum x-coordinate, all nodes on the right subtree will have a minimum x-coordinate $(\mathbf{a}_x)$, that is greater than that of the root. Hence if $\mathbf{b}_x^{test} < \mathbf{a}_x^{root}$ holds, all nodes on the right branch cannot intersect the test box, and the entire right subtree can be eliminated from the search.

Similarly, at subtrees where the hierarchy is such that branching is based on the *maximum* coordinate values (e.g. at hierarchies $4, 5, 6, 10, 11, 12, \cdots$), it can be seen that all nodes on the left subtree will have a maximum coordinate value that is less than that of the subtree's root. Hence if $\mathbf{a}_x^{test} > \mathbf{b}_x^{root}$ holds, all nodes on the left branch cannot intersect the test box, and the entire left subtree can be eliminated from the search.

Note that when branching is based on the *minimum* coordinate value, the above inequalities give no insight into the suitability of the nodes on the left subtree, hence the left subtree should be searched in any case, with only the right subtree being selectively searched. Similarly, when branching is based on the *maximum* coordinate value, the *right* subtree should be searched in any case, with only the *left* subtree being selectively searched. This leads to a possibility which was not present in the basic ADT algorithm, where *both* the left and right branches are searched at any given hierarchy. The fact that both branches of a subtree may need searching is intuitive since if the test box were so big that it intersected every bounding box comprising of the dataset, then clearly the list of intersecting boxes should include the entire dataset, requiring every node in the tree to be visited (i.e. every branch, both left and right, is traversed).

Performance

For the example scenario discussed above, where the test box were so big that it intersected every bounding box comprising of the dataset, clearly every node of the tree would require an intersection test. The ADT algorithm in this case would therefore offer no improvement over an exhaustive search, no matter how well balanced the tree is. Search times are therefore not simply a function of overall tree size, but are also dependent on the tree's balance as well as test box size and position. Specific search time performances are therefore not possible to quantify for a general case. However, logarithmic search times have been found for practical applications involving hole cutting in the work carried out for this thesis.

Once the tree has been generated (which is a reasonably expeditious process), it can be used many times to find interpolation stencils for all interpolation cells (see Section 5.4.1), and for assisting in the search for all wall-intersecting cells or fully overlapping cells (see Section 5.2), without requiring any further tree generation. If sliding meshes are used in a time-dependent simulation, any grids that have been moved in a given time-step must have their tree regenerated in order to remain valid. However, the tree is generated only once per a time-step, yet is used several times, resulting in an overall net reduction in computational costs for large datasets.

Alternative Formulations

The alternating digital tree algorithm presented in this section is different to other algorithms reported in the literature (see for example the method of Bonet *et. al.*, [56]). In their method, all datasets are scaled and translated such that the dataset varies within the interval $[0, 1)$. The scale factor applied to each coordinate direction is generally different, such that the end result is all data items lying within a unit cube, with at least one data item lying on each face of the cube (or, for a two-dimensional dataset, a unit square).

At hierarchies 0 and 1 (for a two-dimensional dataset), the left and right branches split the data items into values less than or greater than 0.5 for the x and y coordinates respectively (following the same alternating/cyclic procedure outlined previously), regardless of the relative position between the data item and the root. For example, Figure 5.14 shows an sample tree following this alternative methodology. Note that node C is placed to the right of node B, despite the fact that node C has a y-coordinate that is less than that of node B. This is the case since node C has a y-coordinate that is greater than 0.5, and hence is placed on the right link following this alternative ADT formulation, regardless of the relative position of the two nodes.

At hierarchies 2 and 3, the left branch will be traversed if the data item has a value in the interval $[0, 0.25)$ or $[0.5, 0.75)$, in the respective coordinate direction. Alternatively, the right branch will be traversed if the data item has a value in the interval $[0.25, 0.5)$ or $[0.75, 1)$. Figure 5.15 shows the link that is made between the branches of the tree, and the positions of the nodes in the unit cube.

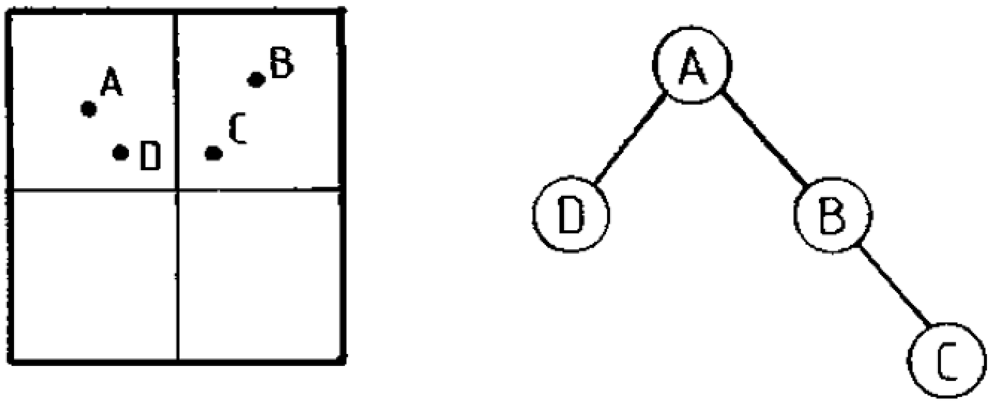


Figure 5.14: An alternative ADT formulation based on the method of Reference [56].

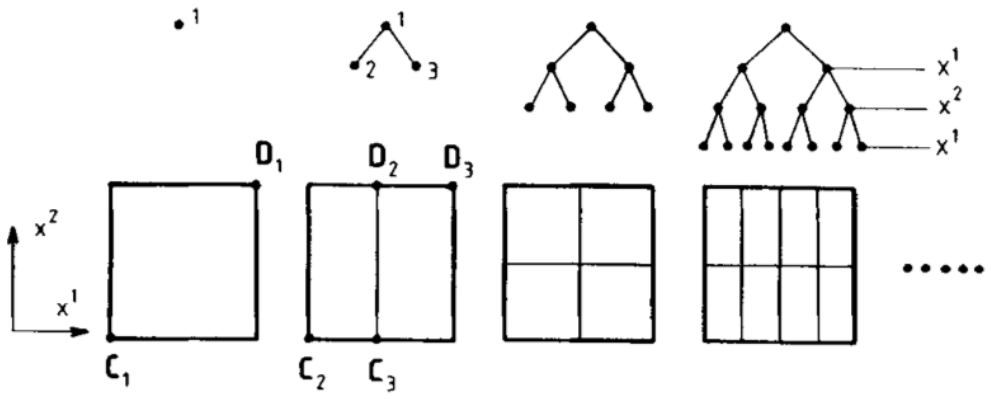


Figure 5.15: Relationship between the bisection process and the location of nodes in space, based on the alternative ADT formulation of Reference [56].

Clearly, the alternative ADT method presented in [56] will generate a different shaped tree, in general, to that of the method presented here. As already outlined, it is generally advantageous to strive for a tree that is as well balanced as possible, however there is no ‘best’ method as sometimes the method presented here will perform better than in [56], and vice versa.

Also, the cost of scaling and translating the dataset to a unit cube, and then the subsequent un-scaling of the result, should be accounted for. Furthermore, each branch traversal is based on predefined intervals (e.g. $[0, 0.25)$ or $[0.5, 0.75)$ for a two-dimensional dataset at hierarchies 2 and 3), and the upper and lower bounds of the interval therefore need to be computed for each traversal, leading to additional computational costs. In the ADT method presented here, this is not necessary, since branching is based purely on the root’s value, which is readily available.

In the present study, both methods have been coded, and it has been seen that for practical applications involving hole-cutting and finding interpolation stencils, the new algorithm presented here performed slightly quicker than the method of [56], and is easier to code. That said, both methods gave a clear improvement over a fully exhaustive search.

5.6 Integration over surfaces with overlapping grids

When using CFD to make predictions of force and moment coefficients, integration of wall shear stress and pressure forces over a surface is required. Furthermore, for internal flows, an overall bulk correction is required to ensure that the mass flux entering the domain is balanced by that leaving the domain (see for example Section 3.6.1), requiring a summation (integration) of mass fluxes over the inlet and outlet. In an overset grid formulation, there is no guarantee that the surface to be integrated over (the surface of a wing say), will be made up of just one grid. In general, there may be several such grids overlapping one another arbitrarily. Cells in the overlap region would require a weighting to prevent their contribution being counted twice. However, the calculation of a suitable weighting factor is non-trivial in the general case for cells that *partly* overlap other grids. This would require complex (and costly) geometrical analysis to determine the proportion of the cell that is overlapping.

A far simpler method than attempting to compute a geometric weighting is to remove any overlapping cells completely, and to fill the ’gap’ with triangular cells, generating a hybrid surface grid. The underlying primitive variables are then interpolated onto the triangular cells, and the resulting (non-overlapping) surface is integrated over [2]. This is known as the 'zipper grid' method, as described by Chan and Buming in [53]. An algorithm to facilitate surface integration has been implemented in the present study, based on the original method of Reference [53]. This section will describe the implementation details of the algorithm.

Consider the overlapping grids shown in Figure 5.16(a). The objective of the zipper grid algorithm is to convert the overlapping surface grids of Figure 5.16(a) into a non-overlapping hybrid surface grid such as that shown in Figure 5.16(b). Non-overlapping grids are first generated by removing some cells. The gap that this generates is then filled with triangular cells to form a hybrid surface grid. No new vertices are inserted in the definition of the triangular surface cells, hence the method is different to traditional unstructured (or hybrid) grid generation techniques, and is somewhat simpler to implement. Instead, the existing vertices that already contribute to the quadrilateral cell definitions, are joined together in such a way as to completely fill the gap, whilst also eliminating the possibility of creating a triangle that overlaps another triangle, or a quadrilateral cell. The triangular cells that fill the gap resemble a zip arrangement in their structure, hence the name 'zipper grids'.

Since the integration is to be performed over non-overlapping surface grids, it is first necessary to 'remove' some of the quadrilaterals in the overlap region such that the surface grids are no longer overlapping. Note that the quadrilateral surface cells are removed from the definition of the zipper grid, but are not actually removed from computational memory. This is implemented through the use of a blanking array, B_{ij}. Any cells that are flagged in the array are cells that will not be used in any subsequent integrations. Initially the blanking array is set to values such that none of the quadrilateral surface cells are flagged as being blanked, except for those that are associated with the hexahedral hole cells defined by the hole cutting algorithm (see Section 5.2). Since all hole cells play no part in the actual flow simulation, and hence would have undefined values stored for their primitive variables, they clearly should not be used in the integration.

Let S_i and S_j be any two subsets, containing the quadrilateral surface cells corresponding to subgrids Ω_i and Ω_j respectively. Using the geometric intersection test for bounding boxes, described in Section 5.5.3, the number of cells on subset S_i, that intersect with at least one quadrilateral belonging to subset S_j is computed, and a list of all such cells is stored. All cells in this list

[2]Note that the underlying flow solver uses the fully overlapping hexahedral cells to solve the discretised governing equations, and makes no use of any hybrid surface grids described in this section. The hybrid surface grids are used only for performing surface integration as a post-processing tool, or for applying a bulk correction on overlapping grids.

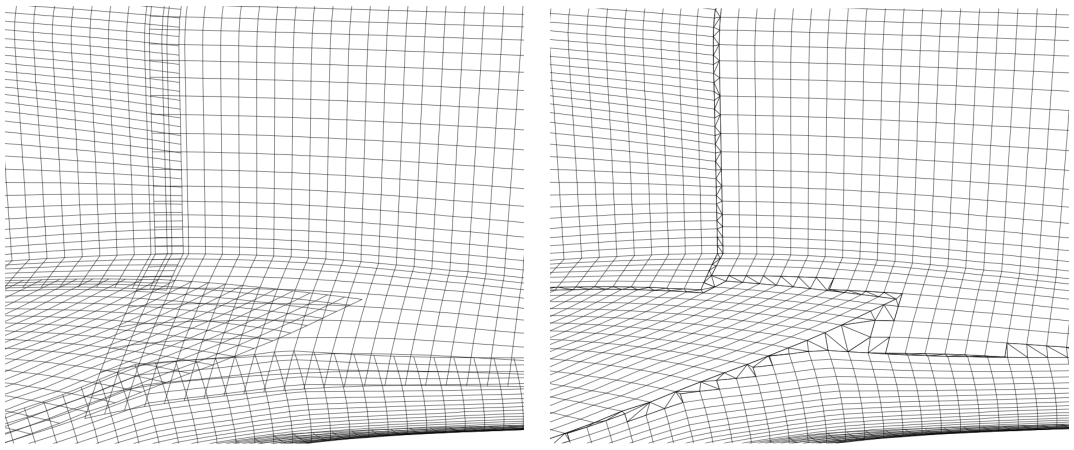

(a) Before zipping (b) After zipping

Figure 5.16: Sample overlapping surface grids which we wish to perform integration over.

belong to S_i, and are either fully overlapping or partly overlapping S_j. By removing all these cells, the subsets S_i and S_j would no longer be overlapping, as desired. However, the cells are not yet removed, as it is desirable to remove the cells on the coarsest subset, while preserving cells of finer subsets. The number of cells on the subset S_j that intersect S_i is therefore also computed, and the overlapping cells of the subset with the fewest number of computed overlaps are removed. That is, if $N(S_i) < N(S_j)$, all overlapping cells on S_i are removed, and vice versa. This step is repeated for all pairs of subgrids. The resulting surface grids after this step are shown in Figure 5.17.

Since a simple bounding box test was used to *estimate* if cells intersect one another, rather than determining if the cells actually intersect through a more accurate (but more expensive) test, more cells than necessary will be blanked for cases where the grid lines are non-Cartesian aligned. However, the basic bounding box test was used and found to perform well for the applications considered in the present study.

The next step of the zipper grid algorithm is to identify ordered strings of points that lie on the gap boundary created in the previous step. This is done in preparation for the final step of the zipping algorithm, in which the strings are 'zipped' together by forming triangular surface cells in the gap between subsets.

Firstly, each non-overlapping quadrilateral surface cell (i.e. all cells that remain unblanked) is tested in turn, and if it is found to be adjacent to a gap (i.e. adjacent to a flagged cell in the blanking array B_{ij}), the common edge between the blanked and unblanked cells is added to a list of *unsorted* 'gap boundary edges'.

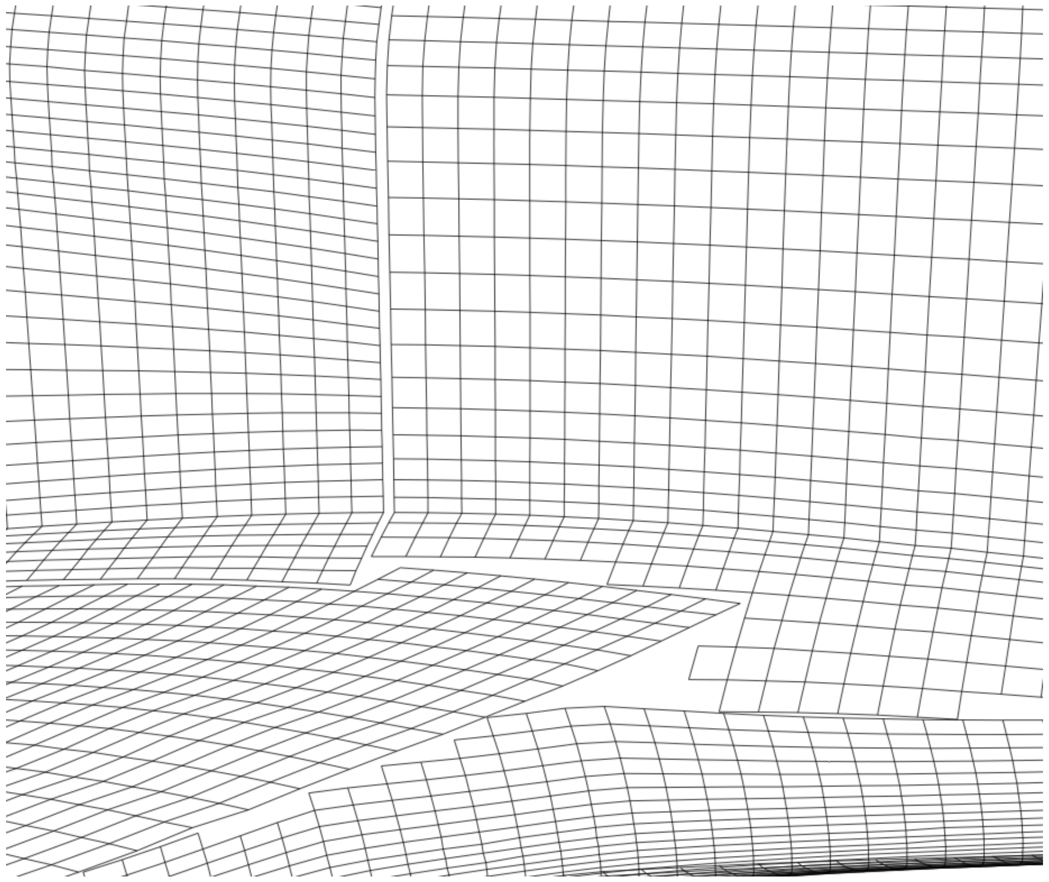


Figure 5.17: The same sample overlapping surface grids, after removing the overlapping cells from the coarsest subset.

Once the unsorted list of *all* gap boundary edges is generated, the edges in the list are connected to one another where possible, forming 'ordered boundary strings'. This proceeds as follows:

- The first string is started by removing the first edge from the unsorted list, and adding the two points that form this edge to a new list of *ordered* points (the order in which the two points are added does not matter). The *first* and *last* elements of this list form the *beginning* and *end* of the string respectively.
- Connections are now attempted to be made between either the beginning or the end of the string, and any of the remaining edges in the unsorted list. An edge can be connected to the string if the edge shares a common vertex with the string's end or beginning.
- Where a connection can be identified, the edge that is to be connected is transferred from the *unsorted* list to the string, by adding the unique point of the edge (and not duplicating the coincident points that were used in identifying the connection in the first place). The point is added to the end of the ordered list if the connection was made with the end of the string, with the added point forming the new end of the string. Similarly, if the connection was made with the beginning of the string, the point is added at the beginning of the ordered list, and the added point forms the new beginning of the string.
- Where no connection can be made between the string, and *any* of the remaining unsorted edges, a new string is started. A minimal string will therefore consist of just one gap boundary edge (i.e. two points), however, in general most strings will be longer than this.
- This process continues until there are no remaining edges in the unsorted list. As a minimum, this will generate two strings which will be zipped together (in general there may be more than two strings).

Figure 5.18 shows the ordered strings that are generated by following the above algorithm for the case considered. Note that in this case, there are four strings generated, corresponding to the four separate subsets. The number of strings generated will, at a minimum, be equal to the number of subsets, since strings cannot be connected to one another if they belong to separate subsets (quadrilaterals from separate subsets share no common vertices).

The next step of the zipping algorithm is to divide each boundary string into segments. The segments are then matched to one another to form matched pairs of boundary string segments. It is

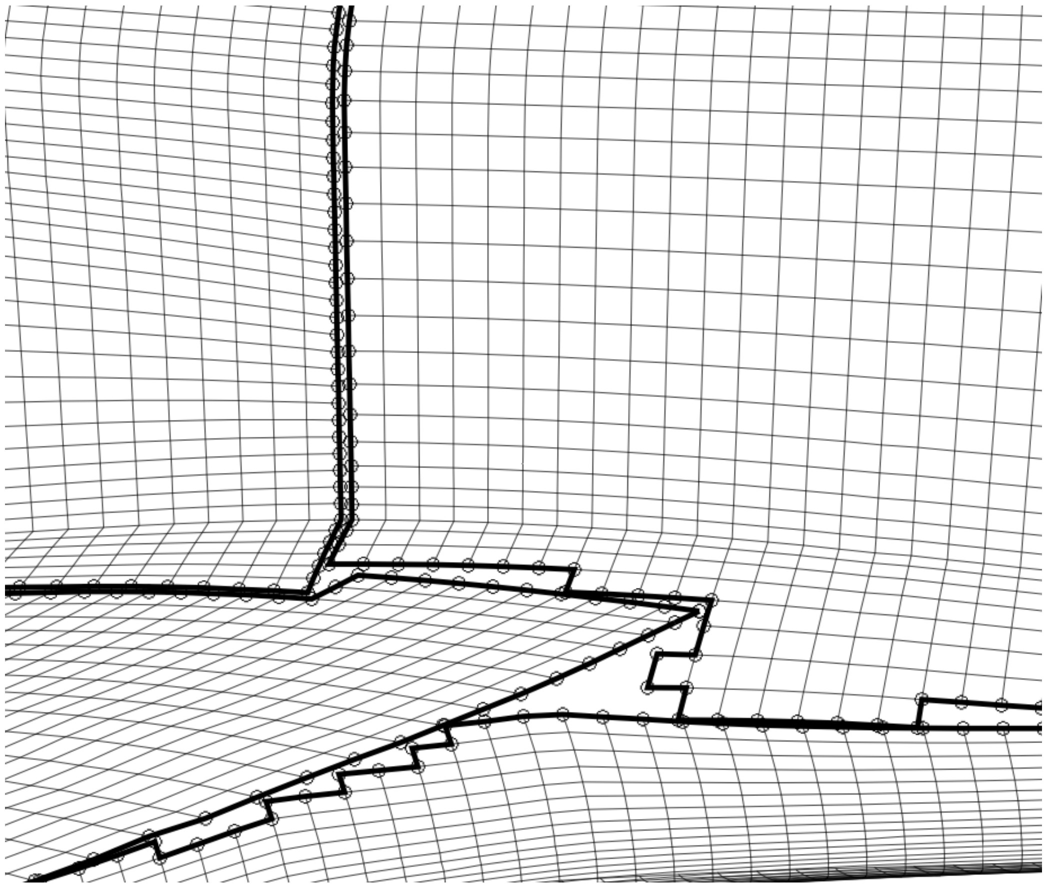


Figure 5.18: Ordered boundary strings (thick lines) and the points used in making up the strings (circles).

these matched pairs of segments that will ultimately be connected to one another in the final step of the zipping algorithm, with triangular cells bridging the gap. To form a matched pair of boundary string segments, take any string, R_i. For the first point on R_i (point P_i), find the nearest point (lowest Euclidean distance) out of all the points on all the other strings (excluding the string R_i) to point P_i. The point P_i is then matched to the string which contains this nearest point to P_i. The next point along R_i is then assessed in the same way, and matched to the string that contains the nearest point to this next point. A segment of R_i is then defined as the set of contiguous points along the string R_i, that are all matched to the same string. All strings are divided into segments following these rules. A matched pair of boundary string segments is one where all the contiguous points in a segment of R_i are matched to the points in a segment of R_j, *and vice versa*. Figure 5.19 shows the resulting matched pairs of boundary string segments, with each segment pair highlighted via the use of a separate line colour. In this case, it can be seen that there are five matched pairs of boundary string segments (ten segments in total). However, in general, there is no relationship between the number of boundary strings, and the number of segments generated (other than the fact that the number of segments cannot be less than the number of strings), since each string may be spit into an arbitrary number of segments, and the number of segments is dependent on the specific geometry of the subsets.

Figure 5.20 shows a pair of matched boundary string segments. To zip these matched segments together, the closest two points to one another (one point from each of the two segments) are first joined, as shown in Figure 5.20, indicated through the use of a thick line.

In the next step of the zipping algorithm, the first triangle is generated. Two of the three vertices of the triangle are defined from the previous step and are the two points that are closest to one another (i.e. points P_i and P_j). The third vertex that will be used will be the next contiguous point along one of the boundary strings, in the positive direction (i.e. toward the end of the string). Either node P_{i+1} or node P_{j+1} will therefore be used as the third vertex in the triangular cell's definition. The decision of whether to use $\triangle P_i P_{i+1} P_j$ or $\triangle P_i P_{j+1} P_j$ proceeds as follows.

In some cases, where the quadrilateral shape formed by the points P_i, P_{i+1}, P_{j+1} and P_j is convex, one of the triangles $\triangle P_i P_{i+1} P_j$ or $\triangle P_i P_{j+1} P_j$ may be invalid (due to it overlapping existing quadrilateral cells, or other triangles). To test if $\triangle P_i P_{i+1} P_j$ is valid, the quad $P_i P_{i+1} P_{j+1} P_j$ is split along the line $\overline{P_{i+1} P_j}$, forming two triangles $\triangle P_i P_{i+1} P_j$ and $\triangle P_j P_{j+1} P_{i+1}$. The normals of these two triangles are computed, and if both normals have the same sign, then $\triangle P_i P_{i+1} P_j$ is valid. A

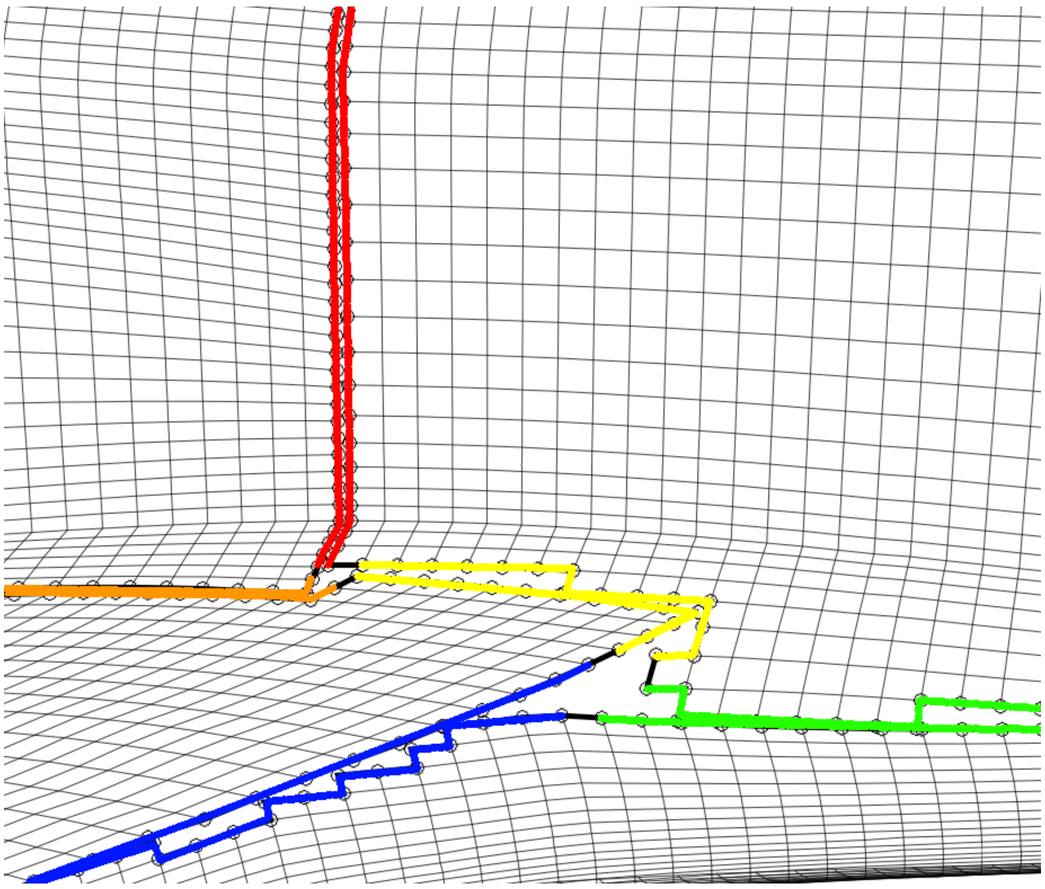


Figure 5.19: Boundary string segment pairs. Different pairs are indicated through the use of different coloured lines.

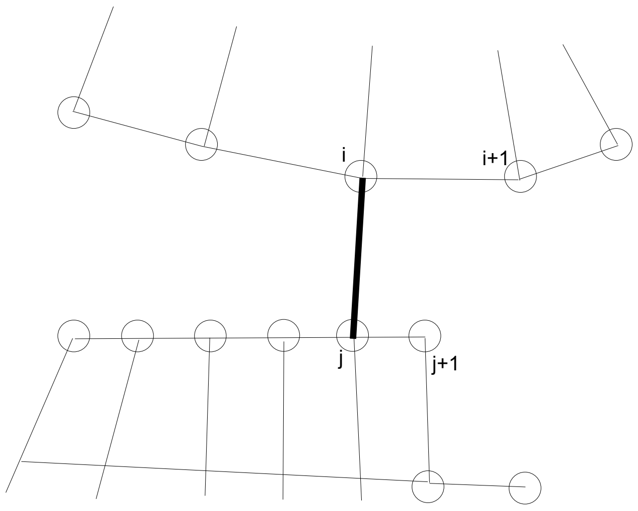


Figure 5.20: A sample matched pair of boundary strings, with the closest two points joined.

similar test is performed to see if $\triangle P_i P_{j+1} P_j$ is valid, by splitting the quad along the line $\overline{P_i P_{j+1}}$, and comparing the sign of the normals of triangles $\triangle P_i P_{j+1} P_j$ and $\triangle P_i P_{i+1} P_{j+1}$.

If both triangles $\triangle P_i P_{i+1} P_j$ and $\triangle P_i P_{j+1} P_j$ are valid, as is the case more often than not, then triangle $\triangle P_i P_{j+1} P_j$ is selected if $\overline{P_i P_{j+1}} \leq \overline{P_{i+1} P_j}$, while $\triangle P_i P_{i+1} P_j$ is selected otherwise. This results in the shortest possible length of the triangle's edges, and hence the smallest possible triangle area (in the absence of introducing extra vertices, which would significantly increasing the complexity of the algorithm).

If one of the triangles $\triangle P_i P_{j+1} P_j$ or $\triangle P_i P_{i+1} P_j$ failed the test above, while the other triangle passed, then one can deduce that the quadrilateral formed by the points P_i, P_{i+1}, P_{j+1} and P_j is convex, and the zipping should proceed by selecting the only conclusively valid triangle option. For example, Figure 5.21 shows the case considered after the insertion of the first triangle. The quadrilateral formed by the points P_i, P_{i+1}, P_{j+1} and P_j in Figure 5.21 is convex and clearly $\triangle P_i P_{j+1} P_j$ is invalid (it overlaps quadrilateral cells, and fails the test outlined above). Triangle $\triangle P_i P_{i+1} P_j$ is therefore used.

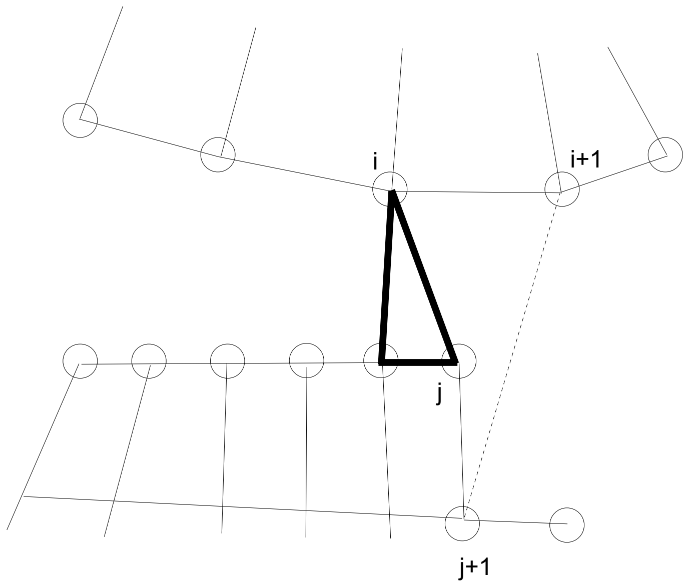


Figure 5.21: After the insertion of the first triangle, in this case a convex quad is encountered. Here it can be seen that only $\triangle P_i P_{i+1} P_j$ is valid.

If neither of the two triangles $\triangle P_i P_{j+1} P_j$ or $\triangle P_i P_{i+1} P_j$ are conclusively valid, then one can deduce that the quadrilateral formed by the points P_i, P_{i+1}, P_{j+1} and P_j is inverted. In this case, the algorithm proceeds as for the case where both triangles are valid, with the shortest distance test determining which triangle is used. This usually results in a satisfactory zipping, however this is not guaranteed and repairs of the zipped gap may be required.

Thankfully, cases requiring repairs are very rare, and only come about if there is a considerable heterogeneity in the grid spacing ratios between subsets. Since a large heterogeneity in the grid spacing is best avoided anyway, due to the spurious discontinuities in the solution that can often result, cases requiring repair were not encountered in the work presented in this thesis. If however, a repair were to be required, Reference [53] describes a simple method which involves removing the bad triangle (the one that overlaps existing quadrilateral cells, or other good triangular cells), and also the two good triangles that are immediately adjacent to the bad triangle, and then simply re-zipping the local area in the reverse direction (from the end of the string segment toward the beginning). This simple method can be used to repair most cases even with significant heterogeneity in the grid spacing ratios. However, this repair technique is not guaranteed to work in *all* cases, and some particularly poor grids (poor in the sense that the heterogeneity in the grid spacing ratios between subsets is exceptionally large) may not be able to be zipped at all without significant modification of the zipping algorithm. In this case, it is advised that one should regenerate the grids (which is advisable in any case to avoid other problems associated with such a large heterogeneity).

Once a triangle has been generated, the i index is incremented if the triangle's definition involved the point $i+1$ (i.e. the triangle $\triangle P_i P_{i+1} P_j$ was used), while the j index is incremented if the triangle's definition involved the point $j+1$. The algorithm then proceeds to insert one triangle after another, until the end of a string segment is encountered (i.e. until $i = i_{max}$ or $j = j_{max}$). Figure 5.22 shows the result so far for the string segments considered.

Once the end of a string segment is reached, all remaining nodes on the j-indexed segment (that are after the point P_j) are connected to the point P_i, and vice versa, thereby completing the zipping in the positive direction. The result of this is shown in Figure 5.23. Once the zipping in the positive direction is complete, the indexes i and j are returned to the nearest two points that were used at the start of the zipping, and zipping in the negative direction is commenced, following exactly the same procedure as outlined for the positive direction. Figure 5.24 shows the final zipped segment for the case considered.

All matched boundary string segments, as identified earlier and depicted in Figure 5.19, are zipped following the same procedure, and the result is shown in Figure 5.25(a). It can be seen from this figure that most of the 'gap' between subsets has been triangulated, however there are still

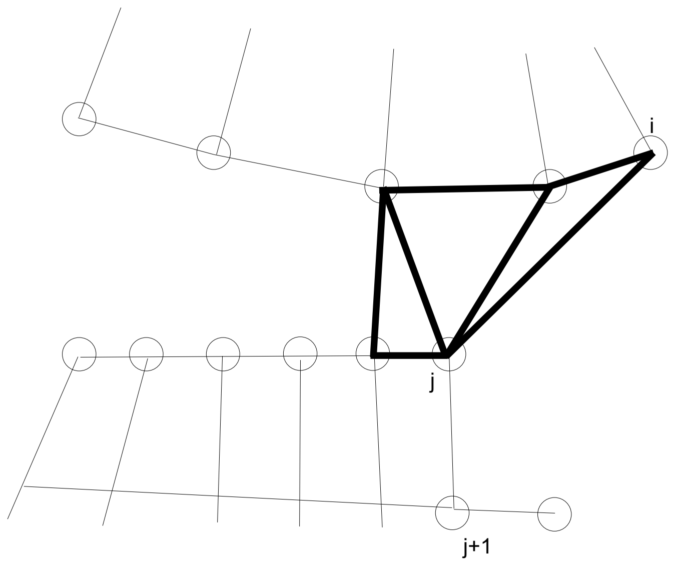


Figure 5.22: The zipping proceeds in the positive direction (toward the end of the string) until the end of a segment is encountered.

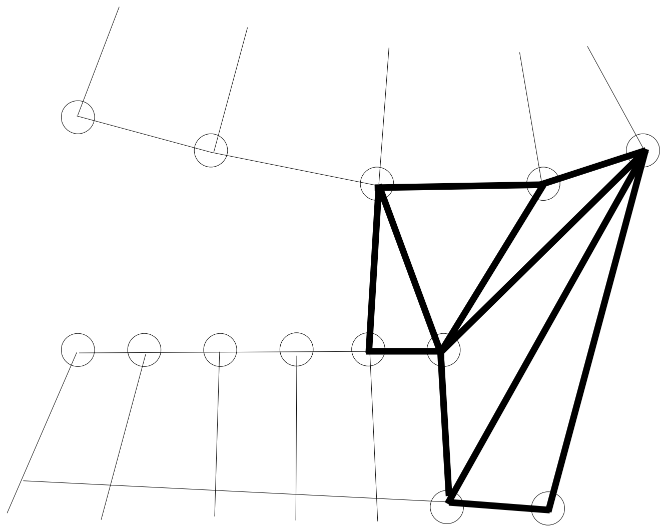


Figure 5.23: The end of each segment is connected to the remaining points of the other segment, thereby completing the zipping in the positive direction.

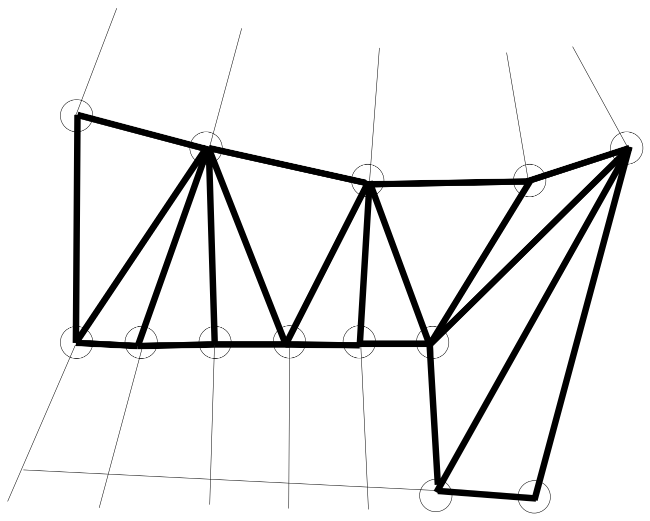
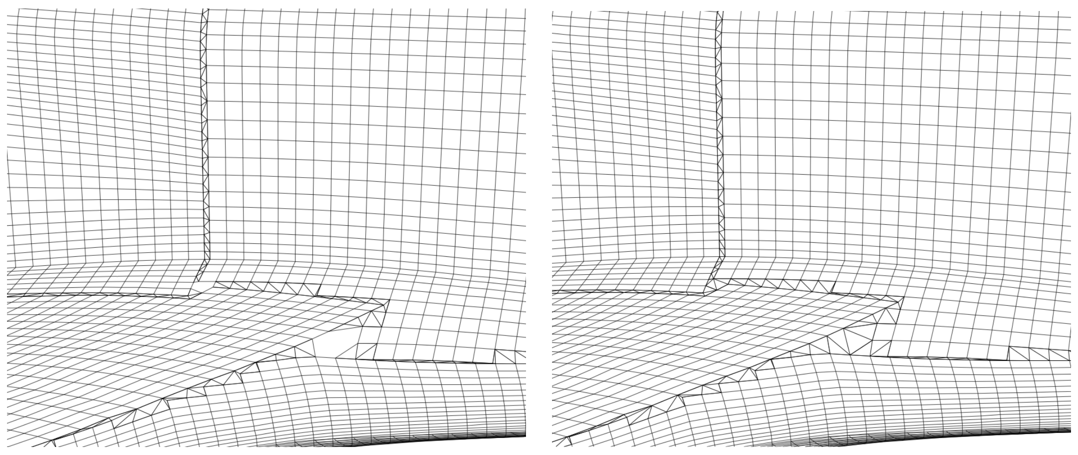


Figure 5.24: The final zipped segment pair.

small 'pockets' where no triangles or quadrilaterals are present. These pockets are polygon shaped, with the edges of the polygon corresponding to parts of the unconnected segments identified earlier, or the ends of the triangular cells inserted by the zipping procedure. These pockets occur in the regions where three or more subsets all overlap one another. The pockets are easily triangulated using a similar method to the method already described for zipping strings. Firstly, the closed loop of edges that form the pockets are identified. This is done simply by visiting each unblanked cell on each subset (including the triangular cells added by the zipping procedure, but excluding boundary cells that are adjacent to the edge of a computational boundary) and testing each edge of the cell for for the absence of an adjoining cell. If such an edge is found, it is added to a list of unordered edges, and once complete, this list is used to form ordered loops of connected edges in a similar way to which the boundary strings were formed.

Once a closed loop of edges forming a polygon pocket is identified, each of the internal angles of the polygon are computed, and 90° is subtracted from the result of each (the reason for this will become apparent). The modulus of the resulting angle (the internal angle less 90°) is then computed, and the edge associated with a average result from both its endpoints that is closest to zero is identified (i.e. the edge whose endpoints contain internal angles closest to 90°). Zipping starts from this edge, and it is this edge that is analogous to the edge formed by the closest two points in the regular zipping algorithm such as the two connected points of Figure 5.20.

Zipping of the pocket then proceeds with the two points just identified forming two of the three vertices of the triangle cell, with the third vertex being one of the two points adjacent to the two points already identified. The quadrilateral formed by these four points is assessed with regards to being convex, inverted or otherwise, and the selection of which point to use in the formation of the triangle is based on the criteria outlined above for the regular zipping algorithm. Once the pockets have all been closed, the zipping procedure is complete and the resulting hybrid surface grid is shown in Figure 5.25(b).


(a) After zipping, but before triangulating the gaps where three subsets meet.

(b) The final result of the zipping algorithm

Figure 5.25: Final zipped grids, before and after triangulating the final gaps.

Once the zipped subsets have been generated, integration proceeds by interpolating the underlying primitive variables from the overset grids to the centre of each of the triangular cells. The solution at the quadrilateral cells is already available since they form part of the underlying grids anyway, so no interpolation to these cells is required. Integration then commences over the non-overlapping zipped grid in the usual way, by summing the contribution from each non-overlapping cell.

5.7 Domain Connectivity

This section describes the domain connectivity procedure that is used in order to provide continuous solutions from one sub-grid to the next, which are equivalent to the block-structured solutions (when grid independence is achieved in both cases).

Interpolation of all primitive variables is conducted at the interpolation cells at the start of each outer iteration. The method of interpolation used to achieve this can be the linear interpolation method, as described in Section 5.4.1, the mass-flux based method, as described in Section 5.4.2, or some other method (such as those discussed in Chapter 2). The use of higher order interpolation methods makes little sense unless higher order discritisation techniques are also employed. The second order methods discussed in section 5.4 have therefore been used in the present study, coupled with sufficient grid resolution to achieve grid independence.

In instances where interpolation boundaries arise at the edge of a sub-grid (e.g. where $i = 1$ or $i = Ni$) halo cells are required in order to store the interpolated variables. The geometry of the halo cells should mirror that of the adjacent standard cells, with the plane of reflection lying on the face that splits the halo and standard cell.

In order to improve continuity between sub-grids, it has been found that it is necessary to treat any interpolation boundaries in a manner that is consistent with the internal boundaries between standard cells. This entails applying a Rhie-Chow type interpolation for the mass flux calculation at the interpolation boundaries (where one is used to calculate the mass flux between standard cells). Also, for the pressure correction equation, the coefficient in the direction of any interpolation cells (where the coefficients are given in Section 3.5) should be evaluated in the usual way (i.e. in the same way as for adjacent standard cells), and not simply set to zero as is often the usual practice for other types of boundary condition.

Since the coefficients of the pressure correction equation depend upon the diagonal coefficient of the momentum equations, it is necessary to evaluate the diagonal coefficients of the momentum equations at each interpolation cell. The diagonal coefficient, A_P, is evaluated at each interpolation cell's centre, and is averaged onto the face that is required for use in the pressure correction equation, and for Rhie-Chow interpolation. In order to evaluate the diagonal A_P coefficient at interpolation cells, the fluxes through each face of the interpolation cell are required. Two options are available, which are to interpolate the flux density at the face centres of the interpolation cell directly, and then multiply this flux density by the face area in order to obtain the required flux. Alternatively, one can interpolate the primitive variables onto a second row of halo cells, and compute the fluxes via linear interpolation between the two adjacent halo cells. Both options have been implemented in the present code, with little difference found between the two options. The latter option was therefore used, as the second row of halo cells also proved to be useful in treating the convective fluxes in a consistent manner where higher order discritisation techniques are used (e.g. the QUICK scheme), since a QUICK interpolation can then be conducted on the interpolation boundaries.

The overall solution procedure is similar to that of the standard SIMPLE algorithm commonly employed on single-block structured grids, with some subtle differences, as outlined below:

1. Set all boundary conditions, including all interpolation boundary values, for each sub-grid.

2. Compute the momentum equation coefficients (A_E, A_W, A_P, etc.) for each equation at each standard cell. Also compute diagonal coefficient (A_P) on all interpolation cells.

3. Solve, on each sub-grid in turn, the discritised momentum equations for the intermediate velocity field (which is generally non-mass conserving. Typically only one iteration of the momentum equations is performed for each outer iteration, however this is not necessarily the case. If more than one iteration of the momentum equations is to be performed, the interpolation boundary conditions should be updated at the end of each inner iteration for the variable that is being changed.

4. Compute the uncorrected mass fluxes through each face, applying Rhie-Chow interpolation to determine the face velocity. Rhie-Chow interpolation is applied at all internal faces, and also at faces that form interpolation boundaries (but not at faces that form other boundaries, such as inlets, due to the lack of sufficient information in the boundary direction). Rhie-Chow is employed on the interpolation boundary in order to provide a consistent boundary treatment.

5. Solve the pressure correction by performing an inner iteration of the discritised pressure correction equation (Equation 3.38) on each sub-grid in turn. This is then repeated several times (e.g. 5-10 times), or until the residual of the pressure correction equation has fallen to a predefined fraction of the initial residual. At the end of each inner iteration, the value of p' at all interpolation cells are obtained through interpolation. This is valid since the pressure will have been obtained through interpolation, hence the mass conserving pressure correction must be the same at interpolation boundary point as it is at the same point on the underlying donor grid in order to maintain the same pressure at the same point on both grids. Failure to set the interpolation boundary conditions of the pressure correction equation in this consistent manner can result in spurious ‘ kinks’ in the pressure field close to the interpolation boundaries.

6. The boundary pressure corrections are updated, via interpolation for interpolation cells, or by applying a zero gradient condition for all other boundary types.

7. The face mass fluxes, cell centre velocities, and pressure are updated via Equations 3.39.

Chapter 6

Selected test cases

The Figures referred to within this chapter can be found from Page 177.

6.1 Introduction

The methods discussed in Chapters 3,4 and 5 have been incorporated into a new overset CFD code that has been developed in this project. This code uses the finite volume method within a general coordinate system (as outlined in Chapter 3). The governing equations that the code solves are the steady state, discretised, Navier-Stokes equations in three spatial dimensions. Two-dimensional simulations are implemented via setting symmetry conditions on all xy-planar faces.

Numerical experiments involving overset grids have been conducted as part of this project, the results of which are presented within this chapter. The main objective of these numerical experiments has been to test the overset method and the finite volume implementation in the present code, rather than to analyse specific flow physics. For this reason, fully grid independent solutions were not always sought. The geometries considered in this chapter are typically rather simple. This enables a single-block or block-structured simulation to be carried out with ease for comparative purposes.

In Section 6.2, the flow in a cavity that is driven by a lid is investigated. Section 6.3 considers the crossflow over two rotating cylinders. Finally, Section 6.5 considers a jet flow impinging onto a concave surface.

6.2 Lid Driven Cavity flow

Lid driven cavity flow involves flow in a closed cavity that is induced by a moving lid. Figure 6.1 shows a schematic of the flow. The no-slip condition causes the fluid in contact with the moving lid to move at the same velocity as that of the lid. This is turn causes the fluid throughout the rest of the cavity to move via viscous interaction with the adjacent fluid. As this fluid impinges onto the side wall (the right wall in this instance since the lid is moving from left to right), a high pressure region is generated in the top right corner. This high pressure causes the fluid to move downwards towards the bottom right corner along the favorable pressure gradient. This favorable pressure gradient however is short-lived and soon turns adverse as the influence of the lower wall is felt. Typically, boundary layer separation from the right side wall will occur as a result of this adverse pressure gradient for all but exceptionally low Reynolds numbers (e.g. Stokes flow within a lid a driven cavity [58]). This separated fluid is then directed towards the lower wall due to its inertia (despite the adverse pressure gradient), where it reattaches onto the lower wall. As the flow follows the lower wall, it experiences a favorable followed by adverse pressure gradient for much the same reasons as were described for the side wall. Separation again typically occurs from the lower wall, near the bottom left corner, for this reason. The flow along the left wall experiences a strong favorable pressure gradient along its entire length owing to the low pressure region that is present in the top left corner of the cavity (the low pressure is required in order to satisfy mass conservation since the lid drives fluid away from this corner which must be replaced). The fluid is seen to move around the perimeter of the cavity forming the primary vortex. Viscous shear interaction between this primary vortex and the fluid in the two separation regions (i.e. in the bottom two corners) causes the formation of two secondary vortices in these regions. At the center of the primary vortex (and to a lesser extent the secondary vortices) there is a low pressure region which is present in order to balance the centrifugal force associated with the streamline curvature.

Lid driven square-cavity simulations have been conducted with various mesh topologies. The Reynolds number is equal to 100 in all instances (based on the lid velocity and the length of one of the square cavity's sides, L). The QUICK convection scheme is used in all cases.

6.2.1 Single block results

A single block solution forms the benchmark from which other mesh topologies shall be compared. Figure 6.2 shows the 900 cell mesh used in this simulation. It can be seen that a slight refinement towards the walls has been made; all near wall cell centres are placed at $0.004L$ from the wall, while a growth ratio of 1.2 is applied in the wall normal direction over 7 cells before a uniform spacing is applied.

Figures 6.3 and 6.4 show contours of the x and y components of the velocity respectively, normalised by the lid velocity. Also shown in the figures are the results that have been obtained by running the same simulation with the same settings and grid with the commercial software FLUENT. Both sets of results have been iterated until the residuals of all variables reached machine accuracy. It can be seen that a near exact match is obtained, thereby offering some validity towards the presently obtained single block result.

Figure 6.5 reports the pressure fields obtained from these single block simulations. Here it can be seen that the agreement between FLUENT and the present code is slightly less satisfactory. It is expected that the discrepancy is due to the boundary condition for the pressure that is applied at the cavity walls. In the present code, a zero gradient boundary condition is applied ($\partial p/\partial n = 0$). Justification for this condition can be made by considering the momentum equation in the wall-normal direction, which after cancellation of the terms that are zero at the wall due to the no slip condition gives:

$$\left.\frac{\partial p}{\partial n}\right|_{wall} = \left.\frac{\partial}{\partial n}\left(\mu\frac{\partial u_n}{\partial n}\right)\right|_{wall} \tag{6.1}$$

Here the continuity condition for an incompressible flow has been employed in order to cancel out additional terms arising from the antisymmetric part of the viscous stress tensor. The term n in the equation is the coordinate in the wall-normal direction.

The bracketed term on the right hand side of Equation 6.1 (i.e. $\partial u_n/\partial n$), must be zero at the wall so as to ensure no flow through the boundary. The full term on the right however will not in general be equal to zero, but is usually negligibly small at the wall. A zero gradient condition is therefore a reasonable approximation and is simple to implement. Even at regions of flow impingement, the RHS of 6.1 will tend to be small since the high stagnation pressure at the impingement point will act to decelerate the wall-normal velocity component to an essentially zero value *before* the wall is encountered (and hence the wall normal velocity will be near zero over some short distance before the wall, thereby ensuring the second derivative is small).

The discrepancy in the pressure field that was seen in Figure 6.5 is expected to arise from FLUENT using the full form of Equation 6.1, with the right hand side retained (although this cannot be confirmed since FLUENT is a commercial code and the precise implementation does not appear in the documentation). While the right hand side of Equation 6.1 is expected to be small, the discrete approximation to Equation 6.1 typically will not be small if the grid spacing is large. Given the rather coarse grids used, this is likely to be the issue. Grid refinement studies have shown that the pressure field improves drastically with additional cells, giving essentially the same solution by both codes. Figure 6.6 shows the results of this test, where 2704 cells have been used in the refined grid. The discrepancy in pressure between the two codes at each cell of the computational domain has been computed for this refined grid. It was found that the discrepancy was, even in the worst case, still less than 0.1% of the dynamic pressure, $(0.5\rho u_{lid}^2)$.

6.2.2 Overset grids with one-way information transfer

The next test that was undertaken involved the use of an overset sub-grid placed towards the upper region of the cavity (where the velocity and pressure gradients are steep, thereby providing a challenging situation for the interpolation algorithm). The overset sub-mesh comprises of a uniform 7×7 cell square that is rotated by 30° from the horizontal. The background grid that is used to mesh the cavity is the same as the initial 30×30 grid used in the previous section. The overset mesh interpolates from this background grid, however the opposite is not true. The solution of the background grid is therefore independent of the overset sub-mesh and will be identical to the single-block solution considered earlier. Figure 6.7 shows the meshes used in this test.

Figures 6.8 and 6.9 show contours of the horizontal and vertical velocity component respectively for the one-way information transfer test. It can be seen that the velocity is convected reasonably well across the interpolation interface. The small discrepancy in the velocity field in the overlap region is likely to be due to differences in the truncation error between the two grids. While the spacing is broadly the same, the orientation of the cells relative to the flow vector is different. It was shown in Chapter 3 that the leading truncation error term for the QUICK scheme is proportional to the third derivative of the variable under consideration, with respect to the coordinate direction normal to the cell face (Equation 3.16). If the flow is aligned with the faces, this term will generally be small, and dependent on the local flow acceleration or deceleration only. If on the other hand the flow is not aligned with the cells, the third derivative will involve the u or v velocity gradient in a direction other than that of the velocity vector. This will introduce a different truncation error relative to the former case (the error is generally larger since the cross-stream velocity gradients are generally larger than the streamwise velocity gradients).

Figure 6.10 shows the pressure field that is obtained from the present one-way information transfer simulation. Here it can be seen that discontinuous solutions have been obtained on the overset sub-grid. The reason for this is due to the differences in truncation error on the two grids. The problem is that the pressure at the interpolating boundary of the overset mesh is being forced to a value that is interpolated from the interior of the background mesh. It has already been noted that the interior of the background mesh will develop a unique solution to that of the overset mesh, due to the differences in the truncation error between the two grids. The fact that the interior of the mesh is being forced to a value that it's surrounding nodes do not agree upon causes a discontinuity since the discrete problem is over-specified at the boundary. In an incompressible Navier-Stokes simulation, one typically sets boundary conditions for the velocity while the boundary pressure emerges as part of the solution, or vice-versa. In this situation the problem is well-posed [59]. However, in the overset method, it is necessary to prescribe boundary conditions for the velocity components *and* pressure. As such, the discrete problem is likely to be ill-posed, unless the interpolated velocity and pressure are consistent with one another (and with the solution on the interior of the interpolating grid). This will be the case if truncation error is negligibly small since unique solutions in the overlap will not materialise in this case. Hence, the discontinuous pressure reported here is of little concern as one would expect convergence to a continuous solution at grid independence (this will be seen in later Sections where grid independence has been sought).

6.2.3 Overset grids with two-way information transfer

In this test, the same overset grids that were used in the previous section are again employed. This time, the hole-cutting algorithm is invoked in order to remove unnecessary cells in the overlap region. Cells are removed from the background grid only. Figure 6.11 shows the grids in this case.

Figures 6.12 and 6.13 show contours of the horizontal and vertical velocity component respectively for the two-way information transfer test. It can be seen that the agreement between grids is again very good for the velocity field.

Figure 6.14 shows the pressure field. In this case the pressure is much more satisfactory than was seen for the one-way information transfer case (Figure 6.10). Even with these coarse grids, near continuous solutions are obtained over the interface.

Since the cells on the background mesh now interpolate from the overset grid, and vice-versa, any differences in the solution that would ordinarily be obtained in the overlap region, due to differences in the truncation error, are smoothed out. The solution on the background grid is fixed to the interpolated values at interpolation cells within its interior. The solution on the background grid's cells that are close to the hole are therefore strongly affected by the solution on the overset grid, and end up converging towards a solution that the overset grid has contributed towards. Since the overlap is reasonably small, these cells end up forming part of the donor interpolation stencils that are used in interpolation onto the overset grid. Both grids therefore converge towards a solution that they have both contributed towards, and hence the solution is smooth. If the overlap were excessively large, unique solutions would again develop in the overlap region due to differences in truncation error over the two grids. Continuous solutions may not be obtained in this case. It is advantageous therefore to minimise the overlap (which is handled automatically by the hole cutter developed in this code).

The solution obtained on the present overset grids is now compared to the single block solution obtained earlier. Figures 6.15 and 6.16 show profiles of the x-velocity and the pressure respectively, taken through the cavity. It can be seen from the former figure that there is no appreciable difference in the x-velocity component between the overset and the single block solutions. The latter figure shows some minor differences in the pressure field. The reason this difference exists is that the grids are rather coarse and hence truncation error is appreciable. The differences that the cell rotation makes to the truncation error have been highlighted. At grid independence, the overset solution will converge to that of a grid independent single block solution.

6.2.4 Multiple overset grids.

The test case considered here serves as a useful demonstration of the hole cutting algorithm. From Figure 6.17 it can be seen that the case consists of the addition of a circular O-shaped overset sub-mesh which overlaps both the original square mesh and the background mesh. It can be noted from the figure that part of the circular mesh extends outside of the domain boundary. This however is of little concern since the hole cutting algorithm cuts all cells that intersect walls (they by definition must lie on the boundary, so cannot be used). The cells that are adjacent to these cut cells must be converted to interpolation cells in order to provide complete computational stencils to adjacent standard cells. However, if no donor grid can be identified then these cells must also be cut (with the cells adjacent to the newly cut cells being set to interpolation instead). This process of attempting to find interpolation stencils continues until no further changes are made by repeating the process, by which point all cells outside of the domain will have been cut. This algorithm is both rigorous and quick (thanks to the use of alternating digital tree data structures, as discussed in Section 5.5.2). See Section 5.2 for further details on the hole cutting algorithm.

Figures 6.18, 6.19 and 6.20 show the x and y component of velocity, and the pressure field respectively for this test case. It can be seen from these figures that reasonable solution continuity is obtained between grids, despite the considerable heterogeneity in the cell size (particularly towards the centre of the blue mesh), and the relative coarseness of the meshes.

6.2.5 Cavity with a circular cylinder inserted.

This section briefly demonstrates the use of the overset method in a situation where its use is advantageous over a block-structured arrangement. In all the prior tests, the overset method has offered no advantage over a single block mesh (it actually complicates the gridding slightly). Here the situation is one where a circular cylinder is placed within the cavity which provides an obstruction to the flow.

The same meshes as were used in the previous section are again used. In this case, the boundary condition of the interior circle of the O-grid is set to a wall before calling the hole-cutting algorithm. The additional cells of the background grid that originally were preserved in order to mesh the portion of the domain at the centre of the cylinder, are now cut since they intersect the cylinder's wall.

Figures 6.21, 6.22 and 6.23 show the x and y component of velocity, and the pressure field respectively for this case. It can again be seen that continuous solutions are obtained. The considerable distortion of the flow, due to the presence of the cylinder, is also apparent from the plots. It can be noticed that the boundary layer around the cylinder is poorly resolved due to the coarse mesh with little near-wall refinement. However this does not detract from the attractiveness of the overset method. The objective here is to demonstrate the method rather than obtain accurate results.

6.2.6 Summary

The flow in a cavity which is driven by a moving lid has been investigated. Single block solutions have been compared against solutions obtained from the commercial software FLUENT, with favorable outcome. With a sufficiently fine mesh, the solutions are essentially the same between the two codes. This important result offers some validation of the underlying flow solver developed in the present study.

Overset simulations with one and two-way information transfer have been conducted over fairly coarse meshes. It was found that the former case did not give a satisfactory pressure field since the solution at the boundary of the overset mesh is forced to a pressure that is not consistent with the interior cells of that mesh (the boundary conditions are over-prescribed). Solution differences due to the different truncation errors on the two meshes lies at the root of the problem. At grid independence, the issue will vanish since truncation and interpolation error will be negligibly small.

For the two-way information transfer case, the pressure field is continuous across the overset grid. In this case the pressure at the boundary of the overset grid is (reasonably) consistent with the interior cells of that mesh since these very same interior cells that have contributed to the solution on the background donor mesh (and vice versa). These findings highlight the importance of minimising the overlap.

Comparisons with the overset simulations have been made against the single block simulation. It was found that the velocity field is broadly the same for the two sets of results. The pressure shows small differences between the two results due to the differences in truncation error that the overset grid rotation introduces.

Other tests involving three overset sub-grids have been conducted to demonstrate the capabilities of the hole cutting algorithm in dealing with moderately complex geometries. It is shown that overset sub-grids need not lie completely within the computational domain. Provided the boundary conditions are appropriately tagged on all sub-grids, the hole cutting algorithm can automatically deal with such cases. Boundary condition tags need be applied regardless of the method of meshing, and hence this is of no additional cost to the user. The hole cutter is also shown to perform well in selecting which cells to cut in an overlap region. In all cases, the smallest cell sizes are preserved.

Finally, the flow in a lid driven cavity with a circular cylinder obstacle demonstrates one practical application of the overset method. A block-structured grid would be moderately challenging to generate in this case since some domain decomposition would be required (a 6 block decomposition seems logical to this author, although other options are available). The overset method however requires very little in pre-processing effort for this case.

6.3 The flow around two rotating cylinders

The cross-flow over two rotating cylinders is considered in this section. Figure 6.24 shows a sketch of the geometry. In the present study, the flow Reynolds number, based on cylinder diameter and freestream velocity magnitude, is equal to 50. This is low enough to ensure that unsteady flow separation should not occur, which may prevent convergence of the steady solver developed here. A dimensionless measure of the rotation rate is given by the ratio $R \cdot \omega / u_\infty$ (where ω is the angular velocity of the rotating cylinders and R is the cylinder radius). This quantity was set to 2 in the present study.

6.3.1 Boundary conditions and numerical model

At the inlet, a uniform velocity has been applied with a magnitude that corresponds to the flow Reynolds number (i.e. $Re = 50$). At the cylinder walls, the no-slip condition is applied; the velocity component normal to the wall is set to zero, while the tangential component is set to twice the inlet velocity (since $R\omega / u_\infty = 2$).

Symmetry boundary conditions have been employed on the upper and lower domain boundaries. This condition is implemented by setting zero gradient conditions for all variables before setting the boundary-normal velocity component to zero.

At the outlet, zero gradient conditions for all flow variables but the pressure is used. The boundary pressure is determined from the bulk correction algorithm described in Section 3.6.1.

All simulations have been conducted with the QUICK convection scheme. Both standard bi-linear interpolation (SI) and the MFBI algorithm have been used for inter-grid interpolation in separate simulations.

6.3.2 Computational Grid

The grids used to simulate this problem are shown in Figure 6.25. The overset simulation is conducted over two identical O-shaped grids in addition to a background Cartesian grid. All computational cells are orthogonal. It is apparent that holes have been cut into the component grids to obtain the overall computational domain. The case in which three grids all overlap one another is again dealt with satisfactorily by the hole cutting algorithm. A total of around $14,000$ active cells make up the computational domain (where active cells exclude those that are cut by the hole cutting algorithm).

6.3.3 Results

Figure 6.26 shows the flow streamlines. It can be seen that the flow does not separate from the cylinder, as would have been expected for a non-rotating case. Instead, the no slip condition causes near-wall fluid to move with the wall, which in turn causes distortion of the cross-flow streamlines via viscous interaction. The tangential velocity at the cylinder wall acts to augment the streamwise velocity component over the lower half of both cylinders, while the opposite is true for the upper half of the cylinders. This velocity differential causes a pressure differential to develop across the cylinder, which generates a force perpendicular to the streamwise direction (this is known as the Magnus effect).

Contours of velocity components in the x and y directions, and contours of pressure are presented in Figures 6.27, 6.28, and 6.29 respectively. From these figures it can be seen that continuous solutions are obtained from one grid to the next for all three primitive variables.

Simulations have been carried out with standard (bi-linear) interpolation, and also with the semi-conservative MFBI method. No appreciable difference in the converged solution was noticed between the two interpolation methods, suggesting that the truncation error in the overlap region is sufficiently low for standard interpolation to perform satisfactorily. It was however noticed that convergence is enhanced and that the solution residuals fall to a lower level through the use of the MFBI method. Figure 6.30 shows a plot of the residuals of the pressure correction equation for the present study and the two interpolation methods used. Even if no appreciable difference in the final converged solution is noticed, the enhanced convergence rate makes the method attractive.

To test the validity of the code, this problem has also been simulated in FLUENT using a block-structured grid, initially with the same number of cells at that of the overset mesh (i.e. 14,000 cells). The block-structured grid that was used is illustrated in Figure 6.31. Both the block-structured grid and the overset grid solutions were tested for grid independence by refining the grids by a factor of two. No discernible difference in the flow field result for the overset simulation was found by taking this step, and hence the initial grids were used. However, for the block-structured grid it was found that a further refinement was required in order to confirm grid independence. Approximately $28,000$ nodes were required to yield satisfactory results on the block-structured grid while $14,000$ nodes was sufficient for the overset simulation. This would suggest that, for this particular case, the overset method offers increased efficiency relative to the block-structured method. The reason for this is likely to be due to the fact that the overset method employed orthogonal grids for this problem, whereas for the block-structured grid there were abrupt changes in cell alignments at block interfaces which introduces error.

It can be seen from Figure 6.31 that the block-structured grid is fairly complex in comparison to the overset grid, even for this relatively simple geometry, thereby highlighting the overset grid method's ability to simplify the pre-processing stage. A total of 13 blocks are used in order to decompose the domain.

Profiles of U, V and P have been taken through the wake, at $3R$ from the cylinder's rotation axis. Plots of these profiles, along with comparisons with the FLUENT simulation, are presented in Figures 6.32, 6.33 and 6.34. From these plots it is apparent that the present code results are broadly in agreement with those of FLUENT, with only minor differences present. The small differences are likely to be attributable to differences between the two grids; the solution was deemed grid independent when the effect of further refinement was small, not when it is zero. The plotting of profiles (rather than contours say) exaggerates any small discrepancy that may be present.

6.4 Summary

The flow over two rotating cylinders has been simulated with overset grids. Continuous solutions are obtained from one grid to the next, thereby giving some support to the validity of the interpolation algorithm and overset method in general.

Results have been compared against a block-structured simulation. It is found that only very minor differences between the two solutions is found, which is presumably due to slight grid dependencies that have not been identified in the grid sensitivity test. In any case, the differences are minor.

6.5 Flow impinging onto a concave surface

This test case consists of a fluid entering into a semi-circular chamber, through an inlet slot positioned at the centre of the semi-circle's straight edge. The fluid impinges onto the chamber's concave wall, and then follows the contour of the wall, before exiting through one of the two outlet slots. Both outlets are also positioned on the semi-circle's straight edge, one at the leftmost side of this edge, and the other at the rightmost side. Figure 6.35 illustrates this setup.

6.5.1 Geometry and computational grid

The outlets are both 0.5 inlet widths wide, and the radius of the concave wall is equal to 3.125 inlet widths. The Reynolds number, based on bulk inlet velocity and inlet width, was equal to $5,000$. The Reynolds averaged Navier-Stokes equations were closed using the low-Reynolds number $k-\epsilon$ model of Launder and Sharma [2].

The governing equations were solved over two overset grids, as illustrated in Figure 6.36. The (red) inlet grid had a resolution of 60×180 nodes while the (green) curvilinear grid used 65×350 nodes in the radial and azimuth directions respectively. All near-wall nodes were placed at a distance of approximately 3.5×10^{-5} inlet widths from the wall. This resulted in a $y+$ value of less than unity for *all* near wall nodes.

6.5.2 Boundary conditions and numerical model

On the inlet boundary, uniform profiles for velocity and turbulence quantities were specified. The turbulence intensity was set to 5 % with a turbulent to laminar viscosity ratio of 8. The velocity was set to a value consistent with the flow Reynolds number of $5,000$.

The outlet boundaries had a prescribed uniform pressure applied across the boundary, while the velocity variables were extrapolated from the interior of the domain.

Simulations have been conducted with the QUICK convection scheme for the velocity components, and the UMIST scheme for the turbulence variables.

6.5.3 Results

Contours of the flowfield variables, U and V, along with the predicted pressure field, are given in Figures 6.37, 6.38 and 6.39 respectively. From these figures it can be seen that continuous solutions are obtained from one grid to the next.

Contours of the turbulent kinetic energy, k, (normalised by the inlet value of k), and of the turbulent to molecular viscosity ratio, μ_t/μ, are presented in Figures 6.40 and 6.41 respectively. Again, it can be seen that continuous solutions are obtained, thereby demonstrating the ability of the method to deal with turbulent flows.

6.6 Figures

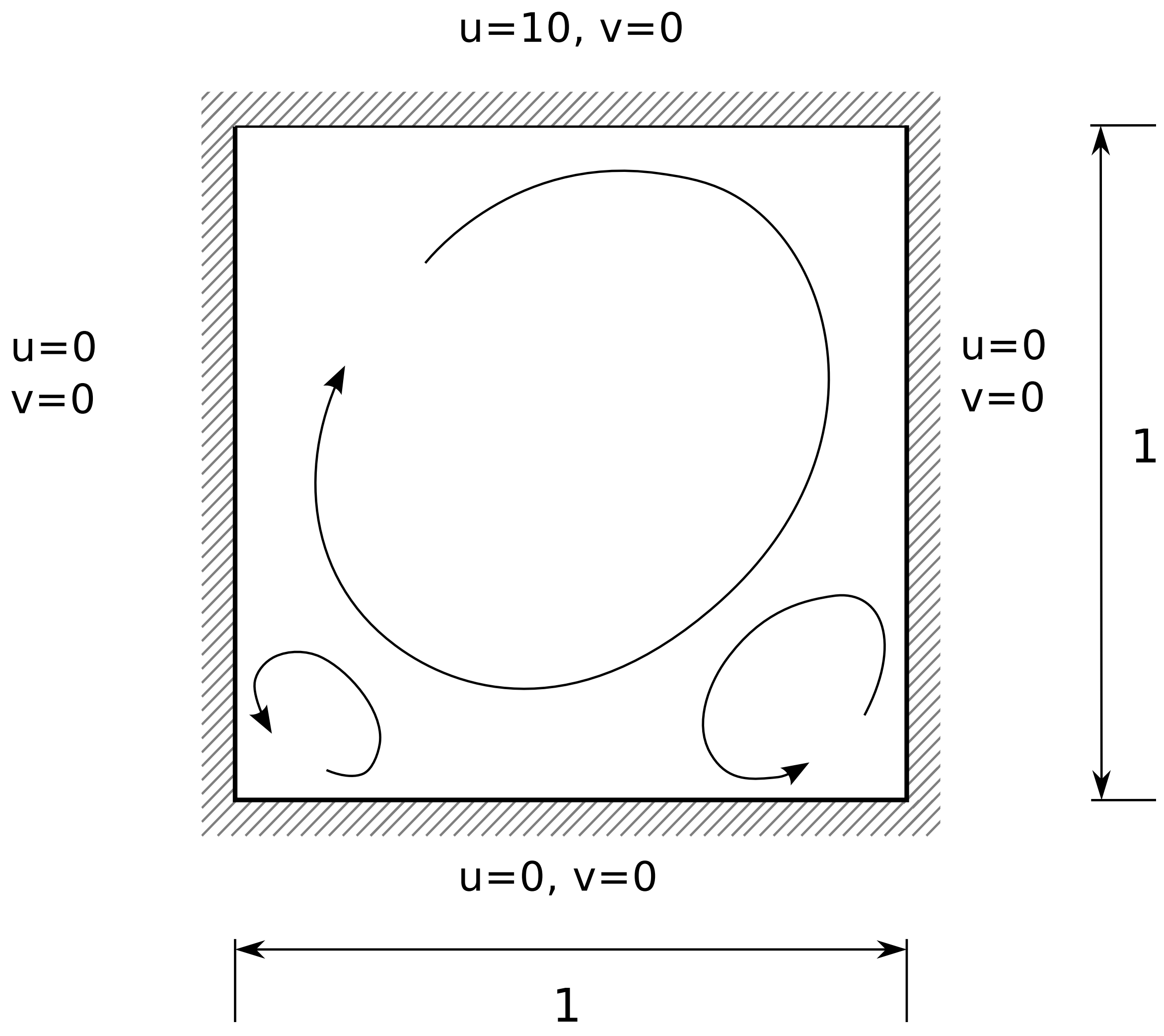


Figure 6.1: A schematic of the lid-driven cavity problem with the physical boundary conditions.

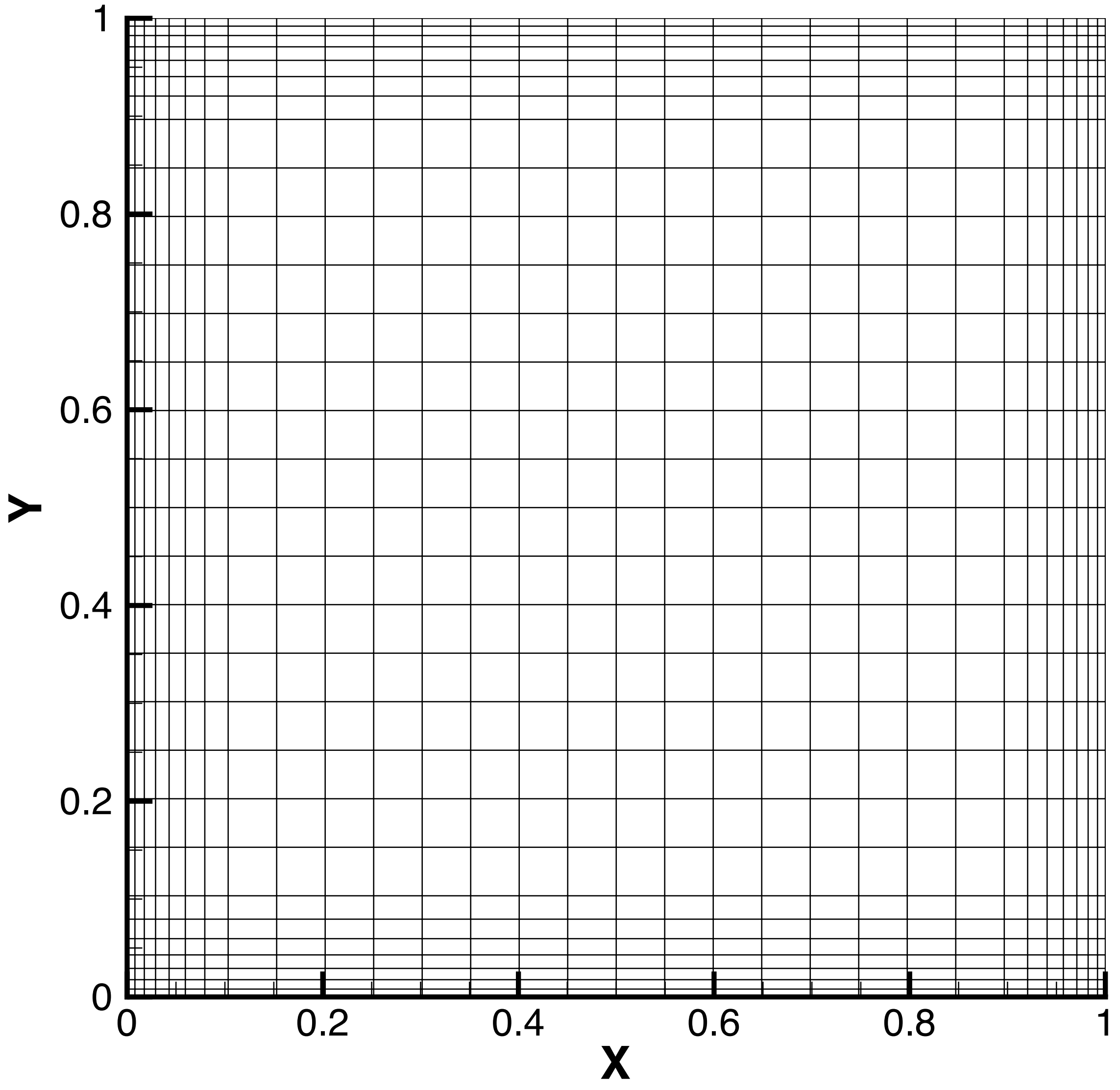


Figure 6.2: The 30×30 mesh used for single block simulations.

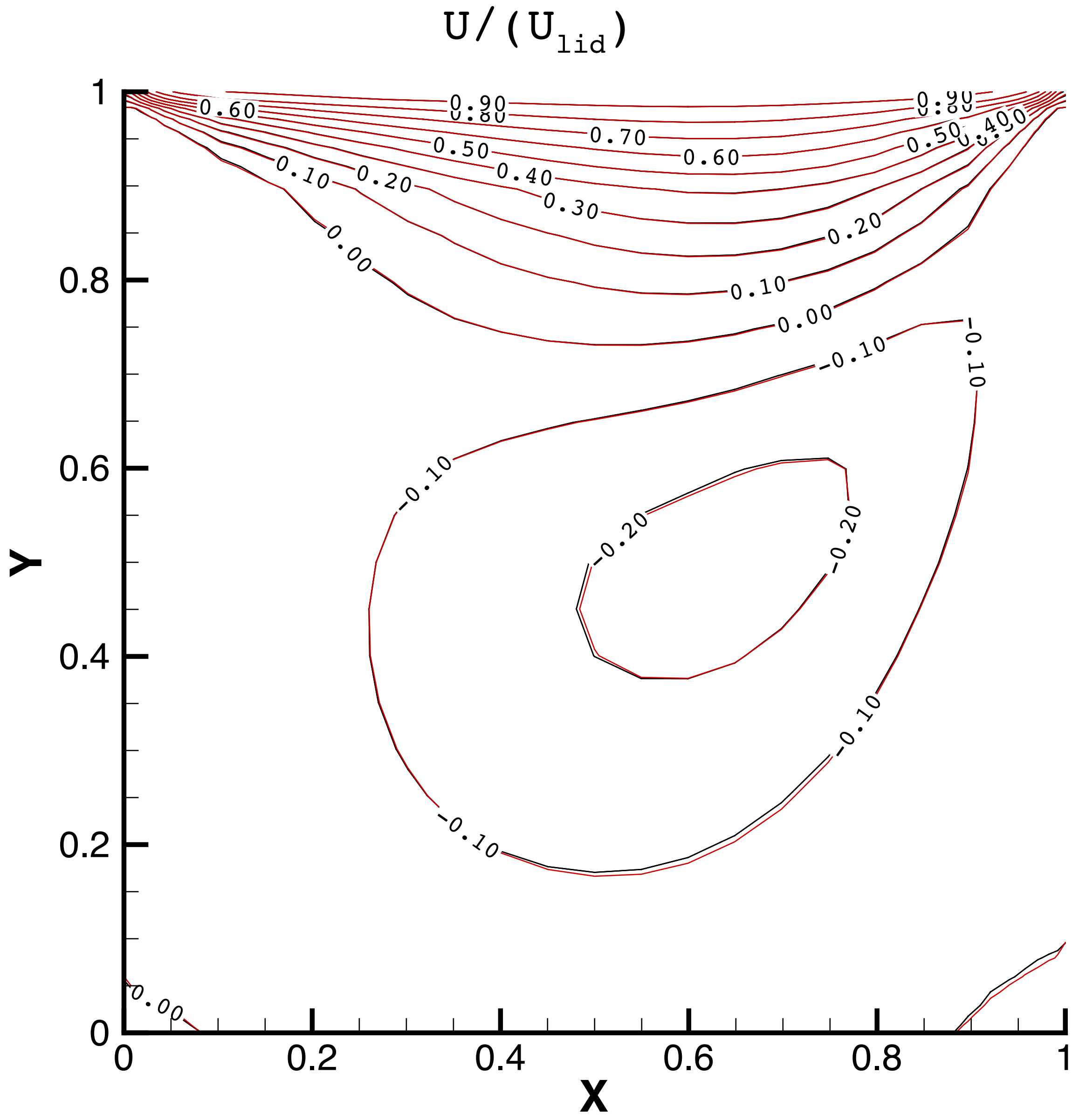


Figure 6.3: Predicted x-velocity contours for single block lid driven cavity computation, normalised by lid velocity. $Re = 100$. Present code (red line) is compared with the commercial CFD package Fluent (black line).

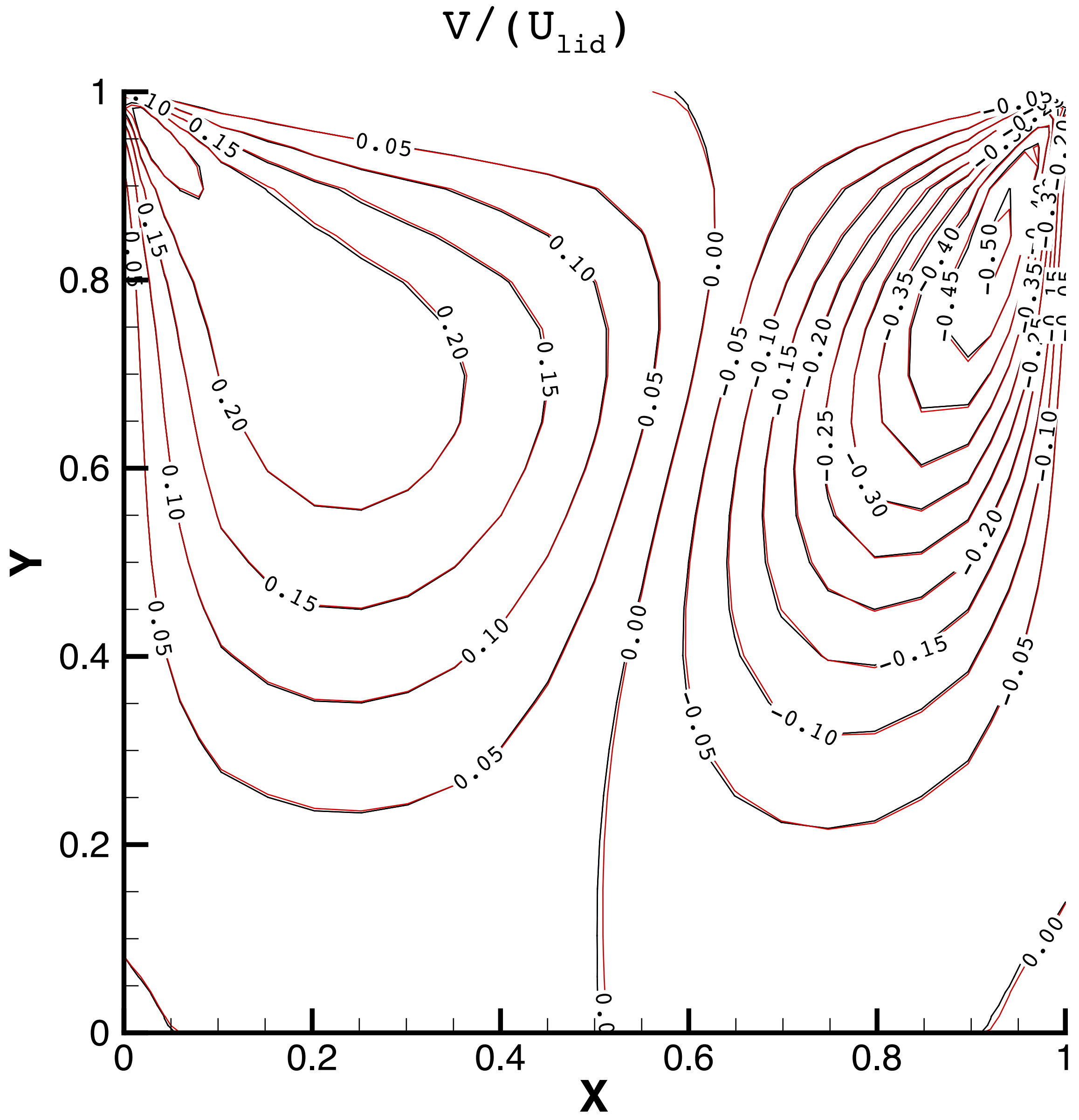


Figure 6.4: Predicted y-velocity contours for single block lid driven cavity computation, normalised by lid velocity. $Re = 100$. Present code (red line) is compared with the commercial CFD package Fluent (black line).

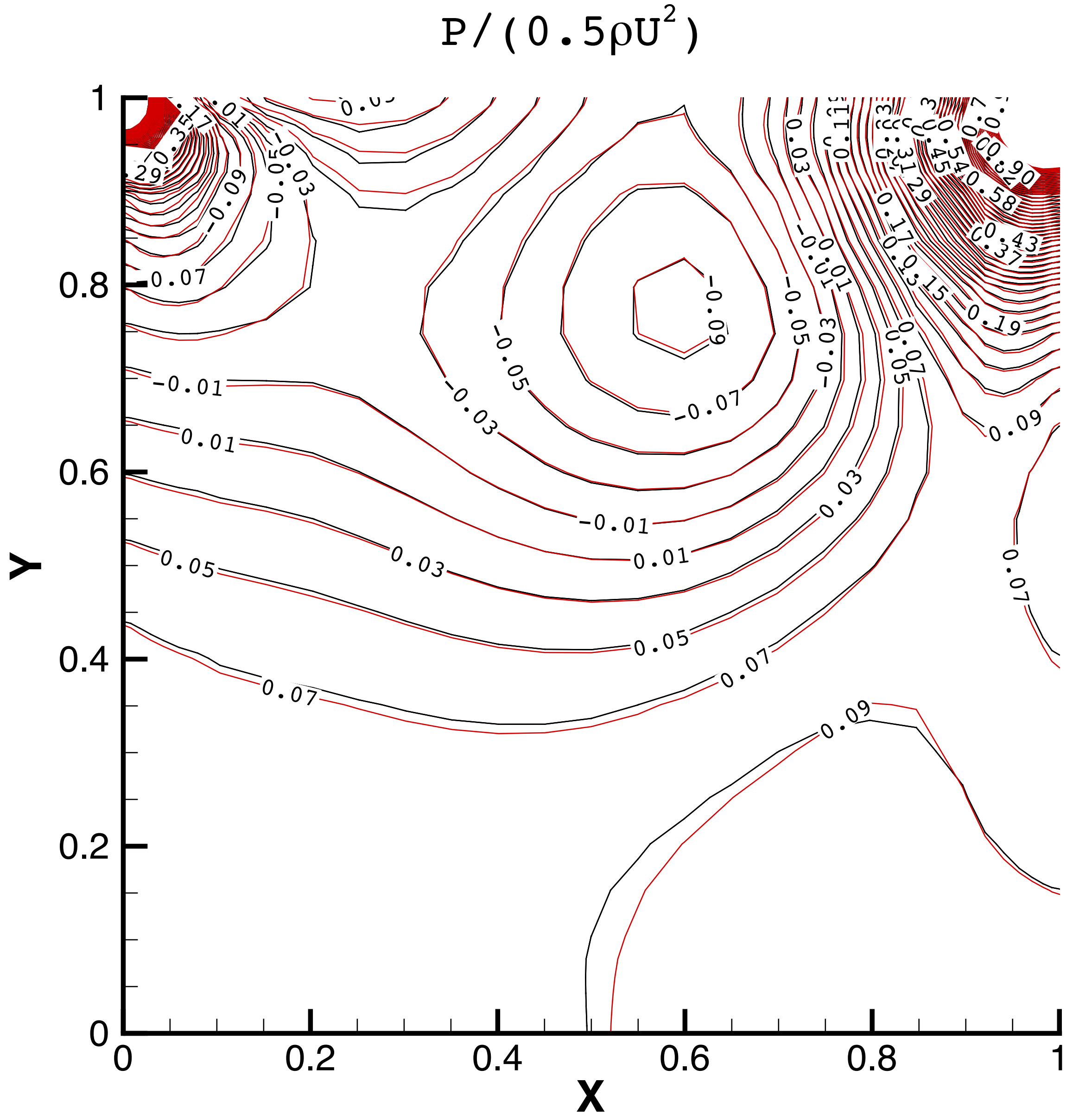


Figure 6.5: Predicted pressure contours for single block lid driven cavity computation, normalised by $0.5\rho U_{lid}^2$. $Re = 100$. Present code (red line) is compared with the commercial CFD package Fluent (black line).

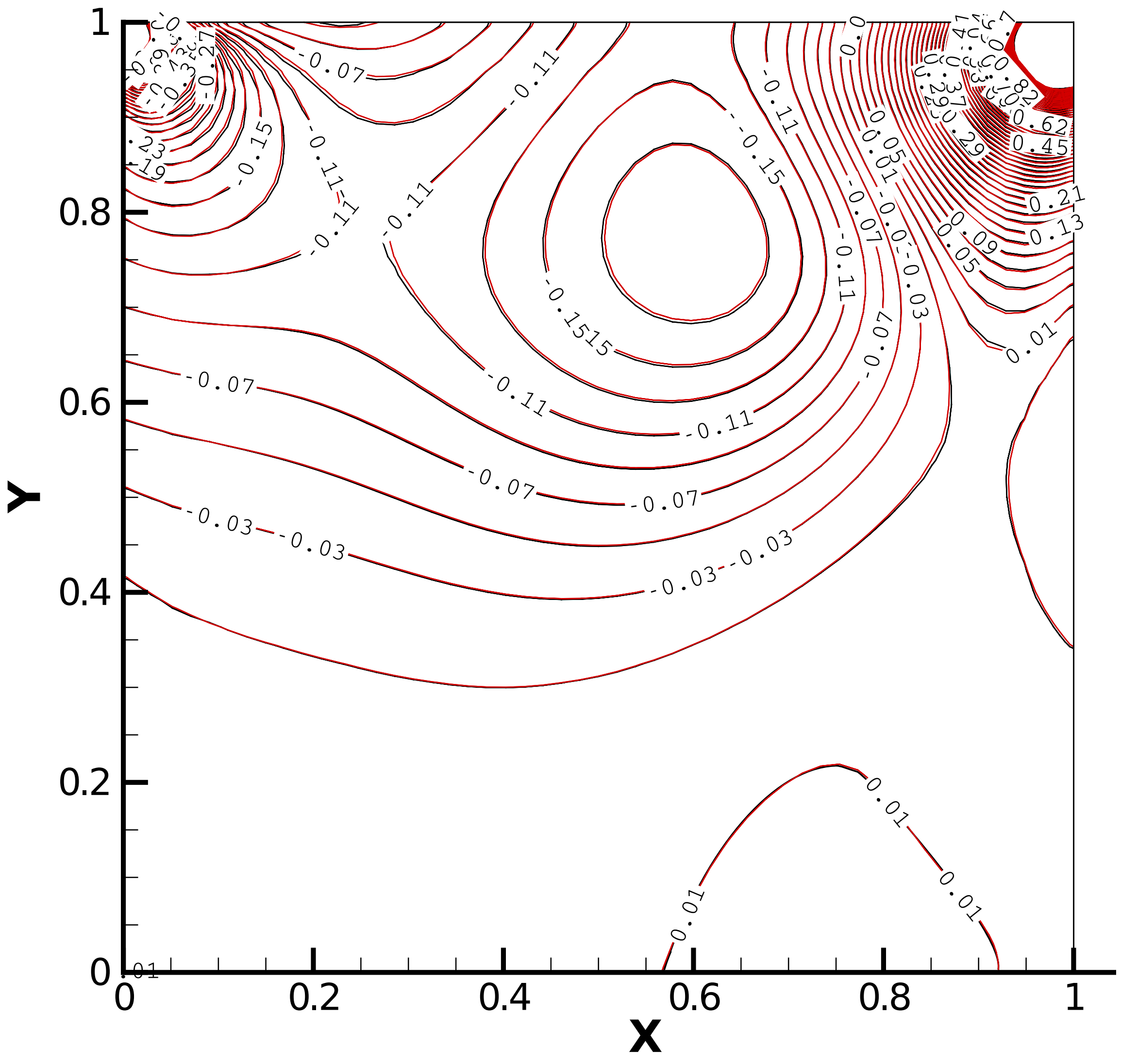


Figure 6.6: Predicted pressure contours for single block lid driven cavity computation when using a 52×52 cell mesh. $Re = 100$. Present code (red line) is compared with the commercial CFD package Fluent (black line).

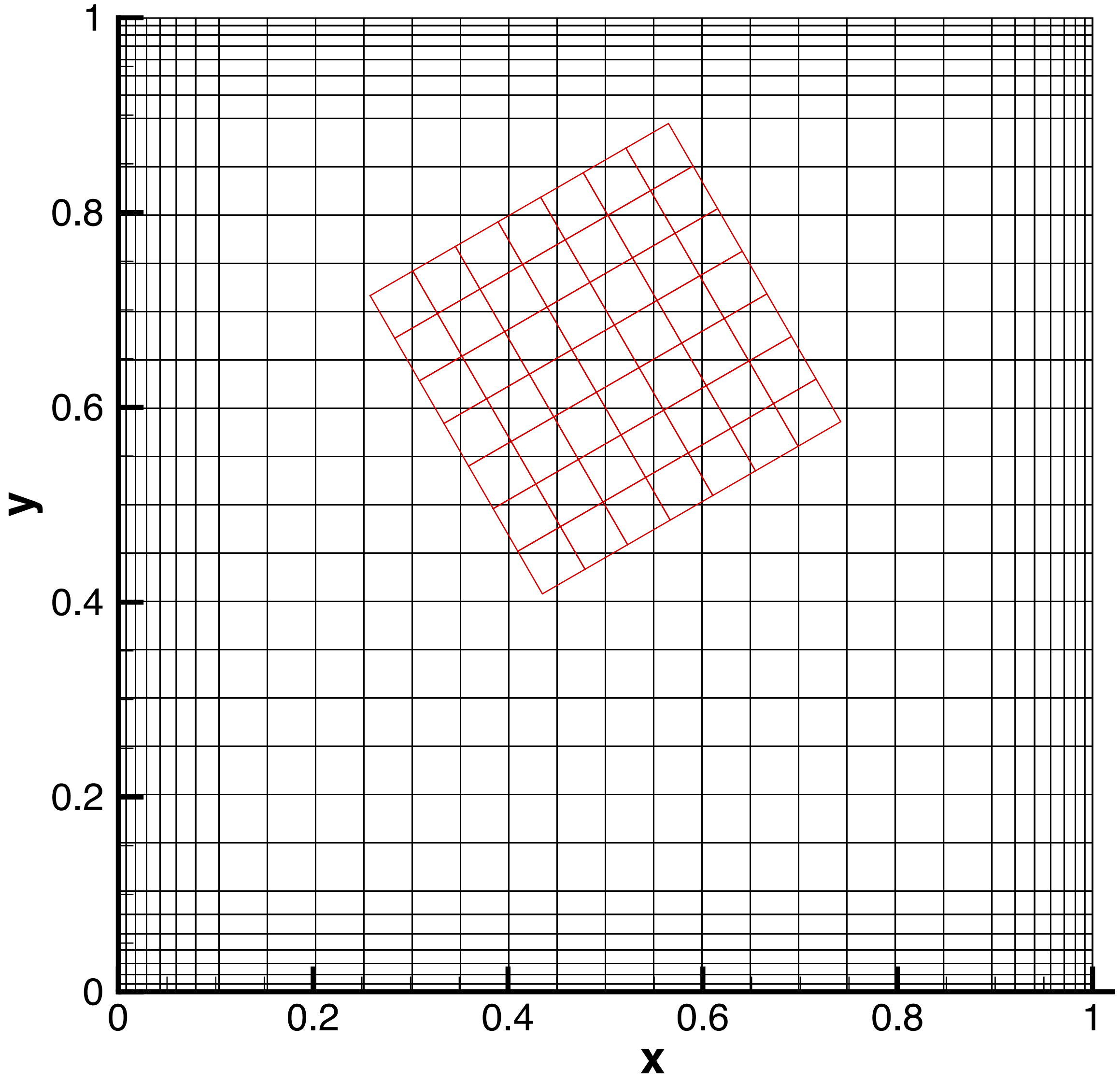


Figure 6.7: Mesh used for one-way information transfer test.

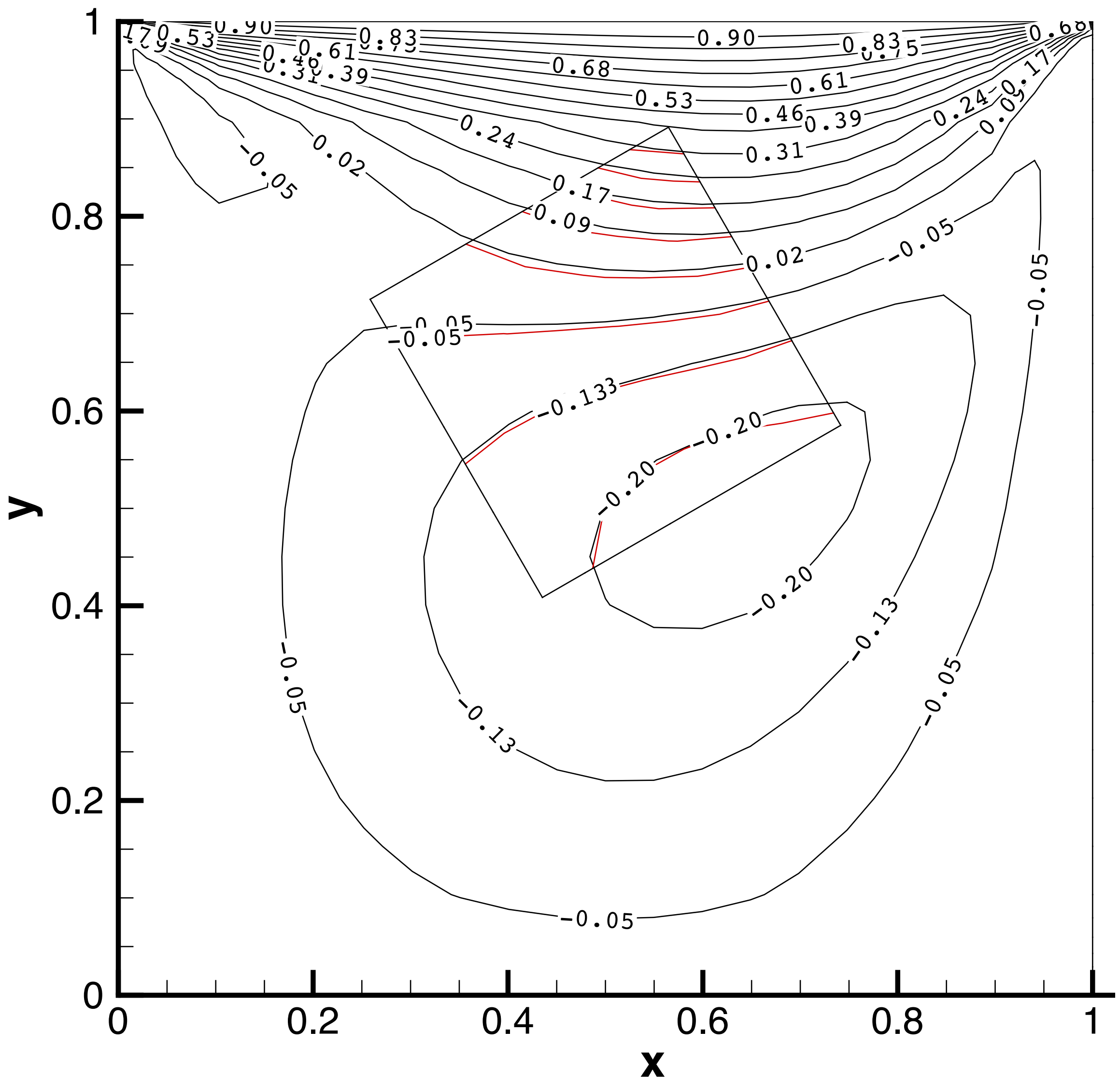


Figure 6.8: Predicted x-velocity contours for one-way information transfer test. $Re = 100$. **Solution on background grid is shown in black, while the solution on the overset grid is shown in red.**

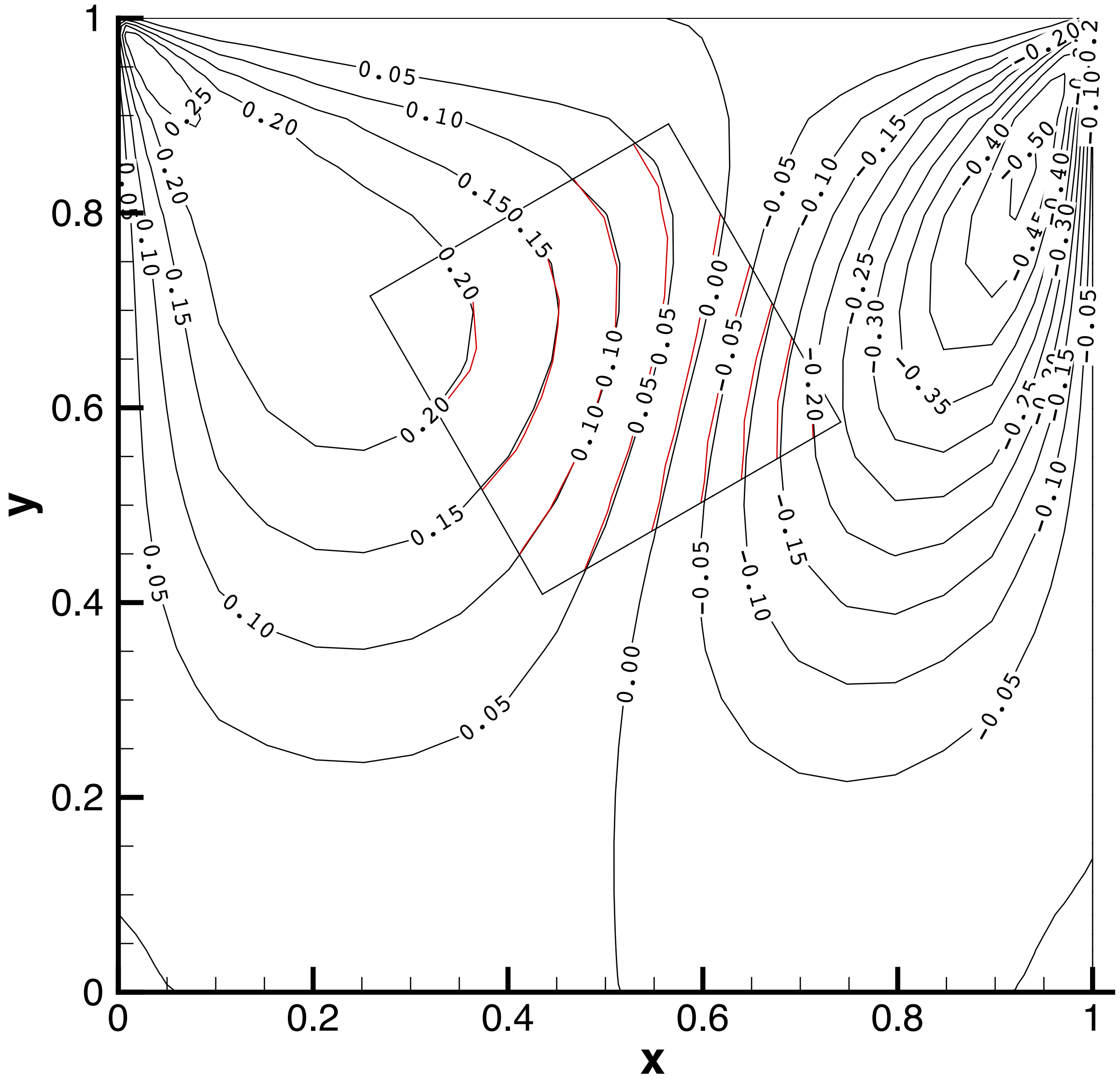


Figure 6.9: Predicted y-velocity contours for one-way information transfer test. $Re = 100$. **Solution on background grid is shown in black, while the solution on the overset grid is shown in red.**

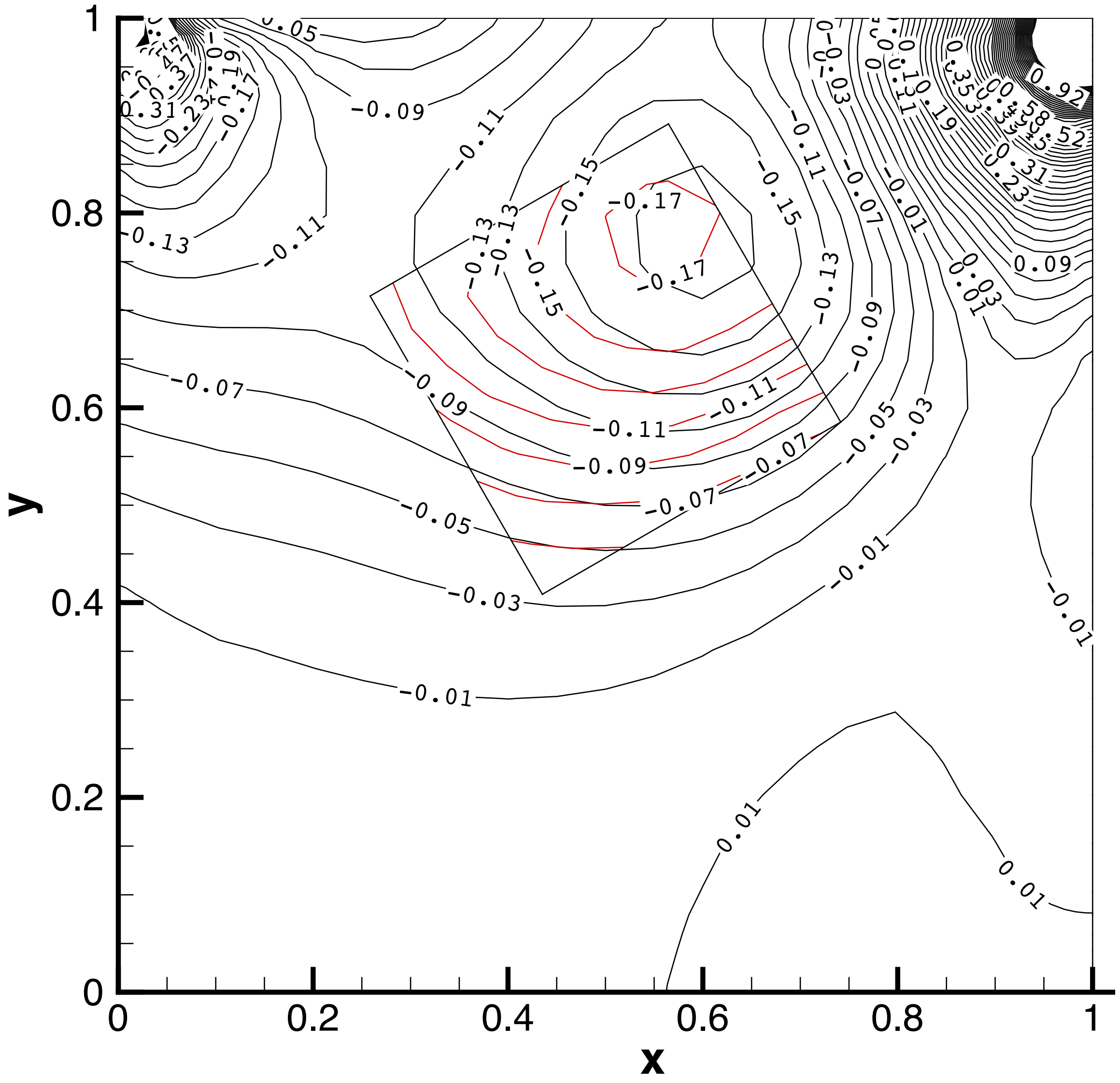


Figure 6.10: Predicted pressure contours for one-way information transfer test. $Re = 100$. **Solution on background grid is shown in black, while the solution on the overset grid is shown in red.**

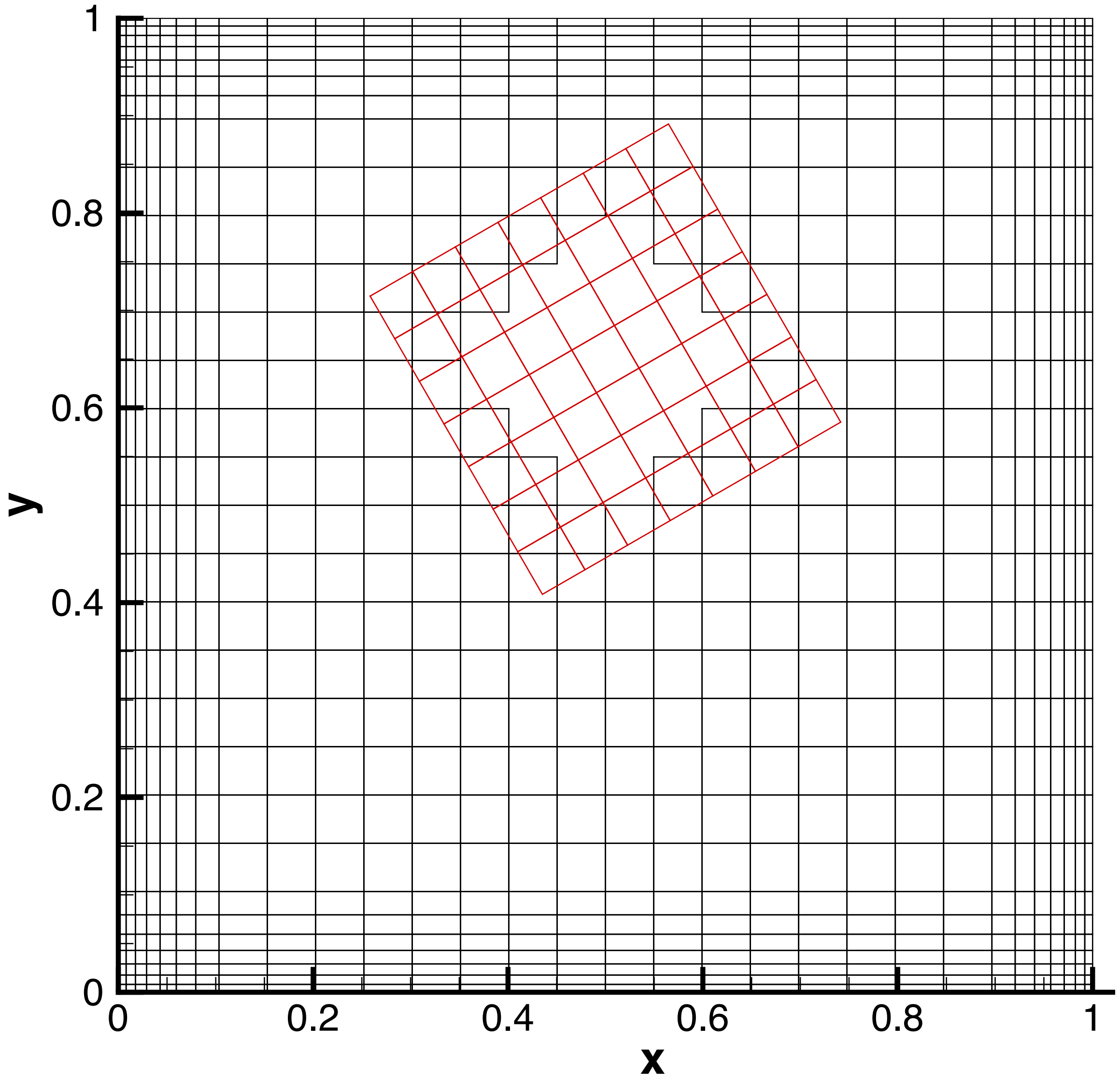


Figure 6.11: Mesh used for two-way information transfer test.

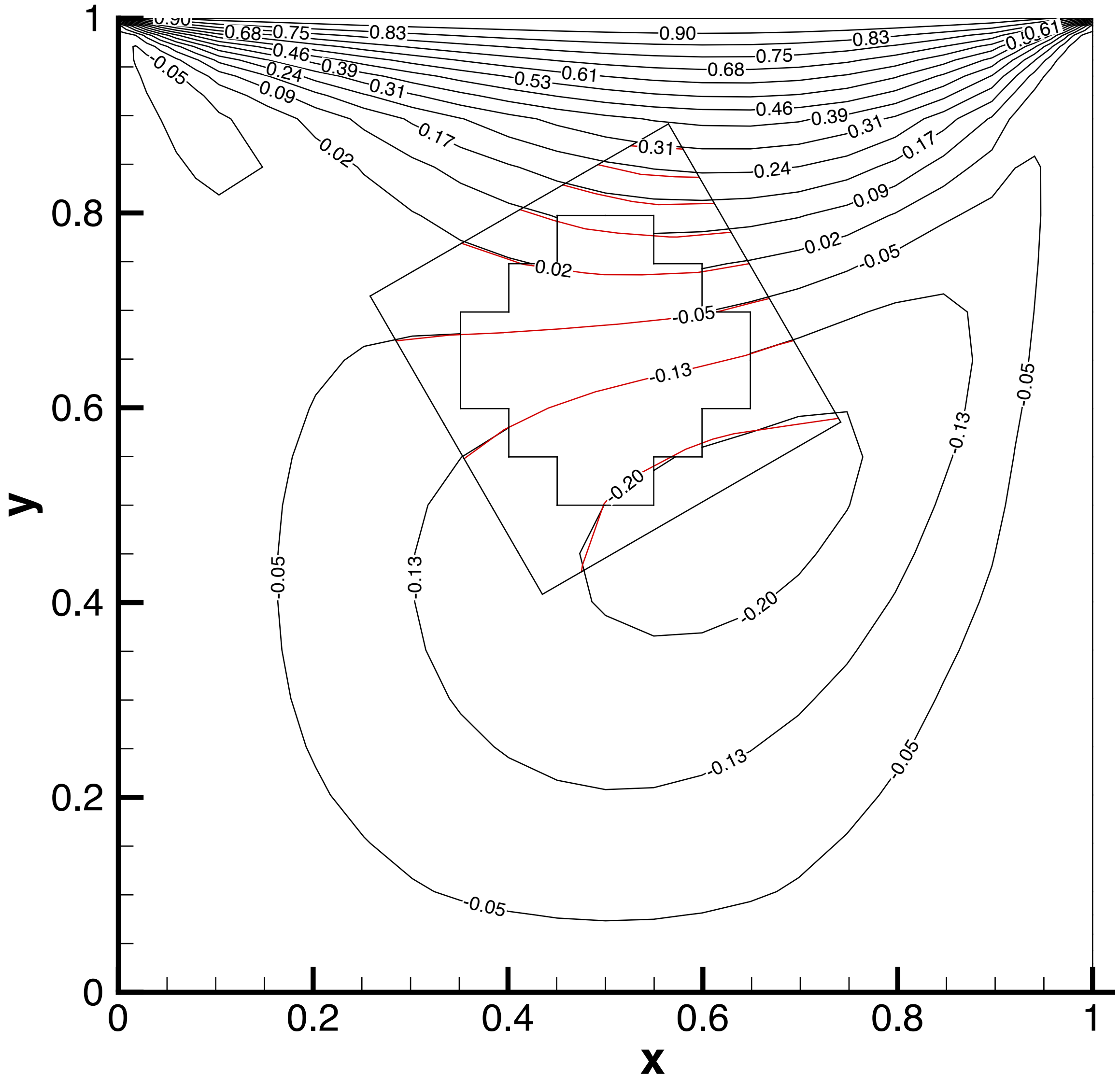


Figure 6.12: Predicted x-velocity contours for two-way information transfer test. $Re = 100$. Solution on background grid is shown in black, while the solution on the overset grid is shown in red.

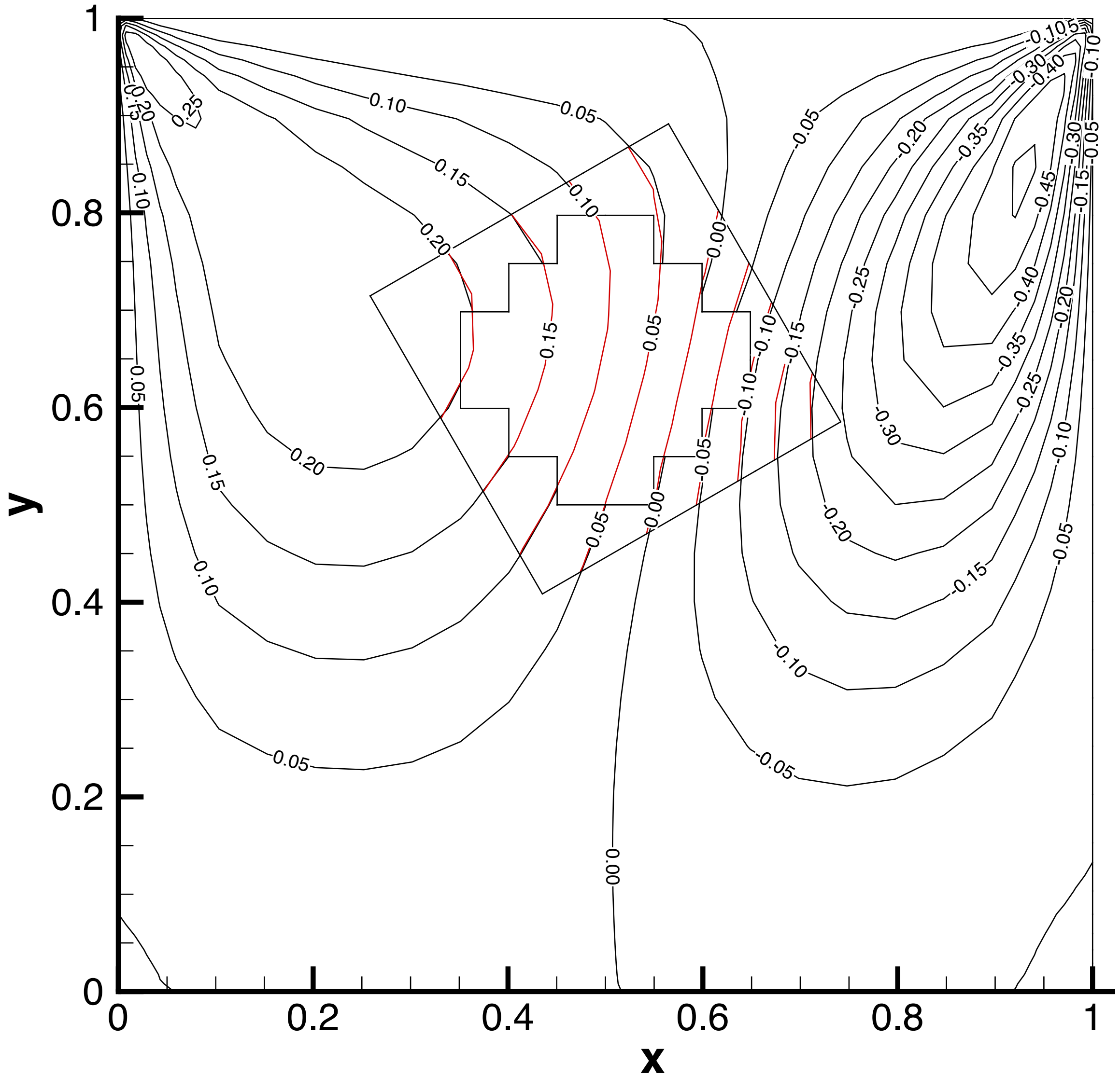


Figure 6.13: Predicted y-velocity contours for two-way information transfer test. $Re = 100$. **Solution on background grid is shown in black, while the solution on the overset grid is shown in red.**

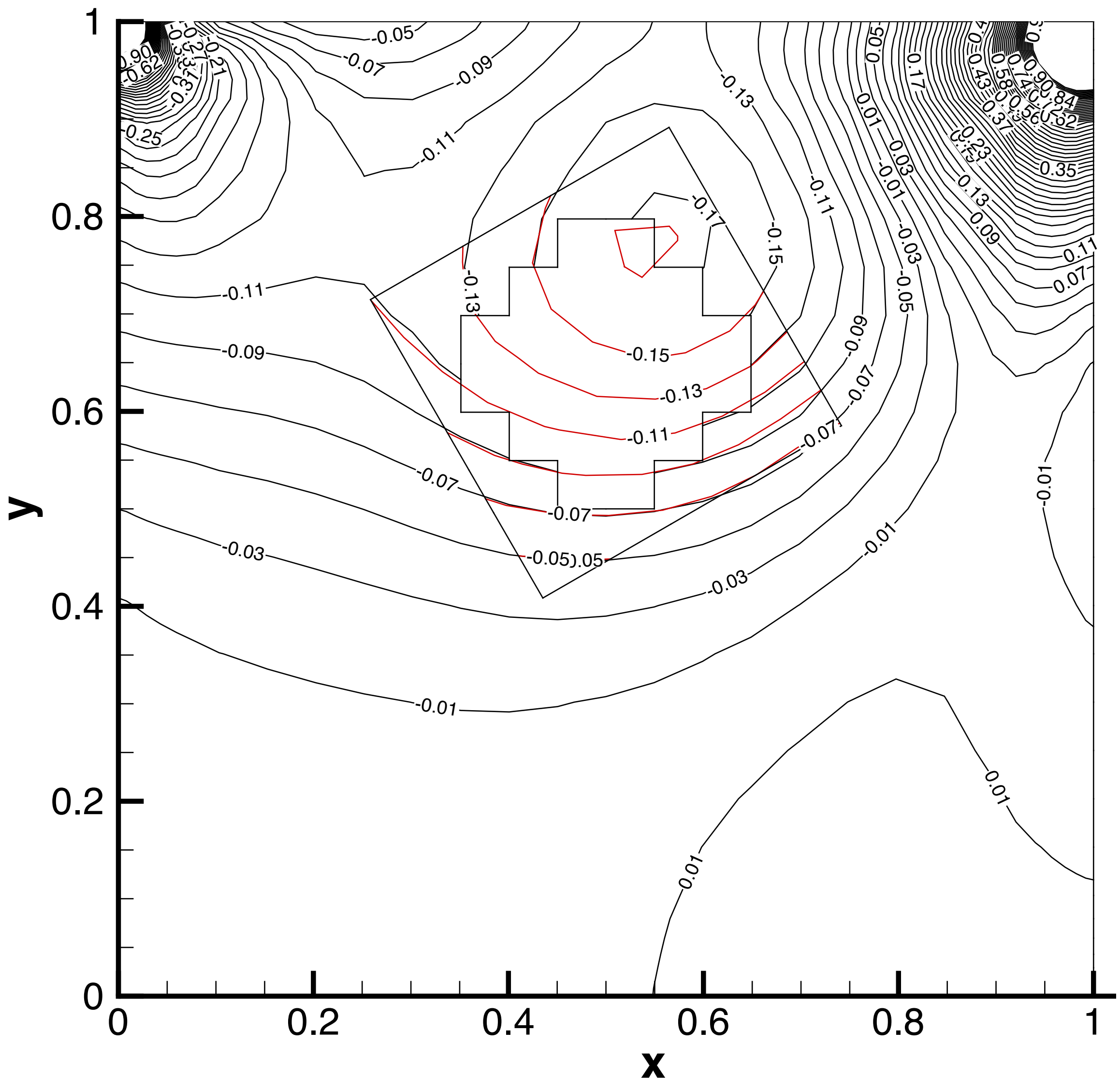


Figure 6.14: Predicted pressure contours for two-way information transfer test. $Re = 100$. **Solution on background grid is shown in black, while the solution on the overset grid is shown in red.**

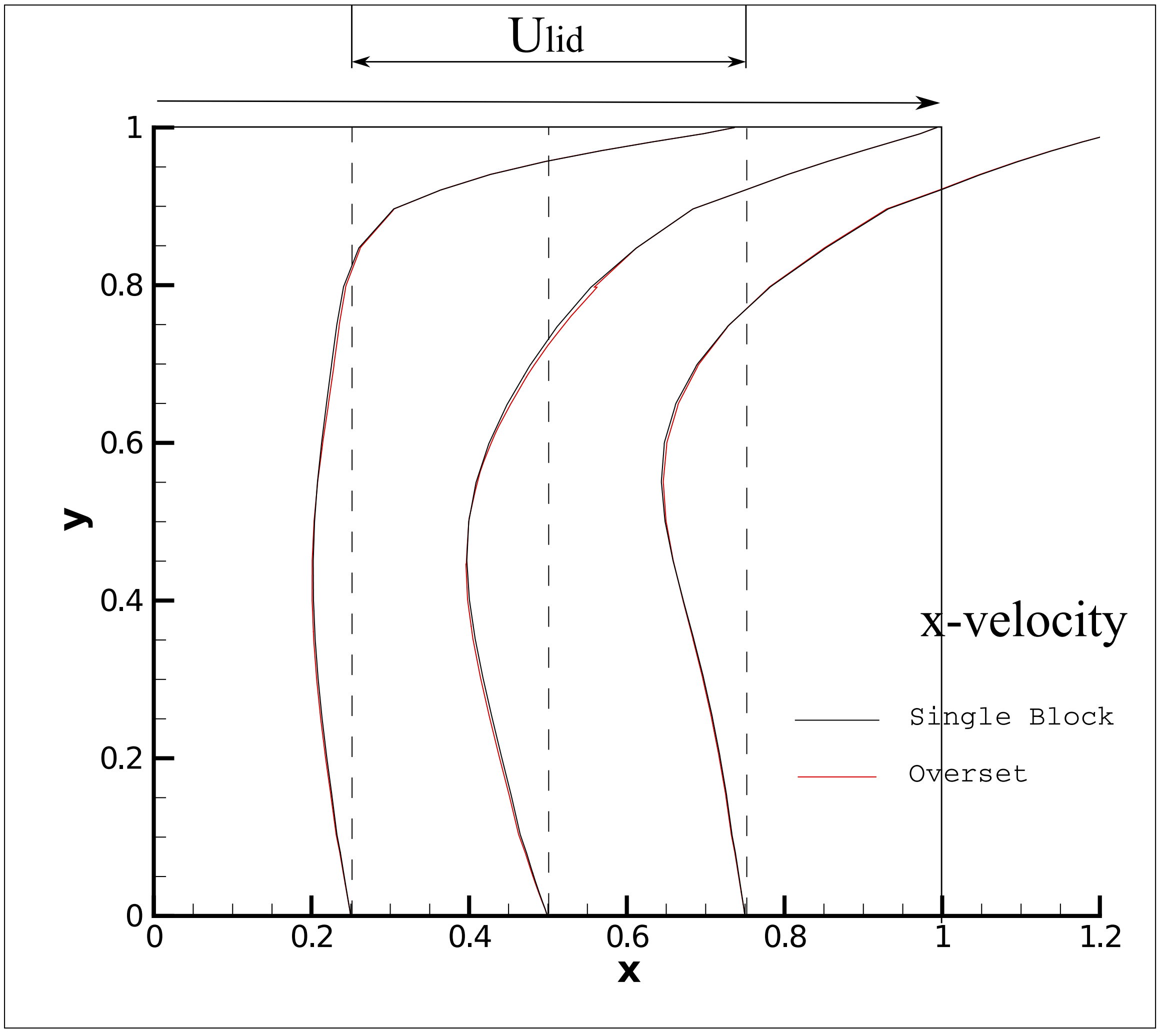


Figure 6.15: Profiles of x-velocity taken vertically through cavity at $x/L = 0.25$, 0.5 and 0.75. Overset grids are compared with single block solution. Distance between adjacent locations where profiles are taken corresponds to half the lid velocity.

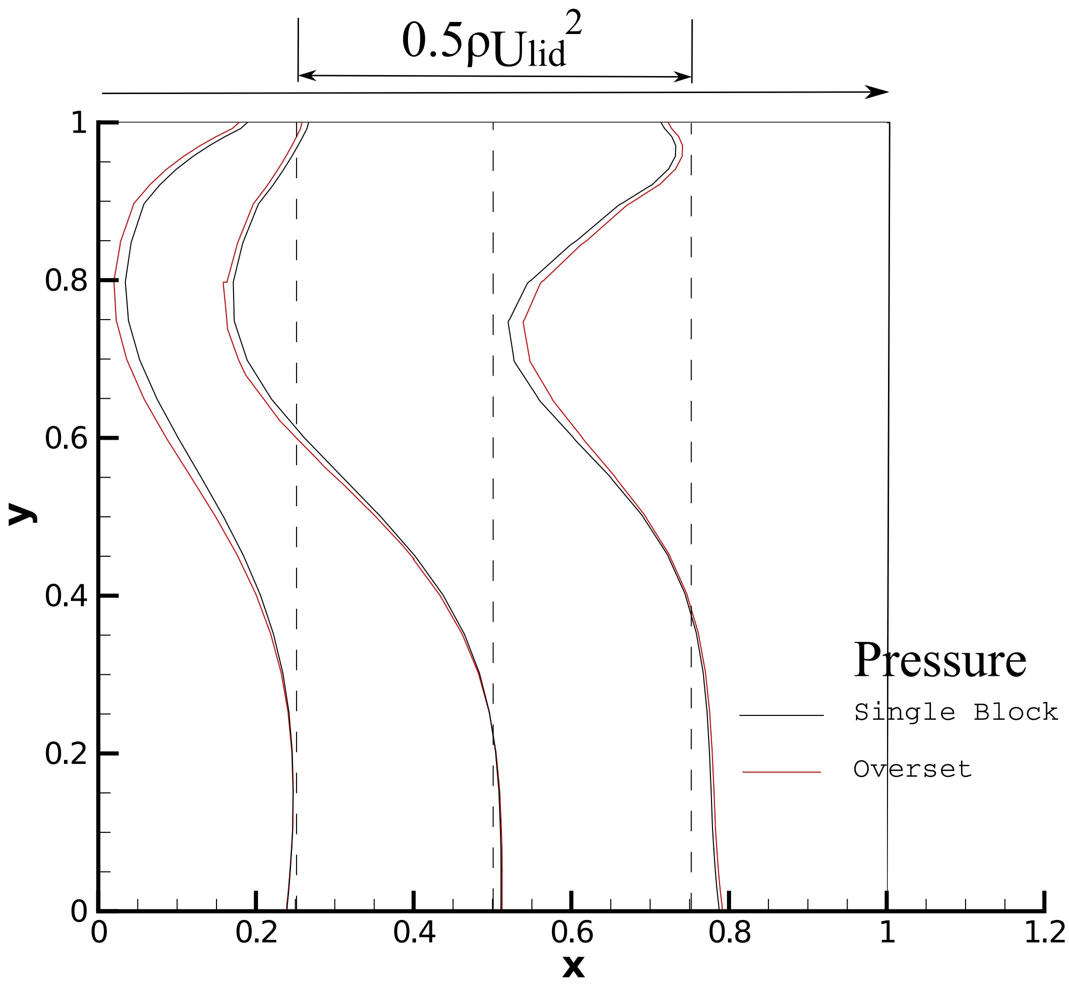
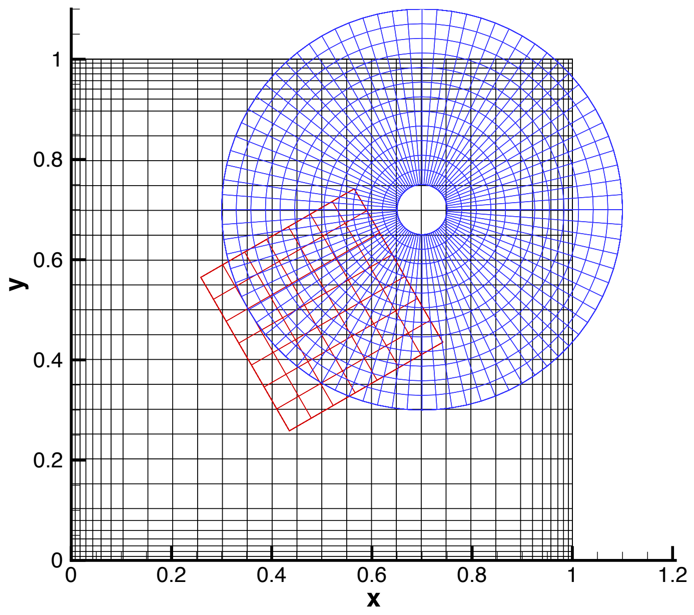


Figure 6.16: Profiles of pressure taken vertically through cavity at $x/L = 0.25$, 0.5 and 0.75. Overset grids are compared with single block solution. Distance between adjacent locations where profiles are taken corresponds to half the dynamic pressure (based on lid velocity).



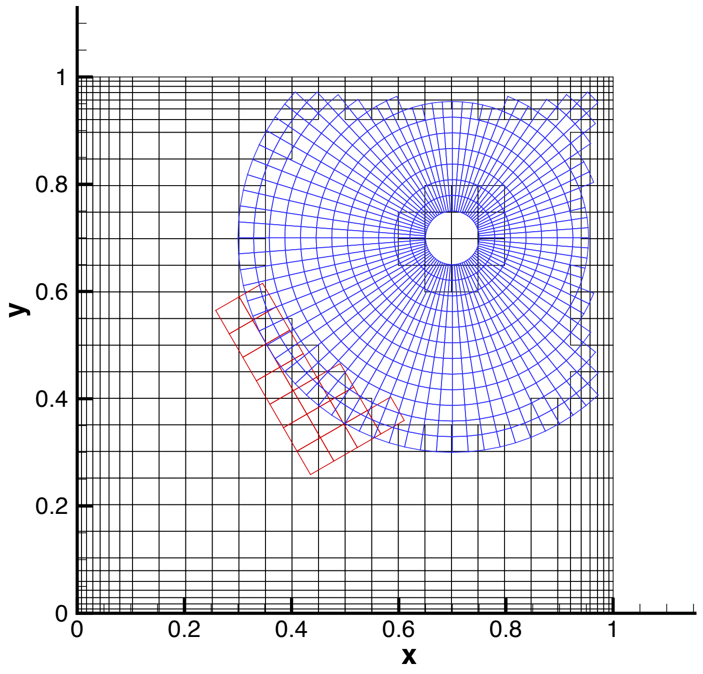


Figure 6.17: Mesh used for multiple overset grids test. User generated sub-grids (above). Overset mesh after automatic hole cutting (below).

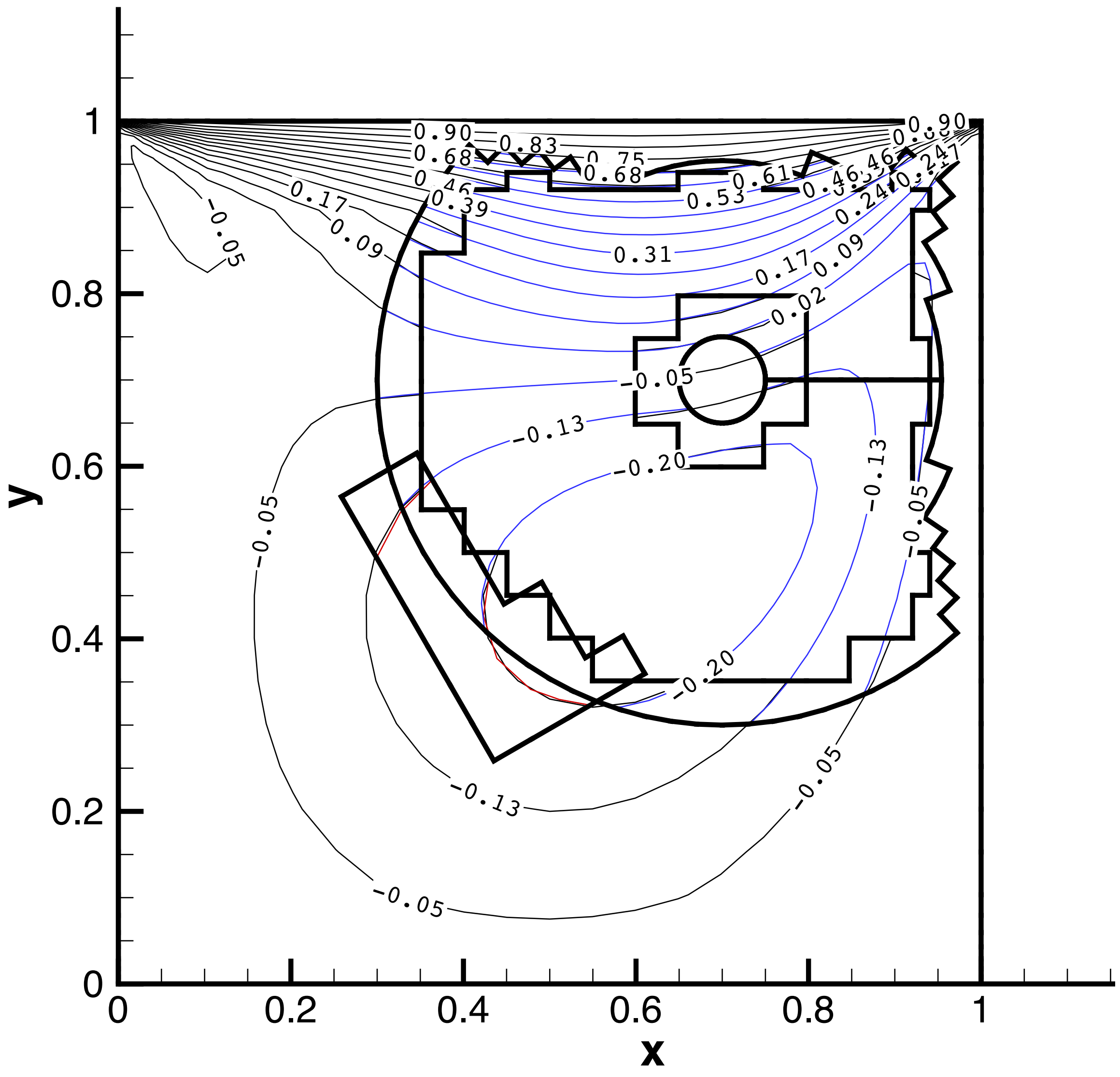


Figure 6.18: Predicted x-velocity contours for multiple overset grids test. $Re = 100$. Solution on background grid is shown in black, while the solution on overset grids is shown in red and blue for the square and circle respectively.

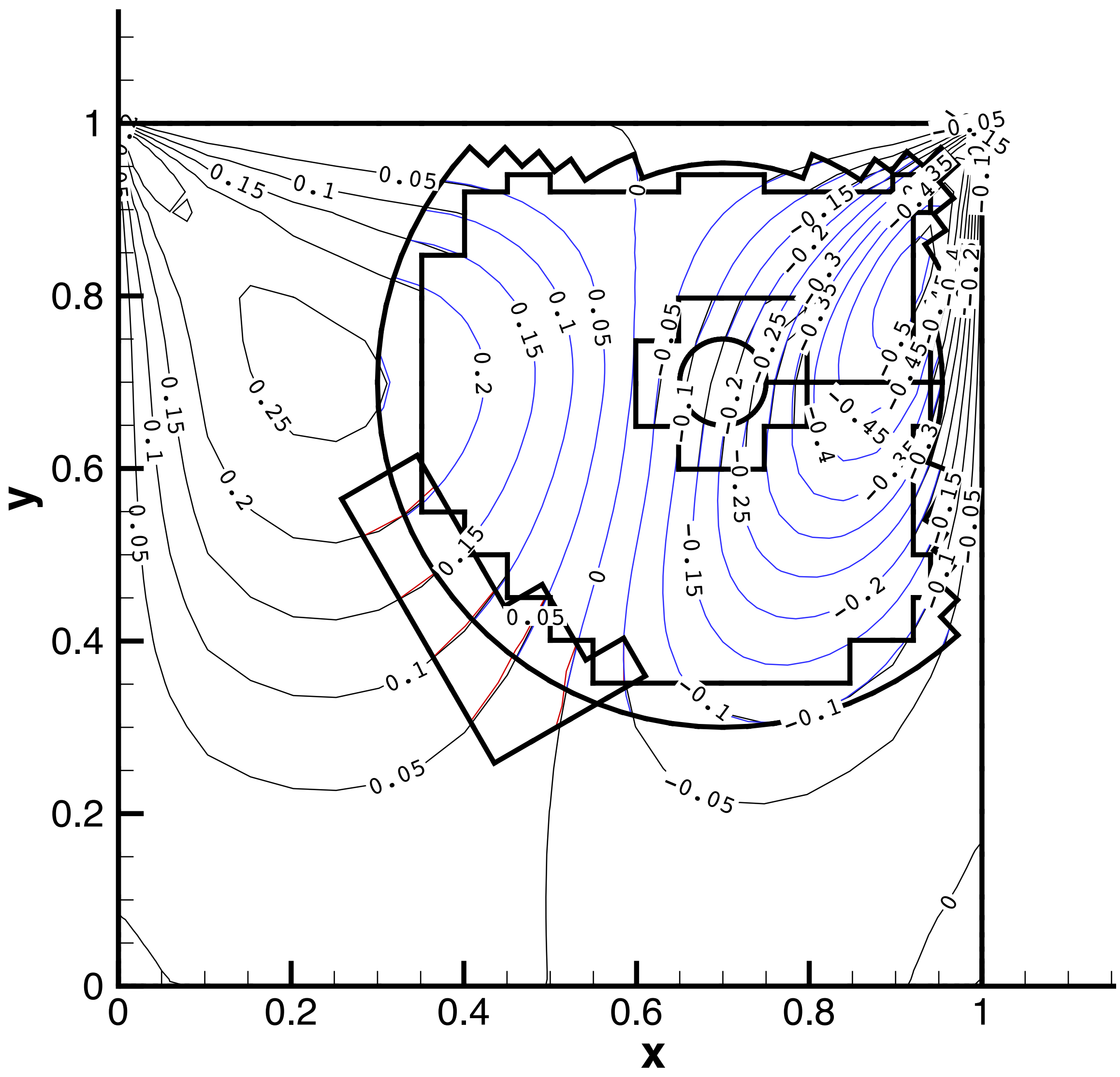


Figure 6.19: Predicted v-velocity contours for multiple overset grids test. $Re = 100$. **Solution on background grid is shown in black, while the solution on overset grids is shown in red and blue.**

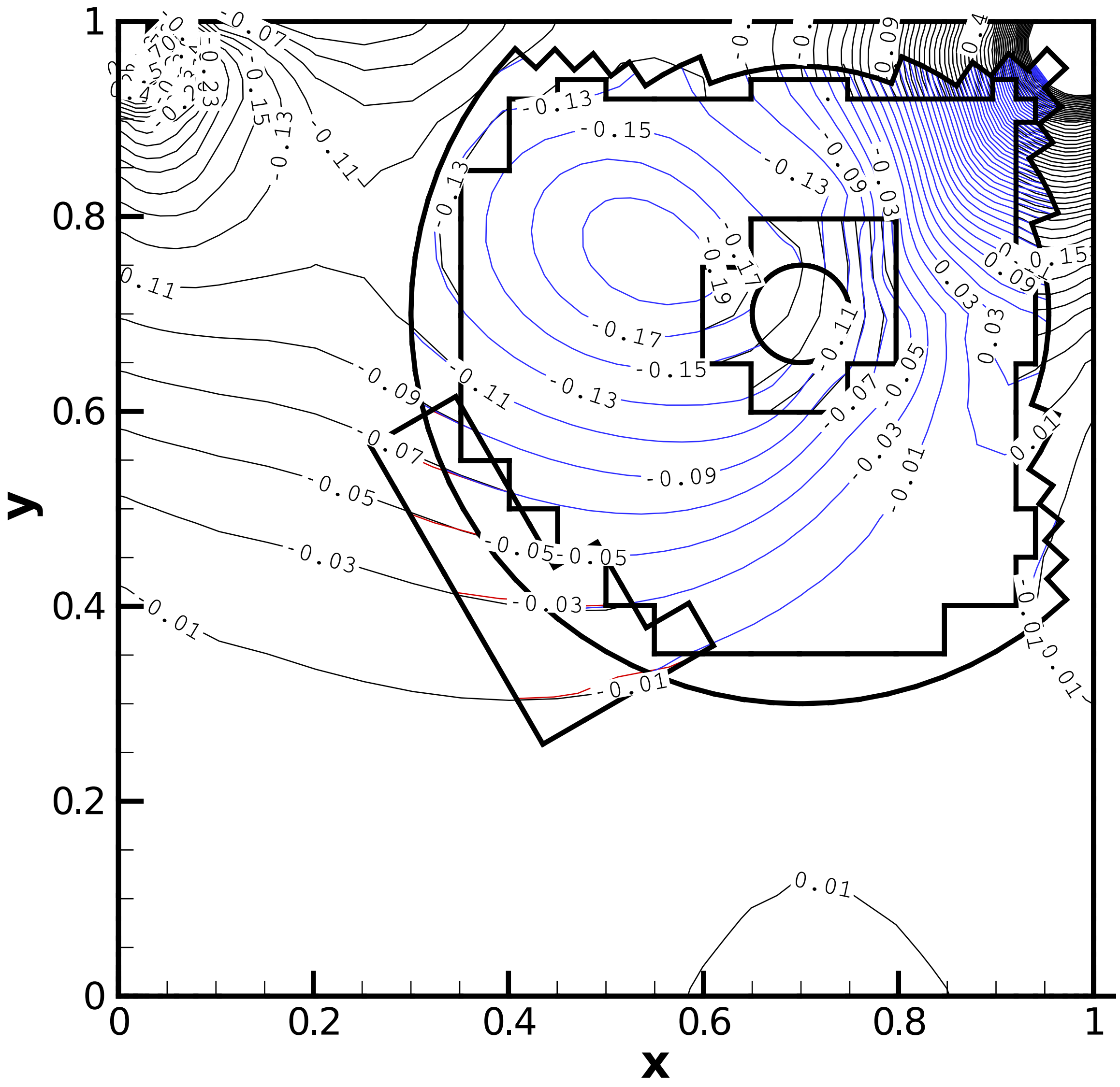


Figure 6.20: Predicted pressure contours for multiple overset grids test. $Re = 100$. **Solution on background grid is shown in black, while the solution on overset grids is shown in red and blue.**

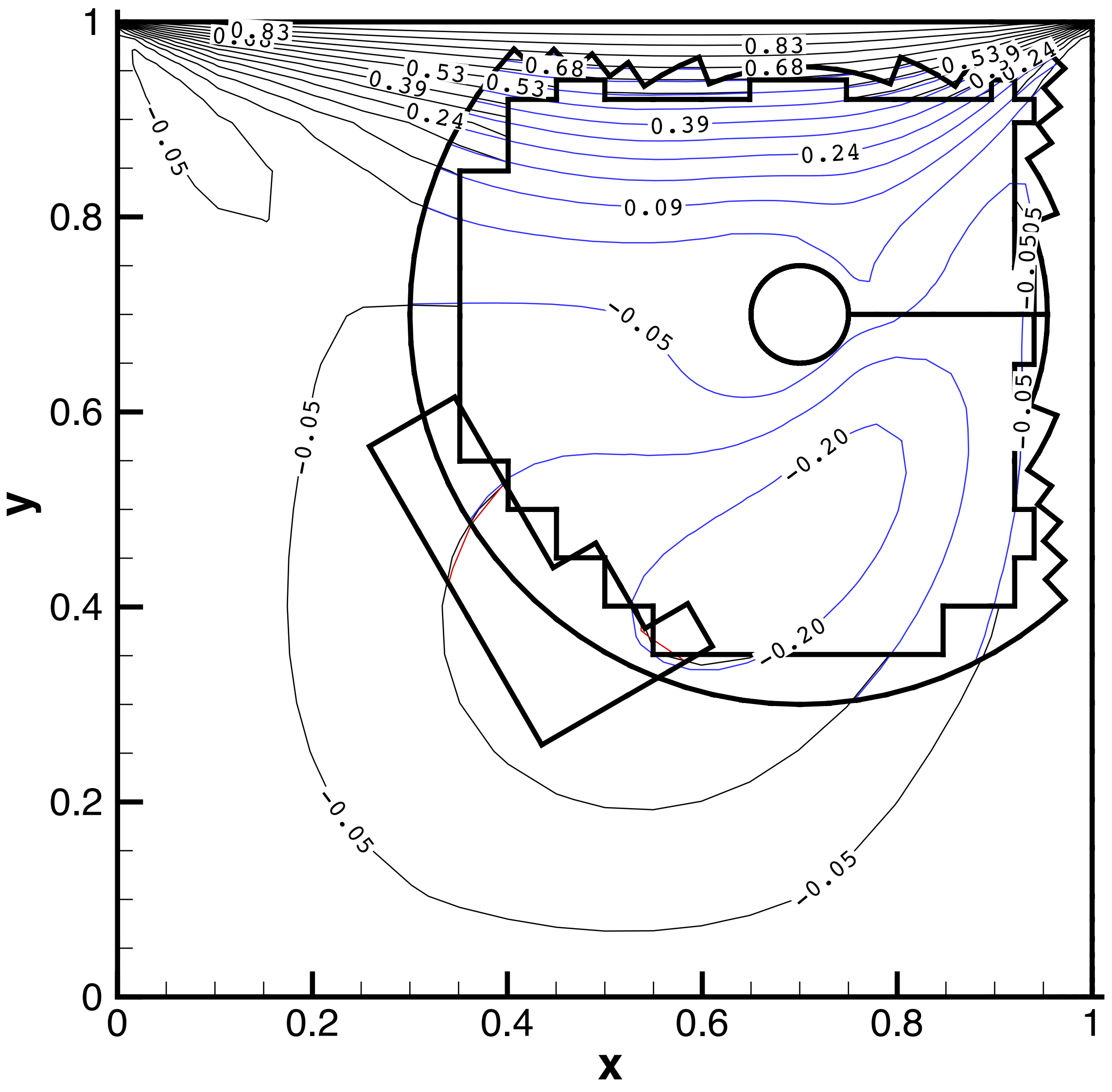


Figure 6.21: Predicted x-velocity contours for the cavity with obstruction case. $Re = 100$. **Solution on background grid is shown in black, while the solution on overset grids is shown in red and blue for the square and circle respectively.**

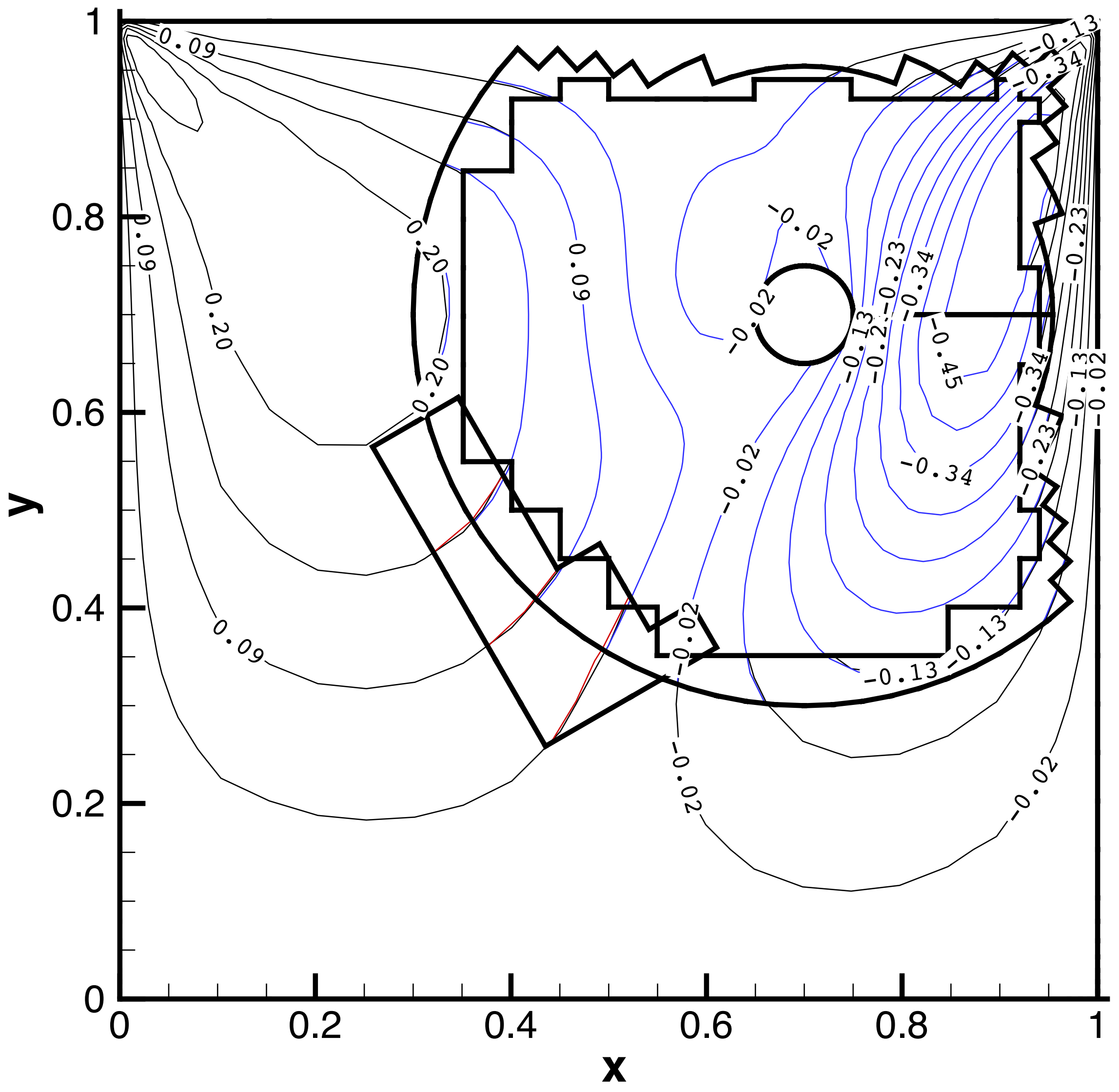


Figure 6.22: Predicted v-velocity contours for for the cavity with obstruction case. $Re = 100$. **Solution on background grid is shown in black, while the solution on overset grids is shown in red and blue.**

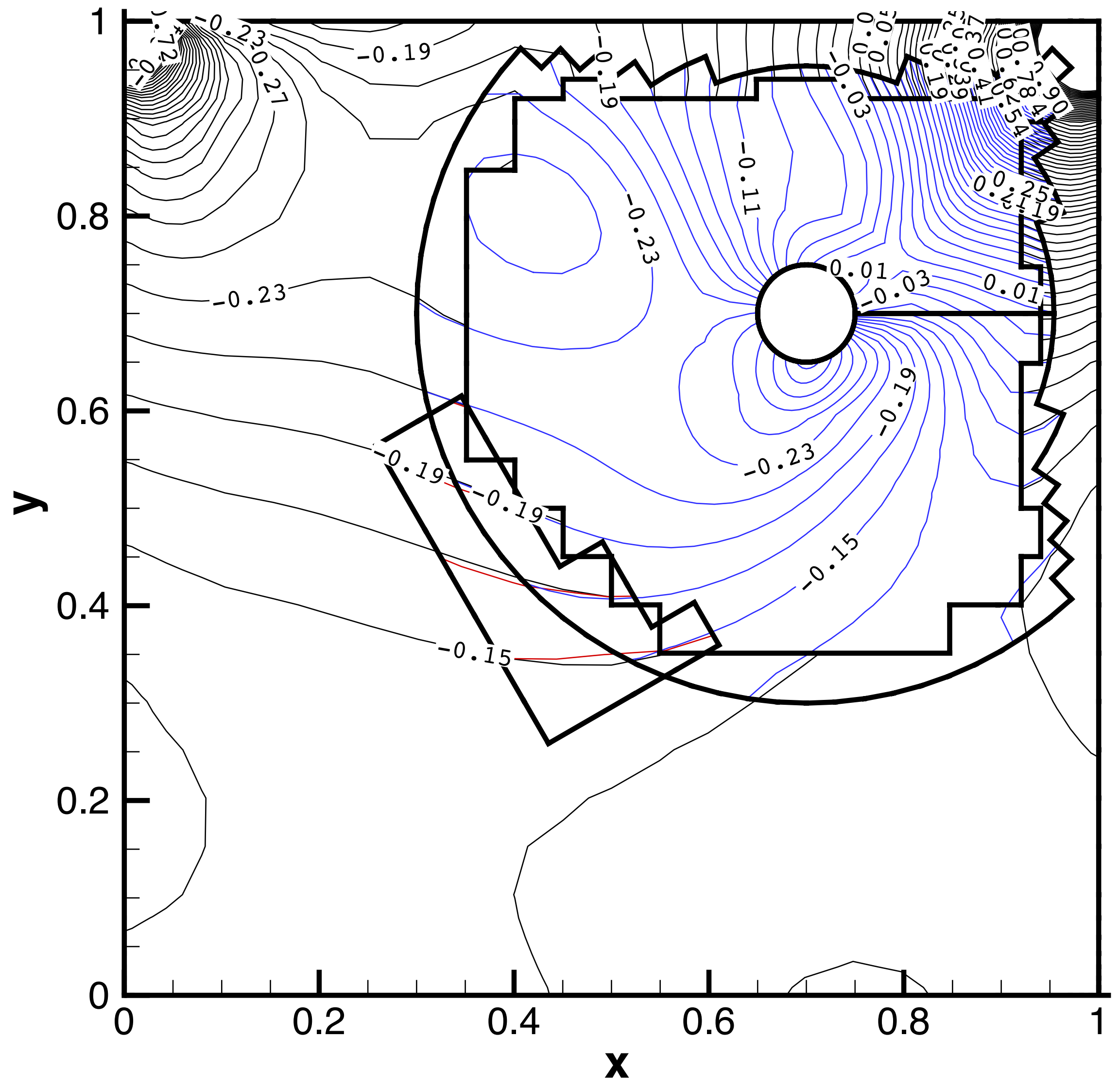


Figure 6.23: Predicted pressure contours for for the cavity with obstruction case. $Re = 100$. **Solution on background grid is shown in black, while the solution on overset grids is shown in red and blue.**

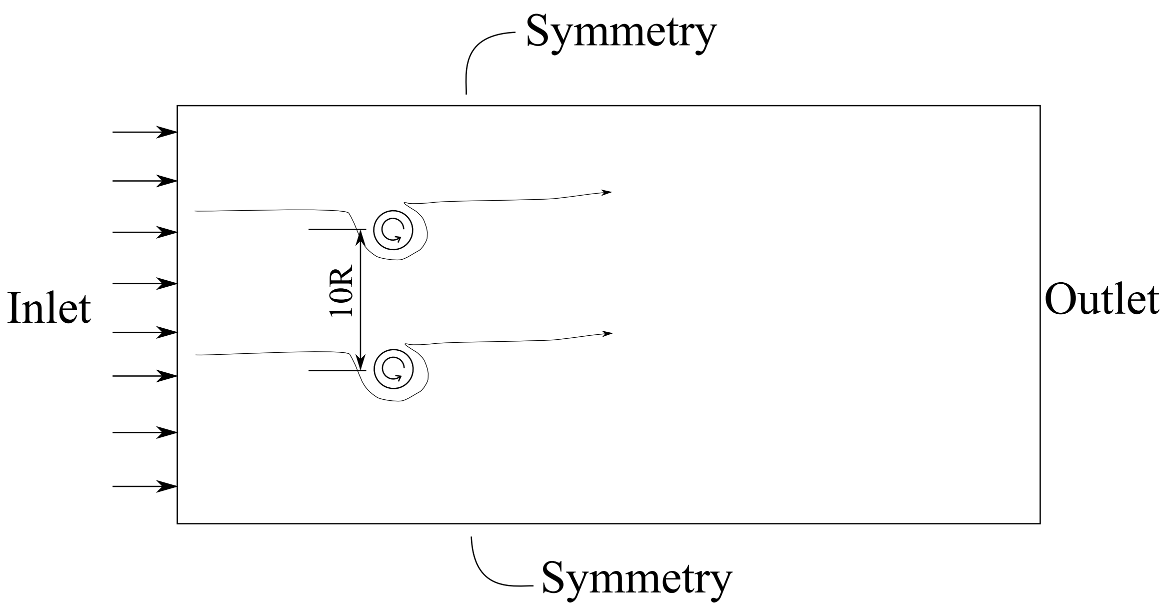


Figure 6.24: Schematic of the two-rotating cylinders case with boundary conditions applied.

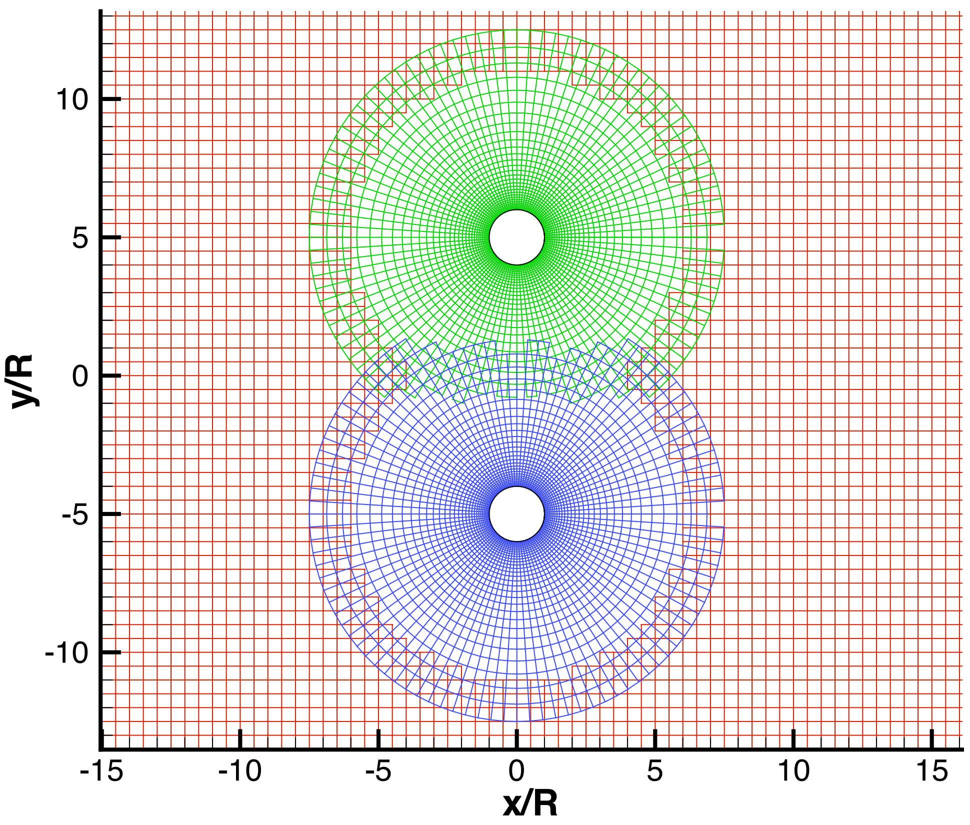


Figure 6.25: Overset meshes used for the flow around two rotating cylinders.

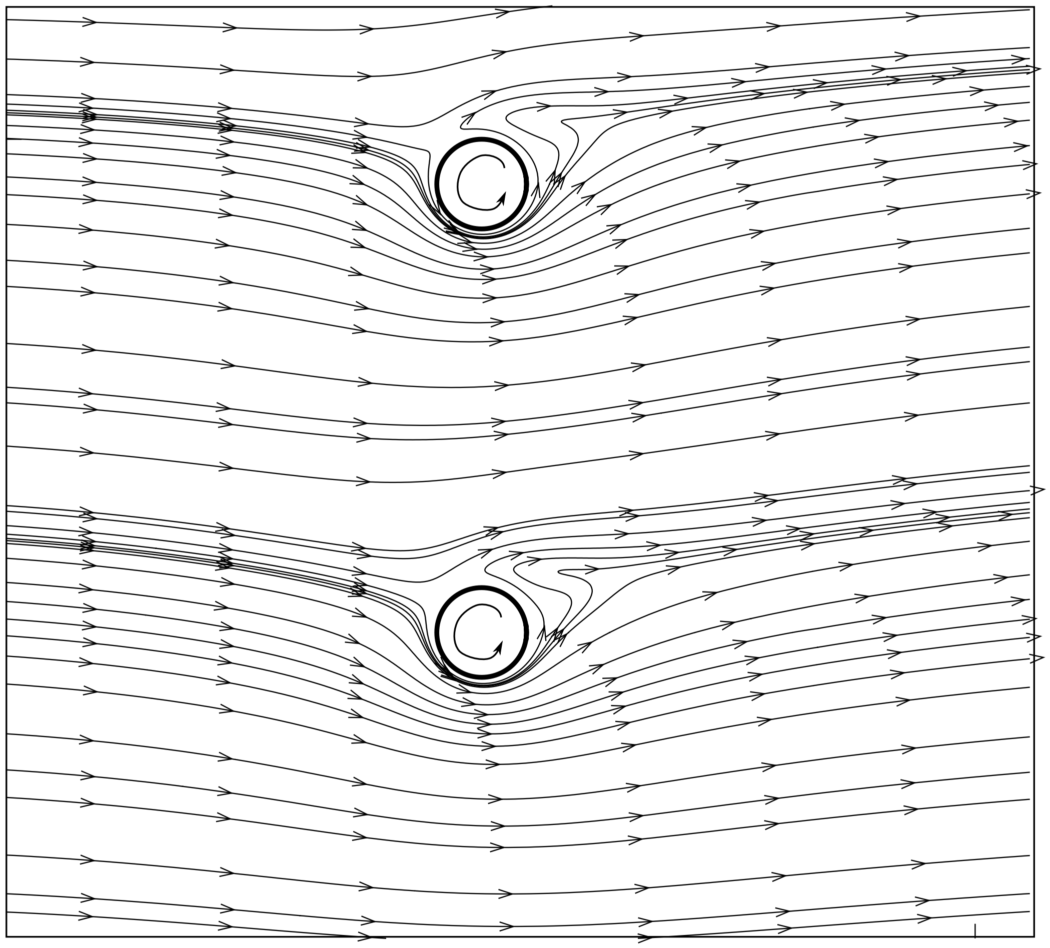


Figure 6.26: Flow streamlines around two rotating cylinders. .

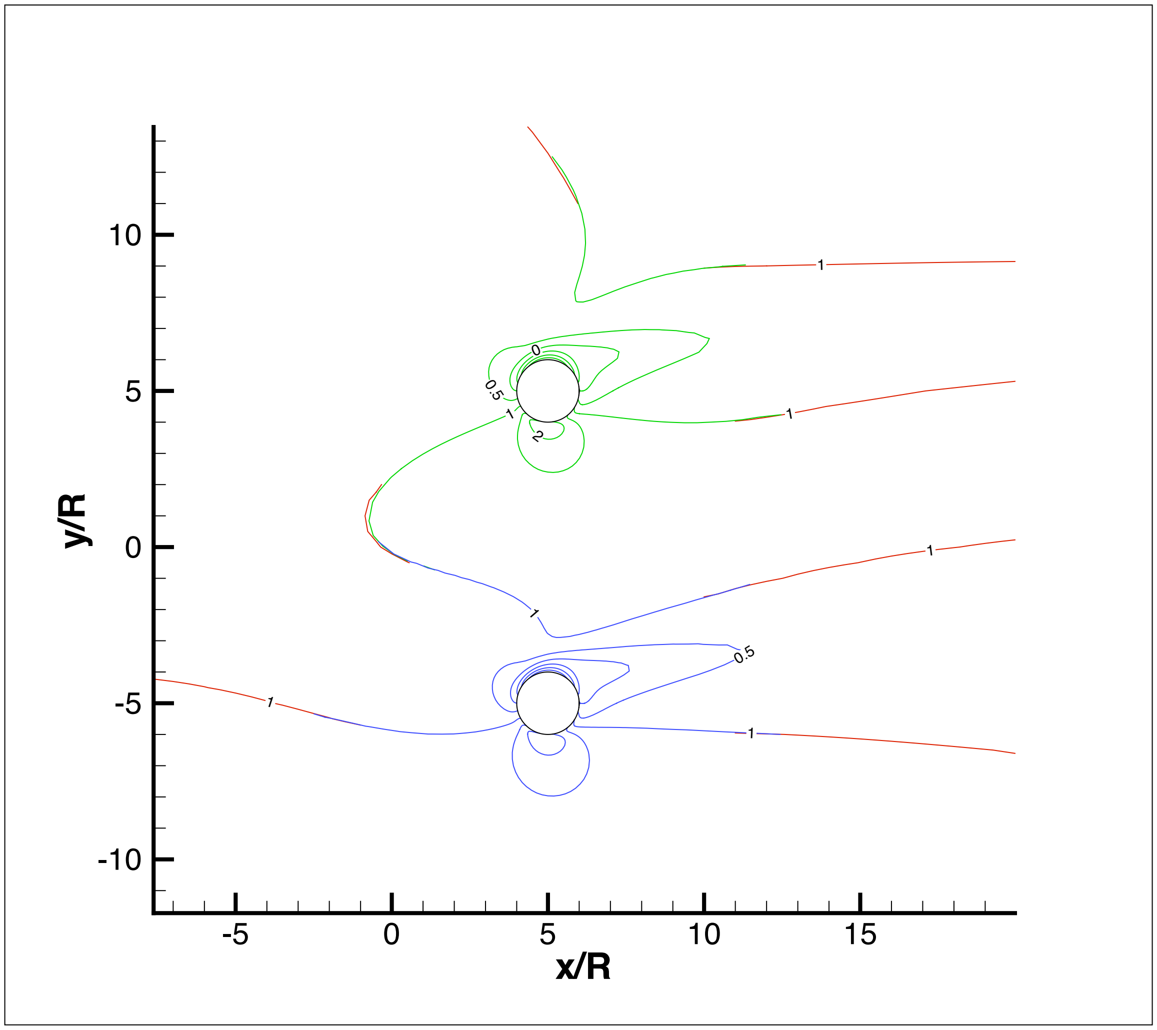


Figure 6.27: Contours x-velocity, normalised by inlet streamwise velocity

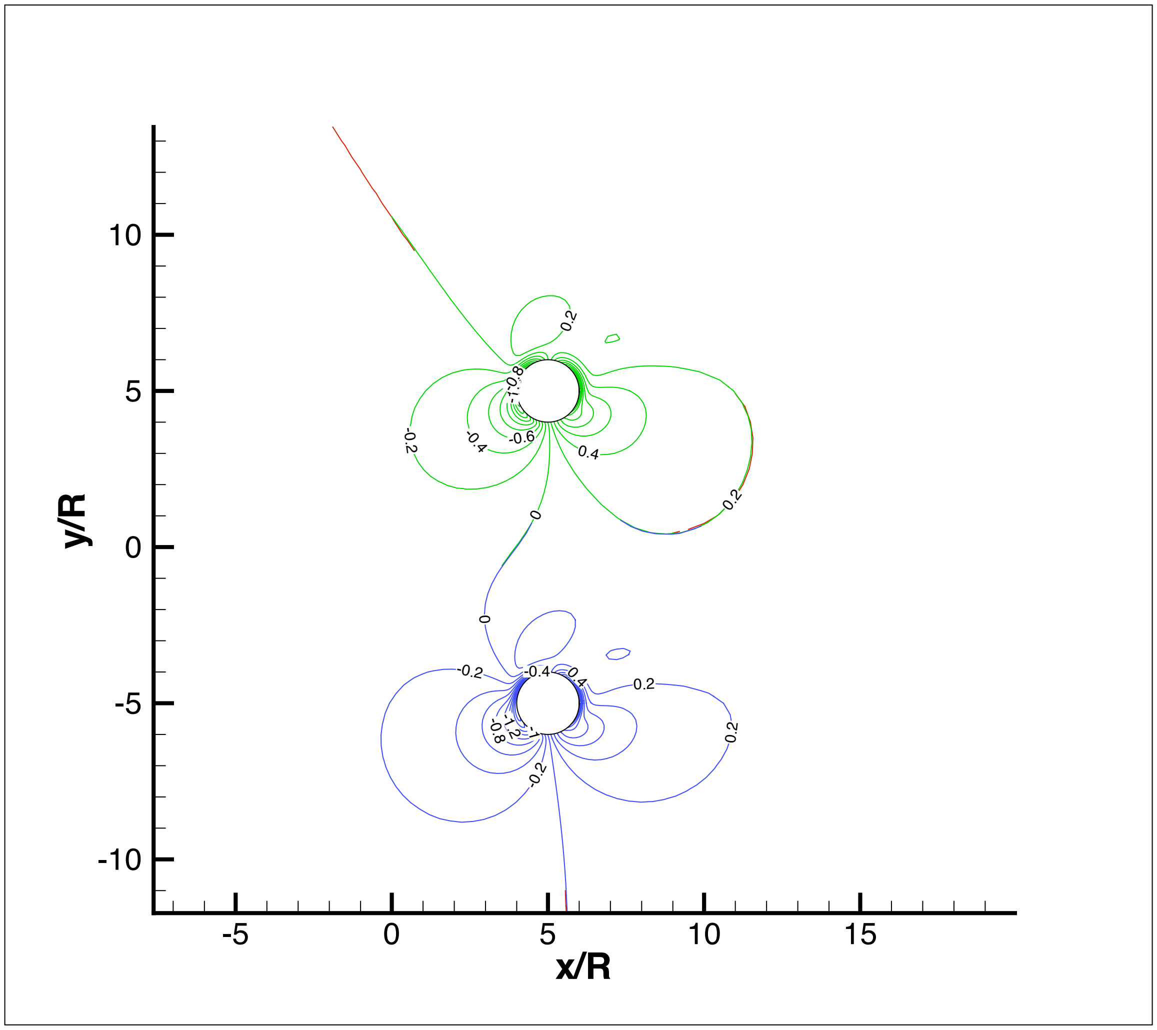


Figure 6.28: Contours of y-velocity, normalised by inlet streamwise velocity

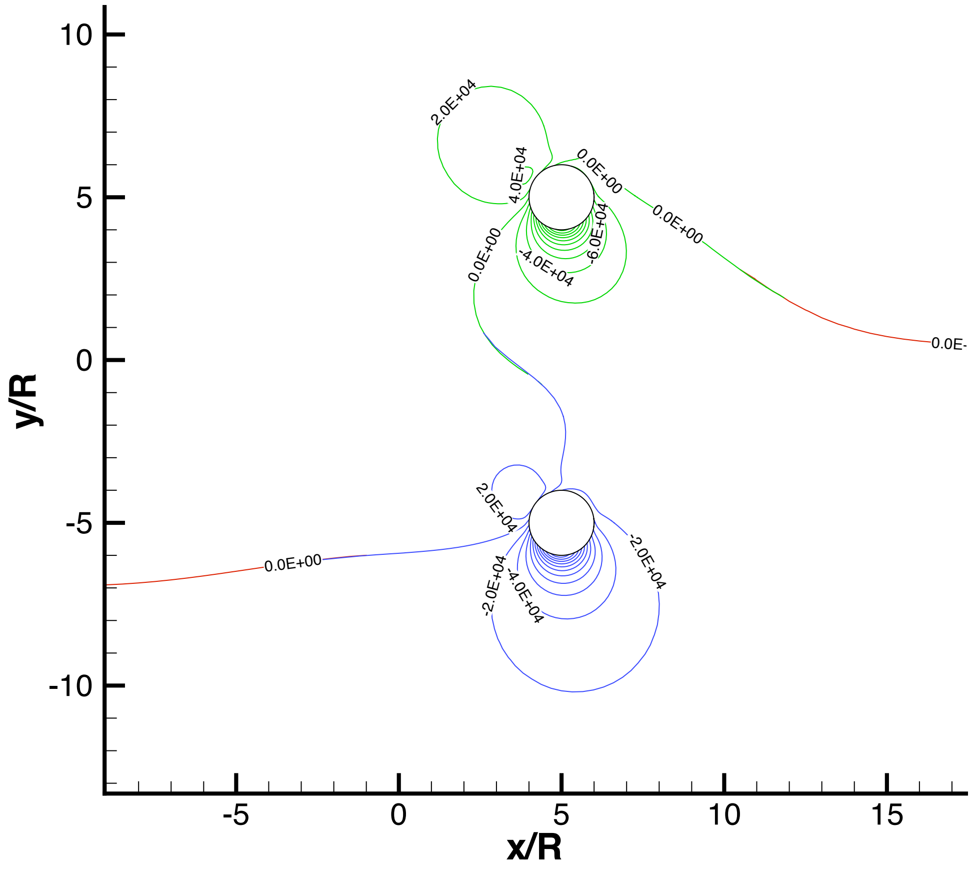


Figure 6.29: Contours of pressure.

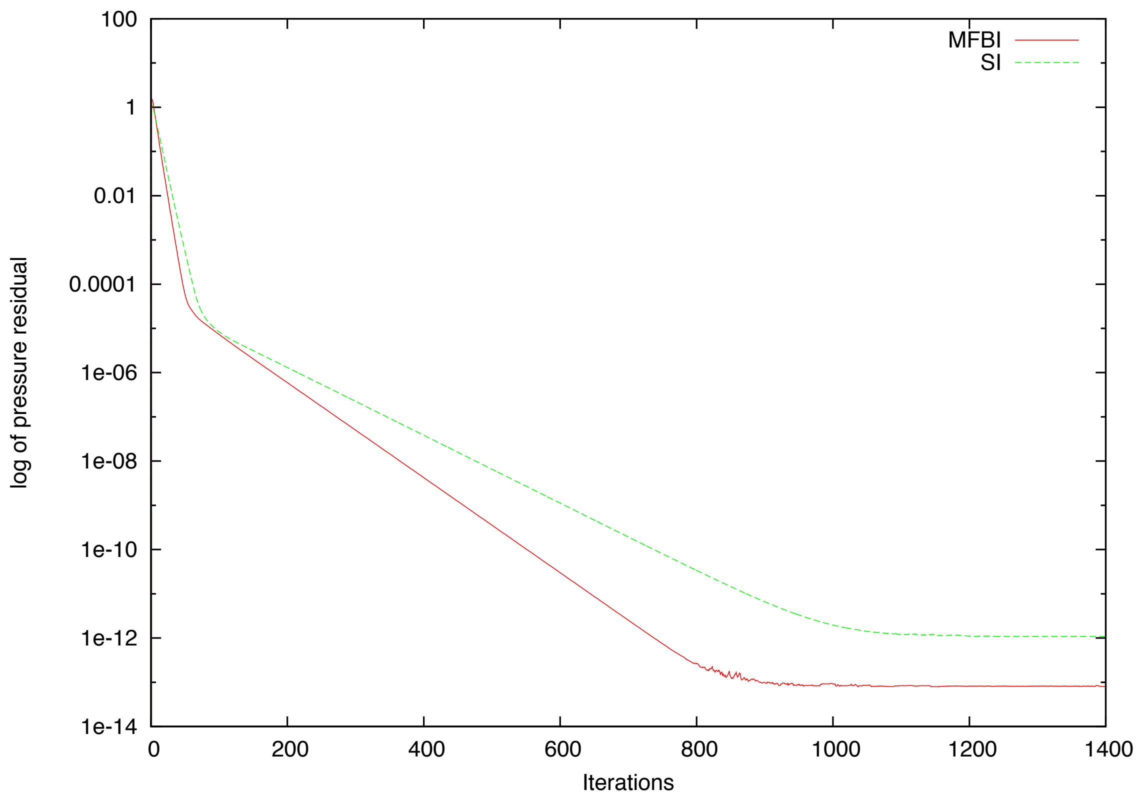


Figure 6.30: Normalised residuals of the pressure correction equation.

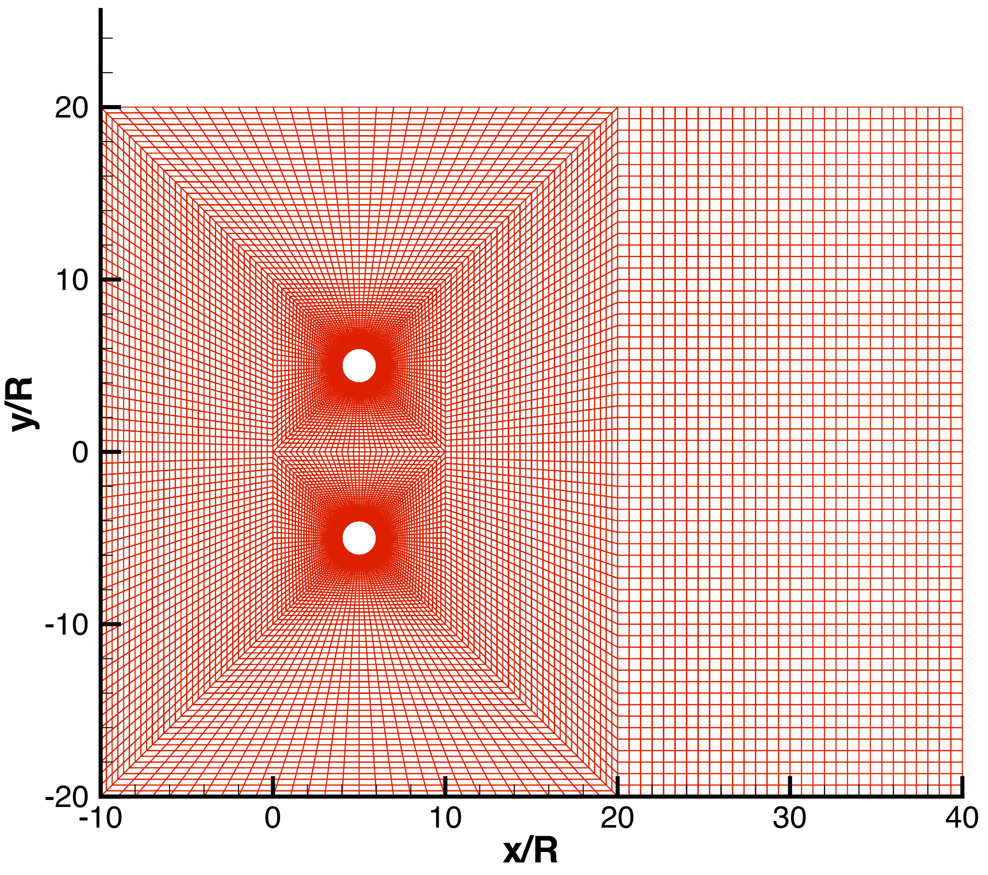


Figure 6.31: Block-structured grid used for comparative purposes.

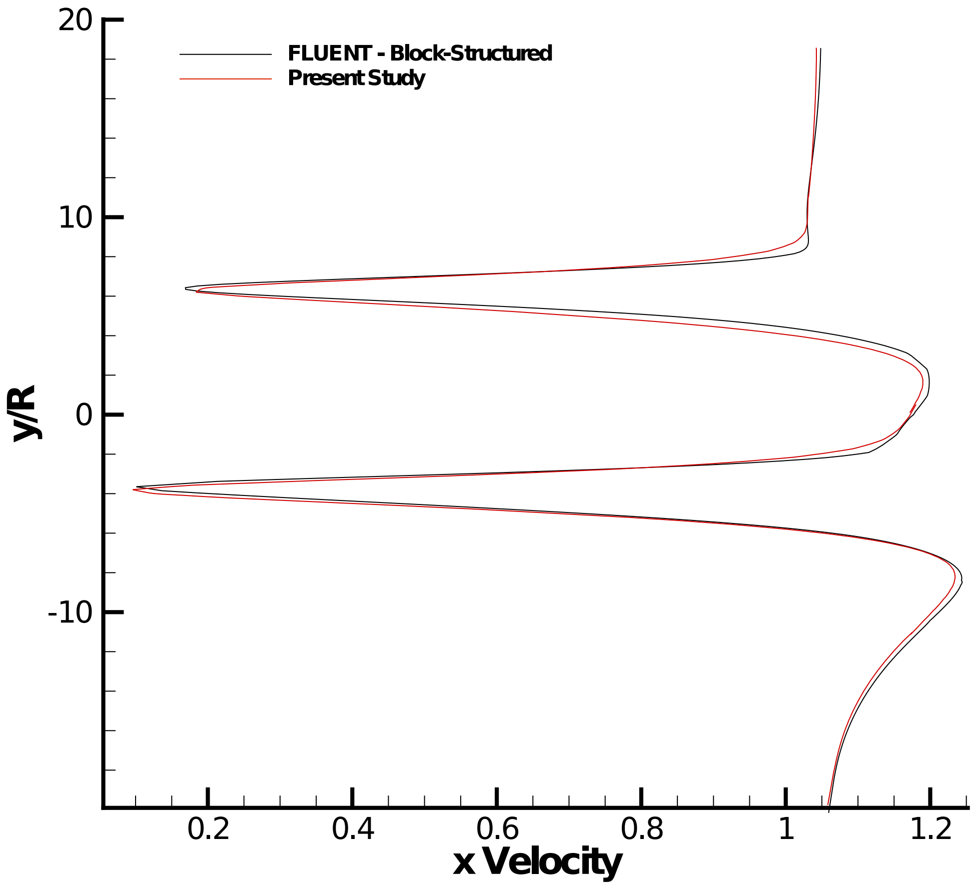


Figure 6.32: Comparisons with FLUENT and the present code. x-velocity in the wake, $3R$ from rotation axis.

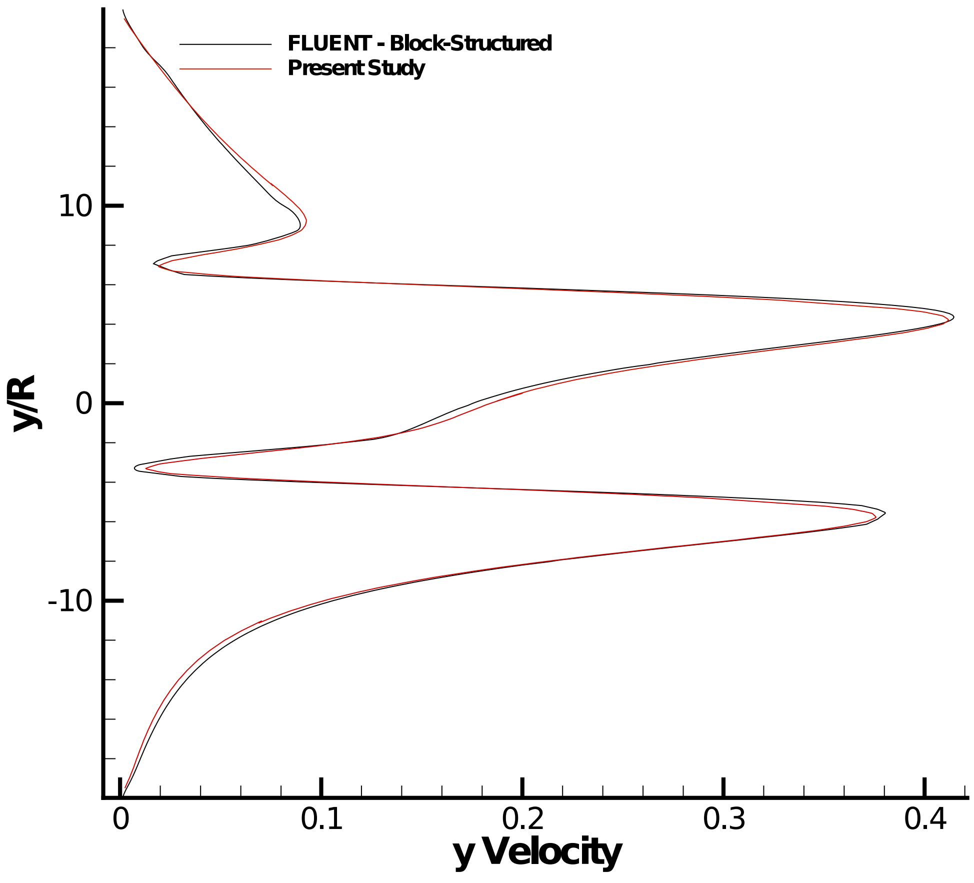


Figure 6.33: Comparisons with FLUENT and the present code. y-velocity in the wake, $3R$ from rotation axis.

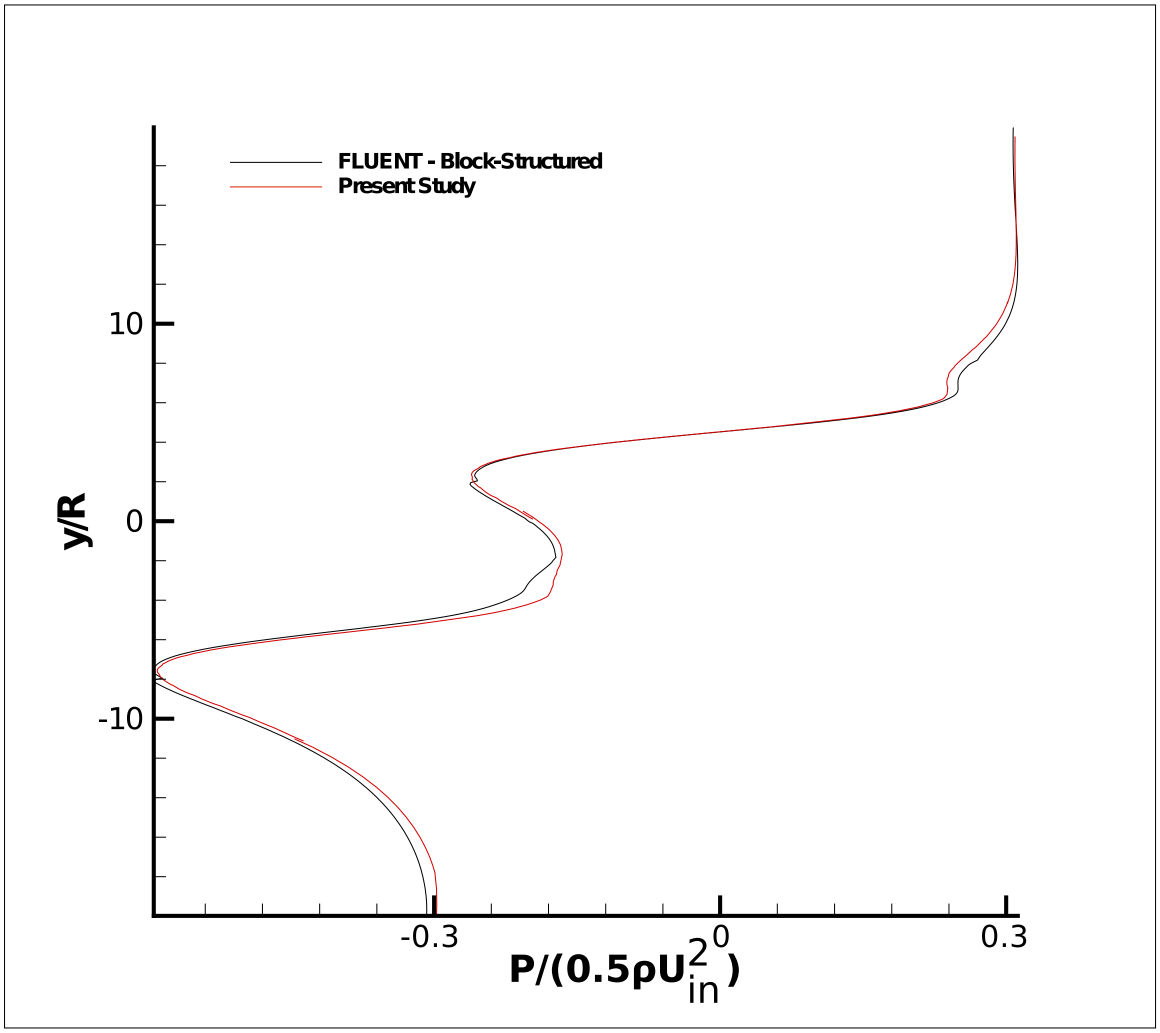


Figure 6.34: Comparisons with FLUENT and the present code. Pressure in the wake, $3R$ from rotation axis.

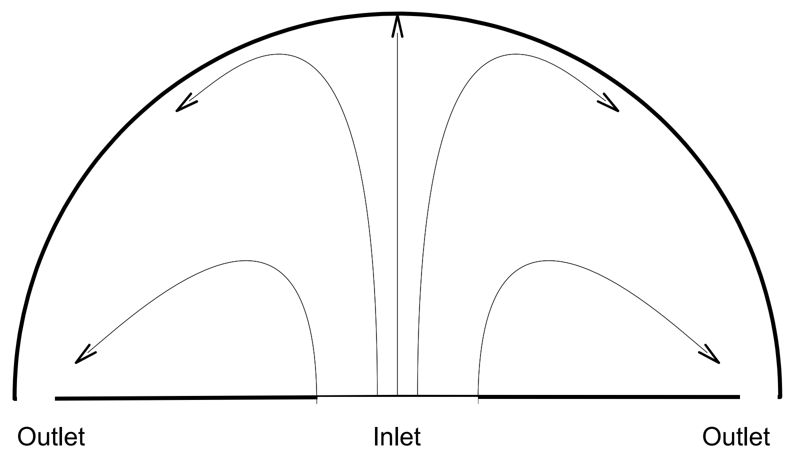


Figure 6.35: Schematic of the impingement flow from a slot onto a concave surface.

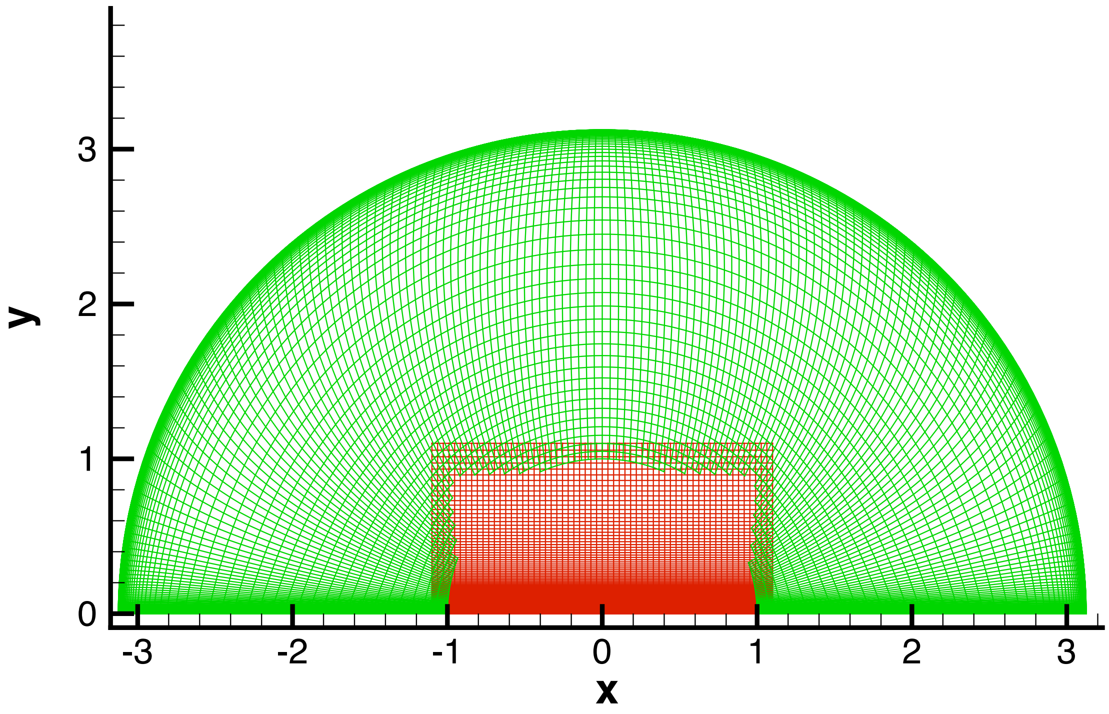


Figure 6.36: Grids used in the calculation involving fluid impinging onto a concave surface.

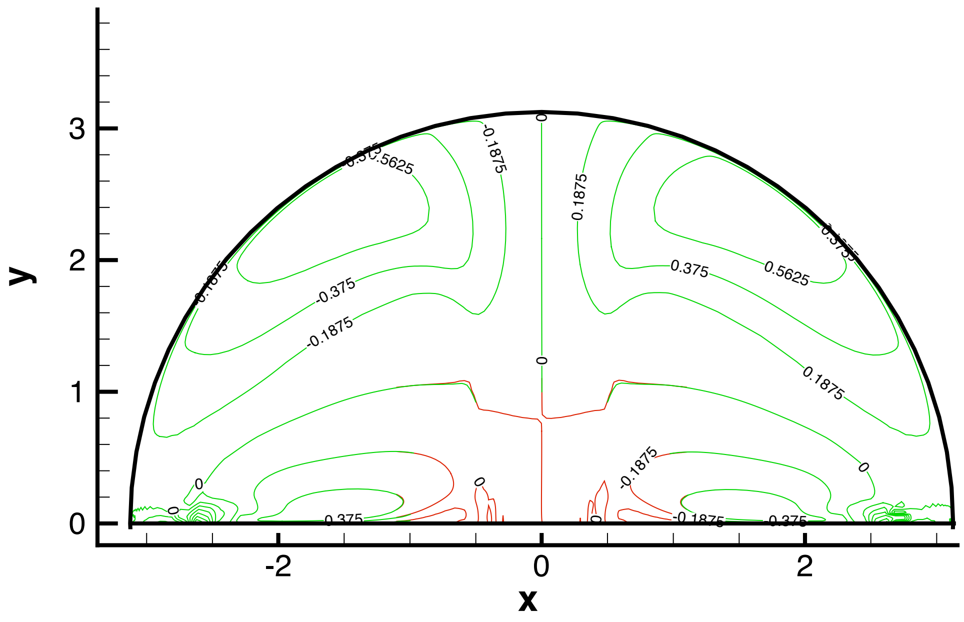


Figure 6.37: Contours of U-velocity, normalised by inlet velocity. $Re = 5,000$.

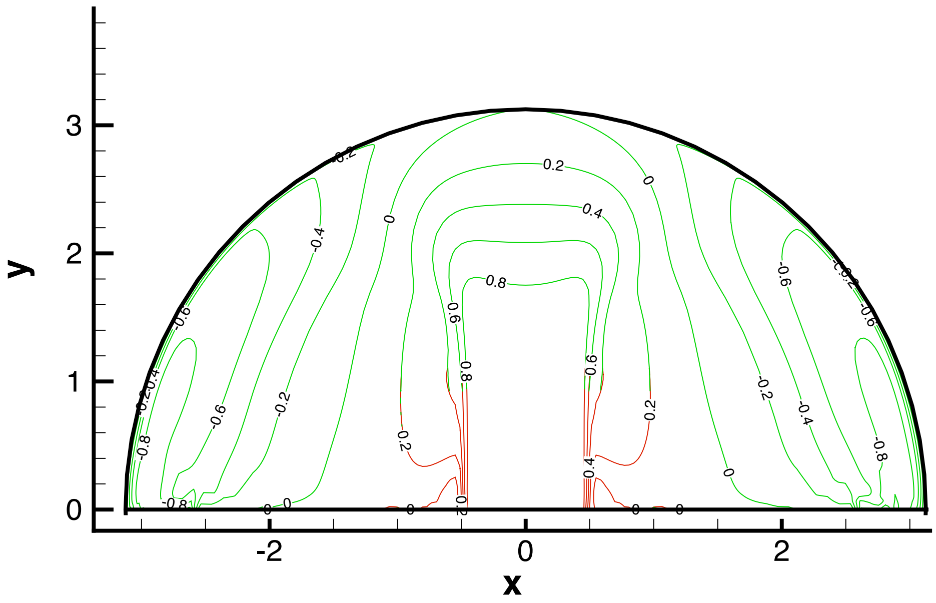


Figure 6.38: Contours of V-velocity, normalised by inlet velocity. $Re = 5,000$.

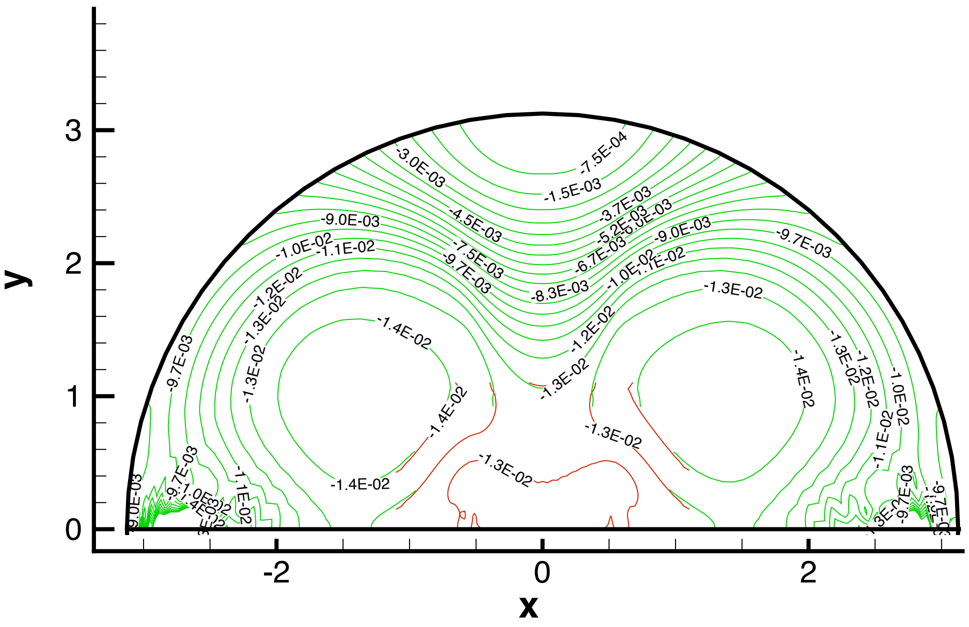


Figure 6.39: Contours of pressure. $Re = 5,000$.

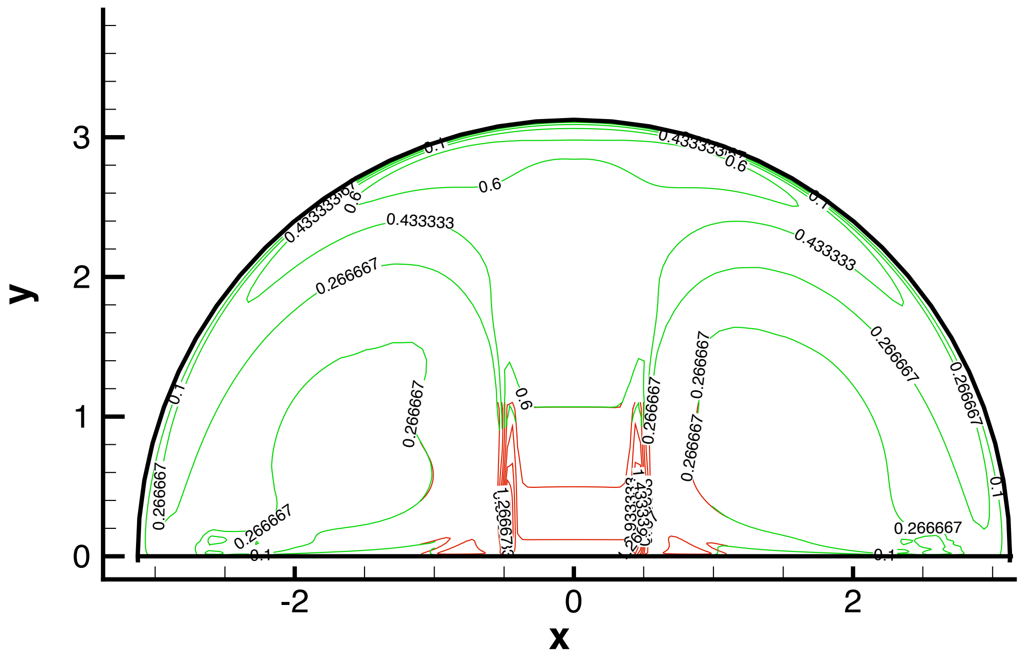


Figure 6.40: Contours of k, normalised by inlet k. $Re = 5,000$.

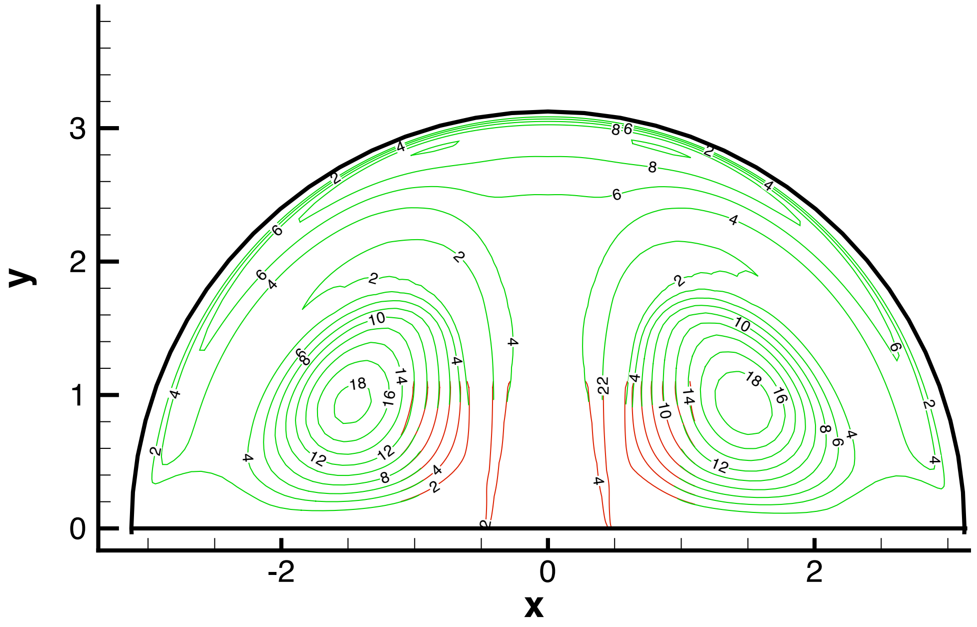


Figure 6.41: Contours of μ_t/μ. $Re = 5,000$.

Chapter 7

Backward Facing Step Flow Calculation

The Figures referred to within this chapter can be found from Page 226 onwards.

7.1 Introduction

The flow over a backward facing step is a commonly used test case for code validation, owing largely to the high complexity of the resulting flow field despite the relative simplicity of the geometry. The case presents a challenging test for the flow solver and, in particular, the turbulence model employed. Furthermore, there is a wealth of experimental data for backward facing step flows at a wide range of different Reynolds numbers, and at various expansion ratios, which aids comparison between computation and experiment.

Figure 7.1 illustrates the geometry and primary flow features that can be found in backward facing step flows. The flow enters the domain with boundary layers forming on the upper and lower walls. At the step, the flow separates from the lower wall and a shear layer is formed in place of the lower boundary layer. Further downstream flow reattachment will occur, at which point the shear layer bifurcates into a new boundary layer, and a recirculation region downstream and upstream of the reattachment point, respectively. At some considerable distance downstream of the reattachment point, a fully developed flow profile will be attained, at which point the flow will be indistinguishable from simple channel flow.

In the region behind the step the flow is particularly complex, with primary and secondary recirculation vortices present. The recirculating flow is driven by the shear forces in the shear layer, and the subsequent fluid entrainment that results. At low Reynolds number the entrainment is predominantly due to viscous shear, however at higher Reynolds numbers the turbulent diffusion of momentum becomes increasingly significant.

An increase in turbulence levels throughout the shear layer will act to augment the fluid entrainment into the shear layer, thereby enhancing momentum transfer, resulting in a faster recirculation flow, with a consequential retardation of the core fluid flow outside of the recirculation region. A shortening of the reattachment length (i.e. the distance from the base of the step to the reattachment point) will therefore result. Accurate prediction of the turbulence levels through the shear layer is therefore particularly important for the accurate prediction of the reattachment length.

As well as the effects of the primary recirculating flow, the secondary recirculation region can have an indirect effect on the predicted reattachment length. This occurs via the momentum transfer between primary and secondary vortices, and the subsequent momentum transfer between the core flow and the primary vortex. Sufficient grid resolution to capture these secondary flow features is therefore important.

7.2 Boundary Conditions and Geometry

The geometry considered here is equivalent to that considered experimentally by Chun and Sung, [60]. The inlet was placed at five step-heights upstream of the step, and the inlet-channel width was equal to two step-heights (yielding an expansion ratio of 1.5). Inlet boundary conditions for all primitive variables were set to those of fully developed channel flow profiles, at a Reynolds number of $33,000$ based on the step height and bulk inlet velocity, obtained via a separate channel flow simulation. The outlet was placed sufficiently downstream of the step to ensure that zero-gradient outlet conditions could be applied without significantly affecting the upstream profiles of any flow variables. A distance of forty step-heights downstream of the step was found to be sufficient. The bulk pressure correction algorithm, as described in Section 3.6.1, has been employed to ensure a zero net mass flux through the domain boundaries. Interpolation boundary conditions were handled via the semi-conservative interpolation method, described in Section 5.4.2.

7.3 Computational Model

The low-Reynolds number $k-\epsilon$ model of Launder and Sharma, [2], has been used to model the effects of turbulence, with integration performed right up to the wall. The third order QUICK scheme has been used to approximate the convective fluxes in the momentum equations, while the UMIST scheme was employed for the approximation of the convective fluxes of the turbulence equations. Pressure-velocity coupling has been achieved via the use of the SIMPLE algorithm.

It is well understood that for separated flows, such as that encountered behind a back-step, the effective viscosity class of turbulence models typically perform rather poorly - and particularly the linear form of such models such as that employed here. The root of the problem lies in the fact that linear eddy viscosity models interact with the mean flow equations via the prescription of a single scalar variable (i.e. the turbulent viscosity μ_t), which acts equally in all directions. In reality, however, turbulence is anisotropic and hence cannot be accurately depicted through the use of a single scalar for arbitrary flow conditions. In separated flows, where a prominent flow direction cannot be identified, all components of the Reynolds stress tensor are significant and hence the use of a single scalar to depict turbulence is particularly unsatisfactory. As a result, eddy-viscosity models have a tendency to predict excessive levels of turbulence generation in the recirculation region, and hence generally underestimate the reattachment length in a backward facing step flow configuration due to the enhanced fluid entrainment into the shear layer.

In order to compensate for some of the inadequacies of the $k-\epsilon$ model in predicting separated flows, a length scale correction term has been employed. The form of this correction is that proposed by Iacovides and Raisee, [51]. In [51], a differential form of length scale correction which is *independent of the wall distance* is presented. This is in contrast to a number of alternative length scale correction proposals, such as the Yap correction, [50] (upon which this differential form is based), which requires a normal distance to the nearest wall to be prescribed. Avoiding the need to prescribe a wall distance is particularly attractive for complex three-dimensional geometries where such a distance may not be defined. Admittedly, for the relatively simple geometry of a backward facing step, the prescription of a normal wall distance is not particularly challenging; nonetheless, the case provides a useful test of the correction term. The length scale correction is added to the ϵ source and is active in non-equilibrium flows. The term acts to reduce the departure of the predicted turbulent length scale from local equilibrium levels, and is essentially an *ad hoc* fix that often improves predictions for such flows. In Section 4.4, the details of this differential length scale correction term have been outlined.

7.4 Computational Grid

Separate grids have been used for the portions of the domain upstream and downstream of the step, via the use of an overset arrangement. Figure 7.2 shows a typical grid arrangement used. For clarity, the coarsest grids considered have been presented with every second grid-line omitted.

One advantage of using an overset grid arrangement over a block-structured grid is that fewer nodes are typically required to achieve grid independence. For example, in order to resolve a wall boundary layer, a large number of nodes are typically required within a short distance normal to the wall. For the lower wall of the inlet channel, this near-wall spacing would typically continue into the interior of the domain when using a block-structured arrangement, as illustrated in Figure 7.3. The resulting high aspect ratio cells within the core of the domain are undesirable and can adversely affect convergence rates. Furthermore, numerical stability may require the use of a reasonably fine grid in other coordinate directions (in order to reduce the cell aspect ratio, thereby increasing numerical stability), leading to an increased number of nodes and higher computational costs. Through the use of the overset method however, this problem is reduced. One can simply "fan out" the near wall cells over a distance of $1-2$ step heights say, as demonstrated in Figure 7.2. High aspect ratio cells within the core of the domain are therefore avoided.

Grids comprising of approximately $16,500$, $33,000$ and $66,000$ nodes have been used. All grids had a similar nodal distribution to that shown in Figure 7.2, with each grid refinement being obtained by increasing the number of nodes along each edge of the domain by a factor of $\sqrt{2}$. For the $33,000$ node grid, 200×165 nodes were used in the streamwise and transverse directions respectively. The centroid of all near-wall nodes was placed at a distance from the wall of $1.14 \times 10^{-4}H$, $1 \times 10^{-4}H$, or $7.07 \times 10^{-5}H$ for coarse, medium and fine grids respectively. The resulting near-wall y^+ values for all near-wall nodes was well below unity for all cases (a mean value of $y^+ \approx 0.4$ was found for the medium grid resolution, for example).

The ratio of the wall-normal cell dimension between successive cells in the wall-normal direction was set equal to 1.2. This was chosen in order to ensure that there were sufficient nodes normal to the wall to fully resolve all boundary layers, while also avoiding excessive resolution. In order to test that sufficient wall-normal grid resolution had been used, all cells where $y^+ \leq 100$ were plotted (with any other cells being blanked), and in all cases it was ensured that there were at least 14, 20 or 28 unblanked nodes present in the wall-normal direction for the coarse, medium and fine grids respectively.

To further test for sufficient near-wall grid resolution, Figure 7.4 shows a typical profile of the dimensionless velocity U^+, versus the dimensionless wall distance y^+ (for the medium resolution grid), with the locations of the cell centroids used to construct the profile indicated. It can be seen from the figure that in this case, ten nodes cover the laminar sub-layer (defined as $y^+ < 5$), while a further nine nodes are used to resolve the buffer region (defined as $5 \leq y^+ < 30$). In total, twenty-five nodes cover the region $y^+ < 100$. It has been found that this near-wall nodal distribution is also fairly typical of other locations around the domain. This is certainly expected to be sufficient, if not excessive.

Using the three grids defined above, a grid sensitivity study has been performed. In order to quantify grid dependence, plots of the wall shear-stress along the lower wall, downstream of the step, have been taken. Figure 7.5 shows these plots. From Figure 7.5 it can be seen that there is a relatively large difference in the predicted wall shear stress between the coarse and medium grids. However, the difference between the medium and fine resolution grids is small, and hence the medium resolution solution can be considered a grid independent solution. All subsequent results have therefore been obtained using the medium grid resolution ($33,000$ nodes).

It is interesting to note that for the coarsest grid resolution used ($16,500$ nodes), a converged solution could not be achieved when using standard linear interpolation at the interpolation boundary cells. Analysis has revealed that the issue is due to the fact that the bulk correction algorithm matches the outlet mass flux with that of the inlet. However, since the outlet and inlet are on different grids, there is no guarantee that a zero net mass flux would be achieved on each sub-grid due to the non-conservative nature of the linear interpolation. As the grid resolution is increased, interpolation errors are reduced and hence the problem is easily avoided simply by refining the grid. Alternatively, by using a (semi-)conservative interpolation algorithm, the issue can be avoided, even for coarse grids. When using the MFBI interpolation algorithm, convergence was achieved at all the grid resolutions considered.

7.5 Results

7.5.1 Reattachment Point

The point of flow reattachment is characterised by a zero wall shear stress. This follows from the fact that at the reattachment point, the velocity component parallel to the wall, at an infinitesimal distance from the wall, is equal to zero. Using this definition, it can be seen from Figure 7.5 that a reattachment point of around $x/H = 8.3$ is predicted (at grid independence). However, in the experimental work of Chun and Sung, [60], the reattachment point is not determined in this way, and is instead defined as the point at which the x-component of the flow velocity *in the vicinity of the wall* (at $y/H = -0.98$), is equal to zero. Using this approximation, a reattachment point of $x/H = 7.3$ is predicted by the present study. A large elongation of the simulated recirculation region very close to the wall has been observed, which explains the relatively large difference between the zero wall shear stress point, and the approximated reattachment point via the method of [60].

The reattachment point, approximated in a way consistent with [60] (i.e. at $x/H = 7.3$), is in reasonable agreement with the experimental findings, in which an approximated reattachment point of $x/H = 7.8$ was found. Given the relative simplicity of the turbulence model, this is an encouraging agreement.

Note that in the absence of the length scale correction term, it has been found the standard Launder-Sharma model predicts a reattachment length of just $6.5H$, as can be seen from Figure 7.6, where plots of the wall shear stress along the lower wall are presented, both with and without the length scale correction term activated. It can be seen from Figure 7.6 that the predicted (negative) wall shear stress in the recirculation region is excessive in the absence of the length scale correction term. This is due to the turbulence generation in the recirculation zone being far from equilibrium, resulting in very high levels of turbulent transport close to the lower wall. This acts to reduce the thickness of the viscous near-wall layer, thereby increasing the wall shear stress. With the length scale correction term active however, the near-wall turbulence is reduced, resulting in a thicker viscously prominent near-wall layer, and hence a lower wall shear stress. The stark difference the length scale correction term makes highlights the importance of its use for separated flows.

7.5.2 Mean Velocity

Figure 7.7 shows the predicted streamlines for the back-step flow considered. It can be seen from the figure that the curvature of the dividing streamline (i.e. the streamline connecting the separation point to the reattachment point) changes sign, with the point of inflection around $1-2$ step heights upstream of the reattachment point. Downstream of the point of inflection, the dividing streamline runs near-parallel to the wall (and close to the wall) over a considerable distance before eventual reattachment (see the insert of Figure 7.7). This is a result of the flow conforming to the contour of the lower wall before impingement, and is due to the adverse pressure gradient in the wall-normal direction near to the impingement point. It is this streamline curvature (and the entrainment of the surrounding fluid), that results in the strong elongation of the primary recirculation region close to the wall that has been observed.

It can also be seen from Figure 7.7 that a small secondary recirculation region is present in the corner where the vertical wall (i.e. the step) and the lower wall meet. The size of this secondary recirculation region is perhaps a little smaller than expected. Previous computational studies had predicted a stronger (larger) secondary vortex (see for example Momeni, [61, 62]). A small secondary recirculation region implies high levels of turbulence in the surrounding region. Since the turbulence levels in the recirculation region are particularly sensitive to the length scale correction term, alternative length scale correction treatments between previous computations and the present study may explain the differing predictions. It is not clear from [61] of the form of length scale correction that was employed (or indeed if any length scale correction was used at all) hence a direct comparison is not instructive. Unfortunately, the experimental data provided in [60] offers little insight into the structure of the secondary recirculation zone, so it is unclear if the size of the secondary recirculation zone is consistent with experiment. In any case, the use of a linear EVM to quantitatively predict secondary flow features where high turbulence anisotropy is expected is perhaps a little optimistic, so a qualitative statement simply that such features exist in the simulated flow-field is arguably sufficient.

Figure 7.8 shows profiles of the normalised streamwise velocity at several streamwise locations within the range $1 \leq x/H \leq 9$. Comparisons have been made against the experimental data provided by [60] (note that only *positive* streamwise velocity data is provided in [60], with any negative contributions omitted, hence comparisons can only be made for the upper portions of the recirculation region and above). It can be seen from Figure 7.8 that there is generally a good agreement between the computation and experiment.

Figures 7.9 and 7.10 show contours of the streamwise and transverse velocity components respectively. It can be seen from the figures that continuous solutions are obtained from one grid to the next, thereby supporting the capability of the MFBI interpolation algorithm, and of the overset method in general.

7.5.3 Turbulence results

Figure 7.11 shows profiles of the normalised streamwise turbulence intensity ($\overline{u'^2}/u_{bulk}^2$), at the same streamwise locations as those of the velocity profiles presented earlier. It can be seen from Figure 7.11 that a generally good agreement in the streamwise normal stress is achieved in the upper portion of the shear layer and above (i.e. at y/H values *above* the location of the maximum predicted stress). However, there is a significant under-prediction in the normal stress levels close to the lower wall (this is particularly apparent at the profiles obtained at $x/H = 7$ and beyond, where more near-wall experimental data is available). There is also a significant underestimation of the peak levels of normal stress, which is also particularly apparent at increasing downstream profile locations, suggesting an insufficient rate of recovery in the shear layer. Part of the reason for these discrepancies is due to the anisotropic nature of the turbulence not being captured. The streamwise normal stress is bound to be the dominant normal stress component since the dominant mean velocity gradient energising the turbulence in the shear layer is the streamwise velocity gradient in the cross-stream direction, $\partial\overline{u}/\partial y$. However, since the computational model sets all three component as equal to one another, the streamwise normal stress component tends to be underestimated, while the cross-stream component tends to be overestimated.

For completeness, profiles of the turbulent shear stress are also presented in Figure 7.12, although no experimental comparison is available in this instance.

Figures 7.13, 7.14 and 7.15 show normalised contours of turbulence kinetic energy, turbulence dissipation rate and the turbulent viscosity ratio respectively, and have been included in order to demonstrate the continuity of the solution across the overset grids for the turbulence variables. It can be seen from the figures that continuous solutions are obtained.

7.5.4 Pressure

Figure 7.16 shows a plot of the pressure coefficient, C_P, along the lower wall downstream of the step, with experimental comparison. It can be seen that a good agreement between computation and experiment is obtained downstream of the recirculation region (e.g. at $x/H > 9$), however the agreement is less satisfactory within the recirculation region. This suggests the overall shape of the recirculation region has not been accurately depicted by the computation. The pressure in the recirculation region is low so as to balance the centrifugal force associated with the streamline curvature in the recirculating flow. Further downstream, the pressure increases since the flow is decelerated relative to the inlet channel due to the expansion. The figure suggests a faster pressure recovery following the recirculation is predicted than is observed experimentally. This suggests that the strong elongation of the streamlines that was observed near to the reattachment point in Figure 7.7 may not occur in the experiments, since such an elongation effectively reduces the curvature, and hence the centrifugal force would be lower.

It has been previously noted (in Section 7.5.2), that the curvature of the dividing streamline changes sign a short distance upstream of the reattachment point. Given that the curvature of a streamline is related to the pressure gradient perpendicular to the streamline, with the radius of curvature pointing in the direction of decreasing pressure, this streamline curvature suggests a low pressure region at the core of the recirculation region, and a high pressure at the point of reattachment. This is supported by Figure 7.17, in which a contour plot of the pressure field throughout the domain is presented. Figure 7.17 also serves as a useful demonstration of the ability of the overset method to provide continuous solutions for the pressure from one grid to the next.

Note that when using standard linear interpolation, there were 'kinks' in the predicted pressure near the interpolation cells, which required a reasonably fine grid in the overlap region to be eliminated. However, when using the MFBI interpolation method, the kinks are not present even for the coarsest grids considered. The cause of the kinks is due to a small net mass imbalance caused by non-conservative interpolation, and also due to the over-prescription of boundary conditions, as previously discussed.

7.6 Summary

The flow over a backward facing step configuration has been simulated via the use of a two-equation, low-Reynolds number turbulence closure. Overset grids have been used to mesh the regions upstream and downstream of the step separately. It has been found that a good agreement between computation and experiment is obtained for the streamwise velocity profiles, despite a relatively poor agreement for the streamwise Reynolds stress predictions. Contour plots for all primitive variables (or derived variable in the case of the turbulent viscosity) have revealed good solution continuity from one grid to the next. The predicted reattachment length is also in reasonable agreement with experiment (around 6% discrepancy). Overall, the overset method has proved to be a useful tool, even for this simple geometry, since high aspect ratio cells within the core of the domain can be easily avoided, thereby reducing numerical stability issues and the resultant need for an excessively low under-relaxation factor to compensate (as is often the case for a block-structured grid arrangement).

7.7 Figures

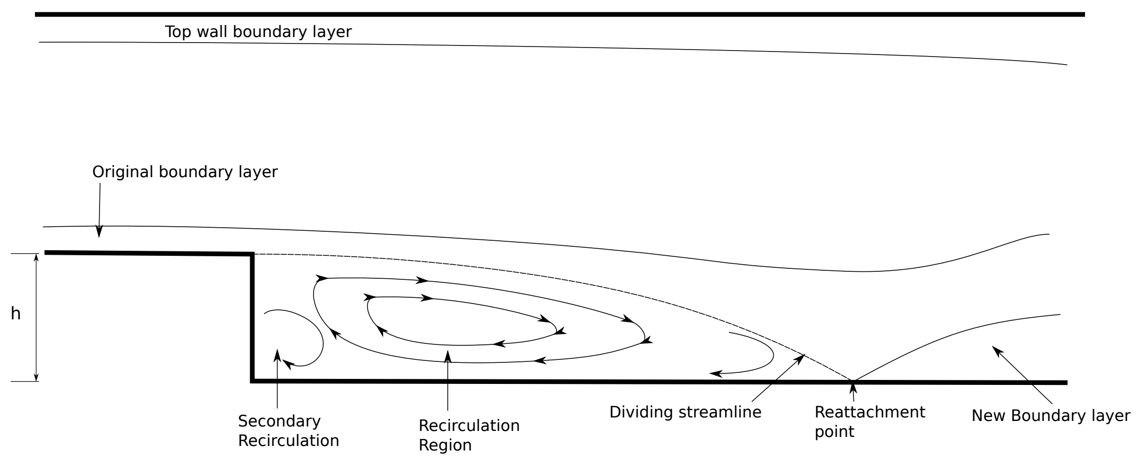
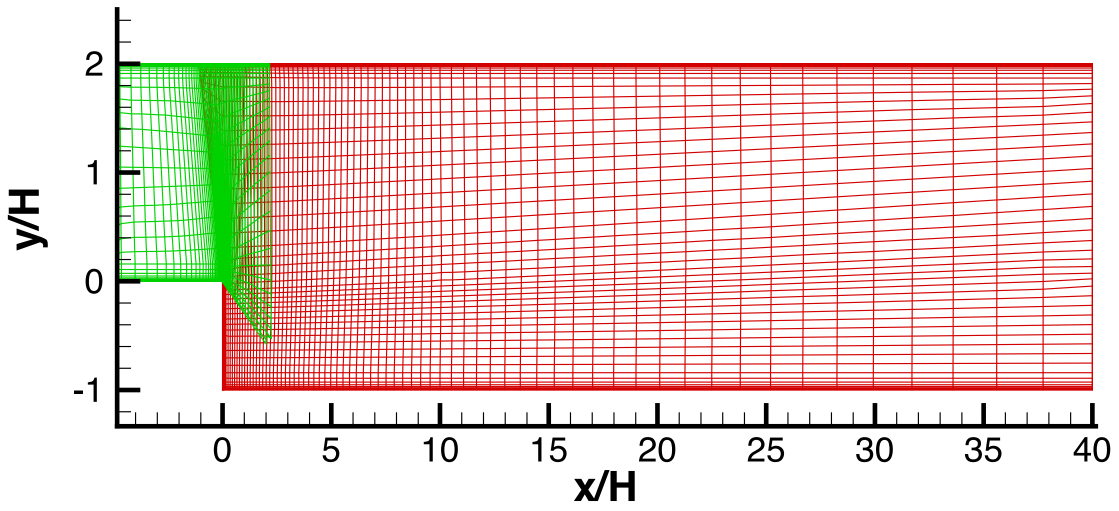
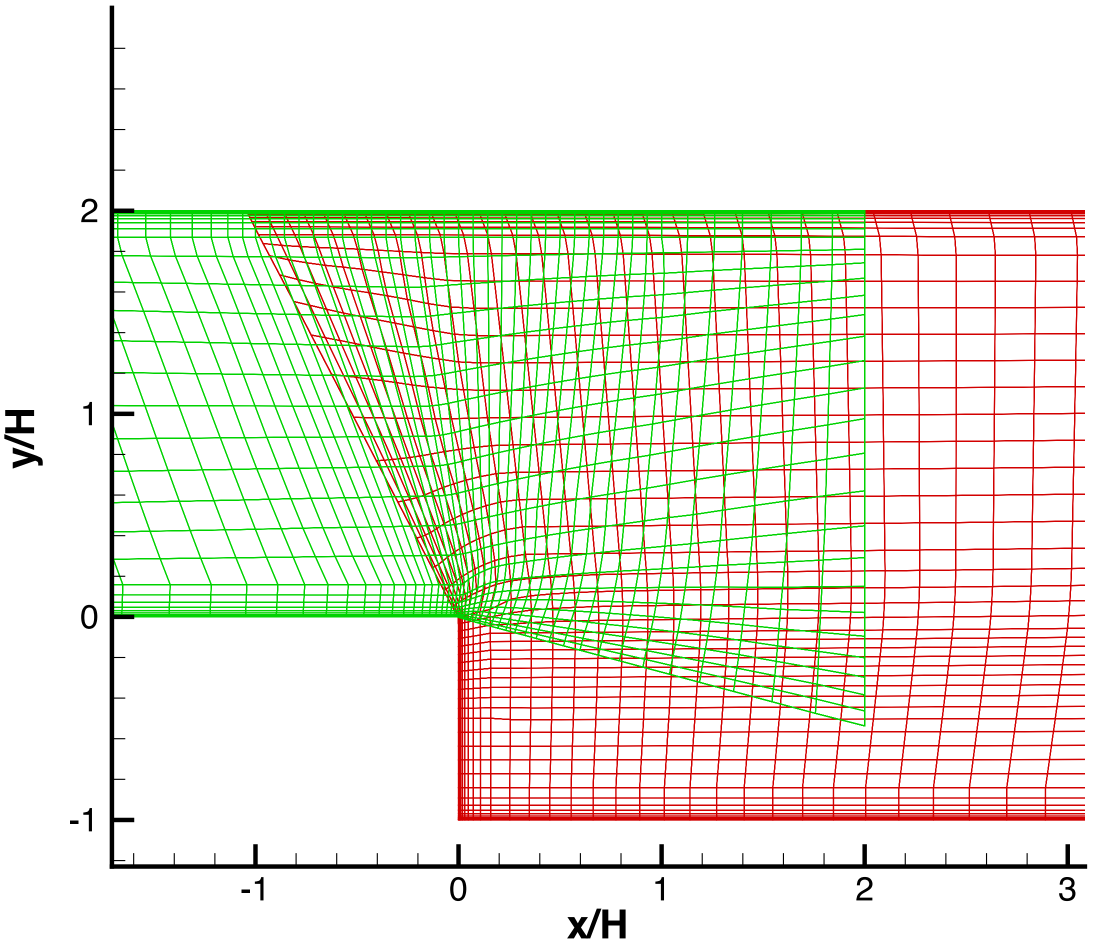


Figure 7.1: Typical flow features encountered in a backward-facing step flow.



(a) Full meshed domain (note the different scales for the x and y axes).



(b) Close up of grids around overlap region.

Figure 7.2: Coarse grids ($16,500$ nodes) used for backward facing step flow simulation. Every second grid-line omitted for clarity.

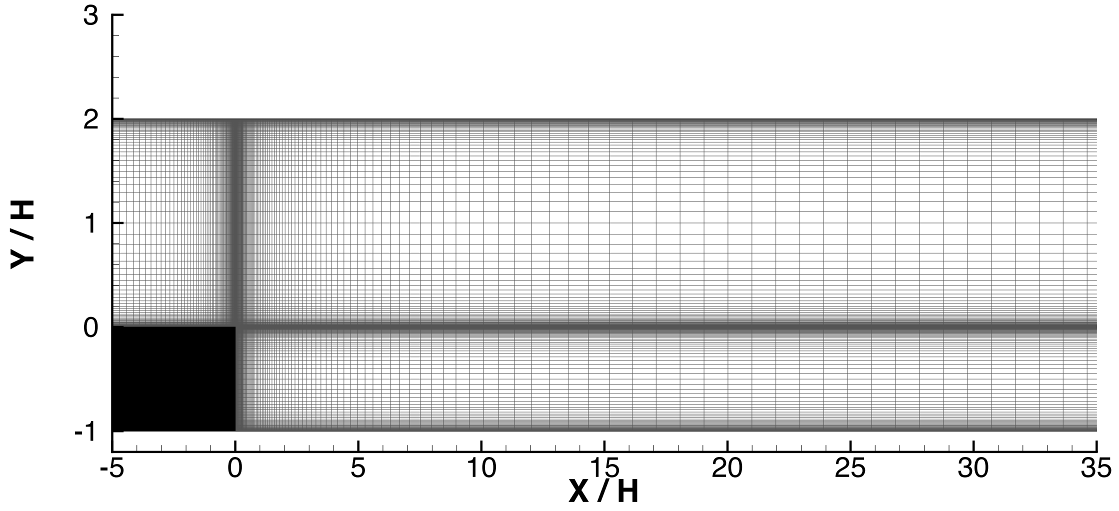


Figure 7.3: Sample Block-Structured grid arrangement.

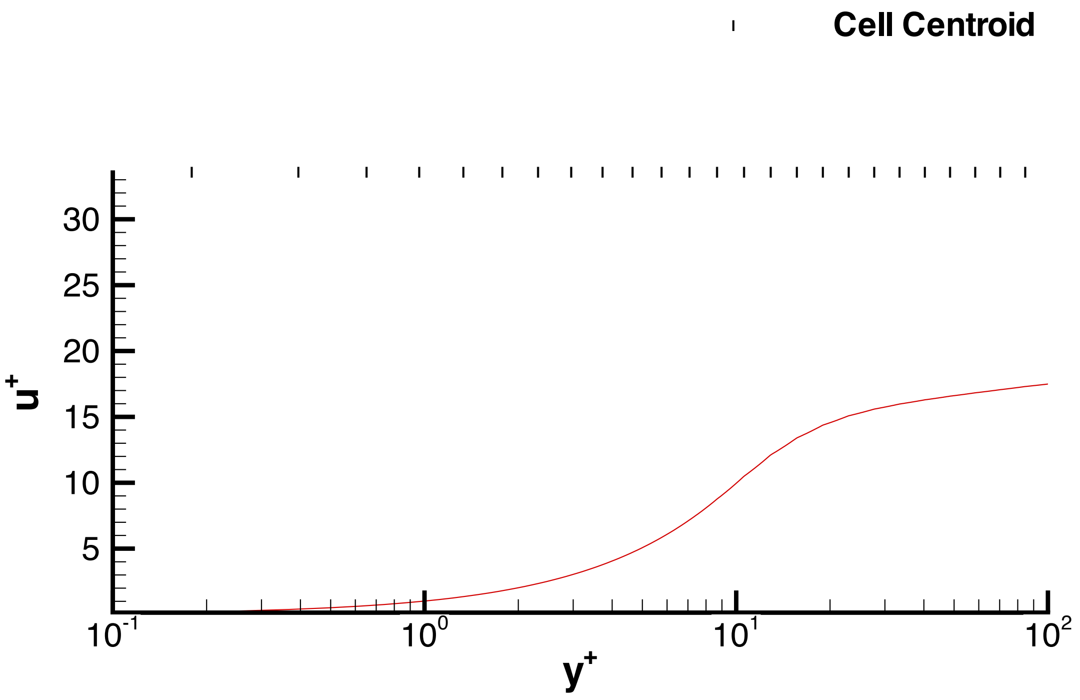


Figure 7.4: Typical near-wall grid spacing. Vertical tags at the top of the plot denote the dimensionless location of the centroid of the first twenty-five near wall cells. Profile obtained at $x/H = 20$ on lower wall, for the medium resolution grid.

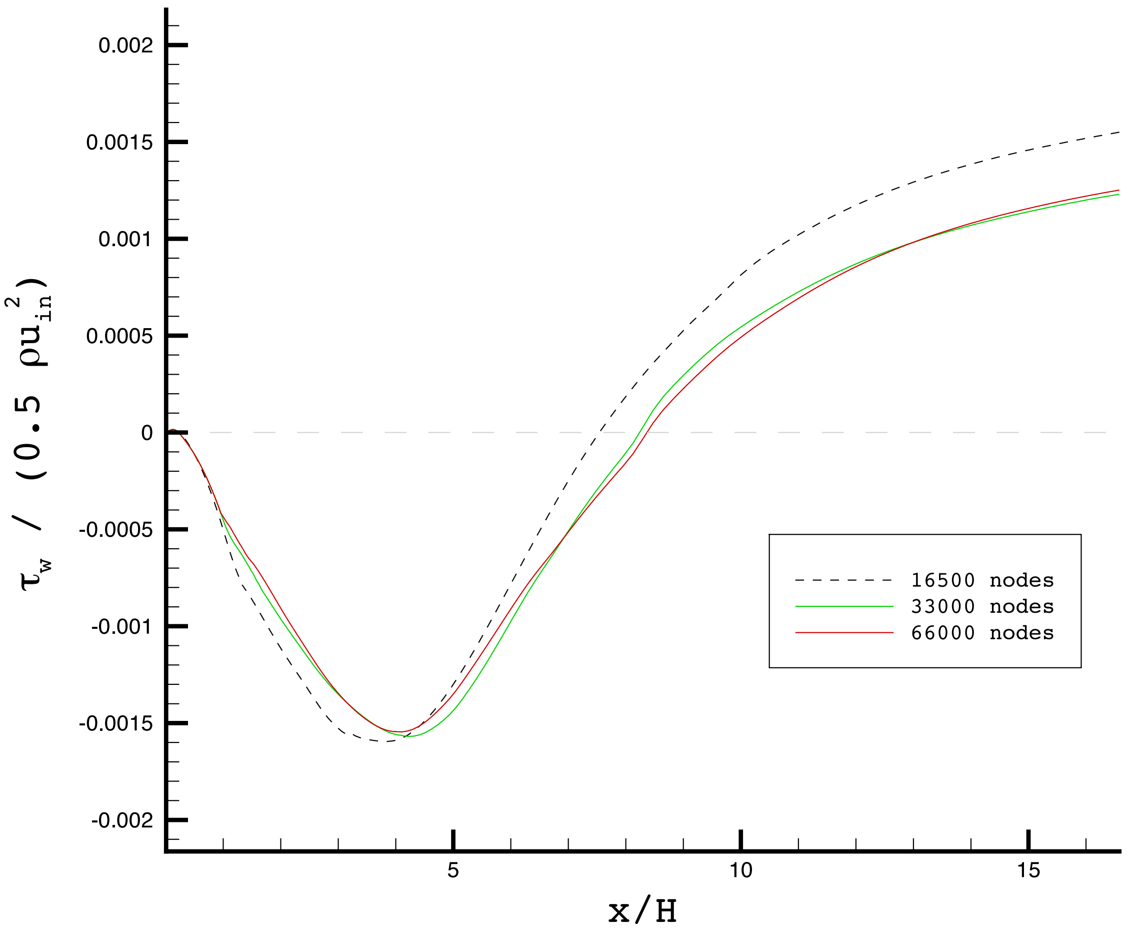


Figure 7.5: Grid sensitivity test. Plots of skin friction coefficient along the lower wall, downstream of the step, for three different grid resolutions.

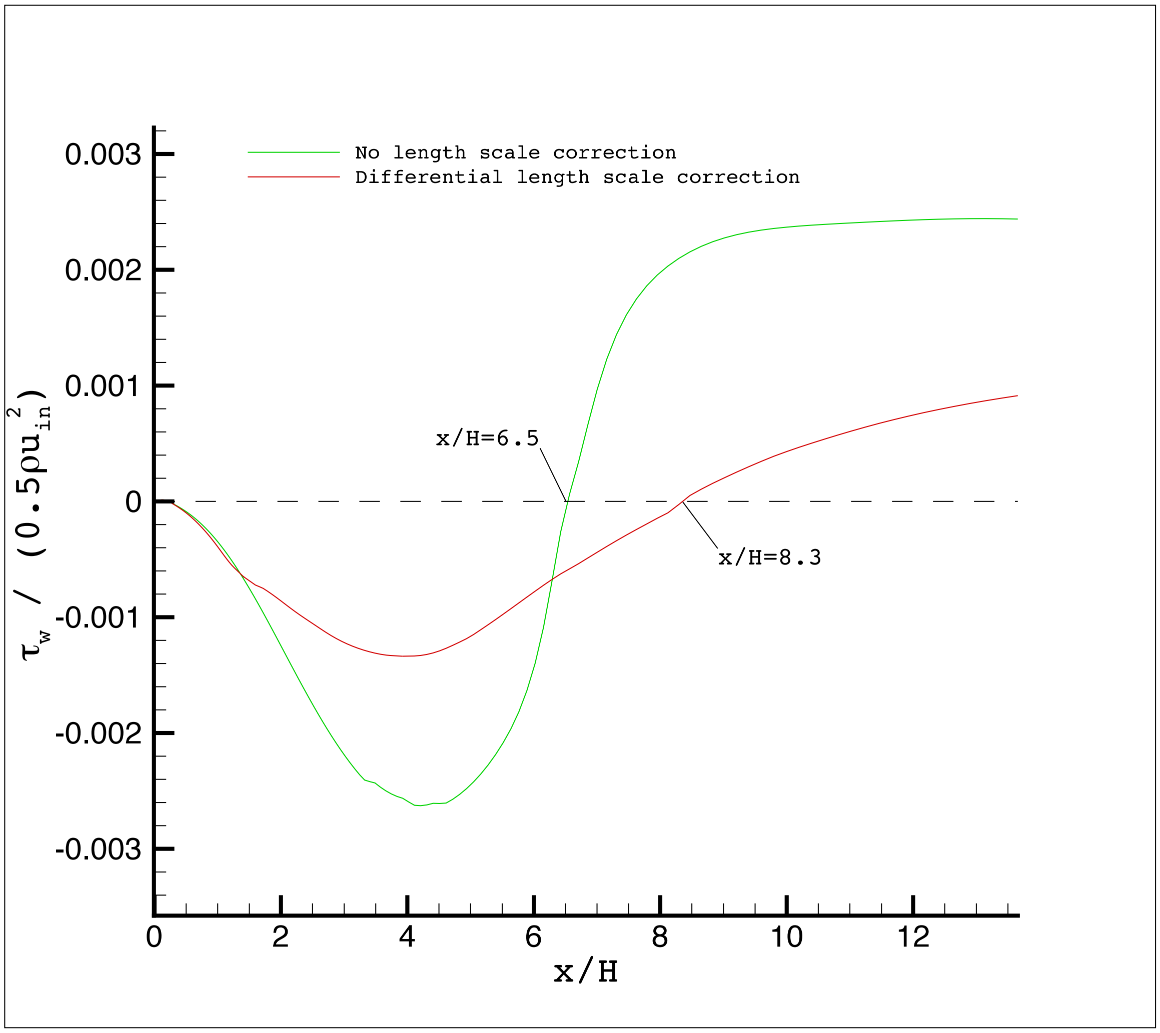


Figure 7.6: Plots of skin friction coefficient along the lower wall, downstream of the step, with and without the length scale correction term (33,000 node grid).

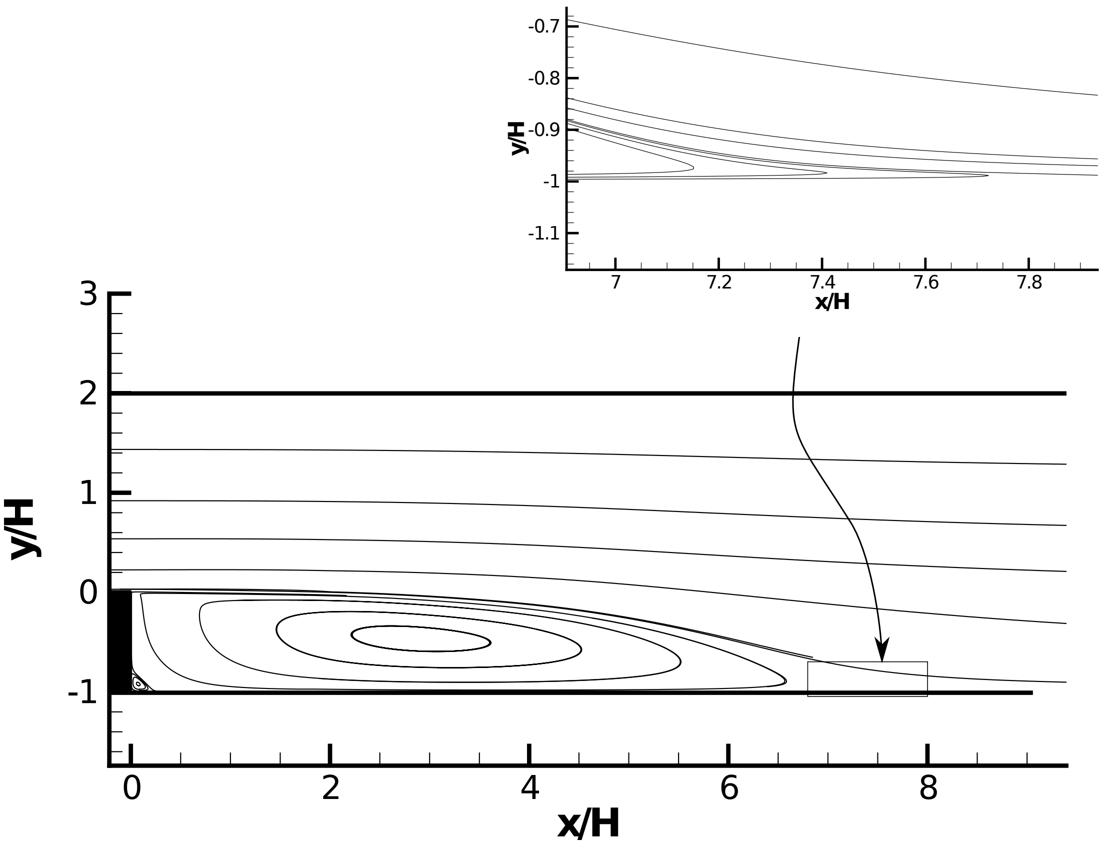


Figure 7.7: Flow streamlines, with close-up around reattachment point. $Re = 33,000$.

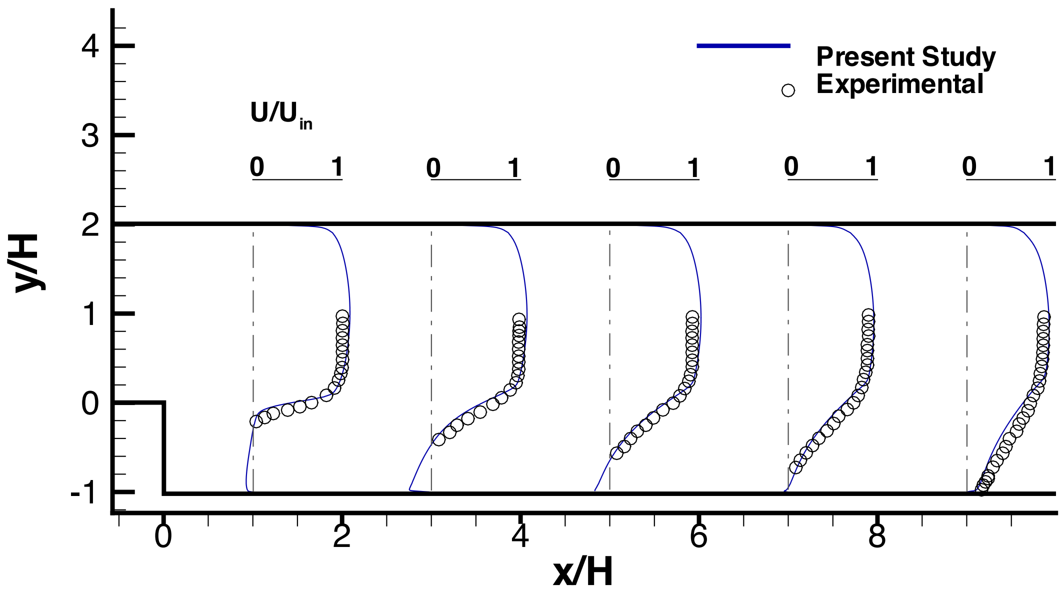


Figure 7.8: Normalised streamwise velocity profiles at x/H =1, 3, 5, 7 and 9. Experimental data from [60]. $Re = 33,000$.

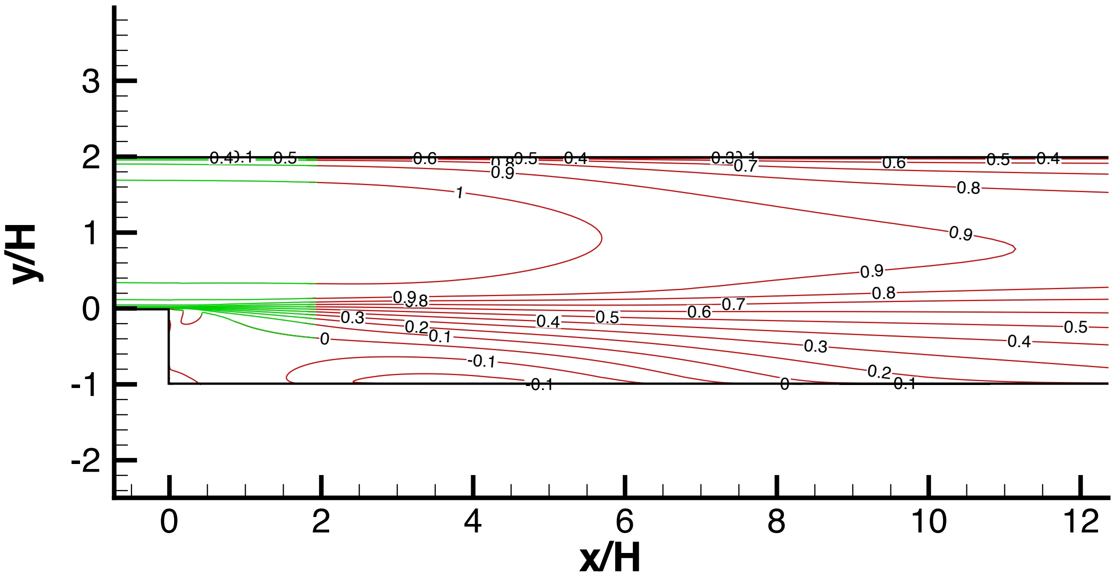


Figure 7.9: Contours of normalised streamwise velocity, $\frac{u}{u_{bulk}}$. $Re = 33,000$.

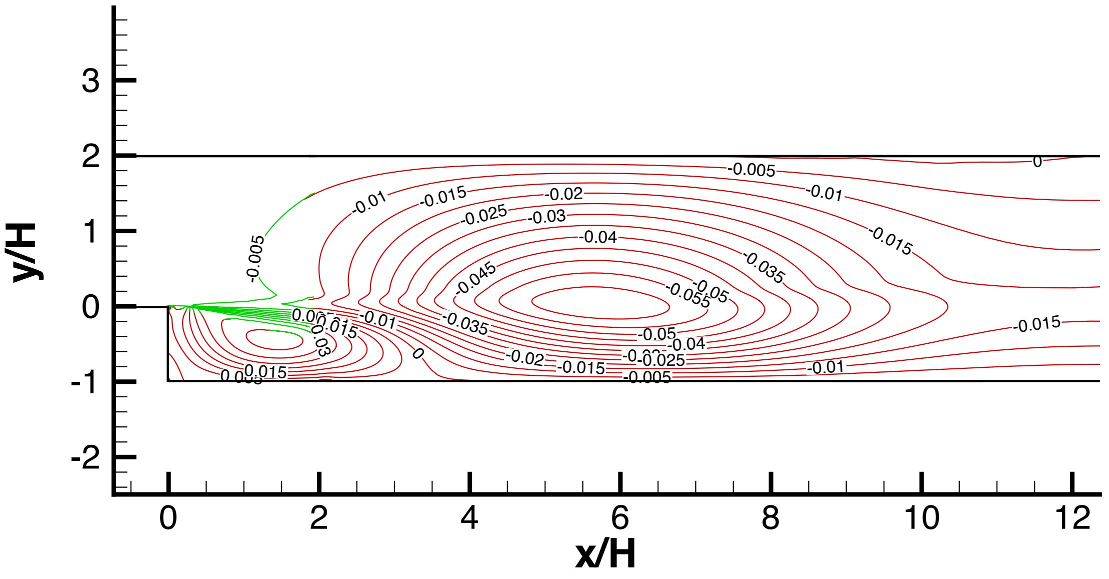


Figure 7.10: Contours of normalised transverse velocity, $\frac{v}{u_{bulk}}$. $Re = 33,000$.

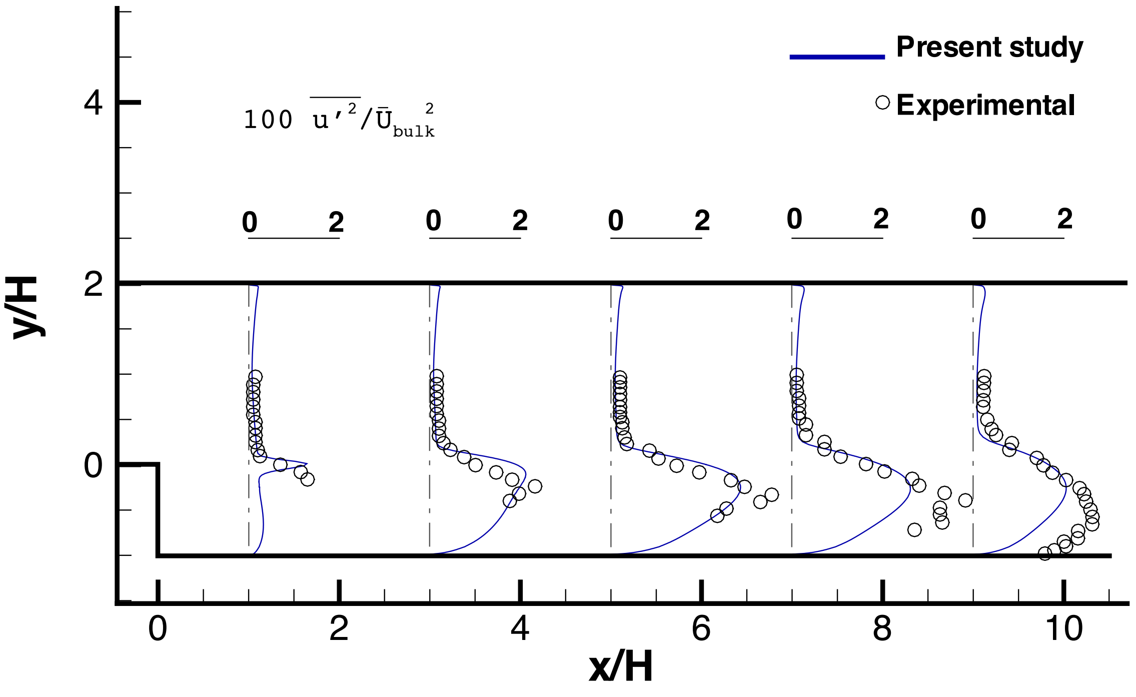


Figure 7.11: Profiles of normalised streamwise normal stress, $100\overline{u'u'}/u_{bulk}^2$ at x/H =1, 3, 5, 7 and 9. Experimental data from [60].$Re = 33,000$.

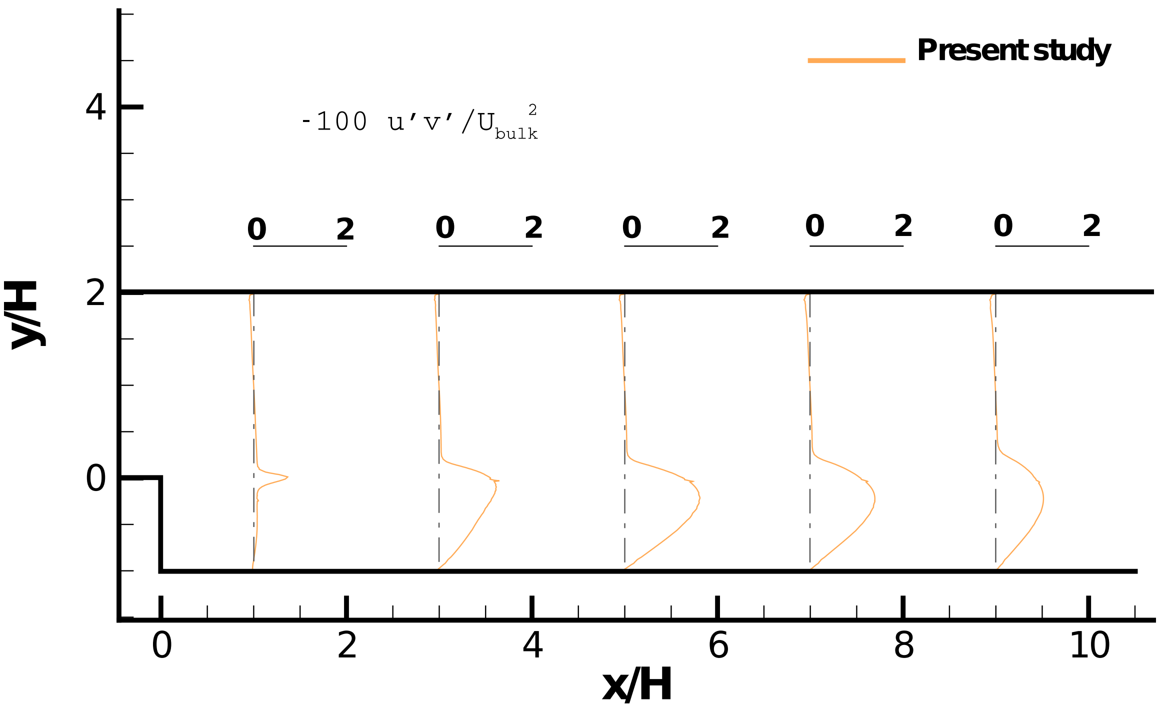


Figure 7.12: Profiles of normalised shear stress, at x/H =1, 3, 5, 7 and 9. $Re = 33,000$.

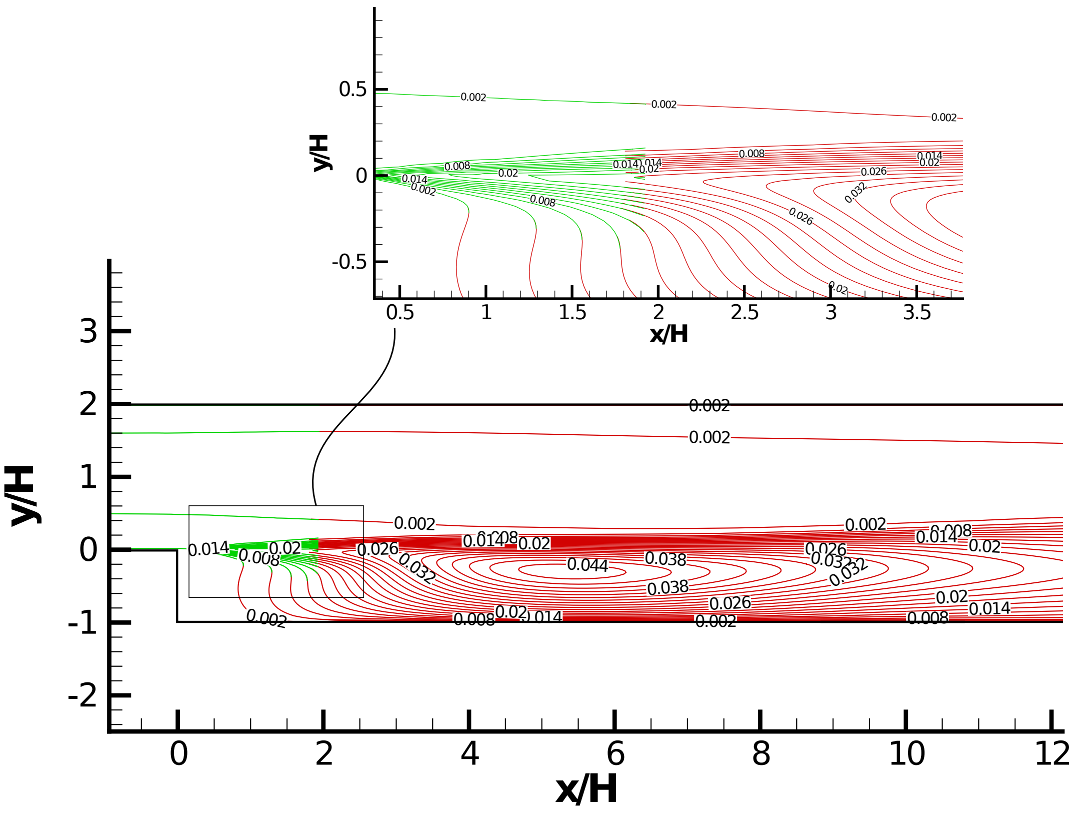


Figure 7.13: Contours of normalised turbulent kinetic energy, k/u_{in}^2. $Re = 33,000$.

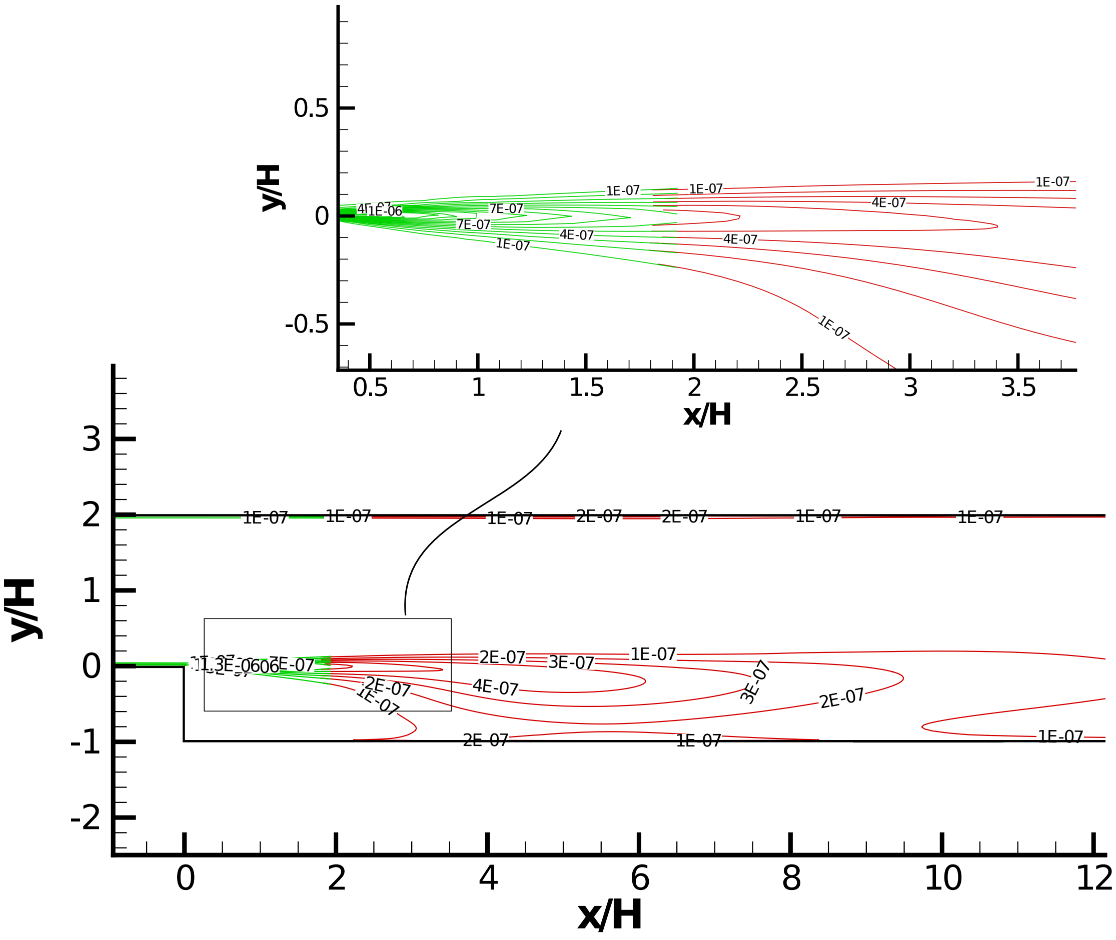


Figure 7.14: Contours of turbulent dissipation rate, ϵ. $Re = 33,000$.

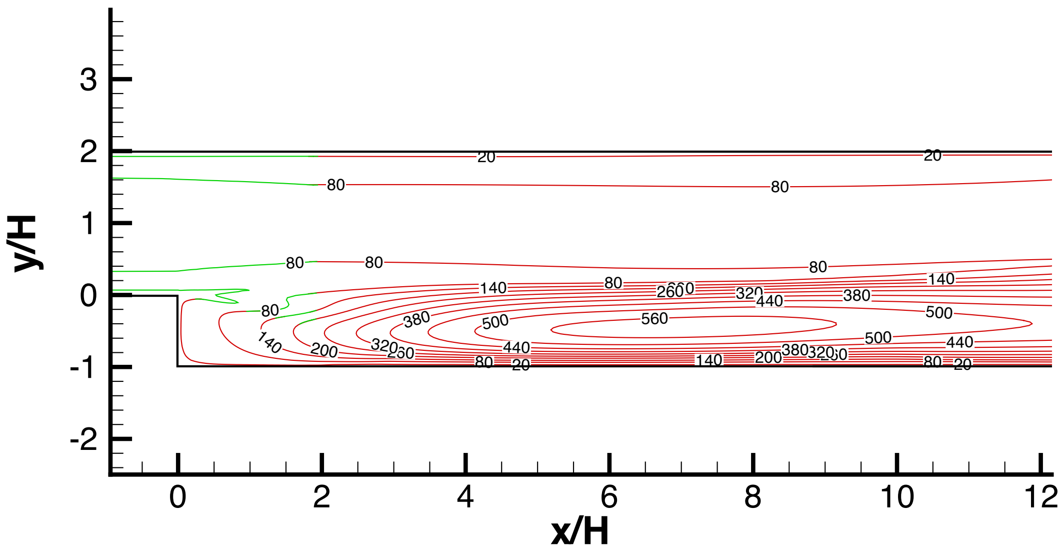


Figure 7.15: Contours of turbulent to laminar viscosity ratio, μ_t/μ. $Re = 33,000$.

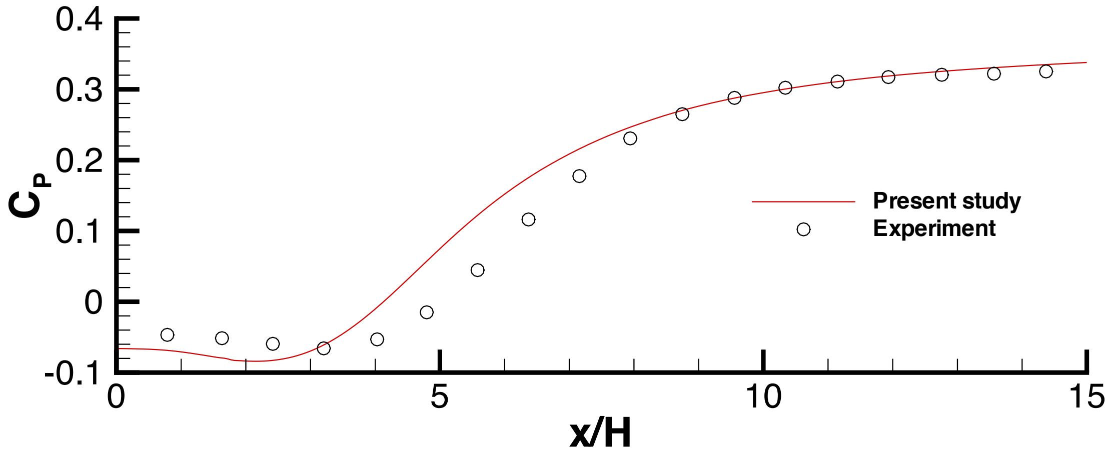


Figure 7.16: Pressure coefficient along the lower wall. Experimental data from [60]. $Re = 33,000$.

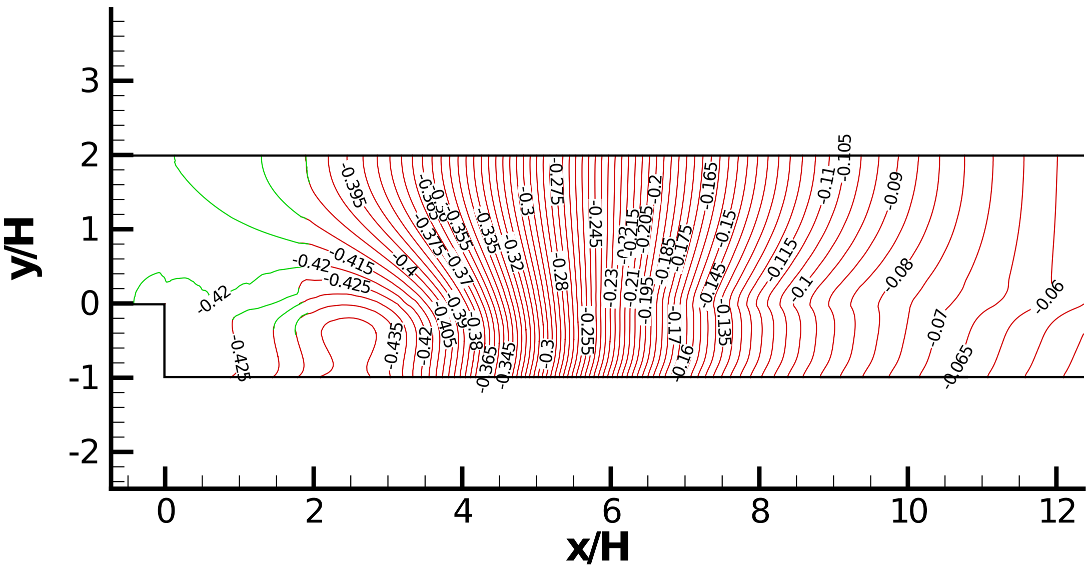


Figure 7.17: Contours of normalised pressure, $P/(0.5\rho u_{in}^2)$. $Re = 33,000$.

Chapter 8

Two element airfoil turbulent flow computations.

The Figures referred to within this chapter can be found from Page 251 onwards.

8.1 Introduction

In this chapter, the turbulent flow over a two element airfoil is considered. Multi element airfoil configurations pose many challenges in mesh generation. A multi-block structured mesh is generally very challenging to generate. This is particularly true for configurations involving both slats and flaps, or flaps which comprise of more than one component. The benefits of the overset method in this situation are clear. Simple body fitted grids around each airfoil component are independently generated and overlaid onto a 'background grid'. The problem is reduced to one of generating C-shape grids around each component, while the background grid can be Cartesian.

The body fitted grids around each airfoil component should ideally extend beyond the fluid boundary layer in order to avoid the undesirable situation where interpolations are conducted at regions of the domain where the flow variables vary rapidly. Since the boundary layer thickness of a slender body such as an airfoil is usually rather thin (e.g. of the same order as the maximum thickness of the airfoil), the total region of the domain occupied by body fitted curvilinear grids does not need to be large. The vast majority of the domain away from the airfoil can be meshed via the use of Cartesian grids. This is desirable from a computational efficiency viewpoint (provided the flow solver is coded to take advantage of this) since Cartesian flow solvers are much more efficient in terms of both CPU and memory usage than their more general counterparts with curvilinear capabilities.

The turbulent flow over a multi element airfoil is characterised by large adverse pressure gradients, strong streamline curvature, possible flow separation with the added possibility of unsteadiness in the separation bubble, in addition to transition at multiple points on the various airfoil components. All these features provide significant challenges for the turbulence model. As such, it is expected that a relatively complex model would be required in order to correctly predict all flow features with a reasonable level of accuracy. However, the focus of the present study is to demonstrate the ability of the overset grid method in dealing with the complex geometry presented by a multi-element airfoil configuration. As such, a relatively simple turbulence model has been employed (a low-Re $k-\epsilon$ model), and accurate flow predictions cannot necessarily be expected. Despite this, it is shown that a very good agreement is found between computation and experiment for the surface pressure (and hence also the lift and moment coefficients). Reasonably satisfactory mean velocity predictions are also made, and any discrepancies are consistent with the shortcomings of a linear eddy viscosity class of turbulence closure reported elsewhere using structured grids (particularly closures that employ a transport equation for ϵ, as is the case in the present study).

8.2 Computational domain and airfoil geometry

Figure 8.1 shows the geometry of the computational domain. This is the same geometry as that considered experimentally by Adair and Horne [63]. Tunnel walls have been included in the simulations owing to the large model dimensions relative to those of the tunnel, thereby leading to the possibility of significant tunnel wall blocking effects.

The two-element airfoil configuration comprises a NACA 4412 main airfoil with a NACA 4415 flap. The flap has a chord length of $0.4c$ (where c is the main airfoil chord length). The location of the flap relative to the main airfoil is specified in terms of the flap gap (FG), the flap overlap (FO), and flap deflection (δ_f) (See Figure 8.1). The values of FG, FO and δ_f used in the present study are $0.035c$, $0.028c$ and 21.8° respectively. The angle of attack of the main airfoil, α, is set to 8.2°. Note that the flap angle is specified relative to the main airfoil angle of attack rather than to the horizontal (the flap deflection from the horizontal is $\delta_f + \alpha = 30°$).

8.3 Numerical Model

The flow considered here is turbulent, and hence a turbulence model is required. The low-Reynolds number $k - \epsilon$ Launder Sharma model [2] has been used for this purpose. The turbulence variables have been discritised via the use of the UMIST scheme, while the QUICK scheme has been used for the velocity components.

The experimental setup detailed in [63] included boundary layer trips on both the pressure and suction surfaces of the main airfoil (at $x/c = 0.025$ and $x/c = 0.01$ respectively). In the present computations, transition is set at these locations for the main airfoil by turning off turbulence production for all cells upstream of the main airfoil's trips. The turbulence variables are permitted to be convected and diffused upstream of the trips in accordance with their governing transport equations. It is simply P_k that is set to zero in this region. It should be noted that there is no transitional state in the present computations; P_k is either on or off.

Experimentally, there is also a boundary layer trip located on the suction surface of the flap (at a distance $x/c = 0.01$ from the flap's leading edge). However, this trip has not been accounted for. It will be seen that transition on the upper flap surface occurs further downstream for the computation than the experiment because of this, thereby adversely affecting the results. There is little that can be done in order to account for the presence of the flap trip given its close proximity to the experimental rake locations (and hence the area of interest). Attempts were initially made to account for the trip by multiplying the production term for the k equation by various set amounts (e.g. a factor of 10) for all computational cells in the vicinity of the trip. While this did improve the predicted transition location, the spurious turbulence levels that this method generated did not have sufficient time to die out before the area of interest (near the flap's leading edge and at the flap's mid-chord there is experimental data). This attempt was therefore abandoned and results presented here are without any modelling of this trip.

8.3.1 Computational Grid

A grid comprising around 66,000 active cells has been employed (where active cells exclude those that have been blanked by the hole cutting algorithm). The total number of cells, including those that are blanked is equal to around 70,000. While these additional blanked cells have some memory overhead, they play no part in the computation, and hence have a negligible CPU overhead. Efficient Cartesian background grids cover the majority of the domain (this is efficient since there are essentially two flow solvers in the present code; one which can deal with Cartesian grids without having to compute or store the unnecessary geometric data, while the other is a more general curvilinear solver). Figure 8.2 shows the mesh used, while Figure 8.3 shows a close-up of this mesh in the vicinity of the gap between the main airfoil and the flap. It can be seen from the former figure that there are two Cartesian grids used. The first is used to mesh the region upstream of the main airfoil's trailing edge, while the second is used to mesh the downstream region where a refinement in the cross-stream direction is applied over the wake.

Previous computational studies of this geometry which have used structured grids have tended to use more computational cells than are used here. Iaccarino and Durbin have used a structured grid with 7 blocks and around 100,000 cells [64] (see Figures 8.4 and 8.5). In Figure 8.4 it can be seen that high aspect ratio cells necessarily propagate outwards in the cross-stream direction towards the tunnel walls, due to the streamwise refinement that has been applied over the flap region. These additional cells increase computational costs, particularly given that significant relaxation of the pressure would likely be necessary for convergence: Since the north and south faces have a significantly different area to the east and west faces, there will be substantial heterogeneity in the coefficients of the pressure correction equation. The larger terms, associated with the larger face areas, will either be treated implicitly or explicitly depending on the sweep direction. In the former case, the diagonal dominance of the equations is diminished, while for the latter case, the source term is large (since the explicit terms are absorbed into the source). Both situations are undesirable. In both cases, the stiffness of the set of equations is high, and hence convergence may be hampered. Additionally, since the larger faces are aligned normal to the flow direction, a perturbation to the streamwise velocity will produce a large mass imbalance in the cell, thereby producing a large correction to the pressure. This correction may introduce a large spurious cross-stream velocity correction through the smaller faces which, if not sufficiently relaxed, could cause instabilities. The overset grid method can easily avoid this issue since the gridding is flexible and hence high aspect ratio cells can generally be avoided (away from walls). Note that high aspect ratio cells are typically still employed close to walls in order to efficiently resolve boundary layers. However in this case the flow will generally be wall bounded, and the high aspect ratio cells are therefore generally aligned with the flow direction. A perturbation to the main streamwise velocity in this case therefore has a relatively small effect on the cell mass imbalance since the face area normal to the flow direction is small.

Figure 8.6 shows a sketch of the domain decomposition that may be required to generate a block structured grid such as that of Figure 8.4. It can be seen that there is considerable complexity in such a decomposition, and this would likely take a great many hours to perform.

The NACA 4-digit specification dictates a blunt trailing edge. In many computational studies, this blunt trailing edge is approximated by a sharp point, thereby facilitating the use of a closed C-shaped grid around the airfoil. However, in the present study (and incidentally in the block-structured grids of Figure 8.4), the blunt trailing edge has been preserved. In order to mesh the geometry while including the blunt trailing edge, two overset grids have been used to mesh each airfoil component (i.e. both the main airfoil and the flap), with one sub-grid comprising an *open* C-shaped grid, while the other is used to close the gap in the vicinity of the trailing edge (see Figure 8.3). Similarly, it can be seen that two aligned blocks are used for each airfoil component in the block structured grid, thereby further complicating the domain decomposition. The overset method however, can deal with such geometrical complexity with ease.

All near wall cells were placed at a perpendicular distance of $4.6 \times 10^{-6}c$ from the wall, except for the tunnel walls, and the very short walls comprising the blunt trailing edge just described. The tunnel walls were modelled as ‘slip walls’ and hence did not require a near wall grid refinement, as will be discussed in Section 8.3.2. At the blunt trailing edge walls, the mesh was not refined to such an extent normal to the wall since it was found that significant refinement offered no appreciable change in the solution, but increased computational costs substantially. The near wall grid spacing was arrived at via a detailed grid sensitivity study, and yielded a y^+ value for the airfoil’s near wall cells of 0.6 or below in all cases (except at the blunt trailing edge).

Computations were carried out on a mesh with double the number of cells. No appreciable difference in the results was observed through the use of such a fine mesh. In any case, the fine mesh was used for all subsequent results presented here. Figure 8.7 shows selected results from this grid sensitivity test, in which profiles of the streamwise velocity component at various locations at and downstream of the main airfoil’s trailing edge are presented.

8.3.2 Boundary Conditions

At the inlet, a uniform velocity has been specified with a magnitude of $30ms^{-1}$, corresponding to a Reynolds number of 1.8×10^6 based on the main airfoil chord length, (c). Turbulence intensity levels at the inlet have been provided in [63] and are equal to 0.0025, 0.0085 and 0.0085 for $\overline{u'}/\overline{u}$, $\overline{v'}/\overline{u}$ and $\overline{w'}/\overline{u}$ respectively. The corresponding turbulent kinetic energy has a value of $0.068J \cdot kg^{-1}$. The value of ϵ at the inlet was set to ensure an essentially laminar flow upstream of the airfoil since the experimental conditions showed very low turbulence levels due to the various screens fitted to the wind tunnel. A turbulent to laminar viscosity ratio of 0.6 was found to work well, from which the inlet value of $\tilde{\epsilon}$ is computed via Equation 4.15 (see Chapter 4).

At the outlet, the pressure is set to zero while a zero gradient condition has been employed for the turbulence variables and velocity components (with an overall bulk correction applied to the normal component).

The no-slip condition has been used for all walls comprising the airfoil surfaces, while a slip boundary condition has been employed for the tunnel walls (this is equivalent to a symmetry condition and is achieved by enforcing a zero gradient condition for all variables, before setting the wall-normal velocity component to zero). A slip boundary condition is used on the tunnel walls since this provides a reasonable approximation of the blocking effect of the tunnel walls (albeit without the slight additional blocking due to the presence of the thin tunnel wall boundary layers), without requiring one to fully resolve the near wall region. A fully resolved no-slip condition at the tunnel wall would require significantly more computational cells, especially given the low Reynolds number turbulence model variant employed, which often requires a sufficiently fine near wall node distribution for convergence.

Interpolations have been conducted via both the standard linear interpolation algorithm described in Section 5.4.1, and the semi-conservative method described in Section 5.4.2. It was found that for this particular case, no appreciable difference in the two methods was observed. This is presumably due to the grid being sufficiently fine for the linear interpolation method to perform in a satisfactory manner, and hence the semi-conservative approach was not required. The semi-conservative approach did however increase convergence rates slightly; a feature of the algorithm that has been discussed in earlier chapters.

8.4 Results

8.4.1 pressure distribution

An overview of the pressure field is provided in Figure 8.8, while Figure 8.9 shows the predicted values of the pressure coefficient over the surfaces of the multi-element airfoil configuration, with experimental comparison using data from [63]. The low pressure region over the upper surface of the airfoil, caused by the flow acceleration over this surface, can clearly be seen. The former figure shows very good continuity from one grid to the next for the pressure, even in regions of steep solution gradients such as the gap between the main airfoil and the flap. It can be seen from the latter figure that there is a very good agreement between the computation and experiment for the surface pressure, suggesting also that the correct lift and moment coefficients have been captured by the computation.

The predicted lift coefficient is equal to 3.28 for the present configuration, while a moment coefficient of $c_m = -0.94$ has been obtained (based on the torque about the quarter chord point). This compares favorably with experimental values of 3.19 and -0.99 for the lift and moment coefficients respectively [63], which were obtained via integration of the measured surface pressure distribution.

8.4.2 mean velocity

An overview of the mean velocity is reported in Figure 8.10 by means of flow streamlines. The strong streamline curvature that is typical of high lift configurations is clearly evident from the figures, presenting a considerable challenge for the turbulence model. This challenge is further compounded by the regions of strong adverse pressure gradients that were earlier seen.

The detail of the flow at and downstream of the main airfoil's trailing edge is assessed by means of six profiles, for which there is also experimental data [63]. The first profile emanates from a short distance upstream of the trailing edge of the main airfoil, while the final profile extends $0.25c$ into the wake (measured from the flap trailing edge). The orientation of all profiles is normal to the wall. Table 8.1 details the specific location and orientation of each profile.

Figure 8.11 shows profiles of the streamwise velocity component at each of the six stations considered. It can be seen that the overall shape of the computed profiles is reasonably close to that of the experimental data for all cases, and that the computation captures, with a reasonable level of accuracy, the main flow features. Having said that, it should be noted that for profiles

Profile	x/c	β (angle from vertical)
1	0.989	23
2	1.031	14
3	1.187	37
4	1.308	44
5	1.322	0
6	1.558	0

Table 8.1: Location and orientation of profiles.

1-3, the distance to the edge of the shear layer (or boundary layer in the case of profile 1) is not well captured by the computations, and hence the computed data is normalised with respect to the computed boundary/shear layer thickness rather than the experimentally obtained value. The discrepancy is around a 10% overprediction at profile 1, and 20% and 15% underprediction at profiles 2 and 3 respectively. The reason for these discrepancies is due to an overprediction of turbulence levels around the main airfoil's trailing edge region. At the root of the problem lies the general inability of the $k - \epsilon$ class of models to deal with adverse pressure gradient flows in a satisfactory way, generally predicting excessive turbulence levels. The excessive turbulence that is predicted acts to increase the boundary layer thickness over the suction surface of the main airfoil upstream of the trailing edge. As this excessive turbulence is convected downstream it acts to 'fill' the upper portion of the wake rapidly, leading to a shorter predicted distance to the edge of the shear layer from the flap surface. In the previously mentioned block structured calculation of Iaccarino and Durbin, [64], a similar discrepancy has been reported despite the use of a more advanced turbulence model in their work (the $k - \epsilon - \overline{v^2}$ model, which employs an additional transport equation for $\overline{v^2}$ - the normal stress component normal to the streamlines - in order to account for some of the turbulence anisotropy).

Table 8.2 shows the values used to normalise the plots.

Profile	U_e/U_in	δ (exp. boundary layer/shear layer/wake thickness, mm)	Correction applied to δ when scaling computed plots
1	1.55	36	+11%
2	1.49	88	−19%
3	1.38	127	−15%
4	0.97	190	0
5	1.05	195	0
6	1.00	203	0

Table 8.2: Edge velocities and distances used to normalise plots.

Profile 1 (Figure 8.11) shows the streamwise velocity close to the trailing edge of the main airfoil. The boundary layer shape seen here is typical of turbulent wall bounded flows. The turbulent fluctuations act to transport high momentum fluid in the upper boundary layer region towards the wall. Similarly, low streamwise momentum fluid close to the wall is transported towards the upper region of the boundary layer. The former result acts to increase the near wall velocity gradient relative to a laminar situation (thereby increasing wall shear stress), while the latter acts to increase the overall boundary layer thickness.

It can be seen from the figure that the computed log layer velocity gradient is lower than that of the experiment. This suggests that the Reynolds shear stress component, $-\overline{u'v'}$, has been overpredicted, which would also explain the reasons behind the overestimate in boundary layer thickness mentioned earlier. It is noted that similar spurious results have also been reported by Rumsey and Gatski, [65] for a different multi-element airfoil configuration using a $k - \epsilon$ class of model. In [66], Rodi and Scheuerer show the shortcoming is due to an inadequate production of ϵ for flows subjected to a strong adverse pressure gradient. It is this characteristic of the turbulence model employed that necessitates the use of a different normalisation between computation and experiment.

Again with reference to Figure 8.11, Profiles 2-4 show the development of the streamwise velocity over the flap. Here it can be seen that the shear layer from the main airfoil has started to merge with the jet of fluid flowing through the gap between the flap and main airfoil, forming confluent shear layers.

At Profile 2, it can be seen that the flap boundary layer is thin (for both computation and experiment). The computed boundary layer however, is perhaps a little too thin. This suggests insufficient levers of turbulence in the computations at this location, which is likely to be due to the absence of the flap boundary layer trip in the computations. This trip is a short distance upstream of the present profile location for the experiments and hence would be expected to have significant affect on the flow here.

Immediately outside of the thin flap boundary layer, the velocity of the jet is seen to decrease with increasing normal distance from the flap wall. This is mainly due to the flow acceleration caused by the suction peak close to the flap's leading edge. The streamwise pressure gradient is greater close to the flap's wall than further away, due to the tighter near-wall streamline curvature of the former. The jet therefore experiences a non-uniform streamwise acceleration across its width.

At yet further wall-normal distance along Profile 2, the streamwise velocity is seen to decrease rapidly as the wake of the main airfoil is traversed. Following the minimum wake velocity, a positive velocity gradient is resumed as the velocity approaches that of the free-stream. It can be seen that the computed results show a higher streamwise velocity in the wake than is observed experimentally. This is again due to the excessive turbulence levels that are predicted around the main airfoil's trailing edge. The turbulent transport of momentum acts to bring high momentum fluid from the free-stream into the wake, thereby 'filling' the wake. Since turbulence is overpredicted at the trailing edge, this effect is overpredicted in the computation.

Moving on to Profiles 3 and 4, it can be seen that a reasonably thick turbulent boundary layer on the flap upper surface is present which is broadly in agreement with experiment, suggesting that transition has occurred at some point upstream of the third rake location. This is supported by Figure 8.12 which shows the skin friction coefficient over the surface of the flap. Transition is characterised by a sharp increase in the skin friction due to turbulent transport of high momentum fluid towards the wall. It can be seen that transition is predicted too late, and hence the development of the flow over the flap will be different in the computations than the experiment.

Profile 4 shows a small region of reversed flow close to the wall as separation occurs at a short distance upstream of the flap's trailing edge. This is observed both computationally and experimentally, although the magnitude of the reversed flow seems to be slightly larger for the latter.

Profiles 5 and 6 (Figure 8.11) lie in the wake. Evidence of the trailing edge separation is again seen via the reversed flap-wake flow for the former profile. This is seen in both the computation and the experiment. At profile 6, the flap and main airfoil wakes seem to have merged into one overall wake for the experimental data, but not for the computation. There is however insufficient cross-stream resolution in the experimental data to determine conclusively if this is indeed the case. The Reynolds stress profiles (to be presented presently) also suffer from relatively poor cross-stream resolution, hence the presence, or otherwise, of the flap wake cannot be inferred from the Reynolds stresses.

Overall, the streamwise velocity component is in reasonably close agreement with the experimental data.

8.4.3 Turbulence Results

An overview of the turbulence field is reported in Figure 8.13 which shows contours of the turbulent kinetic energy. It can be seen that the undisturbed flow turbulence levels are close to zero everywhere, which is consistent with the very low experimental turbulence levels at the tunnel entrance in the experiment. Turbulence levels are high over the suction surface of the main airfoil and flap. The pressure surfaces of both components have a very thin turbulent boundary layer due to the comparatively low mean velocity gradients normal to these surfaces. In addition, the low wall normal mean velocity gradient across the width of the jet (reported previously) is responsible for the low turbulence in the core of the jet that is seen to persist over the length of the flap and some distance into the wake. In the wake, there are two major peaks in the turbulent kinetic energy which are associated with the main airfoil's upper and lower boundary layers (now separated into shear layers). Smaller peaks in turbulence energy are also present and are associated with the layers of high mean shear emanating from the flap surfaces. The turbulence levels at the peaks just described are augmented by the streamline curvature associated with the wake deflection that occurs due to relatively close proximity of the lower tunnel wall.

Figure 8.14 shows the streamwise component of normal Reynolds stress at the same profile locations as were considered for the mean velocity. It can be seen from the figure that for all the profiles, there is a significant underestimate in this Reynolds stress component. This comes as little surprise since turbulence anisotropy is not modelled. The streamwise component is energised via large $\partial\overline{u}/\partial y$ mean velocity gradients and hence would be expected to be the larger normal stress component.

At Profile 2, it can be seen that the computed $\overline{u'^2}$ Reynolds stress shows sharp peak at the flap surface, which is consistent with the very thin boundary layer thickness reported earlier. The experimental data suggests again that this boundary layer is somewhat too thin; this is due to the absence of the flap boundary layer trip in the computation. By Profile 4, it can be seen that the normal stress has grown significantly for the experimental data. This is presumably due to the large mean velocity gradients that are present in the separated recirculating flow at the flap's trailing edge (as was seen in Figure 8.11). That the computed turbulence shows only modest growth in $\overline{u'^2}$ near the flap surface between Profiles 3 and 4 suggests that separation is predicted too late by the computation. This was also suggested in Figure 8.11 where it was seen that the strength of the reversed flow is weaker for the computed results.

Figure 8.15 shows profiles of the Reynolds shear stress. At Profile 1, it can be seen that the shear stress is indeed overpredicted, thus explaining the excessive boundary layer thickness mentioned earlier. The peak in turbulence levels associated with the recirculating flow near the flap surface by

Profile 4 is again unsurprisingly underpredicted by the computation (since separation is predicted too late).

The overall trend in the Profiles of Figure 8.15 suggest that the computed results are reasonable, and are as good as should be expected from a linear eddy viscosity model.

8.5 Conclusion

The turbulent flow around a two-element airfoil has been computed. The results that have been obtained are in reasonable agreement with the experimental data. The main discrepancy seems to be in the boundary layer thickness at the main airfoil's trailing edge which is overpredicted by around 10%. This discrepancy is due to shortcomings of the turbulence model, and more specifically, due to insufficient turbulent dissipation production in regions of strong adverse pressure gradient. Other authors have also found similar shortcomings for the $k - \epsilon$ class of turbulence models, both for this geometry using structured grids, and also for a similar geometry using overset grids.

The excessive levels of turbulence that are convected from the main airfoil's upper surface act to displace the wake downward since turbulent transport of momentum causes the upper portion of the wake to be 'filled' rapidly. The shear layer thickness at Profiles 2 and 3 was therefore seen to be underpredicted. Other than the large differences in the boundary/shear layer thicknesses, it has been shown that the mean velocity is predicted with satisfactory accuracy.

The overset method has proven very useful for this geometry. Airfoil designs consisting of many components could be handled by the method with ease. The same cannot be said for a block structured arrangement, where each additional airfoil component would compound to the already significant domain decomposition task.

8.6 Figures

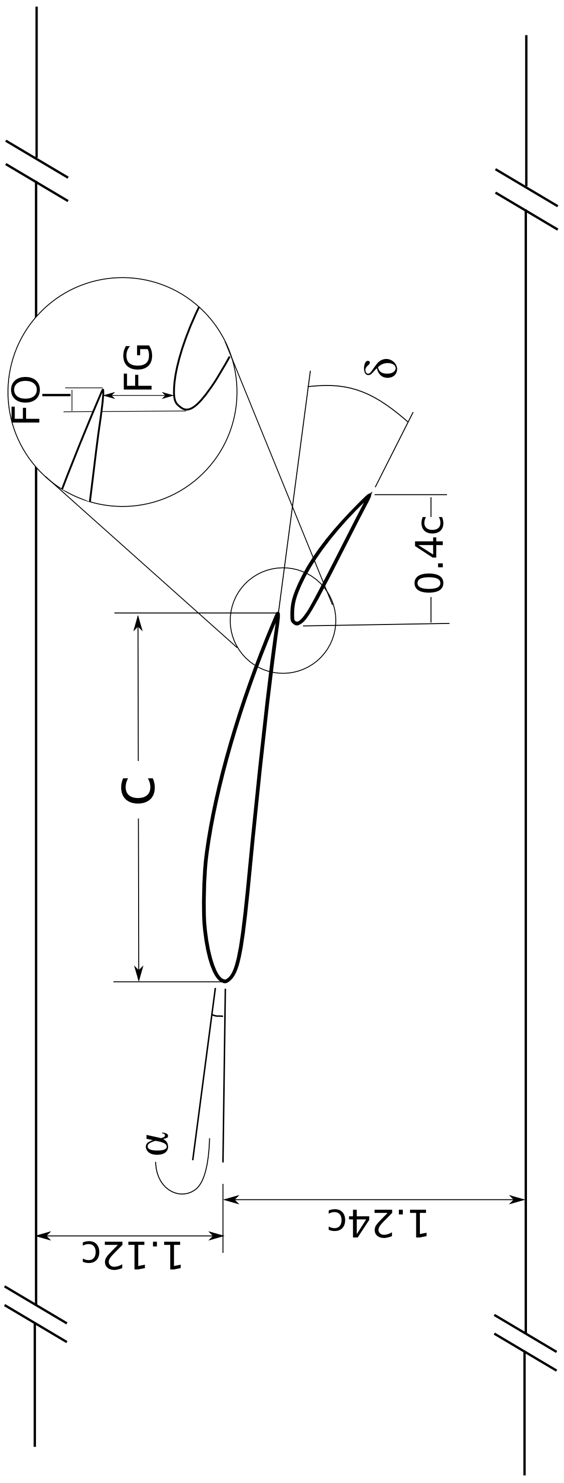


Figure 8.1: Geometry of multi-element airfoil.

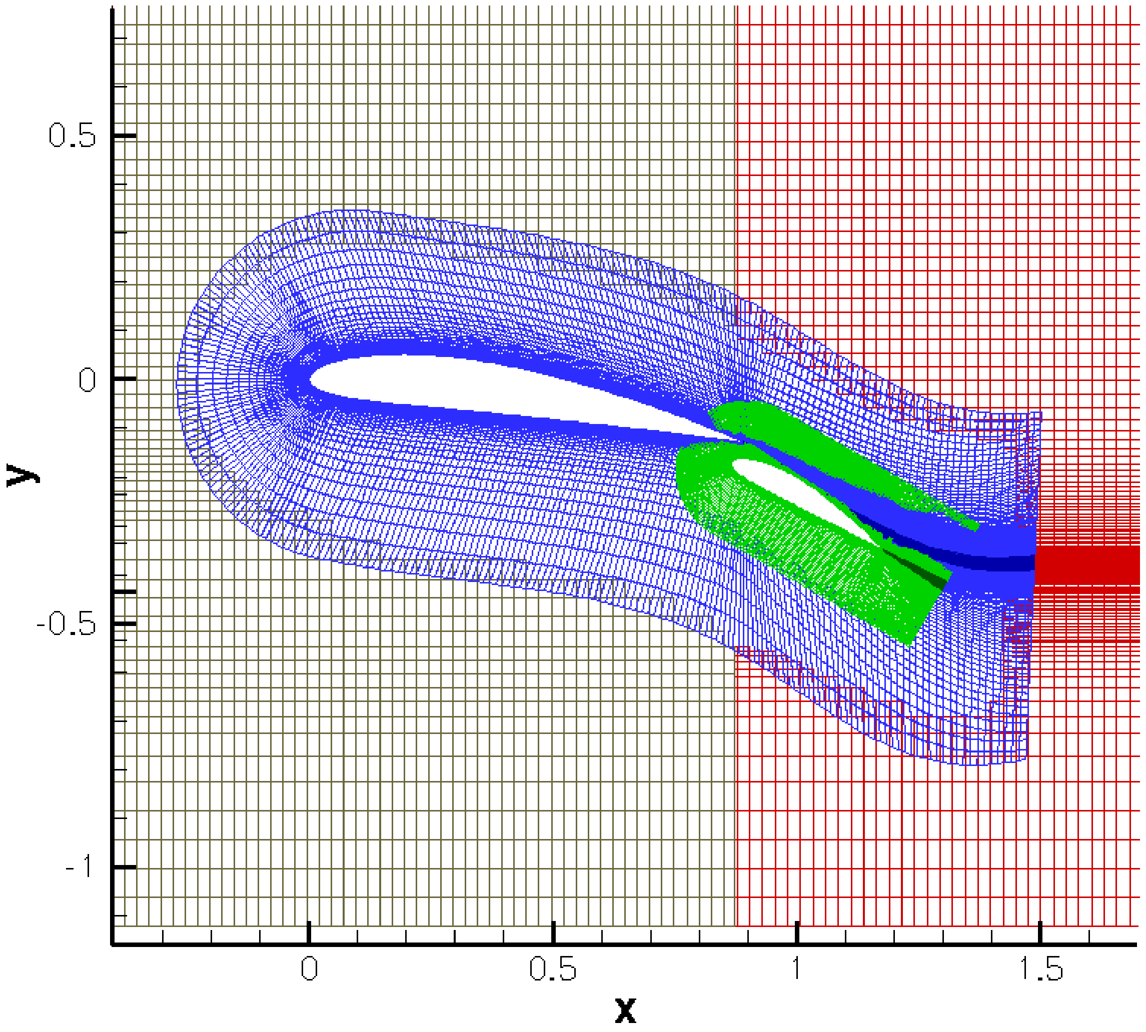


Figure 8.2: Overset mesh used, comprising of a total of 6 subgrids.

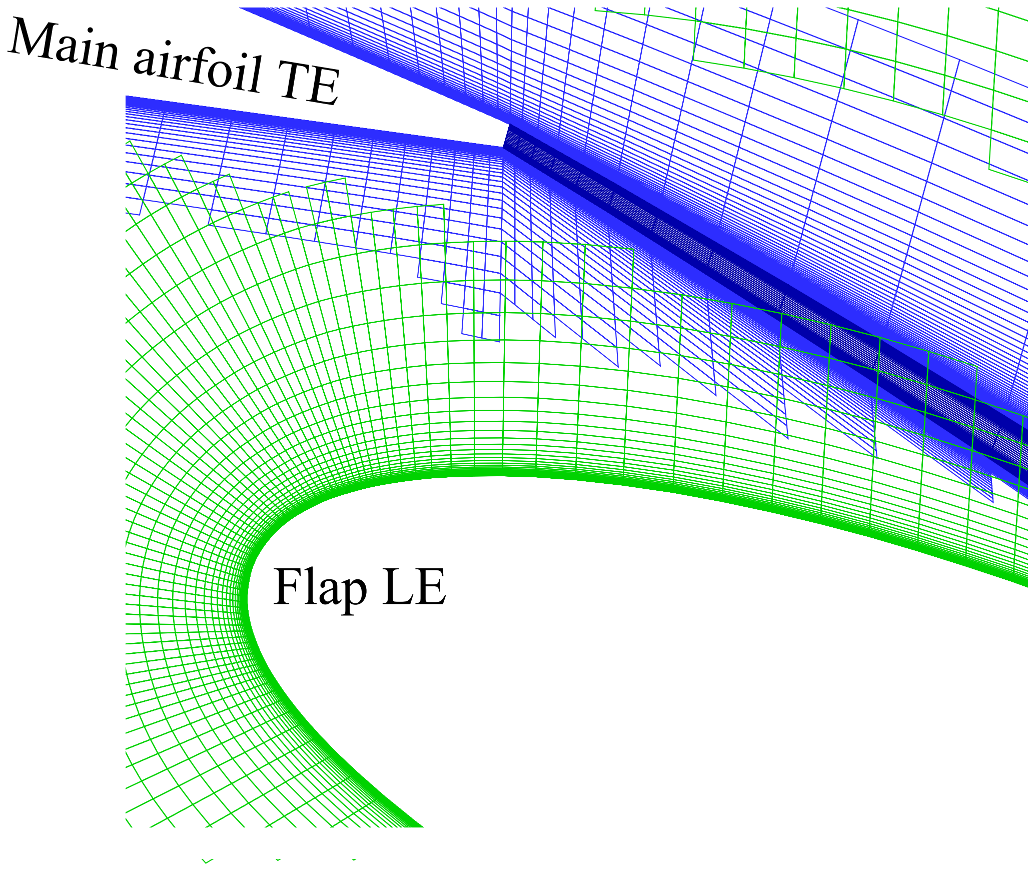


Figure 8.3: Close-up of meshes in gap region.

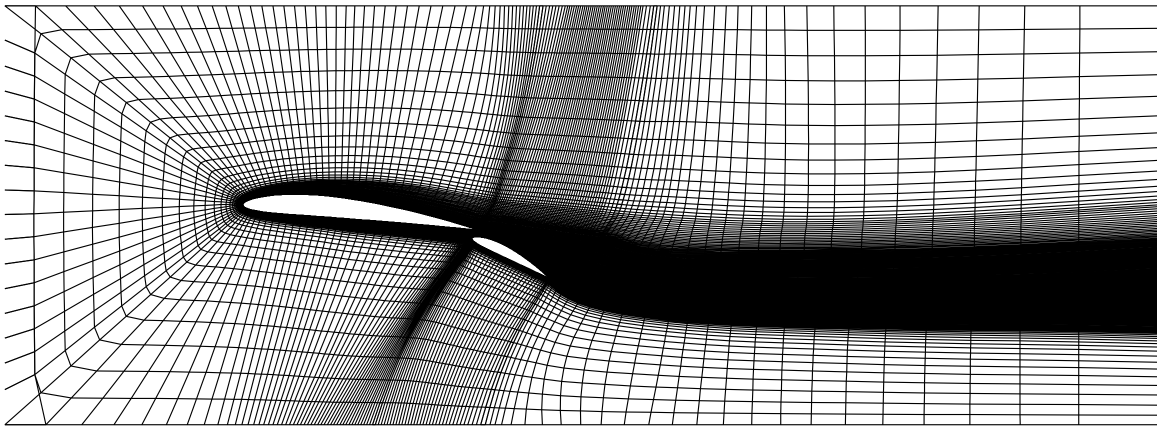


Figure 8.4: Mesh used in [64] for the study of the same geometry using block-structured grids.

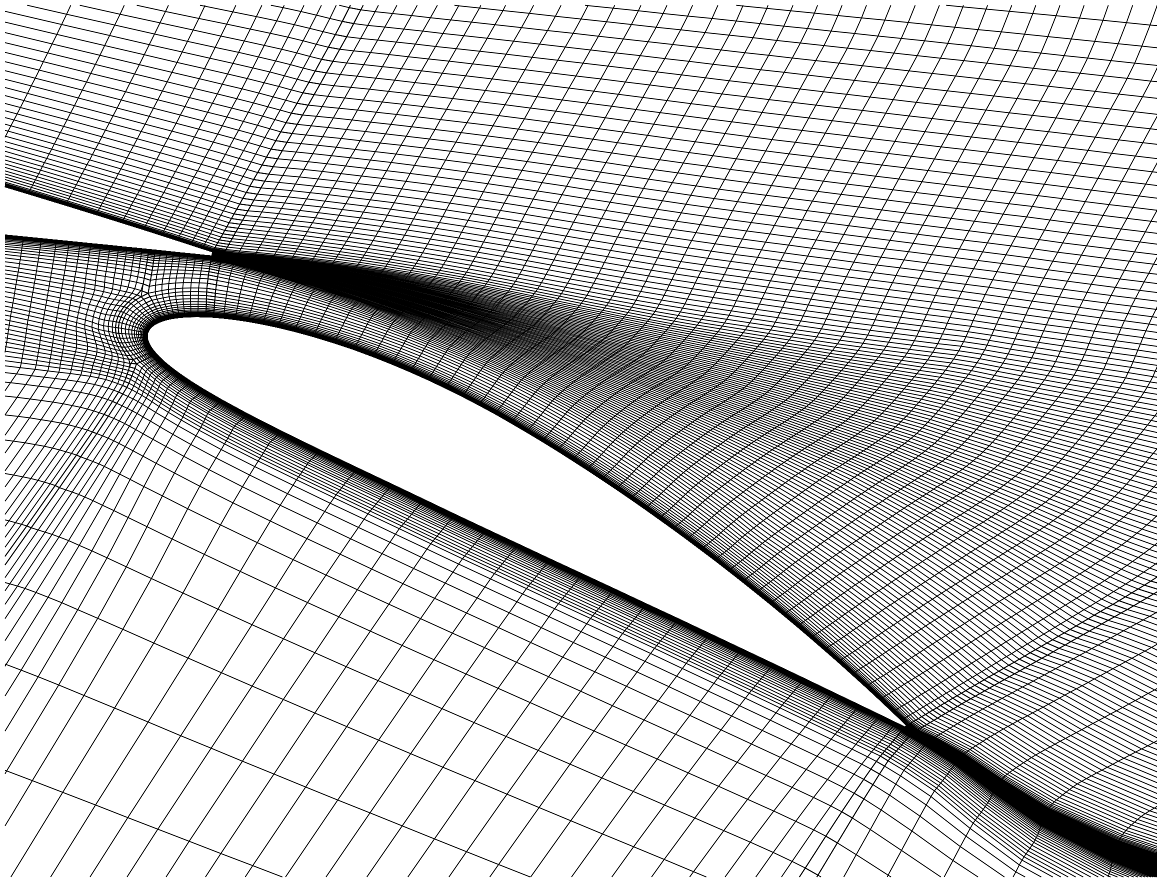


Figure 8.5: Close-up of mesh used in [64] for the study of the same geometry using block-structured grids.

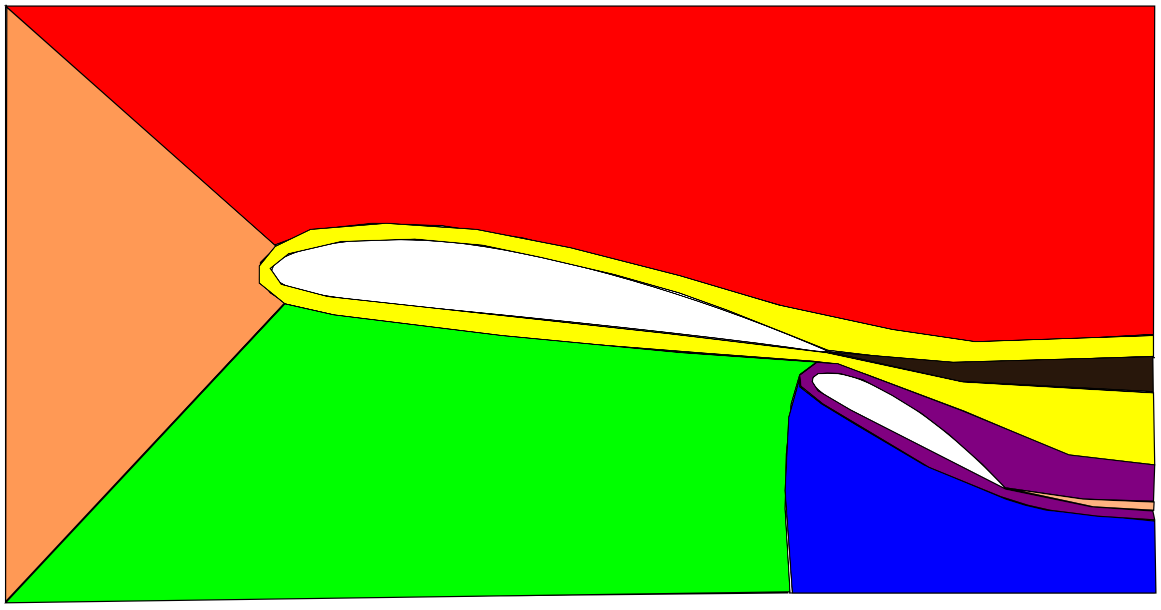


Figure 8.6: An example of the domain decomposition that may be required to achieve a block-structured grid. Each coloured region represents a block.

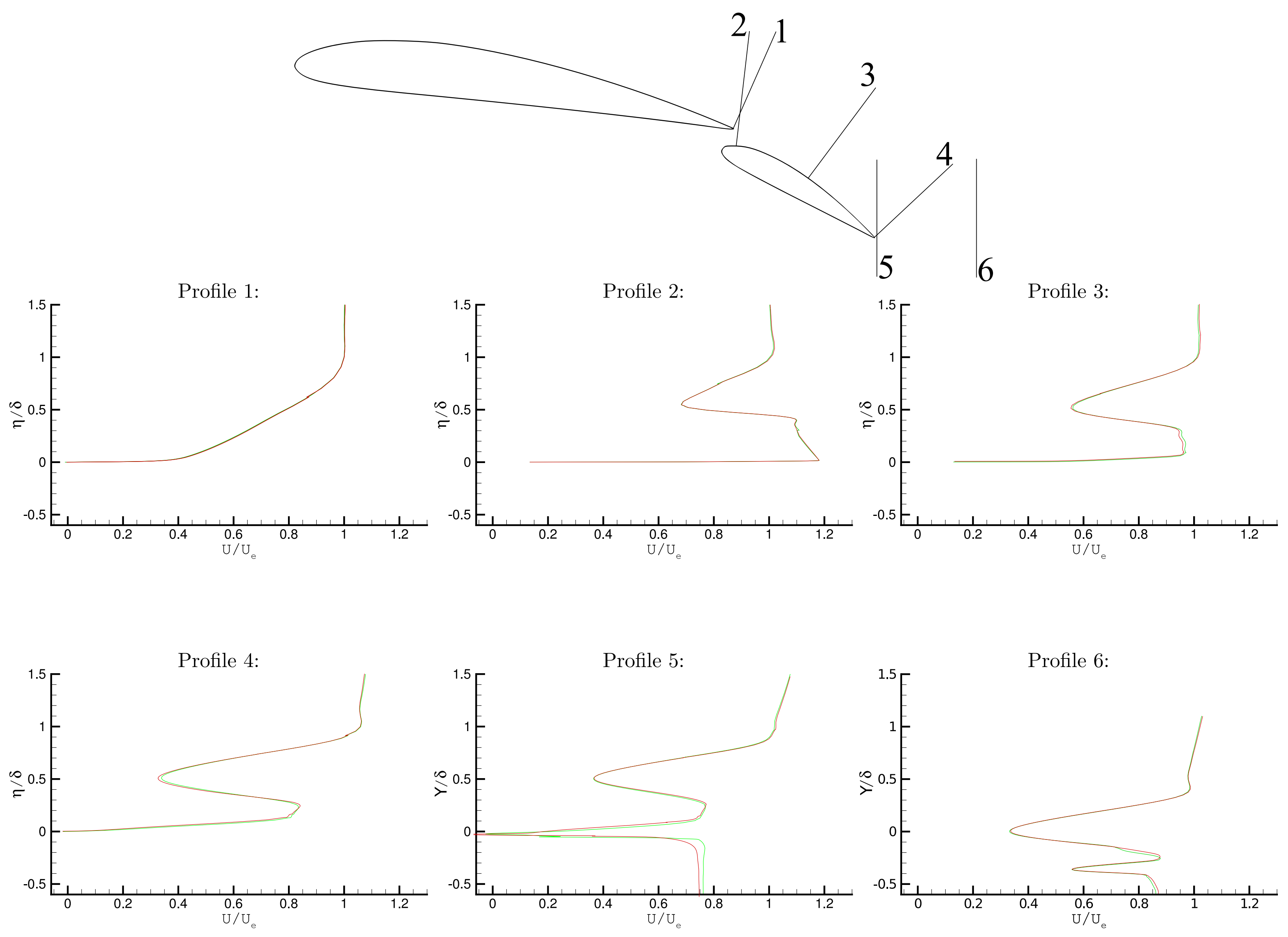


Figure 8.7: Profiles of the streamwise velocity for two grid resolutions. Initial grid, 66,000 cells (Green) and fine grid, 120,000 cells (Red).

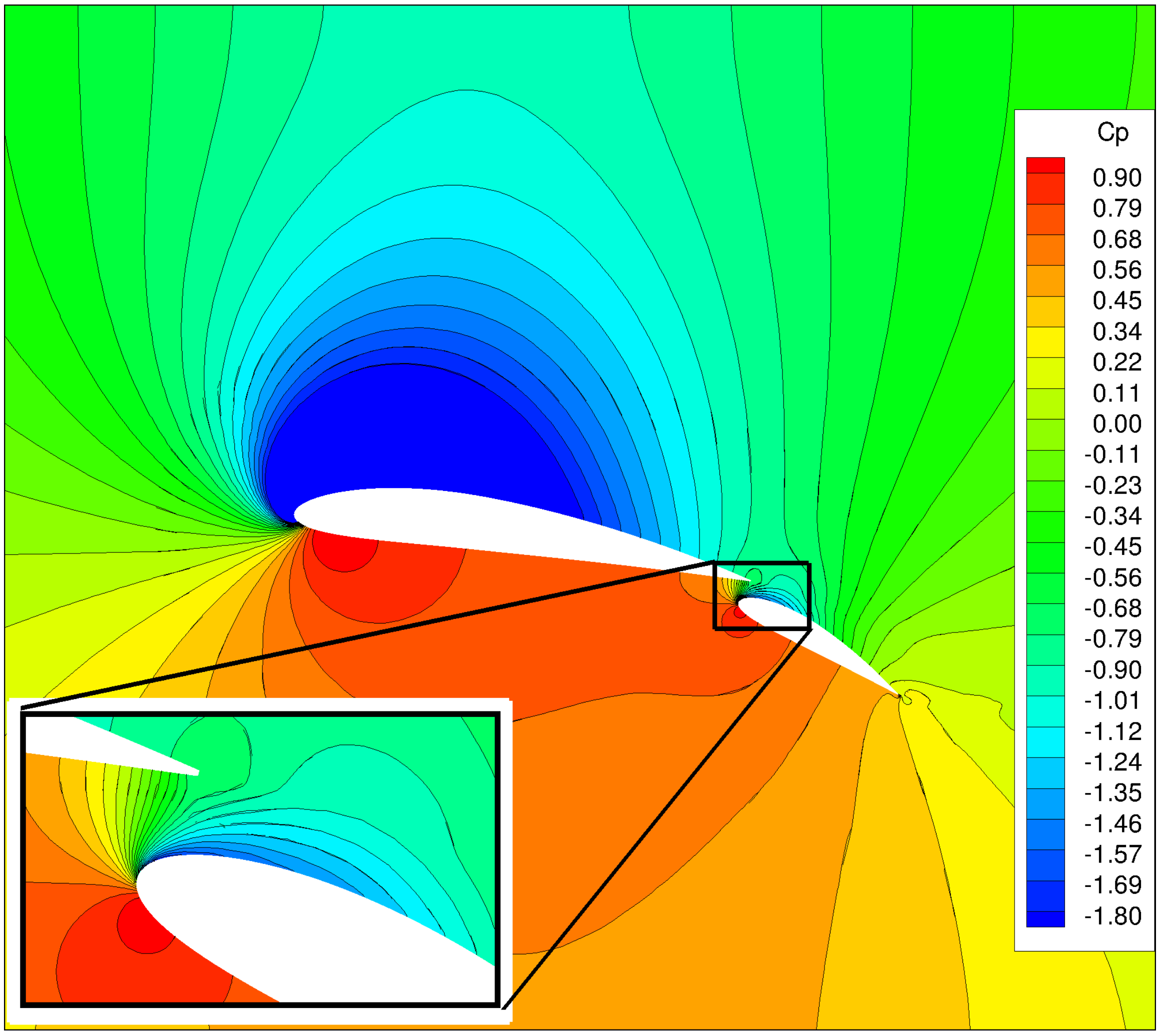


Figure 8.8: Contours of pressure coefficent C_p. Insert shows detail in region of gap.

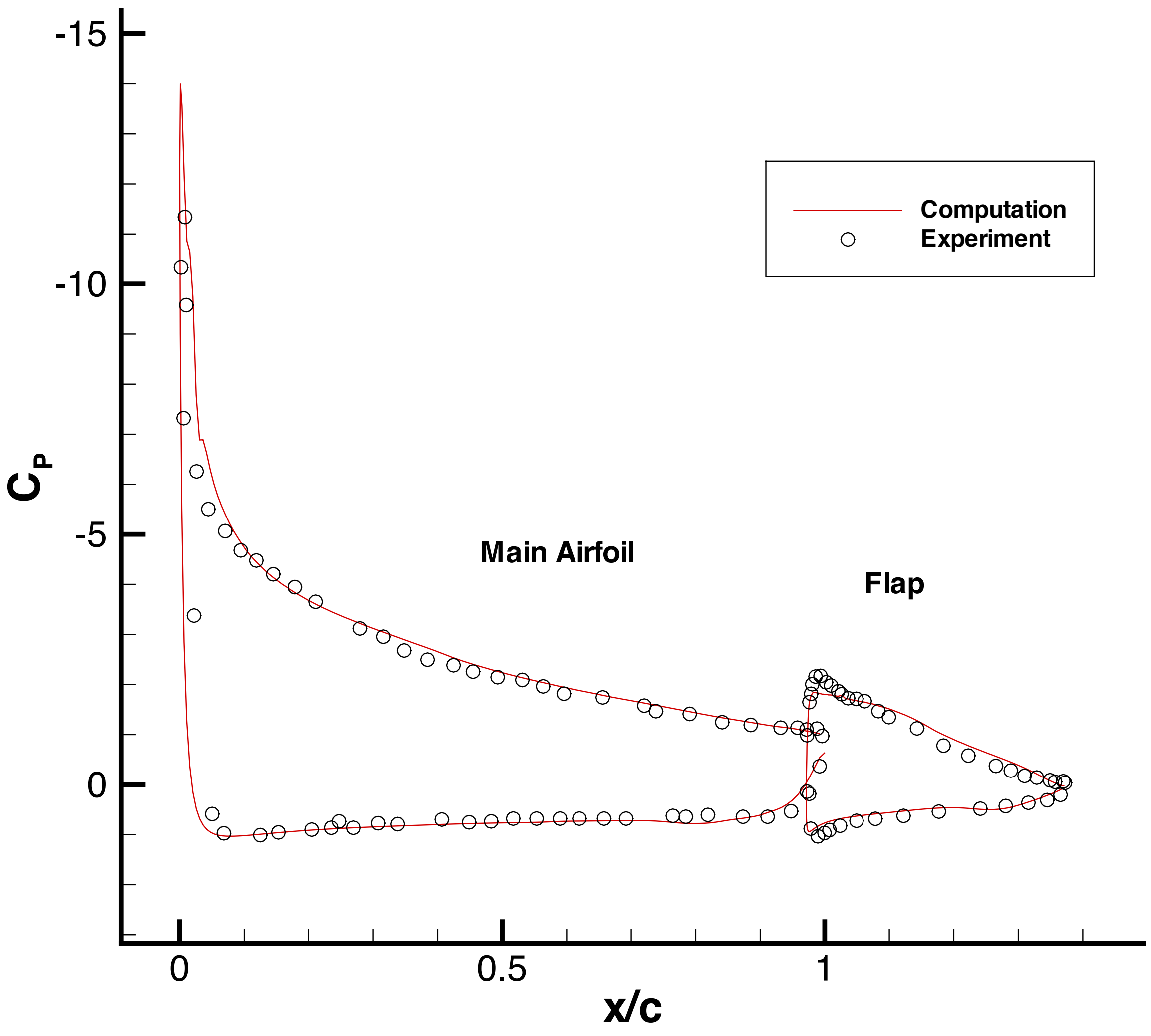


Figure 8.9: Pressure distribution over main airfoil and flap

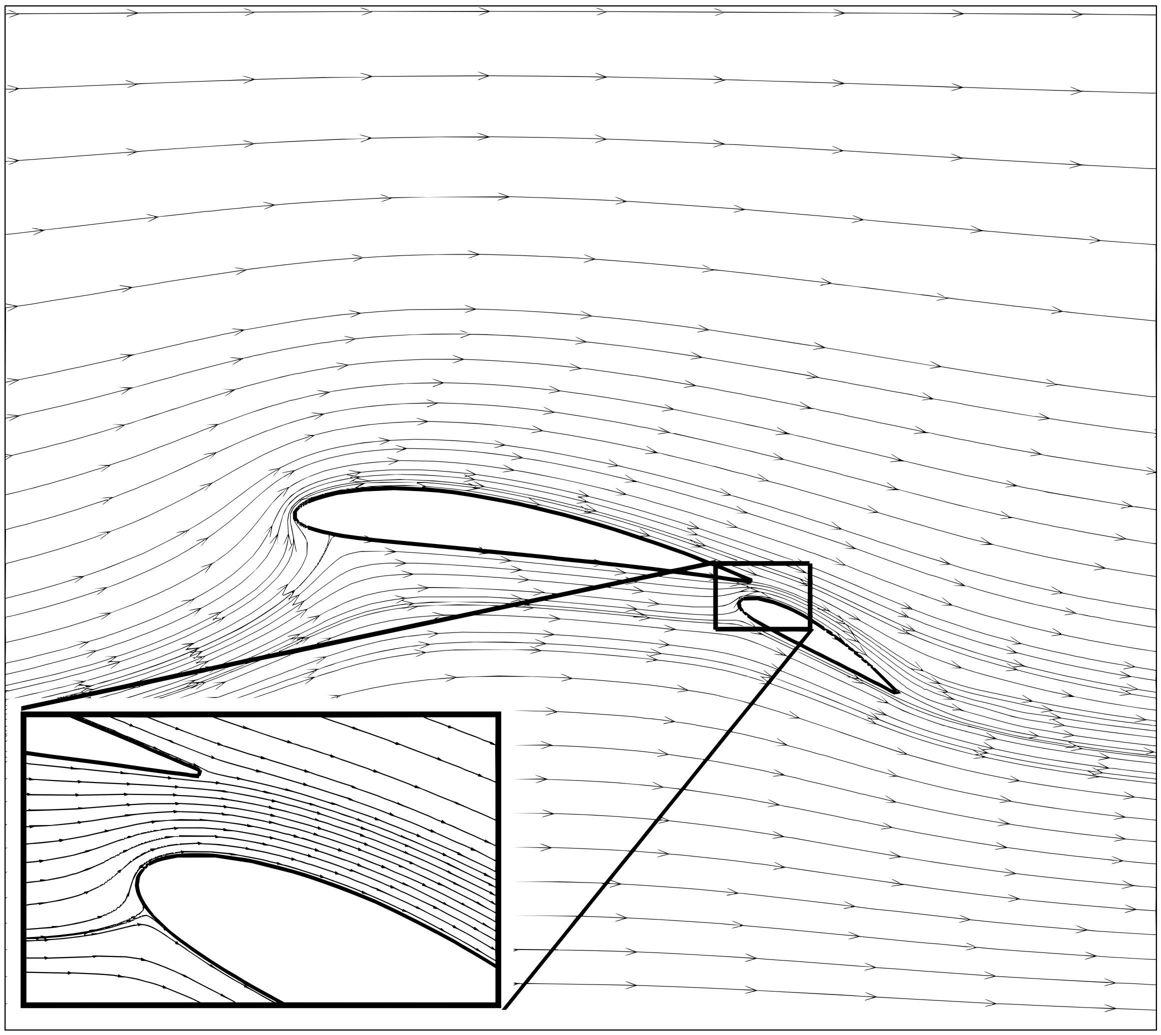


Figure 8.10: Flow streamlines. Insert shows detail in region of gap.

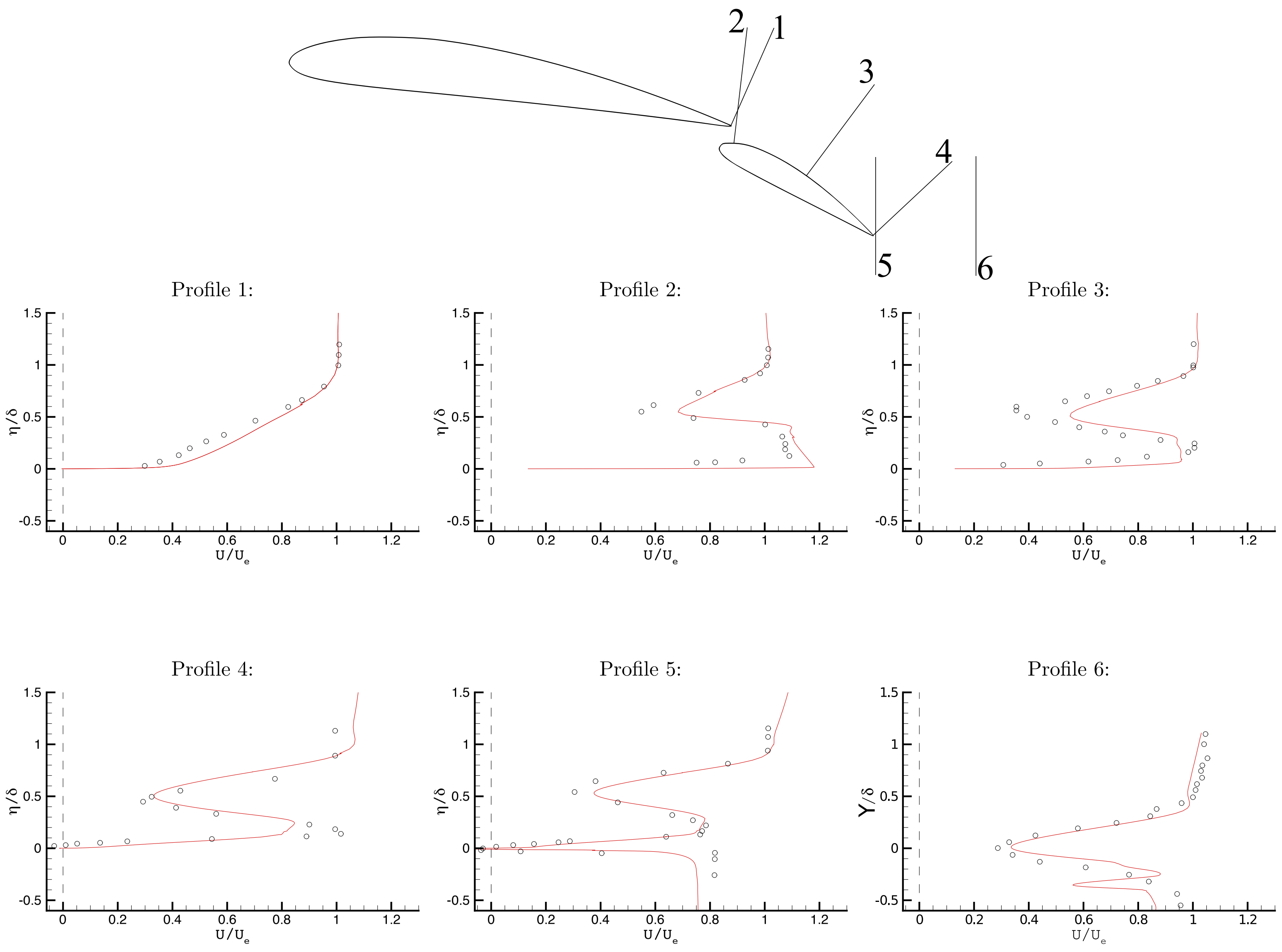


Figure 8.11: Profiles of the streamwise velocity. Profile locations are indicated in the figure (the orientation and location of the profiles in the sketch is accurate, although the length is not). Locations are also stated in Table 8.1.

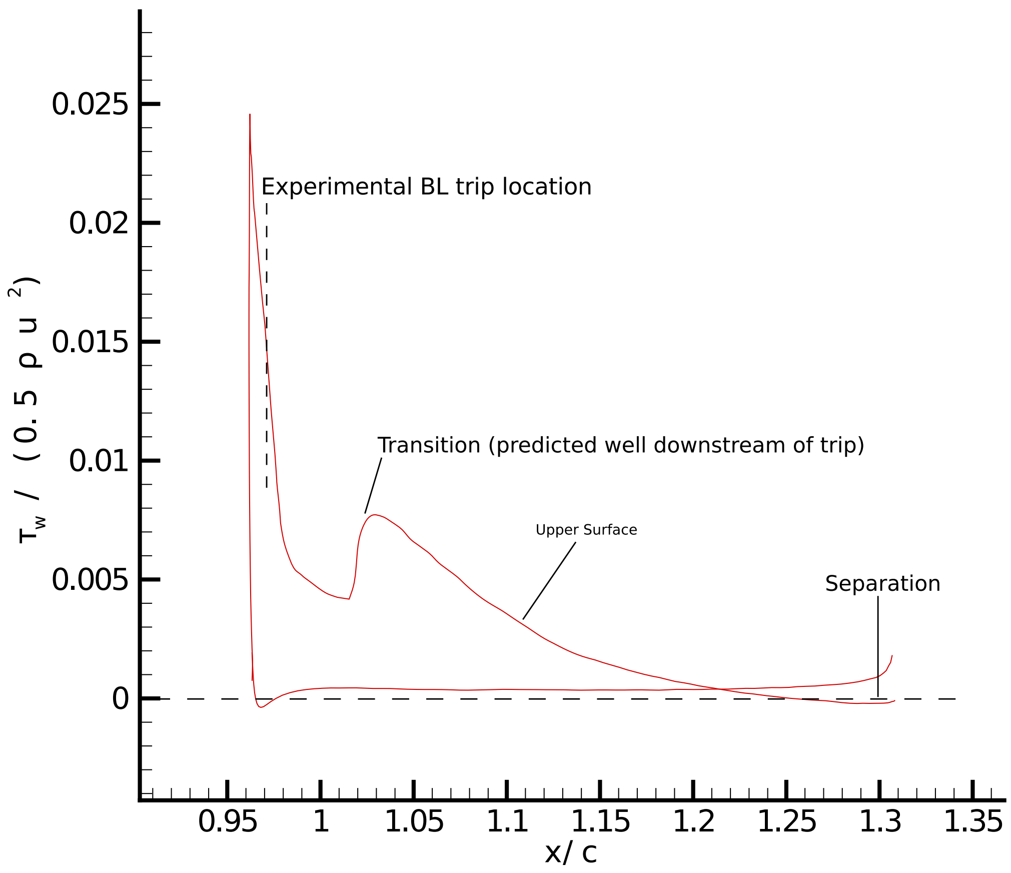


Figure 8.12: Skin friction over the flap surface. Plot indicates that transition is predicted too late.

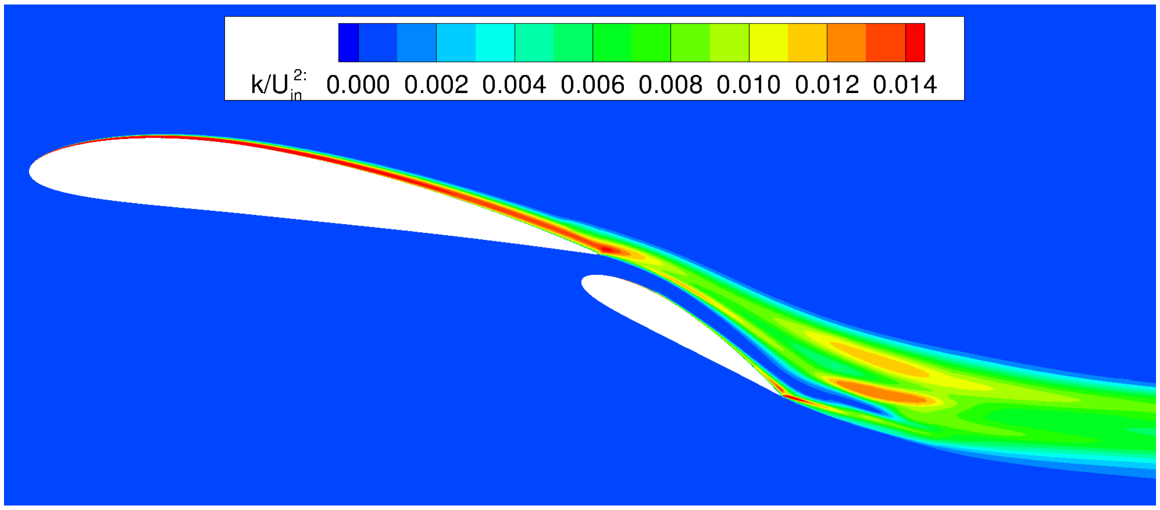


Figure 8.13: Contours of the turbulent kinetic energy, normalised by the square of the inlet velocity.

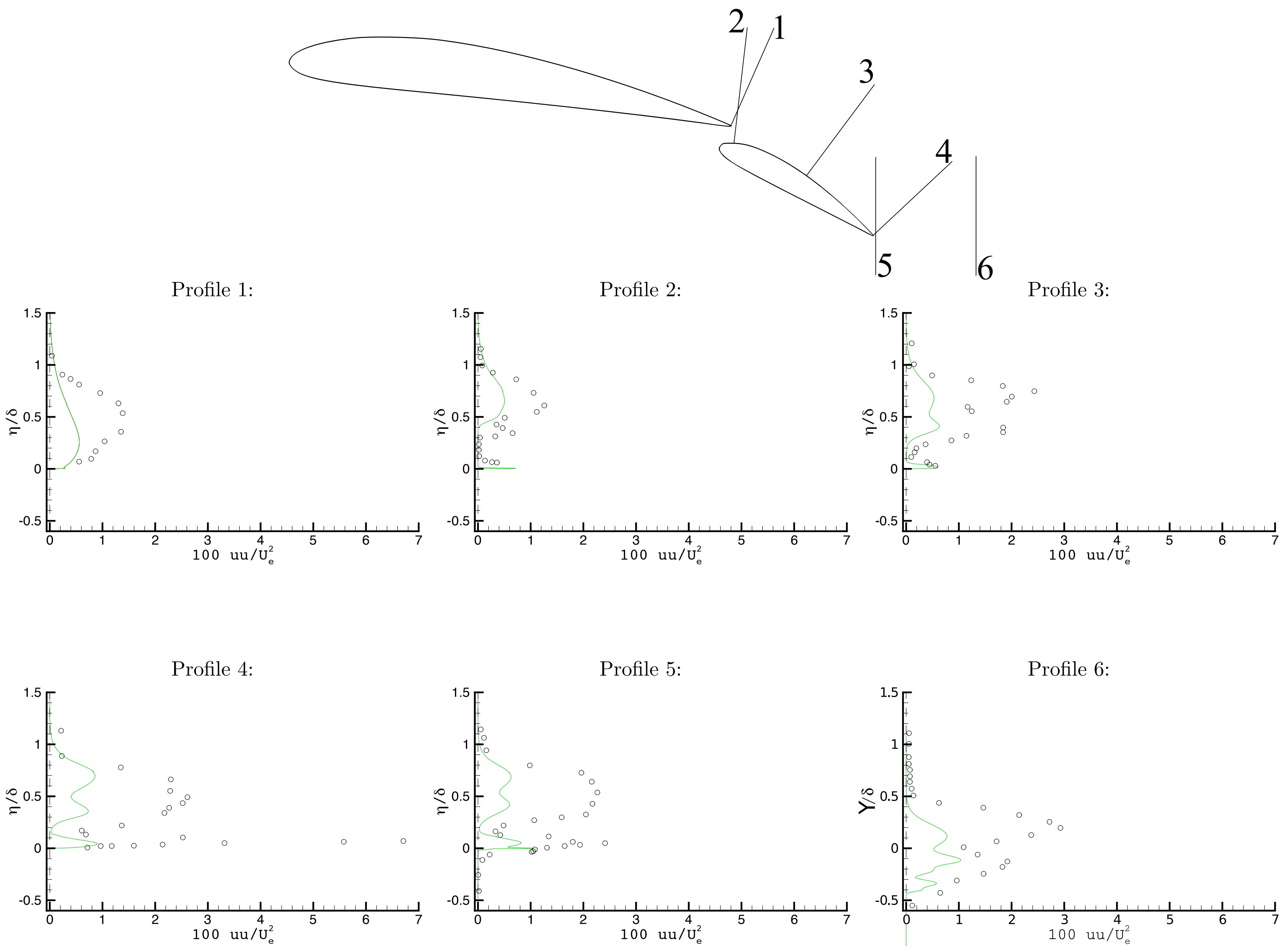


Figure 8.14: Profiles of the streamwise normal stress component $100\overline{u'u'}/U_e^2$. Profile locations are indicated in the figure (to scale) and are stated in Table 8.1.

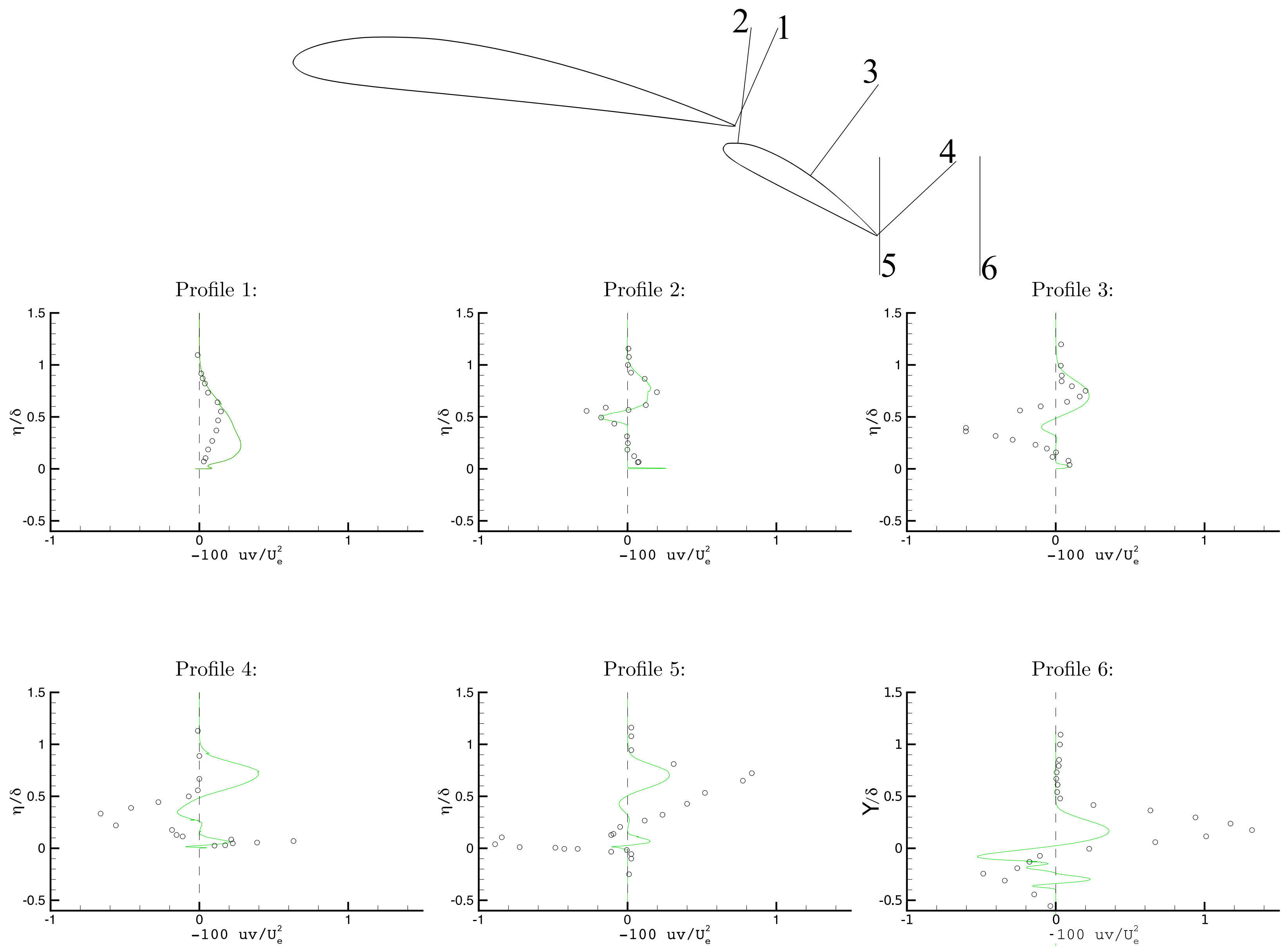


Figure 8.15: Profiles of the shear stress component $-100\overline{u'v'}/U_e^2$. Profile locations are indicated in the figure (to scale) and are stated in Table 8.1.

Chapter 9

Three-Dimensional Pipe Flows

The Figures referred to within this chapter can be found from Page 288 onwards.

9.1 Introduction

The overset grid method is a particularly useful tool for meshing pipe domains. For example, an O-shaped grid can be used to depict the outer portion of the pipe, while the core can be meshed via the use of a separate grid of Cartesian section. This therefore avoids the issues of a grid singularity at the centre of the pipe that would be encountered via the use of a single O-shaped block. Complex domain decomposition issues that are encountered when generating a block structured grid are also avoided. The overset method is particularly useful for meshing junctions between two or more pipes. In this case, a block structured grid would be particularly difficult to generate, while the overset method can handle such domains with ease.

As with all internal flows, it is important in pipe flows to enforce global mass conservation via the use of a bulk correction. The methods described in Section 3.6.1 can be used for this purpose. However, due to the use of the two overlapping grids to depict the pipe geometry, computing the mass flux through the inlet or outlet is complicated somewhat. The issue of avoiding counting the overlap region twice has been outlined already, and is non-trivial for the general case where some cells only partly overlap other grids. The grid zipping method described in Section 5.6 has been used to overcome this problem. Previous computational studies have used individual sub-grids with a singularity at the centre to deal with this issue [67, 68], and hence the flow solver would need to be able to deal with prism shaped cells. This method also introduces high aspect ratio cells into the core of the pipe which is undesirable from a convergence rate viewpoint. An alternative would be to use block structured grids for individual pipe sections, with the overset method being used at the interface between these pipes in order to deal with the geometrical complexity at a junction. While this latter method is reasonably satisfactory, a certain amount of domain decomposition work still needs to be done and hence the zipping algorithm appears to provide a better option.

In this Chapter, the flow through a 90° pipe bend and the flow through a branching artery (the carotid bifurcation) are considered. For each of these flows, it will be apparent that the overset method offers several advantages over alternative griding techniques.

9.2 The flow through a pipe with a 90° bend.

The flow through a pipe with a bend has been widely studied both experimentally and computationally (e.g. [69, 70, 71, 72, 73, 74]) owing to the fact that such a geometry is frequently encountered in many situations such as in the nuclear industry, in heat exchangers, in human anatomy [75], in turbine blade thermal design [76], as well as a great many other situations. The flow in a pipe bend also bears strong similarities to the flow in the human carotid bifurcation which will be considered in the next section. The simpler geometry of a pipe bend has initially been considered in order to further develop and test the overset algorithms described in Chapter 5 before applying them to the more complex case of the flow in a generic human artery bifurcation. The study also serves as a useful validation of the 3D capabilities that have been incorporated into the new CFD code developed in this project.

While the geometry of the pipe bend is relatively simple, the flow is not; the interaction between inertial, viscous and centrifugal forces generates secondary fluid motions (i.e. those in a plane perpendicular to the axial flow direction). This results in the formation of helical vortex structures. The transport of fluid via the secondary motions generates a marked distortion of the axial velocity profile, shifting the peak in velocity from the centre of the pipe towards the outer side (i.e. away from the centre of curvature). This consequently affects the wall shear stress, yielding a non-uniform shear in circumferential and axial directions (contrary to that which would have been observed for Poiseuille flow where the wall shear stress is uniform at all points on the pipe).

9.2.1 Geometry, boundary conditions and numerical model.

The specific geometry studied here is the same as that considered experimentally in [77]. The bend in this case turns through 90° before a straight section is resumed. An important geometrical factor describing the bend is the ratio r/R which in this case is equal to 1/6 (where r is the pipe radius and R is the radius of curvature of the bend). The flow conditions for the present study are laminar at a Reynolds number of 700 based on the pipe diameter and inlet bulk velocity. The corresponding Dean number ($De = Re \times (r/R)^{0.5}$) is approximately equal to 286.

In the experiments, a straight inlet section $100r$ in length was used to ensure fully developed conditions at the entrance to the bend. In the present study, this inlet section length has been reduced to $10r$ and a parabolic profile for the axial velocity has been applied at the inlet boundary. The outlet boundary was placed a distance $14r$ from the exit of the bend. Zero gradient boundary conditions have been applied at the outlet for the three Cartesian velocity components, with an overall bulk correction applied to the axial component in order to ensure global mass conservation (see Section 3.6.1). The grid zipping algorithm [53], described in Section 5.6, has been used in order to integrate the mass flux over both the inlet and outlet. Pressure at the outlet has been set to zero. The pipe walls have the no-slip condition applied.

9.2.2 Computational mesh

Details of the computational mesh are reported via Figure 9.1. A total of $198,000$ cells have been employed ($106 \times 21 \times 21$ cells in the streamwise and transverse directions respectively for the Cartesian section grid, and $106 \times 26 \times 55$ cells in the streamwise, radial and azimuthal directions respectively for the polar grid). This grid is of a slightly finer resolution, with a similar distribution to that used in [1], for which grid independence has been confirmed.

All computations are performed with the QUICK convection scheme. Pressure velocity coupling is achieved via the use of the SIMPLE algorithm. The MFBI algorithm is used for inter-grid interpolations.

9.2.3 Results

Axial flow

An overview of the axial velocity in the symmetry plane is reported via Figure 9.2. Here it can be seen that there is a skewing of the axial velocity towards the outer side of the bend. There is a transport of axial momentum towards the outer wall by the secondary flow, as will be discussed presently.

Figure 9.3 shows comparison between computation and experiment for the axial velocity at the intersection of the symmetry plane with several planes normal to the axial direction. The locations of the planes are defined by the angle θ, which is defined as the bisection angle made between a plane located at the entrance of the bend (perpendicular to the streamwise direction), and the current plane under consideration (see Figure 9.1). It can be seen that the agreement between computation and experiment is very good.

Secondary flow

Figure 9.4 shows vectors of the secondary flow in several planes perpendicular to the streamwise direction. Contours of the axial velocity are superimposed so as to show all three velocity components within the same figure. Enlarged plots of the secondary flow along selected planes are also made available in Figures 9.5 and 9.6 since the overall comparative view provided in Figure 9.4 does not necessarily facilitate clarity.

It can be seen from Figure 9.4 that at the entrance to the bend ($\theta = 0°$), there is evidence of secondary flow before the bend occurs. This is due to the upstream influence of the bend communicated via the pressure, with a local minimum at the inner side of the bend and a local maximum on the outer side. This pressure gradient must be present within the bend in order to balance the centrifugal force associated with the curvature due to the bend. It must also extend a short distance upstream of the bend in order to avoid an unphysical discontinuous pressure field at the bend's entrance. This pressure gradient directs the secondary flow towards the inner side of the bend, along this favorable pressure gradient. The pressure in the symmetry plane is shown in Figure 9.7.

Between $\theta = 4.6°$ and $\theta = 11.7°$ it can be seen that the secondary flow direction is from the outer bend wall ($\alpha = 0$) toward the inner bend wall ($\alpha = \pi$), around the circumference of the pipe. The opposite is true for the flow along the symmetry plane (i.e. the flow direction here is from the inner side of the bend to the outer side). The strength of the secondary flow is seen to intensify between the $\theta = 4.6°$ and $\theta = 11.7°$ planes. It can also be seen from the figure that the axial flow shows little variation from $\theta = 4.6°$ to $\theta = 11.7°$ despite the increased intensity of the secondary flow. This is due to the high axial-momentum of the fluid at the core of the pipe, and its consequent high inertia that resists deflection by the secondary flow.

Between the $\theta = 23.4°$ and $\theta = 58.5°$ planes, however, the axial velocity contours start to show some distortion due to the momentum transport by the secondary flow, as the latter intensifies yet further. The cross flow along the symmetry plane acts to transport low momentum fluid, that was originally at the inner side of the bend, towards the centre of the pipe, and also to transport the high momentum fluid that was originally at the centre of the pipe towards the outer side. Similarly, the circumferential secondary flow acts to transport this displaced high momentum fluid, now located at the outer side of the pipe, around the circumference of the pipe, resulting in a significant distortion of the axial velocity contours which resemble 'C' or kidney bean shapes by $\theta = 58.5°$ (this can also be seen more clearly in the enlarged Figure 9.5).

It can be seen from the figure that between $\theta = 23.4°$ and $\theta = 58.5°$ the cross flow strength along the symmetry plane is decreasing with increasing streamwise distance along the pipe (i.e. with increasing θ). This is due to the fact that the secondary flow along the symmetry plane experiences a strong adverse pressure gradient, and therefore favors moving toward the top of the pipe (at $\alpha = 0.5\pi$), along a near zero pressure gradient, rather than towards the outer side of the bend. The strength of the adverse pressure gradient is also increased as the bend is traversed, thereby enhancing this effect with increasing θ. The result is the smaller tighter vortex structure that is observed at $\theta = 58.5°$, which occupies one quadrant of the pipe from $0.5\pi \leq \alpha \leq \pi$ (in addition to the symmetrical vortex, not presented, located within the quadrant $\pi \leq \alpha \leq 1.5\pi$).

By $\theta = 81.9°$, it can be seen that the extent to which the fluid moves toward the top of the pipe ($\alpha = 0.5\pi$), from the centre of the pipe via the secondary flow has increased relative to the $\theta = 58.5°$ plane. A secondary vortex is generated via the shear force imposed on the fluid by the primary vortex (this is much clearer in Figure 9.6 where an enlargement is provided). The quad-vortex structures in this plane are observed in both the computation and the experiment. The secondary vortex acts to redistribute axial momentum from the outer side of the bend towards the centre, and also reduce the strength of the circumferential flow. This results in an increase in the axial flow velocity at the centre of the pipe (relative to $\theta = 58.5°$) and a consequential decrease in the axial velocity towards the outer side of the bend. This effect is also augmented by the fact that the C-shape contours have started to close in on themselves as a result of the circumferential flow, resulting in near O-shapes. This means that the transfer of momentum by the secondary flow along the symmetry plane will start to bring this high momentum fluid back into the centre of the pipe.

The secondary flow that was observed in Figure 9.4 is generated due to the centrifugal force associated with the streamline curvature which results from the pipe bend. A pressure gradient acting towards the inner side of the bend is established in order to balance this centrifugal force. The fluid located at the core of the pipe resists bending to the same extent as the fluid that is located closer to the pipe wall, due to the high axial-momentum (and inertia) of the former. Conversely, the low axial-momentum fluid located near the top side of the pipe, caused by the viscous interaction imposed by the no slip condition, 'feels' the affect of the pressure gradient to a greater extent than the core fluid, and therefore turns with a smaller radius of curvature (relative to the core flow). A secondary flow following the circumference of the pipe therefore results, flowing from the outer side to the inner side of the bend. A resultant flow is also established acting against the pressure gradient and along the symmetry plane. This is required in order to prevent the accumulation of fluid at the inner side of the bend that would otherwise occur, which would cause a violation of the conservation of mass.

The secondary flow just described is now quantified via Figure 9.8. Experimental comparison is made against the data of [77]. In the figure, profiles of the x-velocity component are taken at three locations normal to the symmetry plane in each of the θ planes considered. It can be seen from the figure that there is generally a very good agreement between the computation and the experiment, as would be expected for a laminar flow. Experimental error is likely to play a significant role in any minor discrepancies that may be observed since further grid refinement yielded no significant change to the computed results. The error for the secondary velocity components is reported as 5% in [77]. However, in addition, due to the finite width of the two incident laser beams used to produce an LDA measurement, the velocity is not measured at a single point, but rather is averaged over an ellipsoid volume. The reported velocities are therefore an average over the volume of this ellipsoid. In [77], the measurement ellipsoid is reported to be $0.12r$ in length, with a diameter of $0.015r$. The experimenters note that this measurement volume may be sufficiently large to introduce significant error in regions of high velocity gradients, such as near the pipe wall. The largest discrepancy between computation and experiment does seem to occur near the top wall for the two off-centre profiles (particularly at

$11.7 \leq \theta \leq 39.8)$. The long length of the measurement volume would necessarily incorporate the very low velocity fluid immediately adjacent to the pipe wall at these probe locations, thus causing an underestimate in the secondary velocity here, as is observed. Whether or not this is the reason behind the discrepancy is not clear, but one should not lose sight of the fact that the discrepancy is very minor.

9.2.4 Conclusion

Comparisons between computation and experiment for the flow through a pipe subjected to a 90° bend have shown that generally a very good match has been achieved. The secondary flow induced by the pipe bend has been shown to have been convected well across the overlapping grid interface. The results found in this section will be important for the carotid bifurcation flow which shall now be considered.

9.3 The Carotid Bifurcation

In human anatomy, there are two common carotid arteries; one on the left of the neck, and the other mirroring on the right. It is these two arteries that provide blood to the head. Within the upper neck, the common carotid artery divides into the internal and external carotid arteries, and it is this division that is described as the carotid bifurcation. Figure 9.9 illustrates a typical location of the carotid bifurcation.

The deposition of plaques of fatty material onto the inner walls of an artery is a common cause of arteriosclerotic vascular disease. It is well recognised that there is a tendency for these deposits to occur within the region of a bifurcation [78, 79], for reasons that are now outlined.

In bifurcation regions, injury to the endothelial lining can occur due to a locally high wall shear stress, or due to a periodic change in stress' direction of action throughout the cardiac cycle [78]. One pivotal role of the endothelial lining is to facilitate blood flow by providing an smooth inner arterial wall that impedes clotting. Any injuries to this lining may impair this function, thereby leading to possible arteriosclerotic vascular disease.

As well as the mechanical damage just described, arteriosclerotic vascular disease also has a tendency to form in regions of low wall shear stress. In this case, the ability of the blood flow to advect fatty deposits away from the wall is inhibited, and hence a build-up may occur.

The geometric features posed by a bifurcation naturally induce regions of both locally high and low wall shear stresses (relative to a typical non-branching artery's wall stresses), and it is for this reason that arteriosclerotic vascular disease is typically encountered around a bifurcation.

Attaining a detailed understanding of the complex flow physics in the bifurcation region may lead to advanced medical diagnosis techniques whereby imaging downstream of a branching may be used to determine the presence of a possible upstream blockage (e.g. by matching the obtained flow profiles with those computed for both diseased and non-diseased arteries) [80]. Furthermore, a detailed understanding of a bifurcation flow physics is important for the design of artificial blood vessels such as those in use in experimental research [81], or even transplant surgery [82].

9.3.1 Computational model

Blood flow in the Carotid artery bifurcation is particularly complicated due to pulsatory nature of the flow, the non-Newtonian nature of the fluid, the elastic nature of the arterial walls, and the complex geometry. However, several simplifying assumptions have been employed here. In the present study, a steady flow is considered with a Newtonian working fluid and rigid walls. Some justification for these simplifications shall be offered. Firstly, in the experimental work of Merrill and Pelletier, [83], it has shown that blood exhibits a Newtonian behaviour for a shear stress of $0.15 N/m^2$ and above. The shear stress throughout the vast majority of the bifurcation region exceeds this value, and hence non-Newtonian effects are expected to play only a small role in this study. The justification of using a steady model and rigid walls lies in the fact that the carotid artery experiences a relatively constant flow for approximately two-thirds of the cardiac cycle [84]. During this period, the arterial walls will cease to distend, and the flow will start to approach a steady state. Hence the use of a steady state model with rigid walls is not wholly inappropriate. In any case, the main objective of the present study is to demonstrate the use of the overset method in dealing with complex geometries rather than to necessarily provide a physically realistic model of the artery. Furthermore, the simple model considered here provides an important first step toward a more complete model including some (or all) of the additional physics neglected here.

Note that in the experimental data provided by References [80] and [84], as performed by Bharadvaj *et. al.*, the same approximations as those outlined above were used, hence a direct comparison is possible.

Computations have been conducted with the QUICK convection scheme and the MFBI interpolation scheme for inter-grid communication transfer. Pressure velocity coupling is achieved via the SIMPLE algorithm.

9.3.2 Boundary conditions and geometry

Throughout the cardiac cycle, instantaneous Reynolds numbers ranging from 380 to 1200 are typically encountered, [80]. Here, (and in Reference [84]), Reynolds numbers of 400, 800 and 1200 have been considered (based on the common carotid artery diameter, and the bulk velocity at the inlet of the common carotid). The flow is laminar over the full range of Reynolds numbers considered, and hence no turbulence modelling is required.

At the inlet, a parabolic velocity profile has been applied, with a bulk velocity consistent with the above mentioned Reynolds numbers. This inlet profile is consistent with fully developed conditions for laminar pipe flows. In the experimental data provided by [84], the inlet pipe was 88 tube diameters in length, hence the fully developed assumption is applicable. In a real human carotid artery, the pulsatory nature of the flow induces periods throughout the cardiac cycle where a non-parabolic inlet profile occurs. The square of the frequency parameter for the pulsitile flow, defined as $\rho\omega R^2/\mu$, typically lies within the range of $25 - 35$, and hence may ordinarily be too high for a fully developed state to be approached at any point of the cycle. However, Doppler ultrasound velocimeter measurements, performed on healthy human subjects, as reported in [84], confirm that a parabolic profile is maintained over approximately two-thirds of the cardiac cycle, with a reasonable steady forward flow throughout this period (hence a steady state approximation being justifiable as a first approximation).

At both outlets, a zero gradient for all velocity components has been assumed. The bulk pressure correction algorithm, described in Section 3.6.1, has been employed in order to satisfy global mass conservation. Since there are two outlets, it is necessary to specify the flow ratio through the external to internal carotid arteries a priori. In reality, such a ratio is clearly established naturally from the presence of any downstream flow restrictions. However, since such restrictions are not included in the computational domain, the ratio must be set here. A ratio of 70 : 30 through the internal and external arteries respectively is used, as this is typical of a healthy human [84].

The geometry considered here is as close as is possible to that considered experimentally by Bharadvaj *et. al.*, [80] (note there is some ambiguity in the precise geometry in [80], so an exact replica could bot be attained). In [80], a representative average bifurcation has been generated from Fifty-Seven angiograms of 22 adult human subjects, supplemented by 67 angiograms from 50 children. Ideally, adult-only data would have been used, however, insufficient data was available from disease free adults. Data from diseased subjects was not used due to the difficulties in identifying where healthy tissue terminated, and diseased tissue was present, hence such datasets did not contribute to any of the measurements, including those away from the diseased area.

Since the angiograms are two-dimensional, certain assumptions are required in order to generate a three dimensional geometry. The first assumption employed in [80] is that the cross section of all three arteries (i.e. the common, interior and exterior arteries) are circular. The second assumption employed is that all three arteries lie in the same plane. That is, the axis of revolution of all three arteries are coplanar. Further assumptions have been made relating to the quality and validity of the angiogram data. Namely, that the carotid artery lies parallel to the longitudinal axis of the neck, and hence the plane of the angiogram (which is ensured to be parallel to longitudinal axis of the neck) is the same as that of the carotid artery. Finally, it has been assumed that all regions of a given angiogram are magnified equally [80].

Due to the different dimensions involved in the arteries from different people (particularly the differences in size between adult and child arteries), all data was scaled to yield a unit diameter for the common carotid artery upstream of the bifurcation. All measurements were then taken with this scale factor applied. Table 9.1 shows the average values of the measurements taken from the angiogram data, with Figure 9.10 showing the location of each measurement.

9.3.3 Computational Grid

A total of five overset sub-grids have been used in the present study to delineate the overall computational domain. The common carotid artery and the internal carotid are both depicted via the use of the same two grids, with the grids bending at the bifurcation in order to cover the internal carotid portion of the domain. An O-grid covers the outer section of the artery, while a separate square-section grid covers the core of the artery. The external carotid artery uses a further two grids, again covering the interior and exterior of the artery via the use of a separate O-grid and square-section grid. A final fifth overset sub-grid is then used at the junction between the external and common carotid arteries. This grid is referred to as a collar grid. The reason this collar grid is required is that the surface cells of the common artery wall will not conform to the shape presented by the intersection between the external and common artery walls. Computational cells of the common carotid wall at the intersection will, in general, be partly overlapping the external artery, and partly non-overlapping. The application of computational boundary conditions in this case clearly poses a problem; should they be treated as interpolation cells with the external artery used as a donor, or should they be treated as part of the common carotid wall? The only valid answer is neither, and instead the cells at the interface should be cut, leaving a hole. In order to fill this hole, a collar grid is required, which is used to transfer information from the common carotid to the external artery. This sub-grid is referred to as a collar grid since it wraps around the external artery like a collar. An overview of the grids used is reported in Figure 9.11.

In previous overset computational studies considering a branching geometry, such as [85], collar grids have not been used. Instead, grid arrangements such as that of Figure 9.12 are used. In this case, one grid ‘sees’ a different intersection shape and area than that of the branching grids at the same location, thereby assuring different solutions on the different grids at the same point in the

Location number shown in Figure 9.10	Mean dimensionless value
1	1.04
2	1.11
3	0.91
4	0.72
5	2.14
6	0.69
7	0.69
8	0.69
9	0.58
10	25.1°
11	25.4°
12	-0.30
13	1.00

Table 9.1: Table showing artery dimensions used.

overlap region. This seems unsatisfactory to the present author, and hence a collar grid is used to resolve the issue.

Initial and refined computational grids comprising of approximately $350,000$ and $700,000$ cells respectively have been employed. Grid sensitivity tests were carried out with flow conditions set to the highest Reynolds number under consideration. The lower prominence of viscous forces at higher Reynolds number generally leads to steeper velocity gradients (since the solution is not 'smeared out' to the same extent as for a lower Reynolds number), and therefore would require a finer mesh for grid independence. Grid dependence has been evaluated via analysis of the wall shear stress. This criterion has been selected since it can be readily shown (via a Taylor's series expansion) that the one-sided difference approximation used in order to calculate the wall shear stress is first order accurate in space, while the underlying flow solver is second order accurate. It would therefore be expected that the wall shear stress be particularly sensitive to grid refinement. If the correct wall shear stress is predicted, the velocity field is likely to be satisfactory. On the other hand, a broadly satisfactory velocity field will not necessarily lead to a satisfactory wall shear stress prediction.

Figure 9.13 shows contours of the wall shear stress in the region of the bifurcation for the two grid densities. It can be seen that, while there are small variations between the two solutions, the magnitude of the variations is sufficiently small to assume grid independence by $700,000$ cells. Figure 9.14 further quantifies the difference in the wall shear stress for the two grid resolutions by plotting the wall shear stress along the wall connecting the common and internal carotid arteries at the symmetry plane. It can be seen from the figure that there is virtually no difference between the two solutions, thereby suggesting that the use of the initial grid is justifiable. The maximum discrepancy between the two solutions is found to be around 2.3% of the theoretical Poiseuille value for a pipe diameter and flow rate equal to that at the inlet.

In addition to the main computational grids on which the underlying flow solver operates, additional zipped grids have been automatically generated in order to apply the bulk correction algorithm. Figure 9.15 illustrates these surface grids for the inlet section and the internal artery outlet. A similar zipped grid to that of the internal artery outlet is also generated for the external artery, but is not presented here.

9.3.4 Results

The flow in the carotid bifurcation bears many similarities to the pipe bend flow considered earlier. However, there are three very important differences that have a marked effect on the flow physics; firstly the branching of common carotid artery (CCA) into the internal and external carotid arteries (ICA and ECA respectively) causes the flow to split at an apex, and the flow division ratio can have a marked effect on the flow field. Secondly, the arteries are of non-uniform diameter, thereby inducing flow acceleration or deceleration. This is particularly significant at the entrance to the ICA where a sinus is present (i.e. a region of relatively tight longitudinal curvature as the local arterial diameter increases then decreased rapidly). Finally, and perhaps most significantly, there is flow separation in the sinus across the range of Reynolds numbers considered.

Since the ICA branch of the bifurcation poses many more interesting flow features than the ECA, primarily due to flow separation and the variable diameter posed by the sinus, the following analysis shall focus almost exclusively on the ICA branch. The underlying flow physics in the ECA is much the same as that of the ICA, but without the additional interesting features. Furthermore, the experimental data reported in [84] typically is non-existent for the ECA branch of the bifurcation, hence comparison is not possible. One notable exception to this is for the wall shear stress along the common-external wall, which is available experimentally, and hence shall be discussed.

For use in the analysis that follows, several planes are defined. Planes through the $\underline{\text{C}}$ommon $\underline{\text{C}}$arotid, taken perpendicular to the axial direction, are referred to as CC1-CC4. Similarly, planes taken through the $\underline{\text{S}}$inus $\underline{\text{S}}$ection, again normal to the axial direction, are referred to SS1-SS6. The location of each of these planes in defined in Figure 9.16. In addition, the left arterial wall in the figure is referred to as the common-internal wall since it is common to both the CCA and the ICA;

similarly the right hand wall will be referred to as the common-external wall. In the ICA (the left branch), the sinus region can be recognised by observing the expanding and then contracting diameter of the artery.

Overview of main flow features

Figure 9.17 shows an overview of the velocity vectors in the symmetry plane for the $Re = 400$ case. The figure demonstrates many of the main flow features. Separation in the sinus is suggested via the essentially zero velocity gradient near to the common-internal wall. While the qualitative data presented in the figure may be insufficient to conclusively state the flow is separated, it will be seen that this is indeed the case via analysis of the common-internal wall shear stress shortly. Skewing of the axial velocity, away from the common-internal wall for the ICA, and away from the common-external wall for the ECA, is also evident. This result is similar to that of the pipe bend flow considered earlier and indeed the secondary flow plays a major role in such skewing. In the figure, both computed and experimentally obtained vectors are shown. It can be seen that there is generally a good qualitative agreement between the two sets of results. The axial velocity in the symmetry plane is also illustrated in Figure 9.18 via the use of coloured contours.

An overview of the secondary flow at planes through the sinus section is presented in Figure 9.19. The figure also provides an overview of the pressure field in the symmetry plane. From the figure it can be seen that at section SS1, located at the entrance to the ICA's sinus, there is a strong secondary flow with features very similar to those found for the flow through a pipe subjected to a bend (Section 9.2). The flow is directed around the circumference of the artery towards the low pressure region located on the inner side of the bend, while a cross flow in the opposite direction along the symmetry plane is established in order to balance the former. In the carotid bifurcation flow however, in contrast to the 90° bend flow, this effect is augmented by the fact that there is a stagnation point at the apex where the internal and external arteries meet. The result is a high pressure gradient and a particularly strong secondary flow, and consequential distortion of the axial velocity, with skewing towards the apex evident. Also, due to the longitudinal curvature of the sinus, there is a relatively high pressure region in the vicinity of the wall downstream of the maximum diameter (near point C in the figure), thereby further affecting the secondary flow.

It is worth noting that the separation region in the sinus is not a closed region of recirculation (fluid is not trapped here). 3D effects play an important role in transporting fresh fluid into the separation region as follows. The circumferential secondary flow around the perimeter of the arterial wall (generated via much the same mechanisms as for the pipe bend flow) acts to transfer fresh fluid into the separation region. This fluid then experiences a negative axial acceleration associated with the adverse axial pressure gradient in the separation region (the reasons for the presence of this adverse pressure gradient will be discussed presently). The secondary flow along the symmetry plane then acts to transport this reversed flow back towards the centre of the ICA, where viscous interaction with the adjacent fluid retards the reversed flow and subsequently carries it back along a positive axial direction towards the outlet (i.e. entrainment of the fluid in the separation region by the axial flow occurs). Figure 9.20 illustrates this effect. This 3D dominated result is very important. Without it, recirculating fluid would be trapped in the separation region. The ability of the fluid to transport fatty deposits away from the sinus wall would therefore be impaired, and evolution would likely have been forced to take a different path.

Axial flow

Figure 9.21 shows profiles of the axial velocity through the CCA. The profile locations are defined by the intersection of the symmetry plane with planes CC1 to CC4, and are indicated in the figure (see also Figure 9.16 where the plane locations have been defined). It can be seen from the figure that the first two profiles show an essentially parabolic shape, indicative of a fully developed laminar flow. This suggests little upstream influence of the bifurcation, particularly given plane CC2 is only half an inlet diameter from the start of the bifurcation. By plane CC3, located at the start of the bifurcation, some evidence of the onset of flow separation along the common-internal wall can be seen. The axial velocity in the vicinity of this wall is slightly decelerated relative to the CC2 profile (note the common-internal wall is located at $r/D = -0.5$). It can also be seen that the axial velocity in the vicinity of the common-external wall is decelerated in the experimental results. This however is not captured in the computation. There is also some discrepancy in the peak value of the axial velocity at CC3, which is consistent with the former discrepancy. Mass conservation dictates that the peak velocity for the experimental data should be higher since the near wall flow velocity is reduced. In any case, the discrepancies are minor, and might be attributable to geometrical ambiguity: The discrepancy suggests that the precise geometrical curvature at the plane CC3, along the common-external wall, has not been accurately depicted. Based on the result seen for CC3, the experimental ECA appears to branch off slightly further upstream than that of the computational geometry. The highly complex geometry of the bifurcation is not fully specified in [80], and hence some assumptions and approximations have necessarily been made. This is particularly true for the wall curvature in the vicinity of the bifurcation's origin, where the three arteries merge, and hence the ambiguity is particularly prominent at CC3. Experimental error may also play some role in the discrepancy. While computational error may also play a role, this is not expected to be significant. Truncation error has been effectively managed via the grid dependence tests reported earlier, and since the flow is laminar, confidence in the computational accuracy is justifiable. In any case, the differences are minor.

By CC4, separation is confirmed along the common-internal wall (since the axial velocity in the vicinity of the wall is seen to be negative). The peak flow at CC4 is decelerated slightly relative to CC3, since the sectional area of the former is increased due to the bifurcation.

It can be seen from Figure 9.21 that the agreement between the computed profiles and the experimental data is very good, with the possible exception of CC3 where some minor discrepancies have been discussed. The good agreement at CC4 is particularly encouraging since this suggests that the correct separation point on the common-internal wall has been captured. It will be seen that the location of the separation point has a major affect on downstream flow development.

Figure 9.22 shows profiles of the axial velocity at the intersection of the symmetry plane with planes SS1-SS6. The Reynolds number is again equal to 400. For all the profiles, skewing of the axial velocity away from the common-internal wall is apparent. This is due to the transport of axial momentum by the secondary flow. Between SS1 and SS3 this effect seems to have increased, which is due to the augmentation of the secondary flow between these planes, caused by the curvature of the sinus wall. The curvature of the separation region also plays a role since this alters the effective geometry.

It can also be seen from Figure 9.22 that between planes SS1 and SS3, there is some reversed flow in the vicinity of the common-internal wall. By SS4, reattachment is observed, indicated by a positive axial flow across the diameter of the artery. Downstream of SS4, there is rapid flow acceleration over the left portion of the artery which is due to the favorable axial pressure gradient (generated via a combination of high pressure at the reattachment point, in addition to the rapid pressure drop associated with the arterial diameter contraction prior to conclusion of the sinus, as will be discussed presently).

The agreement between computation and experiment is generally good. The largest discrepancy appears to be at plane SS5, over the left portion of the artery. Plane SS5 is located in the region of rapid flow acceleration downstream of the reattachment point. The precise location of the reattachment point therefore plays a significant role in the axial velocity here. Since the reattachment point is particularly sensitive to the geometrical wall curvature, accurate predictions of the velocity profile in the vicinity of flow reattachment cannot be expected unless a precise replica of the geometry is attained. It is possible that reattachment is predicted slightly too early in the computation, hence the computed axial velocity may have had spurious additional recovery time, explaining the over-prediction. Another possibility presents itself by noting that the axial velocity at SS5 is over-predicted over the entire diameter, suggesting that a more likely reason for the discrepancy is due to an insufficient arterial diameter at this plane (while this is suggested, it is not confirmed since the axial flow could have been underestimated at locations away from the symmetry plane). The axial velocity would necessarily be accelerated in this scenario in order to satisfy mass conservation. The arterial diameter is specified in [80] at locations $0.92D$ upstream of SS5, and $0.31D$ downstream of the plane. Between these specified diameters, interpolation is necessary and hence the diameter at SS5 is likely to be slightly different to that of the experiment (a cubic spline interpolation has been used in the present study). Again, it is seen that geometrical ambiguity is likely to be the reason behind any discrepancy. It should be emphasised however that the discrepancy is fairly minor.

Figure 9.23 shows profiles of the axial velocity, taken normal to the symmetry plane, at SS1-SS6, through the artery's diameter. The flow Reynolds number, based on inlet conditions, is equal to 800. Note that the flow Reynolds number is different between figures 9.22 and 9.23 due to the availability of experimental data, however the general flow features are the same. Here it can be seen that there is a peak in the axial velocity at the top and bottom of the artery for all profiles but SS1. This feature is generated via the transport of axial momentum by the secondary flow. Similar features have been observed for the pipe bend flow. The agreement between computation and experiment near to the arterial wall is good, suggesting that the secondary flow is accurately predicted. Around the centre of the artery however, the discrepancy is, in some cases, significant. This is particularly true at SS1 and SS2, where the axial flow at the centre is badly underestimated (by around 25%). The axial velocity has been seen to be shifted away from the common-internal wall due to the effective obstruction presented by the separated flow and due to the secondary flow momentum transport. Due to the sharp cross-stream velocity gradient in this region (in the direction parallel to the symmetry plane), a small discrepancy in the shape of the recirculation region could have a large affect on the axial velocity through the vertical profiles considered, since the effective blockage by the reversed flow will be different. This is supported somewhat via the previous result of Figure 9.22 in which the separation region is seen to extend slightly too far towards the centre of the sinus relative to the experimental data, and hence would shift the axial momentum yet further from the centre of the artery where the present profile is taken (this comparison is valid despite the difference in Reynolds number between figures 9.22 and 9.23 since the qualitative flow features remain the same).

At planes SS3-SS5 in Figure 9.23, it can be seen that there is now a slight over-prediction in the centreline velocity. For these profiles at least, the discrepancy is fairly minor. Again, the slight discrepancy is likely to be due to the predicted shape of the separation region, which is strongly influenced by the wall curvature. Figure 9.22 gives some suggestion that the separation region at SS3 does not extend sufficiently far towards the centre of the artery. This is consistent with the over-prediction observed at this plane for the vertical profile under consideration in Figure 9.23.

Secondary flow

Figure 9.24 shows profiles of the secondary velocity component that is parallel to the symmetry plane, at sections through the internal carotid artery. The Reynolds number in this instance is equal to 800. It can be seen that for sections SS1 and SS2, for the centre profile through the diameter, the secondary flow bears much the same flow features as for the pipe bend (the mechanism by which this secondary flow is generated has been discussed for the 90° bend pipe flow in Section 9.2). It can be seen that for the leftmost profiles of SS1 and SS2, the secondary flow is diminished. This is due to separation of the secondary flow as the fluid is directed into the core of the separation region due to the low pressure that is present there.

At planes SS4 to SS6, it can be seen that there is acceleration of the secondary flow close to the symmetry plane for the centre and left profiles. Due to the axial curvature of the sinus, a low pressure region is present at its centre, and a relatively high pressure region is present around the wall circumference. This pressure gradient accelerates the secondary flow towards the centre of the pipe, thereby enhancing the cross-flow component for the left profile and centre profiles, while also causing the cross-flow for the right profile to be in the opposite direction to that which would be expected for a uniform diameter pipe.

The agreement between the computed results and the experimental data is generally quite good.

Any small discrepancies again are likely to be attributable to small differences in the specific geometry of the bifurcation. Experimental accuracy of the secondary flow is also questionable, particularly given the experimental data displays asymmetry where there should be a symmetrical solution (see for example the plane SS1 in Figure 9.24 where there is secondary flow near the top wall, but none at the bottom wall).

Wall shear stress

As the Reynolds number is increased from $Re = 400$ to $Re = 1200$ inertial effects become more prominent and the flow resists following the curvature of the geometry to a greater extent. The effect of the increase in Reynolds number is particularly noticeable in the internal carotid artery's sinus region. Here, the curvature provided by the bend at the bifurcation, plus the additional curvature provided by the sinus wall, is sufficient for flow separation to ensue. The size of the recirculation region and the strength of the back-flow both increase with increasing Reynolds number. This effect can be realised via analysis of the wall shear stress.

Figures 9.25 and 9.26 show the wall shear stress along the common-internal carotid wall for $Re = 400$ and $Re = 1200$ respectively. These figures demonstrate a good agreement between computation and experiment for the wall shear stress at the symmetry plane of the common-internal wall. Figure 9.27 shows the same plots again superimposed, with the addition of a $Re = 800$ case (for which experimental data was not available), in order to facilitate a direct comparison. It can be seen from this latter figure that around $-1 \leq y/D \leq -0.25$ there is a moderate increase in the wall shear stress. This is due to a combination of two factors. Firstly, only 30% of the flow is directed through the ECA. The axial flow through the CCA therefore experiences an adverse pressure gradient on the ECA side, so as to ensure the correct flow split at the bifurcation. A slight skewing of the axial velocity towards the common-internal wall therefore results, contributing toward the increase in the wall shear stress that is observed. In addition, the near wall flow is accelerated upstream of the bend due to the low pressure region associated with the bend's curvature. This flow acceleration close to the wall also contributes to the local peak in wall shear stress.

Immediately following the local peak in wall shear stress just described, the wall curvature due to the bifurcation commences, and the wall shear stress along the common-internal wall is seen to decrease rapidly. This is due to the secondary flow associated with this curvature, causing fluid to be transported along the symmetry plane towards the centre of the CCA. The result is a low velocity gradient near the wall, and hence a low wall shear stress. This effect continues, and eventually flow separation occurs, indicated by zero wall shear stress. It can be seen that the point of separation is fairly consistent across the Reynolds number range. There is however a slight tendency for separation to be postponed at lower Reynolds numbers, as would be expected due to the lower inertial affects. At increasing Reynolds number the secondary flow is stronger, thereby enhancing the transport of fluid away from the wall, causing separation to occur sooner.

At $y/D \approx 1.5$ it can be seen that there is a significant negative peak in the wall shear stress for the highest two Reynolds numbers under consideration. This is due to the reversed flow that occurs in the separation region. As the sinus expands, the additional wall curvature the expansion provides augments the secondary flow, causing additional fluid to be transported around its circumference, into the separation region. This fluid, once deposited within the separation region, travels in the negative axial direction due to the adverse pressure gradient that it is subjected to, thereby causing the large negative peak in wall shear stress. Note that the adverse pressure gradient is associated with the sharp streamline curvature at the reattachment point, as will be described in greater detail presently. It can be seen that the $Re = 400$ case does not display the negative peak in wall shear stress. This is due to the fact that the adverse pressure gradient associated with reattachment is significantly lower for the $Re = 400$ case (the reason for this will also be discussed).

Downstream of the minimum wall shear stress point, flow acceleration occurs due to the sinus contraction, causing steeper near-wall velocity gradients, and hence the rapid increase in the wall shear stress that is seen at all three Reynolds numbers by $y/D \approx 1.5$.

The flow a short distance prior to conclusion of the sinus and following reattachment follows the curvature of the wall, and hence has a cross-flow component directed towards the centre of the artery. The inertia of the fluid causes this cross-flow velocity component to remain for a short distance downstream of the end of the sinus, where a straight section is resumed, explaining the reduction in wall shear stress that is observed downstream of $y/D \approx 2.5$.

At around $y/D = 4$, it can be seen that the wall shear stress starts to increase, as viscosity starts to dissipate the secondary flow, and hence the transport of axial momentum away from the common-internal wall is reduced.

Figure 9.28 shows the wall shear stress at the symmetry plane along the common-external wall. Here it can be seen that there is a local minima in the wall shear stress immediately upstream of the junction between the CCA and the ECA, at around $y/D \approx -0.3$. This again is probably due to the low flow rate through the ECA, causing a high pressure on the ECA side of the CCA, leading to a shift in the peak velocity towards the common-internal wall (and hence away from the common-external wall currently under consideration). At the junction (around $y/D \approx -0.2$), it can be seen that there is a local maxima in the wall stress which is associated with the fluid's acceleration around the sharp corner. In the ECA, it can be seen that the wall shear stress reaches a minimum at $y/D \approx 0.5$. The low wall shear stress here is associated with the secondary flow that acts to shift the axial flow away from the common-external wall. Downstream of $y/D \approx 0.5$, the wall shear stress starts to increase due to the dissipation of secondary flow via the action of viscosity. The axial velocity peak therefore starts to shift from the opposite wall towards the common-external wall.

It can be seen that the agreement between the computation and experimental data is not particularly satisfactory for the wall shear stress along the common-external wall. While the location of the minimum wall shear stress seems to have been captured, the rate of recovery following this minimum is over-predicted in the computation. The reason for this discrepancy is almost certainly due to differences in the geometry between computation and experiment. There is significant ambiguity in the geometry of the ECA since the primary focus of the study is around the sinus region. Specifically, while three diameters are specified via dimensions 7, 8 and 9 in Figure 9.10 and Table 9.1, only the axial location of dimension 7 can be inferred (being at the entrance to the ECA), while the others have been estimated from the schematic geometry figure provided.

Pressure on the common-internal wall

Figure 9.29 shows the pressure coefficient along the common-internal wall at the symmetry plane for the three Reynolds numbers under consideration. Note that the reference pressure location has been taken as the point on the common-internal wall, at the symmetry plane, where the curvature due to the bifurcation begins, hence all three Reynolds number solutions show the same pressure at this point. It can be seen from the figure that around $y/D < -1$ there is a constant negative pressure gradient that is associated with driving the fully developed flow against frictional losses. The gradient of this pressure drop decreases with increasing Reynolds number since viscous losses become less significant at higher Reynolds number. A short distance upstream of the start of the bifurcation, at $y/D \approx -0.5$, it can be seen that for all three Reynolds numbers, the rate of pressure decrease is enhanced as the upstream influence of the bifurcation curvature has an effect. By $y/D \approx 0$ however, it can be seen that the pressure starts to increase as the flow is decelerated due to the increased cross sectional area associated with the bifurcation. It is this adverse pressure gradient that causes the separation seen in Figure 9.27.

It can be seen that a short distance downstream of the maximum sinus diameter there is a peak in the pressure associated with flow reattachment. It is this peak that is largely responsible for the reversed flow in the separation region. The peak pressure here is due to the sharp streamline curvature at the reattachment point: Fluid at the inlet of the CCA, near the upper wall (e.g. $\alpha \approx 0.25\pi$, $r/D \approx 0.4$), is eventually transported via the secondary flow around the circumference of the sinus wall towards the reattachment point. Before reattachment, this fluid is decelerated rapidly in the axial direction via viscous interaction with the reversed flow in the separation region. As a result, at the reattachment point the transported fluid has no axial component of velocity, and must therefore turn with a particularly small radius of curvature as the flow around the circumference is redirected back along the symmetry plane via the secondary flow. This sharp streamline curvature is responsible for the peak in pressure observed, with the latter being required in order to balance the centrifugal force associated with the curvature. Note that this effect is very small at the lowest Reynolds number considered since in this case the secondary flow is substantially weaker, and hence the centrifugal force associated with the streamline curvature is substantially lower.

Yet further downstream of the reattachment point, there is a rapid decrease in the pressure which is due to the flow acceleration caused by the decrease in cross sectional area, in addition to the pressure recovery from the peak just described. This negative pressure gradient continues until the outlet, so as to drive the fluid against frictional losses. The negative pressure gradient downstream of reattachment is seen to be substantially steeper than that of the CCA. The effective Reynolds number of the flow through the ICA is less that half that of the CCA (based on 70% of the inlet bulk velocity and the final arterial diameter of $0.69D$), hence viscous losses are more significant.

The large peak in the pressure on the common-internal wall that has just been described, a short distance downstream of the maximum sinus diameter, is also responsible for the accelerated secondary flow along the symmetry plane that was observed earlier (shown in Figure 9.24) for SS4-SS6, since there is now a strong pressure gradient acting towards the centre of the artery. It also contributes to the rapid flow acceleration following reattachment that was reported in Figure 9.22.

9.3.5 Conclusion

The flow through a generic carotid bifurcation geometry has been computed. It has been seen that the computed results are generally satisfactory. Since the flow is laminar, and grid independence has been confirmed, computational error is not expected to be at the root of any discrepancy. Instead, geometrical ambiguity is likely to be the main cause of such discrepancies.

The overset grid method has proven very useful in this case since a block-structured grid would be extremely difficult to generate. The use of a collar grid has been demonstrated to work well in facilitating information transfer at the intersection of two wall bounded domains, where the hole cutting algorithm would otherwise be unable to apply boundary conditions for cells at the intersection. The grid zipping algorithm, originally devised for computing sectional lift coefficients for overset wing arrangements, has been applied in a novel way in order to enforce global mass conservation. This has been shown to work well, and can deal with multiple outlets giving the correct flow split at the bifurcation. The MFBI algorithm has also been shown to perform well for the present case, allowing grid independence to be attained by $350,000$ cells. When using a linear interpolation, evidence of grid dependence was observed.

9.4 Figures

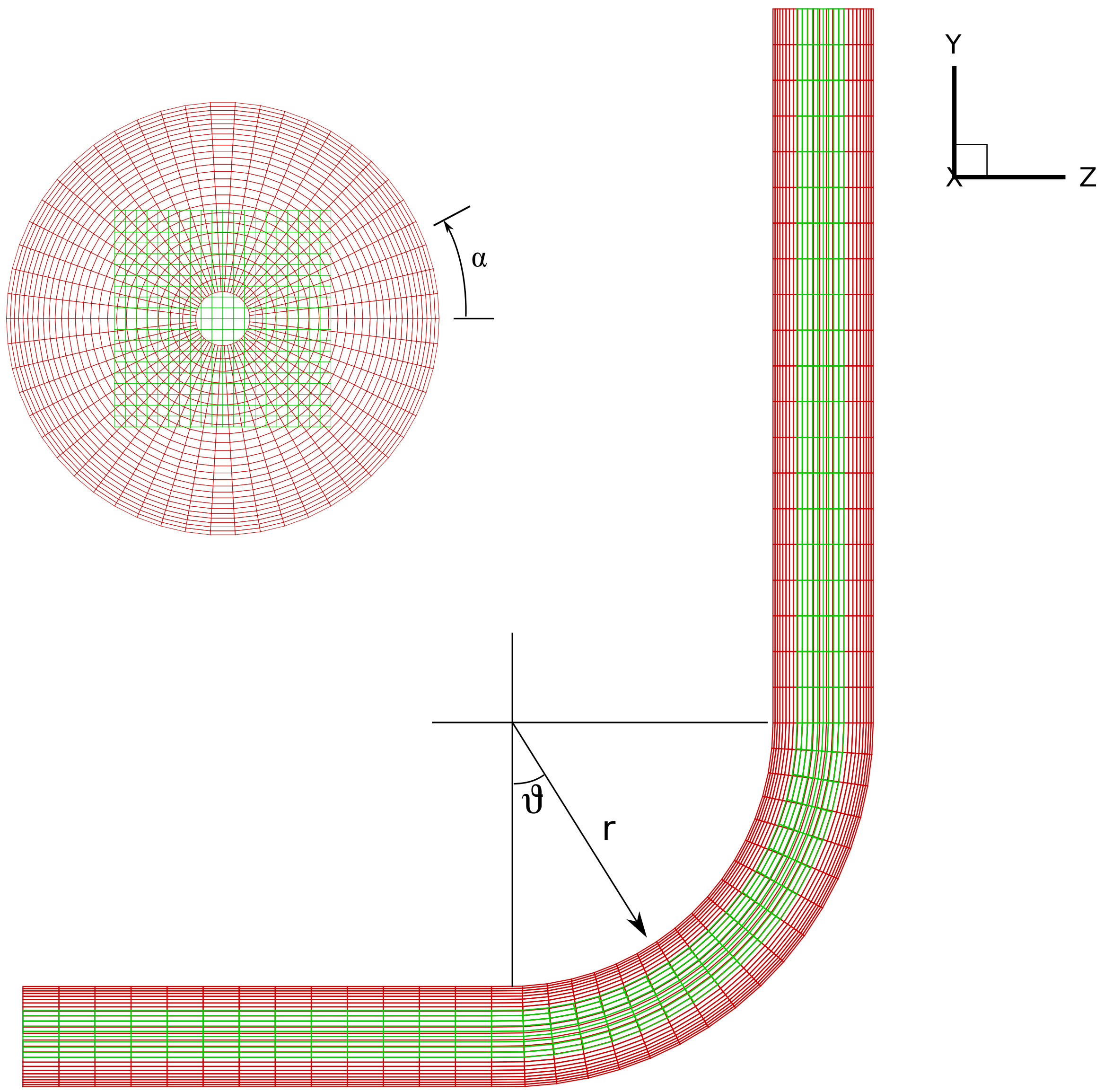


Figure 9.1: Computational mesh

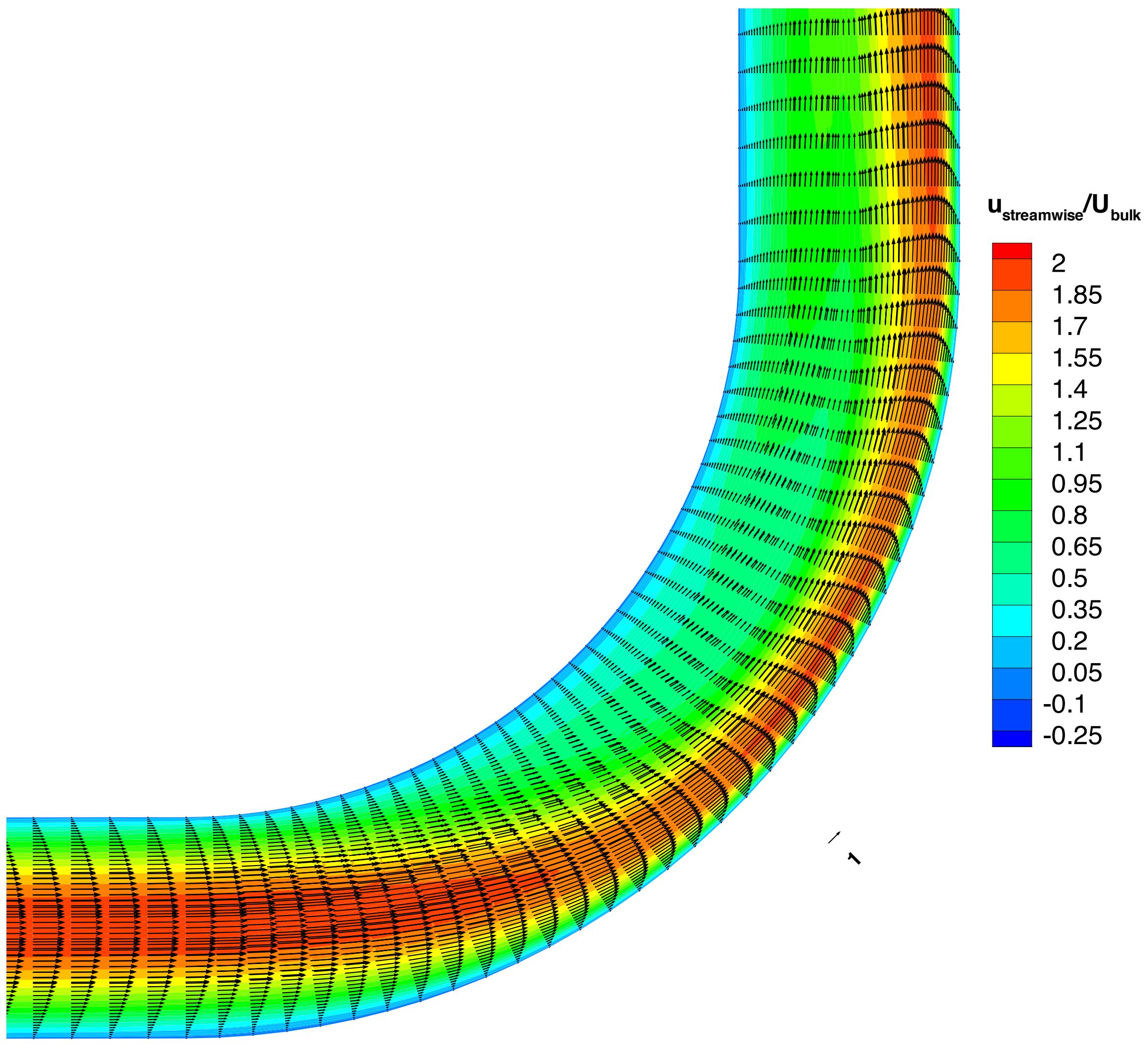


Figure 9.2: Contours of axial velocity. Vectors in symmetry plane.

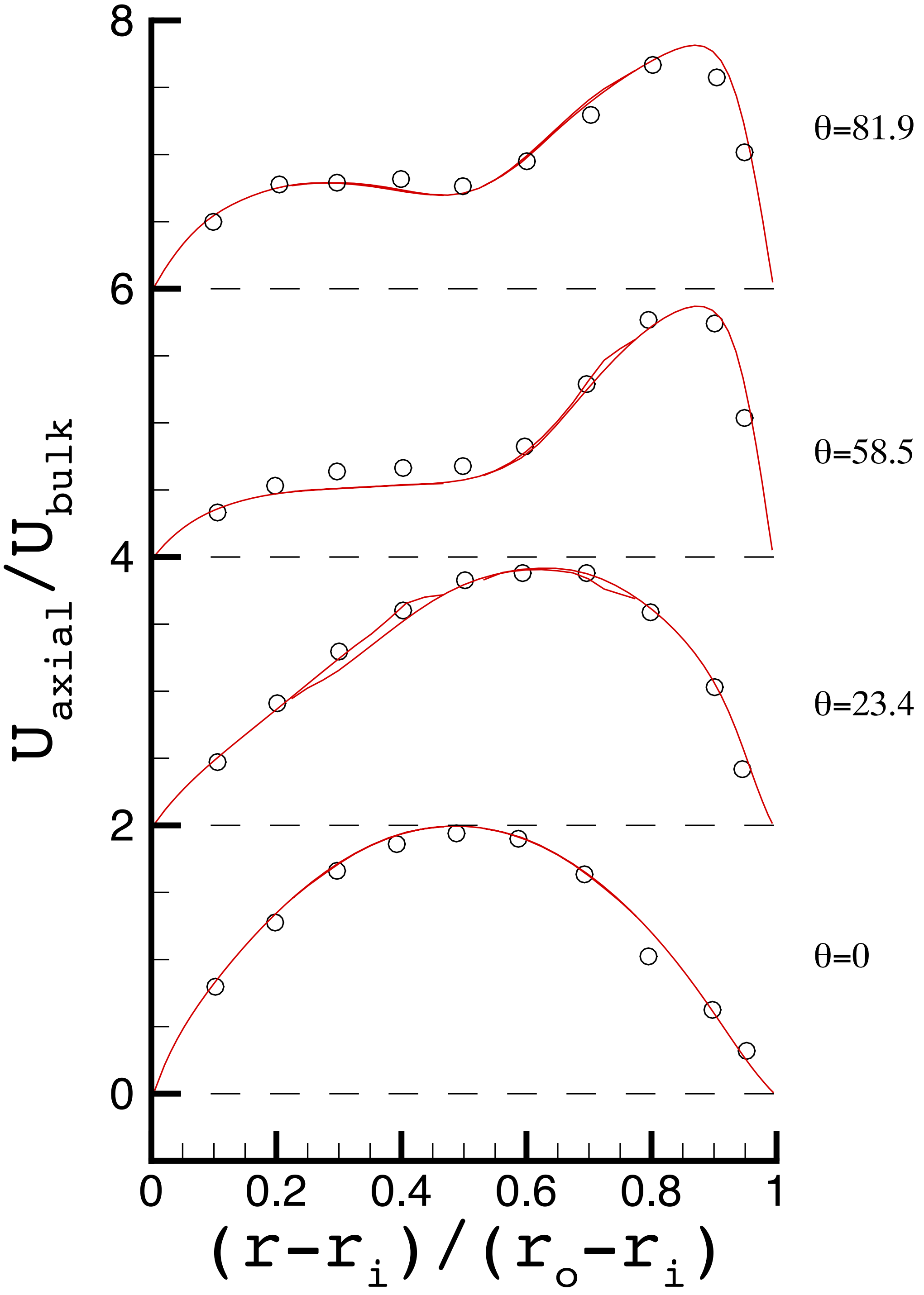


Figure 9.3: Axial velocity at the symmetry plane at four streamwise locations.

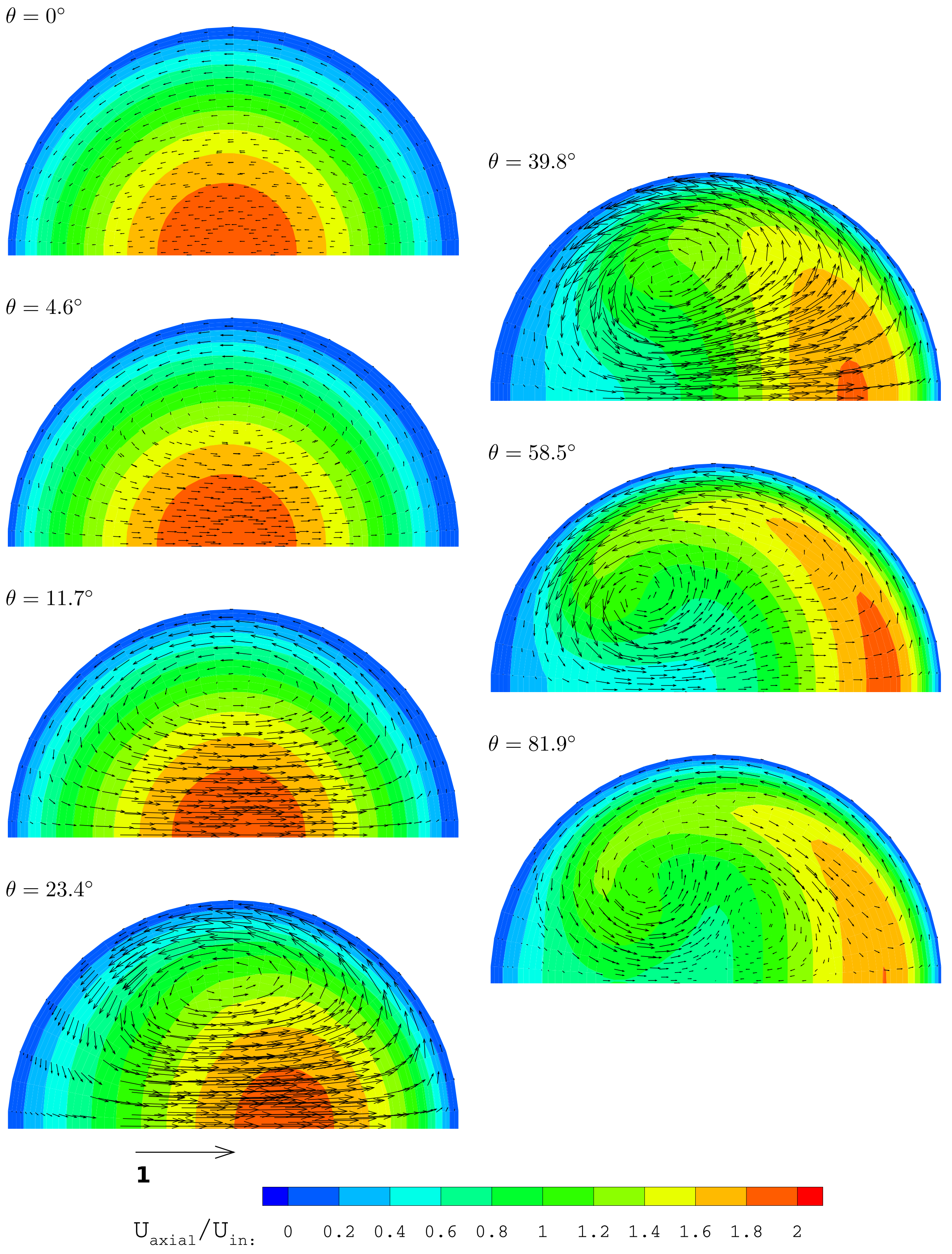


Figure 9.4: Secondary vectors with contours coloured by axial velocity. Inner side of pipe to left of each sub-figure.

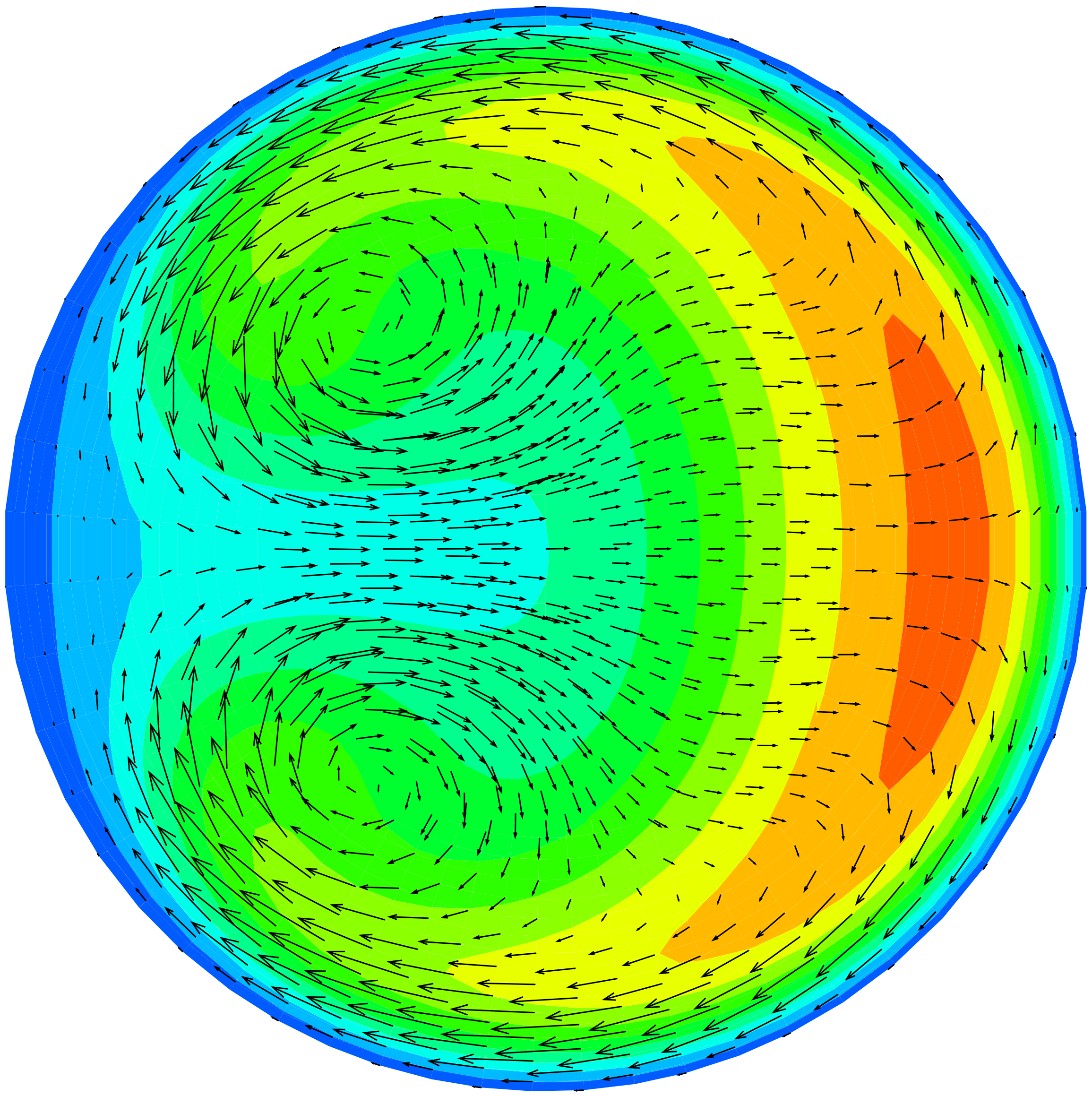


Figure 9.5: Secondary vectors with contours coloured by axial velocity in the plane $\theta = 58.5$. For legend see page 292. Inner side of pipe to left of figure.

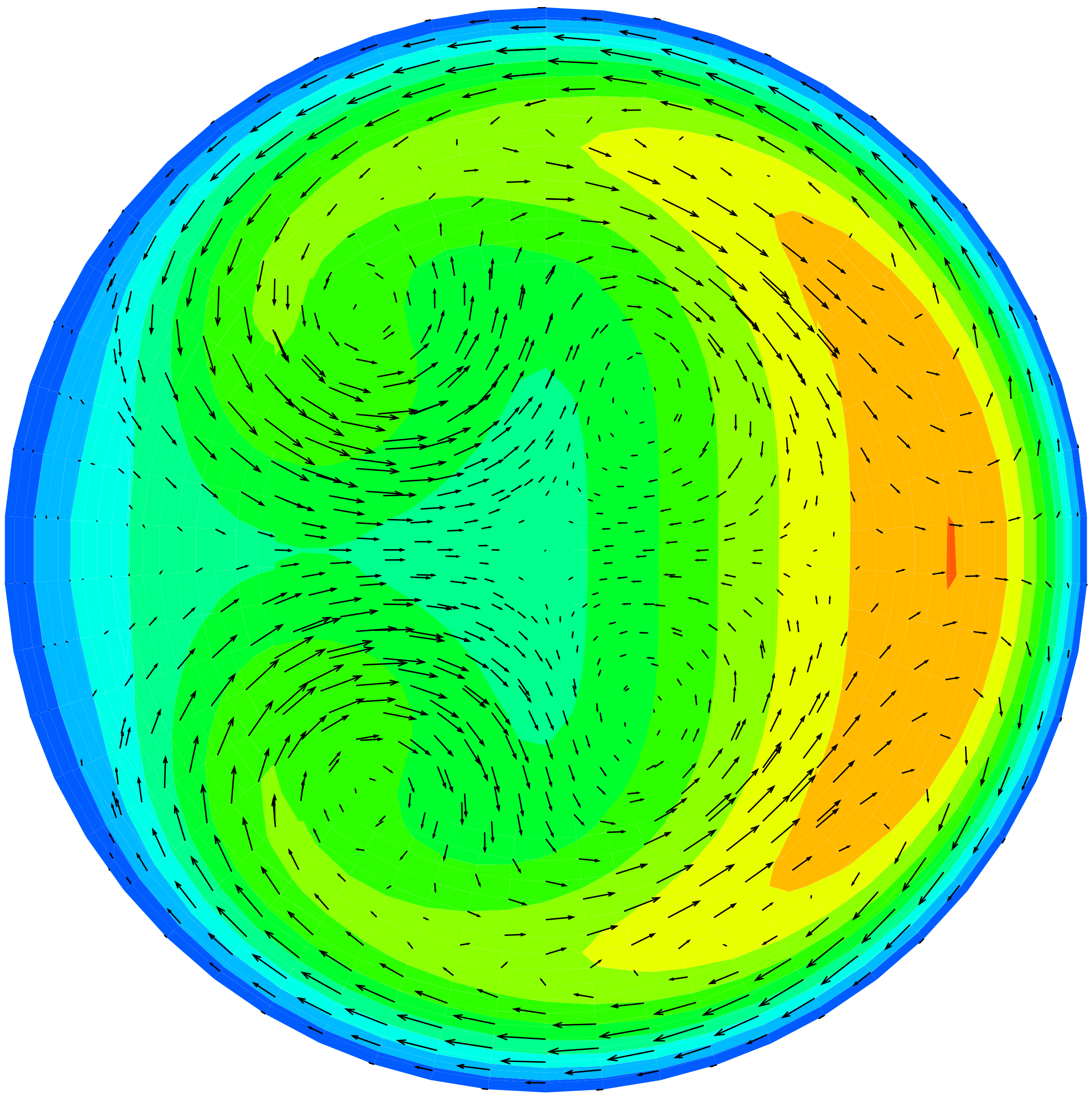


Figure 9.6: Secondary vectors with contours coloured by axial velocity in the plane $\theta = 81.9$. For legend see page 292. Inner side of pipe to left of figure.

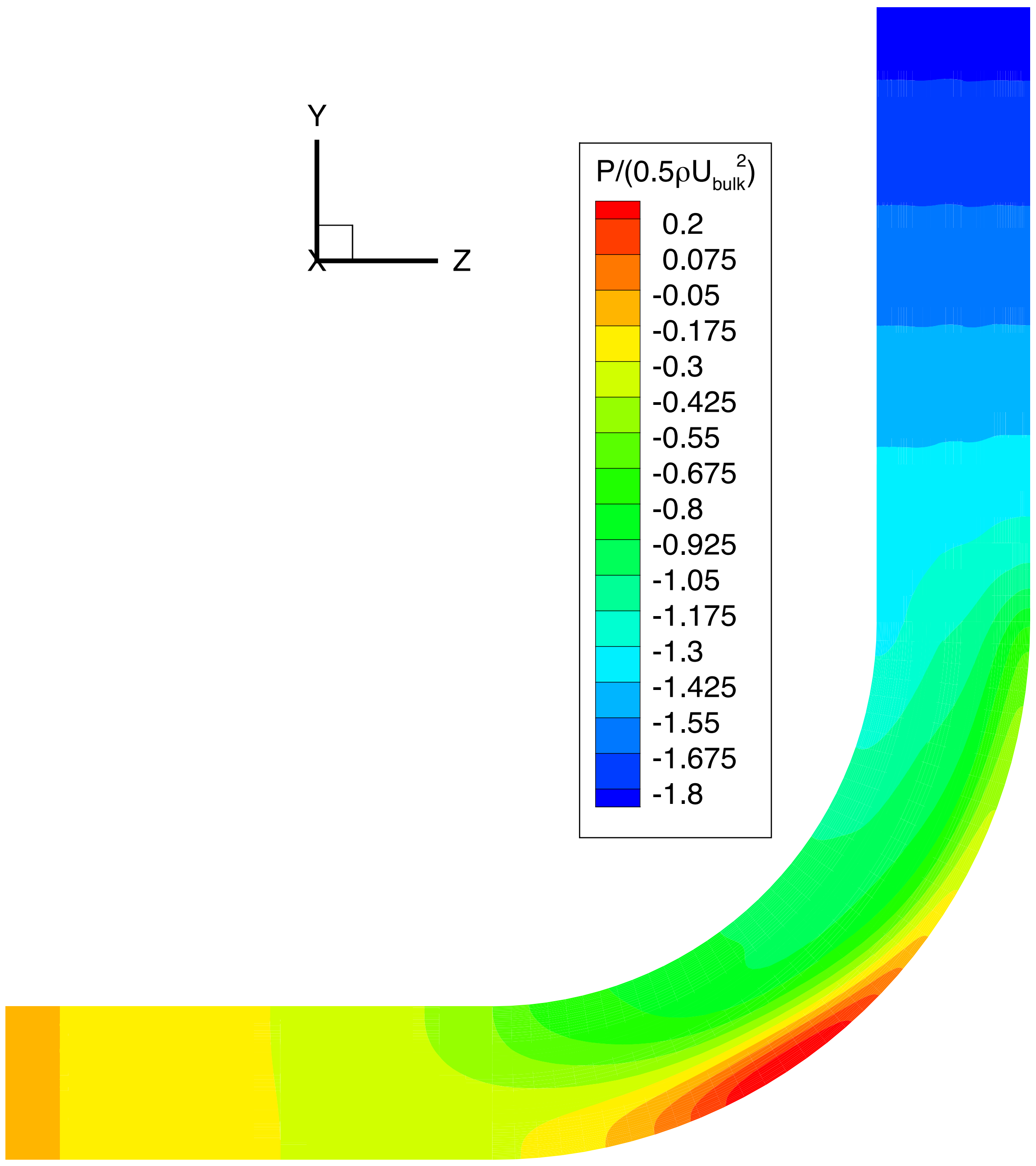


Figure 9.7: Pressure in the symmetry plane.

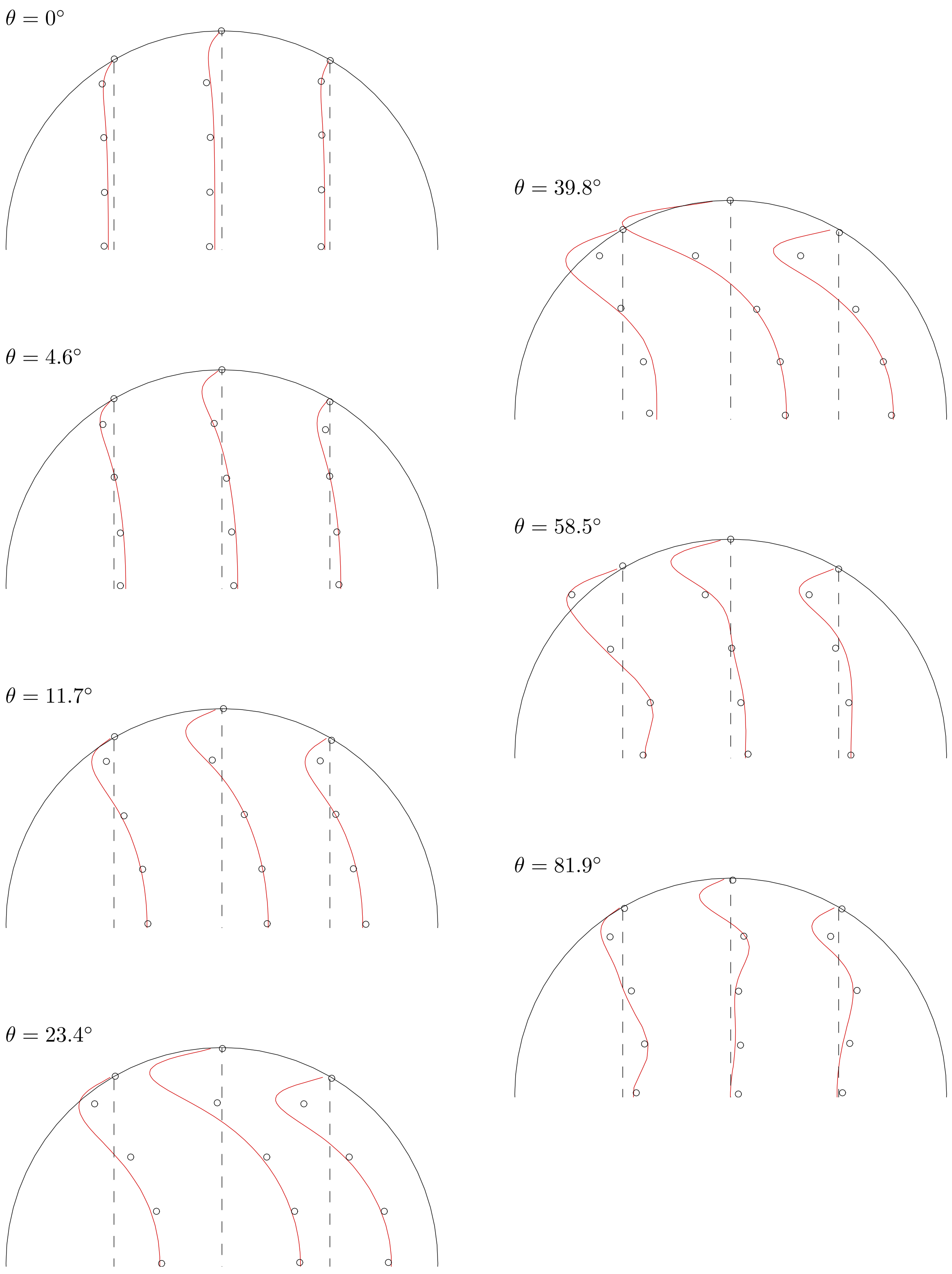


Figure 9.8: Profiles of secondary flow x-component. Distance between two adjacent x-planes at which the profiles are given correspond to a velocity of $0.6U_{in}$. Inner side of pipe to left of each sub-figure.

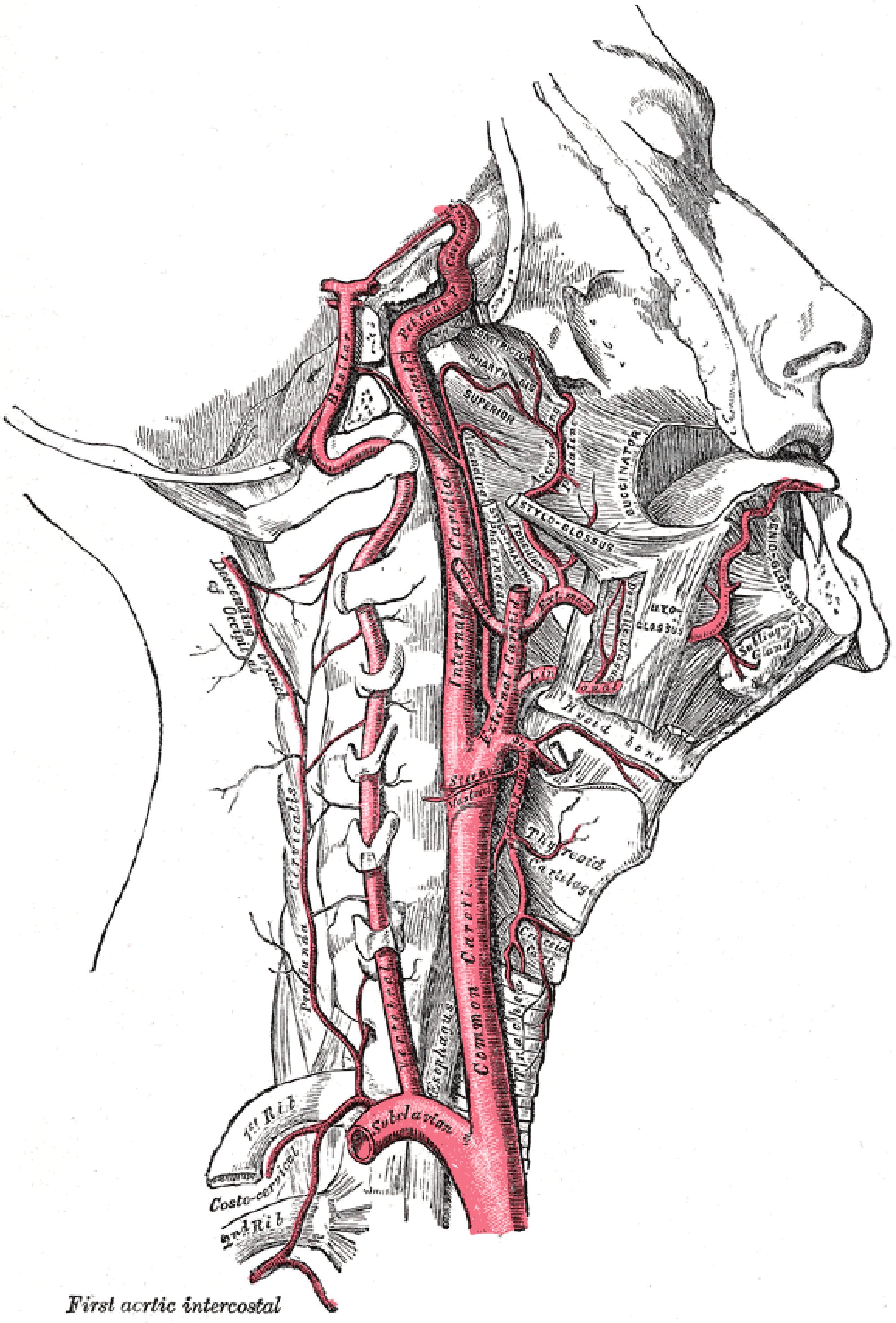


Figure 9.9: Arteries in a human neck. Common carotid artery (labelled common caroti. in the figure) bifurcates into the internal and external carotid arteries.

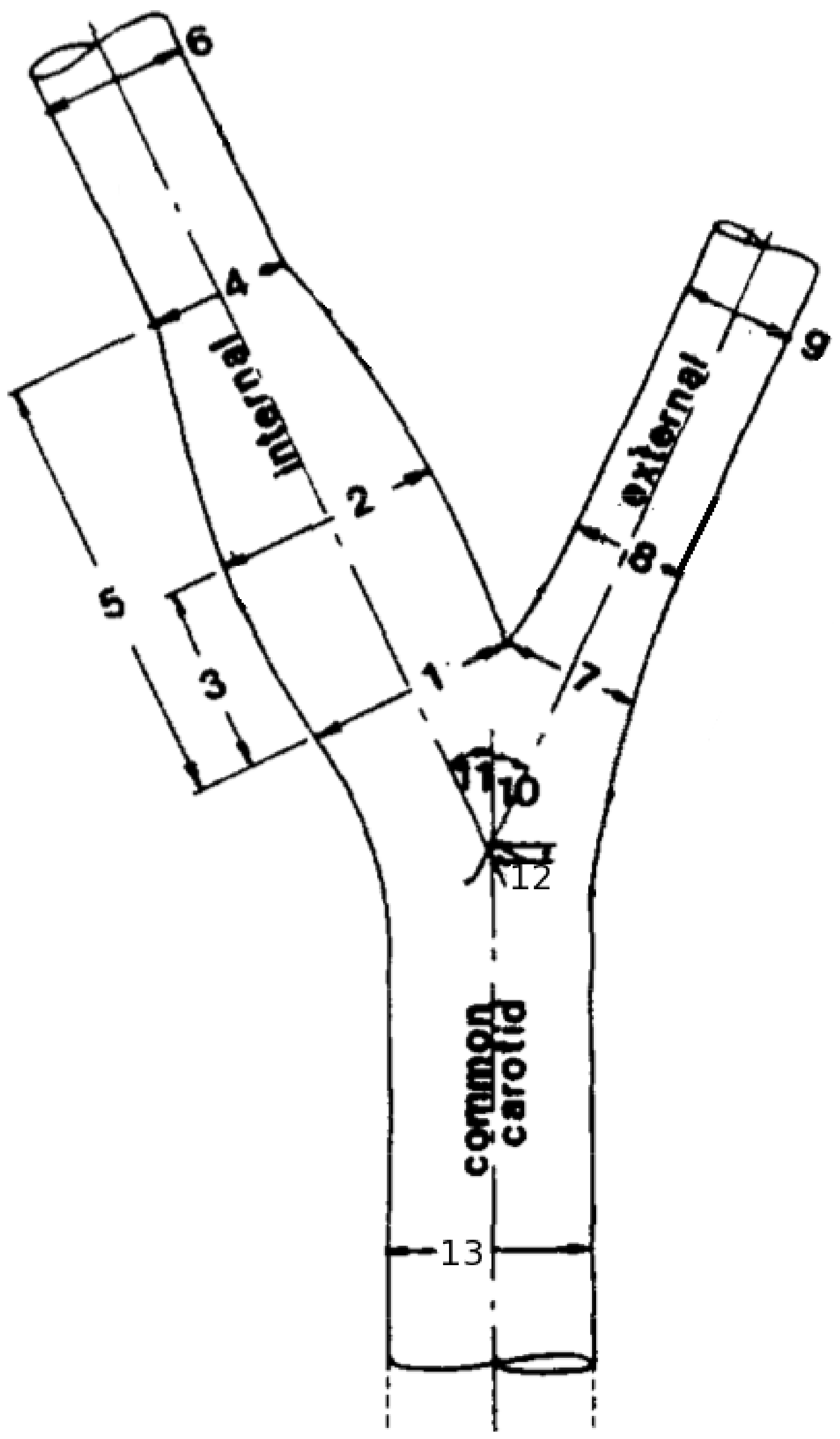


Figure 9.10: The geometry used for the Carotid artery bifurcation. Table 9.1 shows the values of the dimensions labeled $1 - 13$.

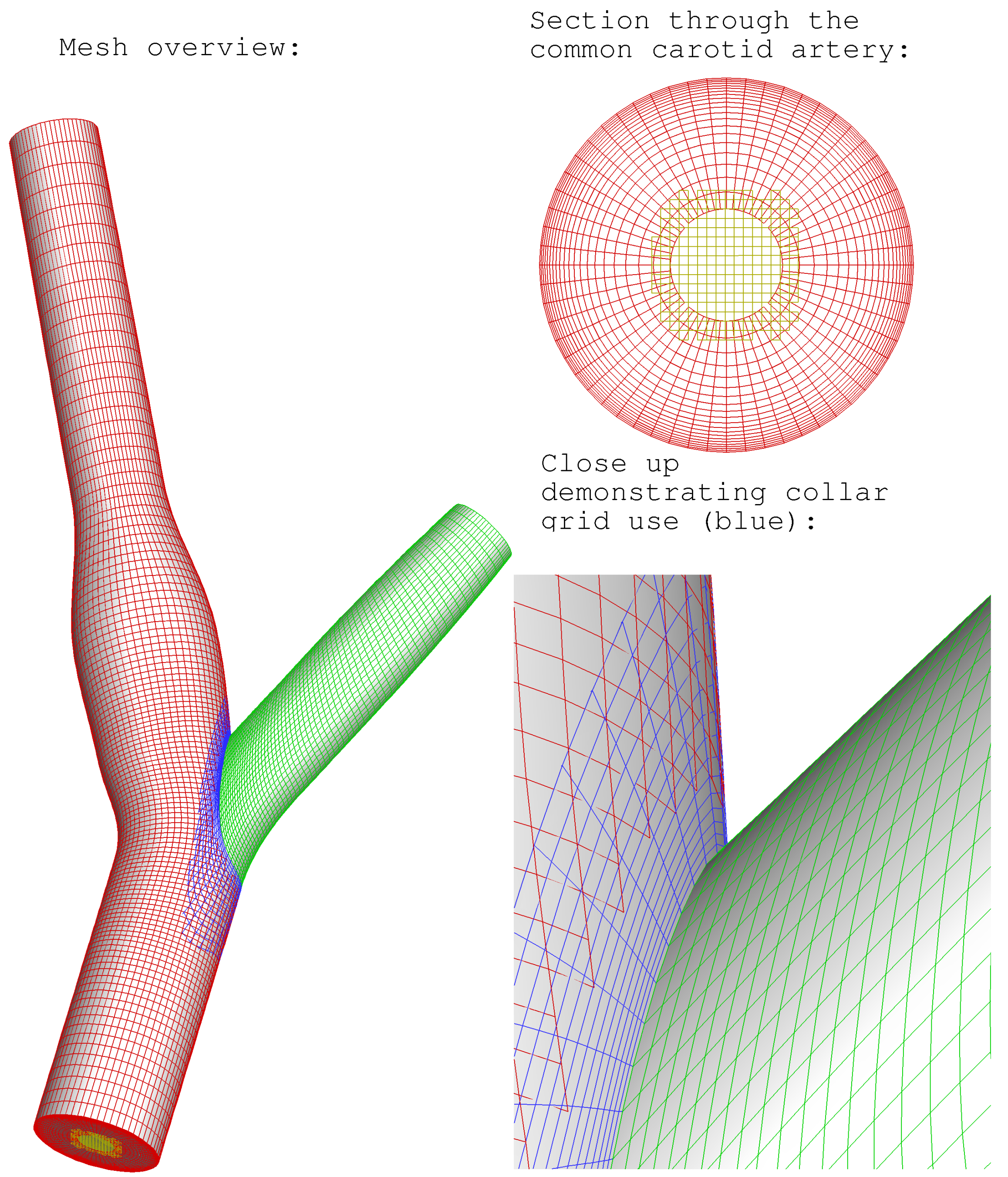

Mesh overview:
Section through the common carotid artery:
Close up demonstrating collar grid use (blue):

Figure 9.11: Meshes used.

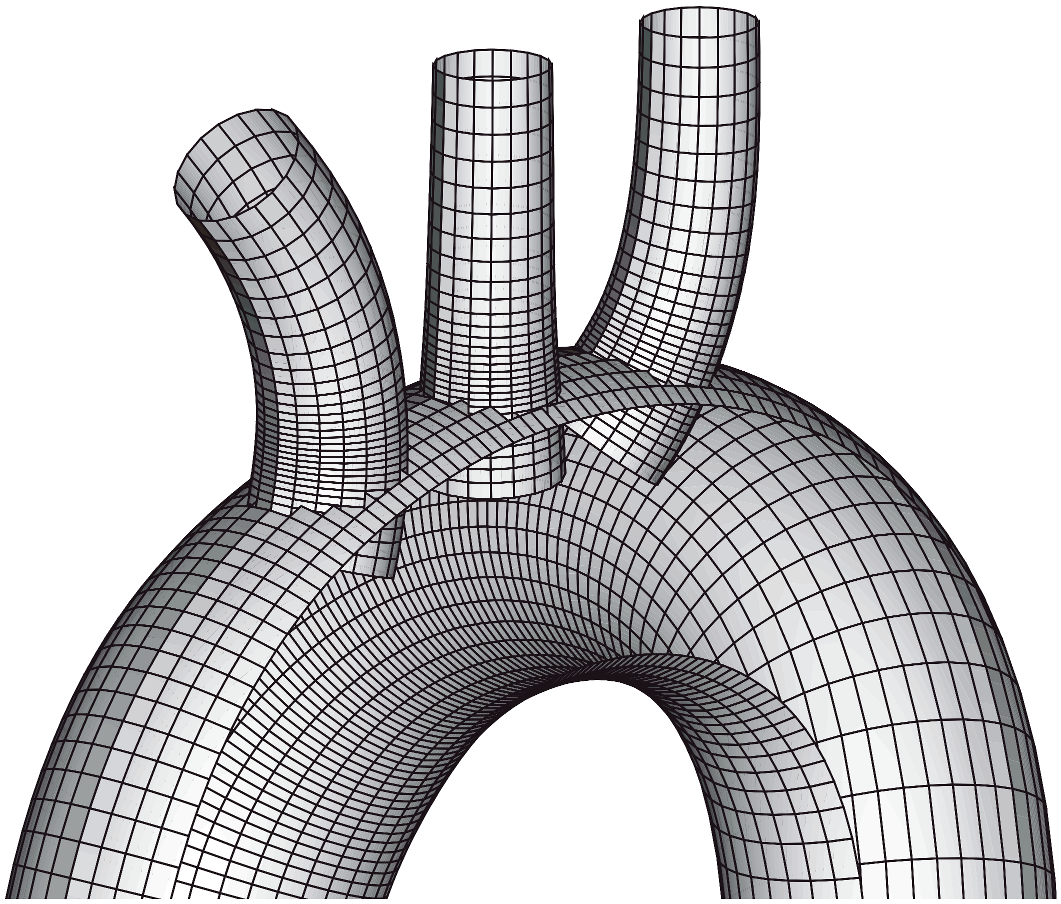


Figure 9.12: Mesh used in [85] for the flow through a human aortic arch. Note the absence of collar grids.

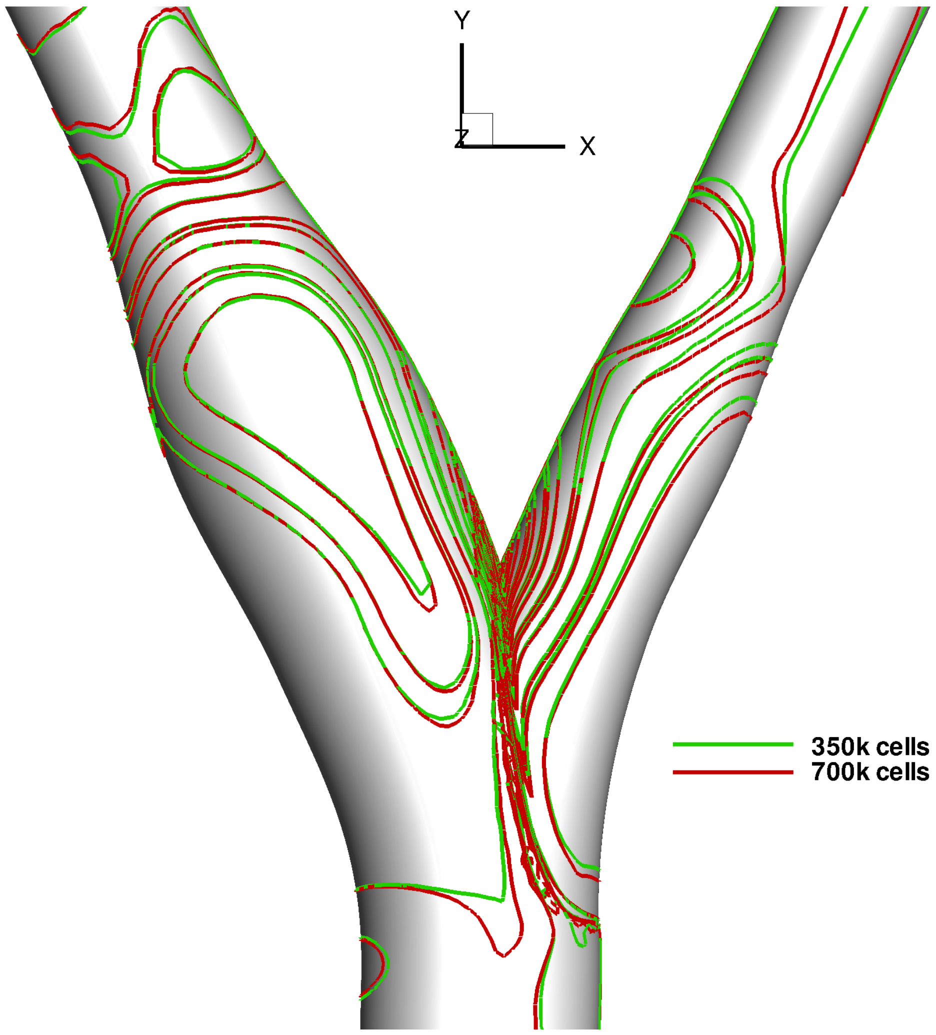


Figure 9.13: Grid sensitivity test showing contours of wall shear stress on bifurcation wall.

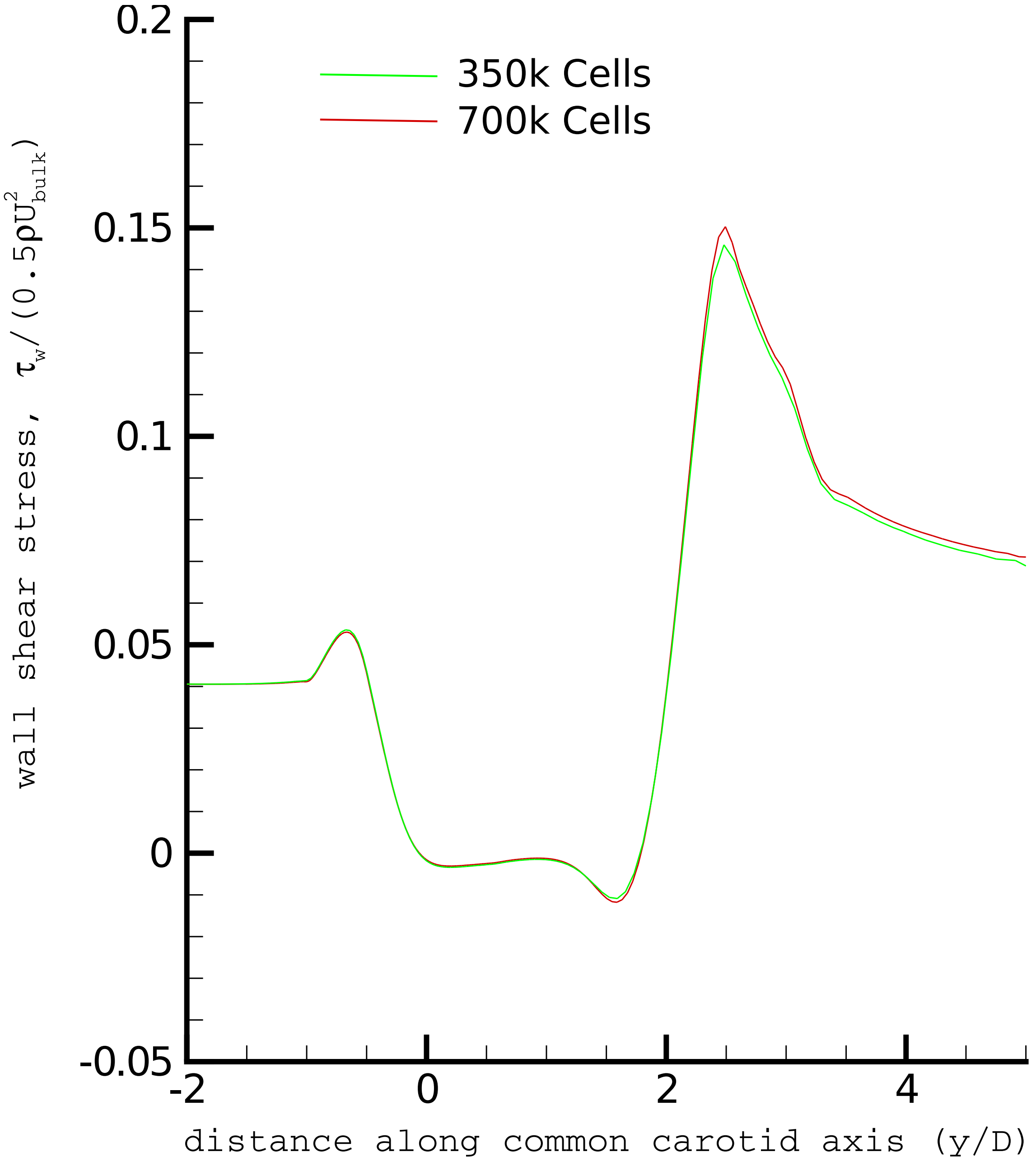


Figure 9.14: Grid sensitivity test showing wall shear stress at the symmetry plane along the common-internal arterial wall (i.e. at $\alpha = \pi$) for the two grid resolutions under consideration.

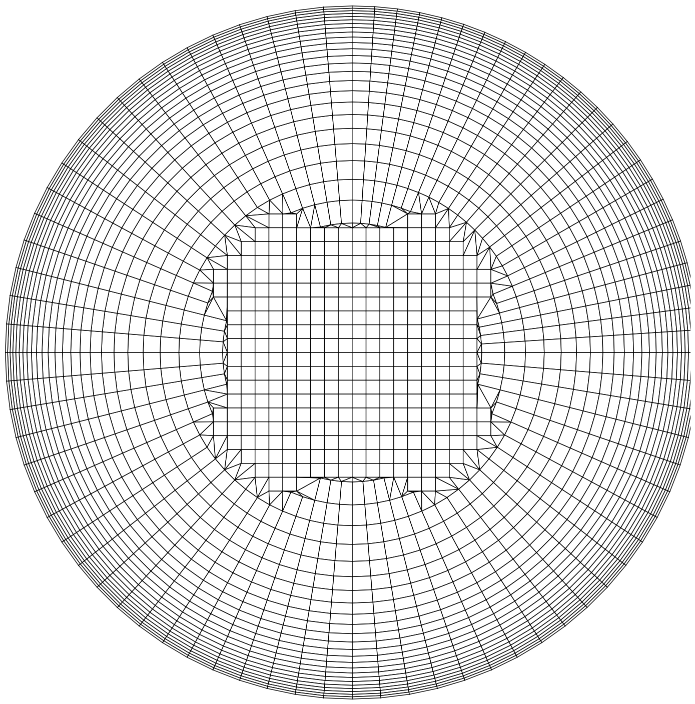
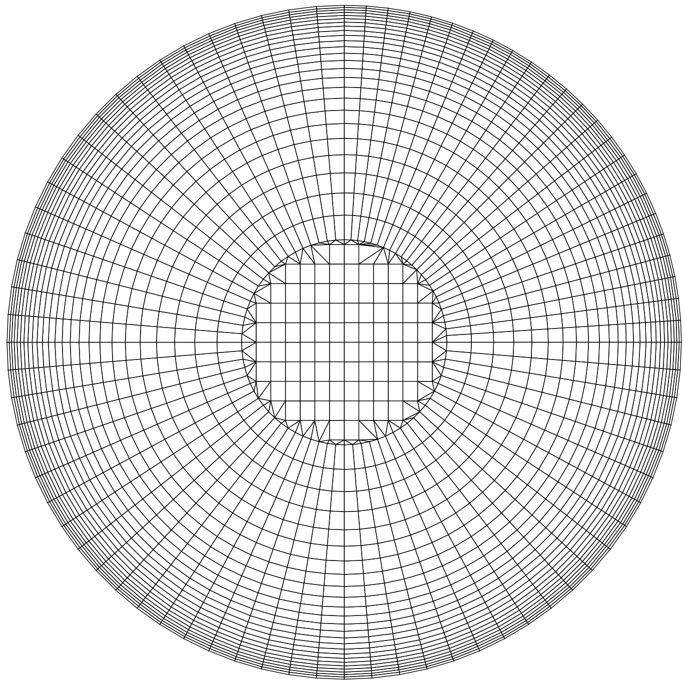



Figure 9.15: Sample of zipped grids used in order to integrate mass fluxes over overset grids. Quadrilaterals are taken from underlying structured grids, while triangular cells have been added by zipper algorithm. Inlet zipped grid (above) and internal carotid artery zipped grid (below).

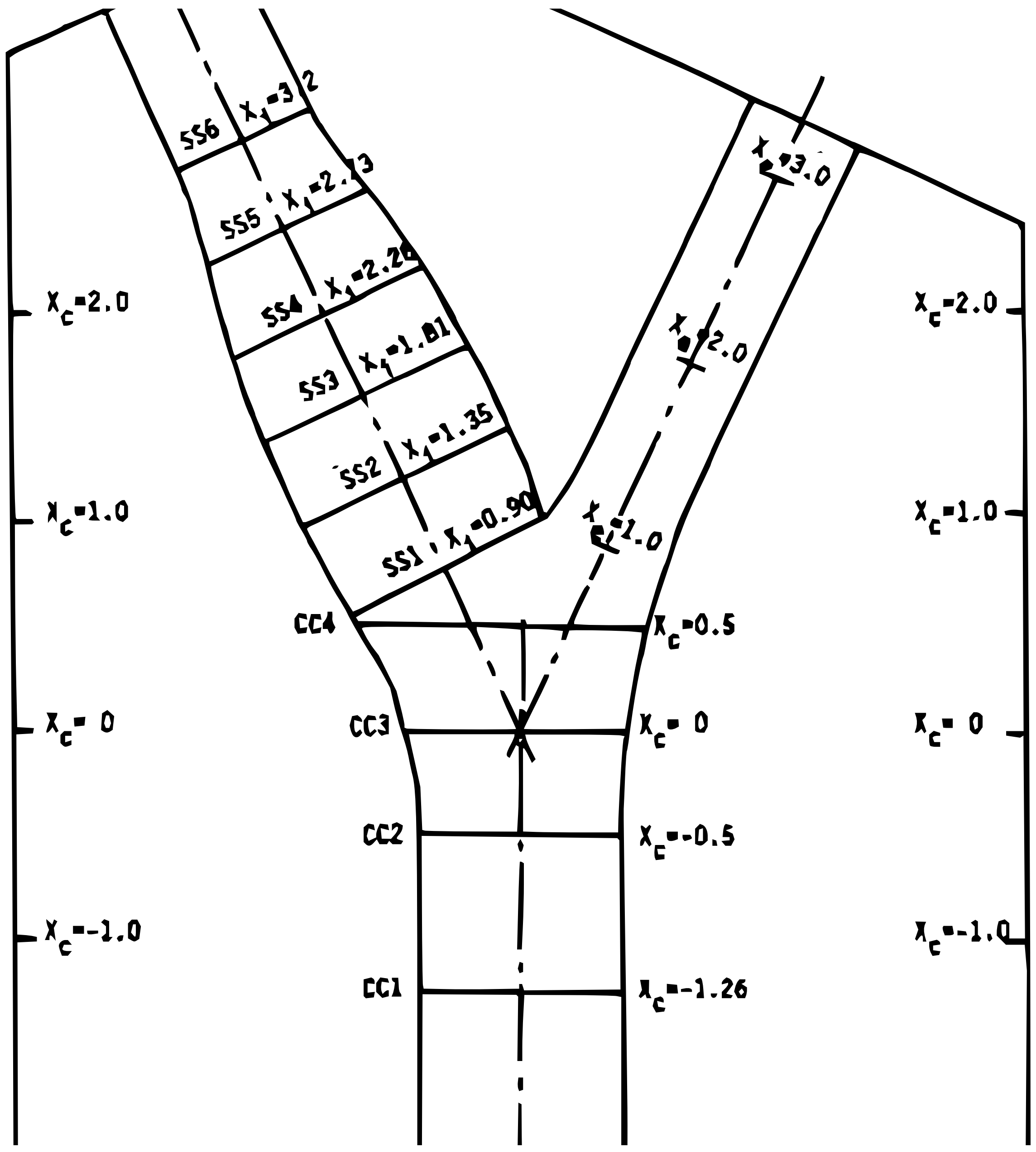


Figure 9.16: View at symmetry plane, showing the locations of planes used in the analysis.

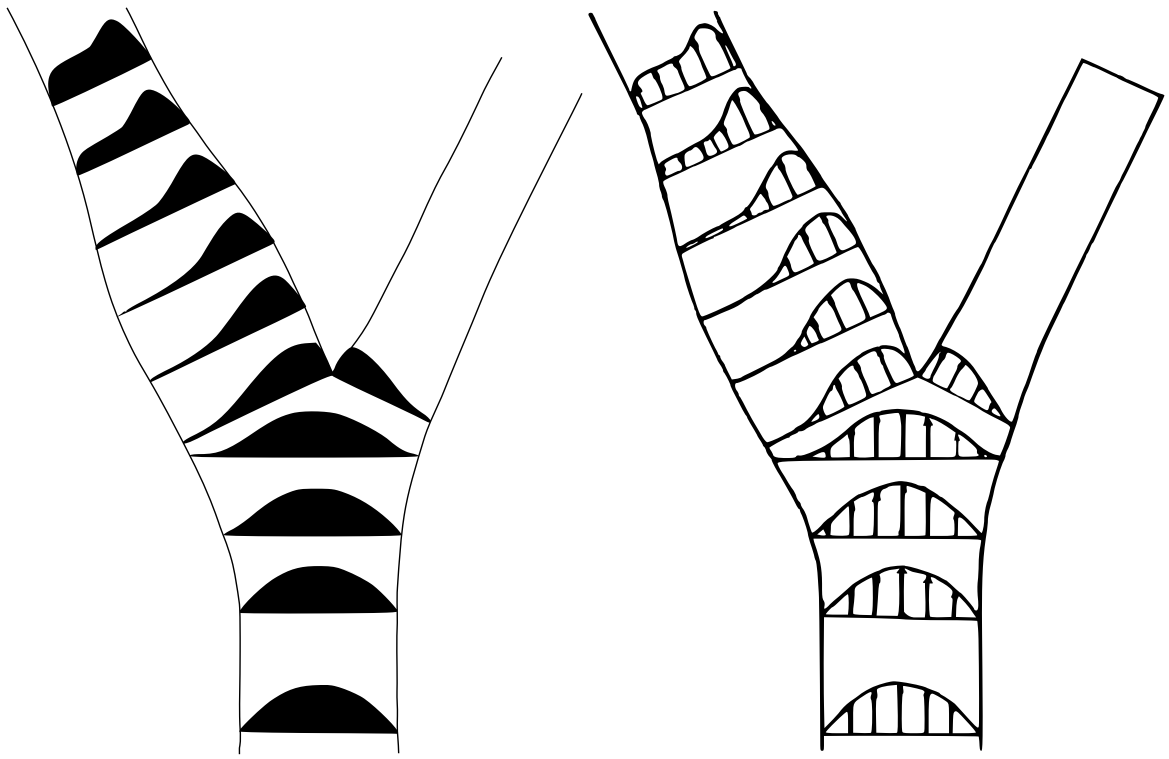


Figure 9.17: Velocity vectors in the symmetry plane. Computed results (left) are compared with experimental data (right).

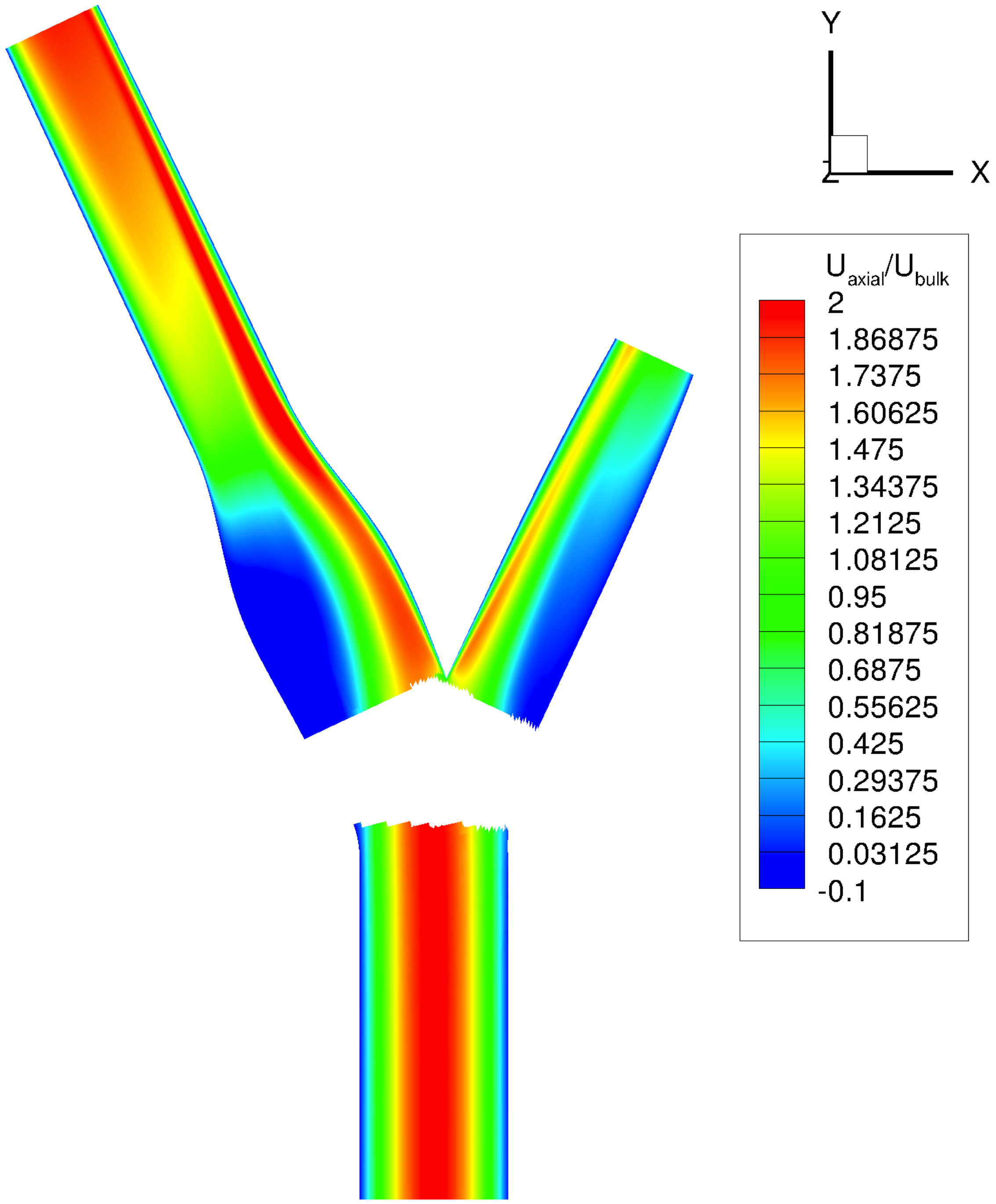


Figure 9.18: Contours of axial velocity. Note that contours are not displayed in the region of the intersection due to the fact an axial direction cannot be defined. $Re = 800$

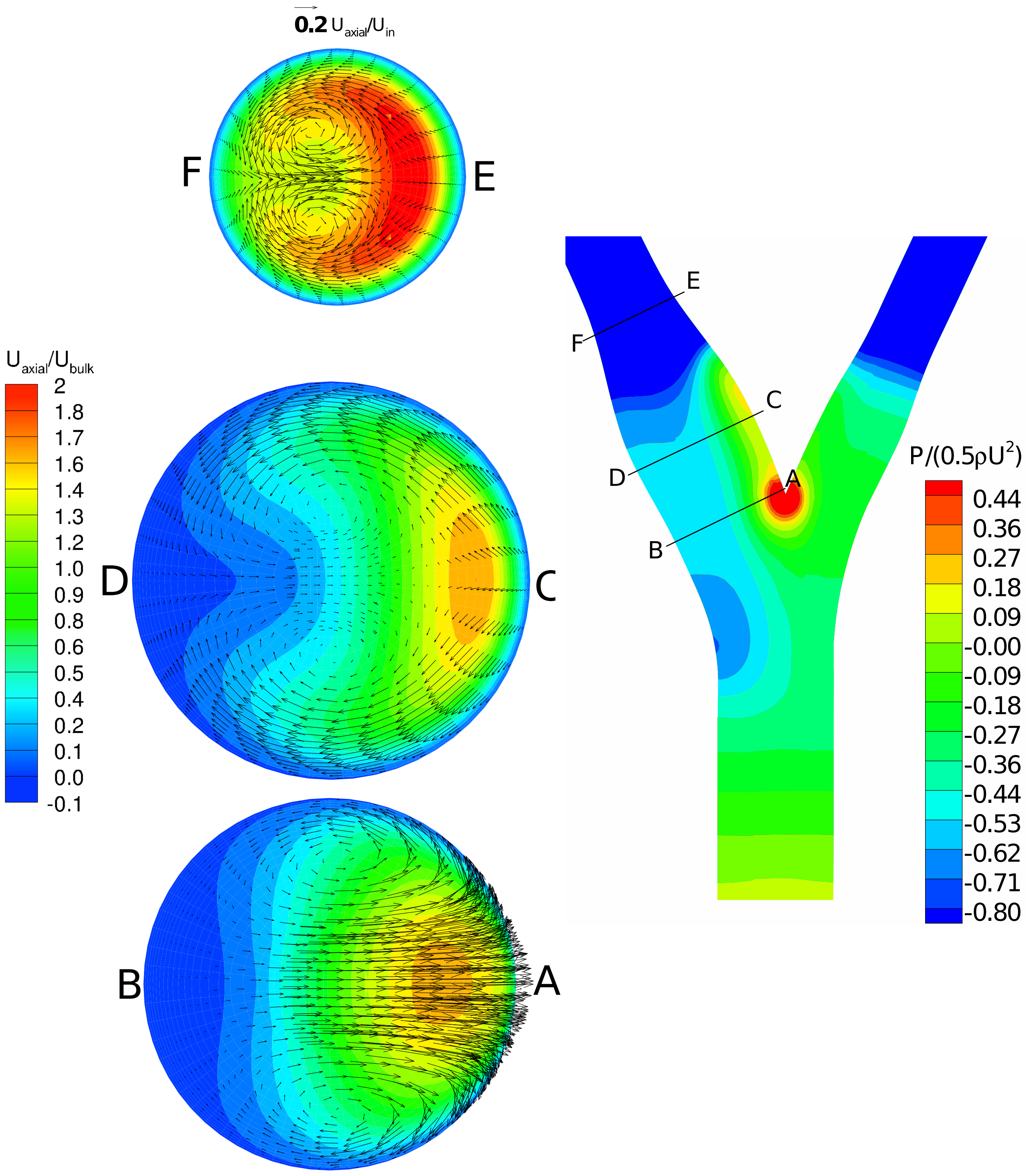


Figure 9.19: Secondary flow vectors and contours of axial velocity at SS1, SS3 and SS6 (left). Contours of the pressure, with the locations of planes SS1, SS3 and SS6 indicated (right).

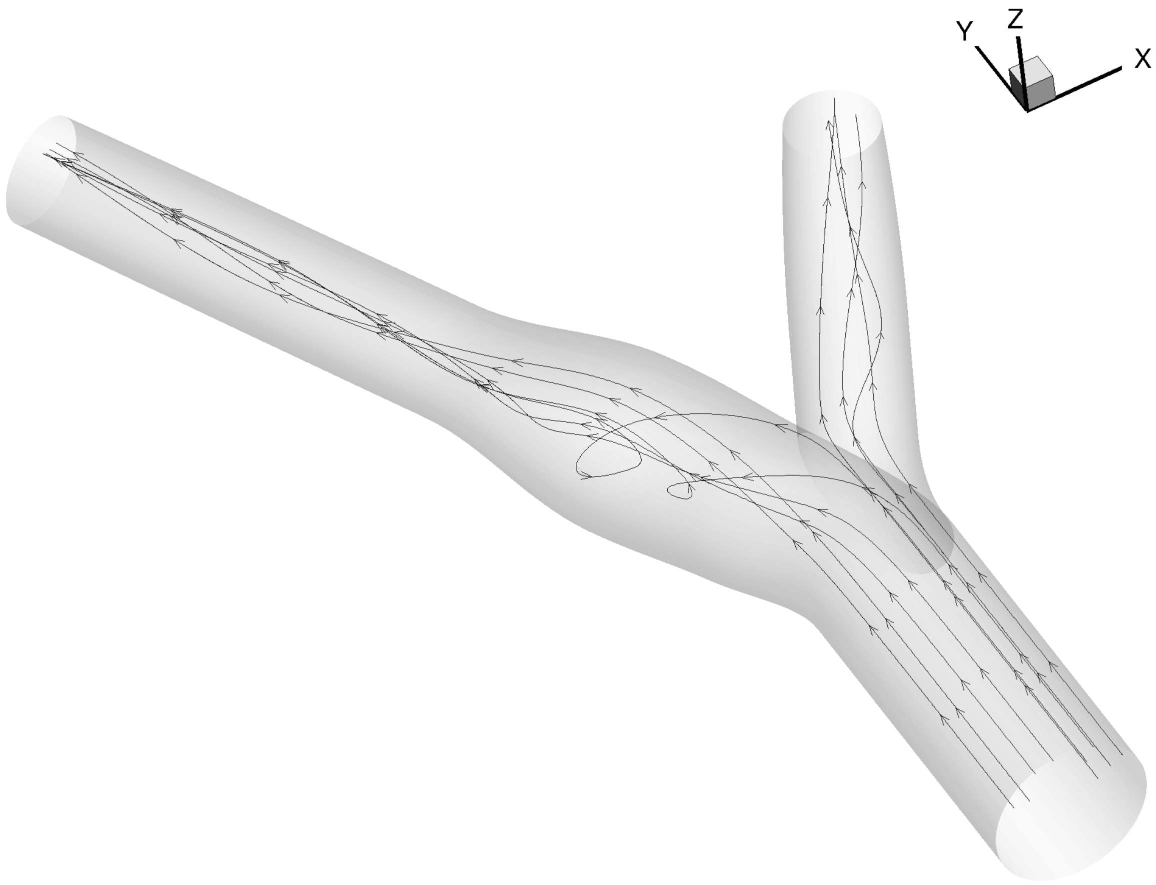
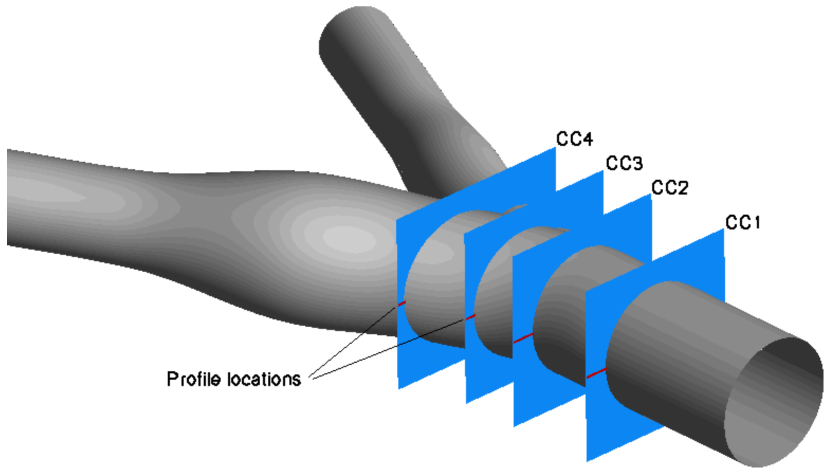


Figure 9.20: Flow streamlines showing helical vortices and flow separation



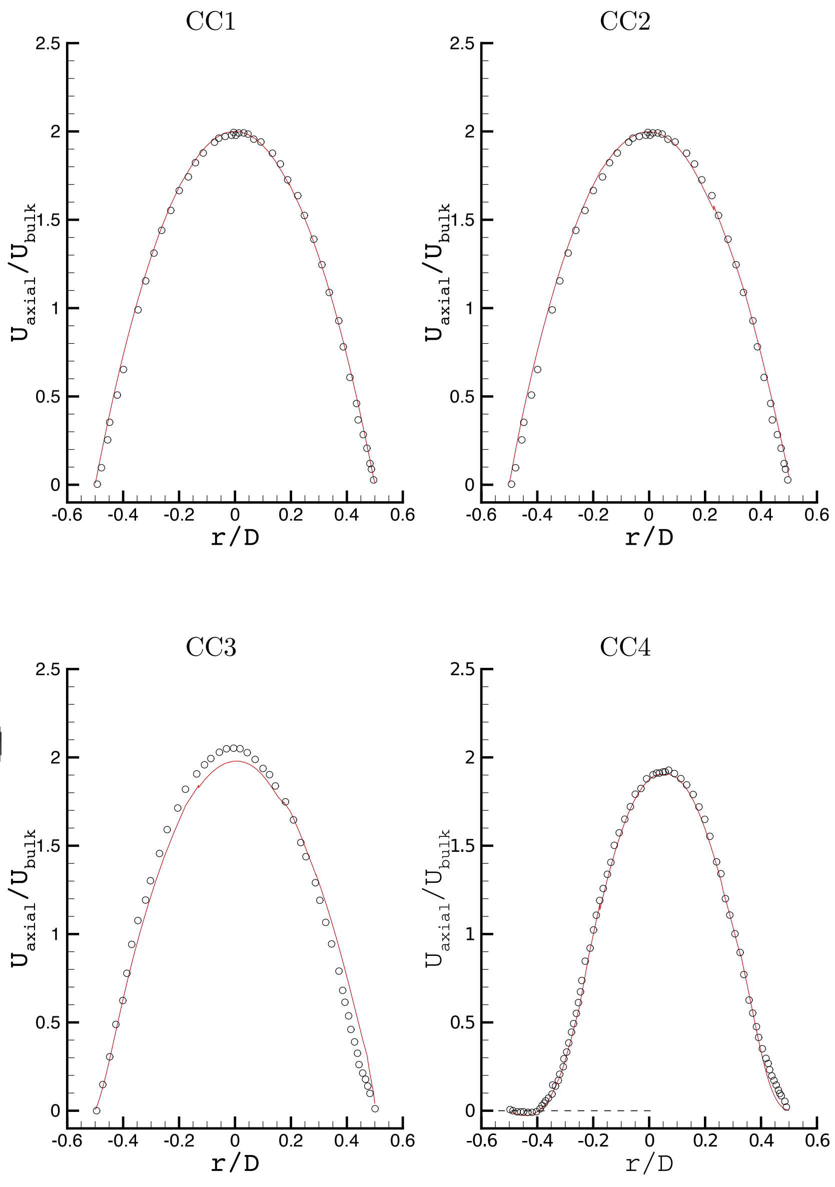


Figure 9.21: Axial velocity profiles at lines defined by the intersection of the symmetry plane with planes CC1-4.

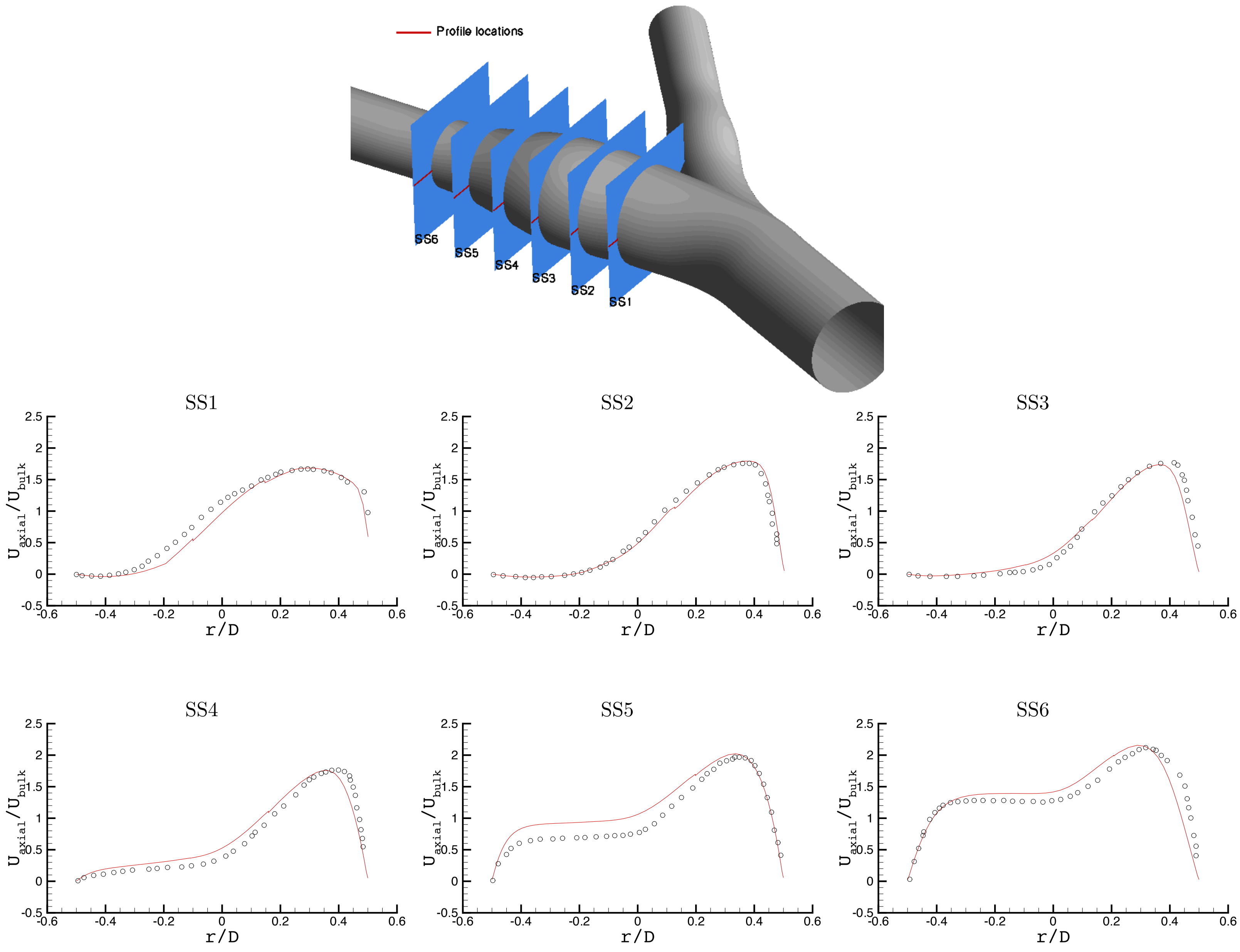


Figure 9.22: Axial velocity profiles at lines defined by the intersection of the symmetry plane with planes SS1-6.

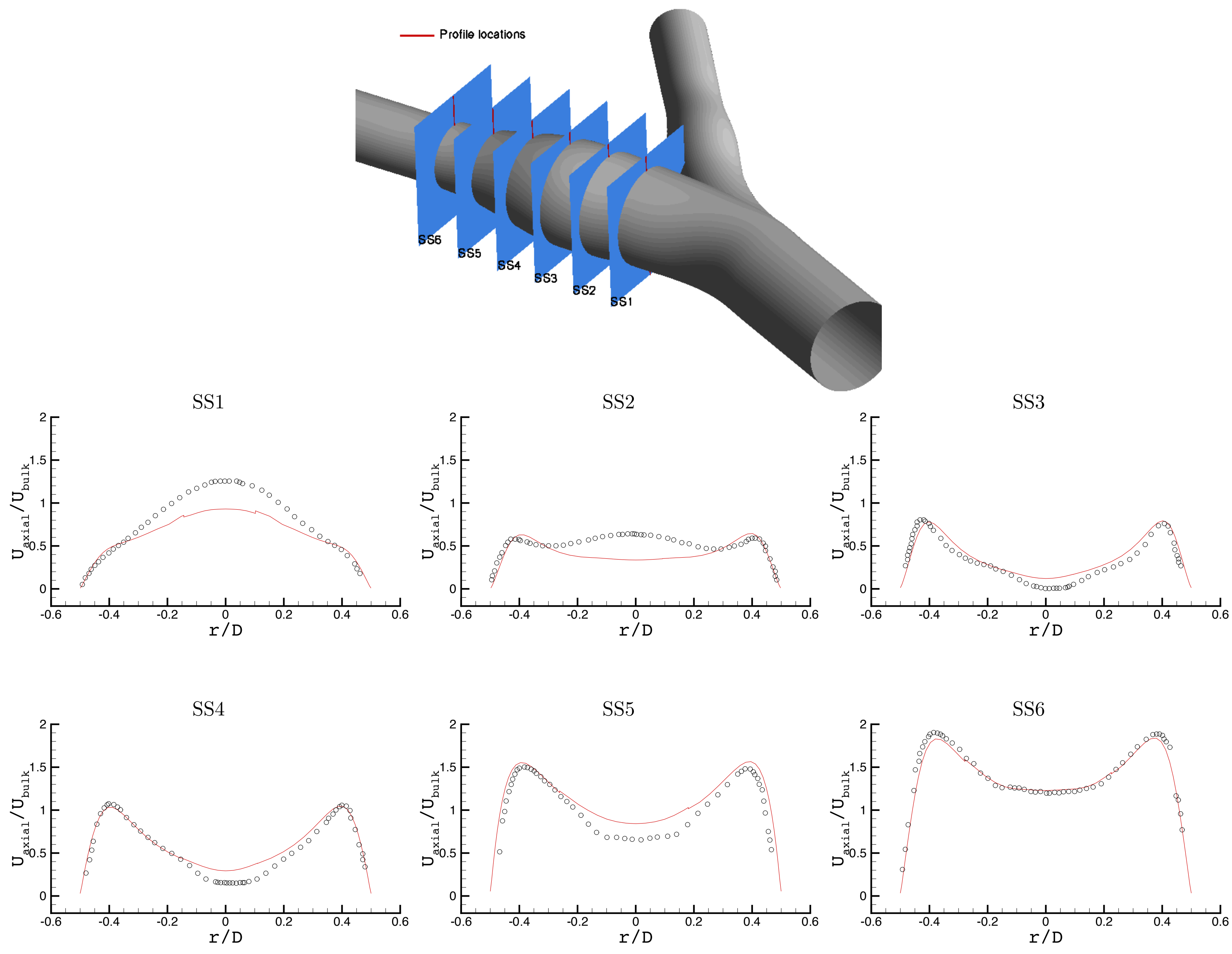


Figure 9.23: Axial velocity profiles taken normal to symmetry plane taken through the diameter, at planes SS1-6. Re=800.

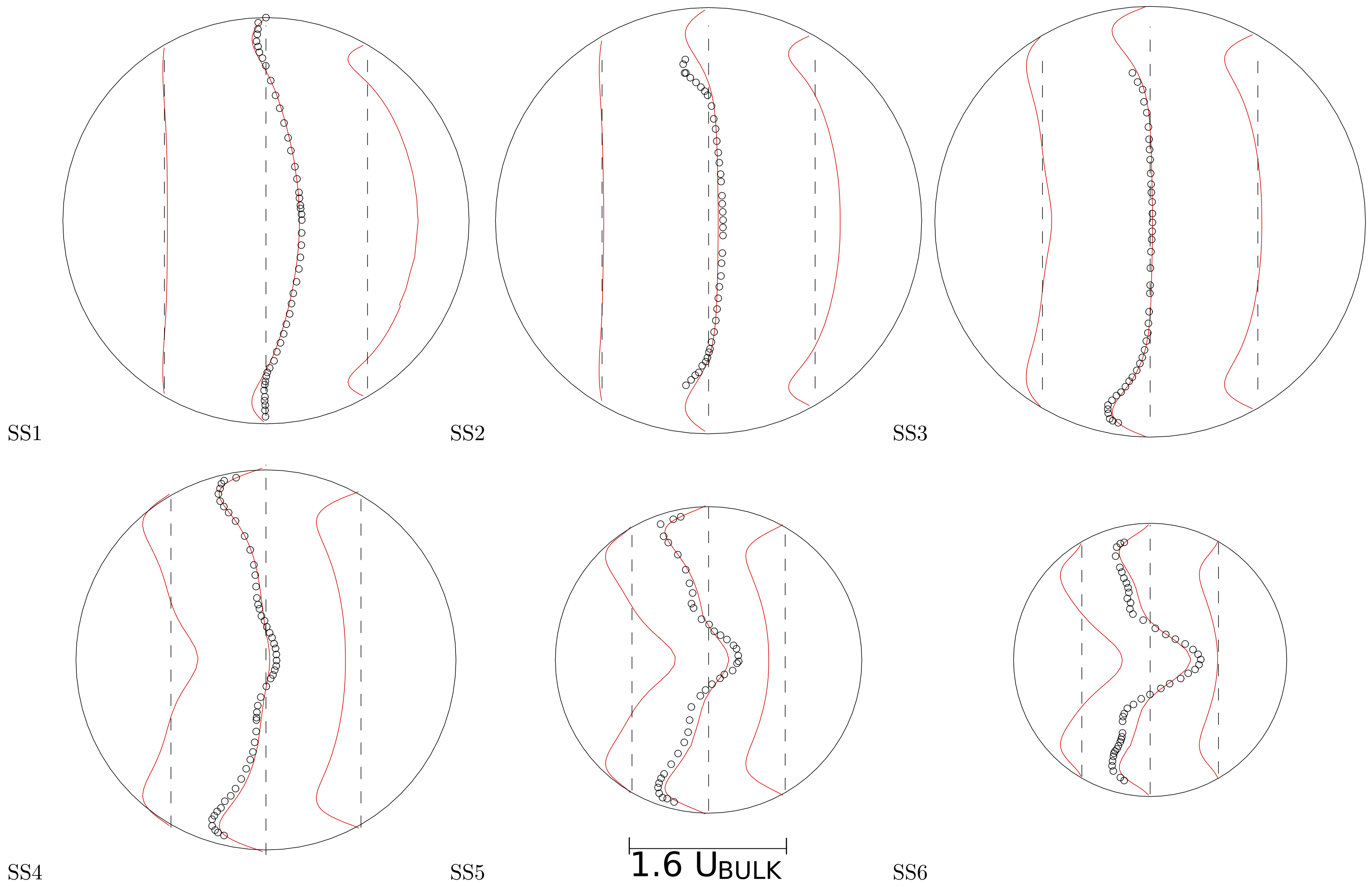


Figure 9.24: Tangential velocity profiles (i.e. velocity component parallel to the symmetry plane) at profile locations taken normal to the symmetry plane, on SS1-6. Re=800.

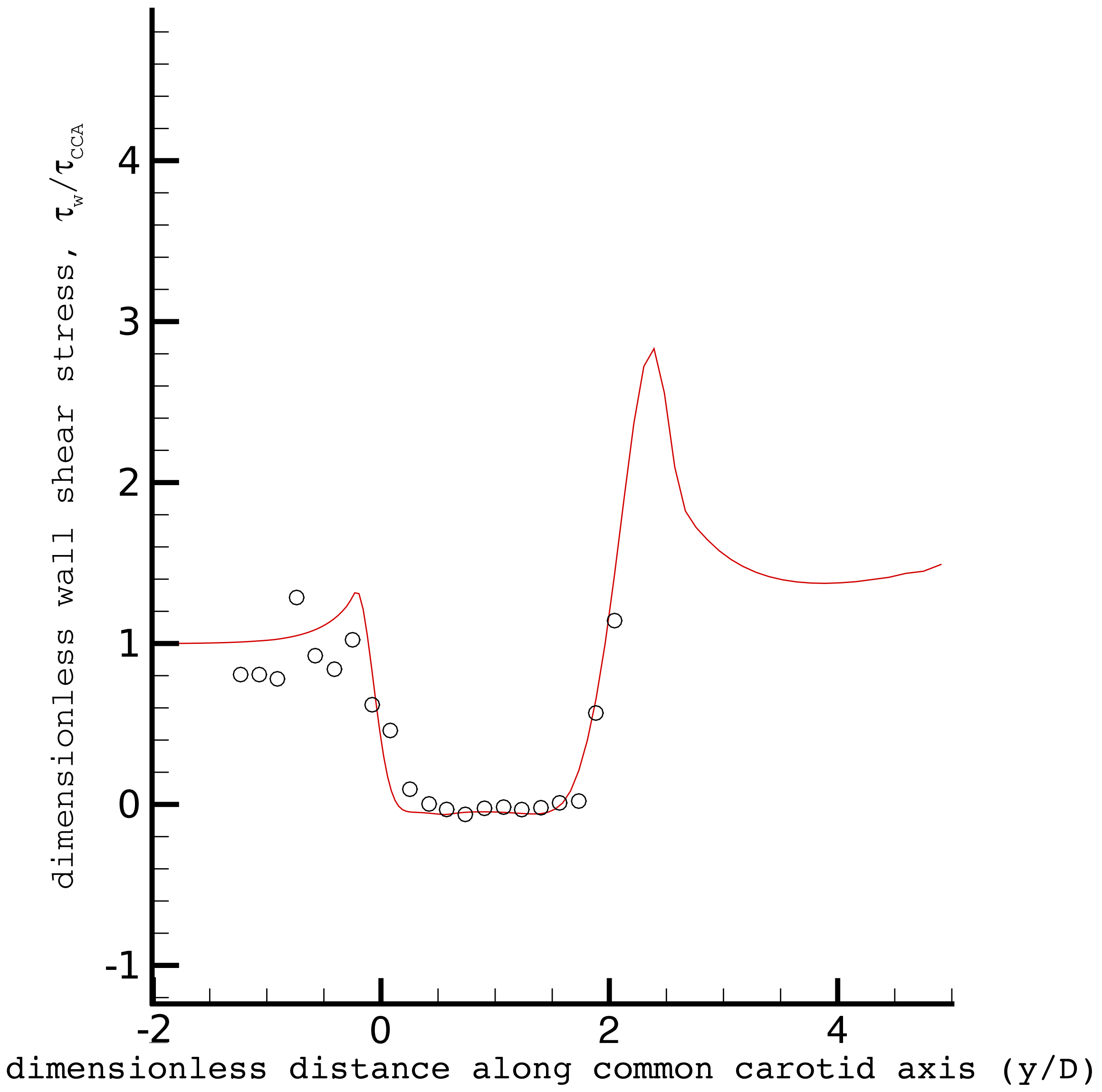


Figure 9.25: Wall shear stress along the common-internal wall. Stress is normalised by the uniform value for the CCA, upstream of the influence of the bifurcation. $Re = 400$

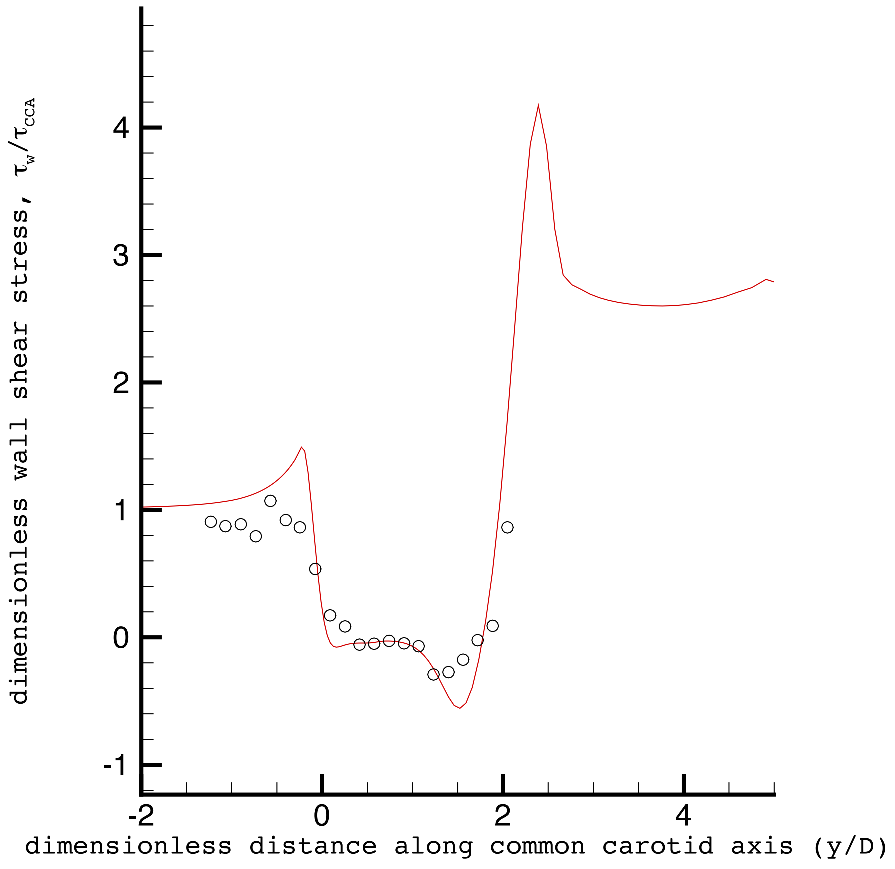


Figure 9.26: Wall shear stress along the common-internal wall. Stress is normalised by the uniform value for the CCA, upstream of the influence of the bifurcation. $Re = 1200$

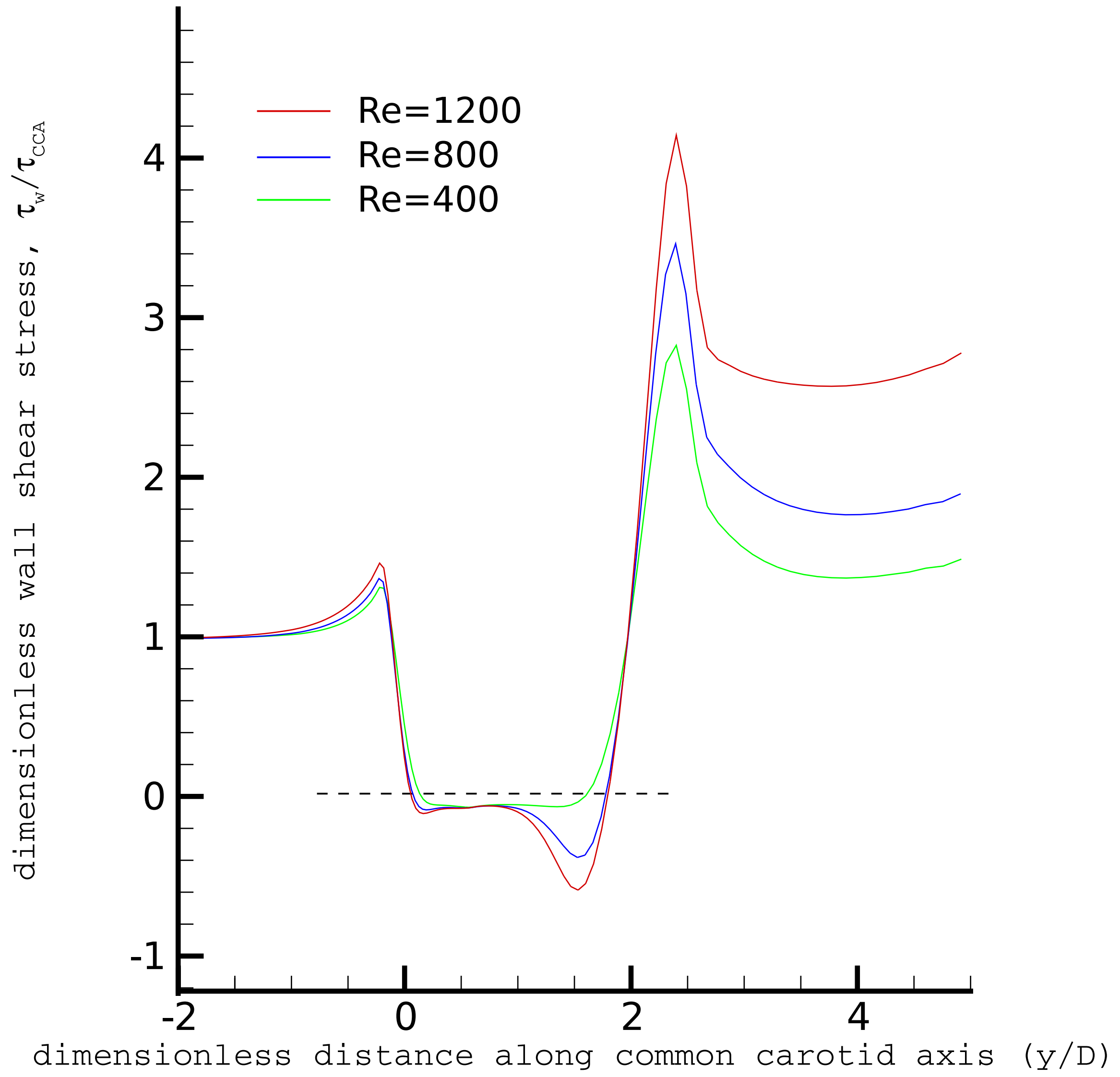


Figure 9.27: Wall shear stress along the common-internal wall at the symmetry plane, normalised by the uniform value for the CCA.

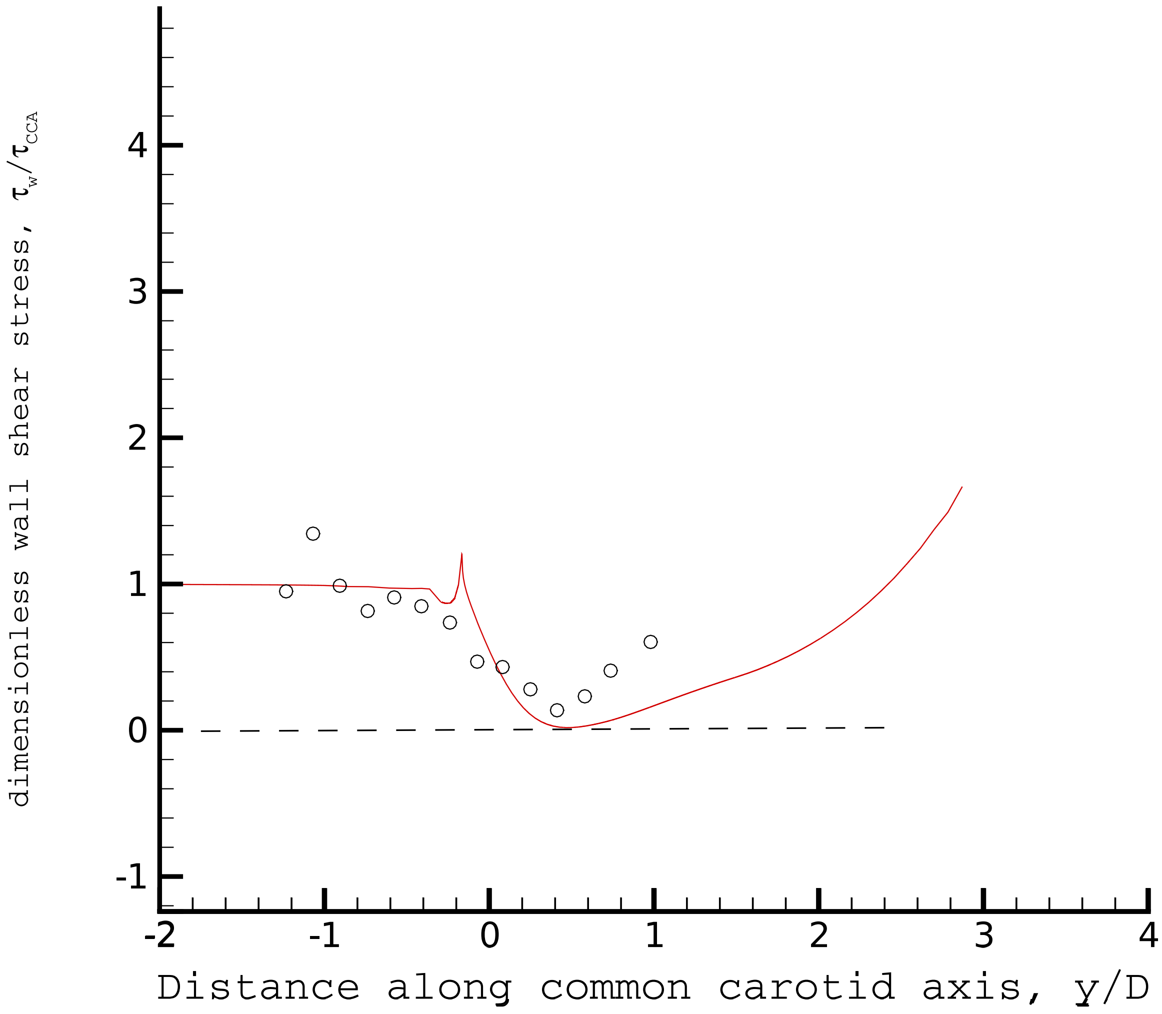


Figure 9.28: Wall shear stress along the common-external wall at the symmetry plane.

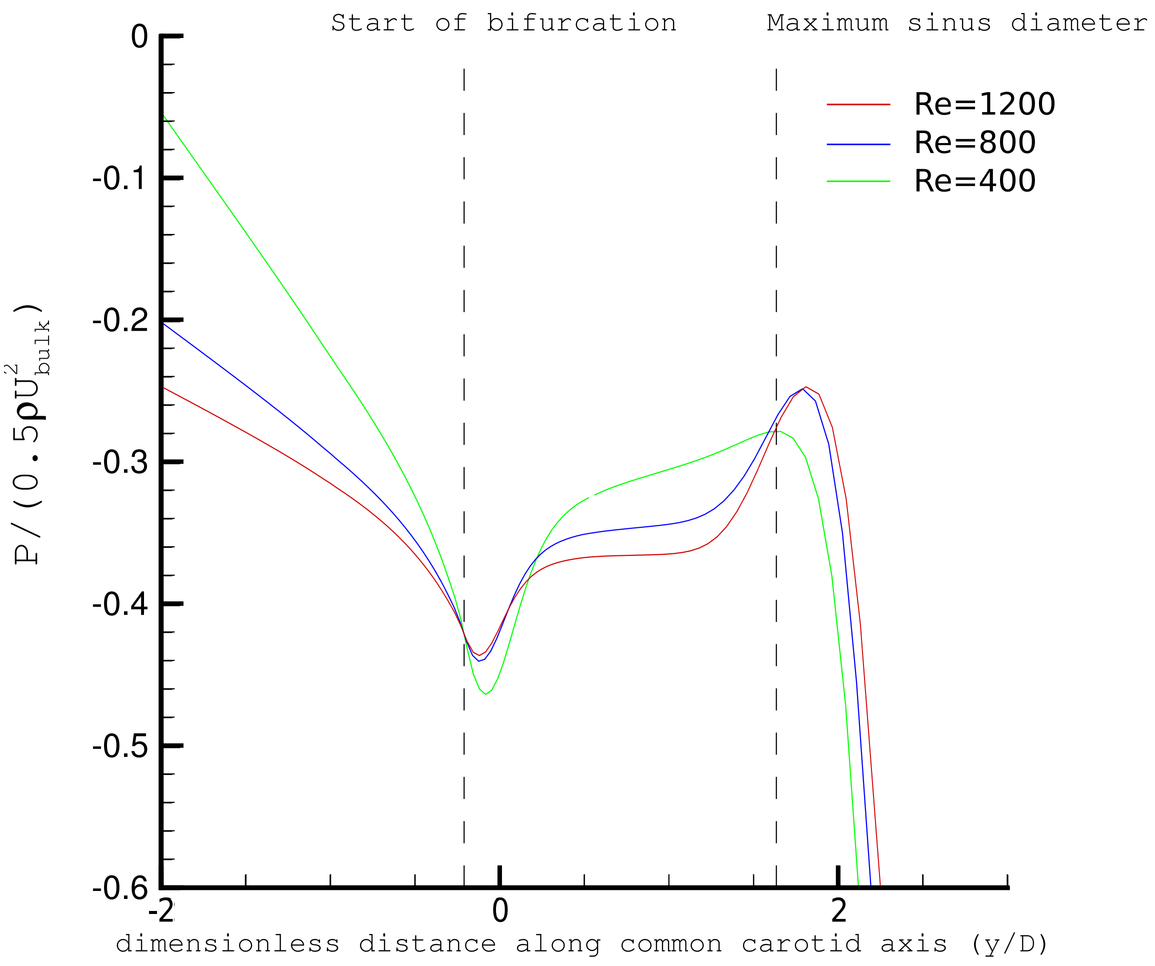


Figure 9.29: Pressure along the common-internal wall at the symmetry plane.

Chapter 10

Wing-body junction flow

The figures referred to within this chapter can be found from Page 333.

10.1 Introduction

The turbulent flow in the vicinity of a wing-body junction is particularly complex. Three-dimensional effects induce secondary flow features, typically causing the formation of vortical structures. Transport of momentum by these vortices significantly affects the flow. Turbulence also plays a major role, with high normal stress anisotropy, strong streamline curvature and an adverse pressure gradient upstream of the leading edge stagnation point all making the flow particularly challenging for a turbulence model to deal with.

The overset grid method is useful yet again for meshing the domain of a wing body junction. Where the wing comprises of multiple components such as flaps and slats, all of the benefits previously outlined for a multi-element airfoil apply (Chapter 8). If additional features are present at spanwise locations on the wing, such as engine nacelles, clearly the overset method would be much better equipped to deal with such a geometry than a block structured grid, with the latter being impractical. Furthermore, if the body has streamwise or cross-stream curvature, as is typically the case for a wing-fuselage junction, a block structured grid would be particularly challenging. If the wing tip is rounded, or has other geometrical complexities, this is also true. The overset grid method therefore shows great potential for a generic wing-body geometry, or for very complex junction geometries such the flow over a space shuttle [86] or a full aircraft configuration [87, 88]. Figure 10.1

shows sample grids for a junction considering some of these features. These grids were used for an early developmental stage of the hole-cutting algorithm. Although no flow simulations have been conducted with these grids, they do demonstrate the potential of the method.

The geometry used for the flow calculation presented in this Chapter is somewhat simpler than that of Figure 10.1; sufficiently simple for a block-structured arrangement to be used without too much difficulty. The specific geometry will be described presently. A simple geometry has been used due to the wealth of high quality experimental data that is available for the case [89, 90, 91, 92]. However, it will be seen that even for the simple geometry considered, the overset method provides significant advantages over other griding techniques. The extension from the simple geometry considered to a more complex one is reasonably straightforward.

10.2 Geometry and computational grids

The geometry considered is the same as that considered experimentally in References [89, 90, 91, 92]. The wing section comprises of a semi-elliptical nose section divided along its minor axis, with a NACA 0020 tail section joined at maximum thickness to the aforementioned ellipse. The elliptical nose has a major to minor axis ratio equal to 3/2. The section of the wing is uniform over its entire span. One end of the wing is attached to a flat plate, while the other end is free. Figure 10.2 illustrates this geometry. Also shown in the figure is the coordinate system adopted. One can see that a right hand coordinate system is used with the origin located at the intersection of the wing's leading edge with the lower wall. The streamwise direction is aligned with the x axis. The y and z axes are in the vertical and cross-stream directions respectively.

Figure 10.3 shows the computational grids used. It can be seen from the figure that a total of three overset grids have been used. A Cartesian background grid (red) covers the majority of the domain, a body-fitted curvilinear grid (green) used to depict the near-wing region, while a second Cartesian grid (blue) has been used to provide the additional grid resolution that is required to satisfactorily resolve the wake and vortex structure. This overset grid arrangement demonstrates another very useful feature of the overset method. Initial computations were carried out without the presence of the blue wake refinement grid. Once it was realised that there may be insufficient grid resolution in this region, the grid was added afterwards with the underlying solution on the background grid being interpolated onto the refined grid in order to provide a good initial solution on this grid (prior to reinvoking the hole cutting algorithm to remove the now unnecessary cells of the background grid). The solution on the refined grid therefore converges rapidly thanks to its good initialisation. This is somewhat similar to adaptive mesh refinement that is available for unstructured grids, but here the capability is effectively brought to a structured solver. It can be envisaged that such a feature would be very useful for compressible flows where the location of shocks may not be known *a priori*. Further, while the wing considered here is symmetric, and hence the wake location is known *a priori*, this is not necessarily the case for a lifting wing since the downwash acts to displace the wake. The overset grids outlined here would therefore be useful in this case since the wake refinement region could simply be shifted to the appropriate location before being reinitialised via interpolation from the underlying cells at the new location. Since the solution shows low spatial variation outside of the wake for all primitive variables, there is little need to attempt to match the grid cell size at the interface; significant heterogeneity will be tolerated.

Another major advantage of the overset mesh over other meshing techniques is that the vast majority of the domain is filled with Cartesian grids, and hence much of the geometric data that would otherwise be required is not stored, thereby reducing the memory footprint. By tagging the Cartesian grids as curvilinear (which is valid since the latter is a generalisation of the former), around 18% increase in memory usage was found, with no difference to the solution (other than the negligible differences due to numerical round-off errors). CPU time is also increased by around 5%. Further optimisation of the present code may yield yet further improvement.

Initial computational grids used around $400,000$ cells, while a refined mesh consisting of $600,000$ cells was used in order to investigate grid convergence. Figure 10.4 shows the result of this grid sensitivity test, from which it can be seen that there is no appreciable difference in the surface pressure for the two grid densities. Similar grid convergence is also observed for the profiles of mean velocity.

Figure 10.5 shows the grids used for this geometry in a study compiled by Apsley and Leschziner, [93]. Note the high aspect ratio cells that propogate out from the trailing edge which is avoided by the more flexible gridding of the present study. In [93], around $330,000$ cells have been employed. While fewer cells are used in [93], despite the off-body high aspect ratio cells that are avoided in the present study, the near wall y^+ is lower here than is reported in [93] (0.3 at the near-wall cell centre for the approach flow boundary layer at the inlet for the present study, versus 0.5 in [93]). The near wall y^+ is below unity for all near-wall cell-centres in the present study.

10.3 Boundary conditions

In the experiment the approaching lower wall boundary layer is tripped by means of a step at $x/T - 21.0$. Profiles of $\overline{u}$, $\overline{u'^2}$ and $\overline{w'^2}$ within the partially developed turbulent boundary layer are provided at $x/T = -18.24$ [89]. The computational inlet has therefore been placed at this location. The non-measured mean velocity components (i.e. $\overline{v}$ and $\overline{w}$) have been assumed to be zero at the inlet, which is reasonable for a 2D boundary layer. A log-law empirical approximation has been used to estimate $\overline{v'^2}$ as $0.4\overline{u'^2}$ which is used in order to fix the inlet profile of the turbulent kinetic energy. A separate 1D channel flow simulation with fixed mean velocity and turbulent kinetic energy fields (equal to those determined from the experimentally provided data), has been used in order to determine an inlet profile for the length-scale determining variable, ϵ.

The wing surface and the lower wall (i.e. the body) both have a no-slip condition applied. The upper tunnel wall and side walls were modelled as slip (zero shear) walls in order to eliminate the need to resolve the tunnel wall boundary layers, while still including their blocking effect within the simulations. These walls were placed at $z/T = 6.3$ for the side wall (i.e. the wall parallel to the symmetry plane), and at $y/T = 3$ for the upper tunnel wall. In the experiment, a small gap was left between the end of the wing and the upper tunnel wall. This was done in order to eliminate the formation of secondary flows that would result from the interaction of the upper tunnel wall boundary layer with the wing. Such secondary flow features were desired only as a result of the lower wall boundary layer interaction, since otherwise the upper and lower secondary flow features may interfere with one another, especially given the relatively short span of the wing. In the computations, however, the need for such a gap has been eliminated since the upper wall has been modelled as a slip wall and therefore no upper-wall boundary layer will be present. Note that the presence of such a gap, if it were to be included in the computations, would significantly increase the domain complexity for a block-structured arrangement. The overset method however would be able to deal with such a domain topology with comparative ease. This work has however not been conducted since it offers very little toward solution accuracy away from the upper wall, yet would require the resolution of an additional boundary layer, thereby significantly increasing computational costs.

Since the geometry and boundary conditions are symmetrical, the computational domain is selected to cover only half the tunnel, with the axis of symmetry lying on the wing's cord. Symmetry boundary conditions were used for all boundary cell faces lying on the symmetry plane. The symmetry condition is implemented in the same way as for the zero shear wall considered earlier. Zero gradient conditions are applied for all primitive variables, while the wall normal component is set to zero.

The outlet has been placed sufficiently far from the wing so as to ensure it has no effect on the solution in the vicinity of the wing. A location of $x/T = 25$ has been found to be sufficient. Although the vortex is not fully dissipated by this location (and hence the use of zero gradient conditions at the outlet is strictly incorrect), the error that this induces does not propagate far upstream due to the directional nature of the convection process. To place the outlet far enough downstream for the vortex flow to have been dissipated by viscosity, leaving a 2D boundary layer, would be prohibitively expensive from a computational resource viewpoint.

10.4 Numerical model

The QUICK convection scheme is used for all velocity components, while the UMIST scheme is used for the turbulence transport variables (Section 3.3.2). The low Reynolds Launder-Sharma model is again used for turbulence closure (Chapter 4). A correction to the length scale is applied. The differential form of length scale correction described in Chapter 4 and [51] is employed. Inter-grid interpolation is conducted via MFBI interpolation.

10.5 Results and discussion

An overview of the solution is reported in Figures 10.6 and 10.7. The former figure considers pressure contours on the wing and groundplate walls, as well as secondary velocity vectors within the wake and vectors in the symmetry plane upstream of the leading edge. The latter figure illustrates the structure of the vortex via an iso-surface of the Q-criterion, [94], ($Q = 0.25[(\partial u_i/\partial x_j - \partial u_j/\partial x_i)^2 - (\partial u_i/\partial x_j + \partial u_j/\partial x_i)^2]$; a $Q = 0.5$ surface is shown). It can be seen from the figures that the flow is characterised by prominent secondary flow features. The vortex structure can be seen to wrap around the contour of the wing and is convected well downstream of the trailing edge before it is eventually dissipated via the action of viscosity. The overall shape of this vortex structure somewhat resembles a horseshoe, and hence is referred to as a horseshoe vortex.

Further insight into the overall flow field can be gained via Figures 10.8 and 10.9, which show, respectively, contours of the streamwise velocity component and turbulent kinetic energy in xz-planes at two spanwise locations; one above the vortex structure and the other through the wake vortex. It can be seen from Figure 10.8 that the wake is much shorter close to the groundplate than away from it. This is due to the vertical fluid flow in the vortex (i.e. the flow normal to the 'body'). Due to the rotation sense of the vortex, close to the symmetry plane the vertical flow acts to transfer high momentum fluid from the free stream downwards towards the lower wall, thereby 'filling' the wake. The opposite effect occurs on the other side of the vortex structure (away from the symmetry plane), where the vertical flow acts to transfer low momentum fluid from the lower wall's boundary layer upwards towards the free-stream, causing the relatively low streamwise velocity labelled in the figure.

In Figure 10.9 it can be seen that the turbulence energy is generally higher close to the groundplate than away from it, as is expected due to the mean shear caused by the lower wall. It can also be seen that the high mean shear generated due to the vortex structure contributes to the production of turbulence kinetic energy.

Flow streamlines above the vortex structure are presented in Figure 10.10. Notice the strong streamline curvature at the leading edge and also (but to a lesser extent) towards the trailing edge. It will be seen presently that this streamline curvature plays an important role in the vortex formation.

10.5.1 Pressure field results

Figure 10.11 shows contours of the pressure coefficient on the groundplate in the vicinity of the wing, with experimental comparison. It can be seen from the figure that a reasonable agreement is observed between the computation and experiment. The high pressure region at the leading edge of the wing is apparent from the figure, and is associated with the flow impingement onto the leading edge; the pressure rises in the region of a stagnation point in accordance with Bernoulli's principle. It can also be seen that the low pressure region around maximum wing thickness, associated with flow acceleration, is also captured reasonably well by the computation.

Figure 10.12 shows profiles of the pressure coefficient on the surface of the wing, at four different spanwise locations. It can be seen from the figure that there is virtually no spanwise variation in the surface pressure between $y/T = 1.72$ and $y/T = 1.46$ (this is observed both experimentally and computationally). On the other hand, there is modest variation between the top-most and bottom-most rows (located at $y/T = 1.72$ and $y/T = 0.13$ respectively) since the latter experiences slower moving fluid due to the lower wall boundary layer, and hence displays a smaller variation between positive stagnation and negative suction peaks due to the lower centrifugal force associated with the flow curvature. It can be seen that there is generally a very good agreement between computation and experiment for all rows considered. There is however a consistent discrepancy around $x/T = 0.75$ (i.e. at maximum thickness) for all four rows which is particularly apparent outside of the lower wall boundary layer. The computed pressure at $x/T \approx 0.75$ is systematically higher than that which is measured. The reason for this discrepancy is not fully clear, however it is possible that the measured pressure at this location is influenced by the trip wires, located a short distance upstream of maximum thickness. Circular cross section trip wires of not insignificant diameter (i.e. $0.014T$ diameter), have been used in the experiment. These are placed at $x/T = 0.61$. The measurements taken at $x/T \approx 0.75$ may therefore lie in the wake of these trip wires, and hence the pressure would be expected to be lower than would have been observed in the absence of the trip wires.

It can also be seen from Figure 10.12 that the experimental pressure measurements at $x/T \approx 0.75$ displays considerable asymmetry between the left and right sides of the wing (note that pressure measurements from both sides of the wing have been included where available, hence the presence of two pressure measurements at some x/T stations). Such asymmetry is not observed at other chord-wise locations. The difference in left and right measurements at $x/T \approx 0.75$ is larger than the reported uncertainty in the pressure measurements (i.e. $\delta C_P = \pm 0.005$, [89]), thereby somewhat undermining the validity of the experimental data in this region. Such asymmetry perhaps supports the hypothesis that the wake induced by the trip wires is the cause of the anomaly; small geometrical uncertainties in the location of the trip wire could be the cause of the asymmetry.

10.5.2 Symmetry plane mean velocity and turbulence results, upstream of leading edge

Figure 10.13 shows profiles of the streamwise velocity in the symmetry plane, at $z/T = 0$. It can be seen that the streamwise velocity magnitude is reduced with increasing x/T. This is due to the adverse pressure gradient this flow is subjected to on approaching the stagnation point at the wing's leading edge. It can also be seen from the figure that at and downstream of rake 6 ($x/T = -0.25$), the flow has separated from the lower wall. This is observed both experimentally and computationally, however the strength of the reversed flow is greater experimentally than is predicted by the computations at this rake location, suggesting that separation is predicted too late by the computations. The reason for this discrepancy is likely to be due to the failure of the turbulence model to account for the individual Reynolds stress components, as will be seen presently.

The vertical mean velocity component profiles are shown in Figure 10.14 (at the same rake locations as considered for the streamwise velocity). Here is can be seen that there is a general underestimate in the strength of the secondary flow. This is also consistent with the delayed separation prediction.

Figure 10.15 shows profiles of the streamwise normal Reynolds stress component at the same rake locations as those considered for the streamwise velocity profiles. It can be seen from this figure that this Reynolds stress component is generally underpredicted for all but the final two rakes. This is due to the turbulence model's failure in resolving the anisotropy of the turbulence, and instead assuming all three normal stress components equal to one another. In reality however, the streamwise component is significantly higher than $\overline{v'^2}$ and $\overline{w'^2}$ upstream of the stagnation point. This is due to the fact that the turbulence energy, generated by mean velocity gradients, is preferentially fed into the $\overline{u'^2}$ component since the dominant velocity gradient is $\partial\overline{u}/\partial y$. The reason for this preference can readily be seen by subtracting the RANS equations from the instantaneous Navier Stokes equations, and combining the results from the i and j direction momentum equations in such a way as to derive a transport equation for $\overline{u'_i u'_j}$. In so doing, the exact production term for the individual Reynolds stress components will become apparent:

$$P_{ij} = -\left(\overline{u'_i u'_k}\frac{\partial\overline{u}_j}{\partial x_k} + \overline{u'_j u'_k}\frac{\partial\overline{u}_i}{\partial x_k}\right) \tag{10.1}$$

The $\overline{w'^2}$ and $\overline{v'^2}$ stress components are lower than $\overline{u'^2}$ since the gradients of $\overline{v}$ and $\overline{w}$ are substantially smaller than $\partial\overline{u}/\partial y$. Instead these components are energised primarily by the redistribution process arising from the instantaneous pressure fluctuations. Additionally, the $\overline{v'^2}$ component is damped by kinematic blocking due to the presence of the lower wall. As such, $\overline{u'^2} > \overline{w'^2} > \overline{v'^2}$ would be expected, however this anisotropy is neglected in the computations.

Close to the stagnation region at the leading edge, it can be seen that the streamwise normal Reynolds stress component is now overpredicted rather than underpredicted as was seen further upstream (see Figure 10.15 rakes 9 and 10, $x/T = -0.10$ and $x/T = 0.05$ respectively). This is due to the fact that the flow experiences rapid deceleration due to the adverse pressure gradient ahead of the stagnation line. It is well understood that the eddy viscosity class of turbulence models tends to produce exorbitant values of turbulence energy in regions of high rates of strain [95, 96]. The reasons for this are partially due to an underprediction in the value of ϵ, leading to excessive turbulent viscosity, which in turn feeds into the production term, P_k, [95]. The $\overline{u'^2}$ stress component is also selectively damped due to the blocking effect of the wing wall; an anisotropic feature of the turbulence that cannot be depicted via the use of a single scalar (i.e. the eddy viscosity) which is essentially the approximation employed by eddy viscosity models.

In addition to the high strain rates and selective dampening of the $\overline{u'^2}$ stress component, the P_{33} term is negative for the real flow (since $\overline{w'^2}\partial\overline{w}/\partial z$ is positive), meaning that some energy is transfered from the turbulent fluctuations to the mean flow. This leads to a reduction of $\overline{w'^2}$ in the vicinity of the stagnation region, and subsequently of the remaining normal stress components via the redistribution process. This is not accounted for in the computations since the gradient $\partial\overline{w}/\partial z$ acts to increase the turbulence energy in the computed flow regardless of sign; there is no mechanism for accounting for the transfer of energy to the mean flow from the turbulent fluctuations within an eddy viscosity class of turbulence model.

While P_{33} is always negative at the leading edge for the present case, P_{11} and P_{22} may also be negative depending on the local gradients of $\overline{u}$ and $\overline{v}$ respectively. In such a situation, the negative production terms would act to reduce the turbulence kinetic energy for the real flow. However, when employing the eddy viscosity model the production of turbulence kinetic energy is again augmented regardless of sign (by a factor proportional to the mean velocity gradient squared). This leads to excessive turbulence energy predictions.

Figure 10.16 shows profiles of the Reynolds shear stress component, $-\overline{u'v'}$ in the symmetry plane, at the same rake locations as those considered for the mean velocity. It can be seen from the figure that the agreement between computation and experiment for rakes 1-4 is encouraging (i.e. for $x/T \leq -0.40$ - See Figure 10.17 for an enlarged scale plot of these profiles). This perhaps comes as little surprise since the turbulence production at these rake locations is dominated by the $\partial\overline{u}/\partial y$ velocity gradient, and the flow deviates only slightly from simple shear flow for which the model is known to perform well (particularly given simple shear is one of the cases used to calibrate the model's coefficients). As the rake progress further downstream however, there is a significant discrepancy. It can be seen from the figure that there is a large growth of the measured stress $-\overline{u'v'}$ close to the lower wall at rakes 5-7, that is not captured in the computations. This is due to the fact that flow separation is predicted too late by the computational model. In the recirculating flow, the turbulence producing mean velocity gradients are large. Since in the computation the recirculation region is predicted too late, the turbulence levels are initially underestimated. On the other hand, rakes 9 and 10 show a general overprediction in the Reynolds shear stress component, $-\overline{u'v'}$, which is again due to the stagnation point anomaly described earlier. The eddy viscosity at these rake locations is excessively high due to an underestimate of the dissipation rate around the stagnation region, leading to excessive predictions of the Reynolds shear stress.

The fact that the separation point is predicted too late in the computational model is unlikely to be attributable to turbulence shear stress levels, since there is good agreement between computation and experiment upstream of the measured separation point (e.g. rakes 1-4, Figure 10.17). Instead one must look toward the normal stress component as the possible culprit. It can be seen from Figure 10.18 that the measured normal stress component grows rapidly with increasing x upstream of the measured separation point. This adverse normal stress gradient contributes to the adverse pressure gradient so as to induce separation earlier than would have been observed in its absence. This effect is not captured in the computations due to the failure of the model to provide anisotropic normal stress predictions. Separation therefore occurs later in the computational model than is measured.

10.5.3 Flow features at maximum wing thickness

Figure 10.19 shows profiles of the streamwise velocity component at several vertically aligned rakes lying in an xy-plane which intersects the maximum wing thickness (i.e. $x/T = 0.75$). One of the most notable features that can be seen from the figure is that the experimental velocity profile displays a marked 'kink' close to the groundplate for rakes $5 - 8$. This is not captured satisfactorily in the computations. The reasons for this kink is likely to be due to the momentum transfer posed by the vertical flow associated with the vortex. At the outside of the vortex (away from the symmetry plane) the vertical flow acts to transfer low momentum fluid in the lower plate boundary layer away from the lower wall, thereby retarding the streamwise fluid away from the wall. Conversely, on the inner portion of the vortex (i.e. closest to the symmetry plane) the vertical flow acts to enhance the streamwise flow in the lower wall boundary layer, causing the observed kinks and also increasing the wall shear stress. This process is not fully captured in the simulations since the simulated vortex structure is smaller and weaker than that of the experiments due to the delayed separation at the leading edge.

In addition to the momentum transfer by the mean secondary flow, the turbulence fluctuations are also likely to play a significant role in the 'kink' development observed in Figure 10.19. The Reynolds shear stress component $-\overline{u'v'}$ is likely to be the dominant stress contributing to such a kink; this stress is contributed to via the large gradients of the vertical mean velocity component in the vortex flow (in addition to streamwise gradients that remain present with or without the vortex). Unfortunately this Reynolds stress component is not presented in the experimental data, hence support of this is unavailable.

Figure 10.20 shows profiles of the cross-flow velocity component at the same rake locations as were considered in Figure 10.19. It can be seen from the figure that there is a general underestimate of the strength of the secondary flow, again suggesting a smaller and weaker vortex structure is predicted than is observed experimentally. This is consistent with the delayed separation ahead of the wing leading edge just reported in Section 10.5.2. Since the initial vortex size (and strength) in the symmetry plane is underpredicted, the vortex structure a short distance downstream of this initial vortex will also tend to be underpredicted (since there is insufficient time for the vortex to have grown significantly via secondary fluid motions).

The secondary flow observed in Figure 10.20 is augmented by the way in which the lower wall boundary layer interacts with the wing. There is a strong streamline curvature upstream of the wing's leading edge as the flow conforms to the wing contour. A pressure gradient acting towards the centre of curvature (i.e. away from the wing wall) is therefore established in order to balance the centrifugal force associated with this streamline curvature. This pressure gradient is reasonably uniform in the vertical direction (this is due to the fact that the pressure gradient in the vertical direction must be small enough so as not to induce spurious vertical fluid motions). Since the fluid close to the lower wall has a lower velocity (due to it being within the lower wall boundary layer), yet it experiences much the same radial pressure gradient as the essentially inviscid fluid above, the radius of curvature of the streamlines within the lower wall boundary layer must be lower than that of the fluid above in order for the pressure and centrifugal forces to balance. It is this sharper turning close to the ground plate, relative to the fluid above, that causes the secondary flow.

There is a second mechanism for the generation of secondary flow within a turbulent flow. At the junction between the wing and the lower wall, normal stress components are selectively damped due to the presence of the various walls, thereby causing a large normal stress anisotropy near the junction. On approaching the wing surface, the $\overline{w'^2}$ stress component is selectively damped, causing a transport of momentum towards the wing surface. Similarly, on approaching the lower wall, $\overline{v'^2}$ is selectively damped, causing a transport of momentum towards the lower wall. The net effect is a net transport of momentum towards the junction corner, with typically twin counter-rotating vortices. Since there is a single vortex in the present case, rather than the twin vortices that would be obtained via this mechanism, one can conclude that predominant mechanism of secondary flow generation for the present study is due to the streamline curvature. However, it is possible that normal stress anisotropy could have a noticeable effect on the vortex shape. Since the anisotropic nature of the turbulence is not captured in the present simulations, this effect will not be present in the computed results, possibly explaining some of the discrepancy between results observed.

In Figure 10.21 the Reynolds normal stress component $\overline{u'^2}$ is shown at the same rake locations as above (i.e. at maximum wing thickness). It can be seen from the figures there is a general underprediction of the normal stress; partly due to the anisotropy reasons outlined earlier. The large discrepancy in the core of the vortex is likely to be enhanced by the use of a length scale correction term in the present study, which acts to reduce the turbulence levels in separated flows. While normal stress predictions may have been impaired through the use of such a correction, the more influential shear stress predictions are likely to be better than would have been obtained in the absence of its use.

Figure 10.22 shows the shear stress component, $-\overline{u'w'}$. It can be seen from the figure that the shear stress prediction is reasonable (while there are clearly major differences between computation and experiment, this is to be expected due to the simplicity of the turbulence model).

10.5.4 Flow in the wake

It was seen in Figure 10.10 that, as well as upstream of the wing's leading edge, the flow streamlines also exhibit reasonably strong curvature a short distance downstream of the wing's trailing edge (since the streamlines must realign with the x-direction vector in order to avoid crossing the symmetry plane). This additional curvature acts to augment the secondary flow via the process described in the previous section. The increased secondary flow acts to strengthen the vortex causing the rapid expansion in the vortex structure that was observed in Figure 10.7.

Figure 10.23 shows velocity vectors and contours of the Q-criterion, in the plane $x/T = 6.38$, both clearly illustrating the vortex.

Figure 10.24 shows profiles of the streamwise velocity component at rakes taken through the vortex, at $x/T = 6.4$ (i.e. half a chord length downstream of the wing's trailing edge). It can be seen from the figure that the 'kink' that the streamwise velocity component experiences (which was also seen further upstream and is due to the momentum transfer by the secondary flow), is enhanced relative to that which was observed at maximum wing thickness. This is due to the enlarged vortex structure in the wake relative to that at maximum thickness. Again, the computed profile does not capture this kink particularly satisfactorily at Profile 2, suggesting a different vortex structure is predicted in the wake. The transfer of momentum by the secondary flow has limited cross-stream range, as is indicated by Figure 10.8. The locations at which the experimental data was taken lie between the location at which the predicted secondary flow momentum transfer is greatest (which is located between Profiles 2 and 3). The fact that a different vortex structure is observed is unsurprising given the differences in the structure previously discussed further upstream. The discrepancies further upstream, due predominantly to the incorrect separation point at the leading edge, are convected downstream.

Figure 10.25 shows profiles of the cross-stream velocity component at the same rake locations as were considered in Figure 10.24. It can be seen that a fairly good agreement is found between computation and experiment at the majority of rakes, although away from the symmetry plane (e.g. Profiles 1 and 2), the strength is underestimated by a fair margin, suggesting an underestimate in the vortex size. This finding is consistent with the lack of the 'kink' at Profile 2 that was seen in Figure 10.24.

In Figure 10.26, profiles of the streamwise normal stress component in the wake are reported. It can be seen that the experimental normal stress levels away from the symmetry plane shows a 'Z' shape profile that is not satisfactorily captured in the computation. In the experiment away from the symmetry plane, the P_{33} term is generally positive close to the lower wall (since $\overline{w'^2}\partial\overline{w}/\partial z < 0$). Further from the wall, the opposite is true (i.e. over the upper portion of the vortex). It is this effect that causes the 'Z' shaped profiles of $\overline{u'^2}$ away from the symmetry plane (note that P_{33} affects the streamwise normal stress component via the redistribution process).

The production term P_{22} also plays a role in the shape of the normal stress profiles via the redistribution process. The affect of P_{22} away from the symmetry plane is to reduce the turbulence energy close to the wall (since $\overline{v'^2}\partial\overline{v}/\partial y > 0$), while increasing it over the upper portion of the vortex.

However, since the gradients of the vertical mean velocity are prominent only over the sides of the vortex structure, P_{22} is significant only over some of the rakes. Also, since the $\overline{v'^2}$ stress component is damped by the presence of the lower wall, the affect of P_{22} is diminished relative to P_{33}.

Since the aforementioned factors are not accounted for in an eddy viscosity model, it is unsurprising that the 'Z'-shape profile is not captured in the computations.

For completeness, profiles of the Reynolds shear stress component $-\overline{u'w'}$ in the wake are presented in Figure 10.27. Here it can be seen that the agreement is reasonable over most of the profiles. There is however an underprediction in the turbulence levels close to the symmetry plane and away from the lower wall (at $z/T = -0.1$). The reason for this may be due to an underprediction in the width of the wake away from the lower wall (i.e. above the vortex). In this situation, the profile taken through $z/T = -0.1$ may be lying on the edge of the computed wake, while going through the experimental wake, thereby explaining the discrepancy. Possible reasons for an underprediction of the width of the wake include the performance of the $k - \epsilon$ class of models in regions of adverse pressure such as that encountered at the trailing edge of the wing. The excessive turbulence levels that are predicted in this situation act to 'fill' the wake rapidly as was discussed in detail in Chapter 8.

10.6 Conclusion

The turbulent flow around a wing body junction has been computed. It has been shown a good agreement between computation and experiment is obtained for the pressure field. Reasonable predictions of the mean velocity components have also generally been achieved (although there is some moderate discrepancy at places). The leading edge separation point, however, is not well captured, and consequently the downstream vortex structure is different to that of the experiments.

The discrepancy in the leading edge separation point location is attributable to the simplicity of the turbulence model. The use of an eddy viscosity model, and its subsequent failure to provide anisotropic normal stress predictions for complex flows involving impingement, is therefore not recommended. A more appropriate selection would perhaps be some variant of a second-moment closure (e.g. Refs. [97, 98]) or even a LES simulation. However, given the purpose of the present computations is not to provide quantitative predictions of the flow-field, but rather to investigate the applicability of the overset method in dealing with such a geometry, the use of a simpler turbulence model is justifiable here.

The overset grid method has been shown to provide good solution continuity over the interface. The method is useful even for this relatively simple geometry since Cartesian meshes cover the majority of the domain.

10.7 Figures

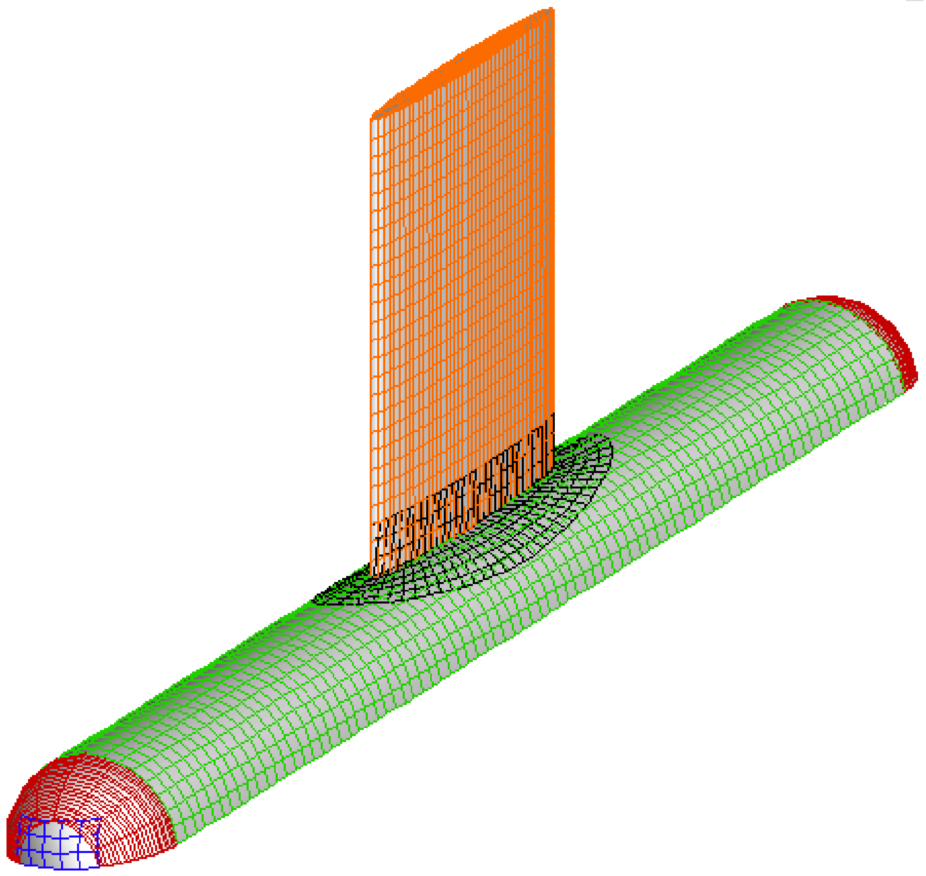


Figure 10.1: Sample grids used for the hole cutting of a generic wing-fuselage junction junction problem.

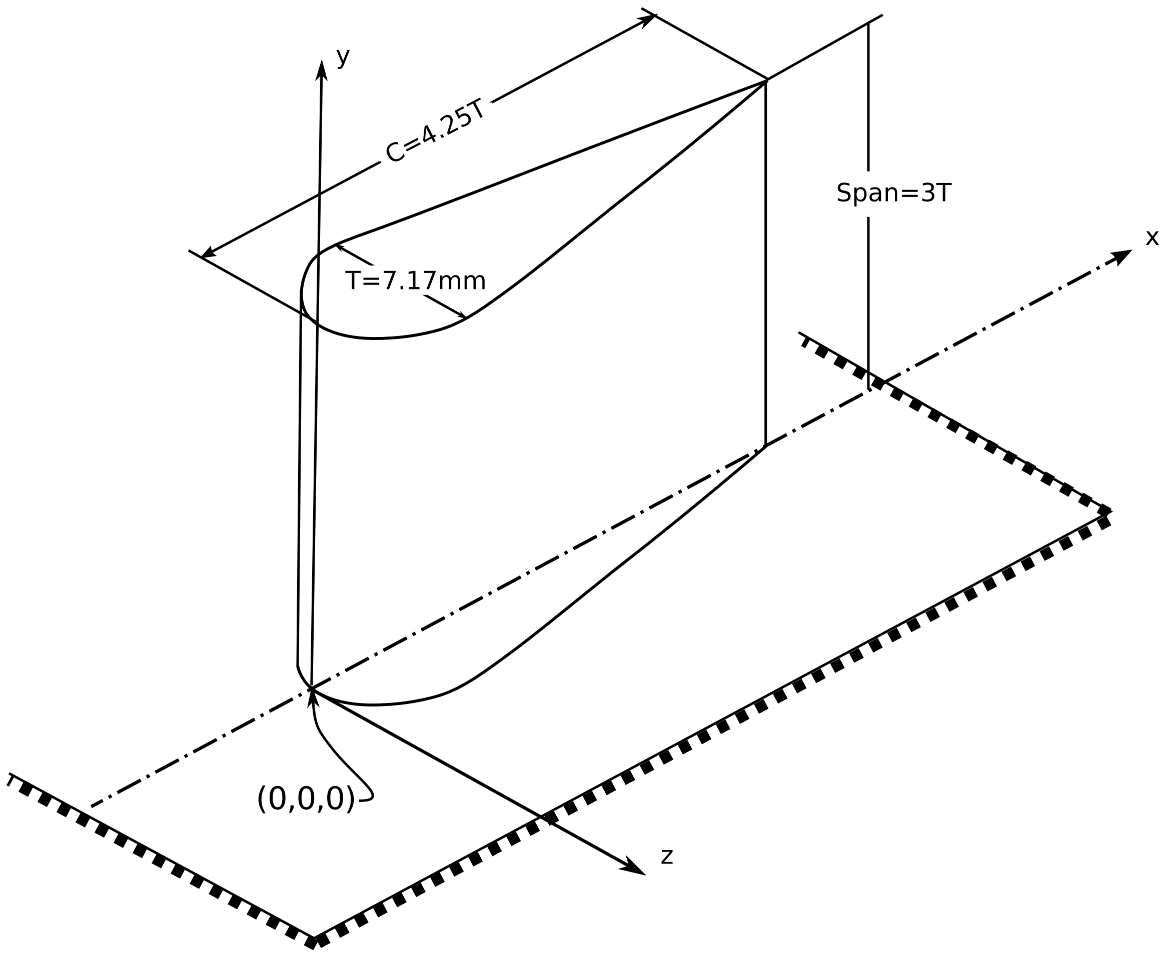


Figure 10.2: Junction geometry and right hand coordinate system used. Origin located at the junction between the wing's leading edge and the groundplate.

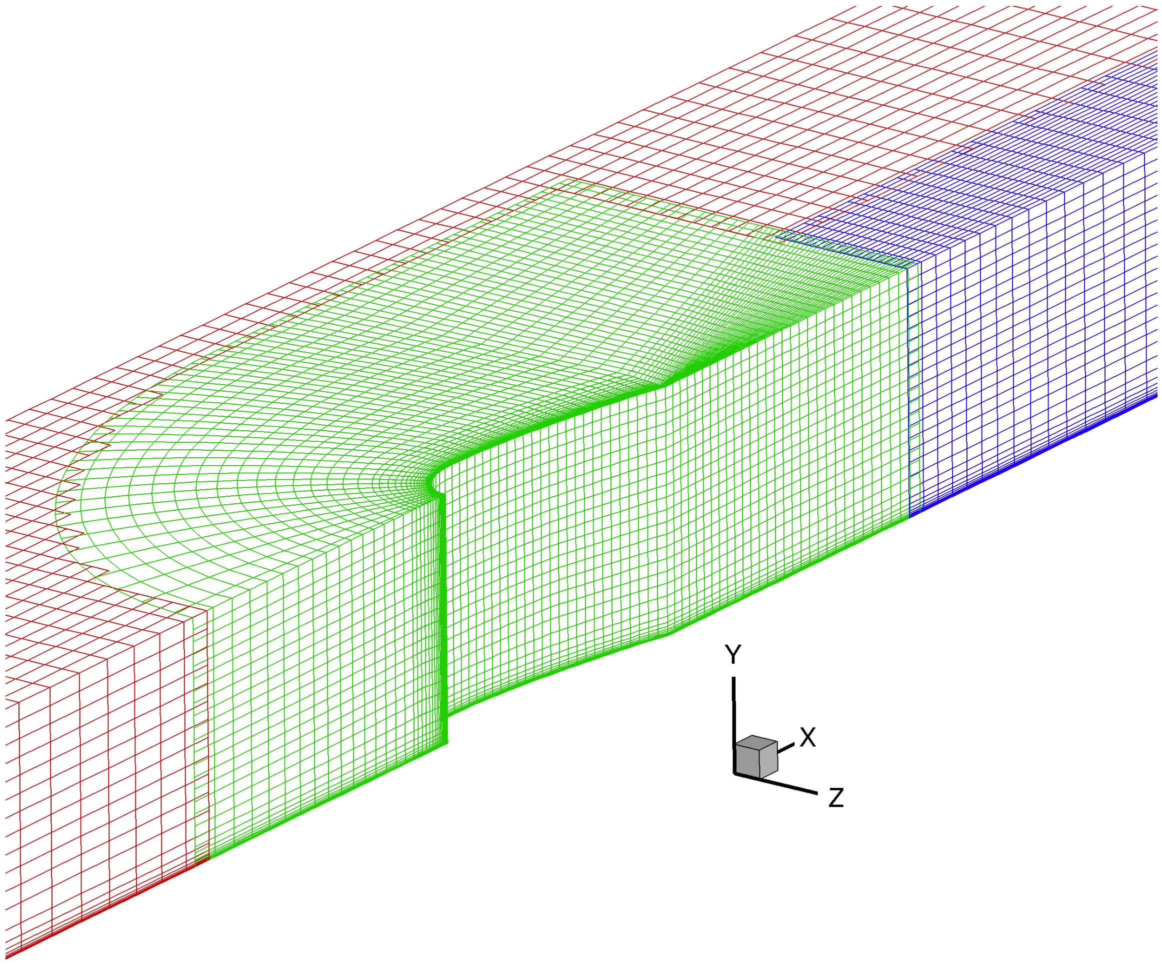


Figure 10.3: Computational grids used. Domain decomposed via the use of three overset grids.

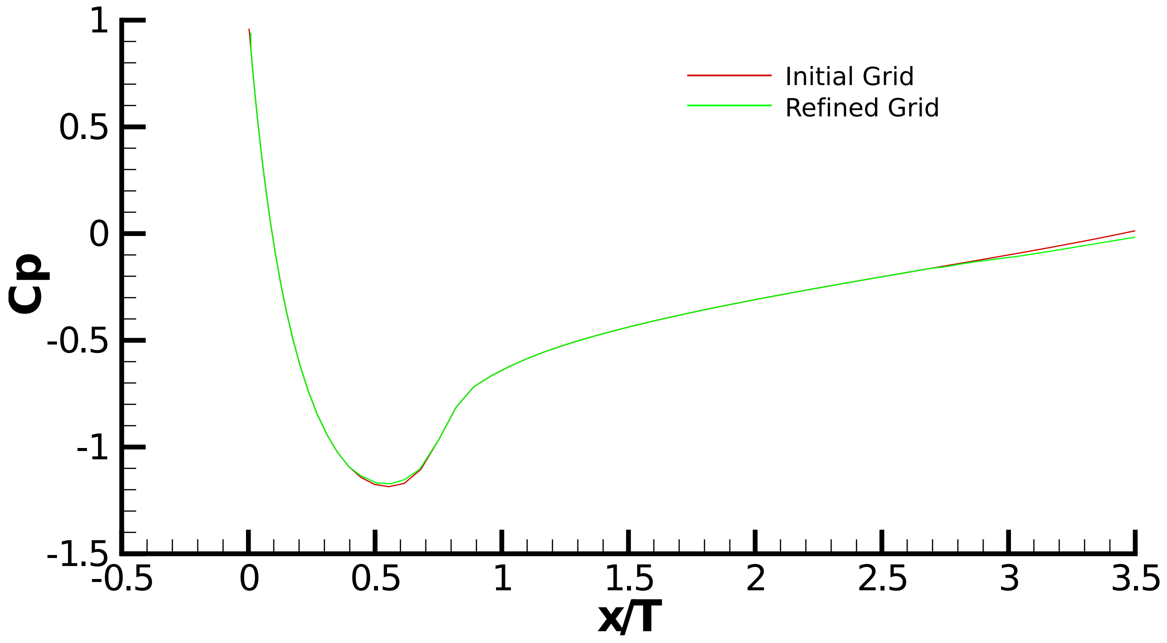


Figure 10.4: Contours of pressure coefficient at the spanwise location $y/T = 2.0$ for the two grid resolutions considered.

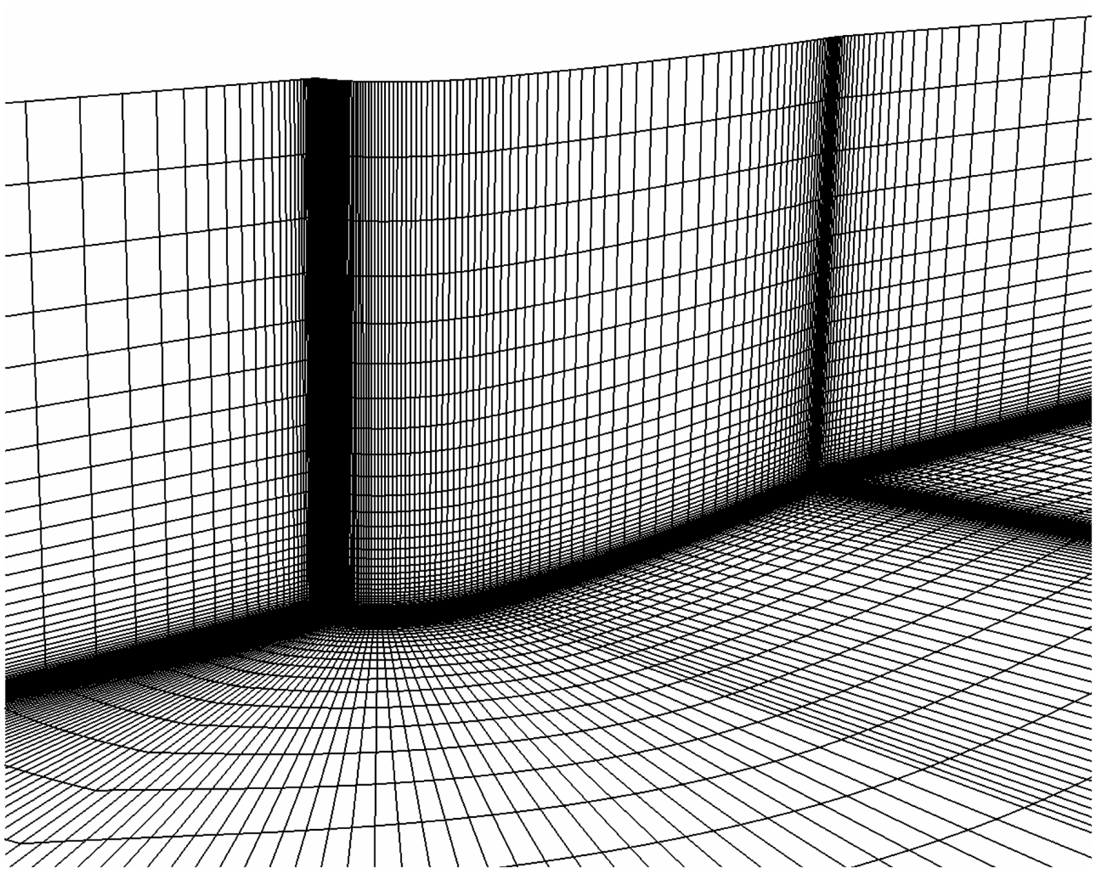


Figure 10.5: Grids used in [93].

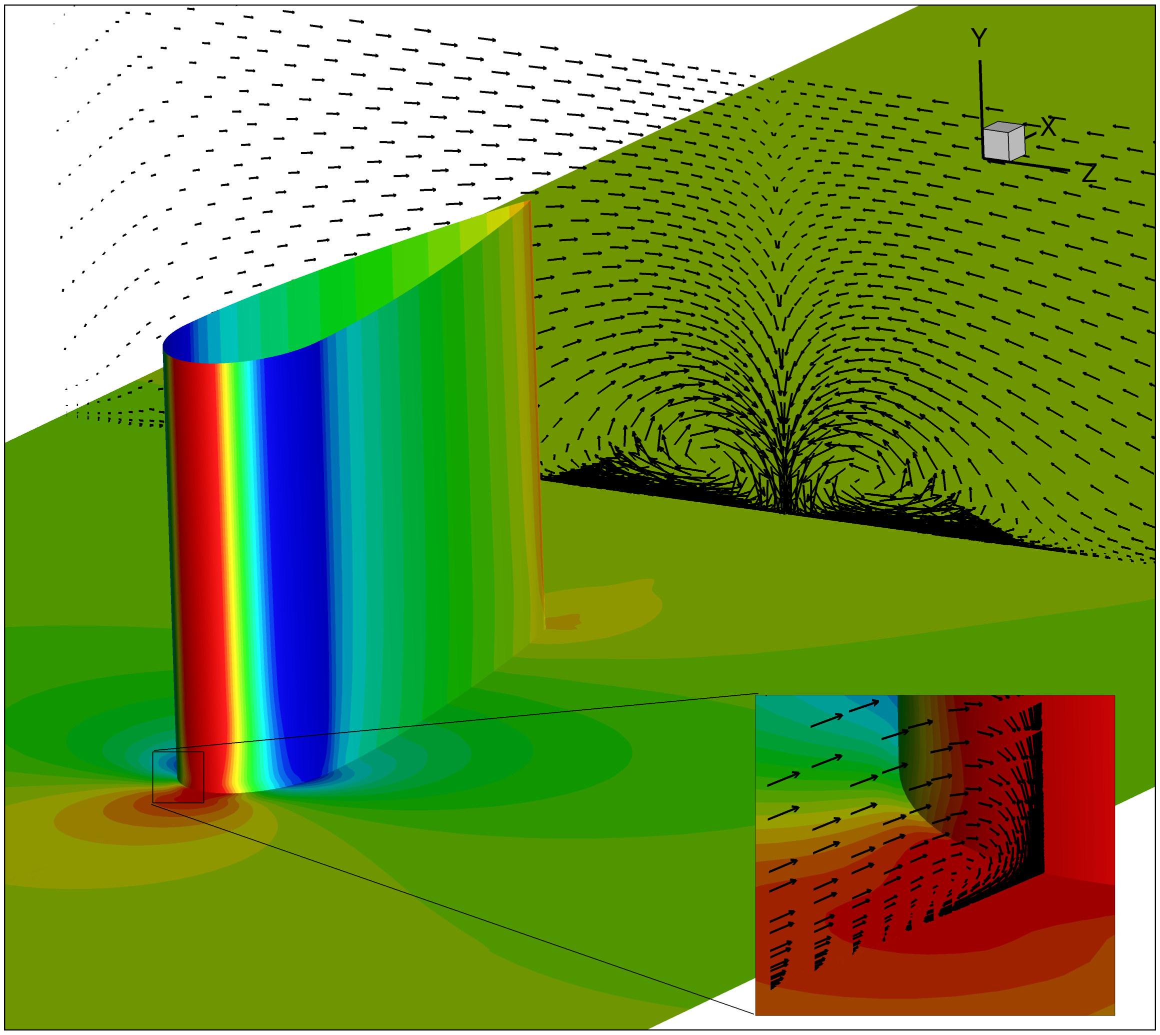


Figure 10.6: Contours of pressure on wing and groundplate walls (scale increasing blue to red), with secondary velocity vectors in the wake and symmetry plane (not drawn to same scale).

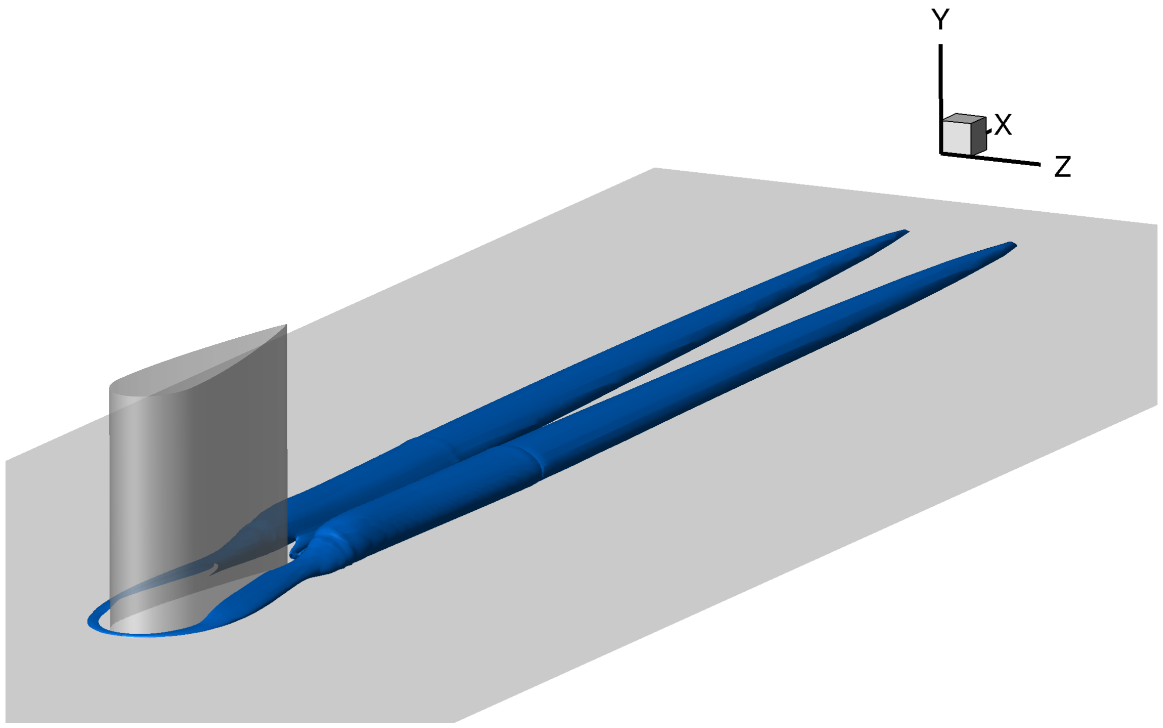


Figure 10.7: Iso-surface of $Q = 0.5$, showing the vortex structure. Note that translucency has been applied to the surface of the wing.

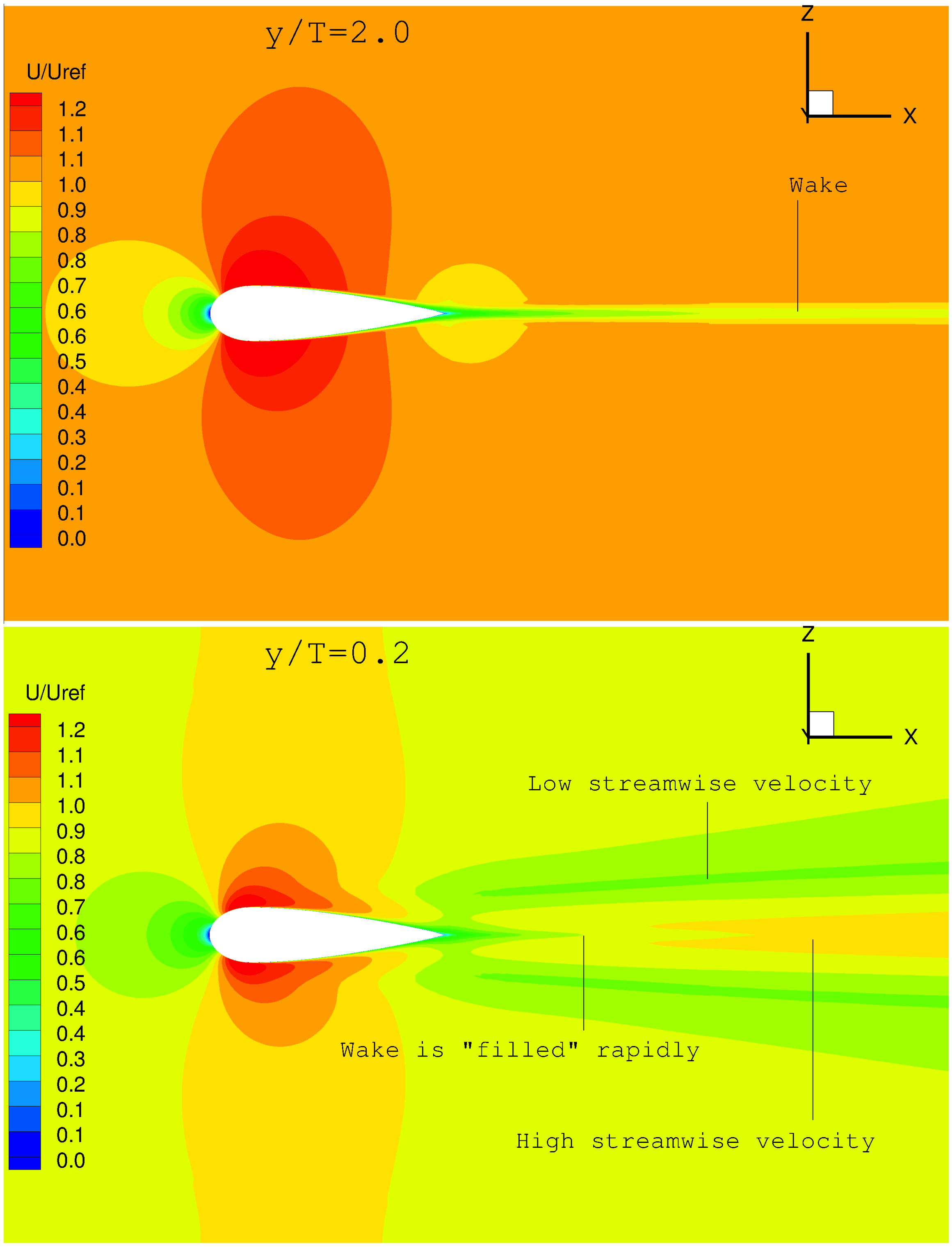


Figure 10.8: Contours of the streamwise velocity component, U, at two spanwise locations, $y/T = 2.0$ (above) and $y/T = 0.2$ (below).

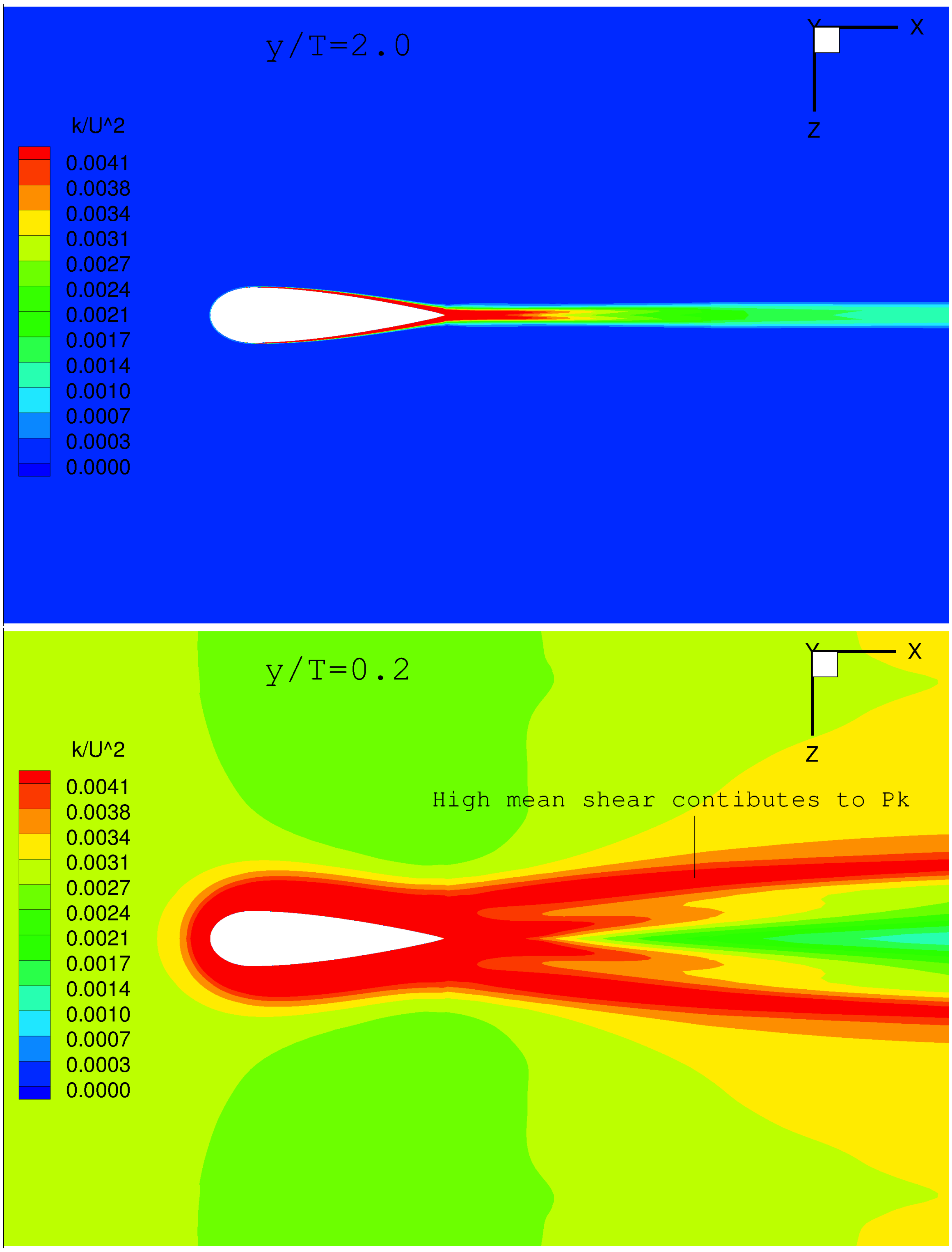


Figure 10.9: Contours of turbulent kinetic energy at two spanwise locations, $y/T = 2.0$ (above) and $y/T = 0.2$ (below).

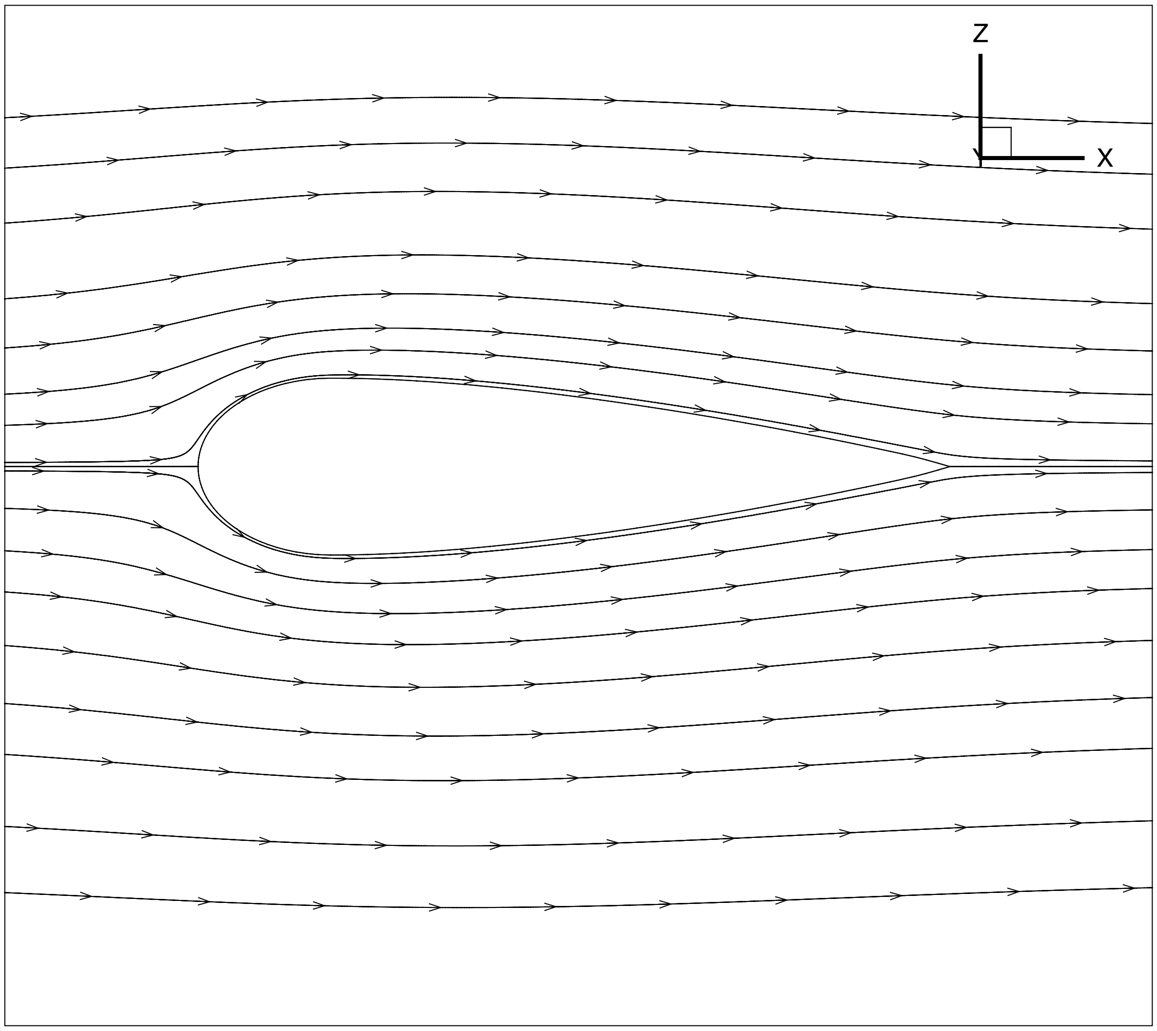


Figure 10.10: Flow streamlines at $y/T = 2$.

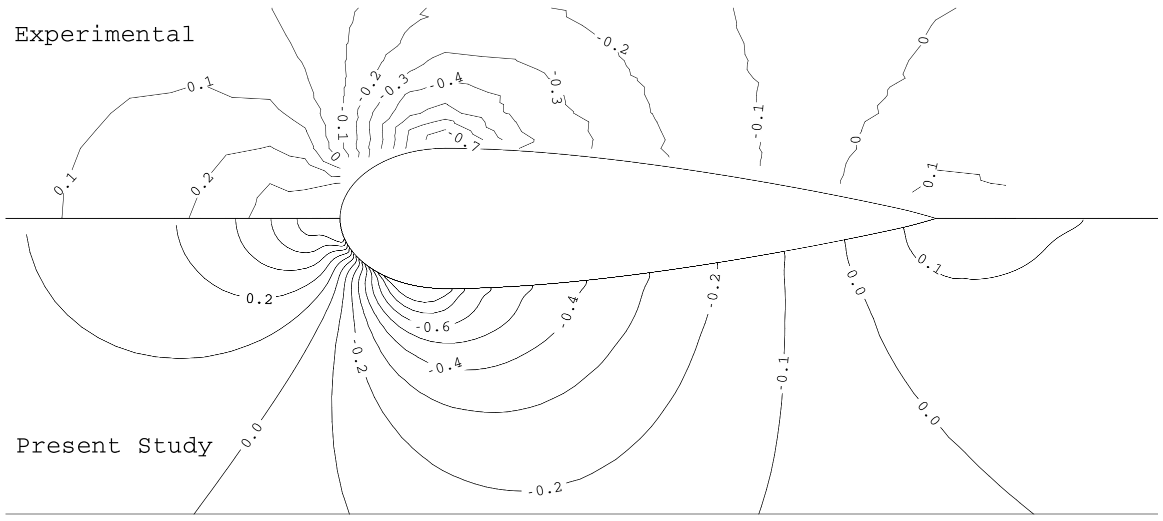
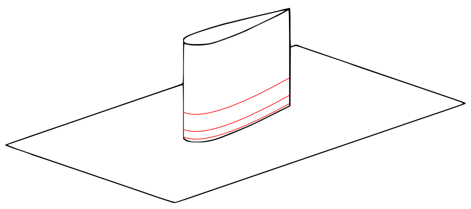
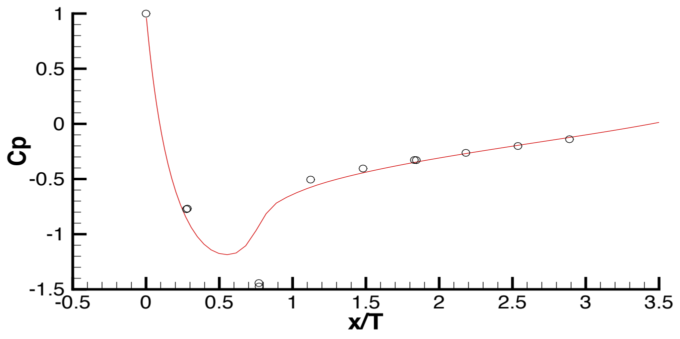
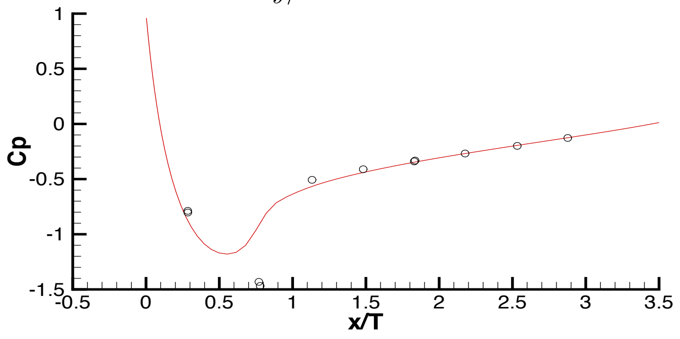


Figure 10.11: Contours of pressure coefficient, $C_p = 2(P - P_\infty)/(\rho U_\infty^2)$, on the lower wall, with experimental comparison.

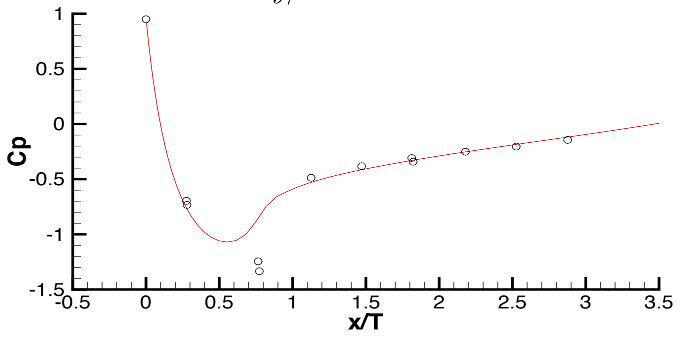




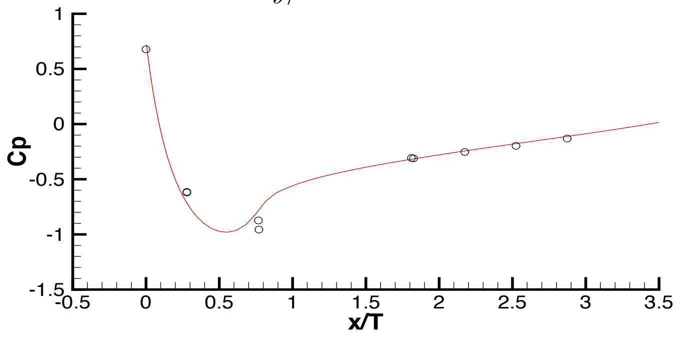
$y/T = 1.72$



$y/T = 1.46$



$y/T = 0.40$



$y/T = 0.13$

Figure 10.12: Profiles of pressure coefficient on the wing surface at four spanwise locations, $y/T = 1.72$, 1.46, 0.40 and 0.13.

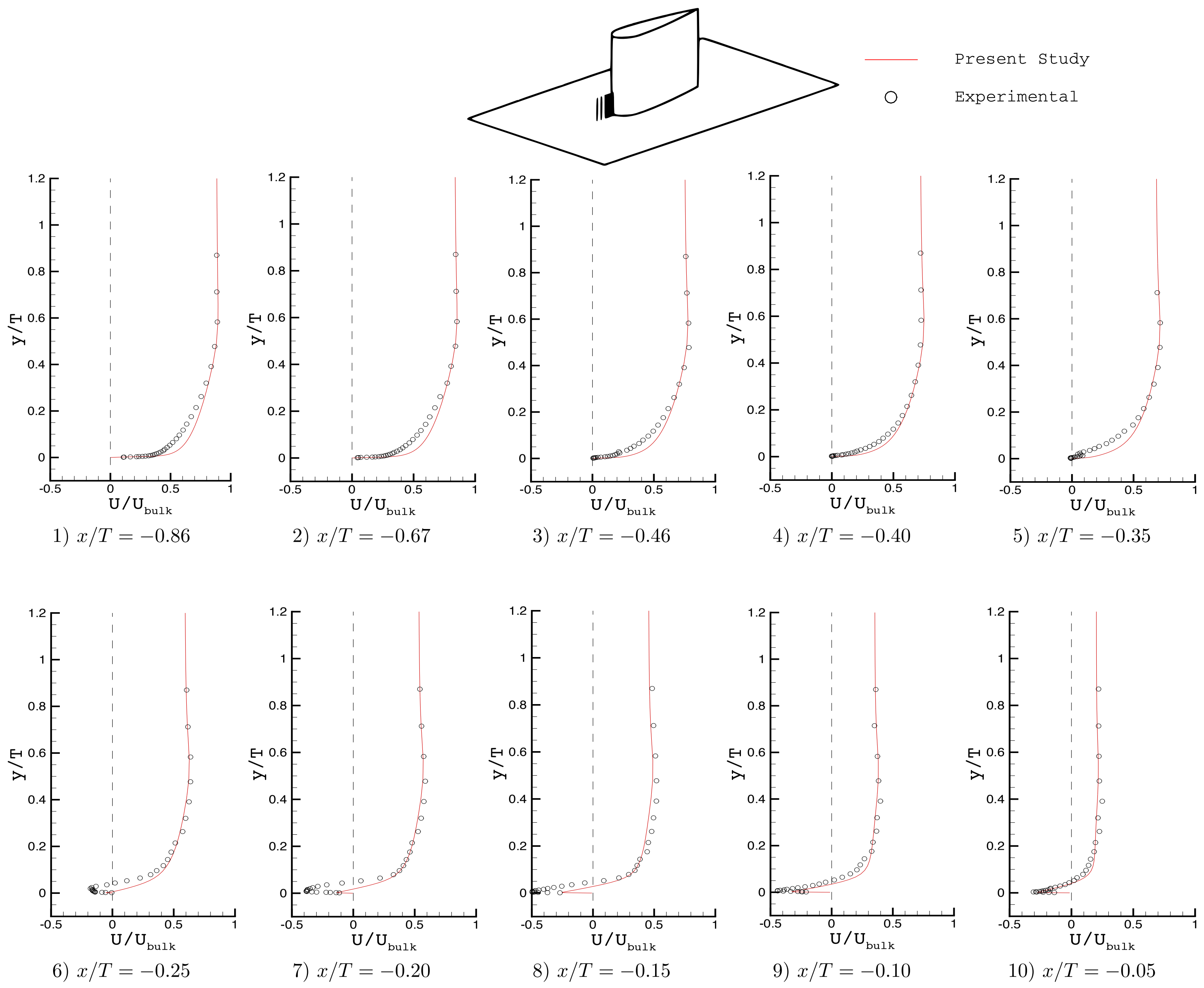


Figure 10.13: Normalised streamwise velocity component in the symmetry plane ($z/T = 0$)

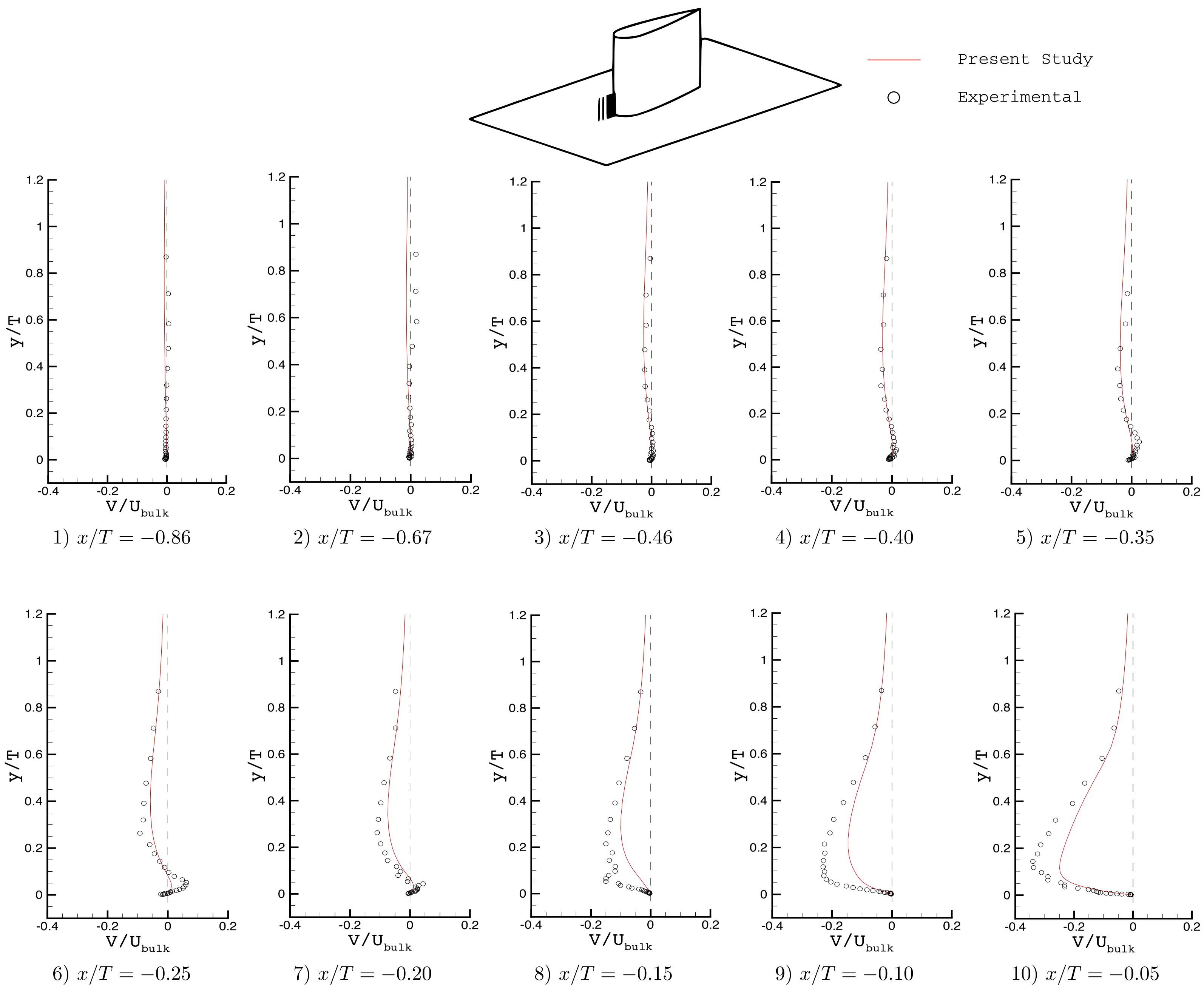


Figure 10.14: Vertical velocity component, V, in the symmetry plane ($z/T = 0$)

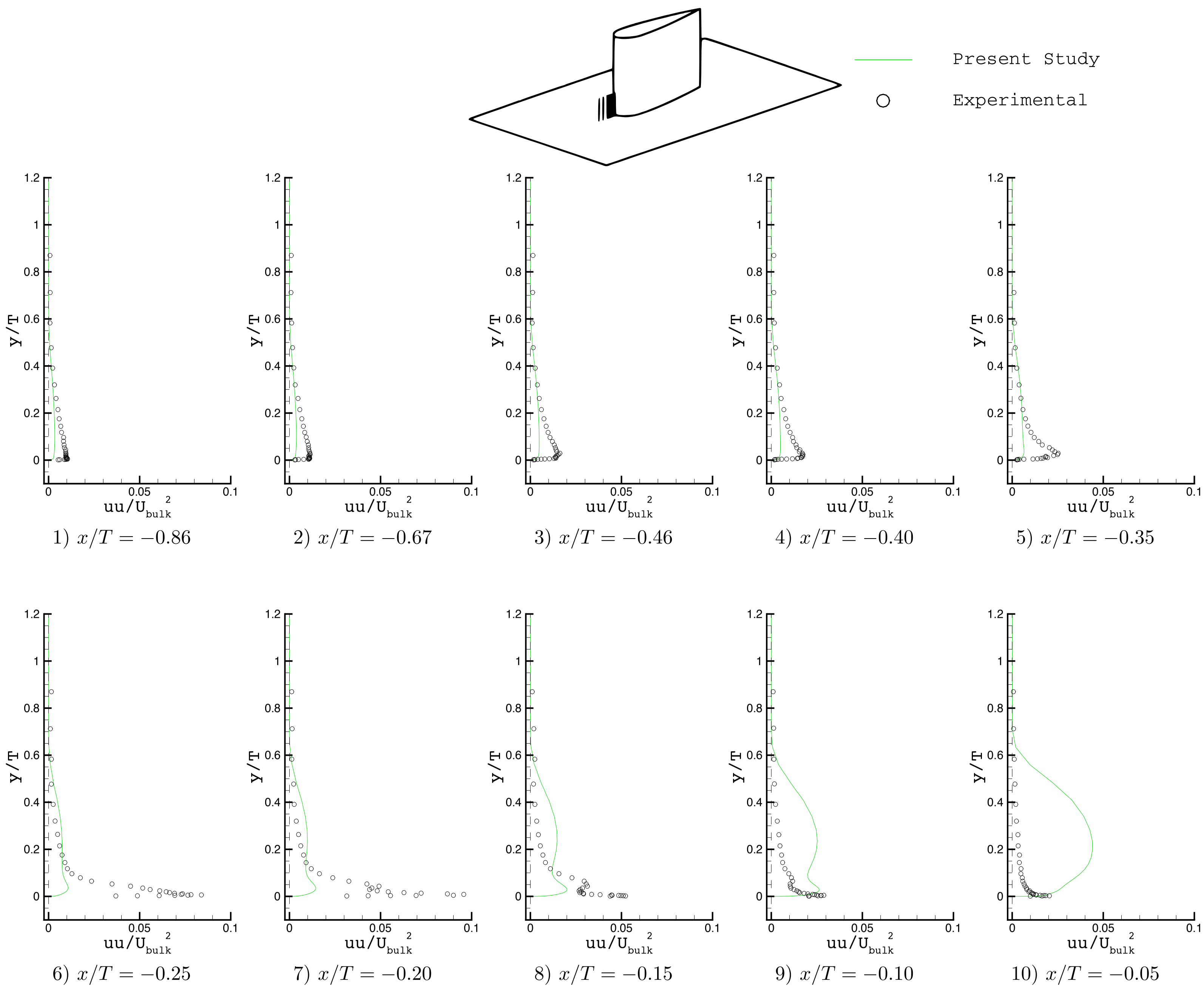


Figure 10.15: Streamwise normal Reynolds stress component, $\overline{u'u'}$, in the symmetry plane ($z/T = 0$)

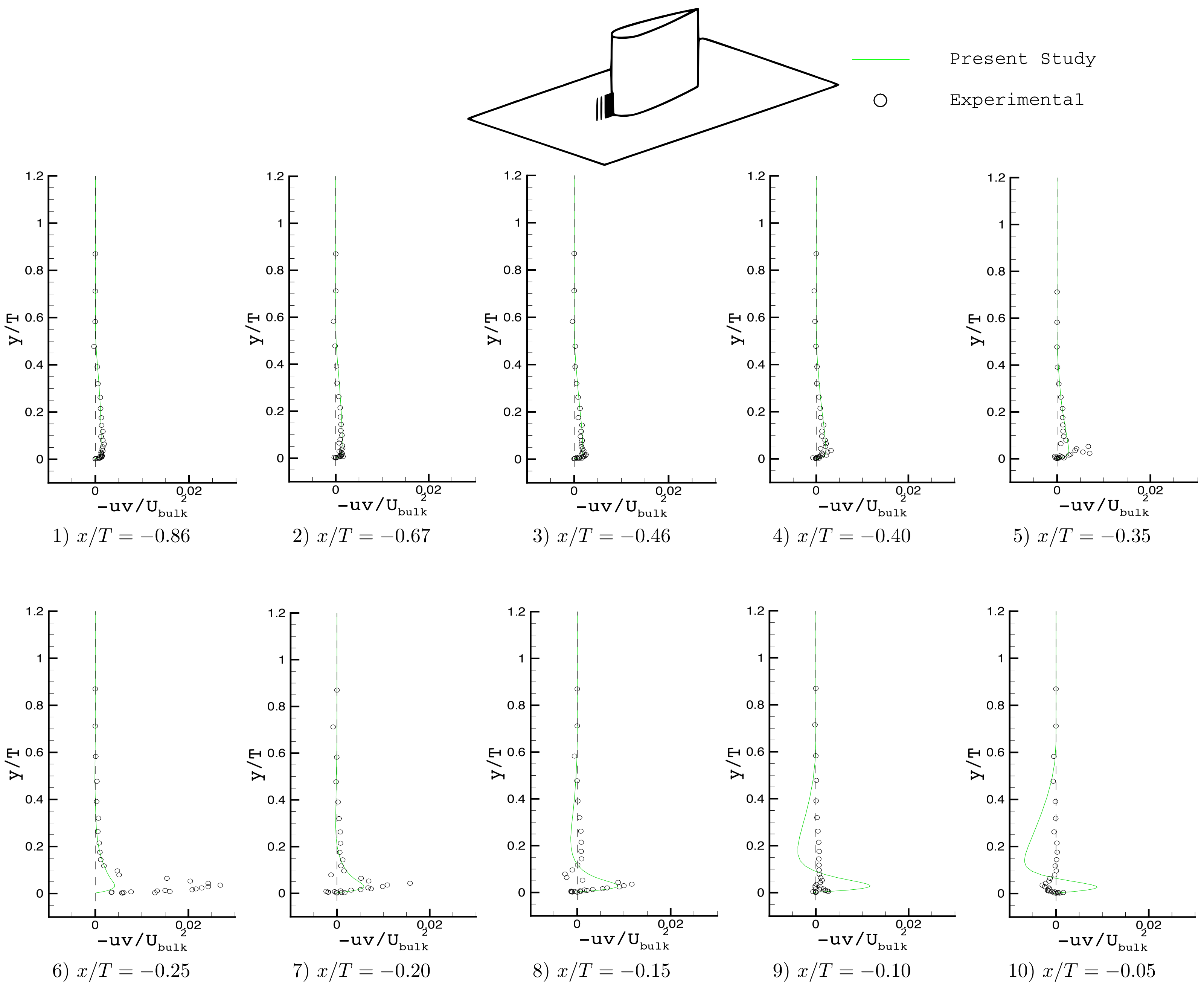


Figure 10.16: Reynolds shear stress component, $-\overline{u'v'}$, in the symmetry plane ($z/T = 0$)

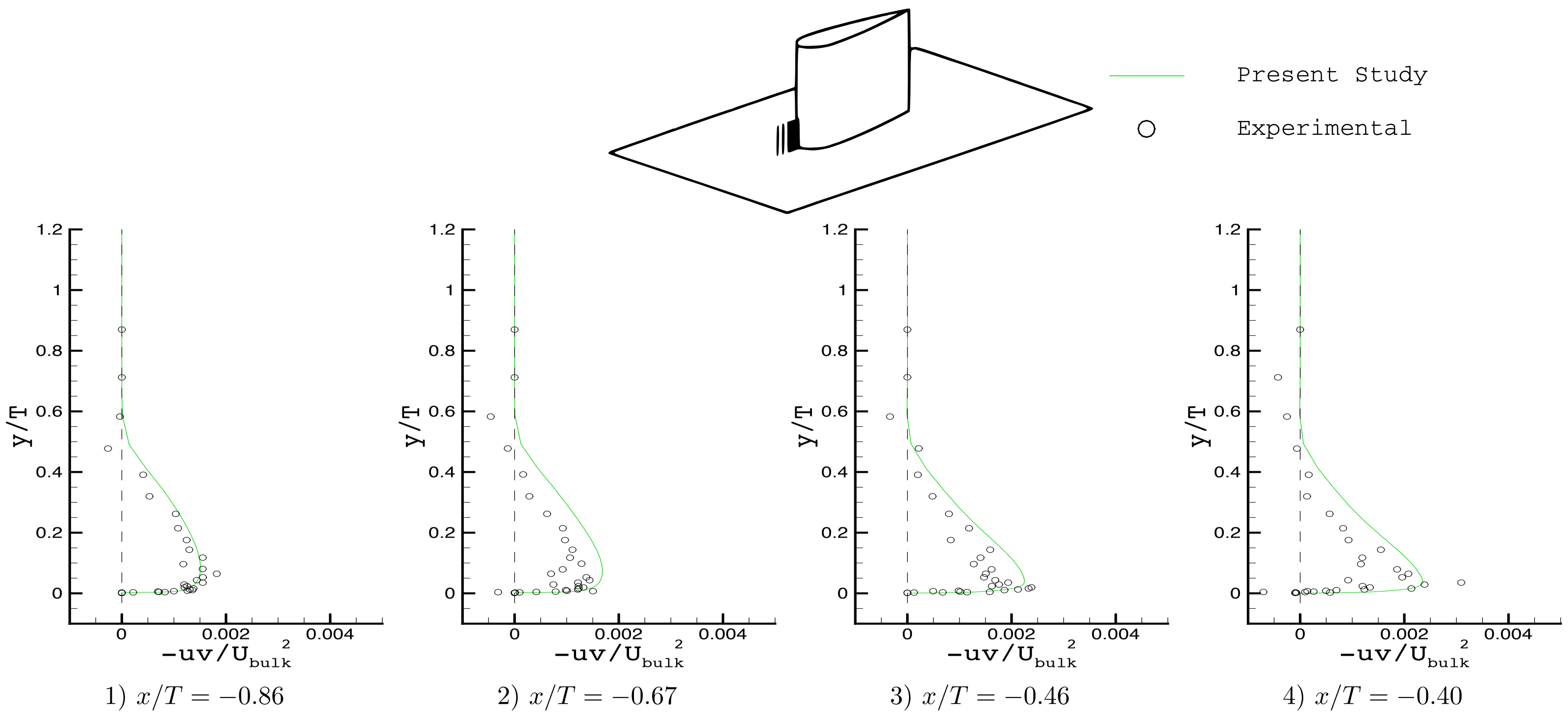


Figure 10.17: Enlargement of the first four profiles of Figure 10.16. Stress component, $-\overline{u'v'}$, in the symmetry plane ($z/T = 0$)

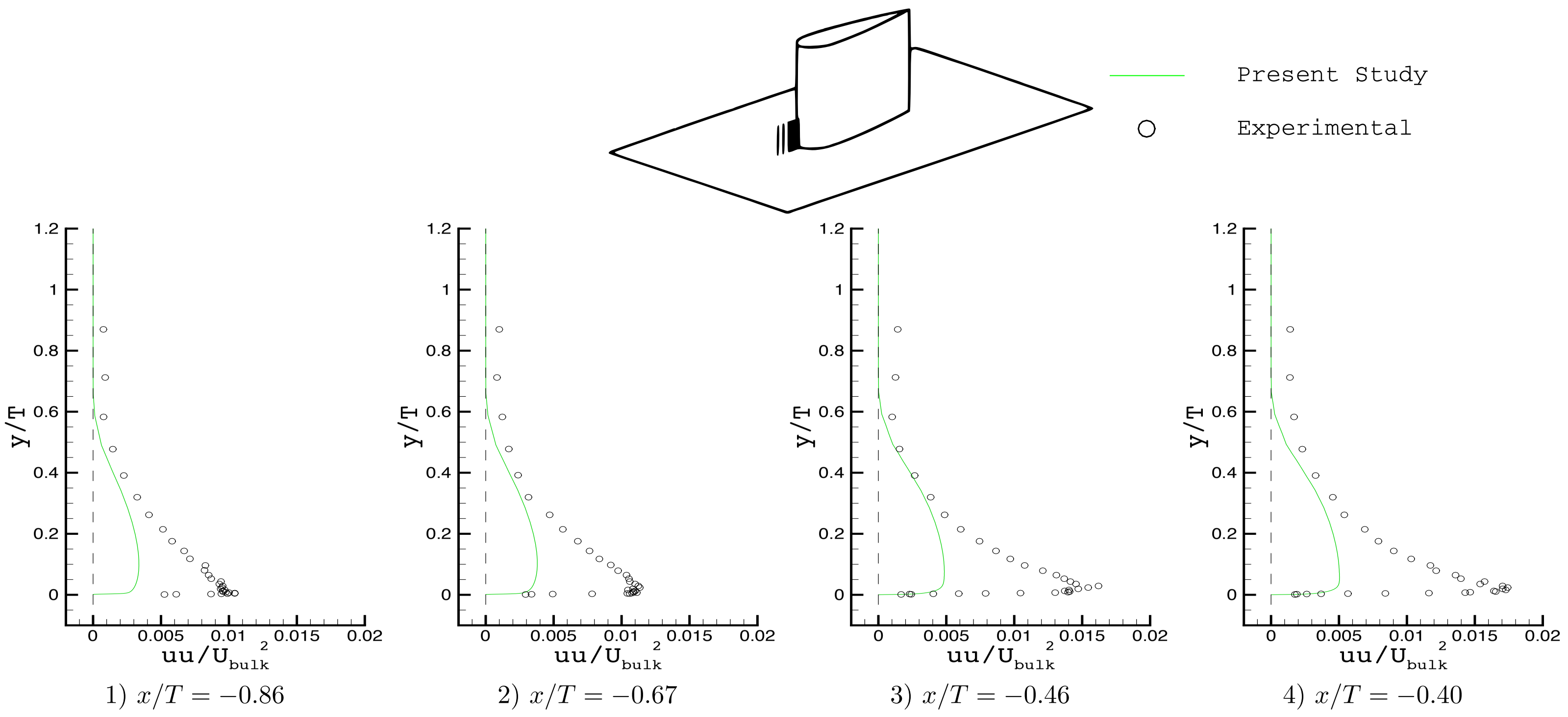


Figure 10.18: Enlargement of the first four profiles of Figure 10.15. Stress component, $\overline{u'^2}$, in the symmetry plane ($z/T = 0$)

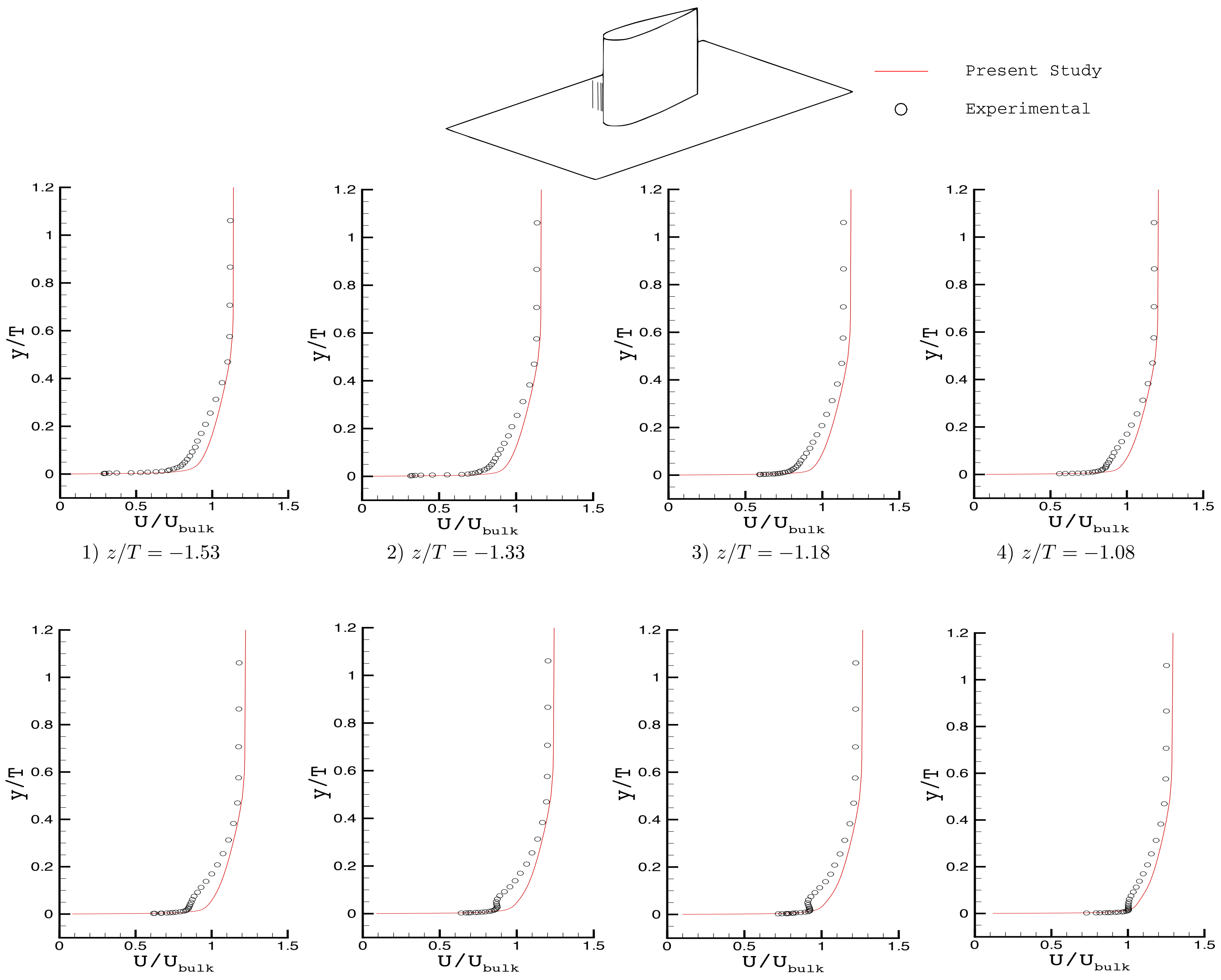


Figure 10.19: Normalised streamwise velocity component, U, in a yz-plane at maximum wing thickness ($x/T = 0.75$)

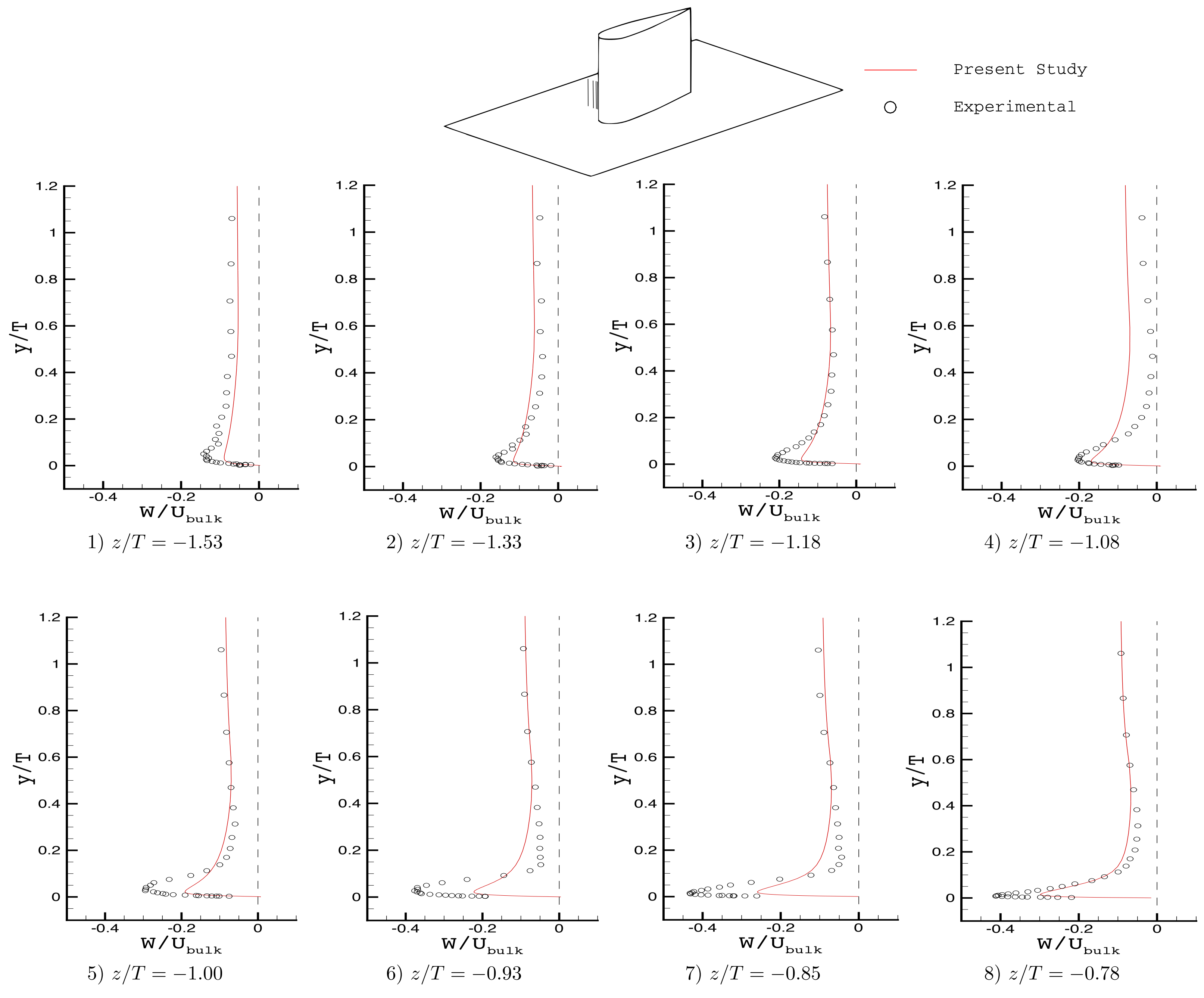


Figure 10.20: Normalised cross-stream velocity component, W, in a plane at maximum wing thickness ($x/T = 0.75$)

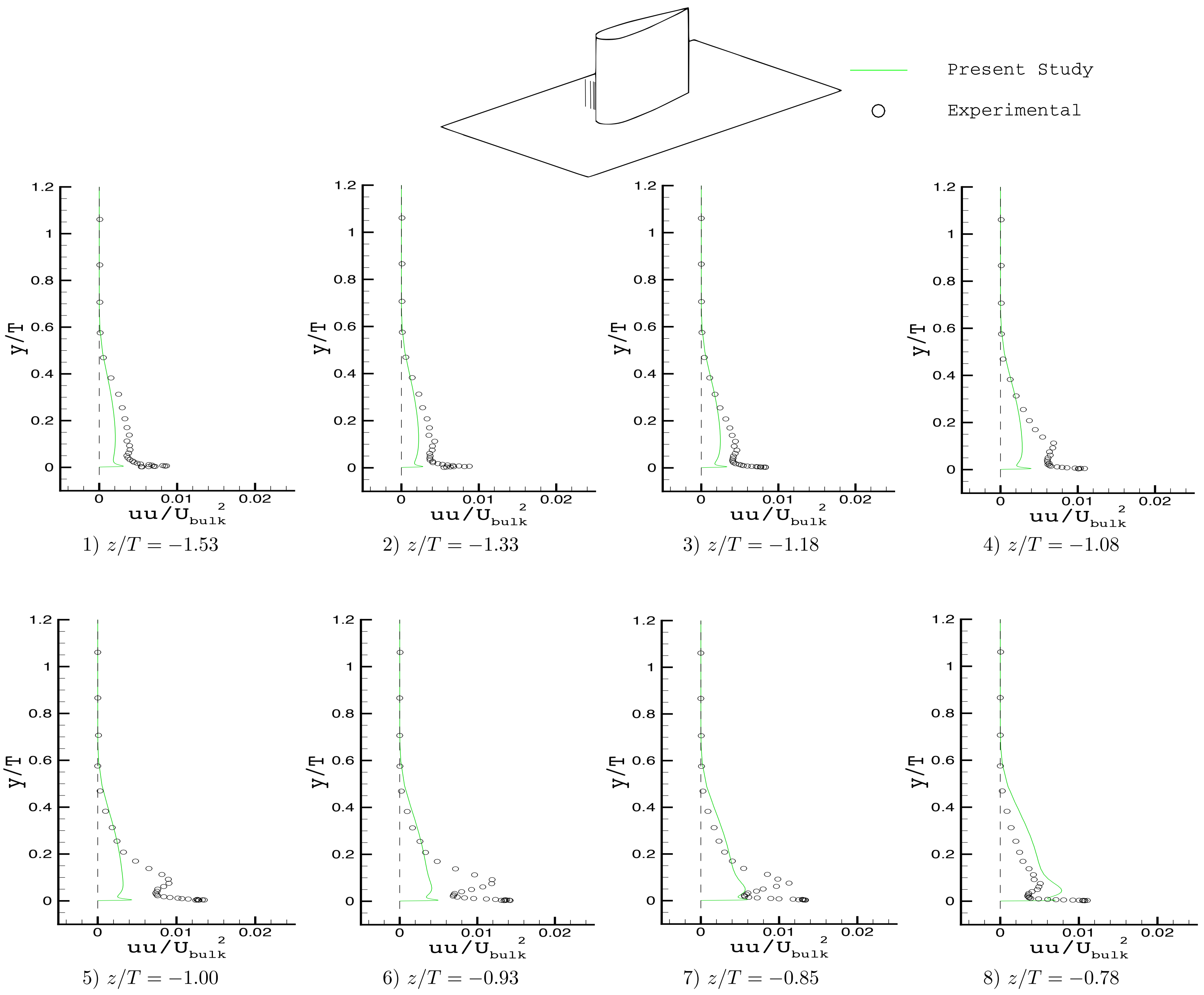


Figure 10.21: Streamwise normal Reynolds stress component, $\overline{u'^2}$, at maximum wing thickness ($x/T = 0.75$)

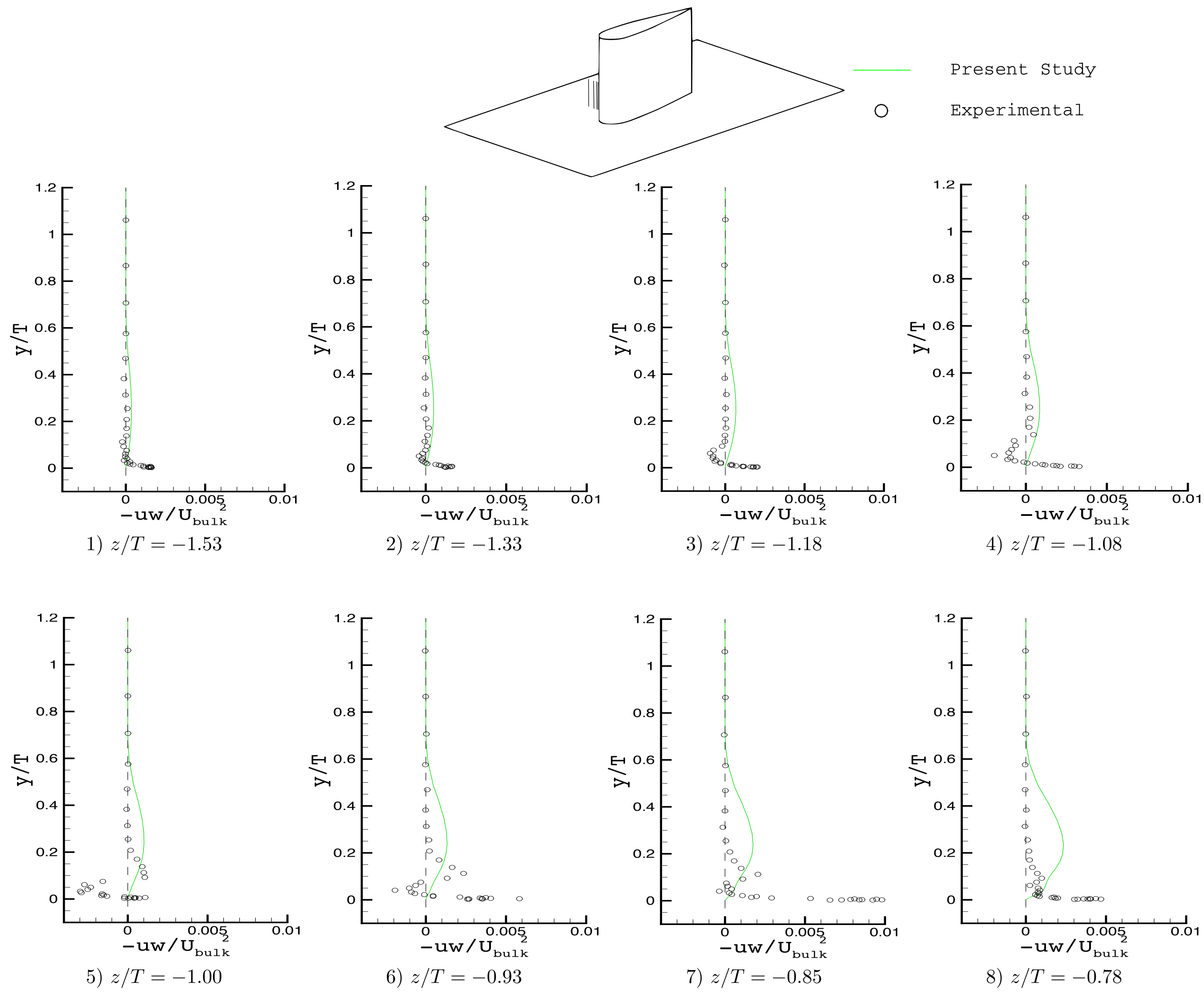


Figure 10.22: Normalised cross-stream Reynolds shear stress component, $\overline{u'w'}$, at maximum wing thickness ($x/T = 0.75$)

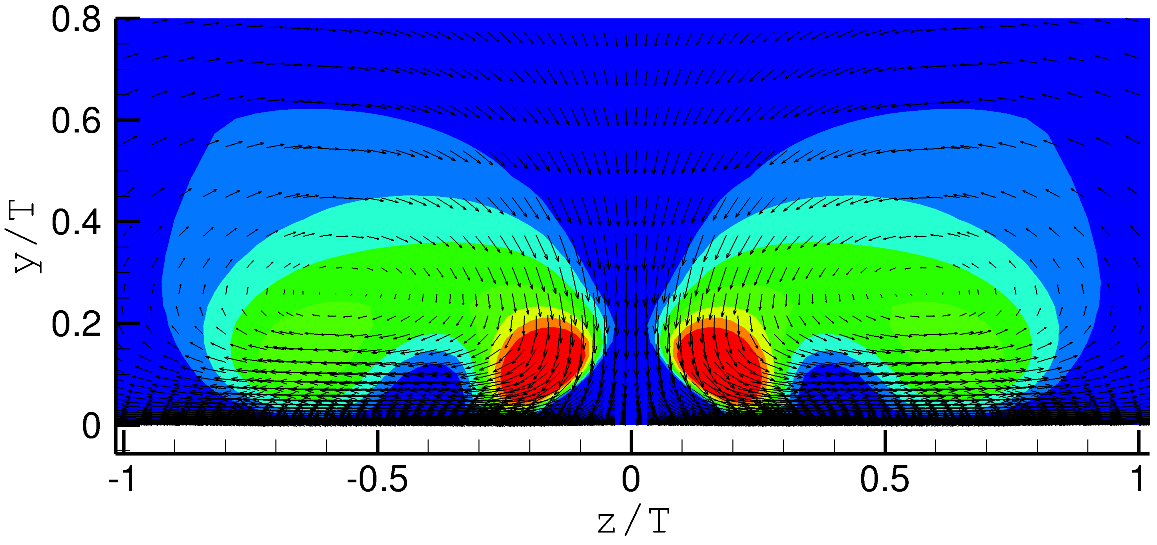


Figure 10.23: Contours of the Q-criterion in the plane $x/T = 6.38$ and secondary velocity vectors.

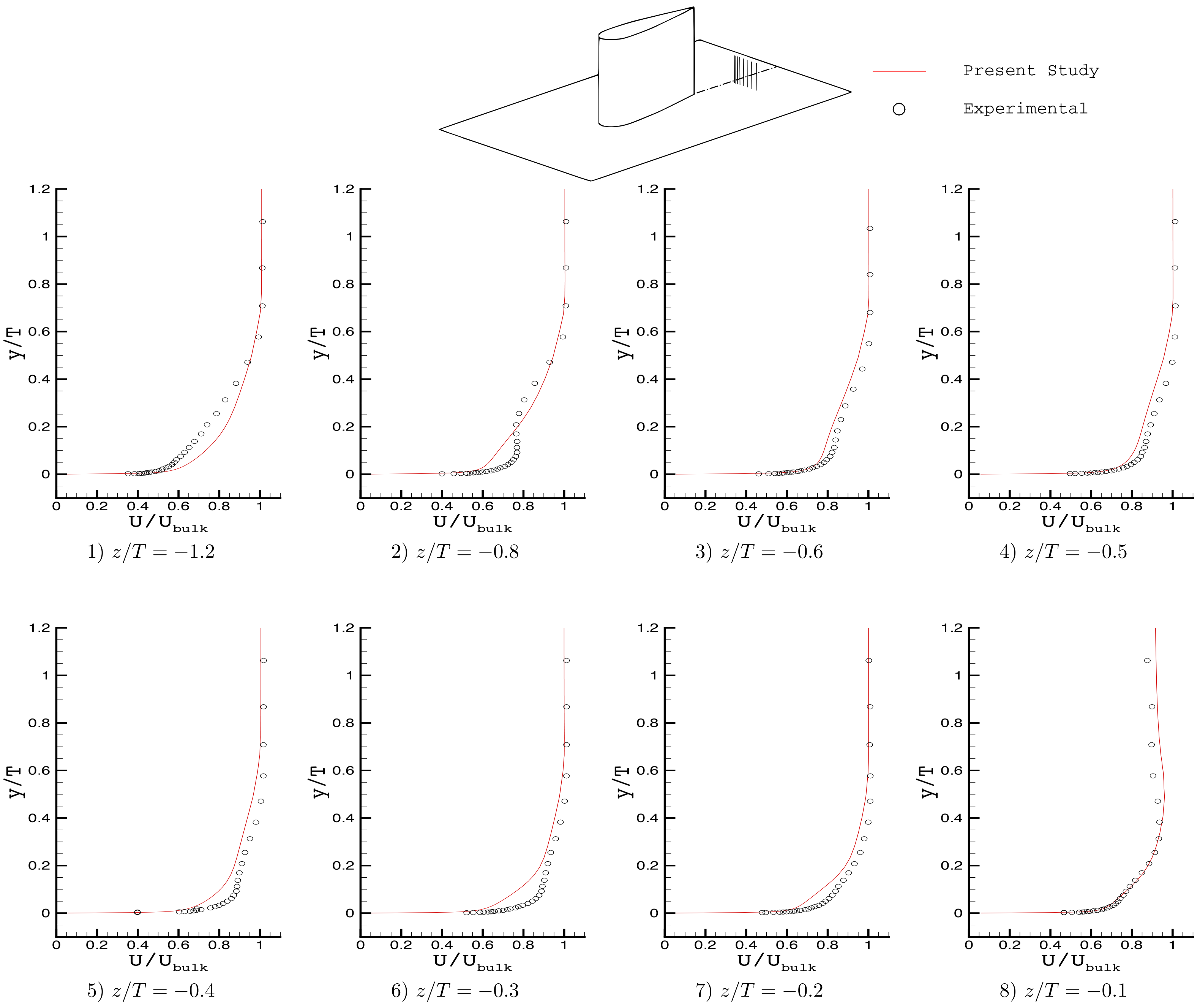


Figure 10.24: Normalised streamwise velocity component, U, in a plane in the wake($x/T = 6.38$)

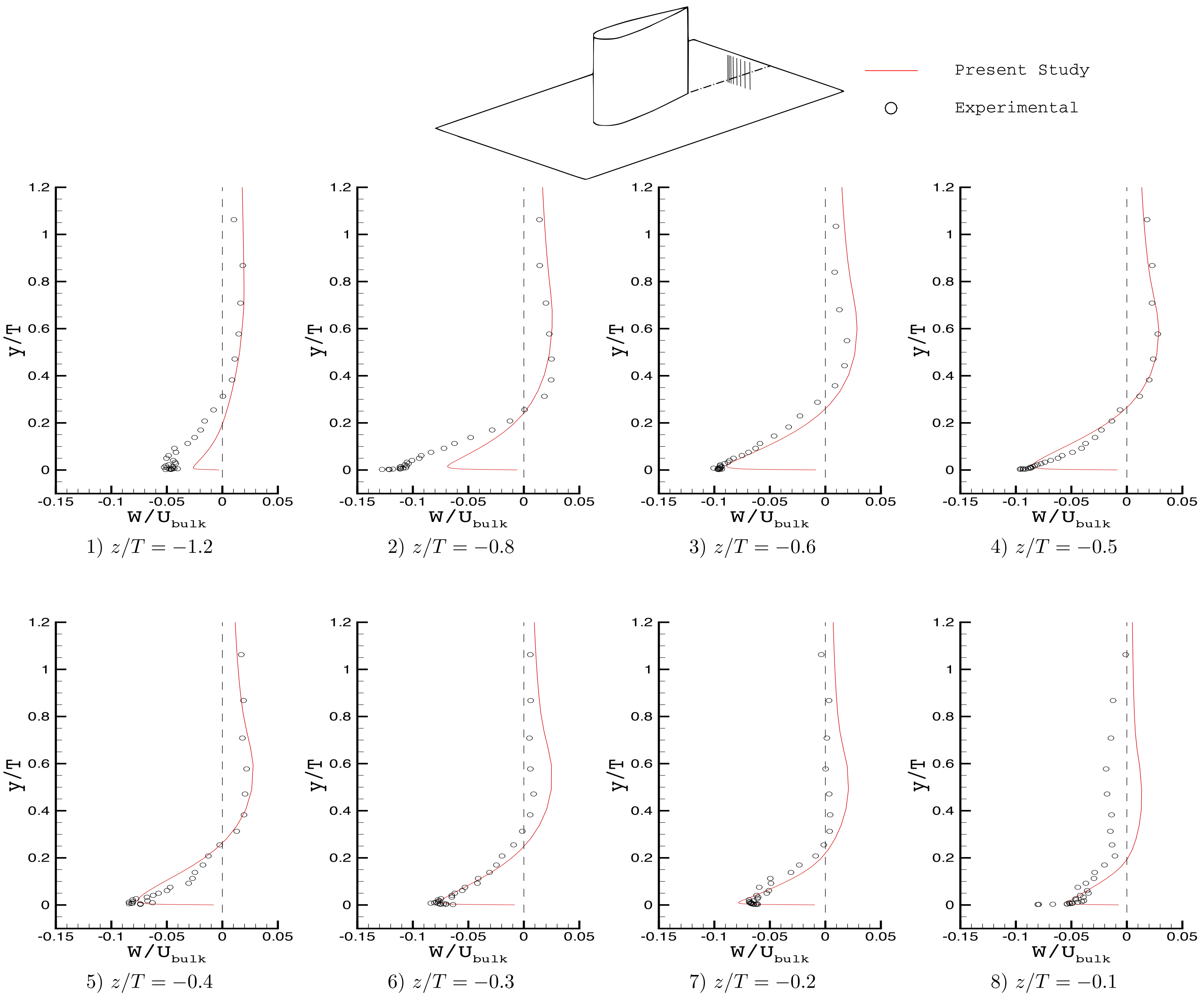


Figure 10.25: Normalised cross-stream velocity component, W, in a plane in the wake($x/T = 6.38$)

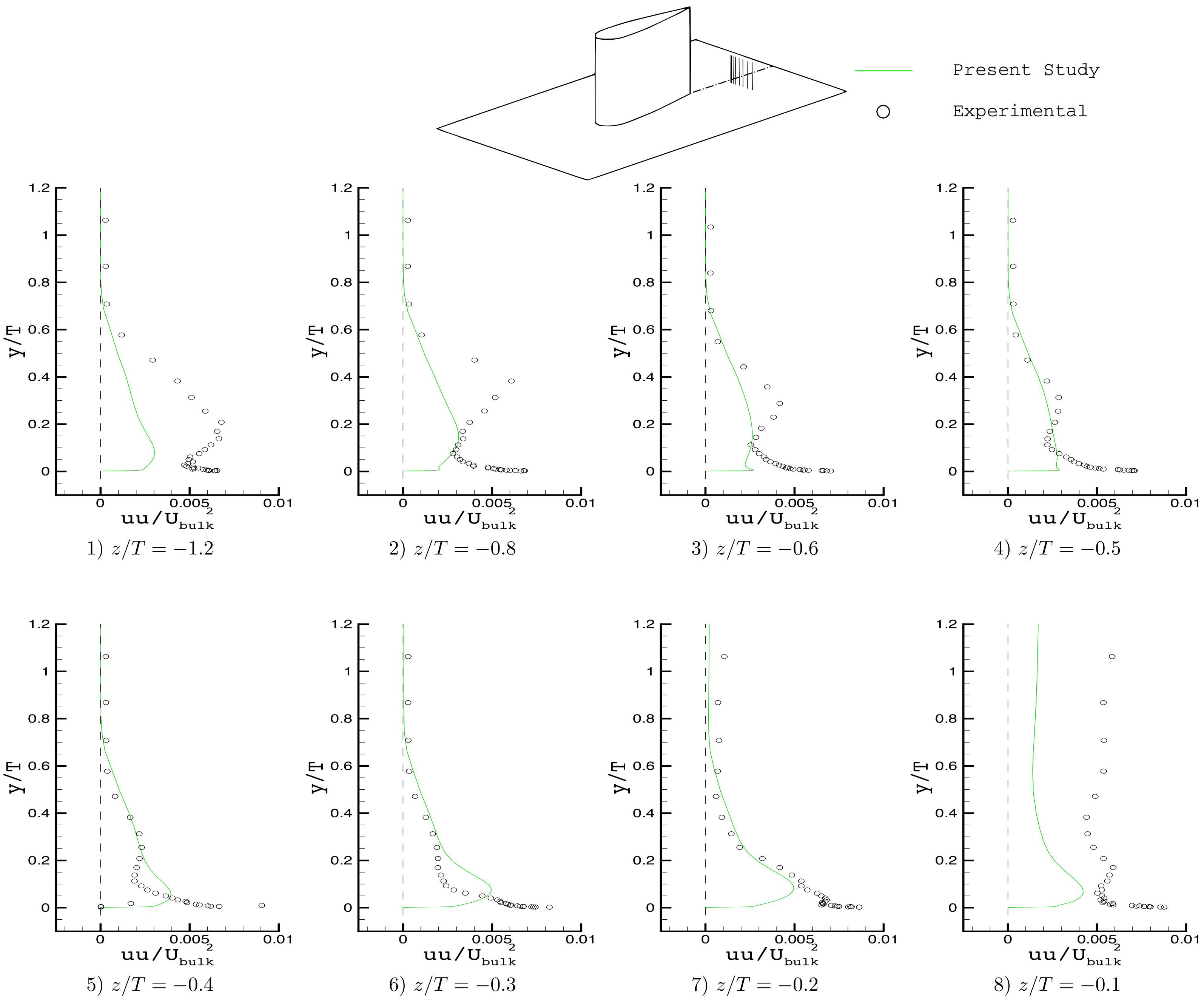


Figure 10.26: Streamwise normal Reynolds stress component, $\overline{u'^2}$, in the wake($x/T = 6.38$)

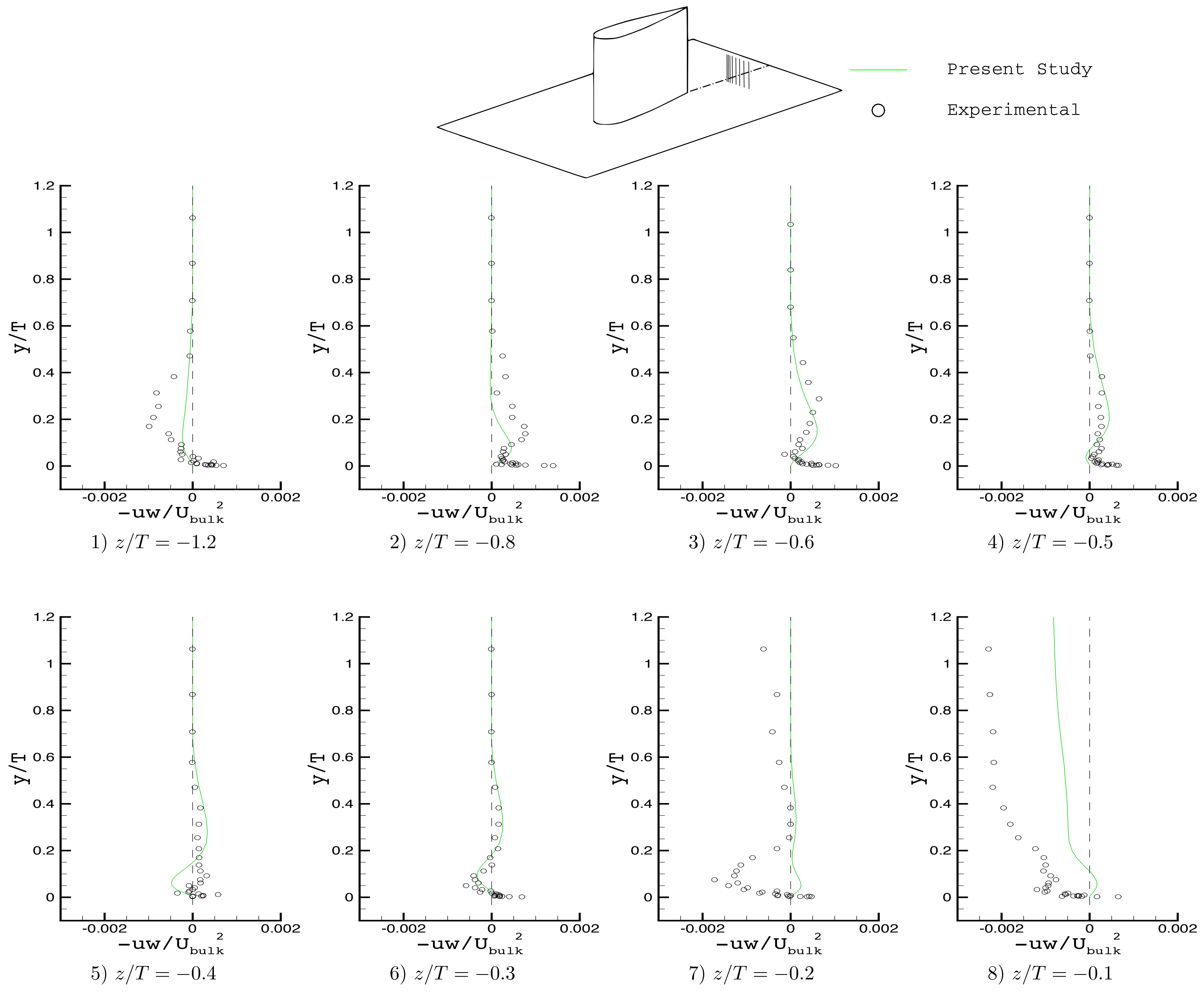


Figure 10.27: Normalised cross-stream Reynolds shear stress component, $\overline{u'w'}$, in the wake($x/T = 6.38$)

Chapter 11

Conclusion

The chapters contained within this Thesis have described the various algorithms that have been employed in the formulation of a new finite volume CFD code, with overset capabilities. A new hole cutting algorithm has been developed that is quick and robust. The hole cutter relies heavily on being able to find interpolation stencils expeditiously. Digital tree data structures are used in conjunction with the rapidly converging Newton Raphson method in order to provide this capability. A new digital tree structure is developed in this project that is based on the original method of Bonnet and Peraire, [56]. The new formulation gives roughly the same speedup as the original method, but is substantially easier to code.

Both standard linear interpolation and a semi-conservative mass flux based algorithm (MFBI) [1] have been implemented in the present code. It has been shown that both algorithms perform satisfactorily at grid independence. However, improvements in the rate of convergence have been noticed when using the MFBI algorithm (relative to non-conservative linear interpolation), and hence the use of the MFBI algorithm is recommended.

The new CFD code has been validated against various block-structured or single block grids from the commercial package FLUENT for a variety of test cases. It has been found that essentially the same solution is obtained for both the codes for the cases considered. Any very minor differences that were observed have been attributed to slight differences in the numerics between the two codes.

Tests involving the overset method have been carried out. It is found that continuous smooth solutions over an overset interface can be obtained via the use of linear interpolation for all variables, even if the grids are fairly coarse. It is, however, noted that the overlap should be minimised in order to prevent unique solutions from developing in the overlap region due to differences in truncation error between the grids.

Validations have also been carried out against experimental data for a variety of different flows. For the backward facing step flow considered in Chapter 7 it is seen that the predicted reattachment length and profiles of mean streamwise velocity are in close agreement with the experimental findings. This case saw one major advantage of an overset formulation relative to block-structured grids; namely that high aspect ratio cells away from the walls can be avoided with an overset formulation, a feat that is not easy to achieve with a block structured grid.

This theme continues into Chapter 8 where the flow over a multi-element airfoil has been considered. In previously reported block structured grids of the same geometry, it is seen that high aspect ratio cells propagate in into the core flow the cross-stream direction. These additional cells, coupled with the numerical instabilities that such high aspect ratio cells bring, mean that a block structured arrangement is not particularly well suited to the considered geometry. This, in addition to the not insubstantial domain decomposition that is required in order to generate block structured grids, makes the overset method particularly attractive for the simulation of flows over multi-element airfoils. The benefits of the overset method are further compounded if one uses efficient Cartesian grids to fill the off-body portion of the domain (provided the flow solver is coded to take advantage of such a situation).

The flow through a pipe bend and the flow through a branching artery have also both been considered. The grid zipping algorithm of Reference [53] has been applied in a novel way to the implementation of a bulk pressure correction over overset grids. The method was originally conceived for the computation of force coefficients over overset grids (i.e. as a postprocessing tool), where the cells in the overlap region require special treatment in order not to count the contribution from the overlap twice. Here, the original method is used to enforce global mass conservation in internal flows (a second order approximation to global mass conservation is applied). The use of the zipping algorithm enables the computation of the flow through the branching artery without the triangular prism cells at the centre of the pipe that would occur if using a single block to discretise each branch (as is typically the method employed by other authors). The correct flow division ratio can be inferred from the good agreement between computational and experimental data.

The branching artery flow also saw the use of a collar grid being applied. This additional grid was deemed of paramount importance to the present author, since without it the intersection area between the two artery branches will appear different depending on which branch the underlying cells belong to. This is due to the poor discrete approximation to the physical geometry that results from the cells not conforming to the interface. Such collar grids have not been used in previous studies of a bifurcation geometry (to this author's knowledge). As such, previous results will by necessity have shown unique solutions on the different grids in the overlap region (although the extent of this is unreported, such a situation is unavoidable if the effective intersection area is different for each branch).

The final test case considered is that of the flow over a wing-body junction. In this case the benefits of the use of additional overset grids in order to apply a mesh refinement over the wake are highlighted. The additional domain complexity that a wing-fuselage junction or a full aircraft configuration would bring is also alluded to, and in such situations it is noted that the overset method would provide a powerful option in domain decomposition. The flow over a simpler geometry is considered due to the availability of experimental data, however, the extension of the overset method to the more complex case is straightforward (if a little tedious). Even for the simple geometry, the method provides advantages in the form of efficient Cartesian solutions over the majority of the domain, in addition to avoiding the significant cell skewing and high aspect ratios that are typically found for a block-structured arrangement of the same geometry. The agreement between computation and experiment for the wing body junction flow is broadly satisfactory (with any discrepancy typically being due to inadequacies of the turbulence closure employed, and is broadly in line with other reported findings when using block-structured grids).

11.1 Future Work

There is much scope for future work. For example, the implementation of more advanced turbulence models within the code should improve predictions for the turbulent test cases considered. A wall function capability could also prove to be a very attractive addition to the present code by allowing the use of a smaller size computational mesh for the turbulent flow simulations.

The present code has reached an upper limit in terms of the number of cells that can be practically analysed on a single CPU core. In order to address even more geometrically complex configurations than those presented so far (such as a full aircraft for example), a parallel version of the code would be all but necessary. This extension would render a whole host of additional test cases accessible.

Another useful extension would be to develop a time dependent solver. The extension of the overset CFD code developed here to time-varying geometries is straightforward in concept (no changes need be made to the overset algorithms of Chapter 5, only the underlying flow solver needs to be modified). The additional capabilities that such an improvement would bring are numerous. Flows involving moving boundaries could be investigated with ease without the need to regenerate meshes between time-steps. Only the hole-cutter needs to be reinvoked - a fast procedure. Simulations involving grids that move relative to one another would then be a simple addition, thereby opening the way for many complex moving systems such as rotor stator systems or piston engines.

By undertaking these three primary extensions, a general parallel overset capability will have been developed which can be used to analyse a very wide variety of different flows. More case-specific future work could include the implementation of a non-Newtonian model for the fluid's viscosity for the artery flow. It would be interesting to test the extent to which the results differ under the assumption of Newtonian fluid behaviour.

The addition of heat transfer capabilities would be a very useful addition as there are many engineering problems involving heat transfer (or driven by heat transfer) which also involve very complex geometries. One such example is the buoyancy driven flow of oil through transformer windings currently under investigation within the author's group [99].

Finally, a code involving both the immersed boundary method described in Chapter 2, as well as the overset algorithm of Chapter 5, could provide even greater geometric flexibility. The immersed boundary method is well suited to elastic walls (although rigid walls can also be approximated), and hence could prove particularly useful for advancing the biomedical simulations presented in this thesis.

Thus concludes an investigation into the use of the overset grid method for complex geometrical configurations.

Bibliography

[1] H.S. Tang, S. Casey Jones, and F. Sotiropoulos. An overset-grid method for 3D unsteady incompressible flows. *J. of Computat. Phys.*, 191(2):567–600, 2003.

[2] BE Launder and BI Sharma. Application of the energy-dissipation model of turbulence to the calculation of flow near a spinning disc. *Letters in Heat and Mass Transfer*, 1(2):131–138, 1974.

[3] L.H. Thomas. Elliptic problems in linear difference equations over a network, Watson Sci. *Comput. Lab. Rept., Columbia University, New York*, 1949.

[4] J.A. Benek, J.L. Steger, and F.C. Dougherty. A flexible grid embedding technique with application to the Euler equations. *AIAA Journal*, pages 373–382, Jul 1983.

[5] JA Benek, JL Steger, FC Dougherty, and PG Buning. Chimera. A Grid-Embedding Technique. 1986.

[6] W.D. Henshaw. A fourth-order accurate method for the incompressible Navier-Stokes equations on overlapping grids. *Journal of Computational Physics*, 113(1):13–25, 1994.

[7] P.D. Lax. *Hyperbolic Systems of Conservation Laws and the Mathematical Theory of Shock Waves.* Society for Industrial Mathematics, 1973.

[8] C.J. Freitas and S.R. Runnels. Simulation of fluid–structure interaction using patched-overset grids. *Journal of Fluids and Structures*, 13(2):191–207, 1999.

[9] P. Fast and M. J. Shelley. A moving overset grid method for interface dynamics applied to non-Newtonian Hele-Shaw flow. *J. Comput. Phys.*, 195:117–142, 2004.

[10] E. Basso and J.L.F. Azevedo. Three-dimensional viscous flow simulations over the VLS using overset grids. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 26:438–445, 2004.

[11] B. Chung, P.C. Johnson, and A.S. Popel. Application of Chimera grid to modelling cell motion and aggregation in a narrow tube. *Int. J. Numer. Methods Fluids*, 53:105–28, 2007.

[12] Z.J. Wang. A Conservative Interface Algorithm for Moving Chimera (Overlapped) Grids. *International Journal of Computational Fluid Dynamics*, 10(3):255–265, 1998.

[13] L. Ge, F. Sotiropoulos, et al. 3d unsteady rans modeling of complex hydraulic engineering flows. i: Numerical model. *Journal of Hydraulic Engineering*, 131:800, 2005.

[14] L. Ge, S.O. Lee, F. Sotiropoulos, T. Sturm, et al. 3d unsteady rans modeling of complex hydraulic engineering flows. ii: Model validation and flow physics. *Journal of Hydraulic Engineering*, 131:809, 2005.

[15] M.S. Liou and K.H. Kao. Progress in Grid Generation: From Chimera to DRAGON Grids. *NASA TM*, 209458, 1994.

[16] Y. Zheng, M.S. Liou, and K.C. Civinskas. Development of three-dimensional DRAGON grid technology. *NASA TM*, 209458, 1999.

[17] Z.J. Wang. A fully conservative interface algorithm for overlapped grids. *J. Comput. Phys.*, 122:96–106, 1995.

[18] P.L. Roe. Approximate Riemann solvers, parameter vectors, and difference schemes. *J. Comput. Phys.*, 135:250–258, 1997.

[19] M.J. Berger. On conservation at grid interfaces. *SIAM J. Numer. Anal*, 24(5):967–984, 1987.

[20] G. Chesshire and WD Henshaw. A Scheme for Conservative Interpolation on Overlapping Grids. *SIAM Journal on Scientific Computing*, 15:819, 1994.

[21] X.K.S. Gang. Assessment of an interface conservative algorithm mfbi in a chimera grid flow solver for multi-element airfoils. In *Proceedings of the World Congress on Engineering*, volume 2. Citeseer, 2009.

[22] C.S. Peskin. Flow patterns around heart valves. In *Proceedings of the Third International Conference on Numerical Methods in Fluid Mechanics, Lecture Notes in Physics*, volume 19, 1973.

[23] C.S. Peskin. Numerical analysis of blood flow in the heart. *J. Comput. Phys*, 25(3):220–252, 1977.

[24] R. Mittal and G. Iaccarino. Immersed boundary methods. 2005.

[25] J.M. Stockie and B.R. Wetton. Analysis of stiffness in the immersed boundary method and implications for time-stepping schemes. *Journal of Computational Physics*, 154(1):41–64, 1999.

[26] G. Kalitzin and G. Iaccarino. Toward immersed boundary simulation of high Reynolds number flows. *CTR Annual Research Briefs*, pages 369–378, 2003.

[27] G. Yun, H. Choi, and D. Kim. Turbulent flow past a sphere at Re= 3700 and 10,000. *Physics of Fluids*, 15, 2003.

[28] HJ Kim and PA Durbin. Observations of the frequencies in a sphere wake and of drag increase by acoustic excitation. *Physics of fluids*, 31:3260, 1988.

[29] H. Sakamoto and H. Haniu. A study on vortex shedding from spheres in a uniform flow. *ASME, Transactions, Journal of Fluids Engineering*, 112:386–392, 1990.

[30] G. Constantinescu and K. Squires. Numerical investigations of flow over a sphere in the subcritical and supercritical regimes. *Physics of Fluids*, 16:1449, 2004.

[31] R. Mittal, V. Seshadri, SE Sarma, and HS Udaykumar. Computational modeling of fluidic micro-handling processes. In *5th Int. Conf. Modeling and Simulation of Microsystems, San Juan, PR*, 2002.

[32] L.J. Fauci and A. McDonald. Sperm motility in the presence of boundaries. *Bulletin of mathematical biology*, 57(5):679–699, 1995.

[33] H. Schlichting, K. Gersten, E. Krause, K. Mayes, and H. Oertel. *Boundary-layer theory*. Springer Verlag, 2000.

[34] S. Chen and G.D. Doolen. Lattice Boltzmann method for fluid flows. *Annual Review of Fluid Mechanics*, 30(1):329, 2003.

[35] R.A. Fine and F.J. Millero. Compressibility of water as a function of temperature and pressure. *The Journal of Chemical Physics*, 59:5529, 1973.

[36] J. H. Ferziger and M. Perić. *Computational methods for fluid dynamics.* Springer, Berlin, 1999.

[37] BP Leonard. A stable and accurate convective modelling procedure based on quadratic upstream interpolation. *Computer methods in applied mechanics and engineering*, 19(1):59–98, 1979.

[38] F.S. Lien and M.A. Leschziner. Upstream monotonic interpolation for scalar transport with application to complex turbulent flows. *Int. J. Numer. Methods Fluids*, 19:527–548, 1994.

[39] H.K. Versteeg and W. Malalasekera. *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*, chapter 7. Prentice Hall, 2007.

[40] S.V. Patankar. Numerical heat transfer and fluid flow. *MAC Graw Hill*, 1980.

[41] S.V. Patankar and D.B. Spalding. A Calculation Procedure for Heat, Mass and Momentum transfer in three Dimensional Parabolic Flows. *Numerical Prediction of Flow, Heat Transfer, Turbulence, and Combustion: Selected Works of Professor D. Brian Spalding*, 1983.

[42] C.M. Rhie and W.L. Chow. Numerical Study of the Turbulent Flow Past an Airfoil with Trailing Edge Separation. *AIAA Journal*, 21(11):1525–1532, 1983.

[43] L. Prandtl. *Essentials of fluid dynamics.* Blackie, 1952.

[44] D.C. Wilcox. Turbulence modeling for CFD. 1998.

[45] K.Y. Chien. Predictions of channel and boundary-layer flows with a low-Reynolds-number turbulence model. *AIAA journal*, 20(1):33–38, 1982.

[46] MA Leschziner. Modelling turbulent separated flow in the context of aerodynamic applications. *Fluid dynamics research*, 38(2-3):174–210, 2006.

[47] K. Abe, T. Kondoh, and Y. Nagano. A new turbulence model for predicting fluid flow and heat transfer in separating and reattaching flows–ii. thermal field calculations. *International journal of heat and mass transfer*, 38(8):1467–1481, 1995.

[48] BE Launder. On the computation of convective heat transfer in complex turbulent flows. *Journal of heat transfer*, 110:1112, 1988.

[49] S. Poroseva and G. Iaccarino. Simulating separated flows using the k-ϵ model. *Center for Turbulence Research Annual Research Briefs*, 383, 2001.

[50] C. Yap. *Turbulent heat and momentum transfer in recirculating and impinging flows.* PhD thesis, University of Manchester, 1987.

[51] H. Iacovides and M. Raisee. Recent progress in the computation of flow and heat transfer in internal cooling passages of turbine blades. *International Journal of Heat and Fluid Flow*, 20(3):320–328, 1999.

[52] AN Kolmogorov. A refinement of previous hypotheses concerning the local structure of turbulence in a viscous incompressible fluid at high Reynolds number. *Journal of Fluid Mechanics Digital Archive*, 13(01):82–85, 2006.

[53] W.M. Chan and P.G. Buning. Zipper grids for force and moment computation on overset grids. *AIAA journal*, 1995.

[54] G. Chesshire and W.D. Henshaw. Composite overlapping meshes for the solution of partial differential equations. *Journal of Computational Physics*, 90(1):1–64, 1990.

[55] J. Gerlach. Accelerated convergence in Newton's method. *SIAM Rev*, 36:272–276, 1994.

[56] J. Bonet and J. Peraire. An alternating digital tree (ADT) algorithm for 3D geometric searching and intersection problems. *International Journal for Numerical Methods in Engineering*, 31(1):1–17, 1991.

[57] KE Barret and RJ Tiddy. Pseudo-recursive FORTRAN. *Advances in engineering software*, 21(1):17–20, 1994.

[58] Fuat Gürcan. Streamline topologies in stokes flow within lid-driven cavities. *Theoretical and Computational Fluid Dynamics*, 17:19–30, 2003. 10.1007/s00162-003-0095-z.

[59] P.M. Gresho and R.L. Sani. On pressure boundary conditions for the incompressible Navier-Stokes equations. *International Journal for Numerical Methods in Fluids (ISSN 0271-2091)*, 7, 1987.

[60] KB Chun and HJ Sung. Control of turbulent separated flow over a backward-facing step by local forcing. *Experiments in Fluids*, 21(6):417–426, 1996.

[61] P. Momeni. *Modelling the effect of pulsation on flow and heat transfer in turbulent seperated and reattaching flows.* PhD thesis, University of Manchester, 2008.

[62] P. Momeni T.J. Craft, H. Iacovides. Convective Heat Transfer in Pulsed Separated Flows. In *UK National Heat Transfer Conference*, 2007.

[63] D. Adair and WC Horne. Turbulent separated flow over and downstream of a two-element airfoil. *Experiments in Fluids*, 7(8):531–541, 1989.

[64] G. Iaccarino and PA Durbin. Application of the k-ϵ-v2 model to multi-component airfoils. In *Proceedings of the Summer Program*, page 23, 1996.

[65] C.L. Rumsey and T.B. Gatski. Recent turbulence model advances applied to multielement airfoil computations. *Journal of aircraft*, 38(5):904–910, 2001.

[66] W. Rodi and G. Scheuerer. Scrutinizing the k-epsilon-model under adverse pressure gradient conditions. In *4th Symposium on Turbulent Shear Flows*, volume 1, page 2, 1984.

[67] N. Shahcheraghi, HA Dwyer, AY Cheer, AI Barakat, and T. Rutaganira. Unsteady and three-dimensional simulation of blood flow in the human aortic arch. *Journal of biomechanical engineering*, 124:378, 2002.

[68] AY Cheer, HA Dwyer, AI Barakat, E. Sy, and M. Bice. Computational study of the effect of geometric and flow parameters on the steady flow field at the rabbit aorto-celiac bifurcation. *BIORHEOLOGY-OXFORD-*, 35:415–436, 1998.

[69] M. Rowe. Measurements and computations of flow in pipe bends. *Journal of Fluid Mechanics*, 43(04):771–783, 1970.

[70] MM Enayet, MM Gibson, A. Taylor, and M. Yianneskis. Laser-doppler measurements of laminar and turbulent flow in a pipe bend. *International Journal of Heat and Fluid Flow*, 3(4):213–219, 1982.

[71] S.W. Jones, O.M. Thomas, and H. Aref. Chaotic advection by laminar flow in a twisted pipe. *Journal of Fluid Mechanics*, 209(1):335–357, 1989.

[72] F. Rütten, W. Schröder, and M. Meinke. Large-eddy simulation of low frequency oscillations of the dean vortices in turbulent pipe bend flows. *Physics of Fluids*, 17:035107, 2005.

[73] S.V. Patankar, VS Pratap, and DB Spalding. Prediction of turbulent flow in curved pipes. *Journal of Fluid Mechanics*, 67(03):583–595, 1975.

[74] J. Azzola, JAC Humphrey, H. Iacovides, and BE Launder. Developing turbulent flow in a u-bend of circular cross-section: measurement and computation. *Journal of fluids engineering*, 108:214, 1986.

[75] RH Kufahl and ME Clark. A circle of willis simulation using distensible vessels and pulsatile flow. *Journal of biomechanical engineering*, 107:112, 1985.

[76] M. Schüler, F. Zehnder, B. Weigand, J. von Wolfersdorf, and S.O. Neumann. The effect of turning vanes on pressure loss and heat transfer of a ribbed rectangular two-pass internal cooling channel. *Journal of Turbomachinery*, 133:021017, 2011.

[77] PHM Bovendeerd, F.N. Vosse, A. Steenhoven, and G. Vossers. Steady entry flow in a curved pipe. *Journal of fluid mechanics*, 177:233–46, 1987.

[78] D.L. Fry. Acute vascular endothelial changes associated with increased blood velocity gradients. *Circulation Research*, 22(2):165, 1968.

[79] CG Caro, JM Fitz-Gerald, and RC Schroter. Atheroma and arterial wall shear observation, correlation and proposal of a shear dependent mass transfer mechanism for atherogenesis. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 177(1046):109, 1971.

[80] BK Bharadvaj, RF Mabon, and DP Giddens. Steady flow in a model of the human carotid bifurcation. Part I–flow visualization. *Journal of Biomechanics*, 15(5):349–362, 1982.

[81] K. Masuda, N. Watarai, R. Nakamoto, Y. Miyamoto, K. Kim, and T. Chiba. Study to prevent the density of microcapsules from diffusing in blood vessel by local acoustic radiation force. In *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pages 402–405. IEEE, 2010.

[82] H.S. Kim, H.S. Chong, A. Nanda, J.O. Park, S.H. Moon, H.M. Lee, H.J. Kim, C.K. Park, Y.S. Park, S.H. Lee, et al. Vascular Injury in Thoracolumbar Spinal Surgeries and Role of Angiography in Early Diagnosis and Management. *Journal of Spinal Disorders & Techniques*, 23(6):418, 2010.

[83] E.W. Merrill and G.A. Pelletier. Viscosity of human blood: transition from Newtonian to non-Newtonian. *Journal of Applied Physiology*, 23(2):178, 1967.

[84] BK Bharadvaj, RF Mabon, and DP Giddens. Steady flow in a model of the human carotid bifurcation. Part II–Laser-Doppler anemometer measurements. *Journal of Biomechanics*, 15(5):363–365, 1982.

[85] T. Kim, AY Cheer, and HA Dwyer. A simulated dye method for flow visualization with a computational model for blood flow. *Journal of biomechanics*, 37(8):1125–1136, 2004.

[86] PG Buning, IT Chiu, S. Obayashi, YM Rizk, and JL Steger. Numerical simulation of the integrated space shuttle vehicle in ascent. In *AIAA Atmospheric Flight Mechanics Conference, Minneapolis, MN*, pages 265–283, 1988.

[87] J.J. Chattot and Y. Wang. Improved treatment of intersecting bodies with the chimera method and validation with a simple and fast flow solver. *Computers & fluids*, 27(5-6):721–740, 1998.

[88] A. Jameson and TJ Baker. Improvements to the aircraft euler method. In *AIAA, Aerospace Sciences Meeting, 25 th, Reno, NV*, page 1987, 1987.

[89] W. DEVENPORT and R. SIMPSON. Time-dependent and time-averaged turbulence structure near the nose of a wing-body junction. *Journal of Fluid Mechanics*, 210(1):23–55, 1990.

[90] J.L. Fleming, RL Simpson, JE Cowling, and WJ Devenport. An experimental study of a turbulent wing-body junction and wake flow. *Experiments in fluids*, 14(5):366–378, 1993.

[91] RL Simpson, MC Rife, and WJ Devenport. An experimental study of the relationship between velocity and pressure fluctuations in a wing-body junction. *Virginia polytechnic inst. and state univ. Blacksburg dept. of Aerospace and Ocean engineering*, 1992.

[92] W.J. Devenport. An experimental investigation of the flow past an idealized wing-body junction. Technical report, DTIC Document, 1990.

[93] DD Apsley and MA Leschziner. Investigation of advanced turbulence models for the flow in a generic wing-body junction. *Flow, turbulence and combustion*, 67(1):25–55, 2001.

[94] Y. Dubief and F. Delcayre. On coherent-vortex identification in turbulence. *Journal of Turbulence*, 1(1):N11, 2000.

[95] PA Durbin. On the k-ϵ stagnation point anomaly. *International journal of heat and fluid flow*, 17(1):89–90, 1996.

[96] G. Medic and PA Durbin. Toward improved prediction of heat transfer on turbine blades. *Journal of turbomachinery*, 124:187, 2002.

[97] BE Launder, GJ Reece, and W. Rodi. Progress in the development of a reynolds-stress turbulence closure. *Journal of Fluid Mechanics*, 68(03):537–566, 1975.

[98] MM Gibson and BE Launder. Ground effects on pressure fluctuations in the atmospheric boundary layer. *Journal of Fluid Mechanics*, 86(03):491–511, 1978.

[99] A. Skillen, A. Revell, H. Iacovides, and W. Wu. Numerical prediction of local hot-spot phenomena in transformer windings. *Applied Thermal Engineering*, 2011.