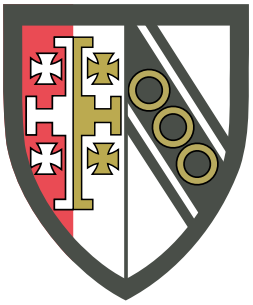
A dimensionally split Cartesian cut cell method for Computational Fluid Dynamics

Nandan Bhushan Gokhale



Selwyn College

This dissertation is submitted for the degree of Doctor of Philosophy

June, 2018

ABSTRACT

We present a novel dimensionally split Cartesian cut cell method to compute inviscid, viscous and turbulent flows around rigid geometries.

On a cut cell mesh, the existence of arbitrarily small boundary cells severely restricts the stable time step for an explicit numerical scheme. We solve this 'small cell problem' when computing solutions for hyperbolic conservation laws by combining wave speed and geometric information to develop a novel stabilised cut cell flux. The convergence and stability of the developed technique are proved for the one-dimensional linear advection equation, while its multi-dimensional numerical performance is investigated through the computation of solutions to a number of test problems for the linear advection and Euler equations. This work was recently published in the Journal of Computational Physics [1].

Subsequently, we develop the method further to be able to compute solutions for the compressible Navier-Stokes equations. The method is globally second order accurate in the L_1 norm, fully conservative, and allows the use of time steps determined by the regular grid spacing. We provide a full description of the three-dimensional implementation of the method and evaluate its numerical performance by computing solutions to a wide range of test problems ranging from the nearly incompressible to the highly compressible flow regimes. This work was recently published in the Journal of Computational Physics [2]. It is the first presentation of a dimensionally split cut cell method for the compressible Navier-Stokes equations in the literature.

Finally, we also present an extension of the cut cell method to solve high Reynolds number turbulent automotive flows using a wall-modelled Large Eddy Simulation (WMLES) approach. A full description is provided of the coupling between the (implicit) LES solution and an equilibrium wall function on the cut cell mesh. The combined methodology is used to compute results for the turbulent flow over a square cylinder, and for flow over the

SAE Notchback and DrivAer reference automotive geometries. We intend to publish the promising results as part of a future publication, which would be the first assessment of a WMLES Cartesian cut cell approach for computing automotive flows to be presented in the literature.

DECLARATION

This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration except as declared in the Preface and specified in the text. It is not substantially the same as any that I have submitted, or, is being concurrently submitted for a degree or diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the Preface and specified in the text. I further state that no substantial part of my dissertation has already been submitted, or, is being concurrently submitted for any such degree, diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the Preface and specified in the text. It does not exceed the word limit of 60,000 words as prescribed by the Degree Committee for the Department of Physics.

Nandan Bhushan Gokhale
June, 2018

ACKNOWLEDGEMENTS

I would like to start by thanking my supervisor, Dr. Nikos Nikiforakis, for his unwavering support, guidance and encouragement over the course of this PhD. Thanks are also due to Prof. Rupert Klein, whose encouragement and technical insight were invaluable when I was proving the numerical properties of the novel cut cell method.

I would also like to acknowledge my laboratory colleagues with whom I have had countless discussions and debates, all of which ultimately contributed towards increasing the quality of this work. Special thanks are due to Dr. Philip Blakely for his support with the parallel AMR code, and to Lukas Wutschitz, Dr. Oliver Stickson and Dr. Paul Bennett for all the fruitful conversations.

This section would be incomplete if I didn't express my deep gratitude towards my friends and family, whose support has been instrumental in allowing me to complete this work.

Last, but not the least, I would also like to thank the Cambridge Commonwealth, European & International Trust for financially supporting my research.

CONTENTS

CHAPTER 1

INTRODUCTION

Cartesian cut cell approaches offer an attractive alternative to conventional body-fitted or unstructured meshing techniques due to the ease of automatic mesh generation for complex geometries, and the computational conveniences offered by the use of Cartesian grids. The regular data structures used to represent Cartesian grids are simple to handle programmatically and allow for straightforward grid partitioning in parallel programming algorithms. Furthermore, the use of powerful Adaptive Mesh Refinement (AMR) techniques such as that of Berger and Oliger [3] is enabled by the regularity inherent to Cartesian grids.

With curvilinear body-fitted meshes, suitable mappings from the physical to the computational domain may be difficult to obtain for complex geometries, particularly in 3D. At the very least, the process is time-consuming and can result in highly skewed meshes with unnecessarily fine resolutions in hard to mesh areas such as sharp internal corners [4]. In an unstructured mesh, the fluid control volumes can be arbitrarily shaped, with triangular or tetrahedral elements being popular choices. Although the procedure allows great flexibility in the meshing process, it brings with it the inevitable loss in the regularity of computational data structures [5]. Furthermore, the generation of a good quality mesh around a complex geometry typically tends to be an iterative process involving significant user intervention [6].

The Cartesian cut cell mesh generation procedure involves computationally 'cutting out' the geometry from a background Cartesian grid to produce a resulting mesh with a sharp representation of the interface. The procedure allows for rapid, automatic mesh generation for complex geometries while still retaining the computational conveniences offered by the use of Cartesian grids. Note that although a number of conservative and

non-conservative Cartesian grid based methods to handle embedded boundaries exist (these are usually clubbed together under the umbrella of 'immersed boundary methods' [6]), we use the term 'cut cell method' to refer exclusively to finite volume based Cartesian approaches that are designed to be fully conservative. Fig. 1.1 shows a cut cell grid generated around the 'DrivAer' automotive model [7] in its fastback configuration. The complexity of the meshing procedure is unaffected by the complexity of the geometry - in smooth, convex areas (as in Fig. 1.1b), or sharp concave regions (as in Fig. 1.1c-Fig. 1.1e), mesh generation always involves subtracting the geometry from the background Cartesian grid.

Of course, since the cut cells that are created next to the interface by the meshing procedure can be arbitrarily small, there is a severe constraint imposed on the explicit time step by which the solution can be evolved. This is called the 'small cell problem' and the challenge involved in designing a cut cell method is to devise some means of evolving the solution in the cut cells using a time step governed by the size of the regular, uncut, Cartesian cells.

A number of different methodologies have been presented in the literature to deal with the small cell problem in the context of both the compressible Euler and Navier-Stokes equations. We provide a brief overview of these approaches in Chapter 3 and Chapter 4 respectively. The common aspect to all of them, however, is that they are implemented largely in an unsplit fashion. We are particularly interested in adopting a dimensionally split approach which is a convenient way to extend one-dimensional methods to solve multi-dimensional problems. In this work, we present the development of a novel dimensionally split Cartesian cut cell method to compute inviscid, laminar and turbulent flows around rigid embedded boundaries. We believe that our work will be of interest to researchers and practitioners who currently use, or are interested in using, dimensionally split approaches for multi-dimensional extensions.

Another point of note is that we focus on solving the compressible governing equations. In terms of potential applications, this means we deal with supersonic flows (e.g. shock diffraction over a complex geometry), and subsonic flows where compressibility is important (e.g. external flows over aerospace or automotive geometries).

1.1 Thesis structure

The rest of this thesis is organised as follows.

1. In **Chapter 2**, we describe the procedure that we use to generate a Cartesian cut cell mesh. Note that the approach used to overcome the small cell problem and

compute solutions on the mesh does not depend on the technique used to create the mesh.

2. In **Chapter 3**, we present a novel dimensionally split Cartesian cut cell method to compute solutions for hyperbolic conservation laws. This content was recently published in the *Journal of Computational Physics* [1]:

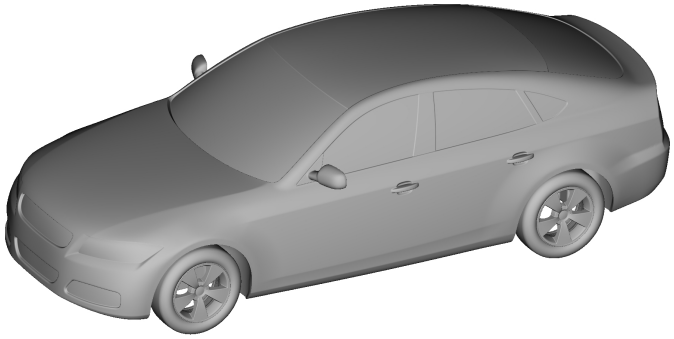
   N. Gokhale, N. Nikiforakis, and R. Klein. A dimensionally split Cartesian cut cell method for hyperbolic conservation laws. *Journal of Computational Physics*, 364: 186-208, 2018.

3. In **Chapter 4**, we present an extension of the cut cell method to compute solutions for compressible Navier-Stokes problems involving rigid embedded boundaries. This content was recently published in the *Journal of Computational Physics* [2] and is the first presentation of a dimensionally split method for the compressible Navier-Stokes equations in the literature:

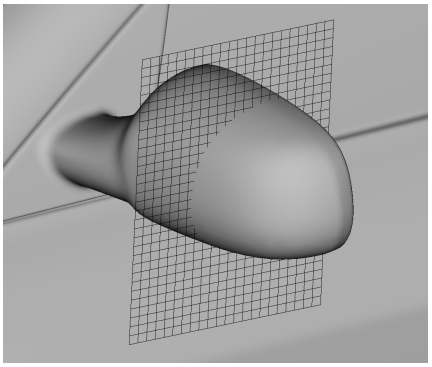
   N. Gokhale, N. Nikiforakis, and R. Klein. A dimensionally split Cartesian cut cell method for the compressible Navier-Stokes equations. *Journal of Computational Physics*, 375: 1205-1219, 2018.

4. In **Chapter 5**, we present a wall-modelled Large Eddy Simulation (WMLES) approach to computing turbulent flows using our dimensionally split cut cell method. This is achieved by using an Implicit Large Eddy Simulation [8] approach to calculate intercell fluxes, and an equilibrium wall function to compute the wall shear stresses. We use the approach to calculate flows over idealised and realistic complex automotive geometries. We expect to present these results, which would be the first assessment of a WMLES Cartesian cut cell approach for computing automotive flows, as part of a future publication.

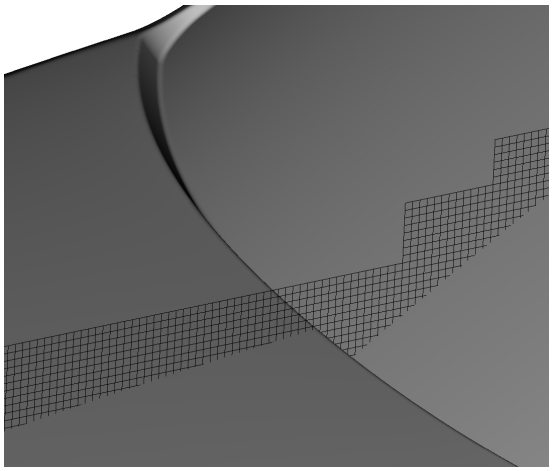
5. Finally, conclusions and areas for future work are presented in **Chapter 6**.



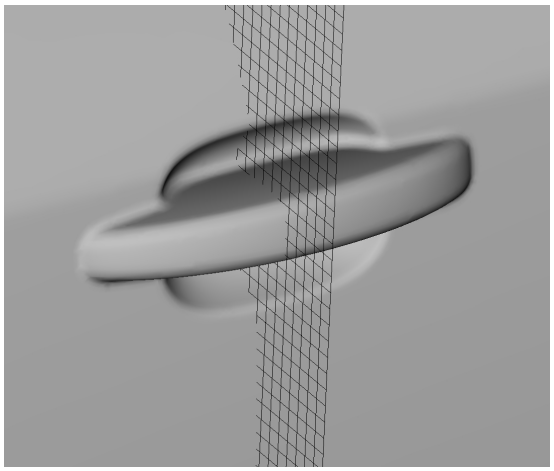
(a) DrivAer geometry.



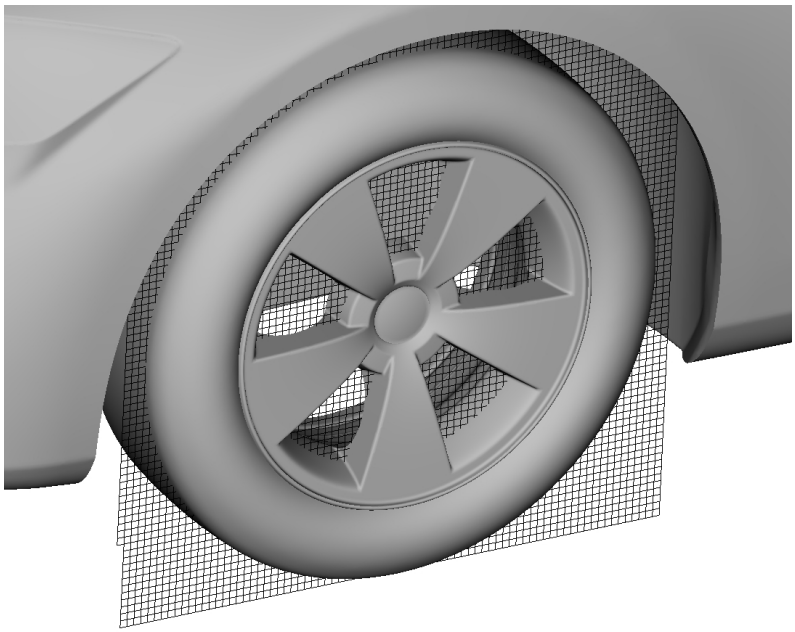
(b) Wing mirror.



(c) Windscreen.



(d) Door handle.



(e) Wheel.

Figure 1.1: Close up of the cut cell mesh around a realistic DrivAer fastback automotive geometry.

1.2 Code development

The code used to produce the results for this work was implemented in 'LSC-AMR'[1], an in-house MPI-parallelised C++ code for Computational Fluid Dynamics developed by the Laboratory for Scientific Computing at the University of Cambridge. LSC-AMR provides high-resolution finite volume solvers to solve hyperbolic conservation laws in an explicit fashion (see Section 1.2.1) on a hierarchical Adaptive Mesh Refinement (AMR) [3] mesh (see Section 1.2.2). Our main contributions to the code base are the following:

1. Calculation of cut cell geometric parameters from a signed distance function, as described in Chapter 2.

2. Implementation of the cut cell solver for hyperbolic conservation laws described in Chapter 3.

3. Implementation of the viscous terms cut cell discretisation and compressible Navier-Stokes equation cut cell solver described in Chapter 4.

4. Implementation of the equilibrium wall function on a cut cell mesh as described in Chapter 5.

We also used Python and MATLAB extensively for prototyping and results post-processing.

Computationally expensive 3D simulations for this work were performed on the *Darwin* and *CSD3* (launched in November 2017) High Performance Computing clusters located at the University of Cambridge.

1.2.1 High-resolution finite volume methods

Consider a general system of conservation laws

$$\partial_t \mathbf{U}(\mathbf{x}, t) + \nabla \cdot \mathbf{F}(\mathbf{U}) = 0, \tag{1.1}$$

where $\mathbf{U}(\mathbf{x}, t)$ is the vector of conserved variables and $\mathbf{F}(\mathbf{U})$ represents the flux tensor. For the 1D Euler equations, for example, $\mathbf{U}(\mathbf{x}, t) = [\rho, \rho u, E]^T$ and $\mathbf{F}(\mathbf{U}) = [\rho u, (\rho u^2 + p), u(E + p)]^T$. ρ, u, p and E are the density, velocity, pressure, and total energy per unit volume respectively, and all four are functions of space and time.

[1]The primary developers of the code base are Dr. Philip Blakely, a Research Associate in the LSC, and Dr. Kevin Nordin-Bates. The latter is no longer based at the LSC.

Integrating Eq. 1.1 over a control volume V (with boundary S) and over a time interval $[t_1, t_2]$ gives, after using Gauss's theorem,

$$\int_V \mathbf{U}(\mathbf{x}, t_2) - \mathbf{U}(\mathbf{x}, t_1) dV + \int_{t_1}^{t_2} \int_S (\mathbf{F} \cdot \hat{\mathbf{n}}) dS dt = \mathbf{0}, \tag{1.2}$$

where $\hat{\mathbf{n}}$ is the outward normal to the control volume. An explicit finite volume scheme to evolve Eq. 1.2 for the computational cell i takes the form:

$$\bar{\mathbf{U}}_i^{n+1} = \bar{\mathbf{U}}_i^n + \frac{\Delta t}{\Delta V} \int_S (\bar{\mathbf{F}}^n \cdot \hat{\mathbf{n}}) dS, \tag{1.3}$$

where $\bar{\mathbf{U}}_i^n$ represents the discrete approximation to the volume averaged conserved variables in cell i at time level n, $\bar{\mathbf{F}}^n$ represents the explicit numerical flux functions, Δt is the explicit stable time step, and ΔV is the volume of the cell. LSC-AMR provides explicit numerical solvers of the form Eq. 1.3 to solve systems of conservation laws.

Since we use dimensional splitting to evolve conservation laws in time, we consider the multidimensional update Eq. 1.3 as split into a series of one-dimensional updates of the form

$$\bar{\mathbf{U}}_i^* = \bar{\mathbf{U}}_i^n + \frac{\Delta t}{\Delta V} (A_{i-1/2} \bar{\mathbf{F}}_{i-1/2}^n - A_{i+1/2} \bar{\mathbf{F}}_{i+1/2}^n), \tag{1.4}$$

where $\bar{\mathbf{F}}_{i\pm1/2}^n$ are the intercell fluxes at the edge of cell i in the current coordinate direction, and $A_{i\pm1/2}$ are the corresponding face areas. $\bar{\mathbf{U}}_i^*$ is the intermediate solution which is used as the Initial Condition for the next dimensional sweep.

LSC-AMR provides an implementation of a number of 'high-resolution' finite volume methods to compute the explicit intercell fluxes. Such methods are designed to have second (or higher) order of accuracy in smooth parts of the solution, while also being able to capture discontinuities without introducing spurious numerical oscillations. For the simulations in this work, we compute hyperbolic fluxes using the second order MUSCL-Hancock high resolution scheme [9], which we summarise in Section 1.2.1.1.

1.2.1.1 MUSCL-Hancock scheme

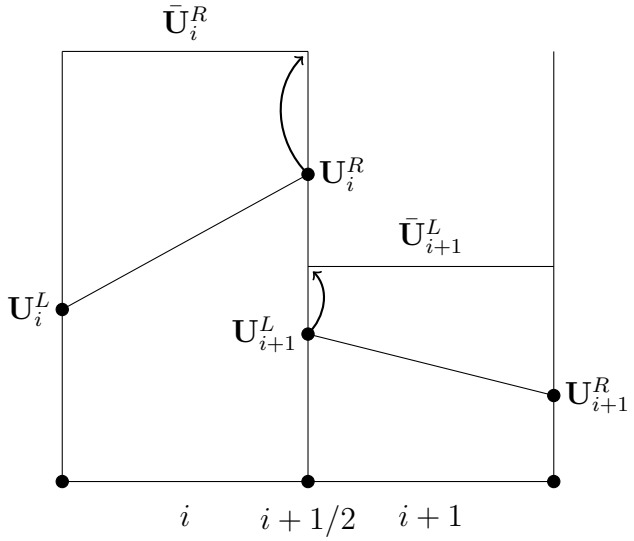


Figure 1.2: Illustration of the intercell flux calculation procedure for the MUSCL-Hancock method.

Consider Fig. 1.2, which illustrates the procedure used to compute the MUSCL-Hancock flux at the interface $i+1/2$ between two adjacent cells i and $i+1$ in the current coordinate direction. Starting from a piecewise constant representation of the solution in each cell, the first step involves using central differences to compute the slopes,

$$\boldsymbol{\Delta}_i = \frac{\mathbf{U}_{i+1}^n - \mathbf{U}_{i-1}^n}{2\Delta x}, \tag{1.5}$$

at each cell centre. Δx is the grid spacing in the current coordinate direction. In order to produce a Total Variation Diminishing (TVD) scheme that prevents the appearance of spurious oscillations in the vicinity of discontinuities, 'limited' slopes $\boldsymbol{\Delta}_i^*$ are calculated using a suitable limiter function ξ_i, such that

$$(\boldsymbol{\Delta}_i^*)_k = (\xi_i)_k (\boldsymbol{\Delta}_i)_k, \tag{1.6}$$

where k is the vector component index. In this work, we make use of the van Leer limiter,

$$(\xi_i)_k^{\mathrm{vl}} = \begin{cases} 0 & , \text{ if } r_k \leq 0, \\ \frac{2}{1+r_k}\min(1, r_k) & , \text{ if } r_k > 0, \end{cases} \tag{1.7}$$

where r_k is the ratio of successive gradients,

$$r_k = \frac{(\mathbf{U}_i^n)_k - (\mathbf{U}_{i-1}^n)_k}{(\mathbf{U}_{i+1}^n)_k - (\mathbf{U}_i^n)_k}. \tag{1.8}$$

As illustrated in Fig. 1.2, the piecewise linear reconstruction can used to compute the 'boundary extrapolated values'

$$\mathbf{U}_i^L = \mathbf{U}_i - \frac{1}{2}\Delta x \boldsymbol{\Delta}_i^*, \tag{1.9}$$

$$\mathbf{U}_i^R = \mathbf{U}_i + \frac{1}{2}\Delta x \boldsymbol{\Delta}_i^*, \tag{1.10}$$

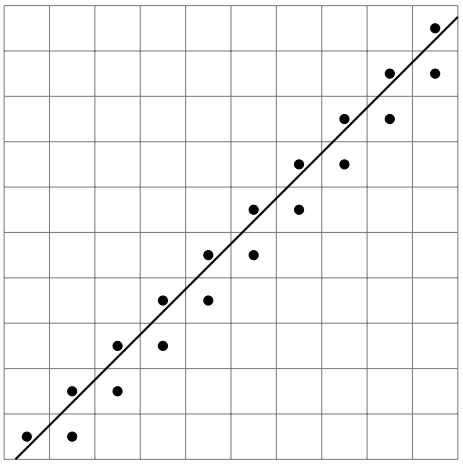
in each cell. These are then evolved by half a time step as follows:

$$\bar{\mathbf{U}}_i^L = \mathbf{U}_i^L + \frac{1}{2}\frac{\Delta t}{\Delta x}(\mathbf{F}(\mathbf{U}_i^L) - \mathbf{F}(\mathbf{U}_i^R)), \tag{1.11}$$

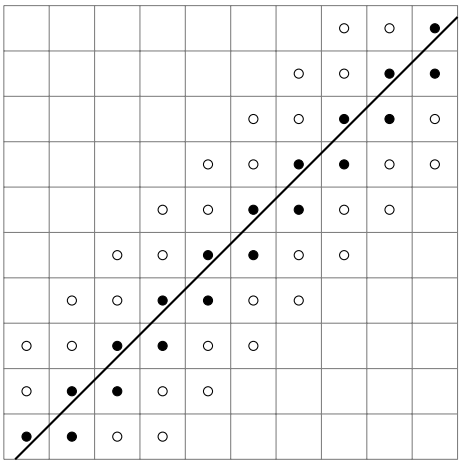
$$\bar{\mathbf{U}}_i^R = \mathbf{U}_i^R + \frac{1}{2}\frac{\Delta t}{\Delta x}(\mathbf{F}(\mathbf{U}_i^L) - \mathbf{F}(\mathbf{U}_i^R)). \tag{1.12}$$

Finally, the Riemann problem defined by the piecewise constant states $\bar{\mathbf{U}}_i^R$ and $\bar{\mathbf{U}}_{i+1}^L$ is solved to calculate the intercell flux at the interface $i + 1/2$. For the simulations in this work, we use an exact Riemann solver for the Euler equations. The reader is referred to Toro [9] for details of its implementation. It may be noted that the calculation of the flux at the interface $i + 1/2$ requires information from two cells on either side of the interface ($i - 1$, i, $i + 1$ and $i + 2$).

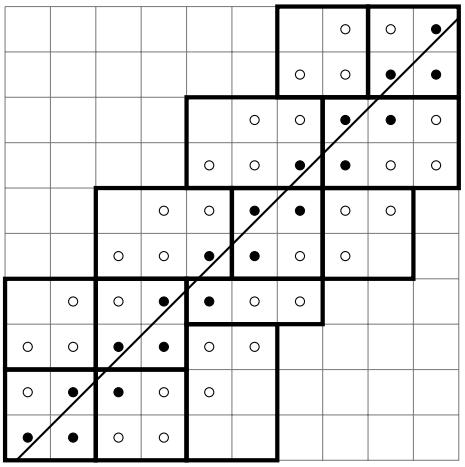
1.2.2 Hierarchical Adaptive Mesh Refinement



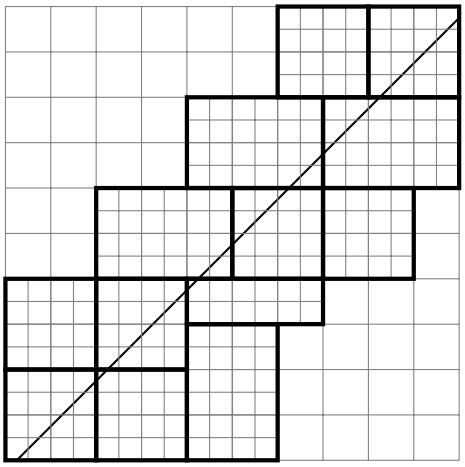
(a) Flagging (flagged cells are marked with filled black circles).



(b) Buffering (cells in the buffer region are marked with filled white circles).



(c) Clustering (patch boundaries are highlighted with bold lines).



(d) Refinement.

Figure 1.3: Illustration of the flagging, buffering, clustering, and refinement procedures involved in refining around a region of interest in the AMR approach.

LSC-AMR provides an implementation of the Hierarchical Adaptive Mesh Refinement (AMR) technique of Berger and Oliger [3]. This allows for finer resolutions to be used only in regions of interest such as at the cut cell interface, or in the vicinity of a shock. We provide a brief summary of the approach here.

Rectangular patches make up the AMR hierarchy, in which refined patches overlay coarser ones. The solution is maintained on every AMR level. Consider Fig. 1.3a, which illustrates the coarsest AMR level on a two-dimensional mesh, and a particular feature of

interest around which we wish to refine. The feature may be a cut cell interface, or indeed a shock wave diagonal to the grid.

The first step involves identifying the cells containing the feature, and 'flagging' them for refinement. For the simulations in this work, we always flag all cut cells for refinement. A cut cell can be identified trivially by comparing its volume to that of an uncut cell.

Furthermore, for subsonic problems, we typically manually specify a boxed region in which all cells are also flagged for refinement. This is useful, for example, when refining the region containing the wake behind a bluff body.

For supersonic problems, the cells neighbouring a moving shock wave are identified dynamically and flagged as per the criteria

$$\frac{|\nabla\rho|^2}{\rho^2} > \Omega, \tag{1.13}$$

where ρ is the density in the cell, and Ω is a user-defined tolerance.

Fig. 1.3a shows the cells flagged for refinement around the feature of interest. To prevent having to regrid the AMR hierarchy every time step when tracking moving shock waves, the flagged cells are padded by a buffer region of additional flagged cells, as illustrated in Fig. 1.3b. Since the CFL stability condition on explicit numerical schemes prevents the shock wave from travelling more than one cell width in each time step [9], the extent of the buffer region is set to correspond to the regridding frequency.

As illustrated in Fig. 1.3c, the flagged cells are then clustered into rectangular patches for refinement. Fig. 1.3d illustrates the result for a refinement factor of 2. The data in fine cells is initialised via interpolation of the parent coarse cell data. Note that the method of Berger and Rigoutsos [10] is the default clustering algorithm used in LSC-AMR.

If additional refinement is required, the same process of flagging, buffering, clustering and refinement can be carried out on the newly created finer level. When evolving the entire AMR hierarchy in time, the levels are evolved in the order of coarsest to finest, with sub-cycling used on the finer levels in order to maintain the stability of the solution across the hierarchy.

CHAPTER 2

CUT CELL MESH GENERATION

In this chapter, we describe the procedure that we use to generate a Cartesian cut cell mesh around a geometry. As discussed in Chapter 1, this results in the creation of arbitrarily small cut cells at the interface. The mesh generation technique is independent of the approach used to efficiently evolve the solution in the cut cells.

In Section 2.1, we provide a description of the approach used to calculate the 'core' geometric parameters which would be required by any cut cell method. In Section 2.2, we describe the calculation procedure for additional geometric parameters required by our split cut cell scheme.

2.1 Calculation of core geometric parameters

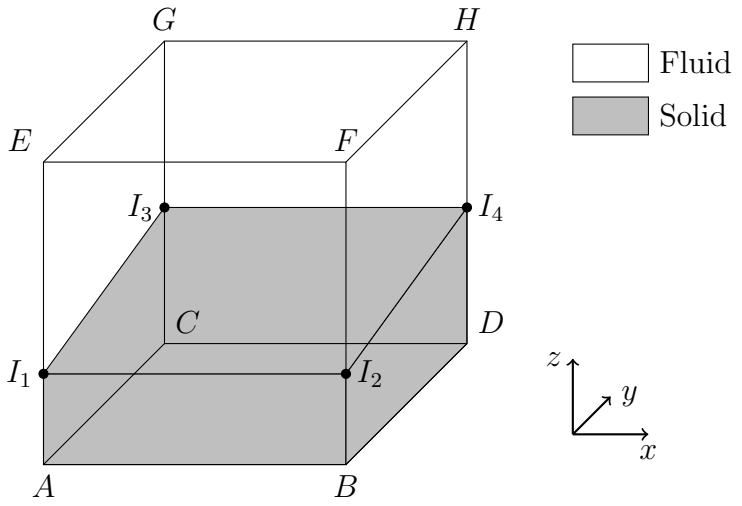


Figure 2.1: Illustration of a cut cell in 3D.

Consider Fig. 2.1, which shows a 3D cut cell where the solid-fluid interface intersects the cell at four points I_1, I_2, I_3 and I_4. The geometric parameters to be calculated are the following:

(i) The face fraction, $\beta \in [0, 1]$, of each cell face. This represents the fluid area of the face non-dimensionalised by total cell face area.

(ii) The area, A^b, of the reconstructed interface in the cell and $\hat{\mathbf{n}}^b$, the interface unit normal. The superscript b is short for 'boundary'.

(iii) The volume fraction, $\alpha \in [0, 1]$, of the cell. This is the fluid volume of the cell non-dimensionalised by the total cell volume.

(iv) The volumetric centroid, $\mathbf{x}_c$, of the fluid part of the cell.

(v) The interface centroid, $\mathbf{x}_c^b$ of the reconstructed solid interface.

We proceed by treating the interface implicitly as the zero level-set of a signed distance function. The technique of Mauch [11] is used to compute the signed distance function, $\phi(\mathbf{x})$, at the vertices of the cell, and this information is used to reconstruct all the required geometric information. The LSC-AMR code contains parallelised CPU and GPU implementations of the Mauch algorithm.

2.1.1 Multiply cut cells

Note that the mesh generation procedure that we proceed to describe is only valid when the intersection of the cell and geometry can be described by a single interface. Sufficient resolution must therefore be used to ensure that all cut cells are 'singly cut'. The signed distance function can be used to deal with 'split cells' created by multiple intersections of the geometry with the cell by using a 'Marching Cubes' [12] based approach as in Gunther et al. [13], but this is beyond the scope of this work.

2.1.2 Calculation of intersection points

Consider edge AE in Fig. 2.1. If it is intersected by the interface, i.e., if the signed distances at points A and E are of opposite sign, then assuming a linear interface in the vicinity of the edge, the non-dimensional distance from A to I_1 may be calculated to be

$$\frac{AI_1}{AE} = -\left(\frac{\phi_A}{\phi_E - \phi_A}\right), \tag{2.1}$$

where ϕ_A and ϕ_E are the signed distances at vertices A and E respectively. This approach can be used to determine the coordinates of all the intersection points.

2.1.3 Calculation of face fractions

With the coordinates of the intersection points identified, calculating the face fractions for each face in 3D is a matter of making use of the formula for the area, A_{poly}, of an arbitrary non-self-intersecting polygon with n ordered vertices $(x_1, y_1), ..., (x_n, y_n)$ [14]:

$$\beta_{\text{face}} = \frac{A_{\text{poly}}}{A_{\text{face}}} = \frac{1}{2A_{\text{face}}} \sum_{i=1}^{n} (x_i y_{i+1} - x_{i+1} y_i), \tag{2.2}$$

where we non-dimensionalise the result with the total area of the cell face, A_{face}. Note that the summation index is periodic so that $(n+1) = 1$.

In a 2D simulation, on the other hand, the face fractions would be obtained by the appropriate use of Eq. 2.1.

2.1.4 Interface area and normal calculation

As shown by Pember et al. [15], the unit interface normal (pointing into the solid), $\hat{\mathbf{n}}^b_{i,j,k}$, and the interface area $A^b_{i,j,k}$ for a cell (i, j, k) can be computed from the face fractions as

follows:

$$\begin{aligned} A^b_{i,j,k}\hat{\mathbf{n}}^b_{i,j,k} &= \Delta y\Delta z(\beta_{i-1/2,j,k} - \beta_{i+1/2,j,k})\hat{\mathbf{i}} \\ &+ \Delta x\Delta z(\beta_{i,j-1/2,k} - \beta_{i,j+1/2,k})\hat{\mathbf{j}} \\ &+ \Delta x\Delta y(\beta_{i,j,k-1/2} - \beta_{i,j,k+1/2})\hat{\mathbf{k}}. \end{aligned} \tag{2.3}$$

It may be noted that Eq. 2.3, which is derived from the divergence theorem, reduces naturally to 2D.

2.1.5 Volume fraction calculation

The volume fraction of the 3D cell can be computed from the formula of the volume of a general polyhedron with N_F faces [16]:

$$\alpha = \frac{1}{3V_{\text{cell}}}\left|\sum_{i=1}^{N_F}(\bar{\mathbf{x}}_\mathbf{i}\cdot\hat{\mathbf{n}}_\mathbf{i})A_i\right|, \tag{2.4}$$

where $\bar{\mathbf{x}}_\mathbf{i}$ is any point on face i, and $\hat{\mathbf{n}}_\mathbf{i}$ and A_i are the outward unit normal and area of face i respectively. Note that we non-dimensionalise the result by the total volume of the cell, V_{cell}.

Cut cells in 2D are like cut cell faces in 3D so in a 2D simulation, Eq. 2.2 would be used to compute the volume fraction of the cut cell.

2.1.6 Volumetric centroid calculation

In 3D, the volumetric centroid $\mathbf{x}_c$ can be calculated using the formula for the centroid of an arbitrary polyhedron with N_F faces [17]

$$\mathbf{x}_c = \frac{3}{4}\frac{\left[\sum_{i=1}^{N_F}(\mathbf{x}^\mathbf{c}_\mathbf{i}\cdot\hat{\mathbf{n}}_\mathbf{i})\mathbf{x}^\mathbf{c}_\mathbf{i}A_i\right]}{\left[\sum_{i=1}^{N_F}(\mathbf{x}^\mathbf{c}_\mathbf{i}\cdot\hat{\mathbf{n}}_\mathbf{i})A_i\right]}, \tag{2.5}$$

where $\mathbf{x}^\mathbf{c}_\mathbf{i}$ is the centroid of face i (computed by the appropriate use of Eq. 2.6-Eq. 2.7), and $\hat{\mathbf{n}}_\mathbf{i}$ and A_i are the outward unit normal and area of face i respectively.

In a 2D simulation, the fluid part of a cut cell is a polygon. $\mathbf{x}_c = [\mathbf{x}_{c,x}, \mathbf{x}_{c,y}]^T$ can therefore be worked out using the formula for the centroid of a non-self-intersecting polygon

with n ordered vertices $(x_1, y_1), ..., (x_n, y_n)$ [18]

$$\mathbf{x}_{c,x} = \frac{1}{6A_{\text{poly}}} \sum_{i=1}^{n} (x_i + x_{i+1})(x_i y_{i+1} - x_{i+1} y_i), \tag{2.6}$$

$$\mathbf{x}_{c,y} = \frac{1}{6A_{\text{poly}}} \sum_{i=1}^{n} (y_i + y_{i+1})(x_i y_{i+1} - x_{i+1} y_i), \tag{2.7}$$

where A_{poly} is calculated as in Eq. 2.2. Note that once again, the summation index is periodic so that $(n + 1) = 1$.

2.1.7 Interface centroid calculation

To compute the interface centroid $\mathbf{x}_c^b$ in 3D, we follow the following procedure:

1. Rotate the interface plane with n vertices $(x_1, y_1, z_1), ..., (x_n, y_n, z_n)$ to give a polygon aligned with the x-y Cartesian plane. This polygon has vertices $(x_1^{\text{rot}}, y_1^{\text{rot}}, z^{\text{rot}})$, $..., (x_n^{\text{rot}}, y_n^{\text{rot}}, z^{\text{rot}})$. Note that because the interface reconstructed from the signed distance function may not be perfectly planar, the z coordinates of the rotated interface may differ slightly from one another. In practice, we set z^{rot} to be the numerical average of these varying z coordinates. Note also that the columns of the required rotation matrix are the unit vectors of the orthonormal coordinate system with a unit vector pointing normal to the cell interface, and unit vector(s) in the interface tangential plane.

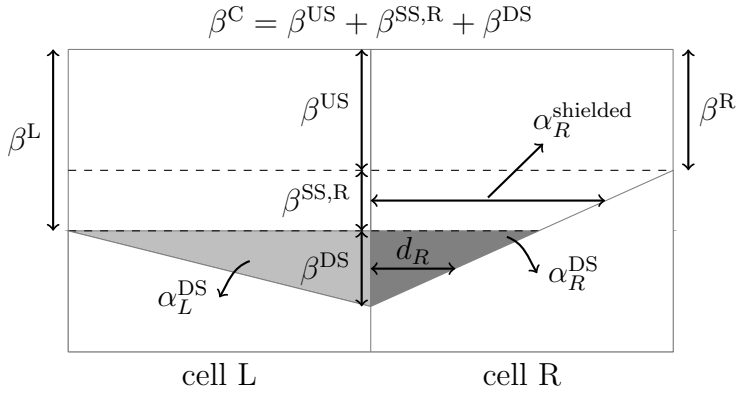
2. Use Eq. 2.6-Eq. 2.7 on the polygon with ordered vertices $(x_1^{\text{rot}}, y_1^{\text{rot}}), ..., (x_n^{\text{rot}}, y_n^{\text{rot}})$ to get $\mathbf{x}_{c,x}^{b,\text{rot}}$ and $\mathbf{x}_{c,y}^{b,\text{rot}}$, which are the x and y coordinates of the interface centroid in the rotated frame, $\mathbf{x}_c^{b,\text{rot}}$. Note that $\mathbf{x}_c^{b,\text{rot}} = [\mathbf{x}_{c,x}^{b,\text{rot}}, \mathbf{x}_{c,y}^{b,\text{rot}}, z^{\text{rot}}]^T$.

3. Rotate $\mathbf{x}_c^{b,\text{rot}}$ back to the original frame to give $\mathbf{x}_c^b$ as required.

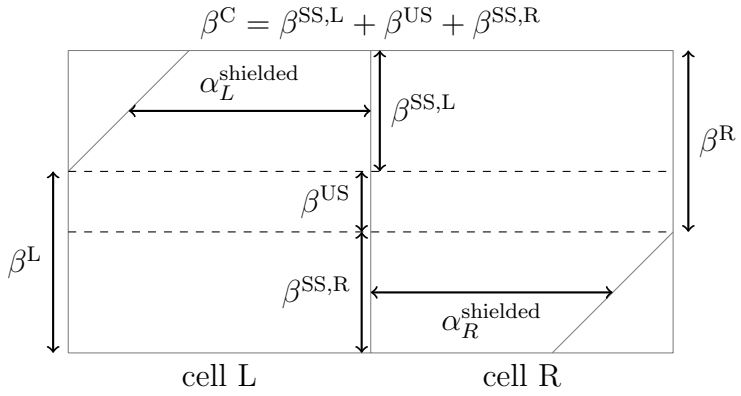
In 2D, since we assume that cut cells are singly cut, $\mathbf{x}_c^b$ is worked out trivially as the average of the positions of the two intersection points.

2.2 Calculation of extra geometric parameters

In this section, we describe the calculation procedure of some additional geometric parameters required by the split scheme. Consider Fig. 2.2, which illustrates two possible cut cell configurations for neighbouring cells in an arbitrary dimensional sweep in 2D. Note that our discussion applies equally to 3D as well.



(a) Concave configuration.



(b) Narrow channel configuration.

Figure 2.2: Illustration of the additional geometric parameters required by the split cell method for two configurations.

The split method requires the common face $I_{\mathrm{C}}^{\mathrm{face}}$ between the neighbouring cells to be divided into the following regions:

- The 'unshielded' (US) region which does not 'face' a boundary in the current sweep direction.
- The 'singly-shielded from the left' (SS,L) region which is 'covered' by the boundary from the left.
- The 'singly-shielded from the right' (SS,R) region which is covered by the boundary from the right.
- The 'doubly-shielded' (DS) region which faces the boundary from the left and right.

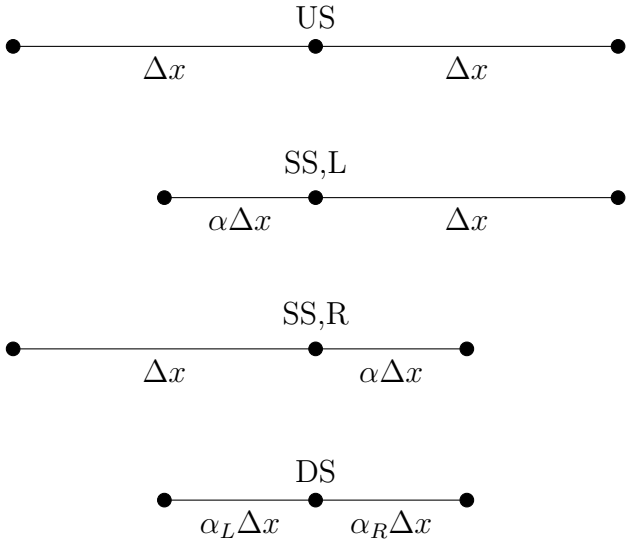


Figure 2.3: One-dimensional stencils corresponding to the unshielded (US), singly-shielded from the left (SS,L), singly-shielded from the right (SS,R) and doubly-shielded (DS) configurations.

Each region has its own face fraction, as labelled in Fig. 2.2a and Fig. 2.2b. Consider Fig. 2.3, which illustrates the one-dimensional stencils corresponding to the four regions. In the split method, the flux on the unshielded part of the interface does not need to be stabilised since it corresponds to the 1D configuration where the cells to its immediate left and right are uncut. The singly-shielded from the left, and singly-shielded from the right parts, correspond, respectively, to the 1D configurations where a cut cell exists to the immediate left or right of the interface. The flux acting on that region therefore needs to be stabilised. The most challenging region for the method to deal with is the doubly-shielded region, which corresponds to the 1D configuration where cut cells exist to the immediate left and right of the interface. We describe our flux stabilisation procedure for all the shielded configurations in Chapter 3.

In a singly-shielded region, $\alpha_K^{\text{shielded}}$ (where K=L, R as appropriate) represents the average distance from $I_{\text{C}}^{\text{face}}$ to the boundary in the current coordinate direction, non-dimensionalised by the corresponding regular cell spacing. α_L^{DS} and α_R^{DS} are the fluid volume fractions in the doubly-shielded regions of the left and right cells respectively.

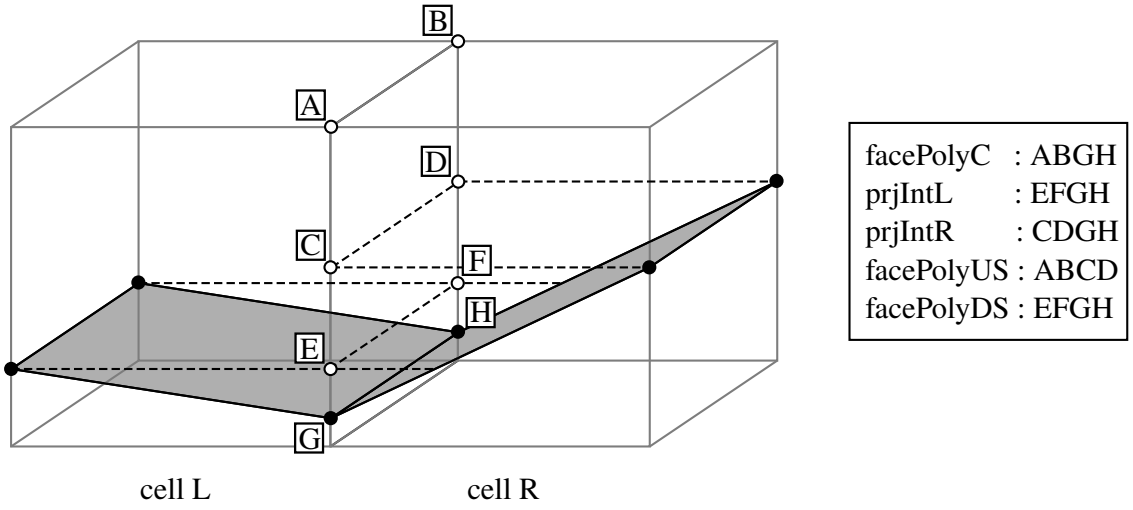


Figure 2.4: Illustration of the deconstruction of $I_{\mathrm{C}}^{\mathrm{face}}$ into constituent polygons used for the calculation of cut cell geometric parameters. The polygon names are shown to the right.

We proceed to outline how all of these additional geometric parameters are computed. Part of the procedure involves deconstructing $I_{\mathrm{C}}^{\mathrm{face}}$ into constituent polygons of interest for the purpose of calculating the geometric parameters. Fig. 2.4 illustrates this process for a three-dimensional configuration in an arbitrary dimensional sweep. Note that extensive use was made of the 'Boost.Geometry' [19] part of the Boost C++ Libraries when implementing the code for this section.

Step 1: Construct the fluid face polygon on $I_{\mathrm{C}}^{\mathbf{face}}$

Use the intersection points of the geometry and the grid on $I_{\mathrm{C}}^{\mathrm{face}}$ to construct the polygon 'facePolyC' that borders the fluid part of $I_{\mathrm{C}}^{\mathrm{face}}$. In 2D, the fluid part of $I_{\mathrm{C}}^{\mathrm{face}}$ is a 'line' which is converted into a polygon using extrusion by unit depth.

Step 2: Projection of the interfaces on $I_{\mathrm{C}}^{\mathbf{face}}$

Project the left and right interface planes on $I_{\mathrm{C}}^{\mathrm{face}}$ in the current coordinate direction to give the polygons 'prjIntL' and 'prjIntR' respectively. In 2D, note that we extrude the interface 'planes' by a unit depth before carrying out the projection.

Step 3: Calculation of $\beta^{\mathbf{US}}$ and $\beta^{\mathbf{DS}}$

Subtract prjIntL and prjIntR, in turn, from facePolyC to give the unshielded polygon 'facePolyUS'. By 'subtraction', we refer to the spatial set theoretic difference of two geometries. The unshielded face fraction β^{US} is the area of facePolyUS.

The doubly-shielded face fraction β^{DS} is calculated from the area of 'facePolyDS', which is the polygon given by the spatial set theoretic intersection of prjIntL and prjIntR.

Step 4: Calculation of $\alpha_L^{\mathbf{DS}}$ and $\alpha_R^{\mathbf{DS}}$

If $\beta^{\mathrm{DS}} \neq 0$, we also have to compute the doubly-shielded volume fractions. These are given by the product of β^{DS} and d_K (where K=L, R as appropriate). d_K is the non-dimensional distance, measured in the current coordinate direction, from the centroid of facePolyDS to the interface in the cell. d_R is illustrated for the doubly-shielded part of 'cell R' in Fig. 2.2a.

Let $\mathbf{x}_{\mathrm{c}}^{\mathrm{DS}}$ be the position of the centroid of facePolyDS, and $\mathbf{x}_{\mathrm{P}}^{\mathrm{int}}$ be the position of any of the intersection points of the interface and the cell of interest. Consider the vector $[\mathbf{x}_{\mathrm{P}}^{\mathrm{int}} - (\mathbf{x}_{\mathrm{c}}^{\mathrm{DS}} + d_K\hat{\mathbf{i}}_d)]$ which lies in the interface plane. $\hat{\mathbf{i}}_d$ is the unit vector pointing in the current sweep direction. Then,

$$[\mathbf{x}_{\mathrm{P}}^{\mathrm{int}} - (\mathbf{x}_{\mathrm{c}}^{\mathrm{DS}} + d_K\hat{\mathbf{i}}_d)] \cdot \hat{\mathbf{n}}^{\mathbf{b}} = 0, \tag{2.8}$$

where $\hat{\mathbf{n}}^{\mathbf{b}}$ is the interface unit normal. Re-arranging Eq. 2.8 gives

$$d_k = \frac{(\mathbf{x}_{\mathrm{P}}^{\mathrm{int}} - \mathbf{x}_{\mathrm{c}}^{\mathrm{DS}}) \cdot \hat{\mathbf{n}}^{\mathbf{b}}}{\hat{\mathbf{n}}^{\mathbf{b}} \cdot \hat{\mathbf{i}}_d}, \tag{2.9}$$

which can then be used to calculate the doubly-shielded volume fraction.

Step 5: Calculation of $\beta^{\mathbf{SS,L}}$ and $\beta^{\mathbf{SS,R}}$

To start with, compute the total shielded face fraction:

$$\beta^{\mathrm{shielded}} = \beta^{\mathrm{C}} - \beta^{\mathrm{US}}. \tag{2.10}$$

Then, $\beta^{\mathrm{SS,L}}$ and $\beta^{\mathrm{SS,R}}$ can be worked out in a straightforward fashion to be

$$\beta^{\mathrm{SS,L}} = \max(0.0, \beta^{\mathrm{shielded}} - (\max(\beta^{\mathrm{C}} - \beta^{\mathrm{R}}))), \tag{2.11}$$

$$\beta^{\mathrm{SS,R}} = \max(0.0, \beta^{\mathrm{shielded}} - (\max(\beta^{\mathrm{C}} - \beta^{\mathrm{L}}))). \tag{2.12}$$

Step 6: Calculation of $\alpha_L^{\text{shielded}}$ and $\alpha_R^{\text{shielded}}$

$\alpha_K^{\text{shielded}}$ (where K=L, R as appropriate) is given by

$$\alpha_K^{\text{shielded}} = \frac{\alpha_K^{\text{singly-shielded}}}{\beta^{\text{SS,K}}}, \tag{2.13}$$

where

$$\alpha_K^{\text{singly-shielded}} = \alpha^K - \beta^{\text{K}} - \alpha_K^{\text{DS}}. \tag{2.14}$$

α^K is the volume fraction of the cut cell.

2.3 Conclusion

The procedure described in this chapter can be used to robustly generate a cut cell mesh as long as sufficient resolution is used to ensure that all cut cells remain singly-cut. We describe our dimensionally split method to overcome the small cell problem and evolve the solution on this mesh in the context of the Euler and Navier-Stokes equations in Chapter 3 and Chapter 4 respectively.

CHAPTER 3

A DIMENSIONALLY SPLIT CARTESIAN CUT CELL METHOD FOR HYPERBOLIC CONSERVATION LAWS

In this chapter, we present a novel dimensionally split Cartesian cut cell method that makes use of local geometric and wave speed information to overcome the small cell problem in the context of hyperbolic conservation laws. This content was recently published in the *Journal of Computational Physics* [1]:

N. Gokhale, N. Nikiforakis, and R. Klein. A dimensionally split Cartesian cut cell method for hyperbolic conservation laws. *Journal of Computational Physics*, 364: 186-208, 2018.

3.1 Introduction

Cartesian cut cell methods have been an active area of research since the early 1980s and a number of strategies to overcome the small cell problem have been published in the literature. The reader is encouraged to look up the accessible recent book chapter by Berger [20] for a thorough overview of the methods published to date. In the following, we highlight and briefly describe some of the noteworthy contributions.

Different methodologies exist to overcome the small cell problem. With cell merging, as in Clarke et al. [21], or the related cell linking strategy, as employed by Quirk [22], Kirkpatrick et al. [23] and Hartmann et al. [24], cells with volume lower than a certain threshold are absorbed into larger neighbouring cells. Based on the static boundary cut cell formulation of Hartmann et al. [24], it may be noted that Schneiders et al. [25] have successfully developed a method to compute moving boundary problems in 3D by

introducing an interpolation routine and flux redistribution step. Meinke et al. [26] have used the developed technique to compute the flow in a relatively complicated engine geometry involving moving parts.

Another alternative is to use the simplified 'h-box' method of Berger and Helzel [27]. Here, the domain of dependence of the intercell flux is extended to the regular cell length, h, on a virtual grid of h-boxes in such a way that a 'flux cancellation' occurs, removing the dependence on cell volume from the update formula. Although second order accuracy at the boundary can be achieved with this method, it is quite complicated and has not yet been implemented in three dimensions.

In the 'flux redistribution' approach of Colella et al. [28], conservative but potentially unstable fluxes are initially computed for each cut cell. A stable but non-conservative part of the update is applied to the cut cells, and conservation is maintained by redistributing the remaining part to surrounding cut and uncut cells. A similar approach is the 'flux mixing' technique of Hu et al. [29]. Here, the explicit fluxes are used to update all cells, following which the solution in the cut cells is mixed with neighbouring cells. This technique has been extended and used in the context of a number of different applications. Grilli et al. [30] have successfully used it at the compression ramp geometry in their study on shockwave turbulent boundary layer interaction. Pasquariello et al. [31] use it at the interface in the coupled finite volume-finite element method they develop to study fluid-structure interaction problems. Furthermore, it may be noted that Muralidharan and Menon [32] have recently managed to use it to develop a third order accurate method.

Another noteworthy approach to obtain high order convergence is the 'inverse Lax-Wendroff' method of Tan and Shu [33]. In a subsequent paper describing an efficient implementation of the technique, Tan et al. [34] are able to demonstrate fifth order convergence for two-dimensional problems for the Euler equations.

'Explicit-implicit' approaches as developed by Jebens et al. [35], or more recently, by May and Berger [36], are also an interesting area of development. Here, the attempt is to develop a combined scheme where regular and cut cells are integrated explicitly and implicitly in time respectively.

All the aforementioned methods have their own relative merits and are implemented in an unsplit fashion. As discussed in Chapter 1, we are particularly interested in adopting a dimensionally split approach which is a simple way to extend one-dimensional methods for hyperbolic conservation laws to multidimensional problems. To that end, we make use of the framework introduced by Klein, Bates and Nikiforakis [37] which provides a description of how cut cell updates can be performed in a split fashion. The stabilised 'KBN' cut cell flux that they devise makes use of local geometric information. In this chapter, we

present a 'Localised Proportional Flux Stabilisation' (LPFS) approach which makes use of local geometric and wave speed information to define a novel cut cell flux. Numerical tests indicate that the LPFS flux alleviates the problem of oscillatory boundary solutions produced by the KBN flux at higher Courant numbers, and enables the computation of more accurate solutions near stagnation points.

The rest of the chapter is organised as follows. We outline the governing equations and explicit numerical schemes that we use in Section 3.2. In Section 3.3, we describe the derivation of the KBN and LPFS fluxes, and describe their multi-dimensional extensions in the framework of Klein et al. In Section 3.4, a theoretical convergence and stability analysis of the LPFS method for the model one-dimensional linear advection equation is presented. In Section 3.5, we present numerical solutions for a number of multi-dimensional test problems to demonstrate the performance of the LPFS method. Finally, conclusions and areas for future work are provided in Section 3.6.

3.2 Governing equations and solution framework

We use the linear advection equation,

$$\partial_t u + \mathbf{a} \cdot \nabla u = 0, \tag{3.1}$$

when proving the convergence and stability of LPFS in Section 3.4, and for some convergence tests in Section 3.5.1. u is the variable being advected at constant velocity $\mathbf{a}$.

For more challenging tests, we solve the compressible, unsteady, Euler equations

$$\begin{aligned} \partial_t \rho + \nabla \cdot (\rho \mathbf{u}) &= 0, \\ \partial_t (\rho \mathbf{u}) + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u} + pI) &= \mathbf{0}, \\ \partial_t E + \nabla \cdot [(E + p)\mathbf{u}] &= 0. \end{aligned} \tag{3.2}$$

In Eq. 3.2, ρ is density, $\mathbf{u}$ is velocity, p is pressure and I is the identity matrix. E is the total energy per unit volume, given by

$$E = \rho \left(\frac{1}{2} |\mathbf{u}|^2 + e \right), \tag{3.3}$$

where e is the specific internal energy. To close the system of equations Eq. 3.2-Eq. 3.3

we use the ideal gas equation of state

$$e = \frac{p}{\rho(\gamma - 1)}, \tag{3.4}$$

where γ, the heat capacity ratio, is assumed to be 1.4.

We compute explicit fluxes in a Godunov-based finite volume framework using an exact Riemann solver and the MUSCL-Hancock scheme in conjunction with the van-Leer limiter [9]. This scheme is second order accurate in smooth regions. The definition of the KBN and LPFS fluxes are independent of the particular choice of flux method used, however. We also make use of hierarchical AMR [3] to refine areas of interest such as shock waves, or the cut cell interface, while allowing the use of coarser resolutions elsewhere for the sake of computational efficiency. Multi-dimensional updates are performed using Strang splitting [38] in 2D, and straightforward Godunov splitting [39] in 3D.

The time step, Δt, is restricted by the CFL condition

$$\Delta t = C_{\text{cfl}} \min_{d,i} \left(\frac{\Delta x_{d,i}}{W_{d,i}^{\max}} \right), \tag{3.5}$$

where d is the index of the coordinate direction, i is the index of a computational cell, and $\Delta x_{d,i}$ and $W_{d,i}^{\max}$ are the spatial resolution and max wave speed for cell i in the d direction respectively. The MUSCL-Hancock scheme has a linearised stability constraint $C_{\text{cfl}} \in (0, 1]$ [9], where C_{cfl} is the Courant number.

For the Euler equations, the wave speed for cell i in the d direction, $W_{d,i}$, is computed using the following estimate suggested by Toro [9]:

$$W_{d,i} = |\mathbf{u}_{d,i}| + a_i, \tag{3.6}$$

where $\mathbf{u}_{d,i}$ is the component of the velocity in cell i in the d direction. a_i is the speed of sound in cell i, given by

$$a_i = \sqrt{\frac{\gamma p_i}{\rho_i}}. \tag{3.7}$$

Domain edge boundary conditions are specified using the 'fictitious cell' approach [9]. Depending on the physics of the particular boundary condition, the flow variables in the domain edge ghost cells are either fully or partially specified, and any unspecified variables are extrapolated from the interior of the domain. For a supersonic inflow boundary condition, for example, the ghost cell states are completely specified because all characteristics in the solution of the boundary Riemann problem point to the right.

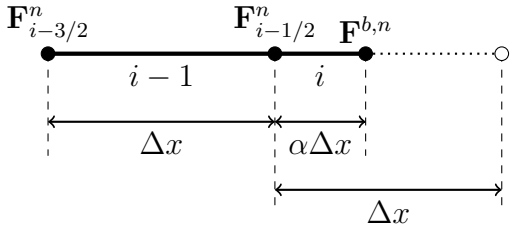


Figure 3.1: Illustration of the KBN flux stabilisation procedure for a boundary cut cell neighbouring a regular cell in 1D.

Conversely, at a supersonic outlet or 'transmissive' boundary, all ghost cell variables are extrapolated from within the domain. At reflective boundaries, ghost variables are extrapolated from the domain, but the sign of the normal velocity component is inverted. The reader is referred to Laney [40] for a comprehensive overview of the topic of enforcing boundary conditions.

The two other domain edge boundary conditions used in this chapter are subsonic inflow, and subsonic outflow. We use the specifications implemented in LSC-AMR which were found to work well in practice. For subsonic inflow, the ghost cell momentum and entropy are specified, while density is extrapolated from within domain. At a subsonic outflow boundary, only the ghost pressure is specified with all other values being extrapolated from the domain.

3.3 Numerical method

In this section, we describe the derivation of the one-dimensional KBN and LPFS fluxes, as well as their multi-dimensional extensions. A convergence and stability analysis of LPFS is presented in Section 3.4.

3.3.1 The KBN flux

Consider Fig. 3.1, which shows a boundary cut cell neighbouring a regular cell in 1D. Let $\mathbf{U}^n_i$ represent the conserved variable state vector for cell i at time level n, and let $\mathbf{F}^n_{i\pm i/2}$ represent the explicit numerical fluxes computed at its ends.

A creative reasoning is used to compute the stabilised cut cell flux, $\mathbf{F}^{\mathrm{KBN},n}_{i-1/2}$. In an explicit finite volume scheme, the state at the new time level is computed from the intercell fluxes at the old time level. On the other hand, if the state at the new time level were known, one could work out the stable intercell flux. An estimate of the new state, $\bar{\mathbf{U}}^{n+1}_i$, is calculated by extending the 'influence' of the cut cell to the regular cell length (this is

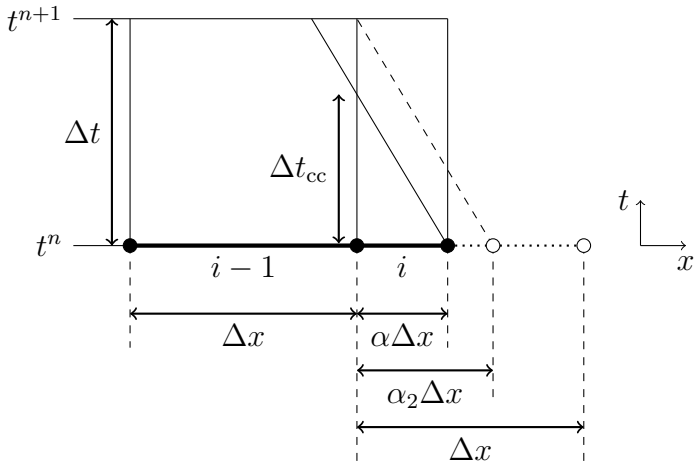


Figure 3.2: Illustration of the LPFS flux stabilisation procedure for a boundary cut cell neighbouring a regular cell in 1D.

illustrated by the dotted line in Fig. 3.1):

$$\bar{\mathbf{U}}_i^{n+1} = \mathbf{U}_i^n + \frac{\Delta t}{\Delta x}(\mathbf{F}_{i-i/2}^n - \mathbf{F}^{b,n}). \tag{3.8}$$

The actual 1D conservative update is:

$$\hat{\mathbf{U}}_i^{n+1} = \mathbf{U}_i^n + \frac{\Delta t}{\alpha\Delta x}(\mathbf{F}_{i-1/2}^{\mathrm{KBN},n} - \mathbf{F}^{b,n}). \tag{3.9}$$

Equating the right hand sides of Eq. 3.8 and Eq. 3.9 provides the expression for the stabilised flux:

$$\mathbf{F}_{i-1/2}^{\mathrm{KBN},n} = \mathbf{F}^{b,n} + \alpha(\mathbf{F}_{i-i/2}^n - \mathbf{F}^{b,n}), \tag{3.10}$$

which is used to update the cut cell and its neighbour (making the scheme conservative) at the regular time step Δt. Note that Eq. 3.10 is consistent with respect to the natural limits of the grid so that as $\alpha \to 1$, $\mathbf{F}_{i-1/2}^{\mathrm{KBN},n} \to \mathbf{F}_{i-i/2}^n$, and as $\alpha \to 0$, $\mathbf{F}_{i-1/2}^{\mathrm{KBN},n} \to \mathbf{F}^{b,n}$. When used to update the cut cell i, the flux $\mathbf{F}_{i-1/2}^{\mathrm{KBN},n}$ provides stability and conservation by applying $\alpha(\mathbf{F}_{i-i/2}^n - \mathbf{F}^{b,n})$ to the cut cell, and applying the remaining flux difference, $(1-\alpha)(\mathbf{F}_{i-i/2}^n - \mathbf{F}^{b,n})$ to the neighbouring uncut cell. In that respect, the one-dimensional flux stabilisation approach is related to the flux redistribution approach of Colella et al. [28].

3.3.2 The LPFS flux

From Eq. 3.10, it may be seen that the KBN flux uses the geometric parameter α to determine a stabilised flux. Here, we describe the use of geometric and wave speed information to define a new stabilised cut cell flux.

Consider Fig. 3.2, which shows a boundary cut cell neighbouring a regular cell in the x-t plane for one time step. Δt is the global stable time step which is determined in part by the fastest wave speed in the domain, $W_{\max}$ (see Eq. 3.5). Intuitively, the CFL time step restriction requires that no wave from the solution of the local Riemann problems should travel more than one cell width during the time step. For the configuration of Fig. 3.2, we illustrate the 'small cell problem' at the cut cell as being caused by the left-going wave from the solution of the boundary Riemann problem. Stability would therefore require the use of the smaller Δt_{cc}:

$$\Delta t_{\text{cc}} = C_{\text{cfl}} \frac{\alpha \Delta x}{W_i}, \tag{3.11}$$

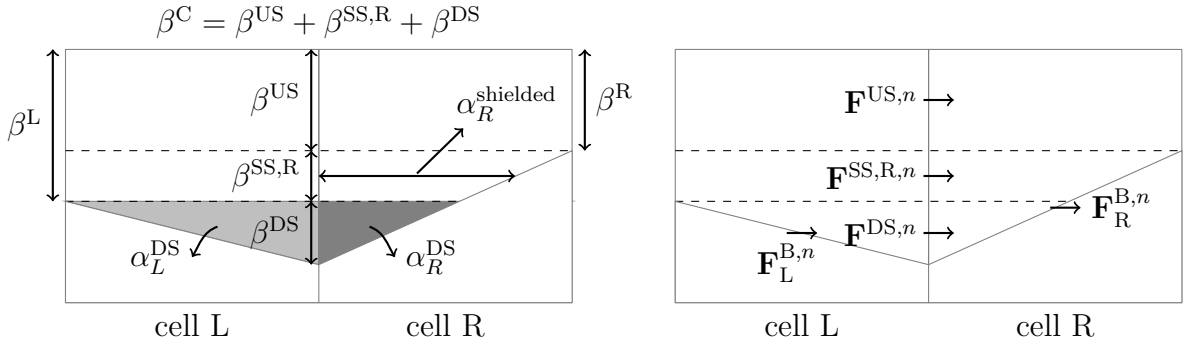
where W_i is the wave speed for the cut cell.

In the LPFS approach, we use the explicit flux $\mathbf{F}^n_{i-i/2}$ for the part of the time step for which it is stable, Δt_{cc}, and employ a different flux which can maintain stability for the duration $(\Delta t_{\text{cc}}, \Delta t]$. This gives the LPFS method an inherent advantage over the KBN method in regions of low velocity. Although $\alpha < 1$ depresses Δt_{cc} relative to Δt, part of the reduction is offset if $W_i < W_{\max}$, an effect which is most pronounced in regions of low velocity near a stagnation point. For larger cut cells, we found the ratio $\Delta t_{\text{cc}}/\Delta t$ to sometimes be greater than 1. The flux in this case requires no stabilisation which we allow for in LPFS by requiring that $\Delta t_{\text{cc}}/\Delta t \leq 1$. Although we only deal with inviscid flows in this chapter, it may be noted that in viscous flows, low velocity regions next to the solid boundary occur also due to the boundary layer.

There is room for creativity in the definition of the flux used in the interval $(\Delta t_{\text{cc}}, \Delta t]$, and we define a modified version of the original KBN flux to be used there. As described in Section 3.3.1, the KBN flux is derived by extending the influence of the cut cell to the full cell length. We interpret this idea in the picture of Fig. 3.2 and propose extending the influence of the cell only as far as necessary so that no wave-interface intersections occur from the solutions of the cut cell Riemann problems. This means extending the influence to a length of $\alpha_2 \Delta x$ instead of Δx and results in the definition of a modified KBN flux:

$$\mathbf{F}^{\text{KBN,mod},n}_{i-1/2} = \mathbf{F}^{b,n} + \frac{\alpha}{\alpha_2}(\mathbf{F}^n_{i-i/2} - \mathbf{F}^{b,n}). \tag{3.12}$$

Like $\mathbf{F}^{\text{KBN},n}_{i-1/2}$, $\mathbf{F}^{\text{KBN,mod},n}_{i-1/2}$ is still consistent with the natural limits of the grid as $\alpha \to 1$ and



(a) Geometric parameters.

(b) Intercell flux components.

Figure 3.3: Illustration of the parameters used in the flux stabilisation process for multi-dimensional simulations.

$\alpha \to 0$. From Fig. 3.2, it is also evident that α/α_2 is just $\Delta t_{\mathrm{cc}}/\Delta t$.

The LPFS stabilised flux is therefore the following:

$$\mathbf{F}^{\mathrm{LPFS},n}_{i-1/2} = \frac{\Delta t_{\mathrm{cc}}}{\Delta t}\mathbf{F}^{n}_{i-1/2} + \left(1 - \frac{\Delta t_{\mathrm{cc}}}{\Delta t}\right)\mathbf{F}^{\mathrm{KBN,mod},n}_{i-1/2}. \tag{3.13}$$

Combining Eq. 3.11 and Eq. 3.5 gives the expression for $\Delta t_{\mathrm{cc}}/\Delta t$:

$$\frac{\Delta t_{\mathrm{cc}}}{\Delta t} = \epsilon\frac{\alpha W_{\max}}{W_i}. \tag{3.14}$$

For the non-linear Euler equations, we employ the 'wave speeds uncertainty' parameter $\epsilon \in [0, 1]$ to account for any errors arising from the use of Eq. 3.6 to estimate the cut cell wave speeds. We found setting ϵ to 0.5 to be a robust choice for the wide range of problems tackled in this chapter. For the linear advection equation, ϵ is of course set precisely to 1 and, in fact, $\Delta t_{\mathrm{cc}}/\Delta t = \alpha$.

Like the KBN flux, the LPFS flux is still only first order accurate at the boundary and for the simulations in this chapter, we do not even linearly reconstruct the solution in the cut cells. However, the numerical results indicate that the LPFS flux not only allows the computation of more accurate solutions near stagnation points as would be expected, but also alleviates the problem of oscillatory boundary solutions produced by the KBN flux at higher Courant numbers.

3.3.3 Multi-dimensional extension

The LPFS flux can be implemented for multi-dimensional simulations in the framework of Klein et al. which requires attention to be given to the irregular nature of the cut cells.

With reference to Fig. 3.3, we explain the flux calculation procedure at the interface $I_{\mathrm{C}}^{\mathrm{face}}$ between two neighbouring cells for an arbitrary dimensional sweep in a 2D simulation. The procedure is exactly the same for 3D simulations.

The division of $I_{\mathrm{C}}^{\mathrm{face}}$ into 'shielded' and 'unshielded' regions has already been introduced in Section 2.2, where we also define all the geometric parameters shown in Fig. 3.3a.

The fluxes labelled in Fig. 3.3b are calculated as follows:

– The boundary flux, for example, $\mathbf{F}_{\mathrm{L}}^{\mathrm{B},n}$, is calculated by evaluating the flux acting in the current coordinate direction for the 'boundary state' given by the solution of the wall normal Riemann problem. For static reflective boundaries, the boundary state is calculated as follows:

  1. Rotate the cut cell state $\mathbf{U}_i^n$ to produce the state $\mathbf{U}_i^{\mathrm{rot},n}$ aligned with the boundary normal-tangential frame.
  2. Invert the sign of the normal velocity component of $\mathbf{U}_i^{\mathrm{rot},n}$ to produce the 'ghost state' $\mathbf{U}_{i,\mathrm{GC}}^{\mathrm{rot},n}$.
  3. Compute the solution, $\mathbf{U}_i^{\mathrm{rot,B},n}$, to the wall normal Riemann problem defined by $\mathbf{U}_i^{\mathrm{rot},n}$ and $\mathbf{U}_{i,\mathrm{GC}}^{\mathrm{rot},n}$.
  4. Rotate $\mathbf{U}_i^{\mathrm{rot,B},n}$ to align back with the Cartesian frame to give the 'boundary state' $\mathbf{U}_i^{\mathrm{B},n}$.

  Note that in order to maintain conservation in a dimensionally split framework, the advective boundary fluxes need to be computed using 'reference' boundary states computed at the start of a time step and kept constant in between sweeps. Since the velocity resulting from the wall Riemann problem is tangential to the wall, the sum of advective fluxes across the interface accummulated over a full cycle of split steps cancels exactly, thus ensuring zero net mass flux across the wall. This restriction does not apply to other fluxes, so that for the Euler equations Eq. 3.2, for example, the boundary state pressure should be updated in each sweep. A specification of this non-obvious requirement for computing the boundary fluxes was one of the unique insights provided by Klein et al. [37].

– $\mathbf{F}^{\mathrm{US},n}$ is taken to be the standard explicit 1D flux for regular cells since it is not shielded by the boundary. Note that linear reconstruction of the solution to compute MUSCL-Hancock fluxes is carried out on the 1D strips neighbouring the unshielded part of the interface.

– The singly-shielded fluxes, $\mathbf{F}^{\mathrm{SS,L},n}$ and $\mathbf{F}^{\mathrm{SS,R},n}$, are calculated as one-dimensional LPFS stabilised fluxes as per Eq. 3.13, with the place of α in Eq. 3.14 being taken

by $\alpha_L^{\text{shielded}}$ or $\alpha_R^{\text{shielded}}$ respectively. Note that the unstabilised fluxes acting on the singly-shielded parts are computed using the first order Riemann solver with no linear reconstruction.

– In the doubly-shielded region, there is a genuine restriction on the time step imposed by the distance between the two boundaries. Here, we propose the use of a simple conservative 'mixing' flux designed to produce the same volume-averaged solution in the doubly-shielded parts of the cells over the course of the dimensional sweep:

$$\mathbf{F}^{\mathrm{DS},n} = \frac{1}{(\alpha_L^{\mathrm{DS}} + \alpha_R^{\mathrm{DS}})} \left[\frac{\alpha_L^{\mathrm{DS}} \alpha_R^{\mathrm{DS}} \Delta x}{\beta^{\mathrm{DS}} \Delta t} (\mathbf{U}_L^n - \mathbf{U}_R^n) + \alpha_L^{\mathrm{DS}} \mathbf{F}_{\mathrm{R}}^{\mathrm{B},n} + \alpha_R^{\mathrm{DS}} \mathbf{F}_{\mathrm{L}}^{\mathrm{B},n} \right]. \tag{3.15}$$

The modified flux $\mathbf{F}_{\mathrm{C}}^{\mathrm{modified},n}$ at $I_{\mathrm{C}}^{\mathrm{face}}$ is taken as an area-weighted sum of the individual components:

$$\mathbf{F}_{\mathrm{C}}^{\mathrm{modified},n} = \frac{1}{\beta^{\mathrm{C}}} \left[\beta^{\mathrm{US}} \mathbf{F}^{\mathrm{US},n} + \beta^{\mathrm{SS,L}} \mathbf{F}^{\mathrm{SS,L},n} + \beta^{\mathrm{SS,R}} \mathbf{F}^{\mathrm{SS,R},n} + \beta^{\mathrm{DS}} \mathbf{F}^{\mathrm{DS},n} \right]. \tag{3.16}$$

The multi-dimensional update formula for a cut cell of index (i, j) in 2D using Godunov splitting would be:

$$\begin{aligned}
\mathbf{U}_{i,j}^{n+\frac{1}{2}} &= \mathbf{U}_{i,j}^n + \frac{\Delta t}{\alpha_{i,j} \Delta x} \left[\beta_{i-1/2,j} \mathbf{F}_{i-1/2,j}^{\mathrm{modified},n} - \beta_{i+1/2,j} \mathbf{F}_{i+1/2,j}^{\mathrm{modified},n} - \left(\beta_{i-1/2,j} - \beta_{i+1/2,j} \right) \mathbf{F}_{i,j}^{\mathrm{B},n} \right], \\
\mathbf{U}_{i,j}^{n+1} &= \mathbf{U}_{i,j}^{n+\frac{1}{2}} + \frac{\Delta t}{\alpha_{i,j} \Delta y} \left[\beta_{i,j-1/2} \mathbf{G}_{i,j-1/2}^{\mathrm{modified},n} - \beta_{i,j+1/2} \mathbf{G}_{i,j+1/2}^{\mathrm{modified},n} - \left(\beta_{i,j-1/2} - \beta_{i,j+1/2} \right) \mathbf{G}_{i,j}^{\mathrm{B},n} \right]
\end{aligned} \tag{3.17}$$

where we use $\mathbf{F}$ and $\mathbf{G}$ to denote fluxes acting in the x and y directions respectively. As in Section 2.1, β represents the cell face fraction. $\beta_{i-1/2,j}$, for example, is the cell face fraction of the interface between cells $(i-1, j)$ and (i, j). Note also that the boundary fluxes $\mathbf{F}_{i,j}^{\mathrm{B},n}$ and $\mathbf{G}_{i,j}^{\mathrm{B},n}$ are computed using the 'reference' boundary states as explained earlier. A 3D simulation would involve one more sweep using fluxes $\mathbf{H}$ acting in the z direction.

Note that the construction principle of the method is to deviate as little from the standard update for regular cells as possible. Over the unshielded parts of cell faces, the fluxes do not call for stabilisation, only those across the shielded parts do. This way, we aim for minimised dissipative impact of the stabilisation method.

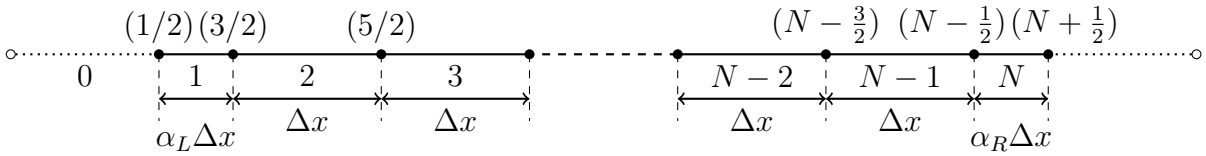


Figure 3.4: 1D mesh with cut cells located at the left and right edges of the domain.

3.3.3.1 Post-sweep correction at concavities

When $\beta^L = \beta^R = 0$, i.e., when $I_{\mathrm{C}}^{\mathrm{face}}$ is completely shielded by the boundary from both sides, we found that the mixing flux Eq. 3.15 was sometimes unable to maintain stability. A simple conservative fix for this problem is to merge the solution in such pairs of 'fully doubly-shielded' cells with that of their immediate neighbours in a volume-fraction weighted manner at the end of the sweep. This is not computationally expensive as the affected cells can be identified at the flux computation stage and directly targeted after the sweep. The performance of this strategy is demonstrated through the results of Section 3.5.4 and Section 3.5.6, which contain examples of fully doubly-shielded cells in two and three dimensions respectively.

The design of a better doubly-shielded flux $\mathbf{F}^{\mathrm{DS},n}$ which would avoid the need for this post-sweep correction is identified as an open research problem. It should be noted, however, that the correction even as it stands affects only a small number of cells. For the complex geometry test case of Section 3.5.6, for example, 'fully doubly-shielded' cells make up less than 0.05% of all cut cells in a given dimensional sweep.

3.4 Convergence and stability analysis

In this section, we demonstrate the convergence and stability of the first order LPFS scheme for the solution of the linear advection equation

$$u_t + a u_x = 0, \tag{3.18}$$

on the one-dimensional mesh consisting of N cells shown in Fig. 3.4. The boundary cells '1' and 'N' are both assumed to be cut cells with respective volume fractions α_L and α_R. For the purposes of the analysis, we can therefore assume without loss of generality that $a > 0$.

From Eq. 3.13, the stabilised LPFS flux at interface (3/2) can be worked out to be:

$$\begin{aligned} f_{3/2}^{\mathrm{LPFS},n} &= \alpha_L f_{3/2}^n + (1-\alpha_L) f_{3/2}^{\mathrm{KBN},n} \\ &= \alpha_L f_{3/2}^n + (1-\alpha_L)[f_{1/2}^n + \alpha_L (f_{3/2}^n - f_{1/2}^n)] \\ &= (\alpha_L - 1)^2 f_{1/2}^n + \alpha_L (2-\alpha_L) f_{3/2}^n. \end{aligned} \tag{3.19}$$

Similarly, the stabilised LPFS flux at interface $(N-1/2)$ is:

$$\begin{aligned} f_{N-1/2}^{\mathrm{LPFS},n} &= \alpha_R f_{N-1/2}^n + (1-\alpha_R) f_{N-1/2}^{\mathrm{KBN},n} \\ &= \alpha_R f_{N-1/2}^n + (1-\alpha_R)[f_{N+1/2}^n + \alpha_R (f_{N-1/2}^n - f_{N+1/2}^n)] \\ &= \alpha_R (2-\alpha_R) f_{N-1/2}^n + (\alpha_R - 1)^2 f_{N+1/2}^n. \end{aligned} \tag{3.20}$$

The update formulas for cells '1', '2', '$N-1$' and 'N' which are affected by the flux stabilisation are thus:

$$U_1^{n+1} = U_1^n + c(2-\alpha_L)(U_0^n - U_1^n), \tag{3.21}$$

$$U_2^{n+1} = U_2^n + c[(\alpha_L - 1)^2 U_0^n + \alpha_L (2-\alpha_L) U_1^n - U_2^n], \tag{3.22}$$

$$U_{N-1}^{n+1} = U_{N-1}^n + c[U_{N-2}^n - \alpha_R (2-\alpha_R) U_{N-1}^n - (\alpha_R - 1)^2 U_N^n], \tag{3.23}$$

$$U_N^{n+1} = U_N^n + c(2-\alpha_R)(U_{N-1}^n - U_N^n), \tag{3.24}$$

where $c = \frac{a\Delta t}{\Delta x}$ is the Courant number. Note that we use capital U_i^n to denote the discrete approximation of the volume averaged solution in cell i at time n.

3.4.1 'Supraconvergence' property of the LPFS scheme

Attempting a truncation error analysis of the scheme in any of the cells '1', '2', '$N-1$' and 'N' affected by the flux stabilisation shows an inconsistency with the governing equation Eq. 3.18. Note that in the following, we use small u_i^n to represent the grid function of the exact solution $u(x,t)$ in cell i at time n.

Consider, for example, cell N. The truncation error in that cell, Lu_N, can be found to

be:

$$\begin{aligned} Lu_N &= \frac{u_N^{n+1} - u_N^n}{\Delta t} - (2-\alpha_R)a\frac{u_{N-1}^n - u_N^n}{\Delta x} \\ &= u_t(x_N, t^n) - (2-\alpha_R)a\frac{u_N^n - \frac{(1+\alpha_R)}{2}\Delta x u_x(x_N, t^n) - u_N^n}{\Delta x} + \mathcal{O}(\Delta t, \Delta x) \\ &= -au_x(x_N, t^n) + (2-\alpha_R)a\frac{(1+\alpha_R)}{2}u_x(x_N, t^n) + \mathcal{O}(\Delta t, \Delta x) \\ &= \frac{1}{2}\alpha_R(\alpha_R - 1)au_x(x_N, t^n) + \mathcal{O}(\Delta t, \Delta x), \end{aligned}$$

so that the scheme is inconsistent unless α_R is 1. Therefore, we cannot invoke the 'Lax equivalence theorem' [41] in the analysis.

Despite this inconsistency, numerical tests show that the scheme does, in fact, converge with first order accuracy (see Section 3.5.1). To prove this 'supraconvergence' property of the method, we follow the approach of Berger et al. [42] who perform a similar analysis for their h-box method.

The aim is to find another grid function w which differs from the grid function of u by an $\mathcal{O}(\Delta x)$ amount, and for which the truncation error in all cells is $\mathcal{O}(\Delta t, \Delta x)$. This gives

$$|w_i - U_i| = \mathcal{O}(\Delta t, \Delta x) \tag{3.25}$$

for $i = 1, \ldots, N$. However, since $w = u + \mathcal{O}(\Delta x)$, this would imply that

$$|u_i - U_i| = \mathcal{O}(\Delta t, \Delta x) \tag{3.26}$$

for $i = 1, \ldots, N$, i.e. that the scheme does in fact approximate the true solution to first order despite the inconsistency in the boundary cells and their immediate neighbours.

Let the grid function w be defined as follows:

$$w_i^n = \begin{cases} u_i^n + \frac{\alpha_L(\alpha_L - 1)}{2(2-\alpha_L)}\Delta x u_x(x_i, t^n) & \text{if } i = 1, \\ u_i^n + \left(\frac{1+\alpha_R(\alpha_R-2)}{\alpha_R - 2}\right)\Delta x u_x(x_i, t^n) & \text{if } i = N-1, \\ u_i^n + \frac{1}{2}(\alpha_R - 1)\Delta x u_x(x_i, t^n) & \text{if } i = N, \\ u_i^n & \text{otherwise,} \end{cases} \tag{3.27}$$

where we note that $w = u + \mathcal{O}(\Delta x)$ as desired.

The truncation error of w in cell '1', Lw_1 can be worked out to be:

$$\begin{aligned}
Lw_1 &= \frac{w_1^{n+1} - w_1^n}{\Delta t} - (2-\alpha_L)a\frac{w_0^n - w_1^n}{\Delta x} \\
&= \frac{u_1^{n+1} + \frac{\alpha_L(\alpha_L-1)}{2(2-\alpha_L)}\Delta x u_x(x_1, t^{n+1}) - u_1^n - \frac{\alpha_L(\alpha_L-1)}{2(2-\alpha_L)}\Delta x u_x(x_1, t^n)}{\Delta t} \\
&- (2-\alpha_L)a\frac{u_0^n - u_1^n - \frac{\alpha_L(\alpha_L-1)}{2(2-\alpha_L)}\Delta x u_x(x_1, t^n)}{\Delta x} \\
&= \frac{u_1^n + \Delta t u_t(x_1, t^n) + \frac{\alpha_L(\alpha_L-1)}{2(2-\alpha_L)}\Delta x u_x(x_1, t^n) - u_1^n - \frac{\alpha_L(\alpha_L-1)}{2(2-\alpha_L)}\Delta x u_x(x_1, t^n)}{\Delta t} \\
&- (2-\alpha_L)a\frac{u_1^n - \frac{(1+\alpha_L)}{2}\Delta x u_x(x_1, t^n) - u_1^n - \frac{\alpha_L(\alpha_L-1)}{2(2-\alpha_L)}\Delta x u_x(x_1, t^n)}{\Delta x} + \mathcal{O}(\Delta t, \Delta x) \\
&= u_t(x_1, t^n) - (2-\alpha_L)a\frac{-\frac{\Delta x u_x(x_1, t^n)}{(2-\alpha_L)}}{\Delta x} + \mathcal{O}(\Delta t, \Delta x) \\
&= u_t(x_1, t^n) + a u_x(x_1, t^n) + \mathcal{O}(\Delta t, \Delta x) = \mathcal{O}(\Delta t, \Delta x).
\end{aligned}$$

Similarly, the truncation error in cell '2', Lw_2 is:

$$\begin{aligned}
Lw_2 &= \frac{w_2^{n+1} - w_2^n}{\Delta t} - a\frac{[(\alpha_L-1)^2 w_0^n + \alpha_L(2-\alpha_L)w_1^n - w_2^n]}{\Delta x} \\
&= \frac{u_2^{n+1} - u_2^n}{\Delta t} - a\frac{\left[(\alpha_L-1)^2 u_0^n + \alpha_L(2-\alpha_L)\left(u_1^n + \frac{\alpha_L(\alpha_L-1)}{2(2-\alpha_L)}\Delta x u_x(x_1, t^n)\right) - u_2^n\right]}{\Delta x} \\
&= u_t(x_2, t^n) - a\frac{[(\alpha_L-1)^2(u_2^n - (1+\alpha_L)\Delta x u_x(x_2, t^n))]}{\Delta x} \\
&- a\frac{\left[\alpha_L(2-\alpha_L)\left(u_2^n - \frac{(1+\alpha_L)}{2}\Delta x u_x(x_2, t^n) + \frac{\alpha_L(\alpha_L-1)}{2(2-\alpha_L)}\Delta x u_x(x_2, t^n)\right) - u_2^n\right]}{\Delta x} \\
&+ \mathcal{O}(\Delta t, \Delta x) \\
&= u_t(x_2, t^n) + a u_x(x_2, t^n) + \mathcal{O}(\Delta t, \Delta x) = \mathcal{O}(\Delta t, \Delta x).
\end{aligned}$$

At the right edge of the domain, the truncation error in cell '$N-1$', Lw_{N-1} is:

$$
\begin{aligned}
Lw_{N-1} &= \frac{w_{N-1}^{n+1}-w_{N-1}^{n}}{\Delta t} - a\frac{[w_{N-2}^{n}-\alpha_R(2-\alpha_L)w_{N-1}^{n}-(\alpha_R-1)^2 w_N^n]}{\Delta x} \\
&= \frac{u_{N-1}^{n+1}+\left(\frac{1+\alpha_R(\alpha_R-2)}{\alpha_R-2}\right)\Delta x u_x(x_{N-1},t^{n+1}) - u_{N-1}^{n} - \left(\frac{1+\alpha_R(\alpha_R-2)}{\alpha_R-2}\right)\Delta x u_x(x_{N-1},t^{n})}{\Delta t} \\
&- a\frac{\left[u_{N-2}^{n}-\alpha_R(2-\alpha_L)\left(u_{N-1}^{n}+\left(\frac{1+\alpha_R(\alpha_R-2)}{\alpha_R-2}\right)\Delta x u_x(x_{N-1},t^{n})\right)\right]}{\Delta x} \\
&- a\frac{\left[-(\alpha_R-1)^2\left(u_N^n+\frac{1}{2}(\alpha_R-1)\Delta x u_x(x_N,t^n)\right)\right]}{\Delta x} \\
&= \frac{u_{N-1}^{n+1}+\left(\frac{1+\alpha_R(\alpha_R-2)}{\alpha_R-2}\right)\Delta x u_x(x_{N-1},t^{n}) - u_{N-1}^{n} - \left(\frac{1+\alpha_R(\alpha_R-2)}{\alpha_R-2}\right)\Delta x u_x(x_{N-1},t^{n})}{\Delta t} \\
&- a\frac{\left[u_{N-1}^{n}-\Delta x u_x(x_{N-1},t^n)-\alpha_R(2-\alpha_L)\left(u_{N-1}^{n}+\left(\frac{1+\alpha_R(\alpha_R-2)}{\alpha_R-2}\right)\Delta x u_x(x_{N-1},t^{n})\right)\right]}{\Delta x} \\
&- a\frac{\left[-(\alpha_R-1)^2\left(u_{N-1}^{n}+\frac{(1+\alpha_R)}{2}\Delta x u_x(x_{N-1},t^n)+\frac{1}{2}(\alpha_R-1)\Delta x u_x(x_{N-1},t^n)\right)\right]}{\Delta x} \\
&+ \mathcal{O}(\Delta t,\Delta x) \\
&= u_t(x_{N-1},t^n) + a u_x(x_{N-1},t^n) + \mathcal{O}(\Delta t,\Delta x) = \mathcal{O}(\Delta t,\Delta x).
\end{aligned}
$$

Similarly, the the truncation error in cell 'N', Lw_N is:

$$
\begin{aligned}
Lw_N &= \frac{w_N^{n+1}-w_N^n}{\Delta t} - (2-\alpha_R)a\frac{w_{N-1}^n - w_N^n}{\Delta x} \\
&= \frac{u_N^{n+1}+\frac{1}{2}(\alpha_R-1)\Delta x u_x(x_N,t^{n+1}) - u_N^n - \frac{1}{2}(\alpha_R-1)\Delta x u_x(x_N,t^n)}{\Delta t} \\
&- (2-\alpha_R)a\frac{u_{N-1}^n+\left(\frac{1+\alpha_R(\alpha_R-2)}{\alpha_R-2}\right)\Delta x u_x(x_{N-1},t^n) - u_N^n - \frac{1}{2}(\alpha_R-1)\Delta x u_x(x_N,t^n)}{\Delta x} \\
&= \frac{u_N^{n+1}+\frac{1}{2}(\alpha_R-1)\Delta x u_x(x_N,t^{n}) - u_N^n - \frac{1}{2}(\alpha_R-1)\Delta x u_x(x_N,t^n)}{\Delta t} \\
&- (2-\alpha_R)a\frac{u_N^n-\frac{(1+\alpha_R)}{2}\Delta x u_x(x_N,t^n)+\left(\frac{1+\alpha_R(\alpha_R-2)}{\alpha_R-2}\right)\Delta x u_x(x_N,t^n)}{\Delta x} \\
&- (2-\alpha_R)a\frac{-u_N^n-\frac{1}{2}(\alpha_R-1)\Delta x u_x(x_N,t^n)}{\Delta x} + \mathcal{O}(\Delta t,\Delta x) \\
&= u_t(x_N,t^n) - (2-\alpha_R)a\frac{-\frac{\Delta x u_x(x_N,t^n)}{(2-\alpha_R)}}{\Delta x} + \mathcal{O}(\Delta t,\Delta x) \\
&= u_t(x_N,t^n) + a u_x(x_N,t^n) + \mathcal{O}(\Delta t,\Delta x) = \mathcal{O}(\Delta t,\Delta x).
\end{aligned}
$$

For all other cells which are updated with the regular upwind formula, since we define w_i^n to be equal to u_i^n, the truncation error is again $\mathcal{O}(\Delta t, \Delta x)$ as desired, which concludes the proof.

3.4.2 Stability of the LPFS scheme

In the previous Section 3.4.1, we have shown that the computed U_i^n approximate the w_i^n (and hence, the u_i^n) to first order assuming that the scheme is stable. In this section, we prove that the U_i^n approach the w_i^n in a stable manner for $c \in (0, 1]$ for all $i = 1, \ldots, N$.

Consider the error function $v_i^n = U_i^n - w_i^n$. Given some sufficiently smooth initial conditions, v_i^0 is $\mathcal{O}(\Delta t, \Delta x)$ for all cells. For the cells unaffected by the flux stabilisation, stability is already guaranteed for $c \in (0, 1]$ since they are updated using the regular upwind formula. v_i^n in those cells continues to remain first order as n increases.

As before, we therefore need only focus our attention on cells '1', '2', '$N-1$' and 'N' and investigate how v_i^n evolves for those cells as $n \to \infty$. Recall from the supraconvergence analysis of Section 3.4.1 that:

$$w_1^{n+1} = w_1^n + c(2 - \alpha_L)(w_0^n - w_1^n) + \mathcal{O}(\Delta t, \Delta x), \tag{3.28}$$

$$w_2^{n+1} = w_2^n + c[(\alpha_L - 1)^2 w_0^n + \alpha_L(2 - \alpha_L)w_1^n - w_2^n] + \mathcal{O}(\Delta t, \Delta x), \tag{3.29}$$

$$w_{N-1}^{n+1} = w_{N-1}^n + c[w_{N-2}^n - \alpha_R(2 - \alpha_R)w_{N-1}^n - (\alpha_R - 1)^2 w_N^n] + \mathcal{O}(\Delta t, \Delta x), \tag{3.30}$$

$$w_N^{n+1} = w_N^n + c(2 - \alpha_R)(w_{N-1}^n - w_N^n) + \mathcal{O}(\Delta t, \Delta x). \tag{3.31}$$

Using Eq. 3.21-Eq. 3.24 and Eq. 3.28-Eq. 3.31, it is straightforward to work out v_1^n, v_2^n, v_{N-1}^n and v_N^n. At the left edge of the domain,

$$\begin{aligned} v_1^{n+1} &= v_1^n + c(2 - \alpha_L)(v_0^n - v_1^n) + \mathcal{O}(\Delta t, \Delta x), \\ &= v_1^n + c(2 - \alpha_L)(-v_1^n) + \mathcal{O}(\Delta t, \Delta x), \end{aligned} \tag{3.32}$$

since $v_0^n = \mathcal{O}(\Delta t, \Delta x)$ by definition (Eq. 3.27). Similarly,

$$\begin{aligned} v_2^{n+1} &= v_2^n + c[(\alpha_L - 1)^2 v_0^n + \alpha_L(2 - \alpha_L)v_1^n - v_2^n] + \mathcal{O}(\Delta t, \Delta x), \\ &= v_2^n + c[\alpha_L(2 - \alpha_L)v_1^n - v_2^n] + \mathcal{O}(\Delta t, \Delta x). \end{aligned} \tag{3.33}$$

At the right edge of the domain,

$$\begin{aligned} v_{N-1}^{n+1} &= v_{N-1}^n + c[v_{N-2}^n - \alpha_R(2-\alpha_R)v_{N-1}^n - (\alpha_R-1)^2 v_N^n] + \mathcal{O}(\Delta t, \Delta x), \\ &= v_{N-1}^n + c[-\alpha_R(2-\alpha_R)v_{N-1}^n - (\alpha_R-1)^2 v_N^n] + \mathcal{O}(\Delta t, \Delta x), \end{aligned} \tag{3.34}$$

and

$$v_N^{n+1} = v_N^n + c(2-\alpha_R)(v_{N-1}^n - v_N^n) + \mathcal{O}(\Delta t, \Delta x). \tag{3.35}$$

We can now use Eq. 3.32, Eq. 3.33, Eq. 3.34 and Eq. 3.35 to write the following linear inhomogeneous recurrence relation for $\mathbf{x}^{n+1} = [v_1^{n+1}\ \ v_2^{n+1}\ \ v_{N-1}^{n+1}\ \ v_N^{n+1}]^T$:

$$\underbrace{\begin{bmatrix} v_1^{n+1} \\ v_2^{n+1} \\ v_{N-1}^{n+1} \\ v_N^{n+1} \end{bmatrix}}_{\mathbf{x}^{n+1}} = \underbrace{\begin{bmatrix} 1-c(2-\alpha_L) & 0 & 0 & 0 \\ c\alpha_L(2-\alpha_L) & 1-c & 0 & 0 \\ 0 & 0 & 1-c\alpha_R(2-\alpha_R) & -c(\alpha_R-1)^2 \\ 0 & 0 & c(2-\alpha_R) & 1-c(2-\alpha_R) \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} v_1^n \\ v_2^n \\ v_{N-1}^n \\ v_N^n \end{bmatrix}}_{\mathbf{x}^n} + \underbrace{\begin{bmatrix} \mathcal{O}(\Delta t, \Delta x) \\ \mathcal{O}(\Delta t, \Delta x) \\ \mathcal{O}(\Delta t, \Delta x) \\ \mathcal{O}(\Delta t, \Delta x) \end{bmatrix}}_{\mathbf{b}^n}.$$

We therefore have

$$\begin{aligned} \mathbf{x}^{n+1} &= A\mathbf{x}^n + \mathbf{b}^n \\ &= A(A\mathbf{x}^{n-1} + \mathbf{b}^{n-1}) + \mathbf{b}^n = A^2\mathbf{x}^{n-1} + A\mathbf{b}^{n-1} + \mathbf{b}^n \\ &= A^3\mathbf{x}^{n-2} + A^2\mathbf{b}^{n-2} + A\mathbf{b}^{n-1} + \mathbf{b}^n \\ &= A^{n+1}\mathbf{x}^0 + \sum_{\nu=0}^{n} A^\nu \mathbf{b}^{n-\nu}, \end{aligned}$$

where we note that $\mathbf{x}^0$ and all the $\mathbf{b}^{n-\nu}$ vectors are made up of $\mathcal{O}(\Delta t, \Delta x)$ components.

Hence,

$$|\mathbf{x}^{n+1}| \leq \|A\|^{n+1}|\mathbf{x}^0| + \max_{k=0,\ldots,n}\{|\mathbf{b}^k|\}\sum_{\nu=0}^{n}\|A\|^{\nu}.$$

In order for $|\mathbf{x}^{n+1}|$ to remain a bounded $\mathcal{O}(\Delta t, \Delta x)$ term, it is clear that we require $\|A\| < 1$ for all $c \in (0, 1]$ and $\alpha_L, \alpha_R \in (0, 1]$. For the first term to the right hand side of the inequality, this would imply that $\|A\|^{n+1}|\mathbf{x}^0| < |\mathbf{x}^0|$. The second term on the right hand side, on the other hand, can be thought of as a geometric series with the 'common ratio' of two consecutive terms being $\|A\|$. $\|A\| < 1$ ensures that the series converges to a finite sum as $n \to \infty$.

To study the behaviour of $\|A\|$, we make use of the convenient result that the spectral radius of A, $\rho(A)$, is the infimum of $\|A\|$, as $\|\cdot\|$ ranges over the set of matrix norms [43]. If we can confirm that the eigenvalues of A have magnitude less than 1 for all $c \in (0, 1]$ and $\alpha_L, \alpha_R \in (0, 1]$, we would have proved the stability of the recursion.

Since A is a block diagonal matrix, its eigenvalues will be the union of the eigenvalues of the following 2×2 matrices

$$A_L = \begin{bmatrix} 1 - c(2 - \alpha_L) & 0 \\ c\alpha_L(2 - \alpha_L) & 1 - c \end{bmatrix}, \; A_R = \begin{bmatrix} 1 - c\alpha_R(2 - \alpha_R) & -c(\alpha_R - 1)^2 \\ c(2 - \alpha_R) & 1 - c(2 - \alpha_R) \end{bmatrix}, \tag{3.36}$$

which are positioned along its main diagonal.

The eigenvalues of A_L are

$$\lambda_L^1 = 1 - c,$$
$$\lambda_L^2 = 1 - c(2 - \alpha_L).$$

It is easy to verify that

1. λ_L^1 and λ_L^2 are real,
2. $\lambda_L^1 \in [0, 1)$ for $c \in (0, 1]$, and
3. $|\lambda_L^2| < 1$ for $c \in (0, 1]$ and $\alpha_L \in (0, 1]$,

so that the eigenvalues of A_L have magnitude less than 1 over the full range of c and α_L as desired.

The eigenvalues of A_R, on the other hand, are

$$\lambda_R^1 = \frac{1}{2}\left[2 - 2c - \alpha_R c + \alpha_R^2 c - (\alpha_R - 1)\left(\sqrt{-4 + \alpha_R^2}\right)c\right],$$
$$\lambda_R^2 = \frac{1}{2}\left[2 - 2c - \alpha_R c + \alpha_R^2 c + (\alpha_R - 1)\left(\sqrt{-4 + \alpha_R^2}\right)c\right].$$

Since the $\sqrt{-4 + \alpha_R^2}$ term is imaginary for all $\alpha_R \in (0, 1]$, it is clear that λ_R^1 and λ_R^2 are complex conjugates. The square of the magnitude of any one of the eigenvalues, $|\lambda_R|^2$, is then

$$|\lambda_R|^2 = \lambda_R^1 \lambda_R^2 = \det(A_R) > 0. \tag{3.37}$$

To ensure that $|\lambda_R| < 1$, we therefore only need to check that $\det(A_R) < 1$. We have

$$\det(A_R) = 1 - [2c(1 - c) + \alpha_R c(1 - \alpha_R + c)], \tag{3.38}$$

where it can be noted straightaway that $\det(A_R) < 1$ since $2c(1 - c) > 0$ and $\alpha_R c(1 - \alpha_R + c) > 0$ for $c, \alpha_R \in (0, 1]$.

This concludes the proof[1]. We can therefore confirm that $\|A\| < 1$ for the full range of c, α_L and α_R and that the U_i^n stably approximate the w_i^n (and hence, the u_i^n) to first order in all cells as desired. Fig. 3.5 shows a contour plot of the spectral radius of A for illustrative purposes.

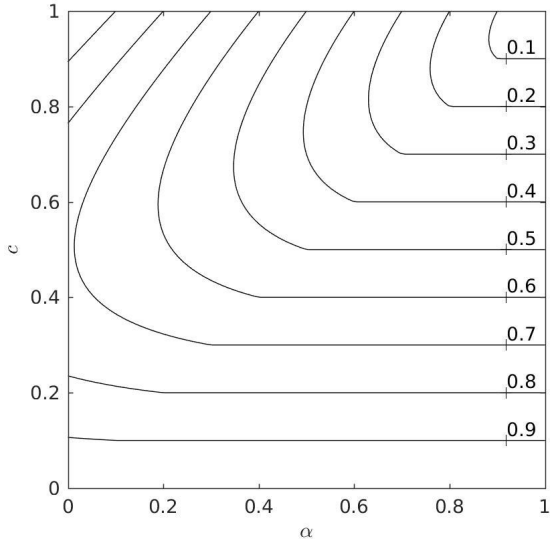


Figure 3.5: Contour plot of the spectral radius of A.

[1]Note that it was Rupert Klein, one of the co-authors of the paper [1], who proposed the idea of analysing stability by looking at the growth of the grid error function v_i^n. The present writer carried out the work of proving the stability of the recursion.

3.5 Results

In this section, we present the results of computations for a number of multi-dimensional test problems for the linear advection and Euler equations. All simulations were run at $C_{\text{cfl}} = 0.8$.

3.5.1 Convergence tests

The numerical order of convergence of LPFS is investigated through a series of advection tests. Note that for these smooth test problems, we do not use the limiter.

The L_p norm of the error for a variable ϕ at a resolution Δx is computed as

$$L_p^{\Delta x} = \left(\frac{1}{N} \sum_{i=1}^{N} (|\phi_i^{\text{sim}} - \phi_i^{\text{exact}}|)^p \right)^{\frac{1}{p}}, \tag{3.39}$$

where N is the total number of cells, ϕ_i^{sim} is the numerical solution in cell i, and ϕ_i^{exact} is the exact solution evaluated at the volumetric centroid of cell i. We note, however, that on a cut cell mesh, a volume-fraction weighted formula such as

$$L_{p,\text{weighted}}^{\Delta x} = \left(\frac{1}{V} \sum_{i=1}^{N} (\alpha_i |\phi_i^{\text{sim}} - \phi_i^{\text{exact}}|)^p \right)^{\frac{1}{p}}, \tag{3.40}$$

where V is the total volume of the fluid part of the domain, and α_i is the volume fraction of cell i, may produce more accurate estimates of the convergence order for the method. For the simulations in this work, however, all L_p norms are calculated using Eq. 3.39 as in, for example, Meyer et al. [44]. The numerical order of convergence is estimated as

$$L_p \text{ order} = \frac{\log\left(\frac{L_p^{2\Delta x}}{L_p^{\Delta x}}\right)}{\log(2)}. \tag{3.41}$$

With first order accuracy at the boundary but second order accuracy in regular cells, we show briefly how the computed order depends on the value of p. Consider that the total number of cells in the domain scales as $\mathcal{O}(n^D)$, where n is the number of cells along one dimension, and D is the number of dimensions, while the number of cut cells scales as

$\mathcal{O}(n^{D-1})$. Further, n scales as $\mathcal{O}(\Delta x^{-1})$. Substituting these values in Eq. 3.39 gives:

$$\begin{aligned} L_p^{\Delta x} &= \left(\frac{\mathcal{O}(\Delta x^p n^{D-1}) + \mathcal{O}(\Delta x^{2p})[\mathcal{O}(n^D) - \mathcal{O}(n^{D-1})]}{\mathcal{O}(n^D)} \right)^{\frac{1}{p}} \\ &= \left(\mathcal{O}(\Delta x^{p+1}) + \mathcal{O}(\Delta x^{2p}) - \mathcal{O}(\Delta x^{2p+1}) \right)^{\frac{1}{p}} \\ &= \mathcal{O}(\Delta x^{\frac{p+1}{p}}). \end{aligned} \tag{3.42}$$

Hence, we would expect the L_1 norm to converge as $\mathcal{O}(\Delta x^2)$, the L_2 norm to converge as $\mathcal{O}(\Delta x^{1.5})$ and the L_∞ norm to converge as $\mathcal{O}(\Delta x)$.

3.5.1.1 One-dimensional advection

This problem involves the linear advection of the smooth profile,

$$u(x) = \sin(2\pi x), \tag{3.43}$$

to the right at a speed $a = 1.0$ in the interval $x \in [0, 1]$ with periodic boundary conditions. The domain edge cells are made to be cut cells with volume fraction $\alpha = 10^{-3}$. The simulation is run for 1 period.

Fig. 3.6 shows the results for a resolution of 50 cells. The errors for various resolutions are shown in Table 3.1, where it may be seen that all the norms converge with the expected rates. Note that the L_∞ norm is the same as the maximum cut cell error.

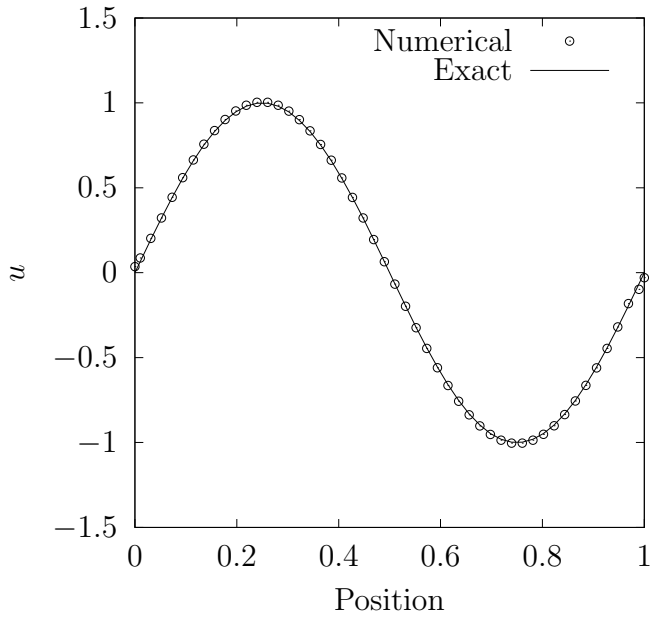


Figure 3.6: Comparison of numerical and exact solutions for the one-dimensional advection problem run with a resolution of 50 cells.

Table 3.1: Error norms and orders of convergence for the one-dimensional advection problem. The L_∞ norm is the same as the maximum cut cell error.

Resolution	L_1 norm	L_1 order	L_2 norm	L_2 order	L_∞ norm	L_∞ order
50	6.33×10^{-3}	-	9.72×10^{-3}	-	3.56×10^{-2}	-
100	1.55×10^{-3}	2.03	3.06×10^{-3}	1.67	1.85×10^{-2}	0.95
200	3.94×10^{-4}	1.98	1.07×10^{-3}	1.51	1.00×10^{-2}	0.88
400	1.00×10^{-4}	1.97	3.82×10^{-4}	1.49	5.29×10^{-3}	0.92

3.5.1.2 Two-dimensional diagonal advection

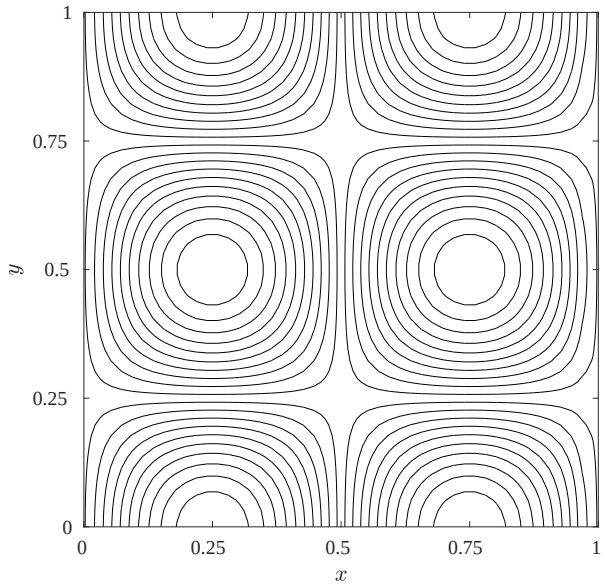
This problem involves the linear advection of the smooth function,

$$u(x, y) = \sin(2\pi x)\cos(2\pi y), \tag{3.44}$$

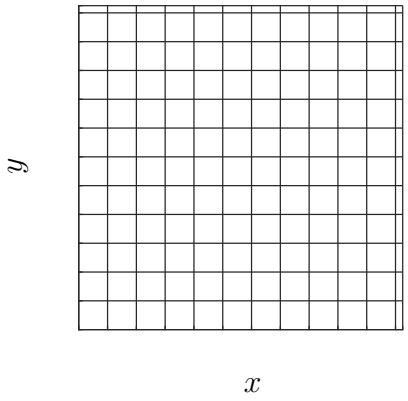
with a propagation velocity $\mathbf{a} = [1.0 \quad 1.0]^T$ in the interval $x \in [0, 1], y \in [0, 1]$. The boundary conditions are periodic and the domain edge cells are all made to be cut cells with volume fraction $\alpha = 10^{-3}$. As shown in Fig. 3.7b, which is an illustration of the top-right portion of the mesh, the 4 cut cells at the corners of the domain have a volume fraction of $\alpha^2 = 10^{-6}$.

The simulation is run for 1 period. Fig. 3.7a shows the numerical contours produced from a 50×50 cells simulation. Table 3.2 shows the errors, where it may be seen that the norms all converge with the expected rates. Note that the L_∞ norm is the same as the maximum cut cell error.

To ensure that the measured convergence rate at the cut cells is not influenced favourably by the use of second order accurate fluxes at uncut cells, we ran the simulations again, this time using first order fluxes (i.e. without linear reconstruction) to compute the fluxes for all cells. Table 3.3 shows the errors for these simulations. The convergence of the global L_1 error verifies the use of the first order solver in the entire domain. The convergence of the cut cells L_1 and L_∞ errors confirm that the solution in the cut cells converges with first order accuracy.



(a) Numerical contours after 1 period from the 50×50 cells simulation.



(b) Close-up of the top-right part of the mesh. The boundary cut cells are illustrated with an exaggerated volume fraction.

Figure 3.7: Plot of the numerical solution and an illustration of the cut cell mesh for the two-dimensional diagonal advection problem.

Table 3.2: Error norms and orders of convergence for the two-dimensional diagonal advection problem. The L_∞ norm is the same as the maximum cut cell error.

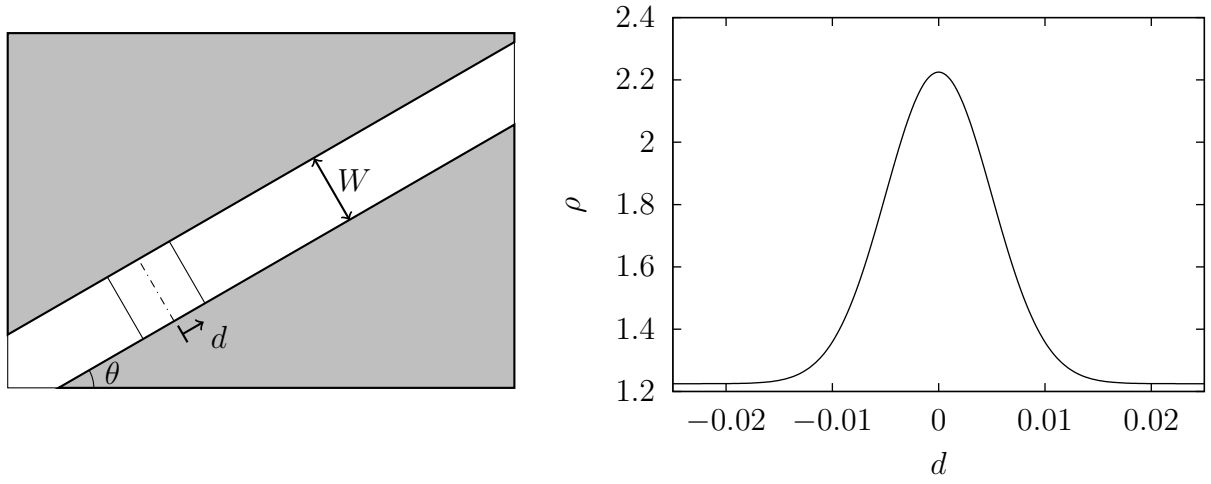
Resolution	L_1 norm	L_1 order	L_2 norm	L_2 order	L_∞ norm	L_∞ order
50×50	6.44×10^{-3}	-	8.75×10^{-3}	-	3.60×10^{-2}	-
100×100	1.56×10^{-3}	2.04	2.51×10^{-3}	1.80	1.85×10^{-2}	0.96
200×200	3.92×10^{-4}	1.99	8.21×10^{-4}	1.61	1.00×10^{-2}	0.88
400×400	9.88×10^{-5}	1.99	2.81×10^{-4}	1.55	5.29×10^{-3}	0.92

Table 3.3: Error norms and orders of convergence for the two-dimensional diagonal advection problem, in which the fluxes for all cells are computed to first order without linear reconstruction.

Resolution	Global L_1 norm	Global L_1 order	Cut cells L_1 norm	Cut cells L_1 order	Cut cells L_∞ norm	Cut cells L_∞ order
50×50	5.56×10^{-2}	-	5.08×10^{-2}	-	1.34×10^{-1}	-
100×100	3.01×10^{-2}	0.88	2.79×10^{-2}	0.87	7.36×10^{-2}	0.87
200×200	1.55×10^{-2}	0.96	1.45×10^{-2}	0.94	3.81×10^{-2}	0.95
400×400	7.88×10^{-3}	0.98	7.39×10^{-3}	0.97	1.94×10^{-2}	0.98

3.5.1.3 Two-dimensional advection in a sloped channel

This test involves solving the full Euler system for the parallel advection of a Gaussian density profile in a sloped channel. This is a non-trivial problem for the split LPFS scheme since any errors in the boundary and cut cell flux computations would affect the ability of the method to maintain the parallel flow.



(a) Simulation set-up

(b) Advection profile

Figure 3.8: Illustration of the simulation set-up and the density profile being advected for the two-dimensional advection in a sloped channel problem.

As illustrated in Fig. 3.8a, the channel with a width $W = 0.0141$ m makes an angle $\theta = 30°$ with the x axis. The pressure and velocity are 101325 Pa and 30 m/s parallel to the channel wall respectively. The density profile is shown in Fig. 3.8b and described by the following equation:

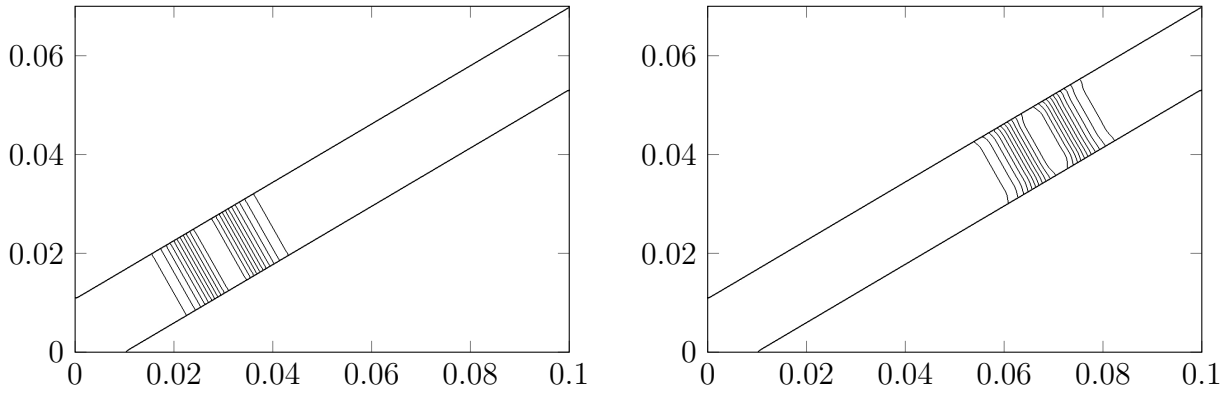
$$\rho(d) = \rho_0 + e^{-\left(\frac{d}{0.5W}\right)^2}, \tag{3.45}$$

where $\rho_0 = 1.225\,\mathrm{kg/m^3}$ and d is distance measured from the centre of the profile in the direction parallel to the channel wall.

The domain size is $[0.0, 0.1]$ m $\times$ $[0.0, 0.07]$ m. At $t = 0$ s, we position the centre of the profile at the point $[0.035\cos(\theta),\ 0.035\sin(\theta)]$ m. The simulation is run till $t = 0.0015$ s, by which time, as seen from Fig. 3.9, the profile has traversed a large portion of the channel and encountered a range of boundary cut cells.

Table 3.4: Error norms and experimental orders of convergence for the two-dimensional advection in a sloped channel problem. The L_∞ norm is the same as the maximum cut cell error.

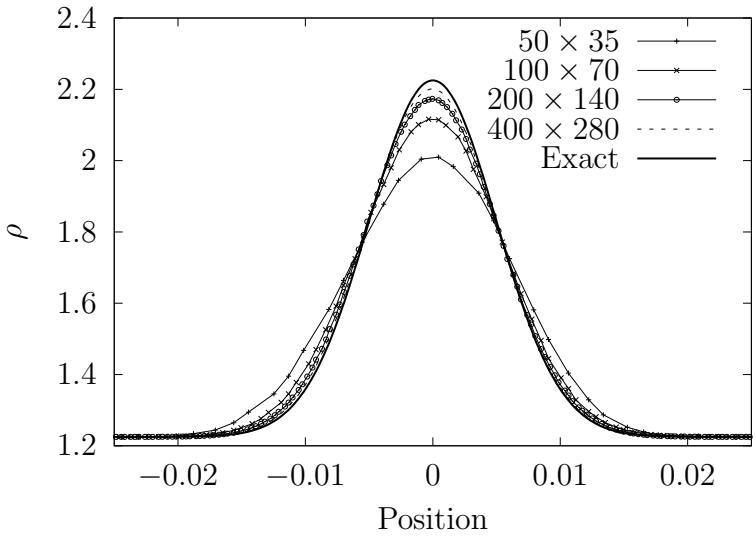
Resolution	L_1 norm	L_1 order	L_2 norm	L_2 order	L_∞ norm	L_∞ order
50×35	1.97×10^{-2}	-	4.59×10^{-2}	-	2.63×10^{-1}	-
100×70	6.19×10^{-3}	1.67	1.69×10^{-2}	1.44	1.32×10^{-1}	0.99
200×140	1.72×10^{-3}	1.84	5.75×10^{-3}	1.56	6.09×10^{-2}	1.12
400×280	4.67×10^{-4}	1.89	1.98×10^{-3}	1.54	2.92×10^{-2}	1.06



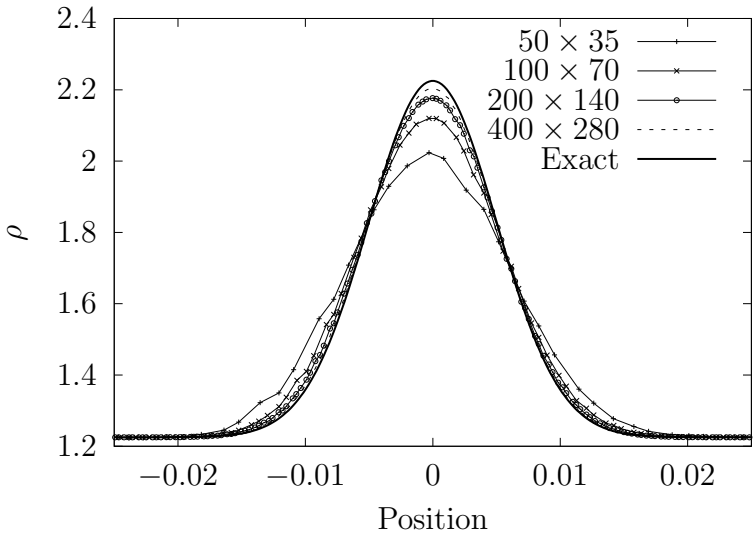
(a) $t = 0.0$ s (b) $t = 0.0015$ s

Figure 3.9: Density contours at the start and end of the simulation for the two-dimensional advection in a sloped channel problem run at a resolution of 200×140 cells.

Table 3.4 shows the errors computed for simulations at various resolutions. With increasing resolution, it may be observed that the norms converge with the expected rates. Note that the L_∞ norm is the same as the maximum cut cell error and that it converges with first order as expected. Fig. 3.10 shows the final numerical solution along the lower and upper cut cell boundaries. The convergence of the results towards the exact solution with increasing resolution is readily observed.



(a) Lower boundary.



(b) Upper boundary.

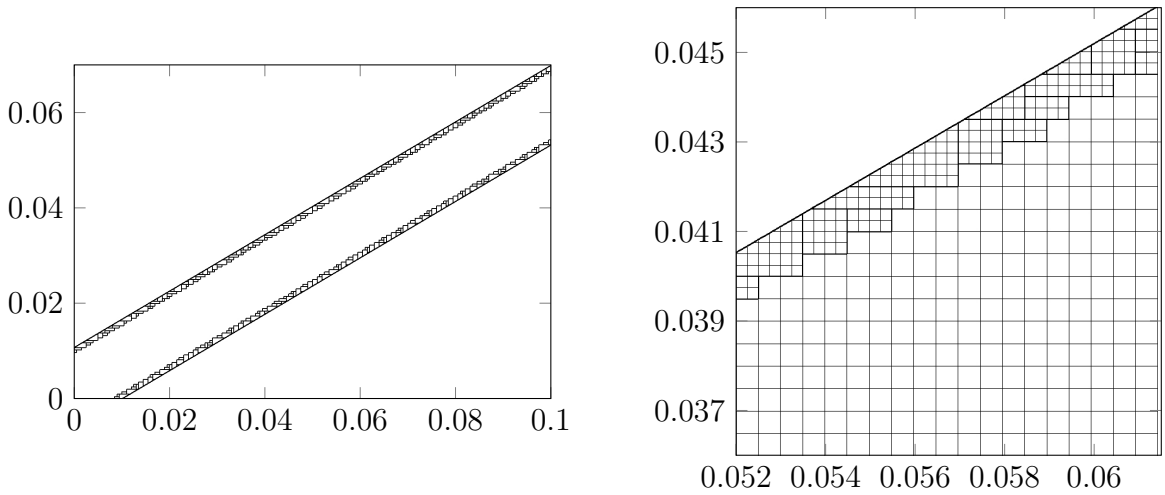
Figure 3.10: Comparison of the numerical solutions at various resolutions with the exact solution along the cut cell boundaries for the two-dimensional advection in a sloped channel problem.

It is also useful to note that a judicious use of AMR can be used to alleviate the effect of reduced order of accuracy at the boundary. Table 3.5 shows the errors for a series of closely related simulations. The first row shows the error norms for a 100×70 cells simulation with no AMR. Using these results as a reference, the second row shows the reduced norms that would be theoretically expected from a 200×140 cells simulation if we had a universally second order method. The errors obtained in practice with the LPFS

Table 3.5: Data illustrating the use of AMR to alleviate the effect of the cut cell method being first order at the boundaries.

Resolution	L_1 norm	L_2 norm	L_∞ norm
100×70 (no AMR)	6.19×10^{-3}	1.69×10^{-2}	1.32×10^{-1}
200×140 (second order, expected)	1.55×10^{-3}	4.23×10^{-3}	3.31×10^{-2}
200×140 (no AMR)	1.72×10^{-3}	5.75×10^{-3}	6.09×10^{-2}
200×140 (with cut cells refinement)	1.31×10^{-3}	3.62×10^{-3}	2.93×10^{-2}

method are clearly larger, as shown in the third row. The fourth row shows the norms from a 200×140 cells simulation where one level of AMR of refinement factor 2 is used to refine the cut cells, as shown in Fig. 3.11. When compared to the theoretical results of the second row, the error norms for this set-up are in fact lower. Of course, we do not lose sight of the fact that the cut cells from the AMR simulation are at an 'effective' resolution of 400×280 cells. The intention here is merely to illustrate how AMR can be used to alleviate the effect of the method being first order at the boundary.



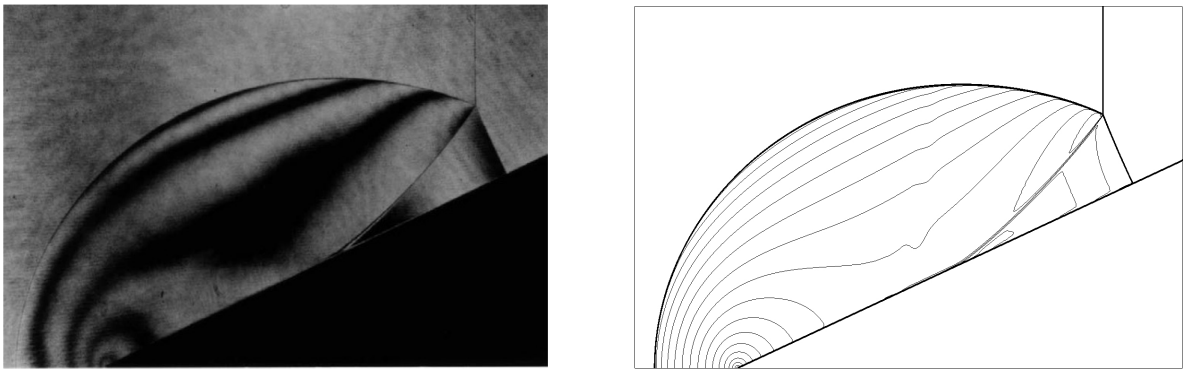
(a) View of entire domain with AMR fine patch boundaries highlighted.

(b) Zoomed in view of AMR mesh.

Figure 3.11: Plots showing the use of AMR to refine around cut cells for the two-dimensional advection in a sloped channel problem. The base resolution is 200×140 cells.

3.5.2 Shock reflection from a wedge

The $M = 1.7$ shock reflection off a 30° wedge test problem from Toro [9] is used to demonstrate the improved performance at the boundary of the LPFS flux compared to



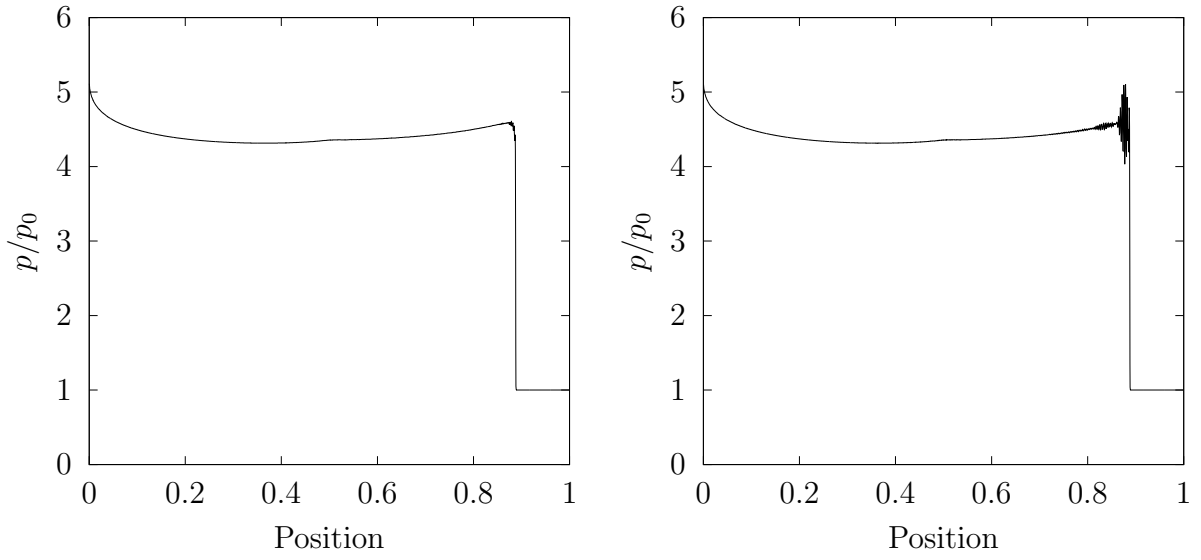
(a) Experimental shadowgraph [9]. (b) Numerical density contours.

Figure 3.12: Comparison of experimental and LPFS simulation results for the shock reflection from wedge problem.

the KBN flux. The ambient state ahead of the shock has a density and pressure of 1.225 kg/m^3 and 101325 Pa respectively. The domain size was $[0.0, 16.5]$ m $\times$ $[0.0, 25.0]$ m. A base resolution of 500×330 cells was used and two levels of AMR refinement of factor 2 each were employed to resolve the shocks and slip line. The boundary conditions were transmissive at the left, right and top boundaries, and reflective at the bottom boundary.

Fig. 3.12 shows a comparison of the experimental and numerical results. The simulation captures all the expected features. The incident shock, reflected shock, and Mach stem (which is perpendicular to the wedge) meet at the 'triple point'. The slip line connecting the triple point to the wedge is also resolved.

Fig. 3.13 shows a comparison of the surface pressure distribution computed using the LPFS and KBN fluxes. The KBN solution behind the Mach stem is highly oscillatory, and the issue is greatly alleviated in the LPFS solution. Reducing the Courant number to say, 0.5, would improve the KBN solution, however the intention of this test is to demonstrate the superior accuracy and robustness of LPFS at higher Courant numbers.



(a) LPFS. (b) KBN.

Figure 3.13: Comparison of the cut cell pressure results for LPFS and KBN for the shock reflection from wedge problem. The pressure is plotted against the normalised distance along the length of the wedge. $p_0 = 101325$ Pa.

3.5.3 Subsonic flow over a NACA 0012 aerofoil

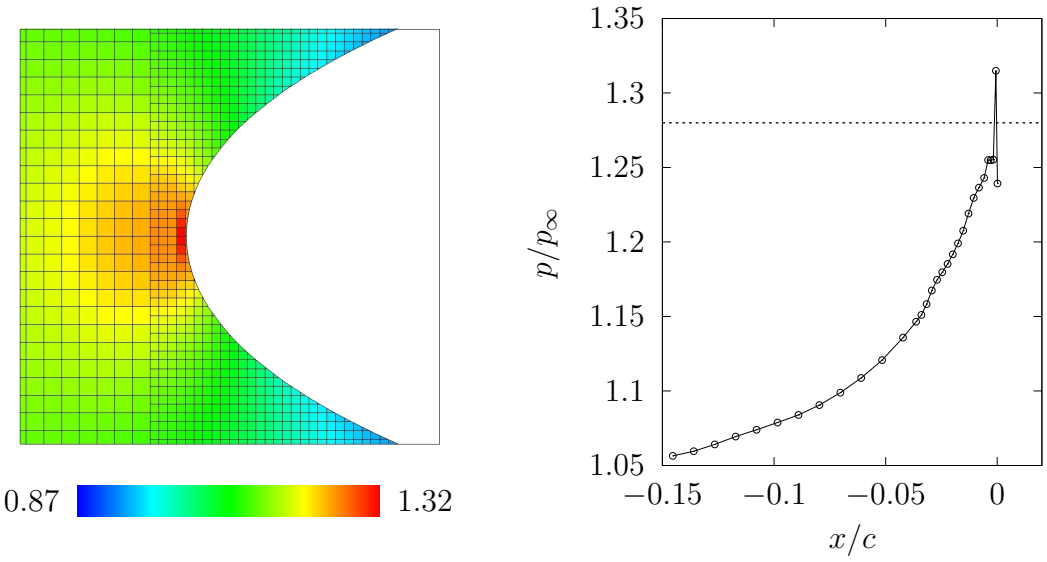
The computation of $M = 0.6$ subsonic flow over a symmetric NACA 0012 aerofoil with 0 angle of attack is used to demonstrate the increased accuracy of LPFS compared to KBN near stagnation points.

The aerofoil with a chord length $c = 0.127$ m is placed at approximately the centre of a domain having a length and height of $30c$. A coarse base resolution of 200×200 cells is used to help accelerate convergence to steady state by diffusing waves bouncing off the domain edge boundaries. The finest resolution simulation employed four AMR levels with refinement factors of 4, 4, 4 and 2. The free stream density ρ_∞ and pressure p_∞ are 1.225 kg/m^3 and 101325 Pa respectively. Subsonic inflow boundary conditions are used at the left boundary with outflow conditions specified at the right, top and bottom boundaries respectively.

Fig. 3.14a is a pseudocolour plot of the pressure ratio p/p_∞ near the stagnation point for the KBN simulation. Fig. 3.14b shows the pressure ratio along the stagnation streamline in the vicinity of the nose of the aerofoil. The dashed line corresponds to the theoretical stagnation pressure ratio calculated from the isentropic flow relations [45]. Clearly, the pressure computed with the KBN flux is too low in the stagnation cut cell

and too high in the regular neighbouring cell.

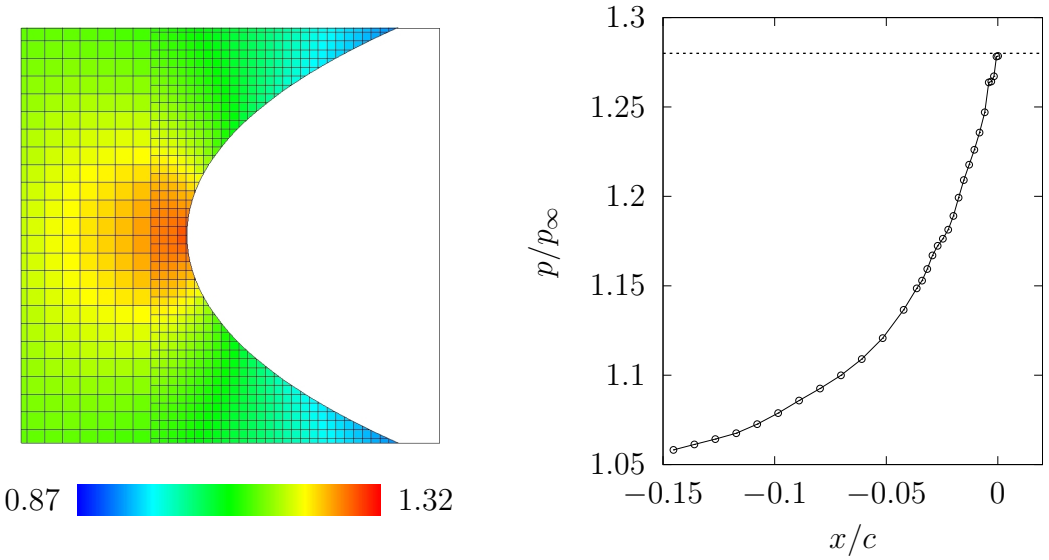
Fig. 3.15 shows the corresponding results computed with the LPFS flux. The solution looks visibly improved in the pseudocolour plot of Fig. 3.15a, where the limits for the colour bar have been set to be the same as those in Fig. 3.14a. Fig. 3.15b confirms that the LPFS stagnation solution shows good agreement with the analytical solution. These results confirm that the introduction of local wave speeds in the LPFS flux stabilisation produces the intended effect.



(a) Pseudocolour plot of p/p_∞.

(b) p/p_∞ results along the stagnation streamline in the vicinity of the aerofoil. The dashed line shows the analytical stagnation pressure ratio.

Figure 3.14: Pressure results with the KBN flux in the vicinity of the stagnation region for the subsonic NACA 0012 problem.



(a) Pseudocolour plot of p/p_∞.

(b) p/p_∞ results along the stagnation streamline in the vicinity of the aerofoil. The dashed line shows the analytical stagnation pressure ratio.

Figure 3.15: Pressure results with the LPFS flux in the vicinity of the stagnation region for the subsonic NACA 0012 problem.

Fig. 3.16a shows a comparison of the numerical and experimental (see Harris [46]) pressure distributions over the aerofoil. Note that C_p is the non-dimensional pressure coefficient:

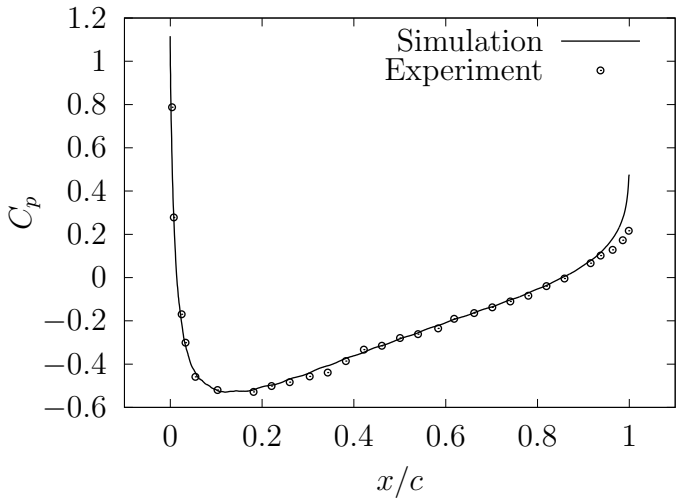
$$C_p = \frac{p - p_\infty}{\frac{1}{2}\rho_\infty u_\infty^2}, \tag{3.46}$$

where u_∞ is the free stream velocity. The computed solution shows good agreement with the experimental measurements over the whole of the aerofoil.

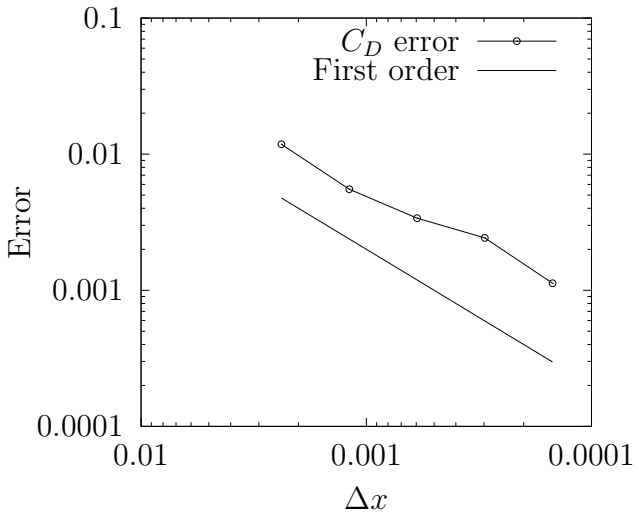
Fig. 3.16b shows how the error in the computed drag coefficient, C_D, decreases with increasing resolution. Note that

$$C_D = \frac{F_D}{\frac{1}{2}\rho_\infty u_\infty^2 c}, \tag{3.47}$$

where F_D is the pressure drag force. The theoretically expected drag force for this non-separating inviscid flow is 0, and any computed positive C_D is a measure of the discretisation error. Fig. 3.16b shows a first order convergence for C_D which is in line with expectations since the cut cell method is first order accurate at the boundary.



(a) Experimental [46] vs numerical surface pressure distributions.



(b) Convergence plot of C_D computation.

Figure 3.16: Surface pressure and drag results for the subsonic NACA 0012 problem.

3.5.4 Shock reflection over a double wedge

The problem of a $M = 1.3$ normal shock reflecting over a double wedge is used to demonstrate the performance of the method for a 2D problem involving concavities. The simulation set-up is shown in Fig. 3.17, where it may be seen that the wedge and shock have been rotated in order to create a fully doubly-shielded concavity (see Section 3.3.3.1) with respect to the grid at point D. Note that points A and B are located upstream of the wedge leading edge, which is at point C. To resolve the detailed solution structure, a fine base resolution of 1200×1200 cells is employed with two levels of AMR refinement of factor

2 each. Boundary conditions on the left, right and top boundaries are all transmissive.

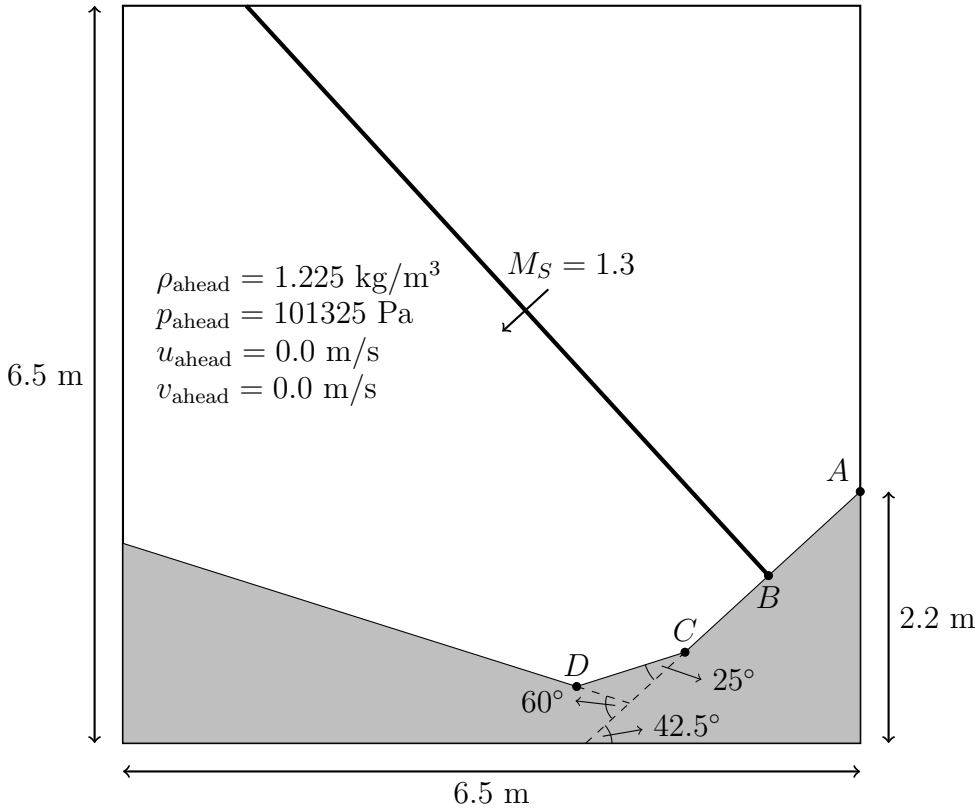
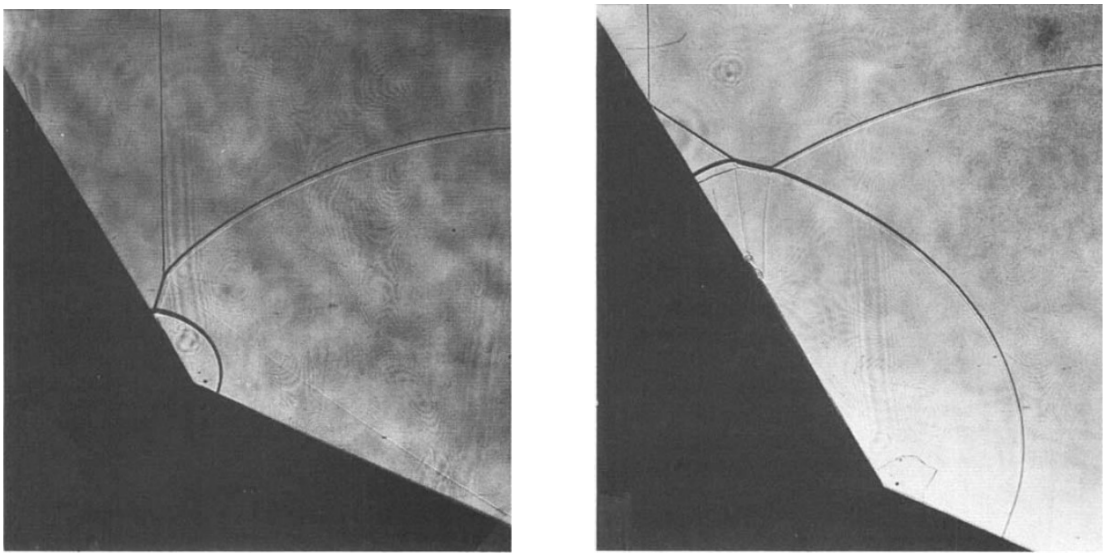


Figure 3.17: Rotated simulation set-up for the problem of shock reflection over a double wedge. Note that $AB = BC = CD = 1$ m.

Fig. 3.18 shows experimental shadowgraphs for this test as measured by Ben-Dor, Dewey and Takayama [47], who provide detailed explanations of the wave interactions that lead to the observed solution structures. Numerical schlierens of the computed density field at $t = 0.0044$ s and 0.005 s are shown in Fig. 3.19a and Fig. 3.19b respectively. For ease of comparison, note that we rotate our results to align them with the experimental frame.

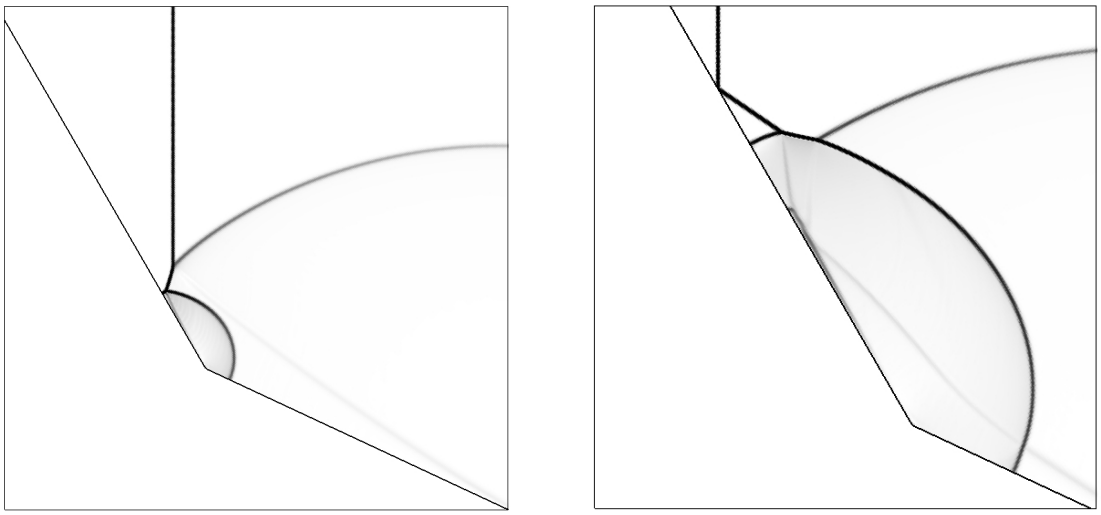
The numerical results show good qualitative agreement with experiment at both times. From Fig. 3.19a, we see that the simulation captures the Mach reflection over the second wedge of the Mach stem from the first wedge reflection. The slip line from the first Mach reflection is also clearly resolved. Noteworthy in Fig. 3.19b is the resolution of the two triple points and their slip lines. Another feature captured by the AMR is the interaction of the slip lines with the contact line from the first Mach reflection. This feature is not discussed by Ben-Dor et al. but may be seen to be just visible in the experimental shadowgraph.



(a) Mach reflection over the second wedge of the Mach stem from the first wedge reflection.

(b) Final flow structure.

Figure 3.18: Experimental shadowgraphs [47] for the shock reflection over a double wedge problem.



(a) Solution at $t = 0.0044$ s.

(b) Solution at $t = 0.005$ s.

Figure 3.19: Numerical schlierens for the shock reflection over a double wedge problem.

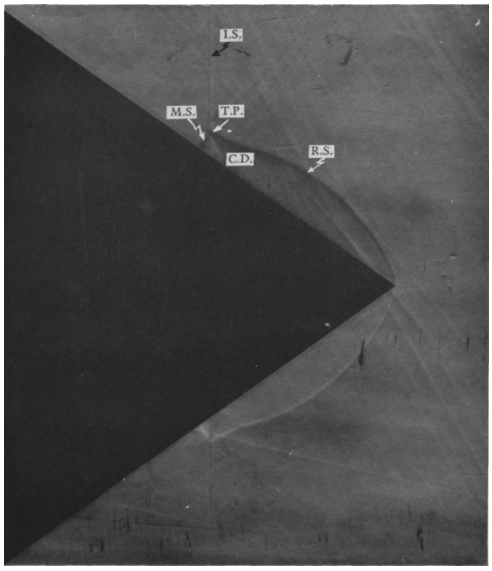
3.5.5 Shock diffraction over a cone

A full 3D simulation of the diffraction of a $M = 3.55$ normal shock over a cone is used to validate the three-dimensional implementation of LPFS. Incidentally, this test has also

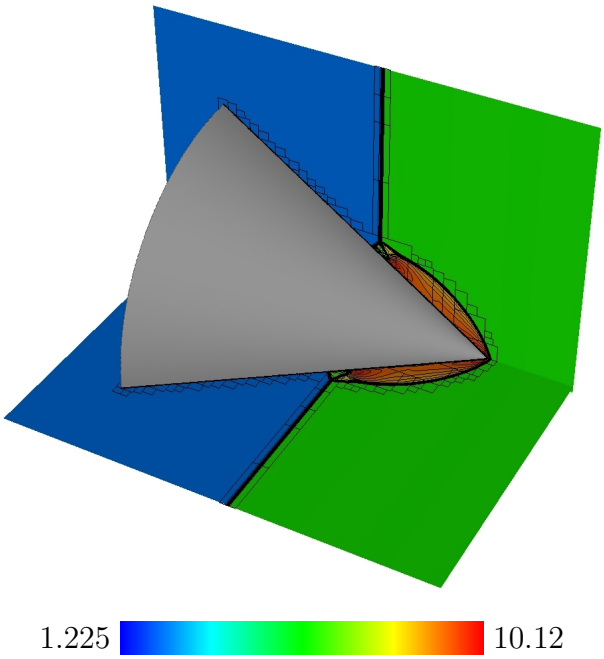
been used by Yang, Causon and Ingram [48] to validate a cell-merging based cut cell method.

The right cone of semi-apex angle 35.1° and length 2.0 m has its apex situated at the point $(0, 0, 0)$ m in a $[-2.5, 0.6]$ m $\times$ $[-2.0, 2.0]$ m $\times$ $[-2.0, 2.0]$ m domain. A base resolution of $62 \times 80 \times 80$ cells is employed with two AMR levels of refinement factors 4 and 2 respectively. The ambient state ahead of the shock has a density and pressure of 1.225 kg/m^3, 101325 Pa respectively.

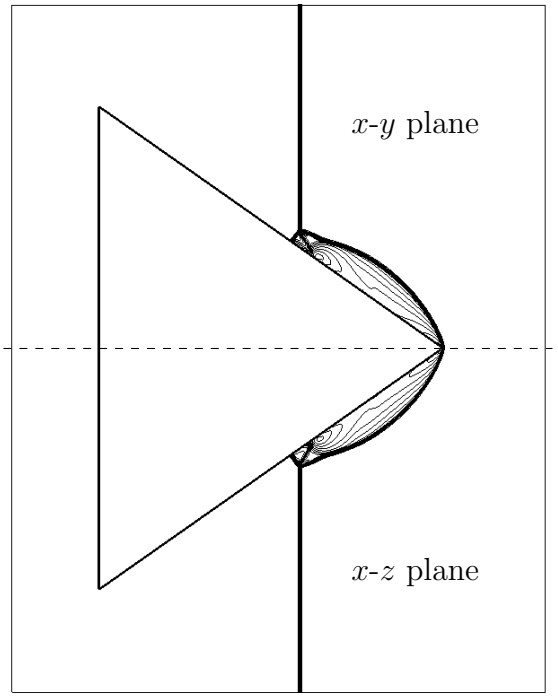
The experimental schlieren measured by Bryson and Gross [49] is shown in Fig. 3.20a. Fig. 3.20b shows the computed density contours on the two symmetry planes (x-y and x-z) of the geometry. All the waves as well as the cut cell boundary are flagged for refinement by the AMR algorithm. The density contours on the symmetry planes are shown more clearly in Fig. 3.20c which shows that the simulation successfully captures the complex three-dimensional Mach reflection pattern. The numerically resolved incident shock, reflected shock, and Mach stem meeting at the triple point are clearly visible, as is the contact wave.



(a) Experimental schlieren [49]. I.S. is the 'incident shock', R.S. is the 'reflected shock', M.S. is the 'Mach stem', C.D. is the 'contact discontinuity', and T.P. is the 'triple point'.



(b) Numerical density contours with AMR patch boundaries on the finest level highlighted.



(c) Numerical density contours on the x-y and x-z symmetry planes.

Figure 3.20: Comparison of experimental and numerical results for the shock diffraction over a cone problem.

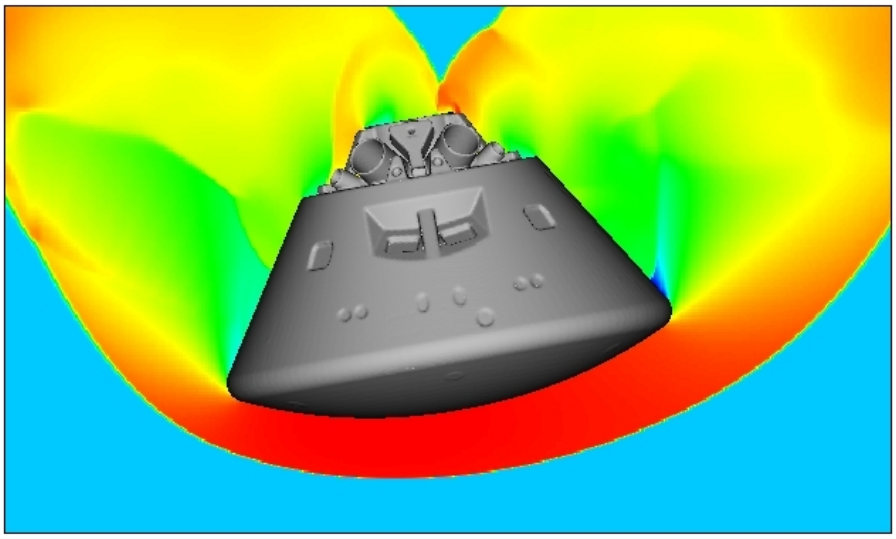
3.5.6 Space re-entry vehicle simulation

The computation of a $M = 20$ flow over a NASA 'Orion' space re-entry vehicle is used to demonstrate the performance of the 3D LPFS implementation when computing a high Mach number flow over a realistic complex geometry. A 'watertight' stl file of the geometry was obtained from the NASA 3D Resources website [50]. Fig. 3.21a shows a view of the geometry from the rear.

In a Mach 20 flow, air molecules undergo dissociation and it is no longer appropriate to use the ideal gas equation of state Eq. 4.4. We stress, therefore, that the aim of this section is only to demonstrate the potential of the methodology and not to compute a physically accurate solution. The simulation is set-up with a supersonic inflow boundary condition at the inlet and transmissive boundary conditions at all other boundaries. The geometry is rotated to make an angle of 10° with the free stream direction. Since this problem is sensitive to the 'carbuncle phenomenon' [51], we use the HLL Riemann solver to compute fluxes from the reconstructed states. We let the simulation run until the bow shock forms ahead of the spacecraft and till the flow impacts all parts of the geometry. One level of AMR is employed to resolve the solid-fluid interface and the bow shock.

Fig. 3.21b shows the natural log of the computed pressure along a plane passing through the model centre line. The potential of the methodology to compute the flow around a complex 3D geometry is apparent from the results. Since we are using an ideal gas equation of state, however, we make no attempt to further analyse the complicated flow field.

(a) Space re-entry vehicle geometry rear view.



(b) Pseudocolour plot of the natural log of pressure.

Figure 3.21: Geometry rear view and simulation results for the space re-entry vehicle simulation problem.

3.6 Conclusions

In this chapter, we presented a 'Local Proportional Flux Stabilisation' (LPFS) approach for computing cut cell fluxes when solving hyperbolic conservation laws, and described its implementation in the dimensionally split framework of Klein et al. [37]. The approach makes use of local geometric and wave speed information to define a novel stabilised cut cell flux.

The convergence and stability of the method was proved for the one-dimensional linear advection equation, and confirmed numerically for multi-dimensional test problems for the linear advection and Euler equations.

Compared to the 'KBN' cut cell flux described by Klein et al., the LPFS flux is designed to give improved accuracy at stagnation points, and this was demonstrated via the computation of a subsonic flow over a NACA 0012 aerofoil. Furthermore, as confirmed from the results of a shock reflection from wedge problem, the LPFS flux was found to alleviate the problem of oscillatory boundary solutions produced by the KBN flux at higher Courant numbers. The performance of the three-dimensional implementation of the method when computing a high Mach number flow over a realistic complex geometry was demonstrated by the computation of a Mach 20 flow over a space re-entry vehicle.

For the future, it is clear that extending the flux stabilisation to maintain second order accuracy at the boundary will yield the greatest improvement in results. The development of a flux to use at concavities which avoids the need to use the current post-sweep conservative correction would also be a useful contribution.

CHAPTER 4

A DIMENSIONALLY SPLIT CARTESIAN CUT CELL METHOD FOR THE COMPRESSIBLE NAVIER-STOKES EQUATIONS

In this chapter, we present a novel cut cell method to solve compressible Navier-Stokes problems containing static rigid boundaries. The method is globally second order accurate in the L_1 norm, fully conservative, and allows the use of time steps determined by the regular grid spacing. This content was recently published in the *Journal of Computational Physics* [2] and is the first presentation of a dimensionally split method for the compressible Navier-Stokes equations in the literature:

N. Gokhale, N. Nikiforakis, and R. Klein. A dimensionally split Cartesian cut cell method for the compressible Navier-Stokes equations. *Journal of Computational Physics*, 375: 1205-1219, 2018.

4.1 Introduction

Since the early 1980s, a number of ways have been presented in the literature to deal with this 'small cell problem' in the context of the Euler equations [21, 22, 52, 27, 28, 29]. Over the past decade, most of these techniques have been extended for solving the compressible Navier-Stokes equations. Hartmann et al. [24] have used cell linking (which is related to the intuitive concept of cell merging) to develop a three-dimensional cut cell method that is implemented in an adaptive octree grid. Using the static boundary cut cell formulation of Hartmann et al. [24], Schneiders et al. [25, 53] have successfully developed a method to compute moving boundary problems in 3D by introducing an interpolation routine and flux

redistribution step. Berger et al. [54] use pseudo-time stepping and a multigrid approach to compute steady state solutions for the 2D Reynolds-averaged Navier-Stokes (RANS) equations. In a subsequent publication [55], they describe a cut cell implementation of a novel ODE-based wall model which, unlike conventional equilibrium wall functions, has the advantage that it can be applied further away from the interface in the wake region of a turbulent boundary layer. Graves et al. [56] extend the 'flux redistribution' technique of Colella et al. [28] for small cell stability to develop a second order accurate method. Another high order discretisation was demonstrated by Muralidharan and Menon [32], who extended the 'flux mixing' technique of Hu et al. [29] to develop a third order accurate scheme.

The aforementioned techniques are all implemented in an unsplit fashion. As discussed in Chapter 1, we are particularly interested in adopting a dimensionally split approach which is a convenient way to extend one-dimensional methods to solve multi-dimensional problems. In that context, Gokhale, Nikiforakis and Klein [1] recently presented a simple dimensionally split cut cell method for hyperbolic conservation laws using a 'Local Proportional Flux Stabilisation' (LPFS) approach and demonstrated its performance through the computation of solutions to a number of challenging problems for the Euler equations. The LPFS method is an improvement on the original split method of Klein, Bates and Nikiforakis (KBN) [37]. Although both methods are first order accurate at the interface, LPFS was shown to produce more accurate solutions near boundaries for hyperbolic problems, and it allows the use of larger Courant numbers [1]. We describe the LPFS approach in detail in Chapter 3. In this chapter, we combine and extend the LPFS and KBN methods to solve compressible Navier-Stokes problems involving rigid embedded boundaries. To the best of our knowledge, this is the first presentation of a dimensionally split cut cell method for the compressible Navier-Stokes equations in the literature and we believe that researchers and practitioners who use dimensionally split approaches for multi-dimensional extensions could find it useful.

The rest of this chapter is organised as follows. In Section 4.2, we outline the governing equations and solution framework that we use. In Section 4.3, we describe the numerical method in detail. In Section 4.4, we present numerical solutions for a number of multi-dimensional test problems to demonstrate the performance of the method. Finally, conclusions and areas for future work are provided in Section 4.5.

4.2 Governing equations and solution framework

The compressible Navier-Stokes equations are

$$\begin{aligned} \partial_t \rho + \nabla \cdot (\rho \mathbf{u}) &= 0, \\ \partial_t (\rho \mathbf{u}) + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u} + pI) &= \nabla \cdot \sigma, \\ \partial_t E + \nabla \cdot [(E+p)\mathbf{u}] &= \nabla \cdot (\sigma \mathbf{u}) + \nabla \cdot (\xi(\nabla T)), \end{aligned} \tag{4.1}$$

where ρ is density, $\mathbf{u}$ is velocity, p is pressure, I is the identity matrix, ξ is thermal conductivity and T is temperature. We assume the fluid under consideration is Newtonian such that σ, the stress tensor, is

$$\sigma = \mu(\nabla \mathbf{u} + \nabla \mathbf{u}^T) + \lambda(\nabla \cdot \mathbf{u})I, \tag{4.2}$$

where μ is the dynamic viscosity and we use the Stokes' hypothesis $\lambda = -\frac{2}{3}\mu$. E is the total energy per unit volume, given by

$$E = \rho \left(\frac{1}{2} |\mathbf{u}|^2 + e \right), \tag{4.3}$$

where e is the specific internal energy. To close the system of equations Eq. 4.1-Eq. 4.3 we use the ideal gas equation of state

$$e = \frac{p}{\rho(\gamma - 1)}, \tag{4.4}$$

where γ, the heat capacity ratio, is assumed to be 1.4.

We defer our description of the procedure for computing the explicit inviscid (Godunov-based) and viscous fluxes till Section 4.3.1, although it may be noted that the flux stabilisation approach is independent of the particular choice of flux methods used. Hierarchical AMR [3] is used to refine areas of interest such as the cut cell interface, or shock waves, while allowing the use of coarser resolutions elsewhere for the sake of computational efficiency. Multi-dimensional updates are performed using Strang splitting [38] in 2D, and straightforward Godunov splitting [39] in 3D, although Strang splitting could also be used in 3D if time order of accuracy was important for the problem at hand.

As in Dragojlovic et al. [57], we assume that the global time step, Δt, is restricted by

the minimum of the hyperbolic and diffusive time steps

$$\begin{aligned}\Delta t &= \min\left[\Delta t_{\text{hyp}}, \Delta t_{\text{diff}}\right], \\ &= \min\left[C_{\text{cfl}} \min_{d,i}\left(\frac{\Delta x_{d,i}}{W_{d,i}^{\max}}\right), \min_{d,i}\left(\frac{\Delta x_{d,i}^2}{2\max(\frac{\mu_i}{\rho_i}, \frac{\xi_i}{(\rho c_p)_i})}\right)\right],\end{aligned} \tag{4.5}$$

where d is the index of the coordinate direction, i is the index of a computational cell, and c_p is the specific heat at constant pressure. $\Delta x_{d,i}$ and $W_{d,i}^{\max}$ are the spatial resolution and max wave speed for cell i in the d direction respectively. As in similar previous works [57, 24, 25, 32], we operate the algorithm in the region where the hyperbolic condition is most restrictive, and all simulations in this paper were run using a Courant number of 0.5.

The wave speed for cell i in the d direction, $W_{d,i}$, is computed using the following estimate suggested by Toro [9]:

$$W_{d,i} = |\mathbf{u}_{d,i}| + a_i, \tag{4.6}$$

where $\mathbf{u}_{d,i}$ is the component of the velocity in cell i in the d direction. a_i is the speed of sound in cell i, given by

$$a_i = \sqrt{\frac{\gamma p_i}{\rho_i}}. \tag{4.7}$$

Domain edge boundary conditions are specified using the 'fictitious cell' approach [9]. In addition to the boundary conditions described in Section 3.2, this chapter also makes use of the no-slip domain edge boundary condition. This is enforced by extrapolating ghost cell variables from the domain, inverting the sign of the ghost velocity, and adding on twice the surface tangential velocity (if the domain boundary is a tangentially moving wall) [9].

4.3 Numerical method

4.3.1 Calculation of explicit fluxes

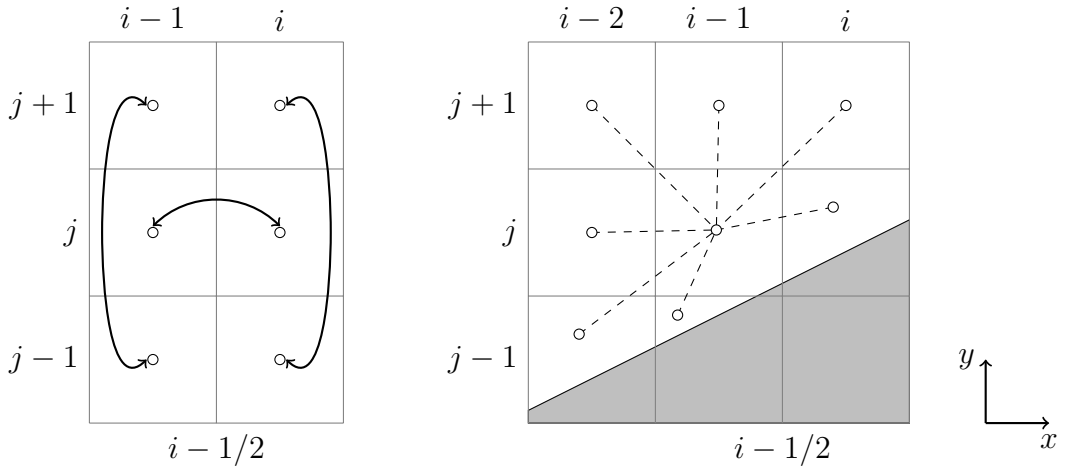
In this subsection, we describe the procedure we use to compute the explicit inviscid and viscous fluxes. Their stabilisation at the cut cells is described in Section 4.3.2.

4.3.1.1 Intercell fluxes

As when computing solutions to the Euler equations in Chapter 3, we compute the inviscid intercell fluxes that appear in the divergence terms to the left of the equal signs in Eq. 4.1

using an exact Riemann solver and the MUSCL-Hancock scheme in conjunction with the van-Leer limiter [9]. This scheme is second order accurate in smooth regions.

The calculation of the viscous fluxes that appear in the divergence terms to the right of the equal signs in Eq. 4.1 requires the computation of $\nabla \mathbf{u}$ and ∇T at the cell faces. In Fig. 4.1, we illustrate our procedure for calculating $(\nabla \phi)_{i-1/2,j}$ for a general scalar ϕ at the left face of cell (i, j) in an x dimensional sweep in two dimensions.



(a) Regular region away from the interface.

(b) Irregular region near the interface.

Figure 4.1: Illustration of the calculation of viscous face derivatives in irregular and regular regions.

In a regular region away from the interface, the required face derivatives can be worked out using central differencing as illustrated in Fig. 4.1a. The derivative in the current coordinate direction is calculated to second order accuracy as

$$\left(\frac{\partial \phi}{\partial x}\right)_{i-1/2,j} = \frac{\phi_i - \phi_{i-1}}{\Delta x}. \tag{4.8}$$

The transverse derivative is computed to second order accuracy as the average of the neighbouring transverse derivatives

$$\left(\frac{\partial \phi}{\partial y}\right)_{i-1/2,j} = \frac{1}{2}\left(\frac{\phi_{i-1,j+1} - \phi_{i-1,j-1}}{2\Delta y} + \frac{\phi_{i,j+1} - \phi_{i,j-1}}{2\Delta y}\right). \tag{4.9}$$

It may be noted that in 3D, the face derivatives can be calculated analogously on a $2 \times 3 \times 3$ point stencil.

The procedure for irregular regions is illustrated in Fig. 4.1b. We start by computing

the inverse distance weighted least squares gradients $(\nabla\phi)^{\mathrm{LS}}_{i-1,j}$ and $(\nabla\phi)^{\mathrm{LS}}_{i,j}$ at the volumetric centroids of cells $(i-1,j)$ and (i,j) respectively. Note that the stencil for the weighted least squares calculations contains all the regular and cut cells from the 8 (26 in 3D) neighbours of the cell. This is illustrated using dotted lines for cell $(i-1,j)$ in Fig. 4.1b. $(\nabla\phi)_{i-1/2,j}$ is calculated as

$$(\nabla\phi)_{i-1/2,j} = \frac{(\nabla\phi)^{\mathrm{LS}}_{i-1,j} + (\nabla\phi)^{\mathrm{LS}}_{i,j}}{2}. \tag{4.10}$$

As with other cut cell discretisations [24, 54], our approximation of the face derivatives in irregular regions is only first order accurate. This does not impact the overall accuracy of the method, however, since the flux stabilisation is also first order accurate at the boundary.

4.3.1.2 Boundary fluxes

As discussed in Section 3.3.3, in order to ensure conservation in a dimensionally split scheme, advective boundary fluxes have to be treated differently to the pressure and diffusive fluxes. The 'reference' boundary state used to evaluate the advective boundary fluxes is calculated using the same approach described in Section 3.3.3 for the Euler equations. For the Navier-Stokes equations, however, both the boundary pressure and the boundary stress tensor have to be updated in between sweeps. Updating the latter is important for computing accurate viscous momentum boundary fluxes. Since we consider only static (no-slip), adiabatic boundaries in this work, note that there are no viscous heating or thermal conductivity boundary fluxes to consider for the energy equation.

To compute $\nabla\mathbf{u}$ at the boundary, we follow the process illustrated for cut cell (i,j) in Fig. 4.2. Note that in the rest of this section, subscripts specified with greek letters are assumed to range from 1 to n_d, the number of dimensions, and repeated indices of that kind imply the use of the Einstein summation convention. Let x_μ represent a Cartesian coordinate system, and $\hat{x}_\mu$ represent an orthonormal coordinate system with a unit vector pointing normal to the cell interface, and unit vector(s) in the interface tangential plane.

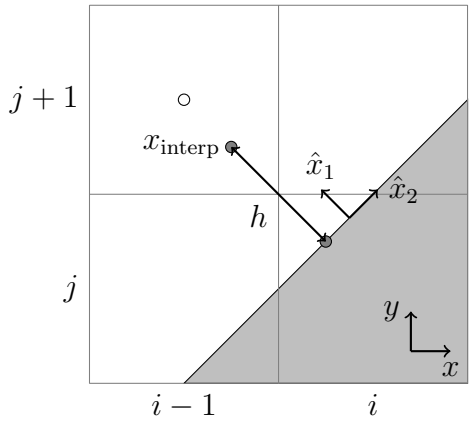


Figure 4.2: Illustration of the procedure used to compute $\nabla \mathbf{u}$ at the boundary of cell (i, j). u^*_μ at x_{interp} is reconstructed from the weighted least squares gradient computed at the nearest cell volumetric centroid (filled white circle).

We start by computing the normal boundary derivatives for each velocity component u_μ as

$$\left(\frac{\partial u_\mu}{\partial \hat{x}_1}\right) = \frac{u^*_\mu - u^b_\mu}{h} = \frac{u^*_\mu}{h}, \tag{4.11}$$

where u^b_μ, the value of the velocity component at the boundary is 0 as required by the no-slip boundary condition. u^*_μ is the value of the velocity component at the interpolation point x_{interp} which is located at a distance h from the interface centroid in the normal direction. We reconstruct the value of u^*_μ using the inverse distance weighted least squares gradient computed at the cell volumetric centroid closest to x_{interp}. h is calculated as in Meyer et al. [44] using

$$h = 0.5\sqrt{(\hat{\mathbf{n}}^b_\nu \Delta x_\nu)^2}, \tag{4.12}$$

where Δx_ν is the spatial resolution in the ν coordinate direction. Note that when using a uniform mesh spacing of Δx in all coordinate directions, $h = 0.5\Delta x$.

With the normal derivatives calculated, and the tangential boundary derivatives known to be 0 because of no-slip and no space dependent wall motion, we can construct the tensor $\partial u_\mu / \partial \hat{x}_\xi$ at the boundary. However, we want to compute the tensor $\nabla \mathbf{u}$ which has Cartesian components $\partial u_\mu / \partial x_\nu$ that are given by

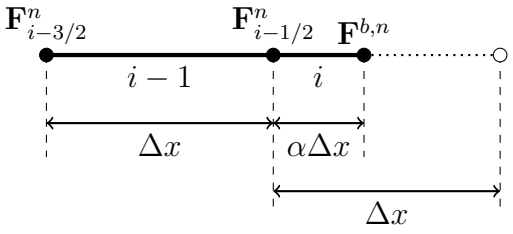
$$\frac{\partial u_\mu}{\partial x_\nu} = \frac{\partial \hat{x}_\lambda}{\partial x_\nu} \frac{\partial u_\mu}{\partial \hat{x}_\lambda}. \tag{4.13}$$

$\partial \hat{x}_\lambda / \partial x_\nu$ can be represented by a matrix whose columns are the unit vectors of the $\hat{x}_\mu$ coordinate system. The normal vector $\hat{\mathbf{n}}^b$ is already known from the information provided by the signed distance function (see Chapter 2), while the vectors spanning the tangential

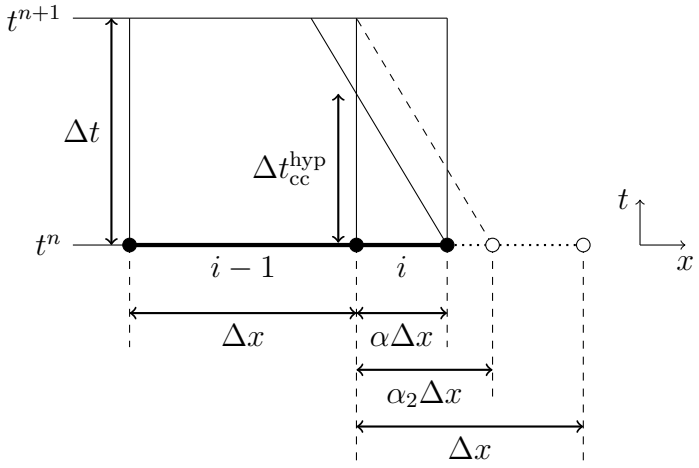
plane are calculated from $\hat{\mathbf{n}}^b$ using a Gram-Schmidt orthogonalisation process. Eq. 4.13 can then be used to compute $\nabla \mathbf{u}$, and hence the stress tensor σ at the boundary as required to compute the momentum diffusion boundary fluxes.

Finally, as discussed in Chapter 5, it may be noted that it is straightforward to introduce the use of a simple algebraic or ODE-based turbulent wall function into the above approach when computing a turbulent flow. The reconstructed state at x_{interp} and the known state at the wall interface centroid can be used as boundary conditions to calculate the wall shear stress.

4.3.2 Flux stabilisation



(a) KBN.



(b) LPFS.

Figure 4.3: Illustration of the KBN and LPFS flux stabilisation procedures for a boundary cut cell neighbouring a regular cell in 1D.

In this subsection, we describe the combination and extension of the KBN and LPFS methods for the Navier-Stokes equations. Fig. 4.3a and Fig. 4.3b illustrate the one-dimensional flux stabilisation procedures used in both approaches for a boundary cut cell i neighbouring a regular cell.

Let $\mathbf{U}_i^n$ represent the conserved variable state vector for cell i at time level n, and let $\mathbf{F}_{i\pm i/2}^n$ represent the explicit numerical fluxes (inviscid and viscous) computed at its ends. As described in Section 3.3.1 [1, 37], the stabilised KBN flux can be derived to be

$$\mathbf{F}_{i-1/2}^{\mathrm{KBN},n} = \mathbf{F}^{b,n} + \alpha(\mathbf{F}_{i-i/2}^n - \mathbf{F}^{b,n}). \tag{4.14}$$

Consider Fig. 4.3b, which shows the boundary cut cell neighbouring the regular cell in the x-t plane for one time step. Δt is the global stable hyperbolic time step which is determined in part by the fastest wave speed in the domain, $W_{\max}$ (see Eq. 4.5). For the configuration of Fig. 4.3b, we illustrate the 'small cell problem' at the cut cell as being caused by the left-going wave from the solution of the boundary Riemann problem. Stability would therefore require the use of the smaller $\Delta t_{\mathrm{cc}}^{\mathrm{hyp}}$

$$\Delta t_{\mathrm{cc}}^{\mathrm{hyp}} = C_{\mathrm{cfl}} \frac{\alpha \Delta x}{W_i}, \tag{4.15}$$

where W_i is the wave speed for the cut cell.

As described in Chapter 3 [1], a suitable LPFS flux is

$$\mathbf{F}_{i-1/2}^{\mathrm{LPFS},n} = \frac{\Delta t_{\mathrm{cc}}^{\mathrm{hyp}}}{\Delta t} \mathbf{F}_{i-1/2}^n + \left(1 - \frac{\Delta t_{\mathrm{cc}}^{\mathrm{hyp}}}{\Delta t}\right) \mathbf{F}_{i-1/2}^{\mathrm{KBN,mod},n}, \tag{4.16}$$

where

$$\mathbf{F}_{i-1/2}^{\mathrm{KBN,mod},n} = \mathbf{F}^{b,n} + \frac{\alpha}{\alpha_2}(\mathbf{F}_{i-i/2}^n - \mathbf{F}^{b,n}), \tag{4.17}$$

and

$$\frac{\alpha}{\alpha_2} = \frac{\Delta t_{\mathrm{cc}}^{\mathrm{hyp}}}{\Delta t} = \epsilon \frac{\alpha W_{\max}}{W_i}. \tag{4.18}$$

The 'wave speeds uncertainty' parameter $\epsilon \in [0, 1]$ is introduced to account for any errors arising from the use of Eq. 4.6 to estimate the cut cell wave speeds. We found setting ϵ to 0.8 to be a robust choice for the wide range of problems tackled in this chapter.

The derivation of the LPFS flux Eq. 4.16 implicitly assumes that the cut cell time step is limited by the local hyperbolic time step $\Delta t_{\mathrm{cc}}^{\mathrm{hyp}}$. Although the global time step for all problems in this work is indeed limited by the hyperbolic time step, it is possible for the local time step at some cut cells to be limited by the local diffusion time step

$$\Delta t_{\mathrm{cc}}^{\mathrm{diff}} = \frac{(\alpha \Delta x)^2}{2 \max(\frac{\mu_i}{\rho_i}, \frac{\xi_i}{(\rho c_p)_i})}, \tag{4.19}$$

in which case one can use the KBN stabilisation Eq. 4.14 but not the LPFS stabilisation.

For a completely stable discretisation, then, we compare the relative magnitudes of Δt_{cc}^{hyp} and Δt_{cc}^{diff} before deciding how to stabilise the flux. For the usual case $\Delta t_{cc}^{hyp} \leq \Delta t_{cc}^{diff}$, we use the LPFS flux Eq. 4.16 at the cut cell cell. If $\Delta t_{cc}^{diff} < \Delta t_{cc}^{hyp}$, on the other hand, we use the KBN flux Eq. 4.14.

4.3.3 Multi-dimensional extension

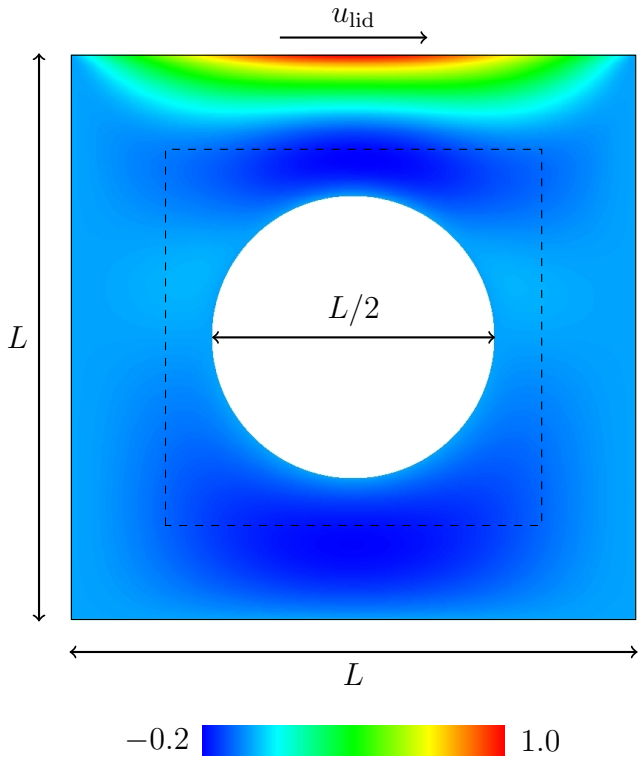
The extension of the 1D flux stabilisation approach to multiple dimensions is performed in exactly the same manner as that described in Chapter 3 for the Euler equations. The reader is referred to Section 3.3.3 for the details.

4.4 Results

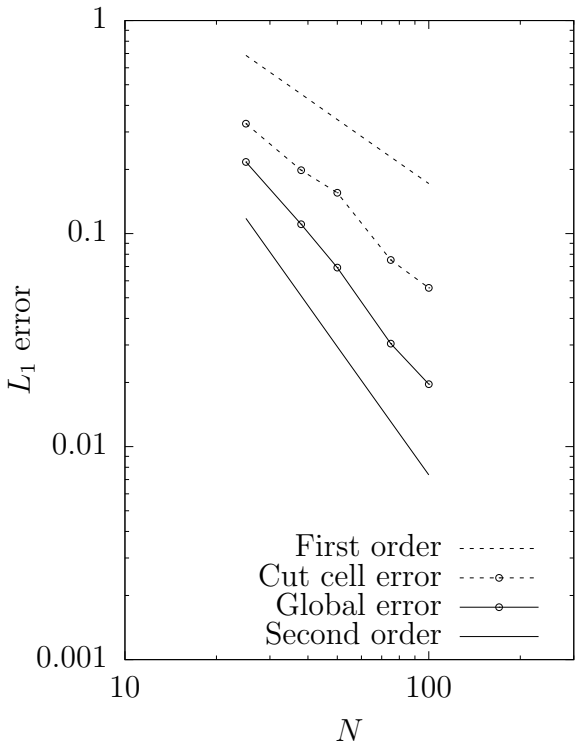
4.4.1 $Re = 20$ lid-driven cavity problem

The lid-driven cavity problem from Kirkpatrick et al. [23] was used to verify the accuracy of the numerical discretisation. Fig. 4.4a illustrates the simulation set-up. The left, right and bottom domain edges are static no-slip boundaries. The top boundary is also no-slip, but with a parabolic x velocity profile u_{lid} which varies from 0 at the boundary edges to u_{lid}^{max} at the centre. u_{lid}^{max} was set to correspond to $M = 0.1$. The Reynolds number of the flow based on the lid length L and u_{lid}^{max} is 20. A cylinder of diameter $L/2$ is placed at the centre of the domain, resulting in the creation of cut cells at the cylinder boundary.

Fig. 4.4a shows the contours of normalised x velocity for the 'reference' solution computed on a fine 400×400 grid, on which the minimum encountered cut cell volume fraction was 2.5×10^{-5}. The simulation was performed at five coarser resolutions, with the errors for these runs computed relative to the reference solution. The dashed box in Fig. 4.4a shows the region used for the error computations. In Section 3.5.1 [1], we showed that for a scheme which is first order accurate at the cut cells and second order accurate elsewhere, the L_p norm of the global solution error converges as $\mathcal{O}(\Delta x^{\frac{p+1}{p}})$. As seen in Fig. 4.4b, the computed solution converges with first order at the cut cells, while the global error measured by the L_1 norm converges with second order accuracy as expected. Note that as discussed in Section 3.5.1, we use Eq. 3.39 to calculate the L_p norm, although it may be argued that the use of Eq. 3.40 would lead to better estimates of the order of convergence of the method.



(a) $u/u_{\text{lid}}^{\max}$ contours for the 400×400 cells reference solution.

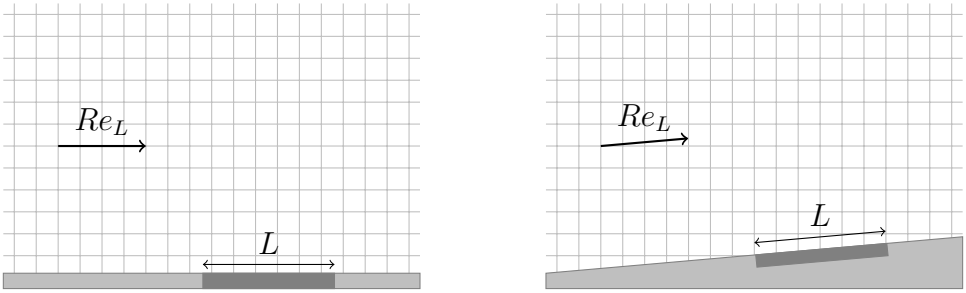


(b) Convergence of cut cell and global L_1 error norms for velocity magnitude. N is the number of cells along one coordinate direction.

Figure 4.4: Velocity results for the $Re = 20$ lid-driven cavity problem.

4.4.2 Laminar flat plate boundary layer

This test involves the computation of a two-dimensional flat plate boundary layer for $M_\infty = 0.2$ and $Re_L = 30000$. We run the test in configurations with the plate both coordinate aligned and non-aligned as illustrated in Fig. 4.5. AMR is used to refine the no-slip region of the plate. Note that a uniform velocity profile is specified at the inflow boundary so that the boundary layer develops on the no-slip part of the plate.

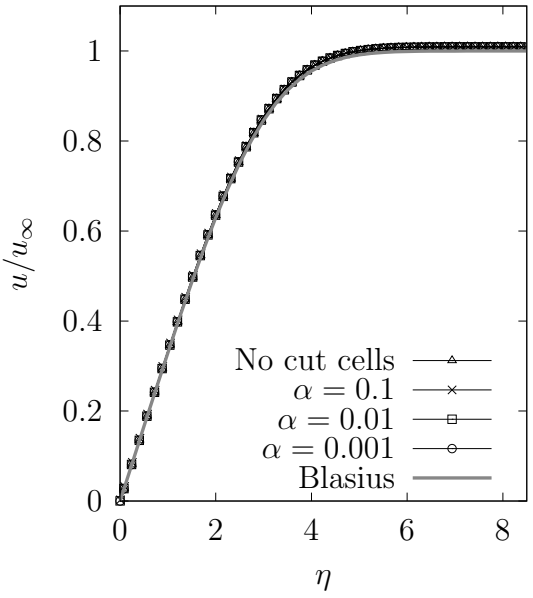


(a) Grid-aligned plate. (b) Plate aligned at 5° to the horizontal.

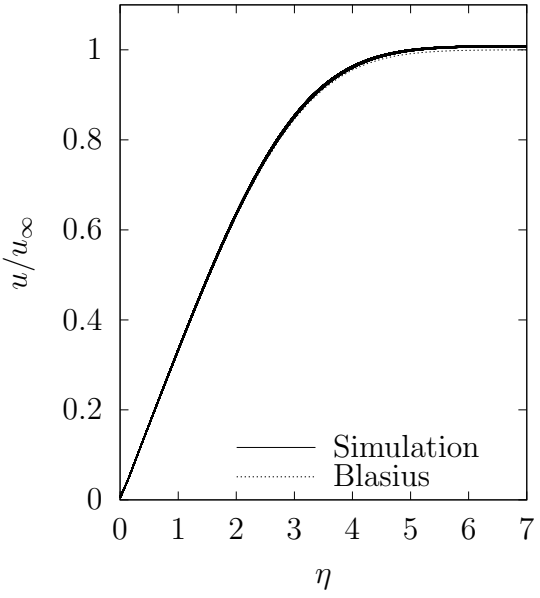
Figure 4.5: Grid aligned and non-aligned configurations for the laminar flat plate boundary layer problem. The shaded region of length L is the no-slip part of the plate. The remainder of the plate is specified as a slip boundary.

For the grid-aligned configuration, the first row of finite volumes adjacent to the plate are all cut cells with the same volume fraction. Fig. 4.6a shows a comparison of the computed boundary layer profile at the centre of the plate for a series of 4 simulations with progressively smaller cut cell volume fractions. Four levels of AMR are employed such that on the finest level, there are roughly 30 cells resolving the 99% boundary layer thickness δ_{99}. All the computed profiles show good agreement with the theoretical Blasius solution, which is a similarity solution for a steady, two-dimensional laminar boundary layer forming over semi-infinite plate in an incompressible flow [58].

We use twice the resolution for the non-aligned plate configuration such that there are roughly 60 cells resolving the δ_{99} thickness at the centre of the plate. Like Graves et al. [56], we compare the computed boundary layer velocity profiles along 'wall normal rays' emanating from every cut cell in the range $5000 \leq Re_x \leq 15000$, which covers cut cells of varying shapes and volume fractions. As seen in Fig. 4.6b, the solutions overlay well and show good agreement with the theoretical Blasius solution.

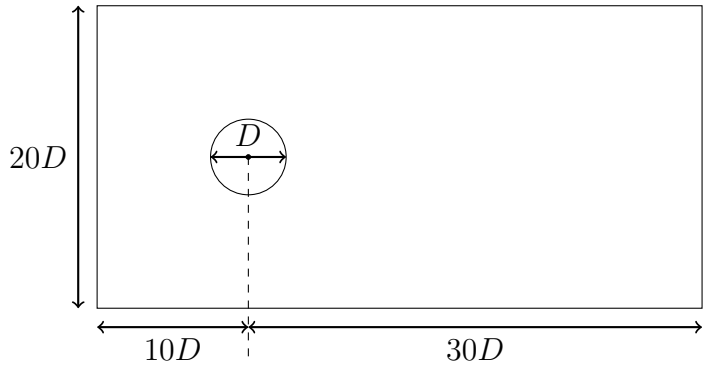


(a) Boundary layer velocity profiles at $Re_x = 15000$ for varying cut cell volume fractions of the lowest row for the grid-aligned configuration.

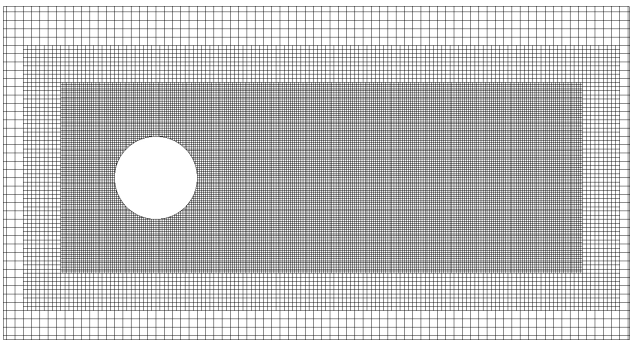


(b) Velocity profiles along wall-normal rays emanating from every cut cell in the range $5000 \leq Re_x \leq 15000$ for the slanted wall configuration with a wall angle of 5°.

Figure 4.6: Computed boundary layer profiles for the (a) co-ordinate aligned, and (b) non-aligned, laminar flat plate boundary layer problem. Note that $\eta = y\sqrt{\frac{u}{\nu x}}$.



(a) Domain dimensions (cylinder not to scale).



(b) AMR mesh in the vicinity of the cylinder.

Figure 4.7: Domain dimensions and an illustration of the Adaptive Mesh Refinement used for the simulations of the flow over a circular cylinder.

4.4.3 Flow over a circular cylinder

The next problem considered is that of two-dimensional flow over a circular cylinder at $Re = 40$ and $Re = 100$. Large amounts of experimental and numerical results exist in the literature for both cases. At $Re = 40$, the eddies behind the cylinder are attached and we can evaluate the accuracy and convergence rate of the computed surface solutions. At $Re = 100$, the wake is unstable and we can also compute the frequency of the vortex shedding process.

The simulation set-up is illustrated in Fig. 4.7a. The left boundary is subsonic inflow while subsonic outflow boundary conditions are specified on the other domain edges. As illustrated in Fig. 4.7b, AMR is used to resolve the near wall solution and cylinder wake. M_∞ is set to 0.1.

The simulations for both Reynolds numbers are run at 4 resolutions which are specified such that there are 40, 60, 80 and 160 cells respectively resolving the cylinder diameter D on the finest AMR level. At the finest resolution, the minimum encountered cut cell volume fraction is 6.7×10^{-5}. The results from the finest resolution simulations are treated

Table 4.1: Comparison of the computed drag coefficient with previous experimental and numerical studies for the $Re = 40$ cylinder flow problem.

Study	C_D
Present work ($D/\Delta x = 160$)	1.57
Tritton [59] (experiment)	1.57
Tseng and Ferziger [60] (simulation, resolution: $\pi D/\Delta x \approx 72$)	1.53
Meyer et al. [44] (simulation, resolution: $D/\Delta x = 72$)	1.56

Table 4.2: Comparison of the computed drag coefficient and Strouhal number with previous experimental and numerical studies for the $Re = 100$ cylinder flow problem.

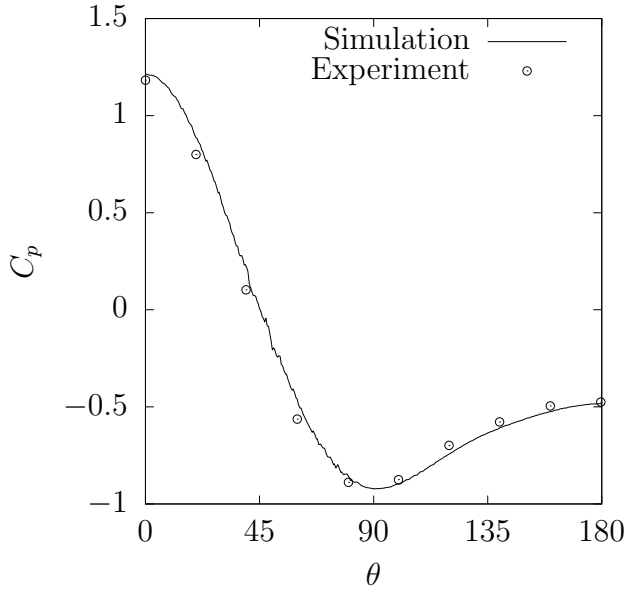
Study	$\overline{C_D}$	St
Present work ($D/\Delta x = 160$)	1.41	0.165
Relf [59] (experiment)	1.39	-
Wieselsberger [62] (experiment)	1.40	-
Williamson [63] (experiment)	-	0.165
Tseng and Ferziger [60] (simulation, resolution: $\pi D/\Delta x \approx 72$)	1.42	0.165
Lai and Peskin [63] (simulation, resolution: $D/\Delta x \approx 38$)	1.45	0.165

as the reference solutions to which errors from the coarser simulations are compared.

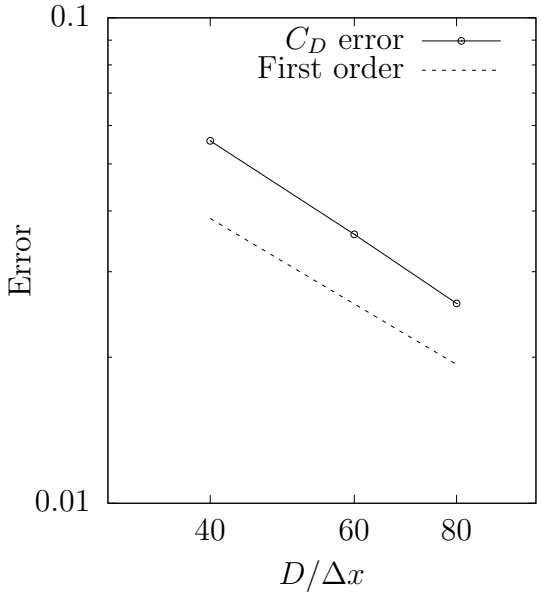
As shown in Table 4.1, the computed drag coefficient from the reference solution for the $Re = 40$ case is in very good agreement with previous experimental and numerical studies.

Fig. 4.8a shows the computed pressure distribution over the cylinder for the $Re = 40$ case. The results compare well with experimental measurements from Grove et al. [61] over the whole of the cylinder. Furthermore, as shown in Fig. 4.8b the computed drag coefficient shows the expected first order convergence rate to the reference solution.

At $Re = 100$, the simulation produces a periodic vortex shedding flow pattern (Fig. 4.9a shows instantaneous computed vorticity contours in the wake of the cylinder). Table 4.2 compares the computed time-averaged drag coefficient $\overline{C_D}$ and Strouhal number St with previous experimental and numerical studies. The frequency of the vortex shedding is calculated from the Fourier Transform of the solution for y velocity at a point in the middle of the wake located a distance of $1.5D$ behind the cylinder. The simulation results are in good agreement with the previous studies. As shown in Fig. 4.9b, the computed drag coefficients also converge with first order as expected.

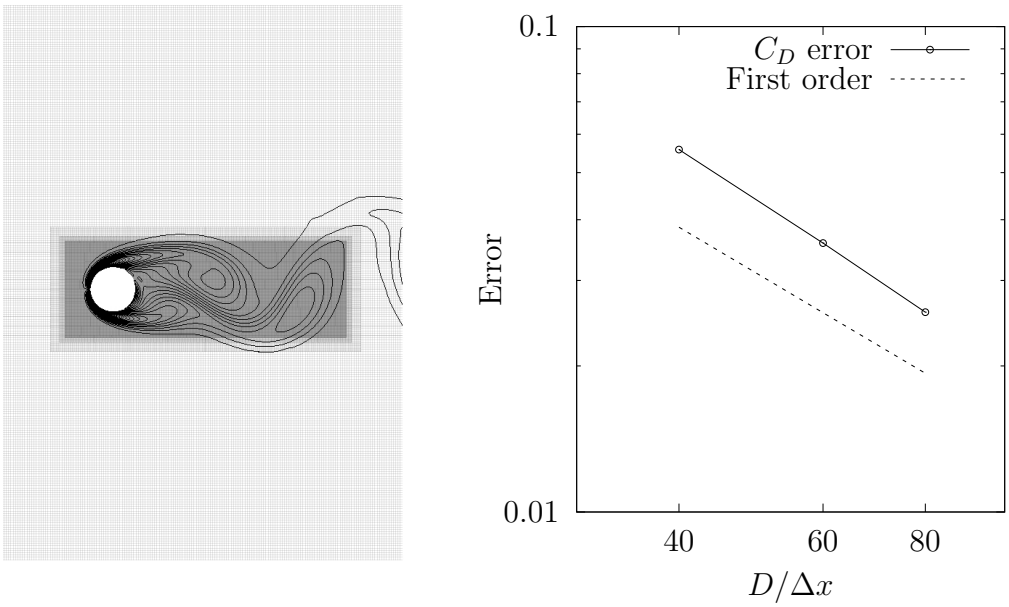


(a) Comparison of the numerical and experimental [61] surface pressure distributions.



(b) Convergence plot of C_D computation.

Figure 4.8: Results of surface pressure distribution and drag coefficient convergence for the $Re = 40$ cylinder flow problem.



(a) Instantaneous vorticity contours overlaid on the mesh in the vicinity of the cylinder.

(b) Convergence plot of C_D computation.

Figure 4.9: Results of vorticity and drag coefficient convergence for the $Re = 100$ cylinder flow problem.

4.4.4 Shock reflection from a wedge

For the compressible flow problem of the shock reflection from a wedge, we use the simulation parameters from Graves et al. [56] shown in Table 4.3. As illustrated in Fig. 4.10, the initial conditions are zero velocity with a discontinuity in pressure and density. The horizontal ground is also discretised with cut cells of volume fraction 10^{-3}. A base resolution of 1024×512 cells is employed with two AMR levels of refinement factor 2 each to resolve the waves and cut cell interfaces. The minimum encountered cut cell volume fraction on the wedge surface is 4.9×10^{-8}. The ground (i.e. the bottom boundary) and wedge are no-slip boundaries, while transmissive boundary conditions are specified at the left, right and top domain edges.

Table 4.3: Initial conditions (with reference to Fig. 4.10) for the shock reflection from wedge problem.

Variable	Value
p_0 (Pa)	1.95×10^3
p_1 (Pa)	7.42×10^5
ρ_0 (kg/m^3)	3.29×10^{-2}
ρ_1 (kg/m^3)	3.61×10^{-1}
μ (kg/(m s))	1.21×10^{-3}
C_v (J/(kg K))	3.00×10^2
ξ (W/(m K))	1.7×10^{-2}
γ	5/3

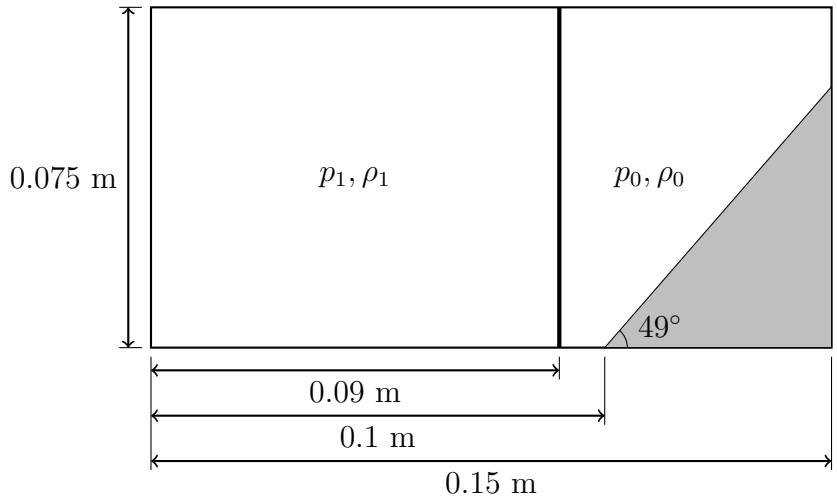
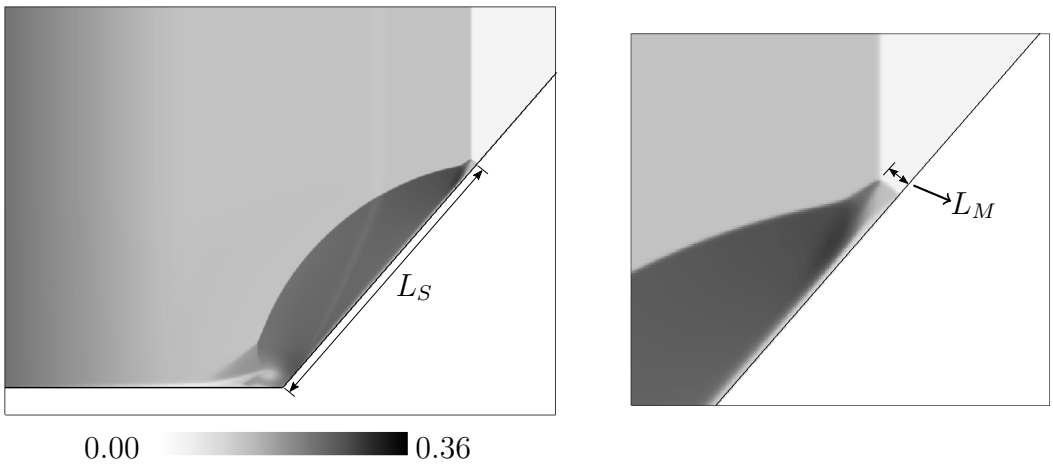


Figure 4.10: Simulation set-up for the shock reflection from wedge problem.

The solution evolves to form a left-propagating rarefaction and a right-propagating $M = 7.1$ shock. A separation bubble is created as the shock reflects off the ground boundary layer. Subsequent compression from the shock reflected off the wedge causes boundary layer reattachment. Fig. 4.11 shows density contour plots of the solution at $t = 9.6\mu$s. Note that the extra wave visible behind the shock in Fig. 4.11a is the numerical 'start-up error' [64] caused by starting the simulation with sharply discontinuous initial conditions.



(a) L_S is the distance travelled by the Mach stem up the wedge.

(b) Close-up of the Mach stem of length L_M.

Figure 4.11: Density contour plots at $t = 9.6\mu$s of the solution for the shock reflection from wedge problem.

The parameter of interest is

$$R_M = \frac{L_M}{L_S}, \tag{4.20}$$

the ratio of the Mach stem length to the distance travelled by the Mach stem up the wedge. L_S is determined from a line-out of the solution taken along the wedge surface. To avoid ambiguity in the determination of the position of the triple point, we compute L_M via an algebraic approach. Consider Fig. 4.12, which illustrates the position of the incident shock (ET) and Mach stem (TD) at the end of the simulation. Once distances BD (L_S) and AC are known, TD (L_M) and hence, R_M can be calculated via trignometry. AC is determined from another line-out of the solution taken horizontally through the incident shock. Assuming an error of $\pm\Delta x$ in the determination of the average positions of the incident shock and Mach stem from the lineouts, we calculate $R_M = 0.0293 \pm 0.004$. Table 4.4 shows a comparison of our computed value for R_M with previous numerical studies. The results from all three studies agree to 1 significant figure - differences in measurement procedures used in the studies are the most likely explanation for why closer agreement is not obtained. In our results, for example, $\Delta x/L_M \approx 6.3\%$, suggesting that the reported value for R_M is very sensitive to the method used to measure L_M.

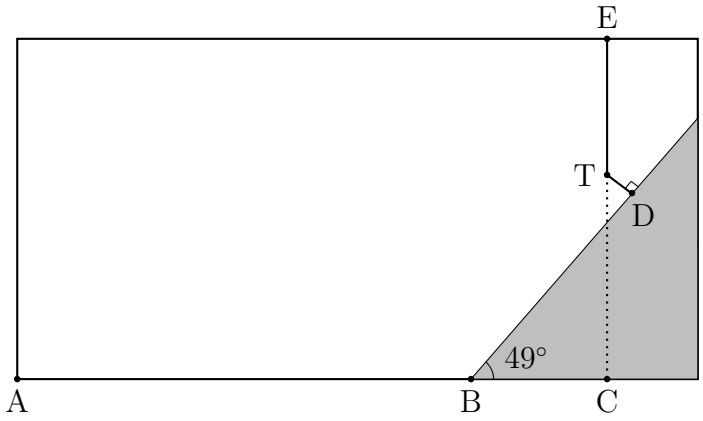


Figure 4.12: Illustration of the positions of the incident shock (ET) and Mach stem (TD) at the end of the shock reflection from wedge simulation.

Table 4.4: Comparison of the computed value of R_M with previous numerical studies for the shock reflection from wedge problem.

Study	R_M
Present work	0.0293 ± 0.004
Graves et al. [56]	0.03
Al-Marouf and Samtaney [65]	0.027

4.4.5 Three-dimensional supersonic flow over a sphere

The final test problem is that of three-dimensional, supersonic flow over a sphere for $M_\infty = 2.0$ and $Re_D = 6.5 \times 10^5$. The sphere of diameter $D = 0.1$ m is placed at a distance of $2.5D$ from the inlet in a domain of size $10D \times 5D \times 5D$. The free stream density ρ_∞, velocity u_∞ and temperature T_∞ are set to be 0.2199 kg/m^3, 527.2 m/s and 173 K respectively as in Uddin et al. [66]. A supersonic inflow boundary condition is specified at the inlet, while transmissive boundary conditions are used at all the other domain boundaries. A base resolution of $100 \times 50 \times 50$ cells is employed and three levels of AMR of refinement factor 2 each are used to resolve the interface and sphere bow shock. The minimum encountered cut cell volume fraction is 2×10^{-5}. The simulation is run until a time of $t = 100D/u_\infty$. Fig. 4.13 shows a plot of the AMR patches overlaid on contours of normalised velocity magnitude ($|\mathbf{v}|/u_\infty$) along the x-y symmetry plane of the sphere at the end of the simulation. Clearly visible in the figure are the bow shock ahead of the sphere and the viscous wake behind it.

The two results parameters of interest to us to validate the solution are the shock

standoff distance δ/D, and the drag coefficient C_D. δ is the shortest distance from the shock to the sphere leading edge. As seen in Table 4.5, our computed values for δ/D and C_D show good agreement with previous experimental and numerical studies. Note that using one less AMR refinement factor produced no change to the drag coefficient computed to three significant figures, suggesting that our final resolution is sufficient to produce a numerically converged solution.

Table 4.5: Comparison of the computed shock standoff distance and drag coefficient with previous experimental and numerical studies for the three-dimensional supersonic flow over a sphere problem.

Study	δ/D	C_D
Present work	0.175	1.01
Krasil'shchikov and Podobin [67] (experiment)	0.17	1.00
Bailey and Hiatt [68] (experiment)	-	1.00
Uddin et al. [66] (simulation)	0.18	1.07
Al-Marouf and Samtaney [65] (simulation)	0.183	0.96

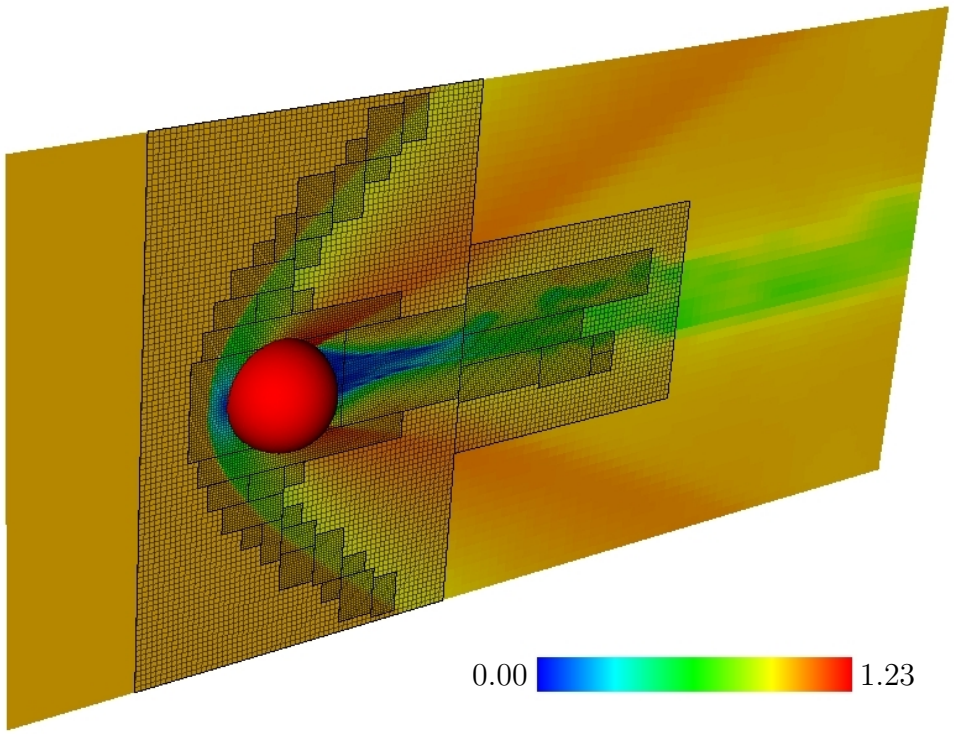


Figure 4.13: Plot of AMR patches on the second and third refinement level overlaid on a contour plot of $|\mathbf{v}|/u_\infty$.

4.5 Conclusions

In this chapter, we presented a novel dimensionally split Cartesian cut cell method for the compressible Navier-Stokes equations. The numerical performance of the scheme was investigated through a number of test problems ranging from the nearly incompressible to the highly compressible flow regimes.

The computation of a laminar boundary layer over a flat plate in both horizontal and inclined configurations was used to demonstrate that the method can handle both coordinate aligned and non-aligned interfaces. The numerical convergence of the method was verified explicitly by computing solutions for a $Re = 20$ lid-driven cavity problem, and for flow over a circular cylinder at $Re = 40$ and $Re = 100$.

For the highly compressible problem of a $M = 7.1$ shock reflecting off a wedge, the complex shock boundary layer interaction pattern was accurately captured by the method, and the computed ratio of the length of the Mach stem to the distance travelled by it up the ramp showed good agreement with previous numerical investigations. Finally, the three-dimensional performance of the method was verified by computing a supersonic flow over a sphere at $Re = 6.5 \times 10^5$. The computed shock standoff distance and drag coefficient showed good agreement with previous experimental and numerical studies.

We believe that practitioners who are interested in using dimensionally split approaches to solve multi-dimensional Navier-Stokes problems in a cut cell code will find this work useful. It is relatively straightforward to extend it to be applicable for problems in the turbulent flow regime by adopting an explicit or implicit Large Eddy Simulation approach combined with a wall function to compute the wall shear stresses. We present the progress we have made on implementing such a 'wall-modelled LES' scheme in the context of the split method in Chapter 5. For the future, modifying the method to achieve second-order accuracy at the boundary would also be a very useful contribution.

CHAPTER 5

ASSESSMENT OF A WALL-MODELLED IMPLICIT LES AND CARTESIAN CUT CELL APPROACH FOR COMPUTING EXTERNAL AUTOMOTIVE FLOWS

In this chapter, we build on the content of the previous chapters to present a wall-modelled Implicit Large Eddy Simulation (WMLES) approach to computing turbulent flows around rigid bodies using the dimensionally split cut cell method. Our target application is the calculation of external flows around automotive geometries.

To date, only Islam and Thornber [69] have published results obtained by using an Implicit LES approach to compute an automotive flow. They use a body-fitted structured mesh and present results only for the relatively simple SAE Notchback reference geometry [70]. With the potential offered by ILES to compute a turbulent automotive flow without user-adjustable parameters, the methodology deserves further investigation. In Section 5.2.2 and Section 5.2.3 of this chapter, we present our results obtained using the Cartesian cut cell approach for the SAE Notchback model and the more realistic DrivAer [7] geometry. Another area in which their work differs from ours is that they use a Detached Eddy Simulation rather than a WMLES approach at the wall.

Aljure et al. [71] recently presented results for the flow around the DrivAer geometry using incompressible WMLES on an unstructured mesh. Apart from the type of mesh used, the main distinctions between our approach and theirs are that we retain compressibility in the governing equations and use an Implicit LES model so that there are no user-tunable parameters.

With the potential for automatic mesh generation offered by the Cartesian cut cell approach, the work described here is of academic and practical interest. Although we

obtain promising results, the contents of this chapter should be looked at as a preliminary assessment to demonstrate the potential of the methdology. Following a more thorough assessment (see Section 5.3), we would expect to present these results as part of a future publication. It would be the first assessment of a WMLES Cartesian cut cell approach for computing automotive flows to be presented in the literature.

The rest of this chapter is organised as follows. In Section 5.1.1, we introduce the ILES approach in the context of other turbulence modelling techniques. In Section 5.1.2, we summarise the Detached Eddy Simulation (DES) and WMLES approaches for modelling turbulence near a wall and provide specific details of the WMLES strategy that we employ. Section 5.1.3 and Section 5.1.4 deal, respectively, with the implementation details of the wall model on a cut cell mesh, and the method we use to set the simulation cell size. In Section 5.2, we present results for the turbulent flow over a square cylinder, and for flow over the SAE Notchback and DrivAer geometries. Finally, conclusions and areas for future work are provided in Section 5.3.

5.1 Theory and numerical method

5.1.1 Turbulence modelling

The Reynolds number,

$$\mathrm{Re} = \frac{\rho u l}{\mu} = \frac{u l}{\nu}, \tag{5.1}$$

represents the ratio of inertial to viscous forces in a fluid. In Eq. 5.1, ρ is density, u and l are the characteristic velocity and length scales respectively, μ is dynamic viscosity and ν is kinematic viscosity.

As described in textbooks on classical turbulence theory (see Davidson [72], for example), in high Reynolds number flows, instabilities in the mean flow generate swirling structures called eddies. The larger eddies extract kinetic energy from the mean flow and transfer it to smaller eddies, setting up an energy cascade which continues down to the Kolmogorov length scale, η, where eddies have a Re of $\mathcal{O}(1)$. Viscous forces just balance the inertial forces at the Kolmogorov length scale, and the energy is completely dissipated. Flows over cars are well into the turbulent flow regime.

By using a spatial resolution of $\Delta x \sim \eta$, it would seem possible to compute turbulent flows to a high degree of accuracy via an approach known as 'Direct Numerical Simulation' (DNS). Unfortunately, it is possible to show through dimensional analysis [72] that η scales

as

$$\frac{\eta}{l} = \mathrm{Re}^{-3/4}, \tag{5.2}$$

so that the number of cells in a 3D simulation $N_x \sim \mathrm{Re}^{9/4}$, and the computation time

$$T_{\mathrm{comp}} \sim Re^3. \tag{5.3}$$

The high Reynolds numbers of $\mathcal{O}(10^6)$ which are typically found in automotive flows [70, 73] make the computation time for a DNS of the flow over a car practically infeasible, necessating the use of turbulence modelling of some description.

To avoid having to resolve all the length scales, a common approach involves time-averaging the Navier-Stokes equations and solving the resulting 'Reynolds Averaged Navier-Stokes (RANS) Equations' for the mean flow. The time-averaging operation results in the creation of six additional unknowns expressed in terms of the fluctuating velocity components. These 'Reynolds stresses' are modelled via user-tunable RANS models, such as the 'k-ϵ' [74] and 'Spalart-Allmaras' [75] models. This approach is usually unable to capture the solution accurately at all points in the domain, however. While the model parameters can often be adjusted to predict boundary layer separation well, for example, they are usually in error in regions of large separation behind a bluff body [76].

Large Eddy Simulation (LES) is a turbulence modelling technique that has resolution requirements that lie in between full DNS and RANS. In the LES approach, the Navier-Stokes equations are spatially filtered using a cut off width such that only the larger more anisotropic eddies are resolved. Spatial filtering results in the creation of unknown 'sub-grid stresses' which represent the effects of the smaller nearly isotropic eddies on the resolved flow. These must be modelled via a sub-grid-scale (SGS) model.

We focus on using the 'Implicit Large Eddy Simulation (ILES)' approach to turbulence modelling. Unlike LES where an explicit SGS model is used, ILES relies on the inherent numerical viscosity in a numerical scheme (caused by even order derivatives in the truncation error) to act as an 'implicit' SGS model. This is where our use of the MUSCL-Hancock scheme and the van-Leer limiter to compute the the hyperbolic fluxes becomes important. Since the numerical method is capable of capturing steep solution gradients (caused by discontinuities such as shocks), it is also capable of capturing the energy cascade from the large eddies down to the smallest resolved eddies. Oran and Boris [77] refer to this as a 'convenient conspiracy' and present a number of results to support the idea.

The form of the numerical viscosity also turns out to be convenient. The reader is referred to Grinstein and Fureby [8] who present a detailed 'modified equation analysis'

for the case of using of a general high resolution method to solve the compressible Navier-Stokes equations. They find that the built-in SGS tensor present in the leading order truncation error terms can be split into 3 parts - one term which is due to the high order nature of the scheme and would also appear with explicit LES, a 'slow' term and a 'rapid' term, where we describe the latter terms using the terminology of Shao et al. [78]. The 'slow' part of the SGS tensor is responsible for capturing locally isotropic turbulence, while the 'rapid' part relates to the anisotropic inhomogeneous part responsible more for turbulence production than dissipation.

It may be noted that Garnier *et al.* [79] have found that the straightforward application of second order shock capturing schemes in ILES can prove to be overly dissipative for the small length scales. A number of remedies to this problem have been presented in the literature, and we briefly mention the noteworthy approaches here. Oßwald et al. [80], for example, present a modified low-Mach Roe approximate Riemann solver that alleviates the excessive dissipation problem and produces correct scaling for the discrete pressure at low Mach numbers. Thornber et al. [81] propose a method that provides similar benefits but involves modifying the reconstructed velocities at the cell faces. Note that the success of the latter approach is dependent on the limiter used, and it is not compatible with the van-Leer limiter. Meinke et al. [82] have proposed the use of a modified version of the Advection Upstream Splitting Method (AUSM) [83] to alleviate the problem with a specific focus on LES. Indeed, the technique has subsequently been used in a number of recent LES studies [84, 85]. Assessing the performance of these techniques is beyond the scope of this work.

5.1.2 Wall modelling

A fundamental assumption in LES is that the smaller eddies are passive so that their effect on the larger eddies can be appropriately accounted for via a SGS model. This assumption breaks down for wall-bounded turbulent flows, however, where dynamically important anisotropic small eddies exist in the turbulent boundary layer close to the wall. Resolving these make the resolution requirements quickly approach that of conventional DNS. Chapman [86], for example, found that for computing flow over an aerofoil with LES, $N_x \sim \mathrm{Re}^{1.8}$. To compute an LES of a a high Reynolds number automotive flow, it is therefore necessary to model, rather than resolve, the region close to the wall. In this context, we may identify two approaches that are commonly used in the literature: 'Detached-Eddy Simulation' (DES) and 'Wall-Modelled Large Eddy Simulation' (WMLES).

5.1.2.1 DES

In the DES approach, the attempt is to combine the benefits of RANS and LES techniques by using unsteady RANS models to compute the solution in the near wall region, and LES in regions away from the wall. A DES limiter which depends on the grid spacing is used to switch between the models. With 'Delayed Detached Eddy Simulation' (DDES), the limiter is adjusted to depend also on the solution by using the eddy viscosity to attempt to detect the presence of the boundary layer [87]. 'Improved Delayed Detached Eddy Simulation' (IDDES) [88] introduces additional modifications to the model to alleviate problems such as the 'logarithmic-layer mismatch', where a discrepancy exists between the computed mean velocity and that provided by the logarithmic law of the wall [72]. The author is referred to the authoritative review by Spalart [76] for a summary of these techniques. Indeed, DES has been used extensively in the context of computing automotive flows. Islam and Thornber [69], for example, employ a DES approach that combines high order Implicit LES and the Spalart-Allmaras one equation RANS model to compute the flow over a SAE Notchback model [70]. Ashton et al. [73] use, among others, an incompressible IDDES (with the SST RANS model) approach to compute the flow over the DrivAer automotive geometry [7] in its fastback configuration. We present our results for the flow around these geometries in Section 5.2.2 and Section 5.2.3 respectively.

5.1.2.2 WMLES

The basic idea behind the WMLES approach is to use the Reynolds-Averaged thin boundary layer equations in the near-wall zone to compute the shear stress at the wall. The most common approach involves neglecting the unsteady, convective and pressure gradient terms in the streamwise momentum equation, leading to the equilibrium diffusion ODE

$$\frac{\partial}{\partial y}\left((\mu+\mu_t)\frac{\partial u}{\partial y}\right)=0, \tag{5.4}$$

where u is the mean streamwise velocity and y represents the wall normal direction. The turbulent viscosity μ_t is usually estimated using a mixing length model and a damping function to ensure the correct behaviour of the turbulent viscosity close to the wall [89]:

$$\mu_t=\kappa\mu y^+(1-e^{-y^+/A}), \tag{5.5}$$

where κ is the von Karman constant and A is the damping factor model constant. The non-dimensional height above the plate,

$$y^+ = \frac{yu_\tau}{\nu}, \tag{5.6}$$

where the friction velocity

$$u_\tau = \sqrt{\frac{\tau_{\text{wall}}}{\rho}}. \tag{5.7}$$

The shear stress at the wall,

$$\tau_{\text{wall}} = \mu \left(\frac{\partial u}{\partial y} \right)_{y=0}. \tag{5.8}$$

Note also that the non-dimensional velocity $u^+ = u/u_\tau$.

Integrating Eq. 5.4 gives the well known solution $y^+ \approx u^+$ for $y^+ \lesssim 5$, i.e., in the viscous sublayer, and the logarithmic law of the wall for $y^+ \gtrsim 30$ [90]. The boundary conditions to solve the ODE are provided by the solution state at the coupling point between the LES and wall models, and the no-slip condition at the wall. In practice, it only makes sense to undertake the extra expense associated with discretising and solving the wall ODE if one or more of the unsteady, convective and pressure gradient terms have been retained, as in Capizzano [91] or Berger and Aftosmis [55].

When using the equilibrium model as we do, it is much more computationally efficient to use an algebraic relation such as the Spalding wall function [92], which provides the following equation for the velocity profile in a fully developed turbulent boundary layer:

$$y^+(u^+) = u^+ + e^{-\kappa B} \left(e^{-\kappa u^+ - 1 - \kappa u^+ - \frac{1}{2}\left(\kappa u^+\right)^2 - \frac{1}{6}\left(\kappa u^+\right)^3} \right). \tag{5.9}$$

Given the solution at the LES/wall-model coupling point, Eq. 5.9 can be used to solve for u^+ which allows the computation of the wall shear stress. We use typical values for κ and B of 0.41 and 5.033 respectively [54]. Fig. 5.1 shows the developed turbulent velocity profile described by Spalding's law.

It is reasonable to question the validity of using the equilibrium model when computing non-equilibrium separating flows. As noted by Larsson et al. [90], however, because the dynamics of the inner layer occur at a much faster time-scale than the larger scale dynamics of the outer layer (which the LES can capture), it is not unreasonable to consider the inner layer as being close to equilibrium even in a separating flow. Furthermore, the convective and pressure gradient terms will approximately balance each other above the viscous layer of the boundary layer (in the limit of weak turbulence, the equations reduce to the Euler equations) so that a WMLES approach based on the equilibrium model is justifiable, even

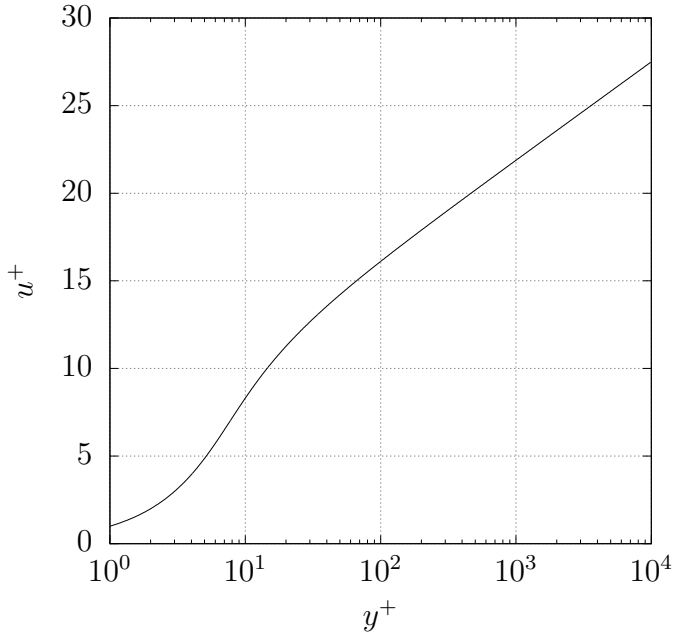


Figure 5.1: Turbulent boundary layer profile described by Spalding's law.

for computing automotive flows.

Perhaps for reasons of computational expense, the WMLES approach has not been as widely employed in the literature as DES-based methods to compute automotive flows. There are noteworth exceptions, however. Serre et al. [93] and Aljure et al. [71], for example, have demonstrated the potential of incompressible WMLES to compute the flow around the Ahmed [94] and DrivAer reference geometries respectively. The main distinctions between our approach and these previous studies are that we retain compressibility in the governing equations, use an Implicit LES model (so that there are no user-tunable parameters) and compute the solution on a Cartesian cut cell mesh.

5.1.3 Implementation of the wall model

As in Section 4.3.1.2, where we compute boundary fluxes for the Navier-Stokes equations, the aim is to make use of the wall function to compute the tensor $\nabla \mathbf{u}$ at the boundary for each cut cell. We follow the process illustrated for cut cell (i, j) in Fig. 5.2. In the rest of this section, subscripts specified with greek letters are assumed to range from 1 to n_d, the number of dimensions, and repeated indices of that kind imply the use of the Einstein summation convention. Let x_μ represent a Cartesian coordinate system, and $\hat{x}_\mu$ represent an orthonormal coordinate system with a unit vector pointing normal to the cell interface, and unit vector(s) in the interface tangential plane.

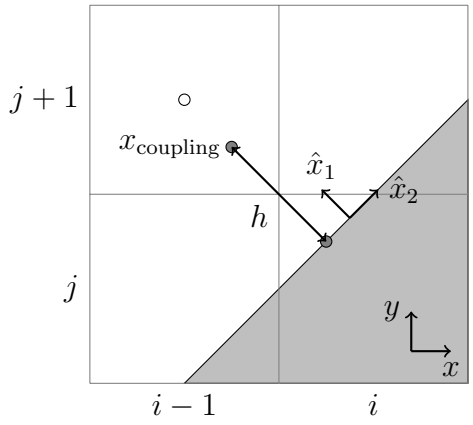


Figure 5.2: Illustration of the procedure used to compute $\nabla \mathbf{u}$ at the boundary of cell (i, j) using the Spalding wall function. u^*_μ at x_{coupling} is reconstructed from the weighted least squares gradient computed at the nearest cell volumetric centroid (filled white circle).

u^*_μ is the velocity at the LES-wall model coupling point x_{coupling} which is located at a distance of h (calculated as per Eq. 4.12) from the interface centroid in the normal direction. We reconstruct the value of u^*_μ using the the inverse distance weighted least squares gradient computed at the cell volumetric centroid closest to x_{interp}. As explained in Section 4.3.1.2, when using a uniform mesh spacing of Δx in all coordinate directions, $h = 0.5\Delta x$. Using a fixed coupling height is important because wall models are known to be sensitive to the location of the first grid point [54], which varies widely from one cell to another in a cut cell mesh.

The first step is to rotate the velocity components u^*_μ at the coupling point to find the components $\hat{u}^*_\mu$ in the normal-tangential frame. The rows of the transformation matrix M^T are the unit vectors of the $\hat{x}_\mu$ coordinate system. These are calculated as described earlier in Section 4.3.1.2. The aim is to compute the tensor $\nabla \mathbf{u}$ at the boundary. It has Cartesian components $\partial u_\mu / \partial x_\nu$ that are given by

$$\frac{\partial u_\mu}{\partial x_\nu} = \frac{\partial \hat{x}_\rho}{\partial x_\nu}\frac{\partial u_\mu}{\partial \hat{x}_\rho} = \frac{\partial \hat{x}_\rho}{\partial x_\nu}\frac{\partial \hat{u}_\sigma}{\partial \hat{x}_\rho}\frac{\partial x_\mu}{\partial \hat{x}_\sigma}. \tag{5.10}$$

For the purposes of implementation, $\frac{\partial \hat{x}_\rho}{\partial x_\nu}$ can be represented by a matrix M whose columns are the unit vectors of the $\hat{x}_\mu$ coordinate system, while $\frac{\partial x_\mu}{\partial \hat{x}_\sigma}$ can be represented by its transpose M^T. What remains is to calculate the partial derivatives $\frac{\partial \hat{u}_\sigma}{\partial \hat{x}_\rho}$. The tangential derivatives are already known to be 0 because of no-slip. The normal derivative of the normal velocity component is computed using finite differences by assuming a simple linear variation as in Eq. 4.11. For each tangential velocity component, the Spalding law Eq. 5.9 can be used to calculate u^+. The corresponding normal derivative then follows from Eq.

5.8.

5.1.4 Method for specifying the cell size

We follow the methodology outlined in [95] to specify the simulation cell size. For y^+ between approximately 30 and 500, the mean velocity profile in a turbulent boundary layer satisfies the logarithmic law of the wall [4]. Beyond this lies the inertia dominated outer layer. In WMLES, the wall model thickness is usually chosen to fall within the log-layer [8, 90]. Given a non-dimensional 'target' wall model coupling height y^+_{target}, and since we use half the cell size as the wall model height (see Section 5.1.3), the required non-dimensional cell size $\Delta y^+ \approx 2y^+_{\text{target}}$.

Hence, the required simulation cell size

$$\Delta y = \frac{\Delta y^+ \nu}{u_\tau} = \Delta y^+ \nu \sqrt{\frac{\rho}{\tau_{\text{wall}}}}. \tag{5.11}$$

Consider the non-dimensional friction coefficient [58],

$$C_f \equiv \frac{\tau_{\text{wall}}}{\frac{1}{2}\rho U^2_\infty}, \tag{5.12}$$

where U^2_∞ is the free stream velocity. Given a value for C_f, Eq. 5.11 can be used to calculate the required cell size.

We estimate C_f for the geometry under consideration by using the formula for the skin friction for the turbulent boundary layer on a flat plate. Several approximation formulas exist for the local skin friction coefficient, and we use the relation

$$\frac{C_f}{2} = \frac{0.037}{Re_L^{1/5}} \tag{5.13}$$

from [95]. Re_L is the Reynolds number based on the characteristic length L of the geometry.

5.2 Results

5.2.1 Turbulent flow over a square cylinder

The problem of $Re_L = 21400$ flow over a square cylinder is used to conduct an initial assessment of the performance of the dimensionally split cut cell scheme when computing a high Reynolds number turbulent flow in conjunction with the Spalding wall function.

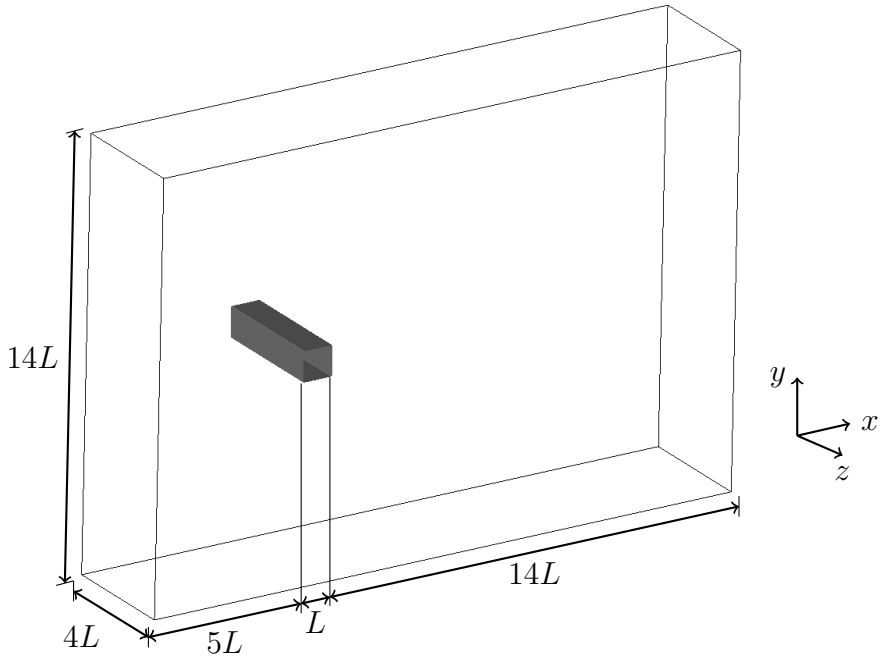
Fig. 5.3a shows the domain extents relative to the cylinder length L. Subsonic inflow conditions are specified at the inlet, while outflow conditions are used for the outlet and cross-streamwise domain boundaries. The spanwise boundaries are periodic. AMR is used to refine the cylinder boundary and near wake such that the mesh on the finest level resolves to $x^+_{\text{target}} = y^+_{\text{target}} = 10$, and $z^+_{\text{target}} = 40$. The boundary cells are all cut cells with those along the top and bottom faces of the cylinder having a volume fraction of 0.143 (3 significant figures). Fig. 5.3b shows a close-up of the AMR mesh and instantaneous vorticity structure captured by the simulation.

The simulation was run for a time of 170 flow units (L/U) where the reference streamwise velocity, U, is set to correspond to $M = 0.1$. Fig. 5.4 shows a comparison of the computed and experimental mean and RMS streamwise velocity profiles at five stations situated along the cylinder surface. Fig. 5.5 shows results at the same stations for the mean and RMS cross-streamwise velocity. The experimental data is taken from the two-component laser-Doppler measurements of Lyn et al. [96]. The agreement between simulation and experiment is good and similar, for example, to that obtained from the LES of Antepara et al. [97].

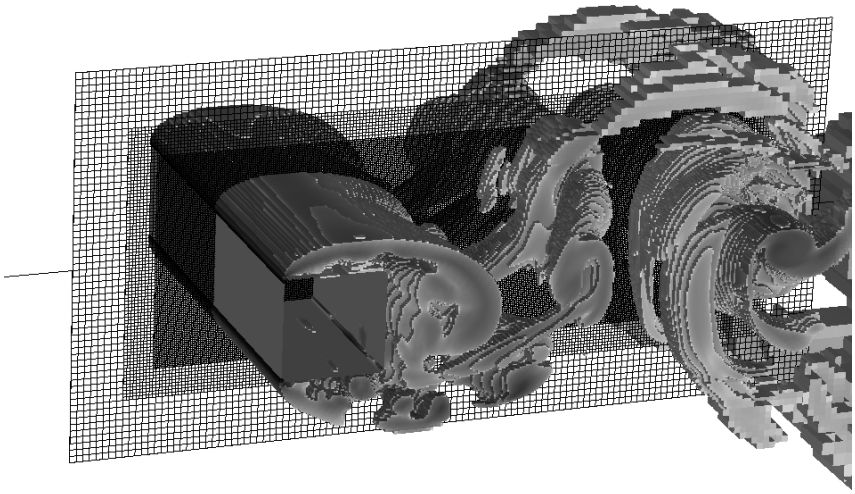
As shown in Fig. 5.6, good agreement with previous investigations is also obtained for the mean streamwise velocity in the cylinder wake. The three other LES results plotted in the figure correspond to one simulation with no explicit subgrid-scale model and a no-slip condition employed at the cylinder boundary, and two simulations where a wall function is used at the boundary but where Dynamic and Smagorinsky subgrid-scale models are used respectively. See Rodi [98] for additional details. The experimental results [96] of Lyn et al. and Durao et al. are also shown.

Finally, Table 5.1 shows a comparison of the computed force coefficients and Strouhal number with previous LES and experimental studies. The frequency of the vortex shedding is calculated from the Fourier Transform of the computed pressure at a point in the middle of the wake located a distance of $2.5L$ behind the cylinder. The computed results are within the range determined by previous studies.

These results demonstrate the possibility of using the cut cell method to compute a turbulent flow.

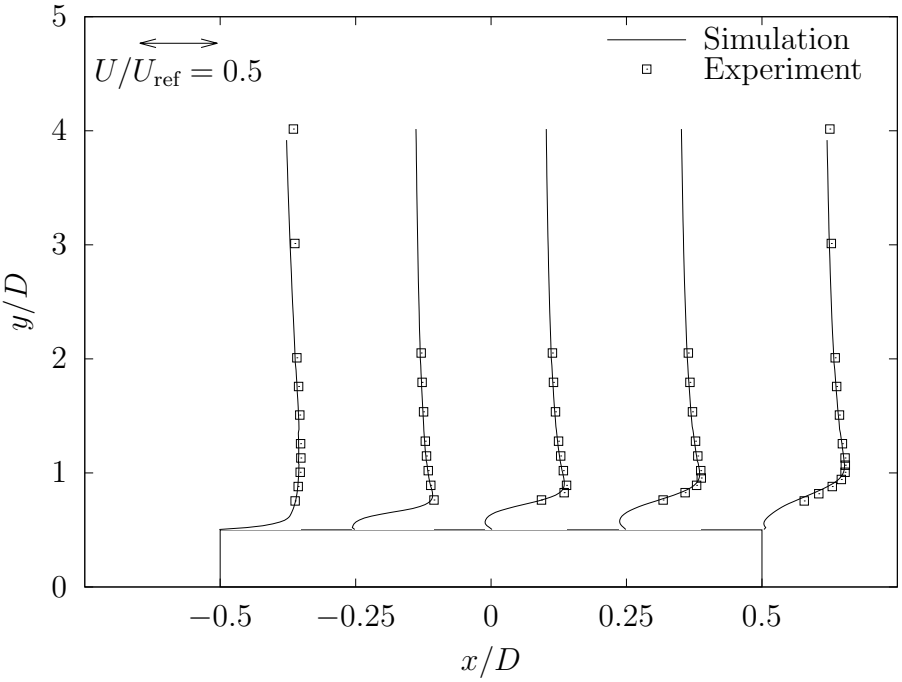


(a) Illustration of the domain dimensions used. The square cylinder is of side length L.

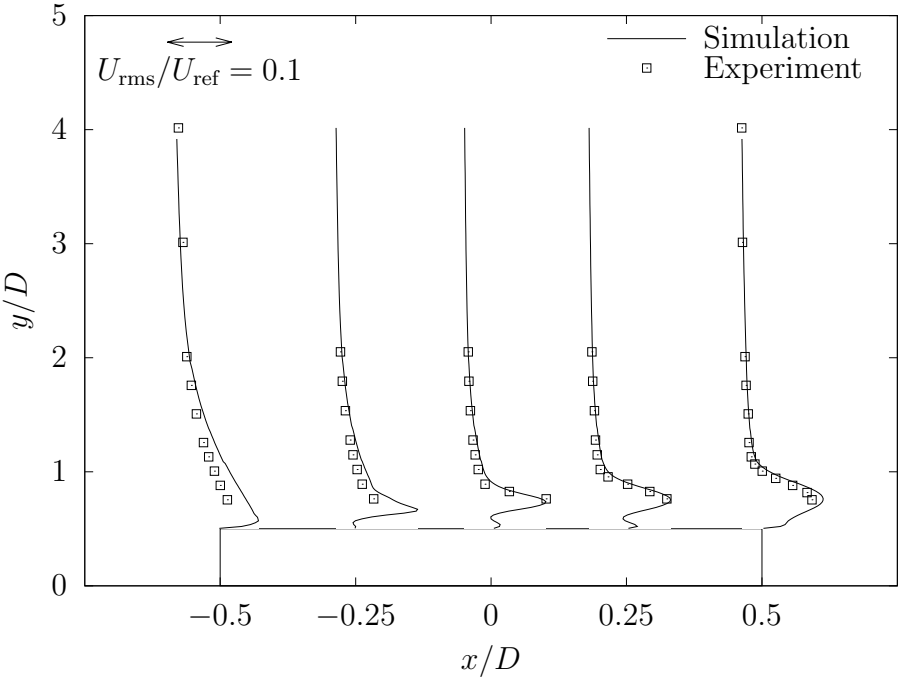


(b) Instantaneous snapshot of vortices in the wake identified using the Q-criterion [99].

Figure 5.3: Domain dimensions and a visualisation of the instantaneous simulation results for the turbulent square cylinder problem.

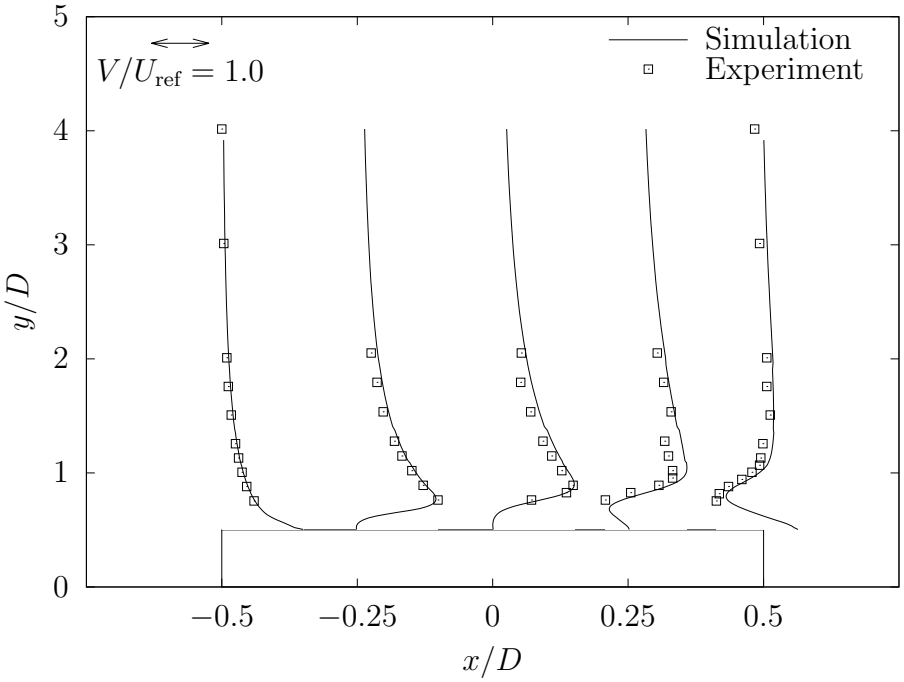


(a) Mean streamwise velocity profiles.

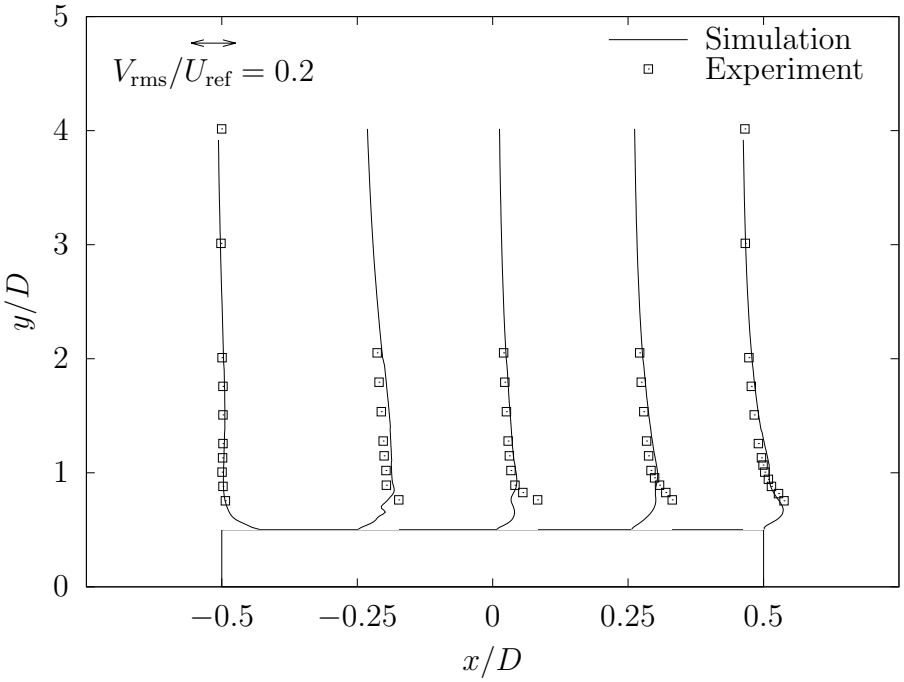


(b) RMS streamwise velocity profiles.

Figure 5.4: Comparison of numerical and experimental [96] mean and RMS streamwise velocity profiles at five stations along the geometry centreline.



(a) Mean cross-streamwise velocity profiles.



(b) RMS cross-streamwise velocity profiles.

Figure 5.5: Comparison of numerical and experimental [96] mean and RMS cross-streamwise velocity profiles at five stations along the geometry centreline.

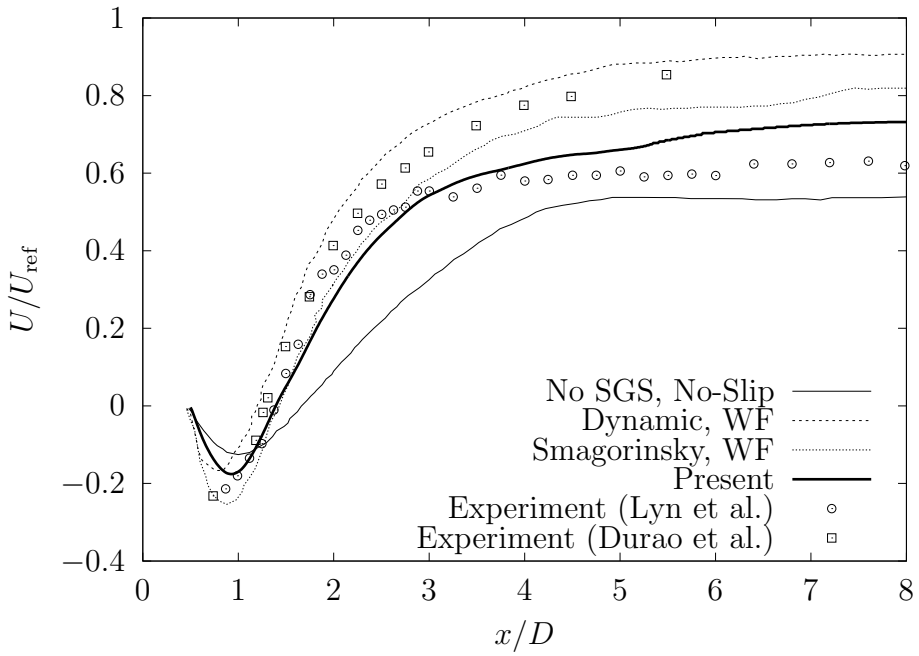


Figure 5.6: Comparison of the computed mean streamwise velocity profile behind the cylinder with previous numerical [98] and experimental [96] studies.

Table 5.1: Comparison of the computed mean and rms flow parameters with previous numerical [98] and experimental [100] studies.

Study	$C_{L,\text{mean}}$	$C_{L,\text{rms}}$	$C_{D,\text{mean}}$	$C_{D,\text{rms}}$	St
Present work	-0.0460	1.07	1.88	0.259	0.130
Experiment	-	0.68-1.4	1.9-2.1	0.1-0.23	0.132
No SGS, No-Slip	-0.005	1.33	2.58	0.27	0.15
Dynamic, WF	0.0	-	2.02	-	0.09
Smagorinsky, WF	-0.04	1.15	2.30	0.14	0.13

5.2.2 SAE Notchback results

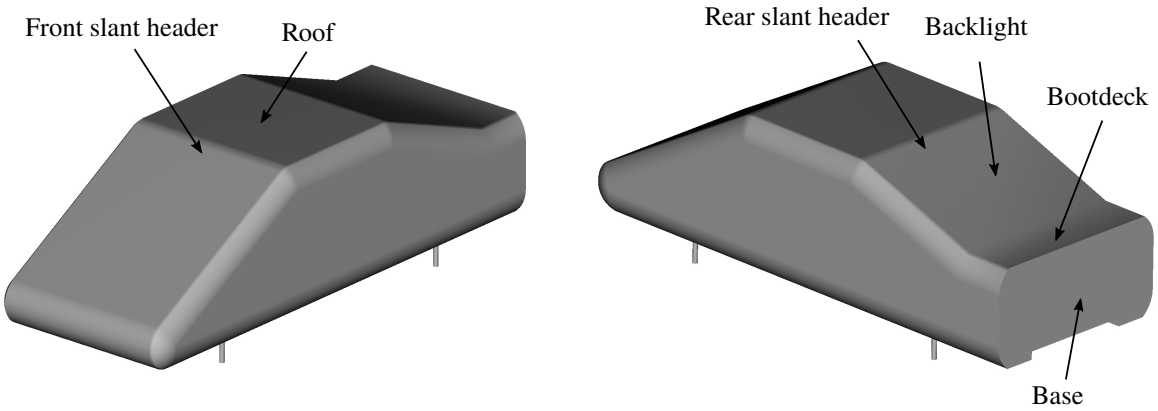


Figure 5.7: Geometric features of interest for the SAE Notchback geometry.

The SAE Notchback geometry, like the Ahmed body, is a simplified automotive reference model. Fig. 5.7 illustrates its geometric features of interest. Unlike the Ahmed body, however, the SAE body exhibits a more realistic three-dimensional flow pattern which includes local backlight/bootdeck separation, making it a more relevant geometry to evaluate the performance of a numerical method with [70]. The relative geometrical simplicity of the model, however, means it does not necessarily challenge the mesh generation capability of the Cartesian cut cell method. Experimental data for this test was obtained from the Loughborough University Institutional Repository [101], which also contains a 'watertight' STL CAD file of the geometry. As described at the beginning of this chapter, Islam and Thornber have previously studied this test case using an ILES/DES approach and a body-fitted structured mesh.

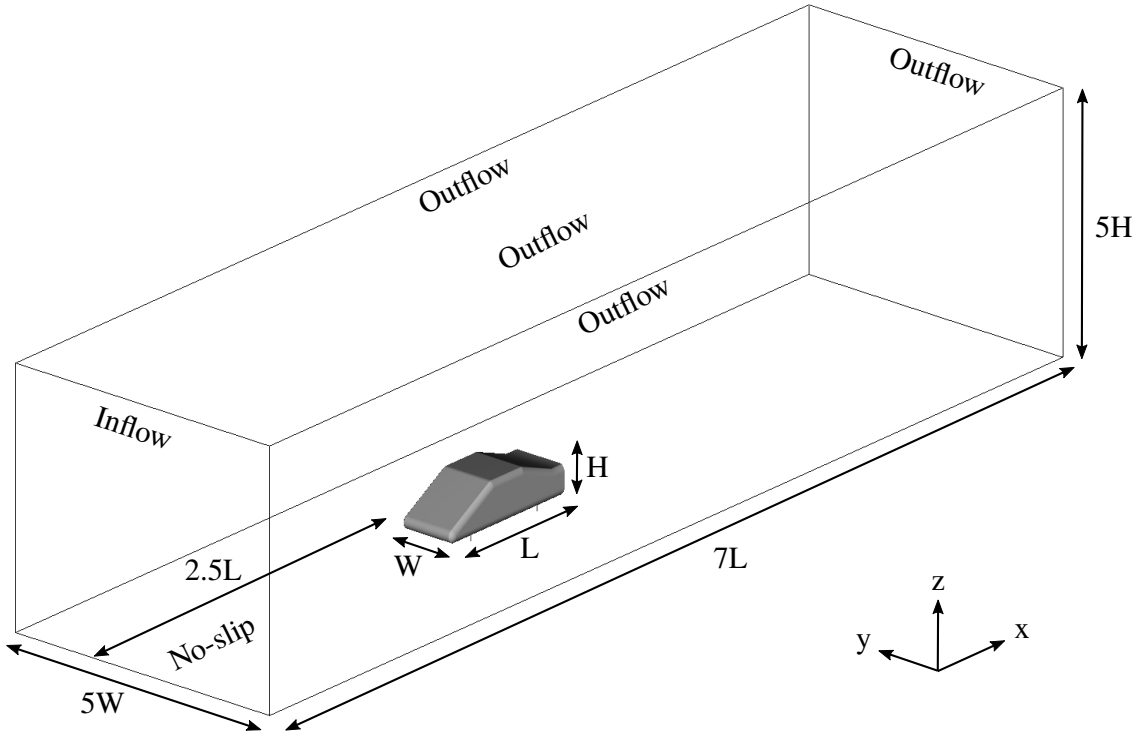


Figure 5.8: Simulation set-up for the SAE Notchback test case.

In our simulation, the geometry with length L, width W and height H is positioned at a distance of $2.5L$ from the inlet in a domain of size $7L \times 5W \times 5H$. Fig. 5.8 illustrates the simulation set-up. The inlet is specified to be a subsonic inflow boundary with the streamwise velocity u specified as following the power law

$$u = U_\infty \min\left(\left(\frac{y}{\delta}\right)^{1/7}, 1\right), \tag{5.14}$$

as in [69]. In Eq. 5.14, the free stream velocity $U_\infty = 40$ m/s as in the experiment, y is the height above the ground plane, and the boundary layer height δ is specified to be 60 mm to be consistent with experiment. The ground-plane is no-slip, while all remaining boundaries are specified as subsonic outflow boundaries. The free stream density ρ_∞ and pressure p_∞ are set to be 1.225 kg/m^3 and 101325 Pa respectively.

AMR is used to to refine the boundary and near wake region such that the mesh on the finest level resolves to $x^+_{\text{target}} = y^+_{\text{target}} = z^+_{\text{target}} = 150$, yielding a mesh with approximately 3.5 million cells. Note that the finest mesh used in the study of Islam and Thornber contains approximately 7.5 million cells.

Fig. 5.9 shows a comparison of the computed and experimental mean pressure distributions in the symmetry plane at the top of the geometry. The pressure reaches a local

minimum at the front slant header as the flow accelerates away from the stagnation point. Pressure recovery occurs on the roof before decreasing again due to the acceleration caused by the curvature at the rear slant header. As described by Wood et al. [70], the flow separates near the top of the backlight, although pressure recovery continues, reaching a local maximum at the bootdeck where the flow re-attaches and stagnates. The simulation appears to be able to capture the complicated flow pattern and the results show good agreement with experiment.

Surface contour plots of the computed and experimental pressure distributions on the backlight, bootdeck and base are shown in Fig. 5.10a. Note that the experimental contours are generated using interpolation from the data points which are denoted by black markers in Fig. 5.10b. The simulation appears able to produce the general trends of the flow measured in the experiment. In particular, the adverse pressure gradient from the notch to the middle of the rear slant which is reponsible for flow separation is captured well. The mean pressure on the vertical 'base' of the geometry appears to be slightly higher than experiment, however.

Given the higher base pressure, the computed mean drag coefficient underestimates the experimental value by $\approx 14\%$, as shown in Table 5.2. It may be noted, however, that Islam and Thornber, who use a higher order solver (fifth order as apposed to second order as we do) and roughly twice the number of cells as us overestimate the drag by a similar amount using the DES approach. The computed mean lift coefficient is similar to that obtained by Islam and Thornber, but in error by almost 24% when compared to experiment. This is likely to be due to errors in the pressure computed at the bottom of the geometry, although there is unfortunately no measured experimental data in that region to compare with. The sensitivity of the lift coefficient to the nature of the flow under the geometry is seen by the fact that when we ran a simulation with a a uniform inlet velocity (as opposed to the power law of Eq. 5.14), the computed mean lift coefficient was -0.080 which is in error by a larger amount ($\approx 45\%$). It is expected that the error to experiment can be reduced by using additional resolution, particularly in the region below the car, and by employing the Spalding wall function to compute the shear stress also on the ground plane. Investigating the effect of these measures is beyond the scope of this preliminary assessment.

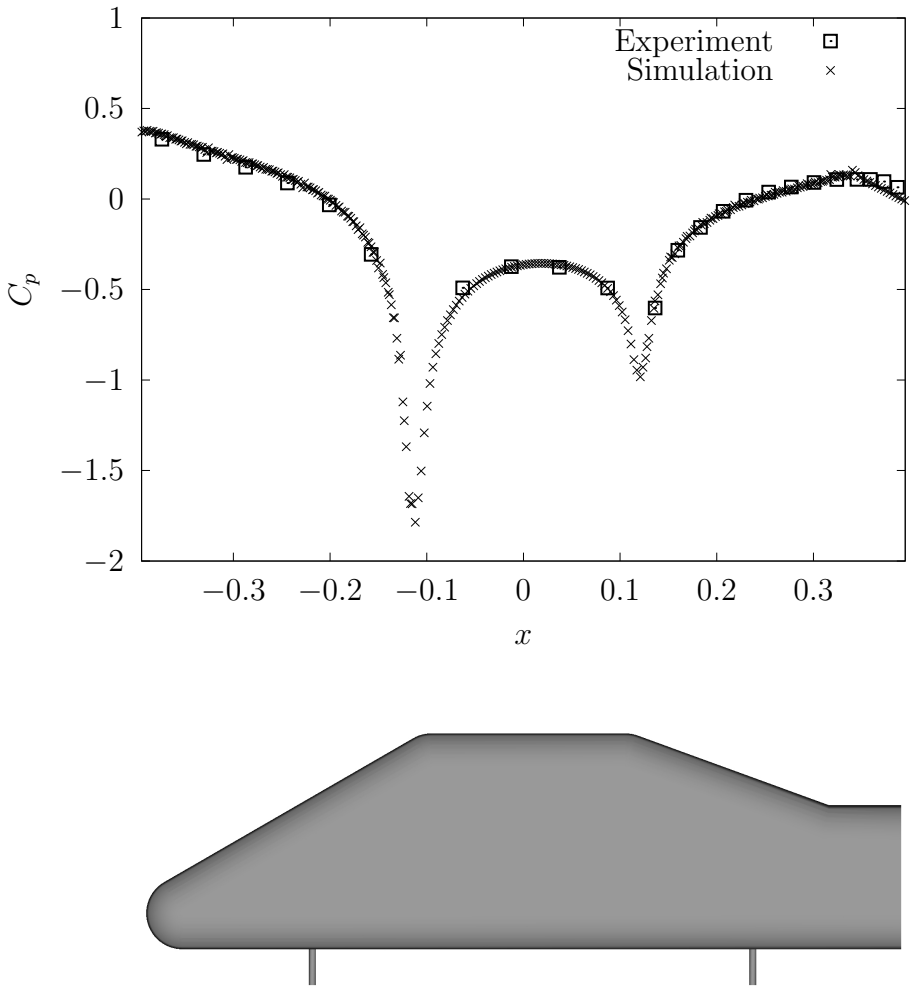
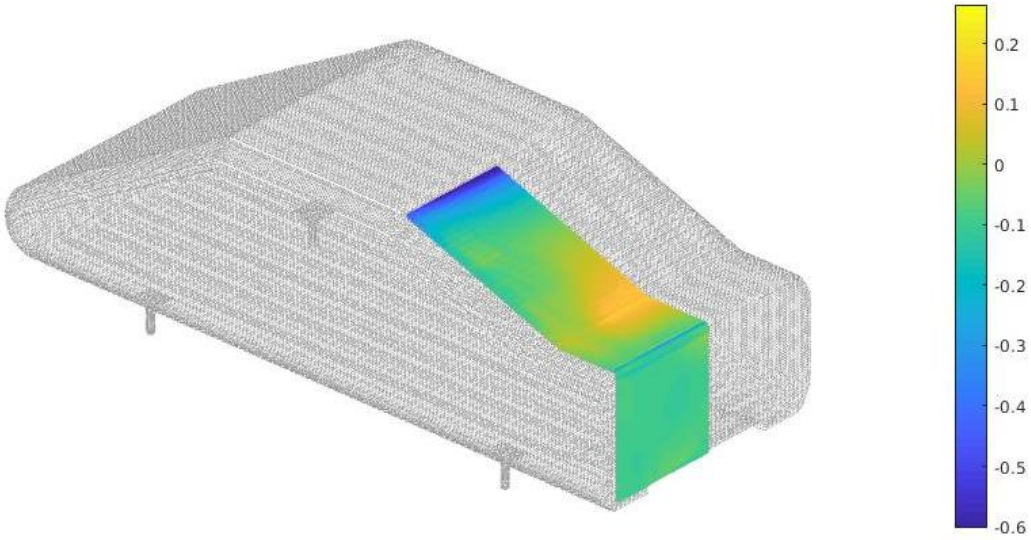
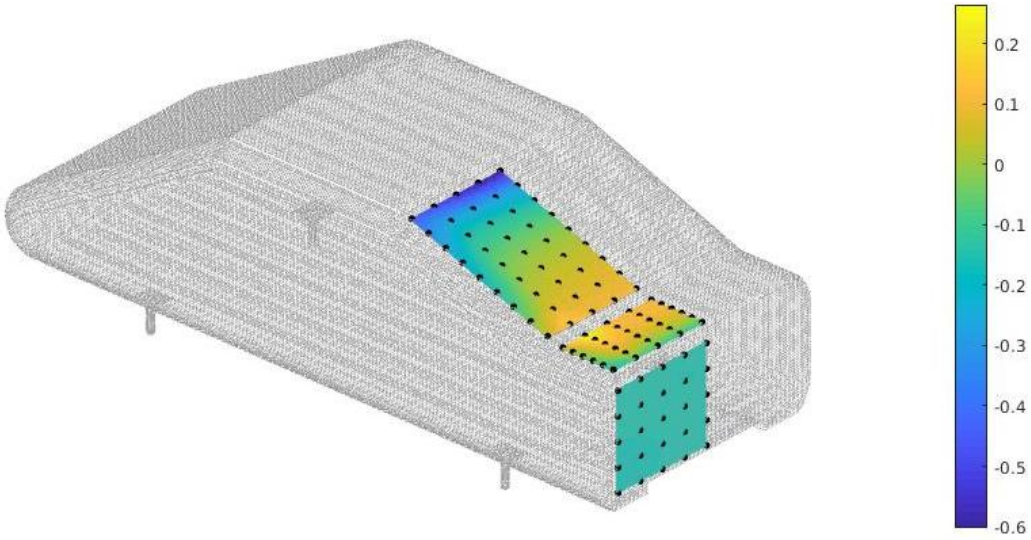


Figure 5.9: Mean pressure distributions in the symmetry plane at the top of the SAE model.



(a) Simulation.



(b) Experiment [70].

Figure 5.10: Mean surface pressure distributions on the rear of the SAE model.

Table 5.2: Comparison of the computed mean lift and drag coeffients with previous experimental and numerical studies.

Study	$\overline{C_L}$	$\overline{C_D}$
Present work	-0.068	0.18
Experiment [70]	-0.055	0.21
Islam and Thornber (Simulation) [69]	-0.064	0.24

5.2.3 DrivAer results

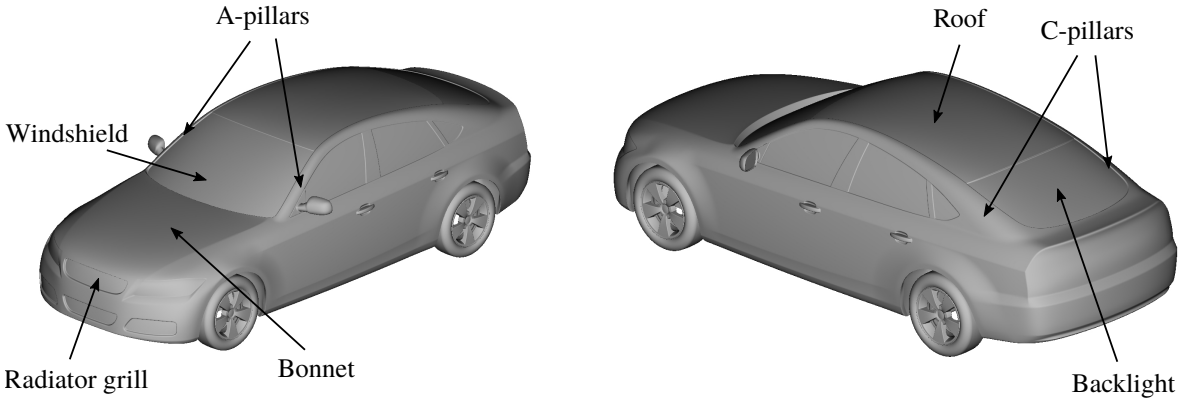


Figure 5.11: Geometric features of interest for the DrivAer geometry.

The DrivAer model is a more realistic automotive reference geometry developed by the Technical University of Munich (TUM) in conjunction with two major automotive companies - Audi AG and BMW [7]. The geometry can be constructed in 3 configurations - Estate, Fastback and Notchback, on top of which additional customisations with respect to the wheels, wing mirrors, and underbody can be made. Watertight STL files for all the components can be obtained from TUM [102]. We focus on using the Fastback configuration with static wheels, wing mirrors, and a smooth underbody configuration. Fig. 5.11 illustrates its geometric features of interest. A close-up of our generated cut cell grid was shown in Fig. 1.1.

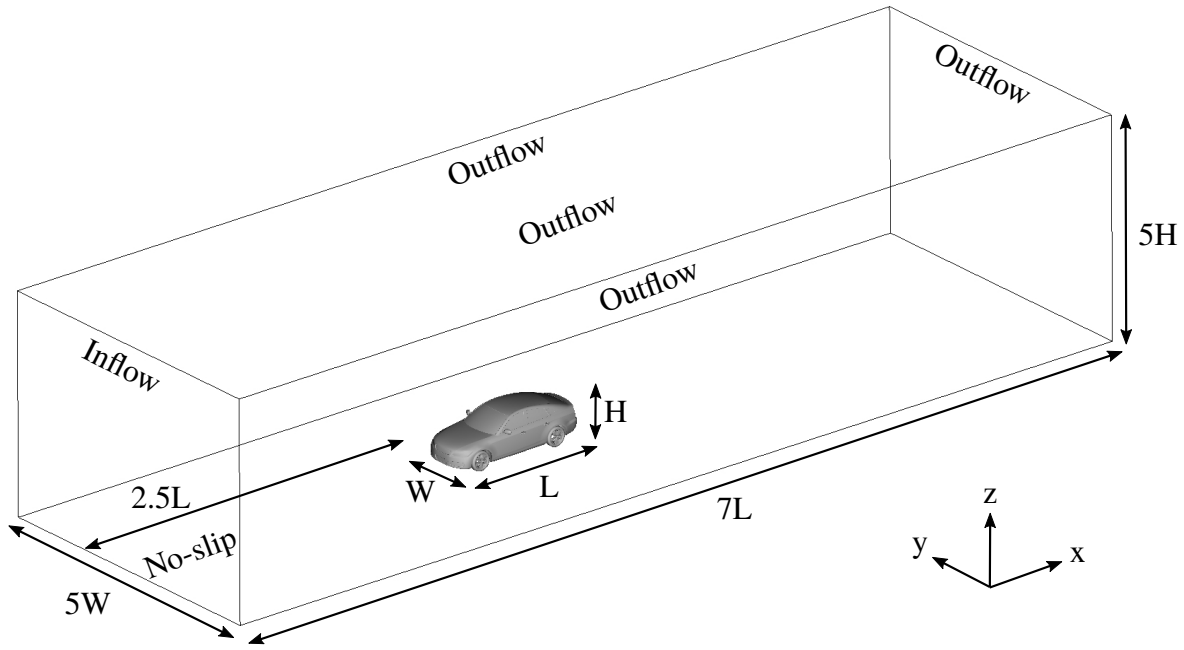


Figure 5.12: Simulation set-up for the DrivAer test case.

In our simulation, the geometry with length L, width W and height H is positioned at a distance of $2.5L$ from the inlet in a domain of size $7L \times 5W \times 5H$. Fig. 5.12 illustrates the simulation set-up. The inlet is specified to be a subsonic inflow boundary with uniform velocity $U_\infty = 40$ m/s as in the experiment. The ground-plane is no-slip, while all other boundaries are specified as subsonic outflow boundaries. The free stream density ρ_∞ and pressure p_∞ are set to be 1.225 kg/m^3 and 101325 Pa respectively. The Reynolds number based on the model height H is 1.48×10^6 as in Ashton et al.

AMR is used to to refine the boundary and near wake region such that the mesh on the finest level resolves to $x^+_{\text{target}} = y^+_{\text{target}} = z^+_{\text{target}} = 150$, yielding a mesh with approximately 23.3 million cells. Note that Aljure et al. used approximately 59 million cells in their recent incompressible WMLES study of this geometry, while Ashton et al. used approximately 100 million cells in their study which used IDDES. The current resolution, although on the coarser side, is sufficient for the purposes of this initial assessment.

Fig. 5.13 shows a comparison of the computed and experimental mean pressure distributions in the symmetry plane at the top of the geometry. Starting at the stagnation point, the flow accelerates rapidly over the radiator grill, producing a local minimum, before rising at a steady rate over the bonnet. A local maximum is produced at the bonnet/windshield intersection, after which flow acceleration over the windshield leads to pressure reduction once again. Starting from the roof of the geometry, pressure recovery begins to take place, and the pressure reaches almost ambient levels behind the

car. Two major areas of discrepancy between simulation and experiment appear at the windshield/roof intersection, and in the region of pressure recovery over the roof. As identified by Heft et al. [103] and Aljure et al., these are most likely caused by the presence of the top retaining strut which is set up in the wind tunnel but not in CFD, as well as by differences in the experimental and CAD models in that region. In general, however, the agreement between simulation and experiment is good, and similar to that obtained by Ashton et al. Results from the incompressible WMLES of Aljure et al. show similar trends, and for the sake of clarity of presentation, we do not plot them in Fig. 5.13.

Fig. 5.14 and Fig. 5.15 show a comparison of the simulated and experimental mean surface pressure distributions on the windshield and rear window of the geometry respectively. Note that the visual appearance of the experimental contours should be interpreted cautiously as, particularly at the rear window, they have been produced using interpolation from a fairly sparse set of measurement points which are denoted by black markers in Fig. 5.14b and Fig. 5.15b.

Good agreement is observed between the computed and experimental surface pressures on the windshield. The stagnation region near the base of the windshield is clearly visible, as is the pressure reduction caused by acceleration of the flow towards the roof and A-pillar. On the rear window, the simulation captures the region of low pressure near the C-pillar caused by the C-pillar vortex, as well as the region of higher pressure on the lower part of the backlight which shows the presence of flow reattachment [7].

Finally, as observed in Table 5.3, the computed mean drag coefficient is in very good agreement with experiment. This is to be expected given the agreement that was shown by the surface pressure results. Because Aljure et al. use a moving ground simulation in their incompressible WMLES study, their computed drag coefficient is not directly comparable with ours and we do not mention it in the table.

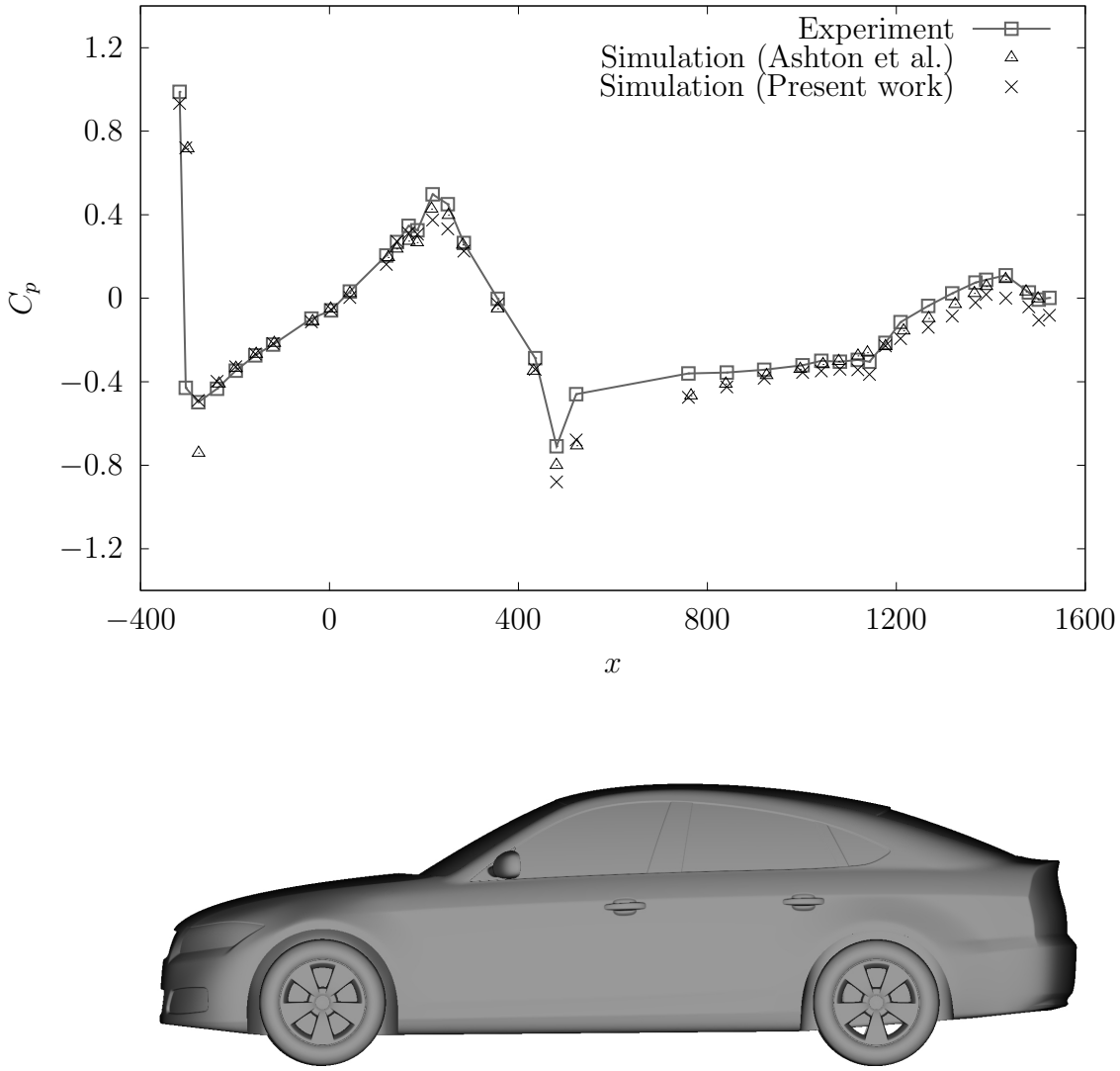
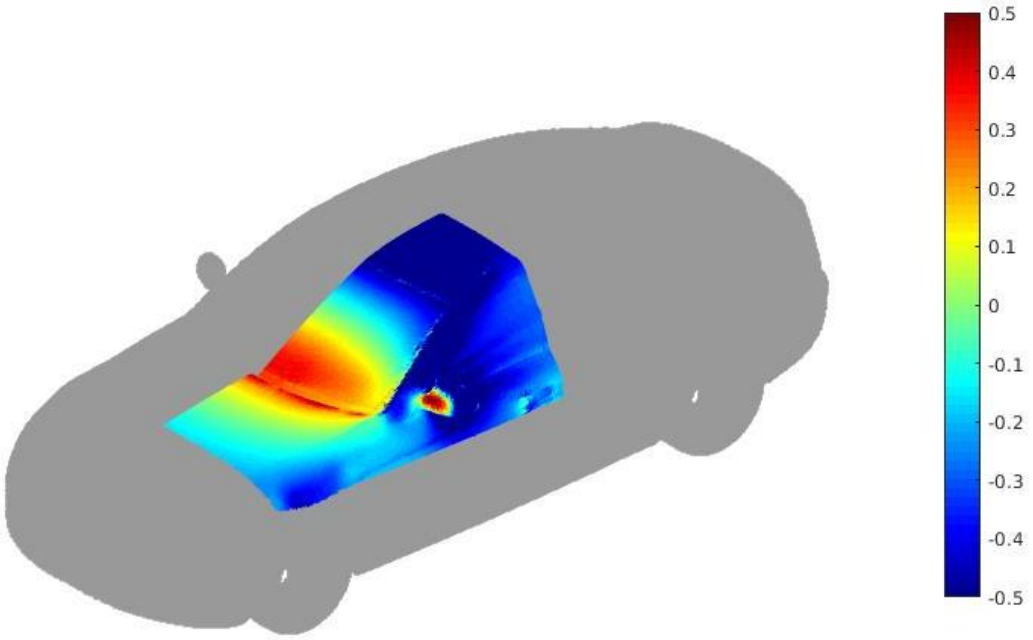
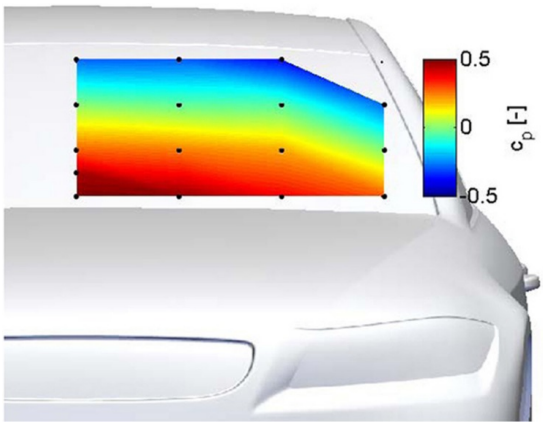


Figure 5.13: Mean pressure distributions in the symmetry plane at the top of the DrivAer model.

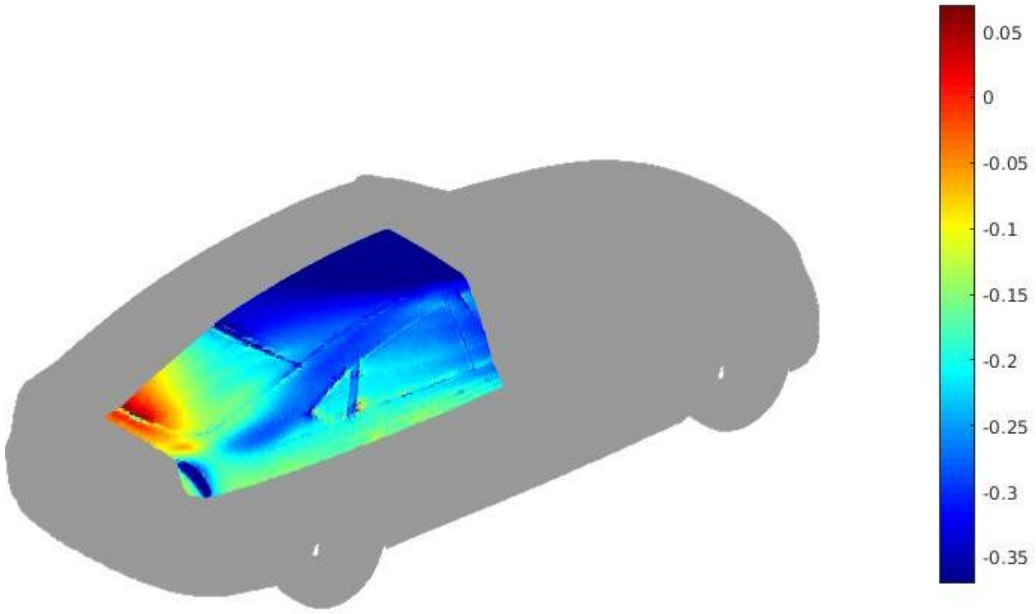


(a) Simulation.

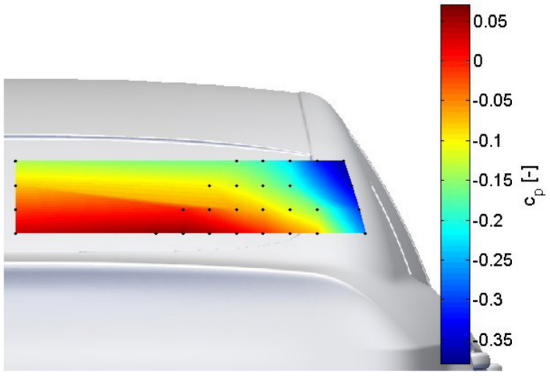


(b) Experiment [7].

Figure 5.14: Mean surface pressure distributions on the windscreen of the DrivAer model.



(a) Simulation.



(b) Experiment [7].

Figure 5.15: Mean surface pressure distributions on the rear window of the DrivAer model.

Table 5.3: Comparison of the computed mean drag coeffient with previous experimental and numerical studies.

Study	$\overline{C_D}$
Present work	0.252
Experiment [7]	0.254
Ashton et al. (SST IDDES)	0.262

5.3 Conclusions

In this chapter, we presented an extension of the dimensionally split Cartesian cut cell method to solve high Reynolds number turbulent automotive flows using an WMLES approach. The high resolution MUSCL-Hancock scheme provided an implicit LES model, while the Spalding equilibrium wall function was used to compute the shear stress at the wall.

For the $Re_L = 21400$ turbulent flow over a square cylinder test problem, the methodology produced results for velocities and force coefficients which were in very good agreement with previous experimental and numerical results. Following this initial validation of the methodology, it was used to compute results for the SAE Notchback and DrivAer fastback automotive reference geometries. The latter model is a realistic and relatively complex geometry which serves as a good test for the Cartesian cut cell mesh generation approach. For both geometries, the computed surface pressure distributions and force coefficients compared well to previous experimental and numerical studies.

We intend to publish the results of this chapter as part of a future publication, which would be the first assessment of a WMLES Cartesian cut cell approach for computing automotive flows to be presented in the literature. Although we have obtained promising initial results, additional work is needed to make the material publication-ready. A mesh resolution study needs to be performed to check whether we have obtained approximate mesh independence in the computed results. In addition, it would also be useful to investigate the effect of using non-uniform meshes on the results. For an automotive flow, for example, it is expected that accuracy will not be affected greatly by sacrificing spanwise resolution in favour of streamwise and cross-streamwise resolution. Finally, it would also be desirable to further assess the quality of the unsteady results for the automotive simulations by examining the RMS results in addition to the mean results.

CHAPTER 6

CONCLUSIONS

Cartesian cut cell mesh generation is practically attractive because of the ease with which meshes can be automatically created around complex geometries. The downside of the approach, however, is that it leads to the creation of arbitrarily small boundary cut cells which places a severe restriction on the stable time-step for an explicit numerical scheme.

Since the early 1980s, a number of solutions to this 'small cell problem' have been developed, and they have been implemented largely in an unsplit fashion. In this work, we presented a novel dimensionally split Cartesian cut cell method to compute inviscid, viscous and turbulent flows around rigid geometries. Dimensional splitting is a convenient way to extend one-dimensional methods to solve multi-dimensional problems.

In Chapter 2, we presented a robust cut cell mesh generation technique that uses the values of the signed distance function at each cell vertex to determine the cut cell geometric parameters. The technique assumes that sufficient resolution has been used to ensure that all cells are singly cut.

In Chapter 3, we presented the novel dimensionally split LPFS cut cell method that solves the small cell problem when computing solutions to hyperbolic conservation laws. After proving the numerical properties of the method for the model linear advection equation, we demonstrated its practical performance by computing solutions to a number of challenging multi-dimensional problems for the Euler equations.

In Chapter 4, we developed the method further to compute solutions for compressible Navier-Stokes problems, and demonstrated the numerical performance of the method by computing solutions to a wide range of test problems ranging from the nearly incompressible to the highly compressible flow regimes. A dimensionally split Cartesian cut cell method for the compressible Navier-Stokes equations has not been previously presented in the

literature.

Finally, in Chapter 5, we extended the method to compute high Reynolds number automotive flows via a wall-modelled Large Eddy Simulation (WMLES) approach. A full description was provided of the coupling between the implicit LES solution and the equilibrium wall function which provides the wall shear stress. The combined methodology was used to compute solutions for turbulent flow over a square cylinder, and for flow over the SAE notchback and DrivAer reference geometries. The results of this preliminary assessment of the technique showed good agreement with previous experimental and numerical studies. A WMLES Cartesian cut cell technique has not previously been presented for computing automotive flows in the literature.

Although we have demonstrated the robustness and readiness of the cut cell method for use in tackling a wide range of practical problems, there is room to develop it further. On the numerical side, there are two major areas where further research is needed. To start with, the method as it stands is only first order accurate at the boundary. Extending it to maintain second order accuracy in the cut cells would be very beneficial. Another useful contribution would be to develop an alternative to the mixing flux in doubly-shielded regions that avoids any need for a post-sweep conservative correction procedure. The development of such a flux in a dimensionally split framework is a challenging research problem. On the mesh generation side, the technique of Chapter 2 as it stands is unable to deal with multiply cut cells. Such cells can be difficult to avoid in three-dimensional simulations, and it would be important to resolve them by using, for example, the 'Marching Cubes' based approach as in Gunther et al. [13].

A number of interesting open questions may also be identified. How does the accuracy of the split method compare to previously published unsplit methods when using the same resolution? In practice, how much difference does second order accuracy at the boundary make when one is using AMR to refine the interface? Can the method be implemented in conjunction with dual time stepping [104]? The larger time steps that the dual time stepping approach permits could make the computation of the subsonic automotive flows of Chapter 5 more computationally efficient.

Clearly, there is no shortage of questions and research topics that need further investigation. As noted by Berger [20], "cut cells promise to be an exciting research area for years to come".

BIBLIOGRAPHY

[1] N. Gokhale, N. Nikiforakis, and R. Klein, "A dimensionally split Cartesian cut cell method for hyperbolic conservation laws," *Journal of Computational Physics*, vol. 364, pp. 186–208, 2018.

[2] N. Gokhale, N. Nikiforakis, and R. Klein, "A dimensionally split Cartesian cut cell method for the compressible Navier-Stokes equations," *Journal of Computational Physics*, vol. 375, pp. 1205–1219, 2018.

[3] M. J. Berger and J. Oliger, "Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations," *Journal of Computational Physics*, vol. 53, no. 3, pp. 484–512, 1984.

[4] H. K. Versteeg and W. Malalasekera, *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*. Pearson Education, 2007.

[5] J. H. Ferziger and M. Peric, *Computational Methods for Fluid Dynamics*. Springer Science & Business Media, 2012.

[6] R. Mittal and G. Iaccarino, "Immersed Boundary Methods," *Annu. Rev. Fluid Mech.*, vol. 37, pp. 239–261, 2005.

[7] A. I. Heft, T. Indinger, and N. A. Adams, "Introduction of a New Realistic Generic Car Model for Aerodynamic Investigations," *SAE 2012 World Congress*, vol. April 23-26, 2012, Detroit, Michigan, USA, Paper 2012-01-1068.

[8] F. F. Grinstein, L. G. Margolin, and W. J. Rider, *Implicit Large Eddy Simulation: Computing Turbulent Fluid Dynamics*. Cambridge University Press, 2007.

[9] E. F. Toro, *Riemann solvers and numerical methods for fluid dynamics: a practical introduction*. Springer Science & Business Media, 2013.

[10] M. Berger and I. Rigoutsos, "An algorithm for point clustering and grid generation," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, no. 5, pp. 1278–1286, 1991.

[11] S. Mauch, "A fast algorithm for computing the closest point and distance transform," Tech. Rep. caltechASCI/2000.077, California Institute of Technology, 2000.

[12] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," in *ACM Siggraph Computer Graphics*, vol. 21, pp. 163–169, ACM, 1987.

[13] C. Günther, D. Hartmann, M. Meinke, and W. Schröder, "A Cartesian cut-cell method for sharp moving boundaries," in *20th AIAA Computational Fluid Dynamics Conference*, AIAA Paper 2011-3387, Jun 2011.

[14] E. W. Weisstein, "Polygon Area." From MathWorld–A Wolfram Web Resource. URL `http://mathworld.wolfram.com/PolygonArea.html`. Accessed: 20-01-2019.

[15] R. B. Pember, J. B. Bell, P. Colella, W. Y. Curtchfield, and M. L. Welcome, "An adaptive Cartesian grid method for unsteady compressible flow in irregular regions," *Journal of Computational Physics*, vol. 120, no. 2, pp. 278–304, 1995.

[16] R. N. Goldman, "Area of Planar Polygons and Volume of Polyhedra," *Graphics Gems II*, pp. 170–171, 1991.

[17] Z. Wang, "Improved formulation for geometric properties of arbitrary polyhedra," *AIAA Journal*, vol. 37, no. 10, pp. 1326–1327, 1999.

[18] B. Paul, "Calculating The Area And Centroid Of A Polygon)." URL `http://www.seas.upenn.edu/~sys502/extra_materials/Polygon%20Area%20and%20Centroid.pdf`. Accessed: 16-02-2018.

[19] B. Gehrels, B. Lalande, M. Loskot, A. Wulkiewicz, M. Karavelas, and V. Fisikopoulos, "Boost Geometry Documentation." URL `http://www.boost.org/doc/libs/1_66_0/libs/geometry/doc/html/index.html`. Accessed: 28-03-2018.

[20] M. Berger, "Cut Cells: Meshes and Solvers," in *Handbook of Numerical Analysis*, vol. 18, pp. 1–22, Elsevier, 2017.

[21] D. K. Clarke, H. Hassan, and M. Salas, "Euler calculations for multielement airfoils using cartesian grids," *AIAA Journal*, vol. 24, no. 3, pp. 353–358, 1986.

[22] J. J. Quirk, "An alternative to unstructured grids for computing gas dynamic flows around arbitrarily complex two-dimensional bodies," *Computers & Fluids*, vol. 23, no. 1, pp. 125–142, 1994.

[23] M. Kirkpatrick, S. Armfield, and J. Kent, "A representation of curved boundaries for the solution of the Navier–Stokes equations on a staggered three-dimensional Cartesian grid," *Journal of Computational Physics*, vol. 184, no. 1, pp. 1–36, 2003.

[24] D. Hartmann, M. Meinke, and W. Schröder, "A strictly conservative Cartesian cut-cell method for compressible viscous flows on adaptive grids," *Computer Methods in Applied Mechanics and Engineering*, vol. 200, no. 9, pp. 1038–1052, 2011.

[25] L. Schneiders, D. Hartmann, M. Meinke, and W. Schröder, "An accurate moving boundary formulation in cut-cell methods," *Journal of Computational Physics*, vol. 235, pp. 786–809, 2013.

[26] M. Meinke, L. Schneiders, C. Günther, and W. Schröder, "A cut-cell method for sharp moving boundaries in Cartesian grids," *Computers & Fluids*, vol. 85, pp. 135–142, 2013.

[27] M. Berger and C. Helzel, "A simplified h-box method for embedded boundary grids," *SIAM Journal on Scientific Computing*, vol. 34, no. 2, pp. A861–A888, 2012.

[28] P. Colella, D. T. Graves, B. J. Keen, and D. Modiano, "A Cartesian grid embedded boundary method for hyperbolic conservation laws," *Journal of Computational Physics*, vol. 211, no. 1, pp. 347–366, 2006.

[29] X. Hu, B. Khoo, N. A. Adams, and F. Huang, "A conservative interface method for compressible flows," *Journal of Computational Physics*, vol. 219, no. 2, pp. 553–578, 2006.

[30] M. Grilli, P. J. Schmid, S. Hickel, and N. A. Adams, "Analysis of unsteady behaviour in shockwave turbulent boundary layer interaction," *Journal of Fluid Mechanics*, vol. 700, pp. 16–28, 2012.

[31] V. Pasquariello, G. Hammerl, F. Örley, S. Hickel, C. Danowski, A. Popp, W. A. Wall, and N. A. Adams, "A cut-cell finite volume–finite element coupling approach for

fluid–structure interaction in compressible flow," *Journal of Computational Physics*, vol. 307, pp. 670–695, 2016.

[32] B. Muralidharan and S. Menon, "A high-order adaptive cartesian cut-cell method for simulation of compressible viscous flow over immersed bodies," *Journal of Computational Physics*, vol. 321, pp. 342–368, 2016.

[33] S. Tan and C.-W. Shu, "Inverse Lax-Wendroff procedure for numerical boundary conditions of conservation laws," *Journal of Computational Physics*, vol. 229, no. 21, pp. 8144–8166, 2010.

[34] S. Tan, C. Wang, C.-W. Shu, and J. Ning, "Efficient implementation of high order inverse Lax-Wendroff boundary treatment for conservation laws," *Journal of Computational Physics*, vol. 231, no. 6, pp. 2510–2527, 2012.

[35] S. Jebens, O. Knoth, and R. Weiner, "Linearly implicit peer methods for the compressible Euler equations," *Applied Numerical Mathematics*, vol. 62, no. 10, pp. 1380–1392, 2012.

[36] S. May and M. Berger, "A Mixed Explicit Implicit Time Stepping Scheme for Cartesian Embedded Boundary Meshes," in *Finite Volumes for Complex Applications VII-Methods and Theoretical Aspects*, pp. 393–400, Springer, 2014.

[37] R. Klein, K. Bates, and N. Nikiforakis, "Well-balanced compressible cut-cell simulation of atmospheric flow," *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 367, no. 1907, pp. 4559–4575, 2009.

[38] G. Strang, "On the construction and comparison of difference schemes," *SIAM Journal on Numerical Analysis*, vol. 5, no. 3, pp. 506–517, 1968.

[39] R. J. LeVeque, *Finite Volume Methods for Hyperbolic Problems*, vol. 31. Cambridge University Press, 2002.

[40] C. B. Laney, *Computational Gasdynamics*. Cambridge University Press, 2008.

[41] P. D. Lax and R. D. Richtmyer, "Survey of the Stability of Linear Finite Difference Equations," *Communications on Pure and Applied Mathematics*, vol. 9, no. 2, pp. 267–293, 1956.

[42] M. J. Berger, C. Helzel, and R. J. LeVeque, "H-box methods for the approximation of hyperbolic conservation laws on irregular grids," *SIAM Journal on Numerical Analysis*, vol. 41, no. 3, pp. 893–918, 2003.

[43] D. Serre, "Matrices: Theory and Applications," *Graduate Texts in Mathematics*, 2002.

[44] M. Meyer, A. Devesa, S. Hickel, X. Hu, and N. Adams, "A conservative immersed interface method for large-eddy simulation of incompressible flows," *Journal of Computational Physics*, vol. 229, no. 18, pp. 6300–6317, 2010.

[45] J. D. Anderson Jr, *Fundamentals of Aerodynamics*. McGraw-Hill, 2010.

[46] C. D. Harris, "Two-dimensional aerodynamic characteristics of the NACA 0012 airfoil in the Langley 8 foot transonic pressure tunnel," Tech. Rep. 81927, NASA, 1981.

[47] G. Ben-Dor, J. Dewey, and K. Takayama, "The reflection of a plane shock wave over a double wedge," *Journal of Fluid Mechanics*, vol. 176, pp. 483–520, 1987.

[48] G. Yang, D. Causon, and D. Ingram, "Calculation of compressible flows about complex moving geometries using a three-dimensional Cartesian cut cell method," *International Journal for Numerical Methods in Fluids*, vol. 33, no. 8, pp. 1121–1151, 2000.

[49] A. Bryson and R. Gross, "Diffraction of strong shocks by cones, cylinders, and spheres," *Journal of Fluid Mechanics*, vol. 10, no. 01, pp. 1–16, 1961.

[50] M. de León, "Orion Capsule (NASA 3D Resources)." URL `https://nasa3d.arc.nasa.gov/detail/orion-capsule`. Accessed: 31-10-2016.

[51] J. J. Quirk, "A contribution to the great Riemann solver debate," *International Journal for Numerical Methods in Fluids*, vol. 18, no. 6, pp. 555–574, 1994.

[52] C. Helzel, M. J. Berger, and R. J. LeVeque, "A high-resolution rotated grid method for conservation laws with embedded geometries," *SIAM Journal on Scientific Computing*, vol. 26, no. 3, pp. 785–809, 2005.

[53] L. Schneiders, C. Günther, M. Meinke, and W. Schröder, "An efficient conservative cut-cell method for rigid bodies interacting with viscous compressible flows," *Journal of Computational Physics*, vol. 311, pp. 62–86, 2016.

[54] M. Berger, M. J. Aftosmis, and S. R. Allmaras, "Progress Towards a Cartesian Cut-Cell Method for Viscous Compressible Flow," in *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, AIAA Paper 2012-1301, Jan 2012.

[55] M. J. Berger and M. J. Aftosmis, "An ODE-based Wall Model for Turbulent Flow Simulations," *AIAA Journal*, vol. 10, no. 2, pp. 700–714, 2017.

[56] D. Graves, P. Colella, D. Modiano, J. Johnson, B. Sjogreen, and X. Gao, "A cartesian grid embedded boundary method for the compressible Navier–Stokes equations," *Communications in Applied Mathematics and Computational Science*, vol. 8, no. 1, pp. 99–122, 2013.

[57] Z. Dragojlovic, F. Najmabadi, and M. Day, "An embedded boundary method for viscous, conducting compressible flow," *Journal of Computational Physics*, vol. 216, no. 1, pp. 37–51, 2006.

[58] P. Kundu and L. Cohen, *Fluid Mechanics*. Academic Press, 5th ed., 2014.

[59] D. Tritton, "Experiments on the flow past a circular cylinder at low Reynolds numbers," *Journal of Fluid Mechanics*, vol. 6, no. 4, pp. 547–567, 1959.

[60] Y.-H. Tseng and J. H. Ferziger, "A ghost-cell immersed boundary method for flow in complex geometry," *Journal of Computational Physics*, vol. 192, no. 2, pp. 593–623, 2003.

[61] A. S. Grove, F. Shair, and E. Petersen, "An experimental investigation of the steady separated flow past a circular cylinder," *Journal of Fluid Mechanics*, vol. 19, no. 1, pp. 60–80, 1964.

[62] C. Wieselsberger, "New data on the laws of fluid resistance," *Physikalische Zeitschrift*, vol. 22 (1921).

[63] M.-C. Lai and C. S. Peskin, "An immersed boundary method with formal second-order accuracy and reduced numerical viscosity," *Journal of Computational Physics*, vol. 160, no. 2, pp. 705–719, 2000.

[64] H. Glaz, P. Colella, I. Glass, and R. Deschambault, "A numerical study of oblique shock-wave reflections with experimental comparisons," in *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 398, pp. 117–140, The Royal Society, 1985.

[65] M. Al-Marouf and R. Samtaney, "A versatile embedded boundary adaptive mesh method for compressible flow in complex geometry," *Journal of Computational Physics*, vol. 337, pp. 339–378, 2017.

[66] H. Uddin, R. Kramer, and C. Pantano, "A Cartesian-based embedded geometry technique with adaptive high-order finite differences for compressible flow around complex geometries," *Journal of Computational Physics*, vol. 262, pp. 379–407, 2014.

[67] A. P. Krasil'shchikov and V. P. Podobin, "Experimental study of sphere aerodynamic characteristics in free flight up to M $\sim$ 15," *Fluid Dynamics*, vol. 3, no. 4, pp. 132–134, 1972.

[68] A. Bailey and J. Hiatt, "Sphere Drag Coefficients for a Broad Range of Mach and Reynolds Numbers," *AIAA Journal*, vol. 10, no. 11, pp. 1436–1440, 1972.

[69] A. Islam and B. Thornber, "High-order detached-eddy simulation of external aerodynamics over an SAE notchback model," *The Aeronautical Journal*, vol. 121, no. 1243, pp. 1342–1367, 2017.

[70] D. Wood, M. A. Passmore, and A. K. Perry, "Experimental Data for the Validation of Numerical Methods - SAE Reference Notchback Model," *SAE Int. J. Passeng. Cars - Mech. Syst.*, vol. 7, no. 1, pp. 145–154, 2014.

[71] D. Aljure, J. Calafell, A. Baez, and A. Oliva, "Flow over a realistic car model: Wall modeled large eddy simulations assessment and unsteady effects," *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 174, pp. 225–240, 2018.

[72] P. Davidson, *Turbulence: An Introduction for Scientists and Engineers*. Oxford University Press, 2015.

[73] N. Ashton, A. West, S. Lardeau, and A. Revell, "Assessment of RANS and DES methods for realistic automotive models," *Computers & Fluids*, vol. 128, pp. 1–15, 2016.

[74] B. Launder and D. Spalding, "The numerical computation of turbulent flows," *Computer Methods in Applied Mechanics and Engineering*, vol. 3, no. 2, pp. 269–289, 1974.

[75] P. Spalart and S. Allmaras, "A one-equation turbulence model for aerodynamic flows," in *30th Aerospace Sciences Meeting and Exhibit, Aerospace Sciences Meetings*, AIAA Paper 92-0439, Jan 1992.

[76] P. R. Spalart, "Detached-Eddy Simulation," *Annual Review of Fluid Mechanics*, vol. 41, pp. 181–202, 2009.

[77] E. S. Oran and J. P. Boris, "Computing Turbulent Shear Flows–A Convenient Conspiracy," *Computers in Physics*, vol. 7, no. 5, pp. 523–533, 1993.

[78] L. Shao, S. Sarkar, and C. Pantano, "On the relationship between the mean flow and subgrid stresses in large eddy simulation of turbulent shear flows," *Physics of Fluids*, vol. 11, no. 5, pp. 1229–1248, 1999.

[79] E. Garnier, M. Mossi, P. Sagaut, P. Comte, and M. Deville, "On the Use of Shock-Capturing Schemes for Large-Eddy Simulation," *Journal of Computational Physics*, vol. 153, no. 2, pp. 273–311, 1999.

[80] K. Oßwald, A. Siegmund, P. Birken, V. Hannemann, and A. Meister, "L2Roe: a low dissipation version of Roe's approximate Riemann solver for low Mach numbers," *International Journal for Numerical Methods in Fluids*, vol. 81, no. 2, pp. 71–86, 2016.

[81] B. Thornber, A. Mosedale, D. Drikakis, D. Youngs, and R. J. Williams, "An improved reconstruction method for compressible flows with low Mach number features," *Journal of Computational Physics*, vol. 227, no. 10, pp. 4873–4894, 2008.

[82] M. Meinke, W. Schröder, E. Krause, and T. Rister, "A comparison of second-and sixth-order methods for large-eddy simulations," *Computers & Fluids*, vol. 31, no. 4-7, pp. 695–718, 2002.

[83] M.-S. Liou and C. J. Steffen Jr, "A New Flux Splitting Scheme," *Journal of Computational Physics*, vol. 107, no. 1, pp. 23–39, 1993.

[84] N. Alkishriwi, M. Meinke, and W. Schröder, "A large-eddy simulation method for low Mach number flows using preconditioning and multigrid," *Computers & Fluids*, vol. 35, no. 10, pp. 1126–1136, 2006.

[85] A. Pogorelov, M. Meinke, and W. Schröder, "Cut-cell method based large-eddy simulation of tip-leakage flow," *Physics of Fluids*, vol. 27, no. 7, p. 075106, 2015.

[86] D. R. Chapman, "Computational Aerodynamics Development and Outlook," *AIAA Journal*, vol. 17, no. 12, pp. 1293–1313, 1979.

[87] P. R. Spalart, S. Deck, M. L. Shur, K. D. Squires, M. K. Strelets, and A. Travin, "A new version of detached-eddy simulation, resistant to ambiguous grid densities," *Theoretical and Computational Fluid Dynamics*, vol. 20, no. 3, p. 181, 2006.

[88] M. L. Shur, P. R. Spalart, M. K. Strelets, and A. K. Travin, "A hybrid RANS-LES approach with delayed-DES and wall-modelled LES capabilities," *International Journal of Heat and Fluid Flow*, vol. 29, no. 6, pp. 1638–1649, 2008.

[89] D. C. Wilcox, *Turbulence Modeling for CFD*. DCW Industries, 3rd ed., 2006.

[90] J. Larsson, S. Kawai, J. Bodart, and I. Bermejo-Moreno, "Large eddy simulation with modeled wall-stress: recent progress and future directions," *Mechanical Engineering Reviews*, vol. 3, no. 1, pp. 15–00418, 2016.

[91] F. Capizzano, "Turbulent Wall Model for Immersed Boundary Methods," *AIAA Journal*, vol. 49, no. 11, pp. 2367–2381, 2011.

[92] D. B. Spalding, "A Single Formula for the "Law of the Wall"," *Journal of Applied Mechanics*, vol. 28, no. 3, pp. 455–458, 1961.

[93] E. Serre, M. Minguez, R. Pasquetti, E. Guilmineau, G. B. Deng, M. Kornhaas, M. Schäfer, J. Fröhlich, C. Hinterberger, and W. Rodi, "On simulating the turbulent flow around the Ahmed body: A French–German collaborative evaluation of LES and DES," *Computers & Fluids*, vol. 78, pp. 10–23, 2013.

[94] S. R. Ahmed, G. Ramm, and G. Faltin, "Some salient features of the time-averaged ground vehicle wake," *SAE Transactions*, pp. 473–503, 1984.

[95] ANSYS, "Introductory FLUENT Training - Modeling Turbulent Flows." URL `http://www.southampton.ac.uk/~nwb/lectures/GoodPracticeCFD/Articles/Turbulence_Notes_Fluent-v6.3.06.pdf`. Accessed: 25-04-2018.

[96] D. Lyn, S. Einav, W. Rodi, and J.-H. Park, "A laser-Doppler velocimetry study of ensemble-averaged characteristics of the turbulent near wake of a square cylinder," *Journal of Fluid Mechanics*, vol. 304, pp. 285–319, 1995.

[97] O. Antepara, O. Lehmkuhl, R. Borrell, J. Chiva, and A. Oliva, "Parallel adaptive mesh refinement for large-eddy simulations of turbulent flows," *Computers & Fluids*, vol. 110, pp. 48–61, 2015.

[98] W. Rodi, "Comparison of LES and RANS calculations of the flow around bluff bodies," *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 69, pp. 55–75, 1997.

[99] J. C. R. Hunt, A. A. Wray, and P. Moin, "Eddies, stream, and convergence zones in turbulent flows," *Center for Turbulence Research Report CTR-S88*, 1988.

[100] W. Rodi, J. Ferziger, M. Breuer, and M. Pourquie, "Status of large eddy simulation: results of a workshop," *Transactions-American Society of Mechanical Engineers Journal of Fluids Engineering*, vol. 119, pp. 248–262, 1997.

[101] M. A. Passmore, A. K. Perry, and D. Wood, "SAE reference model: 20 degree notchback validation dataset (reference SAE paper 2014-01-0590)." URL `https://dspace.lboro.ac.uk/2134/13886`. Accessed: 26-04-2018.

[102] TUM, "DrivAer Model." URL `http://www.aer.mw.tum.de/en/research-groups/automotive/drivaer/`. Accessed: 27-04-2018.

[103] A. I. Heft, T. Indinger, and N. A. Adams, "Experimental and numerical investigation of the DrivAer model," in *ASME 2012 Fluids Engineering Division Summer Meeting collocated with the ASME 2012 Heat Transfer Summer Conference and the ASME 2012 10th International Conference on Nanochannels, Microchannels, and Minichannels*, pp. 41–51, American Society of Mechanical Engineers, 2012.

[104] A. Jameson, "Time Dependent Calculations Using Multigrid, with Applications to Unsteady Flows Past Airfoils and Wings," in *10th Computational Fluid Dynamics Conference, Fluid Dynamics and Co-located Conferences*, 1991.