Immersed-Boundary Wall-Modelled Large-Eddy Simulations of Turbulent Hypersonic Flows

William van Noordt

St. Anne's College
Department of Engineering Science
University of Oxford
October 2024

Supervisors: Dr. Luca Di Mare, Dr. Matthew McGilvray

This dissertation is submitted for the degree of
Doctor of Philosophy

Abstract

Hypersonic flight is of significant research interest due to its implications for space exploration, high-speed transportation, and defence applications. The challenges associated with hypersonic flows arise from a variety of complex phenomena, including strong shocks, thermal nonequilibrium, and high-temperature effects, which complicate the design and analysis of hypersonic vehicles. Experimental methods for hypersonic research are costly and limited, often failing to replicate flight conditions accurately. As such, computational frameworks that can effectively simulate hypersonic flows are critical for advancing understanding and design capabilities in this field.

This thesis presents a novel computational framework that integrates the immersed boundary method (IBM) with wall-modelled large-eddy simulation (WMLES) to accurately simulate turbulent flows around hypersonic vehicles. The framework features a numerical formulation that is specially designed for high-speed turbulent flows. By introducing a new ghost-cell immersed boundary method with enhanced stability characteristics, this work enables the effective application of IBM-WMLES in hypersonic flows, achieving significant improvements in predictions for higher Mach numbers.

An analysis of commutation errors in the wall model shows their substantial impact on heat transfer and skin friction predictions, accounting for approximately 5-10% of errors at high Mach numbers, which are influenced by wall model sensitivity and boundary condition covariance. To address these issues, a correction method based on a dual number system is derived which is particularly effective in regions experiencing shock wave boundary layer interactions, thereby enhancing simulation accuracy. Additionally, the framework's implementation is optimised for computational efficiency, achieving more than 35% peak floating point operations per second (FLOPS) on available test hardware. This optimisation enables IBM-WMLES to compete with traditional Reynolds-averaged Navier-Stokes (RANS) methods in terms of computational cost while offering greater accuracy in complex hypersonic flow scenarios. The new framework achieves this using only consumer-grade computing hardware and eliminates the need for mesh generation.

Acknowledgements

I would like to thank Luca and Matt for supervising me for the last few years. Thank you for all of your help with my work and requirements, for funding assistance when it was needed, and for ensuring that I met my goals within my programme here.

I must extend a great deal of thanks to Dr. Christoph Brehm at the University of Maryland for years of research discussion, meetings, and aid with technical problems. Without his help, and that from Sparsh, John, Joel, Vincenzo, and others at UMD, none of this work would have been possible.

My friends and colleagues at St. Anne's and around the world have supported me throughout my journey, and we have enjoyed many adventures together. James, Emily, Riddhi, Vedang, Harry, Daniel, Greg, Ruby, Alex, Ray, Kieran, thank you for being there for me whenever I needed, and for the endless laughs along the way.

I would like to acknowledge financial support from the Hypersonic Vehicle Simulation Institute with Dr. Russ Cummings as director, as well as computing resources from Cirrus, the University of Maryland, and the NASA Advanced Supercomputing Division.

To Annabel, your support has meant more to me than I can express. Thank you for being there every day, supplying me with tea and cake, and keeping me going through to the end.

And finally, my eternal thanks are extended to my family for all of their love and support throughout my entire academic career and beyond. Mum and Dad, Minnie, Ari, Mark and Lisa, John and Sally-Ann, I could not have done this without you.

"*We feel that even if all possible scientific questions be answered, the problems of life have still not been touched at all.*"

- Ludwig Wittgenstein

To Mum and Dad

Contents

List of Figures

Chapter 1: Introduction

1.1 Background

Hypersonic flight has been a significant research interest since as early as 1938 [11], driven by scientific inquiries into space exploration, commercial interests in high-speed and space transportation, and defence interests in rapid tactical response. Hypersonic flows do not share a single defining feature, but rather bear a family resemblance with each other through a number of common characteristics, including strong shocks, thermal nonequilibrium, aerothermochemistry, high-temperature effects, transition, viscous heating. Each of these characteristics poses a unique challenge in isolation, and the complexity of a hypersonic flow problem dramatically increases as more flow phenomena are accounted for. Hypersonic vehicles often have various features that produce airflows that can be subject to any of these phenomena (see figure 1.1). Furthermore, to design real hypersonic flight hardware, non-flow phenomena (e.g. structural response, material response, etc.) typically need to be accounted for under critical conditions.

One of the most significant barriers to understanding in hypersonics and hypersonic vehicle design is that experimental data of any form necessarily comes at great expense. Because the kinetic energy of a stream is proportional to the square of the stream velocity, it becomes increasingly difficult to accelerate any amount of air to the required speed for hypersonic research. While low-speed wind tunnels can rely on large fans to provide a constant stream in a wind tunnel test section, hypersonic flow experiments rely mostly on shock tubes, as well as expansion tunnels, impulse facilities, and blowdown tunnels, each with unique limitations compared to the steady and controllable conditions of low-speed tunnels. Test times for shock tubes are extremely short ($\approx$ 10-100 ms instead of minutes), facilities and supplies are expensive, the test section is subject to violent startup, and test conditions are difficult to reproduce. These challenges are bearable with sufficient patience and expertise (and funding), but there is one problem that is much more fundamental to most hypersonics experiments: flight conditions are impossible to match in ground facilities.

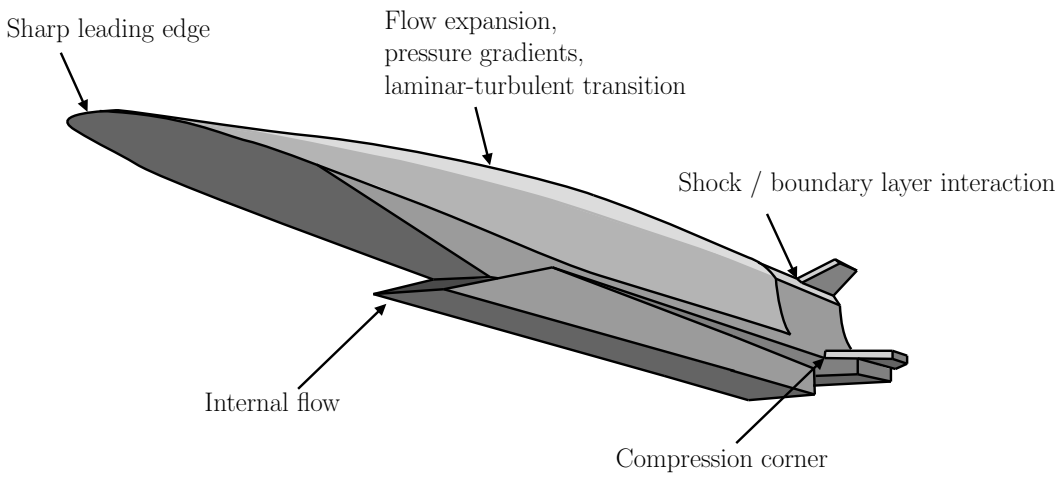


Figure 1.1: Example hypersonic vehicle schematic showing locations of various relevant flow features.

For hypersonic flows, there are, broadly, four nondimensional parameters that characterise the various flow regimes. The first is the Mach number, which is the ratio of the free stream velocity to the speed of sound within that stream. The Mach number determines the angle at which shock waves deflect off of inclined surfaces. At high Mach numbers, shock waves get closer to a vehicle's surface and heating from compression becomes increasingly significant. The second parameter is the Reynolds number, which can be thought of as the ratio of inertial to viscous or dissipative forces. At higher Reynolds numbers, instabilities in the flow amplify small fluctuations that turn into large-scale random noise known as turbulence. In high-speed flows, turbulence is another source of heating and can damage vehicle surfaces. The last two parameters to consider are the total enthalpy and chemical composition of the flow, both of which will have effects on the intermolecular forces, chemical interactions, and ionisation potential of the gas.

In hypersonic ground-test facilities such as shock tubes, it is generally not possible to match all parameters for a particular flight condition of interest. For example, higher Mach numbers can be achieved by lower-temperature freestream flows at the cost of having lower total enthalpy, or higher Reynolds number flows can be achieved at lower velocities by using a different gas, which changes the chemical properties of the freestream fluid. Furthermore, while atmospheric chemistry has had a significant amount of time to come to equilibrium, test gases may not have the opportunity to do so in the rapid timescale of hypersonic tests. Experimental shock-tube data for some conditions are a compromise between certain flow features and may not fully represent

the flow at flight conditions.

An alternative to ground test facilities are hypersonic flight tests, where an instrumented experimental geometry is affixed to a rocket or booster. These tests have the advantage of directly reproducing flight conditions, but come at significant cost over ground tests. A prominent example is the BOLT flight experiment from JHU-APL [12], which offers a rich flight-test data set but suffered a series of setbacks and failed launches. While it is difficult to characterise the financial cost of such tests, it should be unsurprising that such flight tests are not viable as routine for hypersonic vehicle design. Since hypersonic vehicles experience such extreme aerothermodynamic environments during flight, it should be evident from disasters such as the Columbia Space Shuttle landing [13] that having an accurate understanding of the flight environment is critical to mitigate risks to personnel and equipment. Given the limitations of ground and flight testing, it should be no surprise that hypersonic flight simulations bear the most responsibility for the hypersonic vehicle design process.

Although simulations are much cheaper and safer to perform than live tests, they come with their own set of challenges. Numerical algorithms must be able to handle the extreme features of solutions to the Navier-Stokes equations in hypersonic conditions. High-fidelity methods must be efficiently implemented, yet still be able to resolve important features of boundary layers, flow separation, shocks, etc. Nonlinear solvers must be able to handle stiff problems such as chemical source terms without incurring too much computational expense.

Ideally, all hypersonic vehicle design would be digitally analysed and certified through computational fluid dynamics (CFD) simulations. Simulations would be sufficiently cheap to allow for commercially viable analysis turnaround time, while maintaining enough fidelity and accuracy to capture all of the important quantities for safety margins and design optimisations. While this is a lofty goal, this work aims to take a modest step towards that ideal through the development of a computational framework for high-fidelity analysis of hypersonic vehicles, involving a series of novel numerical algorithms, a new physical model for high-speed turbulent boundary layers, a unique combination of turbulent flow simulation methods, and a fast computational algorithm and implementation.

Slotnick et al. [14] list many challenges facing computational aerosciences as a practice, most of which will appear as a theme of any study in CFD. It is impossible for any single body of work to address all of the issues listed, but this work will focus primarily on the simulation of hypersonic, external, transitional/turbulent flows over complex geometries. For the purpose of this work, any flow that has a freestream Mach number $M \gtrapprox 5$ will be considered hypersonic. Flows will be treated as perfect-gas flows with no chemistry.

1.2 Turbulent Flows

The numerical simulation of general turbulent flows has a long history dating back to the 1970s [15]. While there is no formal definition of a turbulent flow, turbulent flows are thought of as flows containing a series of large vortices or unsteady fluctuations that exhibit chaotic behaviour. Through nonlinear interactions, large eddies transfer energy to smaller ones, continuing until the Kolmogorov microscales, where viscous dissipation converts kinetic energy into heat. Notionally, it is possible to take a statistically steady turbulent flow and plot the total kinetic energy at a particular length scale against the wavenumber (inversely proportional to the length scale), which would ideally result in a plot similar to that seen in figure 1.2.

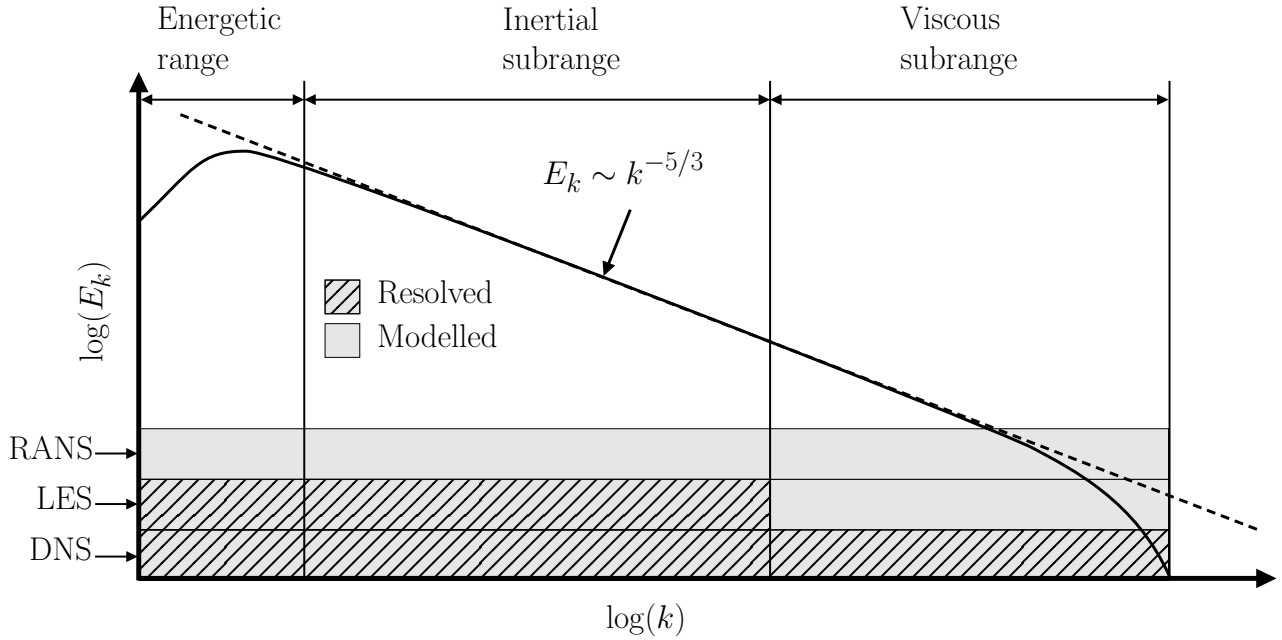


Figure 1.2: Turbulent energy spectrum with various turbulence modelling approaches listed. Drawn based off of ref. [1].

On the low end of the spectrum are the low-frequency scales that describe large-scale motion within the flow. At the point of flow separation, for example, one may expect some degree of large-scale motion. In the centre of the wavenumber spectrum lies the inertial subrange, where the aforementioned turbulent kinetic energy cascade happens. At the high-wavennumber end of the spectrum is the viscous subrange, where small-scale fluctuations are dissipated as heat at the Kolmogorov scales. Figure 1.3 shows a simulation of the breakdown to turbulence of the Taylor-Green vortex [16], where the flow is initially laminar, i.e. orderly and regular, and then small instabilities in shear layers are amplified linearly, after which nonlinear breakdown happens and the flow becomes chaotic.

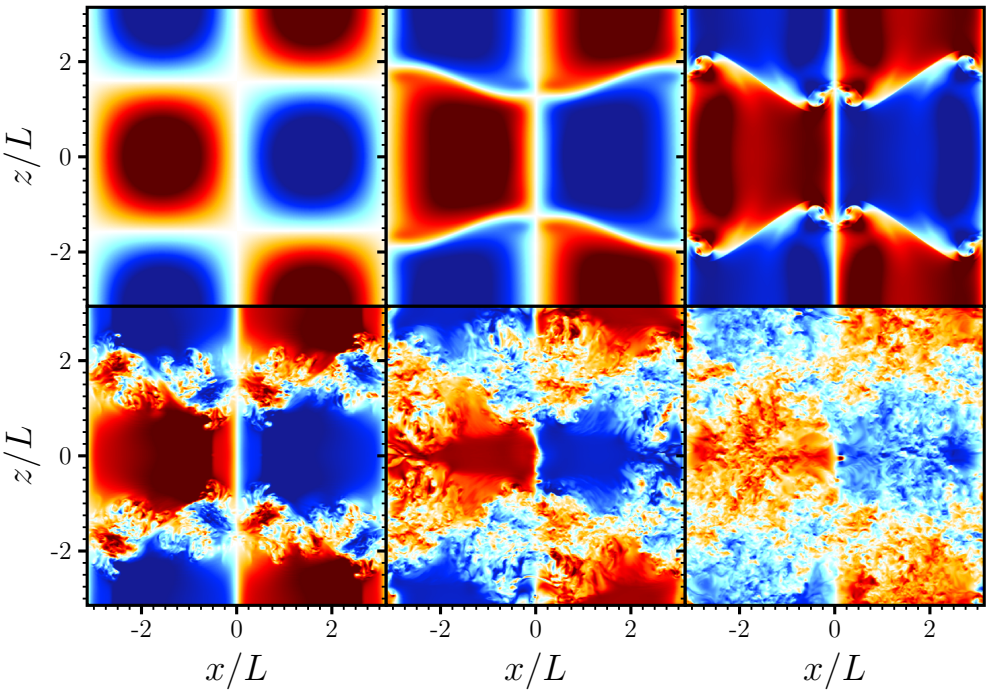


Figure 1.3: Breakdown to turbulence of the Taylor-Green Vortex, contours of velocity component in x-direction

Different turbulent simulation approaches aim to resolve different scales in the energy spectrum. As a theoretical starting point, one may consider the incompressible Navier-Stokes momentum equations, given by

$$\frac{\partial u_i}{\partial t} + \frac{\partial \left(u_i u_j\right)}{\partial x_j} = \frac{1}{\rho}\left(\frac{\partial \tau_{ij}}{\partial x_j} - \frac{\partial p}{\partial x_i}\right), \tag{1.1}$$

with velocity components u_i, density ρ, pressure p, and shear stress τ_{ij}, and the Reynolds decom-

position for any variable ϕ

$$\overline{\phi} = \int_0^\infty \phi(t)\,\mathrm{d}t \quad \text{and} \quad \phi' = \phi - \overline{\phi}. \tag{1.2}$$

Applying the averaging operator from (1.2) to both sides of (1.1) and decomposing the velocities as $u_i = \overline{u_i} + u_i'$ gives

$$\frac{\partial \overline{u}_i}{\partial t} + \frac{\partial \left(\overline{u}_i \overline{u}_j\right)}{\partial x_j} = \frac{1}{\rho}\left(\frac{\partial \overline{\tau_{ij}}}{\partial x_j} - \frac{\partial \overline{p}}{\partial x_i}\right) - \overbrace{\frac{\partial \left(\overline{u_i' u_j'}\right)}^{=\partial R_{ij}/\partial x_i}}{\partial x_i}. \tag{1.3}$$

The right-hand-side term R_{ij} is called the *Reynolds Stress Tensor* and is the subject of many years of turbulence modelling. Since the averaging operator applied to get (1.2) is a full-spectrum ensemble-averaging (expressed as a time-average for ergodic flows), all unsteady motion is averaged out and accounted for by R_{ij}. The action of the Reynolds-stress term in (1.3) is generally diffusive, distributing (in a mean sense) momentum throughout a flow via mixing. The particular approach above is called Reynolds-Averaged Navier-Stokes (RANS), and is among the most popular approaches for turbulence modelling, listed in figure 1.2, but requires an accurate closure for the Reynolds-stress terms based on the time-averaged flow field.

While RANS problems can be solved using steady techniques or unsteady (time-accurate) techniques, large-eddy simulations (LES) of all kinds are inherently unsteady. LES involves using a small-window spatiotemporal filter rather than a time-average, with the aim of filtering out motions in the viscous subrange and in the higher-wavenumber part of the inertial subrange. Direct numerical simulation (DNS) simply involves resolving all scales with unsteady techniques, and results in flows like those depicted in figure 1.3.

It should be relatively unsurprising that, in general, the more scales that are modelled in a flow, the less physically accurate the result may be. Certain flow features are difficult to capture in a time-averaged sense, especially flow features that have large amounts of energy contained in the lower frequencies. It is possible to tune RANS models to allow them to capture certain flow features more effectively, but a truly universal RANS model remains a lofty goal. Development and accuracy evaluation of RANS models is an extensive topic, but the summary in Ashton and van Noordt [2] concludes that RANS models still struggle to capture wake interactions, separation, and physical rates of shear layer mixing over scale-resolving methods, even for low-speed flows

where RANS models have decades of development. Figure 1.4 (adapted from [2]) shows the drag coefficient prediction for an automotive body for scale-resolving methods compared against RANS methods. It is clear from this particular dataset that RANS predictions cluster away from the experimental values (grey lines). This is a recurring theme in these comparison studies.

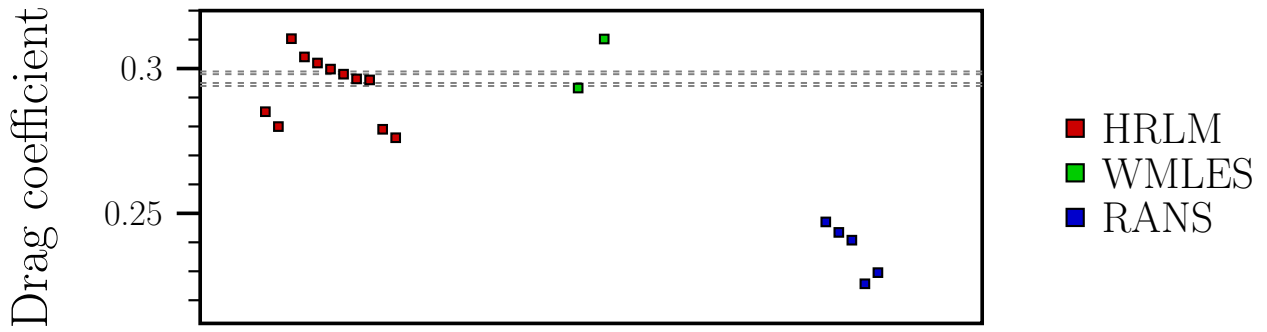


Figure 1.4: Drag coefficient prediction over an automotive geometry using scale-resolving methods compared with RANS methods, adapted from [2]. HRLM represents hybrid RANS-LES methods. WMLES represents wall-modelled large-eddy simulations. Dotted lines represent experimental measurements.

Unlike for low-speed flows, RANS models for high speed flows do not have nearly as much history, meaning that many high-speed RANS calculations suffer from significant errors, especially for flows with significant heat transfer [17]. This combination of factors motivates the use of scale-resolving methods, particularly large-eddy simulation, for this thesis. When performing large-eddy simulations, one first chooses a length scale, Δ, to represent the smallest scale at which fluctuations are resolved. As shown in figure 1.2, this length scale should be small enough that most of the scales within the inertial range are resolved, but the viscous subrange scales are not. The compressible Navier-Stokes equations are then spatially filtered, and the filtered variables are solved for in the LES.

The spatially-filtered (denoted by $\overline{\cdot}$) momentum components of the compressible Navier-Stokes equations are given by

$$\frac{\partial\left(\overline{\rho}\widetilde{u_j}\right)}{\partial t}+\frac{\partial\left(\overline{\rho}\widetilde{u_j}\widetilde{u_i}\right)}{\partial x_i}+\frac{\partial\overline{p}}{\partial x_j}=\frac{\partial}{\partial x_i}\left(\mu S_{ij}+\beta\delta_{ij}\frac{\partial\widetilde{u_k}}{\partial x_k}\right)-\frac{\partial\left(\widetilde{\overline{\rho}u_j''u_i''}\right)}{\partial x_i}, \tag{1.4}$$

where the strain rate $S_{ij}=\partial u_i/\partial x_j+\partial u_j/\partial x_i$, the bulk viscosity $\beta=-\frac{2}{3}\mu$ and the so-called *Favre*

average decomposition given by

$$\widetilde{\phi} = \frac{\overline{\rho\phi}}{\overline{\rho}}, \quad \phi = \widetilde{\phi} + \phi'' \tag{1.5}$$

has been applied to give the compressible Reynolds stress term $\widetilde{\overline{\rho} u_i'' u_j''}$. The influence of this term is usually realised by modelling the sub-grid scales using an *eddy viscosity* μ_t satisfying the Boussinesq approximation

$$-2\mu_t \left(S_{ij} - \frac{1}{3}\delta_{ij} S_{kk} \right) \approx \overline{\rho} \left(\widetilde{u_i'' u_j''} - \frac{1}{3}\delta_{ij} \widetilde{u_k'' u_k''} \right) \tag{1.6}$$

which is usually assumed to be isotropic, i.e. independent of i and j. The final Favre-averaged momentum equations then simply read as

$$\frac{\partial \left(\overline{\rho} \widetilde{u_j} \right)}{\partial t} + \overbrace{\frac{\partial \left(\overline{\rho} \widetilde{u_j} \widetilde{u_i} \right)}{\partial x_i}}^{=C} + \frac{\partial \overline{p}}{\partial x} = \frac{\partial}{\partial x_i} \left(\left(\mu + \mu_t \right) \overline{S}_{ij} + (\beta + \beta_t) \delta_{ij} \frac{\partial \overline{u_k}}{\partial x_k} \right). \tag{1.7}$$

The details of computing μ_t is, once again, the subject of much research (e.g. [18, 19]), but will not be a significant focus of this thesis. Note that β_t is simply set to $-\frac{2}{3}\mu_t$.

Broadly, the resolved turbulent scales that are solved for in LES are the result of modal interactions in the highly nonlinear convective term C in (1.7). As such, the method used to provide a discrete formulation of this particular term can have significant effects on the quality of any numerical solution. Numerical approximations for C must compromise between having low-enough dissipation to preserve turbulent fluctuations and having enough numerical dissipation to be able to maintain solution stability. For low-speed flows, numerical algorithms do not need much numerical dissipation. However, for supersonic and hypersonic Mach numbers, flows develop shock waves which are sharp discontinuities in the flow field. These features pose issues for low-dissipation numerical algorithms, since they can cause degenerate features such as unbounded oscillations which cause the solution to (1.4) to diverge. The presence of both turbulence and shock waves requires that numerical algorithms are *both* stable and non-dissipative, which requires a more sophisticated algorithm than is seen for low-speed and/or laminar flows.

1.3 Turbulent Boundary Layers

As discussed in section 1.2, turbulence is a phenomenon that arises from the interaction of small fluctuations in a flowfield with regions of high shear. Under the right conditions, a disturbance that enters a shear layer can grow linearly in time until it becomes sufficiently large, at which point nonlinear breakdown occurs and the flow transitions into chaotic turbulent flow. High shear rates are ubiquitous in boundary layers, where the no-slip condition $u_i = 0$ at solid walls creates thin regions of flow near solid boundaries with large velocity gradients. After transition has happened, the high-momentum fluid near the edge of the boundary layer begins to mix into the low-momentum fluid deeper into the boundary layer near the wall, resulting in much higher skin friction and, for high-speed flows, significantly higher skin friction and heat transfer, both of which peak within the transitional region. This is depicted in figure 1.5.

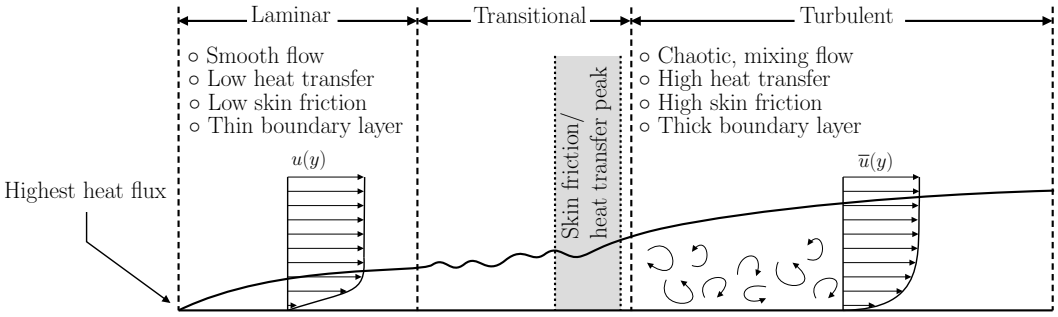


Figure 1.5: Schematic depiction of the anatomy of a boundary layer undergoing laminar-turbulent transition.

Boundary layer transition is its own field of research, and at the time of writing, there is no unified theory of it for hypersonic boundary layers that encompasses all stages of transition. Transition in the hypersonic regime can be affected by many more flow parameters than in the low-speed regime [20]. A detailed description of the mechanics of transition will not be present in this work, but capturing and understanding the effects of transition is of great importance in hypersonic flows.

Low-speed turbulent boundary layers follow the *law of the wall* in the absence of pressure gradients, where the inner-scaled mean velocity obeys different scaling relationships at certain inner-scaled distances from the wall. In the region very close to the wall, the mean velocity scales linearly with the distance from the wall in what's called the viscous sublayer. It is in this layer

that the fluid shear rate is highest. Further away from the wall, the mean velocity scales with the logarithm of the distance to the wall in what's known as the log layer. The "universal" turbulent boundary layer profile is depicted in figure 1.6.

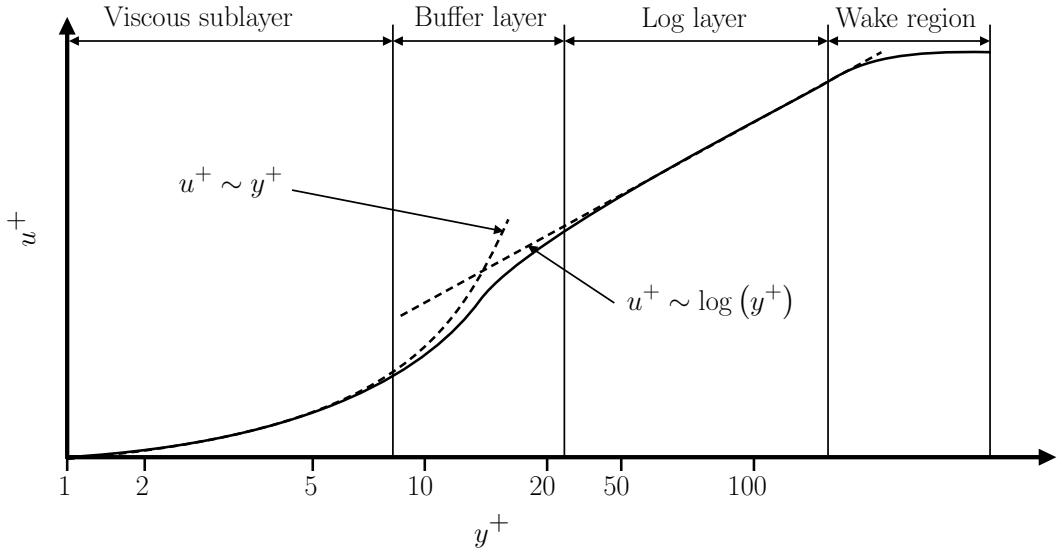


Figure 1.6: Universal turbulent boundary layer profile, showing various regions of the inner parts of the boundary layer.

As mentioned previously, the shear rates near the wall in the boundary layer serve to amplify flow disturbances that eventually lead to transition. However, the energy cascade that is depicted in figure 1.2 also plays a role in shaping the boundary layer, moreso in the log layer. One of the main challenges that boundary layers pose for a CFD calculation is the resolution requirements that are imposed at solid walls. Typical guidelines suggest that the cell size for the grid near the wall should be chosen such that the first cell lies at approximately $y^+ = 1$ Because this is such a small cell, the cell size is typically geometrically grown in the wall-normal direction until cells are large enough to be practical to place over the rest of the domain.

While this is simple enough in practice, there are two issues with this framework. The first issue is that this generates an enormous number of cells that must be solved for (with the exception of RANS, where large aspect ratios significantly reduce cell counts). The other issue is that the time-step for an unsteady simulation is limited by the smallest cell in the domain, meaning that a finer mesh near the wall also requires a smaller timestep, leading to an explosion in spatiotemporal complexity for the whole problem.

Yang and Griffin [21] provide an analysis of this problem for a spatially-developing turbulent boundary layer. In their analysis they estimate the scaling rate of the spatiotemporal requirements, given by $N_t N_x$ where N_t is the number of required timesteps and N_x is the number of required grid cells, with respect to the Reynolds number $\mathrm{Re} = \rho u L/\mu$. They conclude that, for a DNS or LES calculation,

$$(N_t N_x)_{\mathrm{DNS}} \sim \mathrm{Re}^{2.91} \quad \text{and} \quad (N_t N_x)_{\mathrm{LES}} \sim \mathrm{Re}^{2.72}, \tag{1.8}$$

and that the main requirement driving the near-cubic cost scaling is the need to resolve deep within the boundary layer.

To alleviate this significant cost, there is no substitute for simply not resolving the grid to the required extent near the wall. This would degrade the quality of the CFD results unless there were some way to apply a complex boundary condition that would mimic the behaviour of the flow near the wall. This can be done in turbulent flow regions by assuming that the velocity profile obeys the law of the wall in figure 1.6 in a mean sense, as first shown by Deardorff in 1970 [22]. This practice is called *wall-modelled large-eddy simulation* (WMLES). The ramifications of assuming this kind of behaviour near the wall are discussed further in chapter 3. According to the aforementioned Reynolds number scaling analysis,

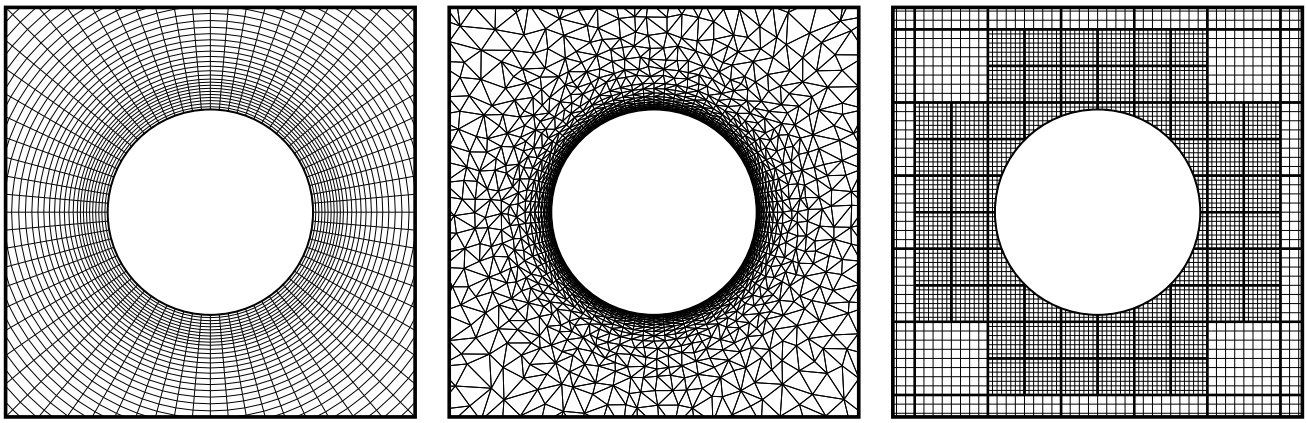
$$(N_t N_x)_{\mathrm{WMLES}} \sim \mathrm{Re}^{1.14}. \tag{1.9}$$

This scaling is clearly favourable when compared to wall-resolving methods in (1.8), especially for flows over large-complex geometries where relevant length scales are very large. It is for this reason that WMLES is chosen as the approach for this work.

1.4 Mesh Generation and Boundary Conditions

Fluid flows representable by discrete values centred at a large number of specific locations, similarly to how images of real-life objects can be represented as aggregates of data-containing pixels. This collection of locations is called the *mesh*, and is one of the most important considerations when performing CFD. More precisely, the mesh is a discretized representation of the geometric domain

over which the governing equations are solved. It divides the domain into small elements or cells (typically triangles, quadrilaterals, tetrahedra, or hexahedra), over which the governing equations are approximated numerically. Mesh generation for CFD falls into one of three broad categories, although more esoteric methods do exist. In *curvilinear* grids, mesh points are organised in a logically cartesian manner and then nodal coordinates are moved around the domain so that the computational boundary of the mesh aligns with the surface of the geometry of interest, as in figure 1.7(a). *Unstructured* meshes consist of an irregular, non-uniform pattern of cells, possibly of mixed type, arranged around the geometry, as in figure 1.7(b). Finally, *block-structured Cartesian* grids consist of a large number of rectangular blocks containing highly regular arrangements of cells within them, as in figure 1.7(c).



(a) Curvilinear grid (b) Unstructured grid (c) Cartesian block-structured grid

Figure 1.7: Notional depictions of common types of meshing strategies around a cylindrical body.

A critical aspect of mesh generation is to ensure that grids have a high *grid quality*. The notion of grid quality varies for different meshing strategies and for different simulation techniques, but broadly consists of a few requirements:

1. in regions of large flow gradients, there must be sufficient resolution to resolve those gradients provided that they are of relevance to any quantity of interest,

2. in regions where it is necessary to resolve turbulence, the grid cells must be spaced regularly and with sufficient density to resolve all turbulent scales that are chosen above the filter cutoff scale (in the case of LES/WMLES), and

3. in regions where the flow is constant or near-constant, or essentially inviscid, the mesh should be sufficiently coarse so as not to generate so many cells that the calculation is intractable.

When compared against these requirements, the strategies outlined in figure 1.7 start to have clear weaknesses and strengths. The curvilinear approach generates high-quality meshes with efficiently-distributed points, but these meshes are incredibly difficult to generate. Unstructured grids are "easy" to generate (although still an active area of research), and can have an efficient distribution of points, but it is difficult to generate unstructured grids with high grid quality, making them less suitable for scale-resolving flows. Block-structured Cartesian meshes are easy to generate and provide highly regular grids (provided one can treat the interface between coarse and fine blocks correctly), but don't have the ability to stretch the size of grid cells as rapidly, leading to a less efficient distribution of cells. Because the "correct" mesh for a particular flow condition will depend on the solution to the field equations for that condition, mesh generation is typically an effort intensive procedure. Mesh generation can account for up to 50% of the human effort required to perform CFD calculations [23, 24], and has a significant impact on the accuracy of the results on a given mesh.

Another aspect of mesh generation is the computational efficiency of storing and accessing mesh data over the course of a calculation. While, for modern computing architecture, this may not seem like a relevant point, it becomes relevant when considering that the meshes used for scale-resolving simulations become very large and calculations are run on memory-limited graphics-processing units (GPUs). While the ramifications of running a CFD solver on GPUs will be discussed at a later point, having a low memory and data-access footprint will come to be a key aspect of choosing a meshing strategy.

In this work, all meshes will be generated using a block-structured Cartesian mesh generation strategy, mainly because they:

- are easy to generate, since one can place fine mesh blocks near the geometry with desired resolution, and
- have efficient storage and access footprints (despite the necessarily increased cell count), since one only ever stores the coordinates of individual blocks (details of memory access will

be addressed in chapter 4), rather than storing the coordinates of every node in the domain.

The main challenge faced when using this approach is accounting for the presence of the solid geometry via boundary conditions. A close look at figure 1.7(c) shows that the grid is not aligned with the boundary, meaning that special treatment must be applied at the boundary to maintain consistency with the boundary conditions for the field equations. The class of methods used to treat the boundary numerically in this instance is called *immersed boundary methods* (IBMs). The main challenge of boundary treatment in this case is to develop a stable and accurate numerical boundary condition, which is the focus of chapter 2.

1.5 Hypersonic Flow Considerations

The previous sections in this chapter describe problems that are present in low-speed simulations of turbulence. However, when simulating hypersonic flows, there are additional considerations for each of the problems above. While hypersonic flow phenomena are numerous, for the majority of this work, two key elements of hypersonic flows will be considered: strong shock waves and viscous boundary-layer heating.

From a physical perspective, strong shock waves radically compress the fluid as it passes through the plane of the shock. This results in a discontinuity in most flow quantities, particularly pressure, temperature, and entropy. A sharp increase in temperature radically increases local viscous heating rates, especially at leading edges of hypersonic vehicles. Furthermore, shock waves can impinge upon other flow regions onto the geometry surface and interact with turbulent boundary layers, leading to viscous-inviscid interactions that augment the state of boundary layers through pressure gradients, often inducing separation and accelerating the transition process. From a numerical perspective, shock waves introduce a sharp discontinuity that poses a challenge to any numerical algorithm that required the numerical differentiation of the solution. Typical finite approximations of derivatives can become large at discontinuities and grow without bound as the discretization becomes finer. This means that numerical algorithms must be modified to be able to handle the presence of discontinuities without erasing solution information at the finer scales (as mentioned in section 1.2). This is especially challenging when applying numerical

boundary conditions at the immersed boundary in under-resolved boundary layers.

Viscous boundary layer heating occurs when the dissipation in shear layers becomes significant when compared to the amount of internal energy. The total energy equation in the Favre-averaged compressible Navier-Stokes equations can take multiple forms. In this work, following [25], reads as

$$\frac{\partial \left(\overline{E}\right)}{\partial t} + \frac{\partial \left(\widetilde{u_i}(\overline{E} + \overline{p})\right)}{\partial x_i} = \frac{\partial \left(\tau_{ik}\widetilde{u_k} - \overline{\dot{q}_i}\right)}{\partial x_i}, \tag{1.10}$$

where $E = \rho T/\gamma + \rho u_i u_i/2$ is the total energy and $\dot{q}_i = -\left(\frac{\mu}{\mathrm{Pr}} + \frac{\mu_t}{\mathrm{Pr}_t}\right)\frac{\partial \overline{T}}{\partial x_i}$ is the diffusive heat transfer, $\mathrm{Pr} = 0.71$ is the Prandtl number, and $\mathrm{Pr}_t = 0.9$ is the turbulent Prandtl number. The term $\partial(\tau_{ik}\widetilde{u_k})/\partial x_i$ is an energy source term that generates a significant amount of heat in high-speed shear layers. Note that the shear stress tensor τ_{ij} is taken directly from the momentum equation, including turbulent contributions. This term becomes significant in turbulent boundary layers approximately when $M \gtrapprox 4$ [7]. When this is the case, aerodynamic heating becomes significant throughout the majority of the boundary layer, specifically the portion of the boundary layer that is modelled using wall-modelling. Under these conditions, the law of the wall in figure 1.6 (which is fundamentally an incompressible flow phenomenon) ceases to be universal, and the boundary layer profile becomes distorted by variable fluid properties. There have been many attempts to provide a relation between an arbitrary high-speed boundary layer profile and the incompressible law of the wall, but this is an active area of research [26, 27, 28].

1.6 Summary of Research Contributions

In this work, the immersed boundary method on block-structured Cartesian grids will be combined with wall-modelled large-eddy simulation on GPUs and applied to hypersonic flow problems. IBMs are uniquely suited for WMLES, since IBMs main disadvantage (unable to efficiently resolve turbulent boundary layers due to low-aspect-ratio cells) is complemented by the main advantage of WMLES (relaxed resolution requirements in boundary layers). In a similar way, block-structured Cartesian meshes are uniquely suited for graphics processing unit (GPU) architecture, since their low-storage requirements and high computational requirements match precisely the strengths of GPU hardware.

The final aim of this approach is to produce a novel computational framework that is capable of simulating complex hypersonic flows around vehicles, with significantly greater accuracy than state-of-the-art RANS methods yet at similar computational cost, and to do so without the need for the process of mesh generation. In this thesis, this goal is realised by the following contributions:

Chapter 2:

- a derivation of a new immersed boundary scheme that is designed specifically for turbulent hypersonic external flows,
- a new method of setting boundary conditions that is specific to the new framework and allows for greater solution accuracy,
- a new mathematical understanding of the role of the inviscid numerical flux when setting boundary conditions,
- a new understanding of the interaction between numerical errors and modelling errors in turbulent boundary layers treated with an immersed boundary method,
- the identification of a robust yet accurate numerical scheme for hypersonic LES,

Chapter 3:

- a new understanding of unsteady stresses that are introduced implicitly by the use of the wall model, including when they become significant,
- an observation that unsteady stresses partially inhibit the development of more accurate turbulence models,
- a new method for removing the effects of these unsteady stresses, separating the time scales for unsteady errors and instantaneous shear stress and heat flux,

Chapter 4:

- a detailed account of the computational performance of the new method,

- presentation of optimisations of the new method that present significant computational efficiency, and
- a demonstration that the new computational framework, taken as a whole, is a competitive approach for hypersonic vehicle simulations.

Chapter 6:

- a demonstration that the new computational framework, taken as a whole, is a competitive approach for hypersonic vehicle simulations.

1.7 Research Impact

This thesis is composed of three main achievements:

1. the development, implementation, and evaluation of a novel and efficient IBM-WMLES approach applied to turbulent hypersonic flows,
2. the analysis and correction of commutation error leading to shear stress and heat transfer depletion, and
3. the optimisation and evaluation of the computational cost of the IBM-WMLES approach, demonstrating significant performance gains.

To the author's knowledge, van Noordt et al. [5] is the first published work on the simulation of hypersonic turbulent flows using WMLES-IBM.

Publications

- **W. van Noordt**, S. Ganju, and C. Brehm. *An immersed boundary method for wall-modeled large-eddy simulation of turbulent high-Mach-number flows*, Journal of Computational Physics, volume 470, 2022.
- **W. van Noordt**, J. McQuaid, and C. Brehm. *Optimization of a Cartesian Immersed-Boundary Navier-Stokes Solver for GPU Architecture*, Journal of Supercomputing, 2024 (in preparation).

- S. Ganju, **W. van Noordt**, and C. Brehm. *Wall-Model Informed Ghost-Cell Reconstruction for Immersed Boundary Large-Eddy Simulations*, Journal of Computational Physics, volume 470, 2024 (in preparation).

- S. Ganju, **W. van Noordt**, and C. Brehm. *Progress in the Development of an Immersed Boundary Viscous-Wall Model for 3D and High-Speed Flows*, AIAA SciTech Forum, 2021.

- J. B. Chapelier, D. Lushner, **W. van Noordt**, C. Wenzel, T. Gibis, P. Mossier, A. Beck, C. Brehm, M. Ruggeri, C. Scalo, and N. Sandham. *Comparison of high-order numerical methodologies for the simulation of the supersonic Taylor–Green vortex flow*, Physics of Fluids, 2021.

- J. Larsson, I. Bermejo-Moreno, D. Garmann, D. Rizetta, R. Baurle, T. Mukha, S. Toosi, P. Schlatter, C. Brehm, S. Ganju, A. Berk Kahraman, **W. van Noordt**, Z. Wang, Z. Duan, M. Blind, A. Beck, S. Dave, and A. Korobenko. *Summary of the Smooth Body Separation Test Case at the 2022 High Fidelity CFD Verification Workshop*, AIAA SciTech Forum, 2023.

- **W. van Noordt**, S. Ganju, L. di Mare, and C. Brehm. *Modelling Errors in Wall-Modelled Large-Eddy Simulations of High-Speed Channel Flows.* AIAA SciTech Forum, 2023.

- **W. van Noordt**, S. Ganju, and C. Brehm. *Immersed-Boundary Wall-Modeled Large-Eddy Simulation of High Mach Number Boundary Layer Flows.* AIAA Aviation Forum, 2021.

- S. Ganju, **W. van Noordt**, and C. Brehm. *An Immersed Boundary Wall-Modeled Large-Eddy Simulation Approach for Low Speed Turbulent Flows.* AIAA Aviation Forum, 2022.

- N. Ashton and **W. van Noordt**. *Overview and Summary of the First Automotive CFD Prediction Workshop: DrivAer Model.* SAE International Journal of Commercial Vehicles, 2022.

- V. Kumar, C. Brehm, J. Larsson, and **W. van Noordt**. *Assessment of Wall-Modeled LES in High-Speed Boundary Layers with Body-Fitted and Immersed-Boundary Codes.* AIAA Aviation Forum, 2024.

- **W. van Noordt**, J. McQuaid, J. Larsson, and C. Brehm. *GPU-Accelerated Simulations of Turbulent Flows with the Immersed Boundary Method.* AIAA Aviation Forum, 2024.

Professional Workshops

- First Automotive CFD Prediction Workshop
- WMLES CFD Verification Workshop
- NATO-AVT High-Speed Turbulence Task Force

Invited Talks

- Wall-Modeled Large-Eddy Simulation of High-Speed Turbulent Flows with an Immersed Boundary Method, NASA Advanced Modelling and Simulation (AMS) Seminar series, 19 August, 2021

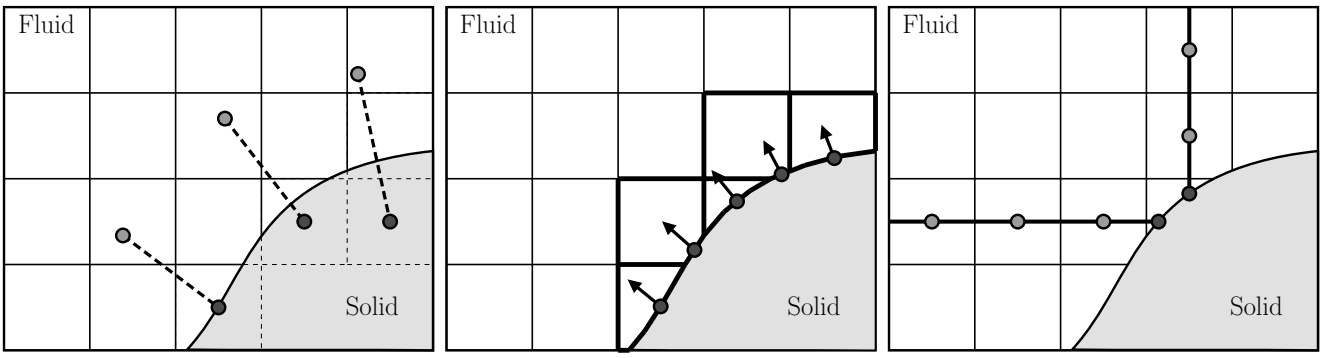
1.8 Thesis Overview

Chapter 2 details the development and validation of numerical algorithms used in this work. Chapter 3 discusses high-speed wall models. In chapter 4, the computational architecture and performance is discussed. Chapter 5 covers validation of the methods developed in this work. Chapter 6 demonstrates the developed methods for a series of more complex flow problems. Finally, chapter 7 concludes.

Chapter 2: Immersed Boundary Approach

2.1 Background

The classical research on immersed boundary methods begins with Peskin [29, 30] more than four decades ago, motivated by the convenience of having all flow quantities defined on a fixed domain. Since the inaugural studies, a variety of immersed boundary methods have been developed. Three common categories of immersed boundary method are shown in figure 2.1.



(a) Ghost-cell method (b) Cut-cell method (c) Boundary-flux method

Figure 2.1: Schematic depiction of three common approaches for immersed-boundary treatments.

Figure 2.1(a) depicts the theoretical basis for the *ghost cell method*. In this method, the boundary is numerically treated by filling cells that lie in the interior of the solid domain with virtual fluid states that, when used to calculate numerical fluxes, approximately account for the presence of the boundary. This method has the advantage of being relatively easy to implement, and benefits form the fact that the numerical formulation can, in the simplest case, largely remain unchanged. Ghost cell methods often require additional data structures that are necessary to handle thing geometries, and have difficulty discretely enforcing conservation. For reasons that are made clear in chapter 2, ghost cell methods are used in this work.

Figure 2.1(b) shows the basis for cut-cell methods. Simply put, cut cell methods essentially treat the Cartesian grid as a special case of an unstructured grid and use a finite-volume formulation at the boundary to ensure discrete conservation. Numerical methods for cut-cells can be

unstable and can be difficult to extend to high-order, and can suffer from the "small-cell" problem, limiting the available timestep size, although techniques exist for alleviating these issues.

Finally, figure 2.1(c) shows the line stencils used in boundary-flux methods (sometimes appearing in the literature under a variety of names). In this case, a finite-difference formulation is used to produce numerical derivatives of fluxes near the boundary. Boundary flux formulations have a number of favourable theoretical properties, but are difficult to stabilise.

Since the original works, much of the development of the immersed boundary method has been performed for incompressible flows. Goldstein et al. [31] extend Peskin's original formulation to investigate turbulent flows, and Johansen and Colella [32] made further improvements for the solution of the Poisson problem for pressure-corrected incompressible flow methods. Throughout the 1990s and into the early 2000s, immersed boundary methods continued to be developed for biological [33, 34, 35] flow problems.

As computing power became more available, immersed boundary methods began to see more application to compressible aerodynamic flows. An early example of this is Clarke et al. [36], who developed a cut-cell method for simulating two-dimensional compressible inviscid flows around airfoils, extended to viscous laminar flows two years later [37], where they noted that surface smoothness was a requirement for their proposed method. Melton et al. [38] performed three-dimensional inviscid compressible calculation at airline cruise conditions ($M = 0.85$) for a large transport aircraft as well as supersonic condition for a supersonic transport plane. At this point, immersed-boundary methods had become a popular method of analysis for inviscid aerodynamics [39, 40, 41]. The application of immersed boundary methods to turbulent flows has been limited by the fact that Cartesian grid cells generally have fixed aspect ratios. Iaccarino and Verzicco [42] give a review of applications of immersed boundary methods to turbulent flows as of the early 2000s, which include low-Reynolds-number turbulent flows around a variety of geometries for incompressible conditions.

For high Reynolds-number turbulent flows, it is almost always necessary to include some kind of wall model or wall treatment to alleviate the need to resolve all scales in the viscous sublayer and buffer layer. This wall treatment can include a simple analytical expressions for the near-wall

velocity profile, WMLES, detached-eddy simulation (DES), or any similar approach. The coupling of near-wall modelling and immersed boundary methods is a powerful combination. Bernardini et al. [43] demonstrate that IBM-DES with an analytic wall treatment is capable of higher-Reynolds-number flow simulations, yielding predictions of noise levels from flow around a primitive model of an aircraft landing gear. Tyagi and Acharya [44] use an asymptotically consistent near-wall treatment to perform LES of moving geometries, notably in the absence of turbulent boundary layers. Chang et al. [45] demonstrate the importance of including wall treatment for turbulent simulations.

Modern research on immersed boundary methods for aerodynamic flows, including the author's, generally makes use of WMLES as the method of choice for treating turbulent boundary layers. As an example, Tamaki and Imamura [46] simulated the NASA common research model using RANS and a wall function based on the linear solution of the Spalart-Allmaras (SA) [47] turbulence model. Shi et al. [48] present a non-equilibrium wall model for simulating massive flow separation over a backward-facing step. Along similar lines, Ma et al. [49] develop improvements to the wall model-IBM coupling and show good surface pressure predictions for a range of cylinder flows, with varying levels of accuracy for skin friction. Kang [50] develops an analytical IBM-WM approach for the numerical simulation of turbulent pipe flows, improving results by solving an equation for the subgrid-scale viscosity at each immersed boundary node. There are many more studies that involve making improvements to immersed boundary methods and wall models for subsonic compressible and incompressible flows [51, 52, 53, 54, 55, 56, 57].

Notably, there are very few studies on the application of immersed boundary methods to hypersonic flows. McQuaid et al. [58] demonstrated an immersed boundary method for hypersonic viscous flows with favourable heat transfer predictions. Similarly, Brahmachary et al. [59] use a sharp-interface method based on a ghost-in-fluid approach to perform laminar viscous hypersonic simulations over a flat plate and cylinder. Bridel-Bertomeu [60] investigated flows up to Mach 20 for a variety of numerical methods in two dimensions. More recently, Baskaya et al. [61, 62] performed comparisons of an immersed boundary method against an unstructured method for chemistry problems in various laminar hypersonic flows. Duan et al. [63] extended a curvilin-

ear (see figure 1.7(a)) framework to include an immersed-boundary treatment for the numerical simulation of hypersonic transition. Their numerical formulation was able to approximate the influence of small surface roughness elements on growth rates of instability modes. Notably, no wall treatment was used, since their grids integrated the full solution to the wall.

2.2 Importance of Centred Schemes

The full compressible Navier-Stokes equations for spatially-averaged (averaging symbols are omitted) variables are given as

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

$$\frac{\partial E}{\partial t} + \frac{\partial ((E+p)u_i)}{\partial x_i} = \frac{\partial (\tau_{ik} u_k - \dot{q}_i)}{\partial x_i}, \tag{2.2}$$

$$\frac{\partial (\rho u_j)}{\partial t} + \frac{\partial (\rho u_j u_i + \delta_{ij} p)}{\partial x_i} = \frac{\partial \tau_{ij}}{\partial x_i}. \tag{2.3}$$

Without the viscous terms, this can be written generally as

$$\frac{\partial \mathbf{w}}{\partial t} + \frac{\partial \mathbf{f}}{\partial x_i} = \frac{\partial \mathbf{w}}{\partial t} + \frac{\partial \mathbf{f}}{\partial \mathbf{w}} \frac{\partial \mathbf{w}}{\partial x_i} = \frac{\partial \mathbf{w}}{\partial t} + J \frac{\partial \mathbf{w}}{\partial x_i} = 0. \tag{2.4}$$

Writing the flux jacobian J with the diagonalization

$$J = V^{-1} \Lambda V \tag{2.5}$$

and left-multiplying by V gives

$$V \frac{\partial \mathbf{w}}{\partial t} + \Lambda V \frac{\partial \mathbf{w}}{\partial x_i} = 0. \tag{2.6}$$

If V is frozen (for the sake of analysis, it is typically assumed to be [64]), one can define the characteristic variables $\mathbf{c} = V\mathbf{w}$, giving

$$\frac{\partial \mathbf{c}}{\partial t} + \Lambda \frac{\partial \mathbf{c}}{\partial x_i} = 0. \tag{2.7}$$

Now, since Λ is a diagonal matrix, individual components can be locally analysed as the simple 1-dimensional scalar transport equation

$$\frac{\partial c}{\partial t} + a\frac{\partial c}{\partial x} = 0. \tag{2.8}$$

The effect of various discretizations of the inviscid term in (2.1 - 2.3) can be notionally analysed by applying similar discretizations to (2.8). Firstly, (2.8) is written discretely as

$$\frac{\partial \mathbf{c}}{\partial t} + a\mathcal{D}\mathbf{c} = 0, \tag{2.9}$$

Where $\mathcal{D}$ is a discrete derivative matrix. Letting $\{\lambda_j\}$ be the eigenvalues and $\mathbf{v}_i$ be the eigenvectors of $\mathcal{D}$ and assuming that $\mathcal{D}$ is normal (the examples in this section will be), the solution $\mathbf{c}(t)$ can be written as

$$\frac{\partial \mathbf{c}}{\partial t} + a\mathcal{D}\mathbf{c} = \sum_j \left(\frac{\partial m_j}{\partial t} + a m_j \lambda_j \right) \mathbf{v}_j = 0, \tag{2.10}$$

where $m_j = \langle \mathbf{c}, \mathbf{v}_j \rangle$. Since $\mathcal{D}$ is assumed to be normal, its eigenvectors are orthogonal for distinct eigenvalues and so (2.10) can be analysed independently for each mode using the equation

$$\frac{\partial m_j}{\partial t} + a m_j \lambda_j = 0, \tag{2.11}$$

with the solution

$$m_j(t) = e^{-a\lambda_j t} m_j(0). \tag{2.12}$$

The form of (2.12) shows that as the solution to 2.8 is evolved in time, the j^{th} mode in the initial condition will grow or decay based on the real part of the associated eigenvalue λ_j. For a linear problem like (2.8), the differentiation matrix is not a function of the solution, and so any mode with a positive eigenvalue will grow without bound.

There are many ways to build a matrix to approximate the differentiation matrix $\mathcal{D}$, but two will be considered here, namely the centred approximation $\mathcal{D}_1$ and forward approximation $\mathcal{D}_2$

given by

$$\mathcal{D}_1 = \frac{1}{\Delta x}\begin{pmatrix} 0 & \frac{1}{2} & 0 & \dots & -\frac{1}{2} \\ -\frac{1}{2} & 0 & \frac{1}{2} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{2} & 0 & \dots & -\frac{1}{2} & 0 \end{pmatrix}, \quad \text{and} \quad \mathcal{D}_2 = \frac{1}{\Delta x}\begin{pmatrix} -1 & 1 & 0 & \dots & 0 \\ 0 & -1 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 0 & \dots & -1 \end{pmatrix} \tag{2.13}$$

assuming periodicity. Both of these matrices are normal, so it will suffice for now to analyse the structure of their eigenvalue-eigenvector pairs.

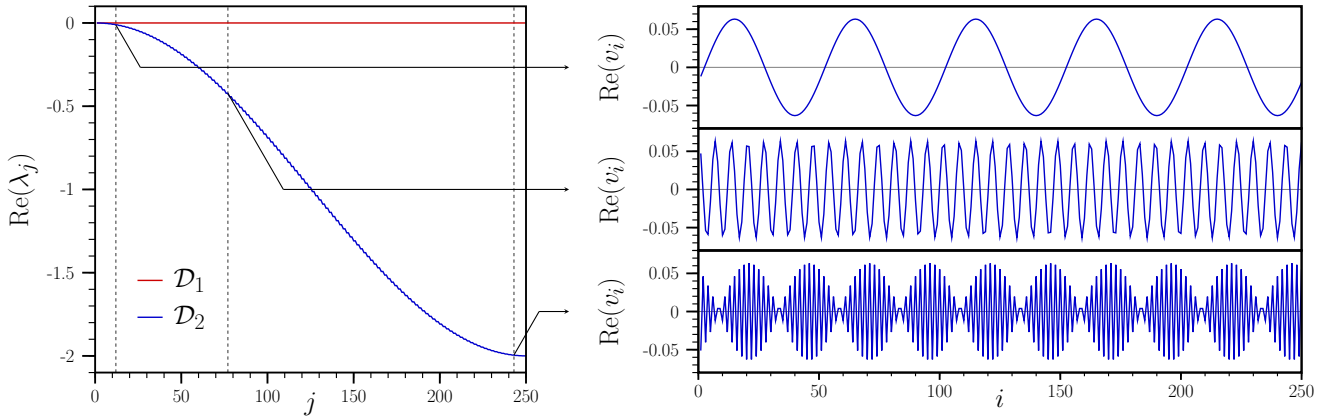


Figure 2.2: Left: eigenvalue spectrum plots for centred ($\mathcal{D}_1$) and forward ($\mathcal{D}_2$) differentiation operators. Right: real parts of corresponding eigenvectors of forward differentiation operator for low-, medium-, and high-frequency modes.

Figure 2.2 depicts the eigenvalues for both operators in (2.13) and corresponding eigenvectors for the forward difference operators in (2.13). The difference in these operators is relatively clear: the centred difference operator $\mathcal{D}_1$ preserves all modes present in the initial condition, while the forward difference operator $\mathcal{D}_2$ rapidly attenuates high-frequency modes. Note that both $\mathcal{D}_1$ and $\mathcal{D}_2$ are circulant, and thus their eigenvectors are exactly the Fourier modes.

In fully-resolved calculations such as DNS, the high-frequency modes occur either when shocks are present in the flow (since Fourier coefficients decay less rapidly in the presence of discontinuities [65]), or when these modes have been excited and produce an instability. Therefore, the appeal of dissipative numerical methods is clear for fully-resolved calculations, since turbulent scales are well-resolved and appear as lower-frequency modes relative to the grid spacing. However, for marginally-resolved turbulent flow calculations, care must be taken not to use too much numerical

dissipation lest the turbulent fluctuations be numerically annihilated. The use of minimally-dissipative schemes is of critical importance to the accuracy of numerical solutions and is the subject of a great deal of research [3, 66, 67, 68, 69].

While the basic analysis here has been applied only to linear scalar advection, it provides a clear illustration of the motivations for the use of dissipative and non-dissipative schemes. In summary, the goal is to use a sufficiently low dissipation in turbulent regions, while applying enough dissipation in regions with flow discontinuities to maintain numerical stability. Generally, this translates to employing centred derivative approximations whenever possible, and switching to an upwind formulation near shocks.

2.3 Kinetic Energy and Entropy Preservation

The behaviour of kinetic energy within a turbulent flow is critical to capture, since almost all turbulence modelling efforts appeal to properties of the kinetic energy spectrum. Formulations for eddy viscosity are designed to obey the five-thirds power-law in a mean sense [70, 71, 72]. However, since turbulent solutions to the Navier-Stokes equations contain energy at all scales, it is clearly of great importance to capture the large-scale behaviour of kinetic energy.

To understand the role of kinetic energy in numerical solutions to the NS equations, the total energy equation is expressed as

$$\frac{\partial}{\partial t}\left(\rho e + \rho k\right) + \frac{\partial}{\partial x_i}\left(\rho u_i \left(e + k + \frac{p}{\rho}\right)\right) = 0. \tag{2.14}$$

Note that the viscous terms will be ignored for the purpose of this analysis. Because the kinetic energy is expressed as

$$\rho k = \frac{1}{2}\rho u_k u_k, \tag{2.15}$$

it evolves through time as

$$\frac{\partial(\rho k)}{\partial t} = \frac{1}{2}\frac{\partial(\rho u_k u_k)}{\partial t} = \frac{\partial(\rho u_k u_k)}{\partial t} - \frac{1}{2}\frac{\partial(\rho u_k u_k)}{\partial t} = u_k\frac{\partial(\rho u_k)}{\partial t} + \rho u_k\frac{\partial u_k}{\partial t} - \frac{1}{2}u_k u_k\frac{\partial(\rho)}{\partial t} - \frac{1}{2}\rho\frac{\partial(u_k u_k)}{\partial t}$$

$$= u_k \frac{\partial(\rho u_k)}{\partial t} + \rho u_k \frac{\partial u_k}{\partial t} - \frac{1}{2} u_k u_k \frac{\partial(\rho)}{\partial t} - \rho u_k \frac{\partial(u_k)}{\partial t} = u_k \frac{\partial(\rho u_k)}{\partial t} - \frac{1}{2} u_k u_k \frac{\partial(\rho)}{\partial t} \tag{2.16}$$

By substituting the momentum and mass equation inviscid flux divergence terms for the time-derivative terms in (2.16), the inviscid kinetic energy equation is obtained as

$$\frac{\partial(\rho k)}{\partial t} + \frac{\partial(\rho u_i k)}{\partial x_i} + u_i \frac{\partial p}{\partial x_i} = 0. \tag{2.17}$$

Note that this step will be expanded further in later analysis. The fact that the kinetic energy equation (2.17) can be derived from the mass and momentum equations, coupled with the fact that kinetic energy is directly solved for in the total energy equation (2.14) allows for erroneous kinetic energy production or destruction by using inconsistent approximations.

Other than kinetic energy, another secondary quantity that plays an important role in high-speed flow phenomena is thermodynamic entropy. According to the 2$^{\text{nd}}$ law of thermodynamics, thermodynamic entropy within a closed system can only be increased, i.e.

$$\frac{\partial(\rho s)}{\partial t} + \frac{\partial(\rho u_i s)}{\partial x_i} \geq 0. \tag{2.18}$$

For inviscid flows, only shock waves can generate entropy. One of the most critical numerical considerations for thermodynamic entropy is that entropy is not erroneously destroyed [73]. Following Subbareddy and Candler [74] and Kuya et al. [75], the thermodynamic entropy obeys the equation

$$\frac{\partial(\rho s)}{\partial t} + \frac{\partial(\rho u_i s)}{\partial x_i} = \frac{1}{T}\left(\left(sT - e - \frac{p}{\rho}\right)\left(\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u_i)}{\partial x_i}\right) + \frac{\partial(\rho e)}{\partial t} + \frac{\partial(\rho u_i e)}{\partial x_i} + p\frac{\partial u_i}{\partial x_i}\right) \tag{2.19}$$

in the absence of viscous effects. By subtracting (2.17) from (2.14), the inviscid internal energy equation is obtained as

$$\frac{\partial(\rho e)}{\partial t} + \frac{\partial(\rho u_i e)}{\partial x_i} + p\frac{\partial u_i}{\partial x_i} = 0. \tag{2.20}$$

By (2.20) and (2.1), it is clear that the right-hand side of (2.19) is zero analytically.

Therefore, to generate a numerical discretization of the inviscid fluxes that will work for high-

speed turbulent flows, the following requirements must be satisfied:

1. The discretization for the kinetic energy term in (2.14) must be consistent with the identity (2.16).

2. The discretization of the pressure work term in (2.14) must obey the product rule $\partial(u_i p)/\partial x_i = u_i \partial(p)/\partial x_i + p\partial(u_i)/\partial x_i$ in order to preserve entropy in the absence of shocks, and

3. any numerical dissipation added for stabilisation to the mass and energy equations must appear as source terms in (2.19) that add to total entropy.

Kuya et al. [75] present a numerical formulation of the convective flux that satisfy these conditions, based on the following identities for variables ϕ, ψ and η:

$$(\phi_i + \phi_{i+l})(\psi_i + \psi_{i+l}) - (\phi_i + \phi_{i-l})(\psi_i + \psi_{i-l}) = (\phi_{i+l}\psi_{i+l} - \phi_{i-l}\psi_{i-l}) + \phi_i(\psi_{i+l} - \psi_{i-l}) + \psi_i(\phi_{i+l} - \phi_{i-l}), \tag{2.21}$$

$$\begin{aligned}&(\phi_i + \phi_{i+l})(\psi_i + \psi_{i+l})(\eta_i + \eta_{i+l}) - (\phi_i + \phi_{i-l})(\psi_i + \psi_{i-l})(\eta_i + \eta_{i-l}) \\ &= (\phi_i\psi_i\eta_i - \phi_{i-l}\psi_{i-l}\eta_{i-l}) + \phi_i(\psi_{i+l}\eta_{i+l} - \psi_{i-l}\eta_{i-l}) + \psi_i(\phi_{i+l}\eta_{i+l} - \phi_{i-l}\eta_{i-l}) \\ &+ \eta_i(\phi_{i+l}\psi_{i+l} - \phi_{i-l}\psi_{i-l}) + \phi_i\psi_i(\eta_{i+l} - \eta_{i-l}) + \phi_i\eta_i(\psi_{i+l} - \psi_{i-l}) + \eta_i\psi_i(\phi_{i+l} - \phi_{i-l}), \quad \text{and}\end{aligned} \tag{2.22}$$

$$(\phi_i\psi_{i+l} + \phi_{i+l}\psi_i) - (\phi_i\psi_{i-l} + \phi_{i-l}\psi_i) = \phi_i(\psi_{i+l} - \psi_{i-l}) + \psi_i(\phi_{i+l} - \phi_{i-l}). \tag{2.23}$$

Each of these identities is analagous to a differentiation rule, in the sense that linear combinations of them produce discrete derivative estimates. As an example, if the right-hand side of 2.23 is expressed as $D_l = \phi_i(\psi_{i+l} - \psi_{i-l}) + \psi_i(\phi_{i+l} - \phi_{i-l})$, then the higher-order derivative term is given as

$$\frac{2}{3}D_1 - \frac{1}{12}D_2 = \phi_i\left(\frac{1}{12}\psi_{i-2} - \frac{2}{3}\psi_{i-1} + \frac{2}{3}\psi_{i+l} - \frac{1}{12}\psi_{i+2}\right) + \psi_i\left(\frac{1}{12}\phi_{i-2} - \frac{2}{3}\phi_{i-1} + \frac{2}{3}\phi_{i+l} - \frac{1}{12}\phi_{i+2}\right). \tag{2.24}$$

The identity (2.21) is analogous to

$$\frac{\mathrm{d}(\phi\psi)}{\mathrm{d}x} = \frac{1}{2}\frac{\mathrm{d}(\phi\psi)}{\mathrm{d}x} + \frac{1}{2}\psi\frac{\mathrm{d}\phi}{\mathrm{d}x} + \frac{1}{2}\phi\frac{\mathrm{d}\psi}{\mathrm{d}x}, \tag{2.25}$$

the identity (2.22) is analogous to

$$\frac{\mathrm{d}(\phi\psi\eta)}{\mathrm{d}x} = \frac{1}{4}\frac{\mathrm{d}(\phi\psi\eta)}{\mathrm{d}x} + \frac{1}{4}\phi\psi\frac{\mathrm{d}\eta}{\mathrm{d}x} + \frac{1}{4}\eta\frac{\mathrm{d}(\phi\psi)}{\mathrm{d}x} + \frac{1}{4}\psi\eta\frac{\mathrm{d}\phi}{\mathrm{d}x} + \frac{1}{4}\phi\frac{\mathrm{d}(\psi\eta)}{\mathrm{d}x} + \frac{1}{4}\phi\eta\frac{\mathrm{d}\psi}{\mathrm{d}x} + \frac{1}{4}\psi\frac{\mathrm{d}(\phi\eta)}{\mathrm{d}x}, \tag{2.26}$$

and (2.23) is analogous to

$$\frac{\mathrm{d}(\phi\psi)}{\mathrm{d}x} = \phi\frac{\mathrm{d}\psi}{\mathrm{d}x} + \psi\frac{\mathrm{d}\phi}{\mathrm{d}x}. \tag{2.27}$$

The derivation of (2.16) relies on the differentiation rules (2.25) and (2.26), and therefore the identity (2.16) will hold discretely (2.25) and (2.26) are used to discretize the corresponding terms in the fluxes. Similarly, the pressure diffusion term in (2.14) must be discretized according to (2.27) in order for the product rule to hold discretely.

This results in the following formulae for each term in the Navier-Stokes equations:

$$\frac{\partial(\rho u_i)}{\partial x} = \frac{1}{2}\mathcal{D}(\rho u_i) + \frac{1}{2}\rho\mathcal{D}(u_i) + \frac{1}{2}u\mathcal{D}(\rho_i) \tag{2.28a}$$

$$\begin{aligned}\frac{\partial(\rho u_i u_j + \delta_{ij}p)}{\partial x_i} &= \frac{1}{4}\mathcal{D}(\rho u_i u_j) + \frac{1}{4}\rho\mathcal{D}(u_i u_j) + \frac{1}{4}u_i\mathcal{D}(\rho u_j) + \frac{1}{4}u_j\mathcal{D}(\rho u_i) \\ &+\frac{1}{4}\rho u_i\mathcal{D}(u_j) + \frac{1}{4}\rho u_j\mathcal{D}(u_i) + \frac{1}{4}u_i u_j\mathcal{D}(\rho) + \delta_{ij}\mathcal{D}(p)\end{aligned} \tag{2.28b}$$

$$\begin{aligned}\frac{\partial(\rho u_i e)}{\partial x_i} &= \frac{1}{4}\mathcal{D}(\rho u_i e) + \frac{1}{4}\rho\mathcal{D}(u_i e) + \frac{1}{4}u_i\mathcal{D}(\rho e) + \frac{1}{4}e\mathcal{D}(\rho u_i) \\ &+\frac{1}{4}\rho u_i\mathcal{D}(e) + \frac{1}{4}\rho e\mathcal{D}(u_i) + \frac{1}{4}u_i e\mathcal{D}(\rho)\end{aligned} \tag{2.28c}$$

$$\begin{aligned}\frac{1}{2}\frac{\partial(\rho u_i u_k u_k)}{\partial x_i} &= \frac{1}{4}(\rho u_k)\mathcal{D}(u_i u_k) + \frac{1}{4}(u_k)\mathcal{D}(\rho u_i u_k) \\ &+\frac{1}{4}u_i u_k\mathcal{D}(\rho u_k) + \frac{1}{4}\rho u_i u_k\mathcal{D}(u_k)\end{aligned} \tag{2.28d}$$

$$\frac{\partial(u_i p)}{\partial x_i} = u_i\mathcal{D}(p) + p\mathcal{D}(u_i) \tag{2.28e}$$

Note that, although not written in conservative form, (2.28a) - (2.28e) are implicitly conservative as noted by Pirozzoli [76].

2.4 Immersed Boundary Challenges

The numerical treatment of the boundary conditions of (2.1) - (2.3) is the principal challenge in developing the framework presented in this thesis. In more traditional body-fitted approaches such as those given in figure 1.7(a) and 1.7(b), numerical treatment of the boundary can be handled by simply imposing boundary conditions in a direct manner. Using IBM-WMLES, especially for hypersonic flows, the numerical boundary conditions must delicately compromise between satisfying a number of conditions, each of which has the potential to corrupt the numerical solution entirely. A brief discussion of each challenge is provided below. This chapter represents the bulk of the contribution of this thesis.

2.4.1 Numerical Stability

Historically, immersed-boundary methods have always required special consideration when it comes to numerical stability. Generally, this comes from the fact that the numerical discretization is necessarily different at every point on the boundary, making stability difficult to demonstrate. Typically, numerical stability is demonstrated approximately by using a procedure like that in section 2.2, where the governing equations, in inviscid form, are modelled as one-dimensional scalar advection equations and discretized. They are then analysed through matrix stability analysis, energy methods, or summation-by-parts approaches [77, 78, 79]. Though imperfect from a theoretical perspective, this gives an excellent practical estimate of the behaviour of any particular numerical scheme. In this work, matrix stability will be used to analyse stability.

It is typical of linear matrix stability analysis to express a time-dependent discrete evolution law as

$$\mathbf{v}^{(n+1)} = A\mathbf{v}^{(n)}, \tag{2.29}$$

where $\mathbf{v}^{(n)}$ is the discrete solution at a time-step n and A represents the so-called "update" matrix

(this will be discussed further in section 2.10). It is clear from (2.29) that

$$\mathbf{v}^{(n)} = A^n \mathbf{v}^{(0)}, \tag{2.30}$$

indicating that the discrete solution is only bounded in time if

$$\rho\left(A\right) < 1, \tag{2.31}$$

where $\rho(\cdot)$ is the spectral radius operator. For statistically-steady turbulent flows, analytical solutions are necessarily bounded.

However, as noted by Trefethen [64], the condition (2.31) describes the asymptotic behaviour of the discrete solution as $n \to \infty$, which indicates only that the solution is *eventually* bounded. Consider the following example, adapted from [64]:

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & \dots \\ \frac{1}{4} & 0 & 1 & 0 & \dots \\ 0 & \frac{1}{4} & 0 & 1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \in \mathbb{R}^{N \times N}. \tag{2.32}$$

It is fairly simple to show that (2.31) is satisfied, but successive powers of A grow very rapidly until eventually decaying to zero. This is shown in figure 2.3. While $||A^k||$ eventually tends to

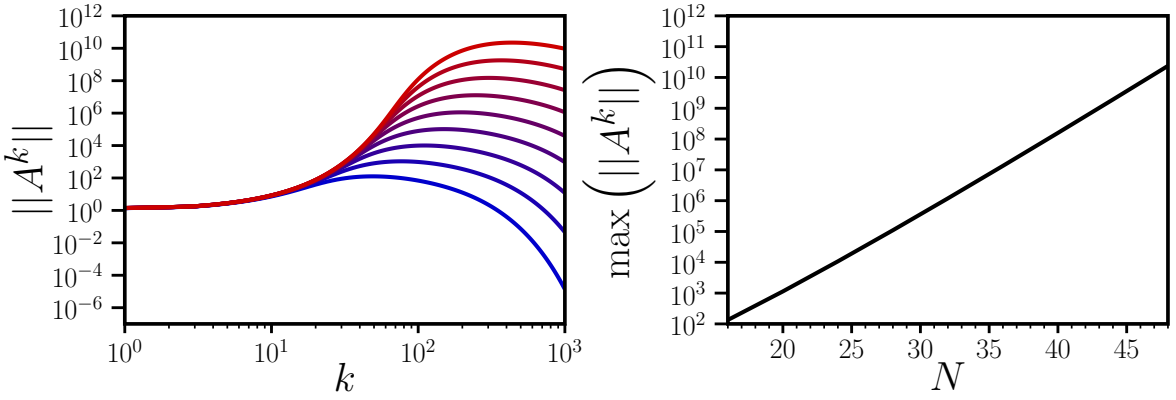


Figure 2.3: Left: norms of successive powers of A for various matrix sizes between $N = 16$ (blue) and $N = 48$ (red). Right: maximum norm of successive powers of A as a function of the matrix size N.

zero for all N, the short-term growth (invisible to eigenvalue analysis) leads to values that make numerical treatment difficult. In the context of a CFD simulation, this could be realised as the short-term divergence of the solution.

According to the theory of pseudospectra, this effect can be observed in systems with update matrices that have highly non-normal eigenvectors, which is the case at the boundary, where locally biassed differentiation and reconstruction can lead to highly non-normal matrices, as shown in Brehm and Fasel [80]. Therefore, the method developed in this chapter must be shown to be pseudospectrally stable (i.e. not exhibit the behaviour in figure 2.3) as well as meet the other traditional stability requirements of numerical schemes.

2.4.2 Modelling Considerations

Similarly to the numerical discretization away from the boundary, the immersed boundary treatment must apply the lowest amount of numerical dissipation possible while still maintaining stability. This is challenging given the stability considerations in section 2.4.1. Not only can numerical dissipation corrupt the small-scale features of turbulent flow, but it introduces significant errors even in steady laminar boundary layers.

This can be demonstrated by considering classical incompressible channel flow. Taking the time, sreamwise-, and spanwise-averaged Navier-Stokes momentum equation with a fixed pressure gradient ϕ and constant density yields

$$\frac{\partial \left(f_c - f_v\right)}{\partial y} = \phi, \tag{2.33}$$

where $f_c = \rho u v$ is the convective flux and $f_v = \mu \frac{\partial u}{\partial y}$ is the viscous flux. Physically, the convective flux is zero at the wall by virtue of the no-slip condition. Consider the approximation

$$f_c\Big|_{j+1/2} = \frac{1}{8}\left(\rho_{j+1} + \rho_j\right)\left(u_{j+1} + u_j\right)\left(v_{j+1} + v_j\right) + \alpha\left((\rho u)\Big|_{j+1} - (\rho u)\Big|_j\right), \tag{2.34}$$

leading to the convective flux derivative approximation

$$\left.\frac{\partial f_c}{\partial y}\right|_j = \frac{1}{\Delta y}\left(\left.f_c\right|_{j+1/2} - \left.f_c\right|_{j-1/2}\right). \tag{2.35}$$

In (2.34), the difference term is added as numerical dissipation, scaled by the coefficient α, to ensure stability.

At the boundary, the flux must be constructed using boundary condition information. This can be done by considering a virtual flow state at a position inside the solid geometry, near the flowfield, depicted as the $j = -1$ position in figure 2.4. The obvious (and typical) choice for this

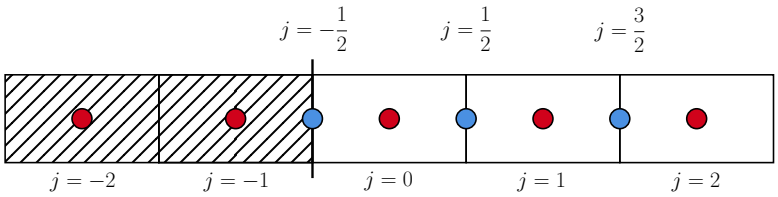


Figure 2.4: Simple near-wall stencil and numbering convention for one-dimensional channel flow. Blue circles indicate face centers (at which fluxes are computed) and red circles indicate cell centres (at which flowfield values are stored).

virtual state is to adhere to the no-slip condition: given the velocities at $j = 0$ and that $u = v = 0$ at $j = -1/2$, then the approximations $u_{-1} = -u_0$ and $v_{-1} = -v_0$ would yield $(u_{-1}+u_0)/2 = 0$ and $(v_{-1} + v_0)/2 = 0$, both of which are attractive properties when considering the flux approximation (2.34).

However, the problem is clear when taking the approximation for the full flux. Considering any consistent approximation for $\rho_{-1/2}$, the full flux is evaluated as

$$\left.f_c\right|_{-1/2} = \alpha\left((\rho u)_0 - (\rho u)_{-1}\right). \tag{2.36}$$

This means that the inviscid flux is nonzero on the boundary and that it acts as an offset to the shear stress experienced by the flow near the wall.

With the numerical contribution from the convective flux ignored, (2.33) becomes

$$-\mu\frac{\partial^2 u}{\partial y^2} = \phi \tag{2.37}$$

with the solution over $y \in [-\delta, \delta]$

$$u = \frac{\delta^2 \phi}{2\mu}\left(1 - \left(\frac{y}{\delta}\right)^2\right). \tag{2.38}$$

The expression (2.33) can be integrated from $-\delta$ to δ to obtain the conservation property

$$(f_c - f_v)\Big|_{y=\delta} - (f_c - f_v)\Big|_{y=-\delta} = 2\tau_w = 2\delta\phi. \tag{2.39}$$

Note that the dissipation term

$$\alpha\left((\rho u)\Big|_{j+1} - (\rho u)\Big|_{j}\right) \approx \alpha \Delta y \rho \frac{\partial(u)}{\partial y}, \tag{2.40}$$

assuming incompressibility, and that (2.40) is in the form of the viscous flux, leading to and effective viscosity of

$$\widetilde{\mu} = \mu + \alpha \rho \Delta x. \tag{2.41}$$

The first implication of this observation is that the shape of the velocity profile (2.38) is that of a more viscous fluid. From a practical perspective, this is generally not an issue since numerical dissipation is scaled to zero away from the boundary and in smooth flow regions. The second implication is that, since there is necessarily dissipation at the boundary, the conservation property (2.39) now strongly depends on the boundary dissipation value. Ordinarily, this is not taken to be a significant problem, as many high-fidelity CFD solutions are obtained in the limit as the wall spacing $\Delta y \to 0$. However, when making use of WMLES, it is never truly the case that $\Delta y \to 0$ (lest the primary advantage of wall modelling be lost). This means that at all relevant mesh spacings, this numerical error is present and affects the quality of turbulent and laminar boundary layers alike.

This problem has the potential to affect flow features in deeply unphysical ways, especially when the physical shear stress is relatively small compared to the numerical dissipation term, such as near separation points in compressible boundary layers. It is therefore critical to develop a numerical scheme that addresses this issue without losing numerical stability.

2.4.3 Summary

In summary, an immersed boundary method is desired that must:

- use sufficiently little dissipation so as not to corrupt turbulent scales in the solution,
- remain numerically stable at high Mach numbers, even when severe flow features such as shocks are present in the cells with immersed boundary treatment,
- use a boundary condition that minimises the effective dissipation flux at the boundary to avoid augmenting sensitive viscous boundary layers,
- ensure regularity in the boundary conditions experienced by the flow, and
- not introduce significant artificial noise into the solution.

Such a method is described in the latter sections of this chapter, as well as in van Noordt et al. [5].

2.5 Geometry Processing

2.5.1 General Algorithm

The variant of immersed-boundary treatment used for this work will be a ghost cell method. This choice was made primarily based on the author's past experiences with all three methods depicted in figure 2.1.

In this method, cells that depend on information inside the geometry are identified as "irregular" and are referred to as such. A ghost cell is then, approximately, any point within the complete stencil of an irregular point that resides inside the solid geometry. There are exceptions to this rule that will be mentioned later in this section.

The procedure for identifying irregular cells is shown in figure 2.5. Since all irregular cells must be in the vicinity of the boundary, an x-ray tracing procedure is employed to identify points on the boundary that are aligned with grid lines. In order to perform this procedure efficiently, a two-dimensional bounding box hierarchy is created in each Cartesian direction by projecting the

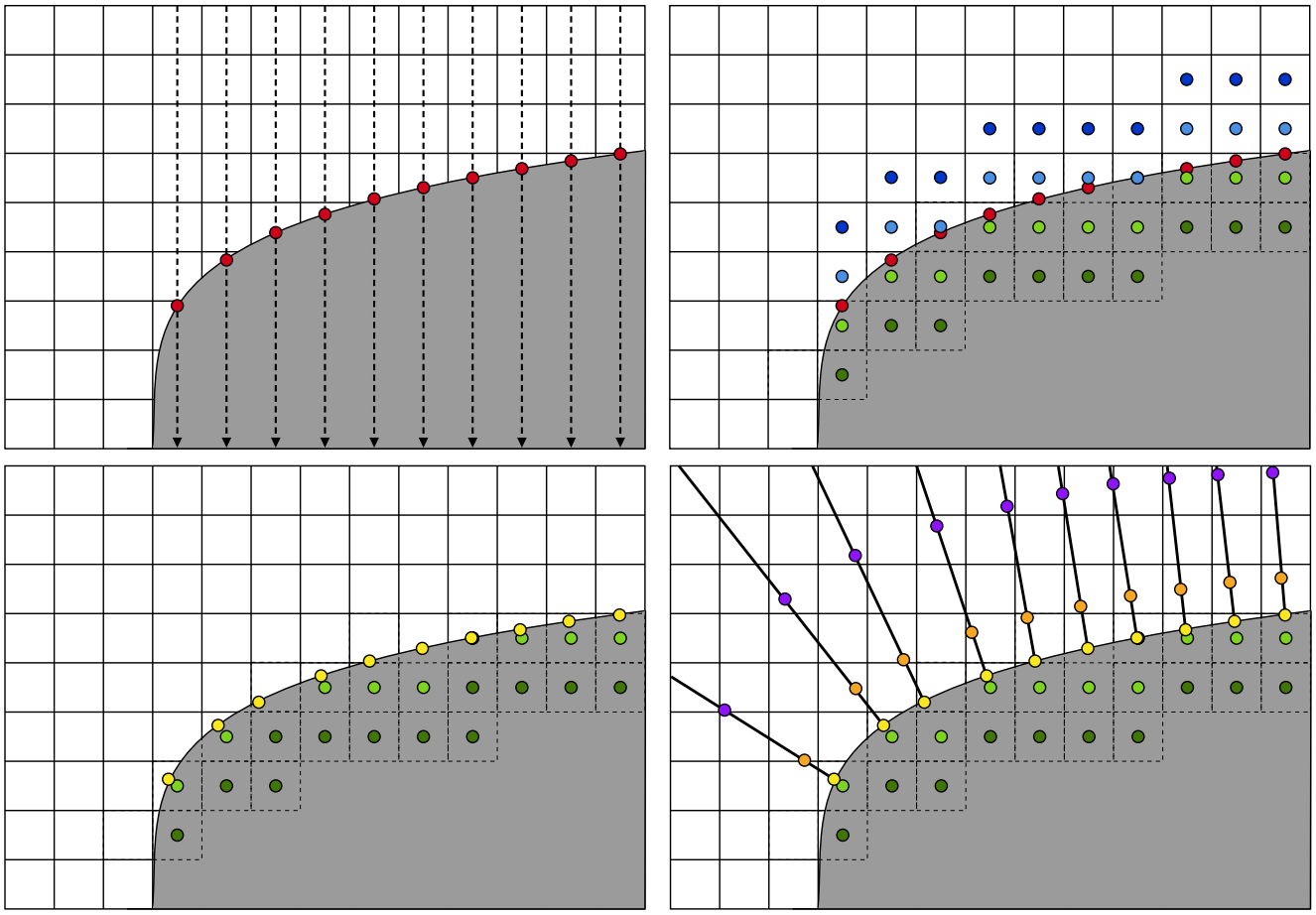


Figure 2.5: Immersed boundary identification procedure, depicted for the vertical direction. Top-left: axis-aligned x-ray tracing procedure produces grid-line intersection points (red). Top-right: identification of first-layer (light blue) and second-layer (dark blue) irregular cells and first-layer (light green) and second-layer (dark green) ghost cells. Bottom-left: identification of boundary points associated with ghost cells (yellow, shown only for the first-layer ghosts to avoid clutter). Irregular cells and grid-line intersection points removed for clarity. Bottom-right: near (orange) and far (violet) image points.

geometry normal to each axis. The bounding box hierarchy then allows a fast lookup for surface elements that are nearby candidates for x-ray tracing.

After these grid-line intersection points are identified, the associated irregular cells and ghost cells are identified, using the necessary number of layers for the numerical flux formulation being used (e.g., 1 layer for a 2^{nd}-order scheme, 2 layers for a 4^{th}-order scheme, and so on). Boundary points are then identified, associated with each ghost cell. In most cases, this is simply the nearest point to the ghost cell on the boundary. Finally, rays are cast from the boundary points, normal to the surface, and two sets of sampling points are generated. These sampling points are where the solution will be interpolated for the purpose of setting the numerical boundary conditions. The

use of two sampling points in discussed in section 2.6, and the details of the sampling procedure are discussed in section 2.7.

The procedure shown in figure 2.5 does not identify all ghost cells. The viscous flux requires the calculation of velocity gradients, meaning that some cells diagonal to an irregular point might not be identified as ghost cells. The full stencil for the flux derivative is shown in figure 2.6(a), separated into inviscid and viscous stencils. Figure 2.6(b) shows a ghost cell that is not identified by the x-ray tracing procedure. Therefore, after the first-pass ghost cell identification from figure 2.5, these cells are identified by looping over all irregular points and checking each point in the stencil.

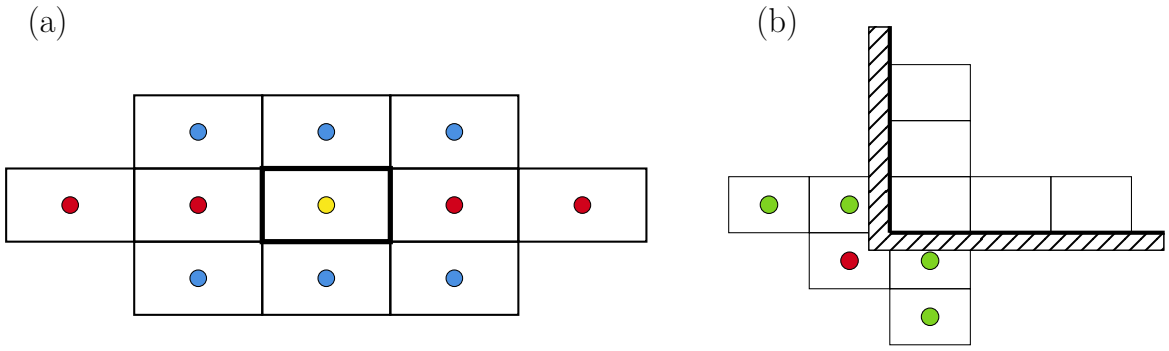


Figure 2.6: Left: stencil for viscous and inviscid flux derivatives at an arbitrary point in the x-direction, shown as the combination of the inviscid flux stencil (red points) and the viscous flux stencil (blue points). The yellow point indicates the cell to compute the flux derivative. Right: ghost cells identified by the x-ray procedure (green), and a ghost cell required by the viscous stencil that is not identified by the x-ray procedure (red).

2.5.2 Special Considerations

While the procedure described in section 2.5.1 works for the vast majority of irregular cells in the domain for most geometries, the procedure can fail in the cases of thin geometry and sharp corners.

Figure 2.7 shows a ghost cell (green) that has been identified at a sharp corner such as a stair-step or cube. As mentioned in section 2.5.1, the boundary points (yellow) must be identified to that a consistent boundary condition may be applied. However, in this case, there are two equidistant boundary points that are closest to the ghost cell, meaning that small perturbations of the geometry will result in different solution behaviour at the corner. It should be relatively clear

that the desired sampling point for the ghost cell depends on the direction in which the fluxes are being differentiated. The green ghost cell in figure 2.5 must also be counted twice; the first time for the vertical direction (with the boundary point on the upper surface), and the second time for the horizontal direction.

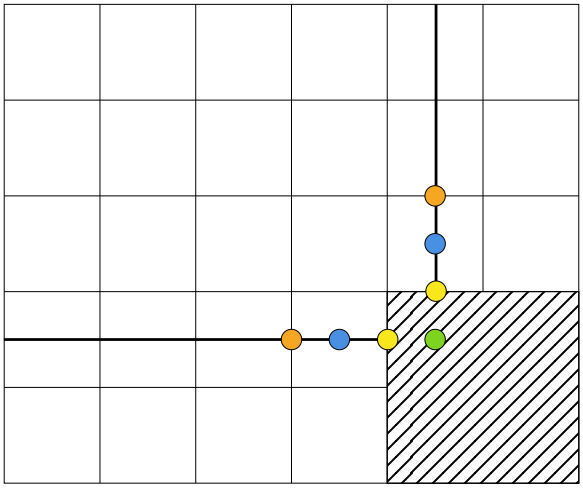


Figure 2.7: A single ghost cell at a sharp corner with two equally-viable boundary points. Colours used are identical to those in figure 2.5.

Thin geometry considerations are more complex. A number of problems are identified in figure 2.8(a). Point (1) is a ghost cell that resides inside the thin geometry, but the boundary point has been identified on the wrong side of the geometry, leading to an erroneous sampling point that will not enforce the correct boundary condition. Point (2) is a second-layer ghost cell that has protruded back into the fluid domain, additionally resulting in an incorrect boundary point and this an incorrect sampling point. Point (3) suffers from two issues. As with the other points, an incorrect sampling point residing inside the geometry has been identified, but the algorithm in section 2.5.1 checks for diagonal ghosts being inside the geometry. Using the basic procedure, this point is not identified as a ghost. To fix these issues, two procedures must be used to identify correct boundary points and more precisely identify diagonal ghost cells.

The procedure for finding boundary points is relatively simple. Each boundary point is initialised as the grid-line intersection point associated with the ghost cell, then an optimisation problem is solved to minimise the distance between the ghost cell and the boundary point. Critically, the boundary point is only allowed to slide across the surface of the geometry. This results

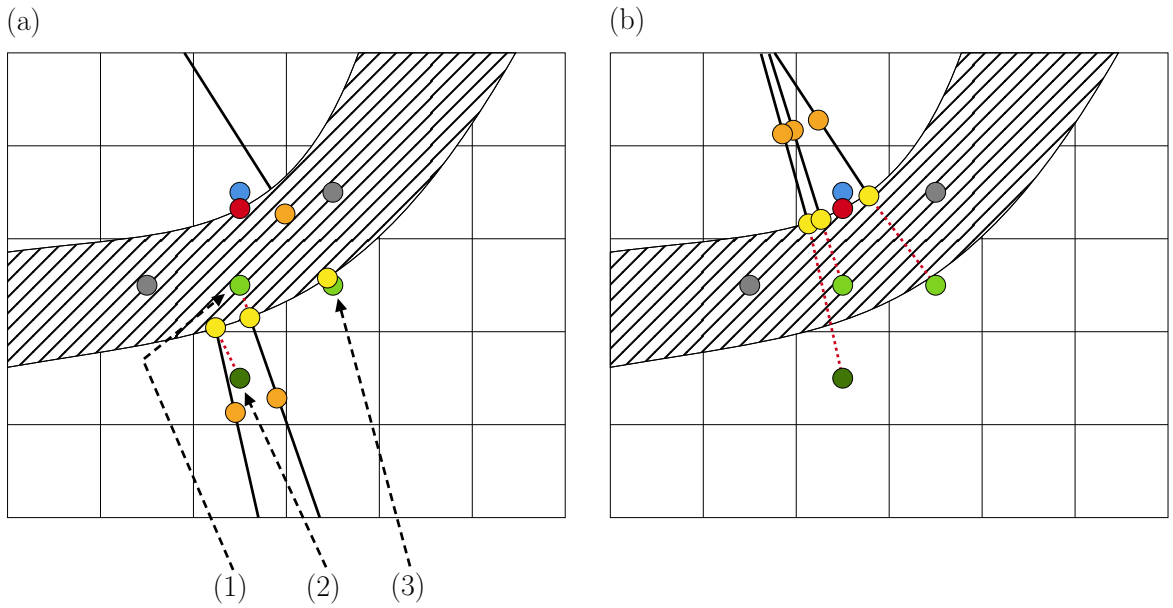


Figure 2.8: Left: degenerate behaviour of the irregular and ghost point identification in the case of a thin geometry feature. Colours used are identical to those in figure 2.5. Right: desired behaviour of the irregular and ghost point identification. Grey points are ghost cells that are not under consideration.

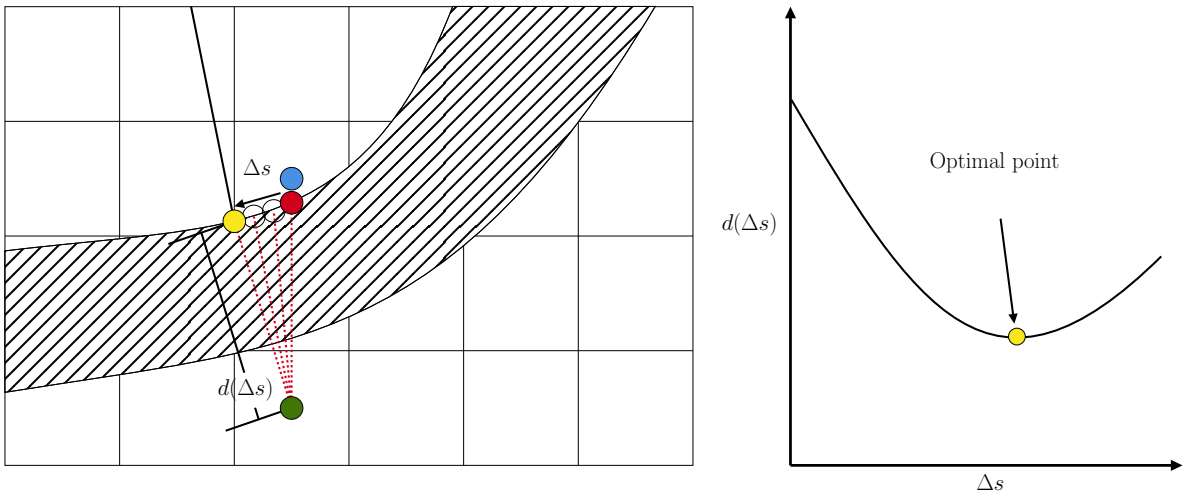


Figure 2.9: Optimal-point procedure beginning with the grid-line intersection point (red) and seeking the locally-optimal boundary point (yellow).

in a boundary point that is guaranteed to be on the same side of the geometry as the grid-line intersection point. At sharp corners, the boundary point is not allowed to slide over a sharp angle more than a specified tolerance, set to 15° for the rest of this work. This procedure is visualised in figure 2.9. Note that the distance-arc curve in figure 2.9 may be sharp in the case of sharp geometry. To summarize, boundary points are computed using the procedure in algorithm 1.

To more robustly identify diagonal ghost cells (since checking interiority is not sufficient), it is

Algorithm 1 Computation of closest boundary point

$f_{\text{opt}} \leftarrow f_{\text{intersect}}$ ▷ Initialize with the point where the x-ray hits
$d_{\text{opt}} \leftarrow \infty$ ▷ The smallest distance found between a boundary point and the ghost point
$f_{\text{new}} \leftarrow$ (invalid value) ▷ The "new" face gets an invalid value
while $f_{\text{new}} \neq f_{\text{opt}}$ **do**
    for each f in $\{f_{\text{opt}}\} \cup \{\text{neighbours}(f_{\text{opt}})\}$ **do** ▷ Neighbours share a triangle vertex
        if $\text{dist}(f, x_{\text{ghost}}) < d_{\text{opt}}$ **then** ▷ Compare distance of current face to ghost cell
            $d_{\text{opt}} \leftarrow \text{dist}(f, x_{\text{ghost}})$ ▷ Update the best distance
            $f_{\text{new}} \leftarrow f$
        end if
    end for $f_{\text{opt}} \leftarrow f_{\text{new}}$
end while

necessary to consider three cases, visualised in figure 2.10:

- elevated ghosts (red), elevated away from the surface relative to the irregular point (blue),
- peer ghosts (orange), on the same normal grid cell as the irregular point, and
- sunken ghosts, sunken into the geometry by one cell layer relative to the irregular point.

In each of these three cases, a ray-tracing test is performed along a specific path, and if there are any boundary intersection points on this path, then the point is considered a diagonal ghost. The ray-tracing test paths are also shown in figure 2.10. To the author's knowledge, these tracing

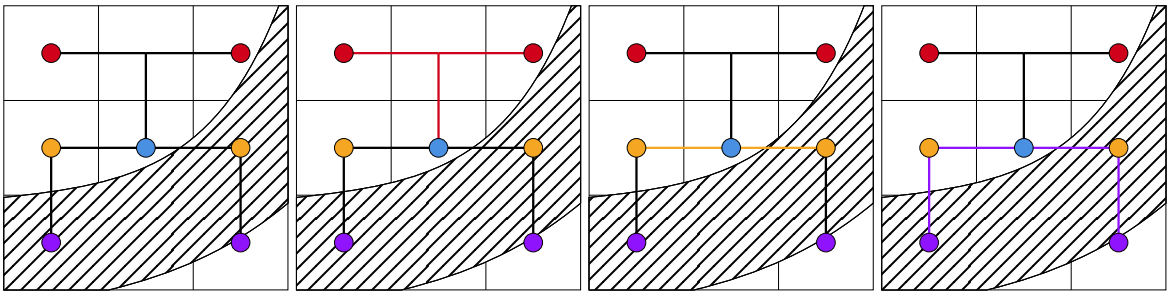


Figure 2.10: Ray tracing procedure to identify elevated (red), peer (orange) and sunken (violet) ghosts.

paths are the only ones that correctly identify diagonal ghosts in all simple test cases. These can be extended to three dimensions simply by applying them in each coordinate direction separately.

2.5.3 Upwind Scheme

The inviscid flux formulation used for this work is of the form

$$\hat{\mathbf{f}}_{\text{conv}} = (1-\alpha)\hat{\mathbf{f}}_{\text{cent}} + \alpha\hat{\mathbf{f}}_{\text{diss}}, \tag{2.42}$$

where $\hat{\mathbf{f}}_{\text{cent}}$ is a non-dissipative centred flux, $\hat{\mathbf{f}}_{\text{diss}}$ is the upwind-biassed WENO flux, and α is the dissipation switch that determines the contributions from the centred and upwind fluxes. To detect the presence of shocks, the value for α in (2.42) is computed according to the shock-sensing introduced by Ducros et al. [81], in the form

$$\alpha\Big|_{\text{volume}} = \frac{(\nabla\cdot\mathbf{u})^2 + |\nabla\times\mathbf{u}|^2}{(\nabla\cdot\mathbf{u})^2 + \left(\frac{u_\infty}{\delta_0}\right)^2} \tag{2.43}$$

where u_i are the velocity vector components, δ_0 is the reference boundary layer thickness, and u_∞ is the free-stream velocity. The factor u_∞/δ_0 is used to shield the boundary layer from introducing artificial dissipation near the wall. The dissipation coefficient is regularised by applying a simple filter across adjacent faces, e.g. $\hat{\alpha}_{j+1/2} = (\alpha_{j-1/2} + \alpha_{j+1/2} + \alpha_{j+3/2})/3.$, where $\hat{\alpha}$ is the regularised coefficient. The coefficient is set manually at the boundary.

The centred component of the total inviscid flux $\hat{\mathbf{f}}_{\text{cent}}$ is relatively simple to treat at the boundary, so all analysis in this section will focus on the development of the stabilising flux. The idea behind WENO schemes is reintroduced here to address special concerns for the boundary formulation.

The full flux derivative is written as the divided difference of adjacent numerical fluxes in the form

$$\frac{\partial\mathbf{f}}{\partial x} \approx \frac{\hat{\mathbf{f}}_{j+1/2} - \hat{\mathbf{f}}_{j-1/2}}{\Delta x}. \tag{2.44}$$

Two separate procedures must be specified to compute this quantity: a numerical flux function and a numerical flux reconstruction procedure. The numerical flux function used in this work is

the well-known (see e.g. Blazek [82]) Rusanov flux, given by

$$\hat{\mathbf{f}}_j^{\pm} = \frac{1}{2}\mathbf{f}(\mathbf{w}_j) \pm \frac{1}{2}\sigma \left(\left. \frac{\partial \mathbf{f}}{\partial \mathbf{w}} \right|_j \right) \mathbf{w}_j, \tag{2.45}$$

where $\mathbf{f}(\mathbf{w}_j)$ is the physical flux, $\mathbf{w}_j$ is the state vector in conservative variables, σ is the spectral radius of the local flux Jacobian, $\partial\mathbf{f}/\partial\mathbf{w}$, and all quantities are evaluated at the node j. Note that $\mathbf{w}$ denotes the conserved variables. A primitive function $\mathbf{h}(x)$ can be introduced following Shu and Osher [83] in the form

$$\mathbf{f}(x) = \frac{1}{\Delta x} \int_{x-\Delta x/2}^{x+\Delta x/2} \mathbf{h}(s)\, \mathrm{d}s, \tag{2.46}$$

such that an exact differentiation (or within the order of accuracy n of the flux reconstruction scheme) can be obtained

$$\frac{\partial \mathbf{f}}{\partial x} = \frac{\mathbf{h}_{j+1/2} - \mathbf{h}_{j-1/2}}{\Delta x}. \tag{2.47}$$

The primitive function, $\mathbf{h}$, in (2.46) is reconstructed from the nodal values of the numerical flux $\hat{\mathbf{f}}_j$ with an order of accuracy.

When reconstructing a face value with a specified order of accuracy, the face value is written as a linear combination of adjacent node values. However, when computing the derivative with two reconstructed values, the same order of accuracy is not necessarily achieved. The subtle switch to the reconstruction of the primitive function addresses this by including additional terms in the Taylor expansion.

Using the 3$^{\text{rd}}$-order formulation for the WENO reconstruction procedure, the face flux $\hat{\mathbf{f}}_{j+1/2}$ is written as the sum of the left- and right-travelling components

$$\hat{\mathbf{f}}_{j+1/2} = \hat{\mathbf{f}}^{+}_{j+1/2} + \hat{\mathbf{f}}^{-}_{j+1/2} \tag{2.48}$$

To avoid differentiating across discontinuities in the characteristic waves, each individual flux is reconstructed two times, and then the two reconstructions are weighted according to how much numerical irregularity there is over the extent of the stencil. In particular, the following formulations

are used for the m$^{\text{th}}$ component of the flux:

$$\hat{f}_{m,j+1/2} = \hat{f}^{+}_{m,j+1/2} + \hat{f}^{-}_{m,j+1/2}, \tag{2.49}$$

where

$$\hat{f}^{+}_{m,j+1/2} = w^{+}_{0,m}\left(-\frac{1}{2}\hat{f}^{+}_{m,j-1} + \frac{3}{2}\hat{f}^{+}_{m,j}\right) + w^{+}_{1,m}\left(\frac{1}{2}\hat{f}^{+}_{m,j} + \frac{1}{2}\hat{f}^{+}_{m,j+1}\right) \tag{2.50}$$

and

$$\hat{f}^{-}_{m,j+1/2} = w^{-}_{0,m}\left(\frac{1}{2}\hat{f}^{-}_{m,j} + \frac{1}{2}\hat{f}^{-}_{m,j+1}\right) + w^{-}_{1,m}\left(\frac{3}{2}\hat{f}^{-}_{m,j+1} - \frac{1}{2}\hat{f}^{-}_{m,j+2}\right), \tag{2.51}$$

where $w^{\pm}_{n,m}$ are the computed weights satisfying

$$w^{\pm}_{0,m} + w^{\pm}_{1,m} = 1. \tag{2.52}$$

In order to choose the weights for each stencil, a smoothness parameter is computed for each stencil. For a general WENO scheme, this is done by integrating each derivative approximation over the k^{th} stencil as

$$\alpha^{\pm}_{k} = \sum_{n=1}^{k-1} \int_{x_{i-1/2}}^{x_{i+1/2}} \Delta x^{2n-1} \left(\frac{\partial^{n} \hat{f}^{\pm}_{m}}{\partial x^{n}}\right)^{2} dx, \tag{2.53}$$

and then computing the weights as

$$\beta^{\pm}_{k} = \frac{c_k}{(\varepsilon + \alpha_k)^2}, \quad \text{and} \quad w^{\pm}_{k} = \frac{\beta^{\pm}_{k}}{\sum \beta^{\pm}}. \tag{2.54}$$

The coefficients c_k are the "optimal" coefficients that result in maximal order of accuracy when $\beta_k \approx 1$. Note that the constant $\varepsilon = 10^{-16}$ to prevent division by zero. In the case of a 3$^{\text{rd}}$-order WENO scheme, the optimal coefficients are 1/3 and 2/3, at which point maximal nominal order-of-accuracy is attained. It should be noted that in the 3$^{\text{rd}}$-order case, the smoothness indicators (2.54) for every stencil is simply

$$\alpha_k = \left(\hat{f}_R - \hat{f}_L\right)^2, \tag{2.55}$$

since all candidate stencils have just two points.

When simulating flows with especially high Mach numbers, additional stabilisation is needed

for this procedure. The strategy employed is to perform the characteristic transform on all numerical fluxes before applying the reconstruction procedure. In particular, before the reconstruction procedure is performed, each flux is premultiplied as

$$\hat{f}^{+}_{m,j+1/2} = V^{-1}\left(w_0^{+}\left(-\frac{1}{2}V\hat{f}^{+}_{j-1} + \frac{3}{2}V\hat{f}^{+}_{j}\right) + w_1^{+}\left(\frac{1}{2}V\hat{f}^{+}_{j} + \frac{1}{2}V\hat{f}^{+}_{j+1}\right)\right) \tag{2.56}$$

and

$$\hat{f}^{-}_{m,j+1/2} = V^{-1}\left(w_0^{-}\left(\frac{1}{2}V\hat{f}^{-}_{j} + \frac{1}{2}V\hat{f}^{-}_{j+1}\right) + w_1^{-}\left(\frac{3}{2}V\hat{f}^{-}_{j+1} - \frac{1}{2}V\hat{f}^{-}_{j+2}\right)\right), \tag{2.57}$$

where V is the eigenvector matrix from (2.5) evaluated at $j + 1/2$ using any consistent approximation. The critical advantage of this method is that the smoothness indicators are computed consistently with the characteristics of the inviscid equations (see ref. [66]), and this method provides significantly enhanced stability at high Mach numbers at the cost of additional matrix operations.

Solution artefacts are typically introduced when the numerical discretization is not regular. This is typically the case when the numerical dissipation depends on two ghost cells, with the preference being dependence on only a single ghost cell. With this observation in mind, and using the numbering in figure 2.4, two modifications will be made to the WENO reconstruction procedure:

- the reconstruction in (2.50) must be modified so that the reconstructed flux $\hat{f}^{+}_{-1/2}$ does not to include the contribution from the second-layer ghost cell at $j = -2$, and
- the reconstruction in (2.51) will be modified such that $\hat{f}^{+}_{1/2}$ does not include any contribution from $j = -1$.

These two choices are made to minimise the effect of the numerical boundary condition and the irregularity of the numerical discretization at the boundary. Note that both of the modifications above will also need to be made for the smoothness indicators over the irregular stencils.

The modified reconstruction for $\hat{f}^{+}_{m,-1/2}$ are computed as

$$\hat{f}^{+}_{m,-1/2} = w^{+}_{0,m}\left(\frac{1}{6}\hat{f}^{+}_{m,-1} + \frac{7}{6}\hat{f}^{+}_{m,0} - \frac{1}{3}\hat{f}^{+}_{m,1}\right) + w^{+}_{1,m}\left(\frac{1}{2}\hat{f}^{+}_{m,-1} + \frac{1}{2}\hat{f}^{+}_{m,0}\right), \tag{2.58}$$

and the reconstruction for $\hat{f}^{-}_{m,-1/2}$ is given as

$$\hat{f}^{-}_{m,1/2} = w^{-}_{0,m}\left(\frac{5}{2}\hat{f}^{-}_{m,0} - \frac{5}{2}\hat{f}^{-}_{m,1} + \frac{1}{2}\hat{f}^{-}_{m,2}\right) + w^{-}_{1,m}\left(\frac{3}{2}\hat{f}^{-}_{m,0} - \frac{1}{2}\hat{f}^{-}_{m,1}\right). \tag{2.59}$$

The modified smoothness indicators are given as

$$\alpha^{+}_{m,1} = \left(\hat{f}_j - \hat{f}_{j-1}\right)^2, \quad \alpha^{+}_{m,2} = \left(-\frac{5}{3}\hat{f}_{j-1} + \frac{7}{3}\hat{f}_j - \frac{2}{3}\hat{f}_{j+1}\right)^2,$$

$$\alpha^{-}_{m,1} = \left(\hat{f}_{m,j+1} - \hat{f}_{m,j}\right)^2, \quad \text{and} \quad \alpha^{-}_{m,2} = \left(-3\hat{f}_{m,j} + 5\hat{f}_{m,j+1} - 2\hat{f}_{m,j+2}\right)^2. \tag{2.60}$$

The motivation for these choices will be explained in section 2.10. The centred flux component is not modified in any way at the boundary and is allowed to depend on both ghost cell states. Values of α in (2.43) are required to assume a minimal value when computing fluxes for all irregular cells.

2.6 Boundary Conditions

As noted in section 2.4.2, the presence of numerical dissipation at the boundary acts as a virtual shear stress that affects the solution. When using a wall model, the viscous stress at the wall is no longer computed using a numerical no-slip condition, but rather taken as the solution to an iterative system of differential equation (see section 2.8 and chapter 3). That is, the shear stress (and heat transfer) is available independently of the numerical boundary condition. For the sake of conservation and accurate boundary layer growth, there must still be no convective flux contribution in the wall-normal direction, i.e. it is necessary to enforce the no-penetration boundary condition. Note that this condition is relaxed in other wall modelling approaches such as the dynamic slip method (see e.g. Bose and Moin [84]) or the virtual wall method (see e.g. Gao et al. [85]).

Using the example approximation (2.34), the no-penetration condition is enforced by virtue of the fact that the boundary-normal v velocity is negated in the ghost cell. This means that the ghost cell values of the tangential velocity u can be freely chosen to minimise the effect of the necessary added dissipation. To see how this yields a method for setting the tangential velocity boundary condition, it is necessary to derive a differential operator to which the numerical dissipation is consistent, as was done in (2.40). Ignoring the physical flux component of (2.45) and assuming optimal WENO weights, the dissipation flux reconstruction is (removing component subscripts) given by

$$\hat{f}_{j+1/2} = \hat{f}^{+}_{j+1/2} + \hat{f}^{-}_{j+1/2} = \frac{1}{3}\left(-\frac{1}{2}\left(\frac{1}{2}\sigma w\right)_{j-1} + \frac{3}{2}\left(\frac{1}{2}\sigma w\right)_{j}\right) + \frac{2}{3}\left(\frac{1}{2}\left(\frac{1}{2}\sigma w\right)_{j} + \frac{1}{2}\left(\frac{1}{2}\sigma w\right)_{j+1}\right)$$

$$+\frac{2}{3}\left(\frac{1}{2}\left(-\frac{1}{2}\sigma w\right)_{j} + \frac{1}{2}\left(-\frac{1}{2}\sigma w\right)_{j+1}\right) + \frac{1}{3}\left(\frac{3}{2}\left(-\frac{1}{2}\sigma w\right)_{j+1} - \frac{1}{2}\left(-\frac{1}{2}\sigma w\right)_{j+2}\right), \tag{2.61}$$

leading to the approximation

$$-\frac{1}{12}(\sigma w)_{j-1} + \frac{5}{12}(\sigma w)_{j} + \frac{1}{6}(\sigma w)_{j+1} - \frac{1}{6}(\sigma w)_{j} - \frac{5}{12}(\sigma w)_{j+1} + \frac{1}{12}(\sigma w)_{j+2}$$

$$= -\frac{1}{12}(\sigma w)_{j-1} + \frac{1}{4}(\sigma w)_{j} - \frac{1}{4}(\sigma w)_{j+1} + \frac{1}{12}(\sigma w)_{j+2}. \tag{2.62}$$

Taking the divided difference gives

$$\frac{1}{\Delta x}\left(\hat{f}_{j+1/2} - \hat{f}_{j-1/2}\right) = \frac{1}{12\Delta x}\Big((\sigma w)_{j-2} - 4(\sigma w)_{j-1} + 6(\sigma w)_{j} - 4(\sigma w)_{j+1} + (\sigma w)_{j+2}\Big)$$

$$\approx \frac{1}{12}\Delta x^{3}\frac{\partial^{4}(\sigma w)}{\partial x^{4}}. \tag{2.63}$$

The expression in (2.63) is the explicit dissipation operator in the case where the optimal WENO weights are selected. In cases where the optimal weights are not chosen (such as near discontinuities), the dissipation operator can be found to be a linear combination of operators of the form

$$D_n = C(-1)^{n}\Delta x^{2n-1}\frac{\partial^{2n}(\sigma w)}{\partial x^{2n}}. \tag{2.64}$$

A relatively simple approach is taken in an effort to make this term approximately zero at the boundary. Since the order of differentiation in (2.64) is even, it must vanish for approximately linear arguments. Therefore, the numerical boundary condition to be used is, while enforcing the no-penetration boundary condition, to simply extrapolate the best available linear approximation of the solution. For simplicity of analysis, the wavespeed σ is assumed to be approximately constant near the boundary, and therefore the linear extrapolation is used for the conservative variables.

As shown in figure 2.5, two image points are generated from every ghost point. The analysis in this section gives one reason why this is the case: linear extrapolation requires two points of reference since the boundary condition is not specified. Importantly, the far sampling point distance scales with the thickness of the boundary layer since this data is used as an input to the viscous wall model (see Larsson et al. [86] and chapter 3 for further details). By contrast, the closer interpolation point distance scales with the local grid spacing to be consistent with the numerical discretization. Note that scaling the large sampling distance with the thickness of the local boundary layer is difficult to implement in practice and is an active area of research.

2.7 Flowfield Sampling

In order to avoid solution artefacts at the immersed boundary, care must be taken when sampling the flowfield for setting the numerical boundary condition. The methods used for sampling have generally been developed moreso through experimentation than analysis. For completeness, the following observations have been made when experimenting with these procedures:

- the sampling operator for the far sampling point has less of an effect on the solution quality than that for the close sampling point,
- the sampling operator for the close point performs favourably when using a large cloud over a small one,
- extending the close point too far into the flowfield generates noise in the solution that has potential to artificially transition the flow to turbulence,

- using a biassed cloud for the close point has the potential to affect shock locations and separation locations, and
- the use of a large sample cloud for either point has a significant effect on computational performance.

While these observations are not justified through any prior analysis, they have not been reported in any relevant literature to the author's knowledge.

The far sampling point is sampled using a straightforward multilinear interpolation, where linear interpolation is used for all three coordinate directions. This is shown in figure 2.11(a). If any of the stencil points (magenta) lie inside the geometry, the procedure for the close point is deferred to.

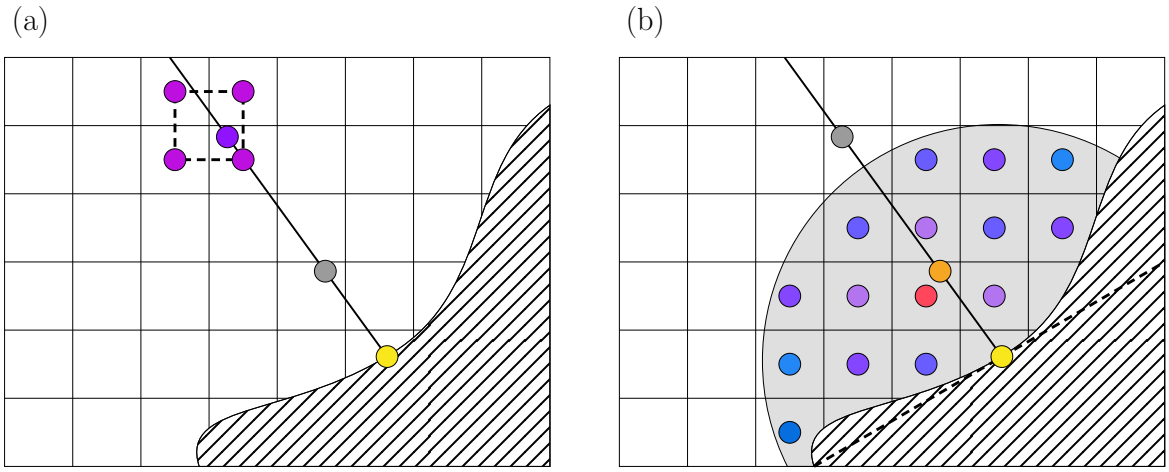


Figure 2.11: Left: sampling operation for far sampling point. Right: sampling operation for close sampling point. Colours used are identical to those in figure 2.5

The close point is sampled by identifying a range of candidate stencil points nearby the boundary. Candidate stencil points are required to lie in the semi-infinite space in the direction of the surface-normal vector and within a distance less than the distance of the far sampling point. The closest 20 points are then selected based on the distance in index space, and the sample coefficients are generated by solving a least-squares reconstruction problem, following ref. [87].

2.8 Viscous Boundary Treatment

To obtain the viscous fluxes at the immersed boundary, a wall model is used. As previously mentioned, this choice relaxes the strict resolution requirements for turbulent flows. For the validation cases presented later in this chapter, the equilibrium wall model is employed. The solution is sampled at a distance that is approximately 10-20% of the way through the viscous boundary layer. This requirement is discussed in Larsson et al. [86] and chapter 3. The sampled flow velocity is decomposed into tangential u_f and normal u_n components based on the local surface normal. The normal component is discarded for the wall model. The equilibrium wall model consists of the set of ordinary differential equations

$$\frac{\partial}{\partial y}\left(\left(\mu+\mu_t\right)\frac{\partial u}{\partial y}\right)=0, \quad u(0)=0, \quad u(y_f)=u_f \tag{2.65}$$

and

$$\frac{\partial}{\partial y}\left(\frac{\gamma R}{\gamma-1}\left(\frac{\mu}{\mathrm{Pr}}+\frac{\mu_t}{\mathrm{Pr}_t}\right)\frac{\partial T}{\partial y}\right)+\frac{\partial}{\partial y}\left(\left(\mu+\mu_t\right)u\frac{\partial u}{\partial y}\right)=0, \quad T(0)=T_{\text{wall}}, \quad T(y_f)=T_f. \tag{2.66}$$

The boundary conditions for (2.65) is how the no-slip condition is realised within this computational framework. Additionally, the wall temperature boundary condition must be specified in order to solve (2.66). Alternatively, the condition

$$\left.\frac{\partial T}{\partial y}\right|_{y=0}=0 \tag{2.67}$$

can be enforced for an adiabatic wall. The solution to (2.65) and (2.66) are used in lieu of the flowfield to yield the viscous flux boundary condition, shown in figure 2.12.

For high-speed flows, the calculation of the eddy viscosity μ_t is critical in obtaining accurate numerical solutions. There is extensive literature on this topic, and an extended discussion is given in chapter 3. For the validation cases here, the semilocally-scaled formulation is used:

$$\mu_t=\kappa\mu y^*\left(1-e^{-y^*/A^+}\right)^2, \tag{2.68}$$

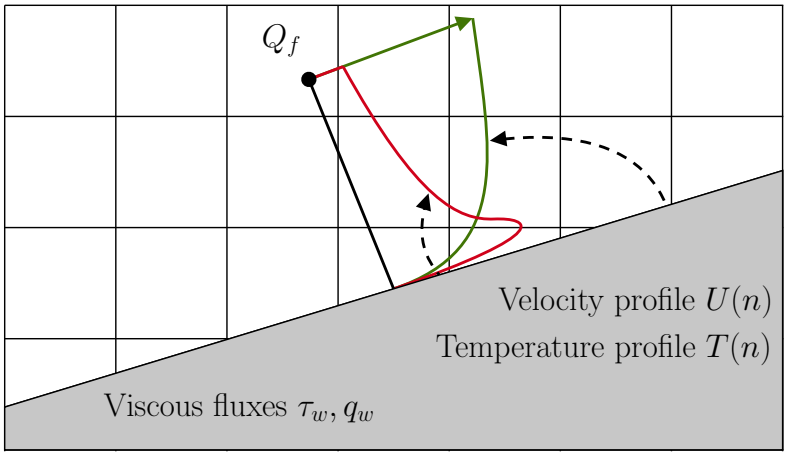


Figure 2.12: Schematic of the wall model solution as it relates to the solid boundary.

where $\kappa = 0.41$ is the von Kármán constant and $A^+ = 17$ is chosen to be the approximate location of the base of the log layer. The non-dimensional semi-locally-scaled wall-normal distance is given by

$$y^* = y\sqrt{\rho(y)\tau_w}/\mu(y). \tag{2.69}$$

The viscosity is calculated consistently with the outer solution, and is given by Sutherland's law

$$\mu(T) = \mu_{\text{ref}}\frac{T^{3/2}}{T + T_{\text{ref}}}. \tag{2.70}$$

After the viscous flux is obtained at each sampling location, it must be used to compute the viscous flux derivative. As shown in figure 2.13, the wall model flux must be reconstructed on the grid-line intersection point (red). This is done with a simple average of nearby shear stress values (note that the details of this have little effect on the solution). Then, to reduce dependence on ghost cell information, the viscous face flux is fully reconstructed using a procedure with similar stencil selection shown in figure 2.11.

In an effort to increase the regularity of the numerical discretization, the truncation error coefficients of the viscous flux interpolation are matched using the reconstruction procedure. This has the advantage that the largest term in the truncation error has a smooth distribution, reducing numerical irregularities. As with the inviscid flux, the viscous flux $\mathbf{g}$ is computed in conservative

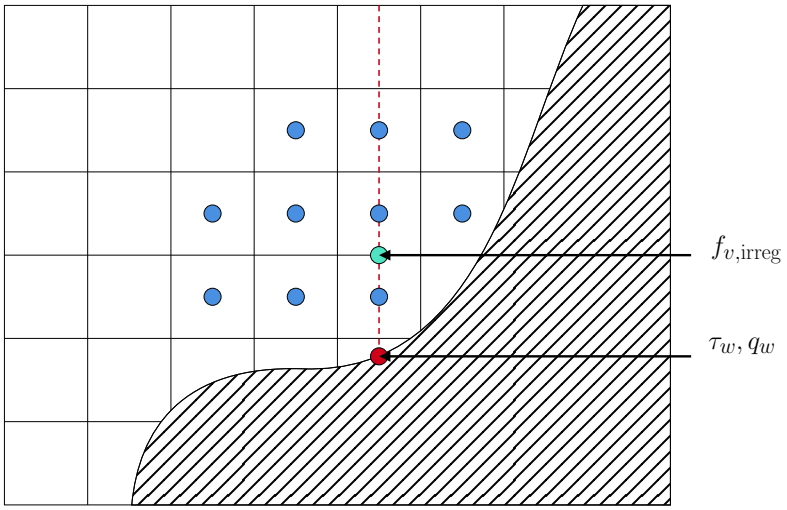


Figure 2.13: Procedure for computing the irregular flux derivative using wall model information.

form as

$$\left.\frac{\partial \mathbf{g}}{\partial x}\right|_i \approx \frac{1}{\Delta x}\left(\hat{\mathbf{g}}_{i+1/2} - \hat{\mathbf{g}}_{i-1/2}\right). \tag{2.71}$$

For the discussion here, the viscous flux at a grid point $x_{i\pm1/2,j}$ is considered such that the derivative in x-direction can be taken along the grid line and the derivative in y and z directions are computed in the tangential planes normal to the grid line. The numerical treatment of the viscous fluxes relies on error cancellation when applying identical stencils at the left and right faces. In the vicinity of the immersed boundary, the symmetry gets broken and, therefore, in order to maintain the formal order of accuracy the viscous fluxes need to be designed in a way to achieve this error cancellation. The discretization stencils for interior operators are briefly introduced as the operators at the irregular stencils need to match the leading term of the truncation error to maintain formal second-order accuracy. The variable ϕ will be used as a placeholder for the physical variable, i.e., velocity, temperature and viscosity. The x-derivative at the mid point $x_{i+1/2}$ is simply taken to be a second-order accurate centred difference operator:

$$\left.\frac{\partial \phi}{\partial x}\right|_{i+1/2,j} = \frac{\phi_{i+1,j} - \phi_{i,j}}{\Delta x} - \frac{1}{24}\Delta x^2 \phi_{xxx} + \mathcal{O}(\Delta x^3). \tag{2.72}$$

The primitive variable reconstruction operator at the mid point is simply

$$\phi_{i+1/2,j} = \frac{1}{2}\Big(\phi_{i,j} + \phi_{i+1,j}\Big) + \frac{1}{8}\Delta x^2 \phi_{xx} + \mathcal{O}(\Delta x^3). \tag{2.73}$$

A combination of these two procedures is used to construct the derivatives in y-direction:

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

$$+\frac{1}{6}\Delta y^2 \phi_{yyy}\Big|_{i+1/2,j} + \frac{1}{8}\Delta x^2 \phi_{yxx}\Big|_{i+1/2,j} + \mathcal{O}(\Delta y^3, \Delta x^3, \Delta x \Delta y^2). \tag{2.74}$$

The basis of the viscous boundary treatment is to match the relevant properties of the interior scheme at all irregular faces, particularly the order-of-accuracy and the leading coefficient of the truncation error. The motivation for matching the truncation error coefficient is primarily to maintain the formal order of accuracy (through truncation error cancellation) and to produce a continuous distribution of error, which reduces the irregularity of the total error distribution. Once the stencil points are identified the stencil coefficients $c_{i,j}$ are constructed to approximate the linear operator $\mathcal{L}$ within truncation error constraints

$$\mathcal{L}(\phi) = \sum_{i,j \in J} c_{i,j}\phi_{i,j} + k_1 \Delta x^2 + k_2 \Delta y^2 + \mathcal{O}(\Delta x^3, \Delta x^2 \Delta y, \Delta x \Delta y^2, \Delta y^3). \tag{2.75}$$

Given (2.72), (2.73), and (2.74), the constants k_1 and k_2 above take on the values $k_1 = -1/24$, $k_2 = 0$ for $\mathcal{L} \equiv \partial/\partial x$, $k_1 = 1/8$, $k_2 = 0$ for $\mathcal{L} \equiv 1$, and $k_1 = 1/8$, $k_2 = 1/6$ for $\mathcal{L} \equiv \partial/\partial y$. Note that extension to three dimensions is trivial.

For the final viscous flux derivative term, the differentiation procedure is given as

$$\left.\frac{\partial \mathbf{g}}{\partial x}\right|_i = c_w \hat{\mathbf{g}}_w + \sum_{k=1}^{3} c_k \hat{\mathbf{g}}_{i-1/2+k} + \mathcal{O}(\Delta x^2), \tag{2.76}$$

where $\hat{\mathbf{g}}$ is the numerical viscous flux, $\hat{\mathbf{g}}_w$ is the viscous flux applied at the grid line intersection (in this case, using a wall-model viscous flux), c_k's are the finite-difference coefficients for a second-order accurate finite-difference operator. The differentiation coefficients satisfy the linear equation

in the form

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ \tilde{x}_0 & \tilde{x}_1 & \tilde{x}_2 & \tilde{x}_3 \\ \tilde{x}_0^2 & \tilde{x}_1^2 & \tilde{x}_2^2 & \tilde{x}_3^2 \end{pmatrix} \begin{pmatrix} c_w \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \tag{2.77}$$

where $\tilde{x}_k = (x_{i+k-1} - x_i)/\Delta x$ represents a index-based coordinate to account for the distance of the stencil point to irregular grid point x_i and $\tilde{x}_0 = (x_b - x_i)/\Delta x$ is distance from the irregular grid point to the immersed boundary. The overdetermined system of equations is solved by employing the Penrose pseudo-inverse, similarly to the procedure described in section 2.7.

2.9 Method of Manufactured Solutions

The method of manufactured solutions (MMS) can be used to verify the implementation of all of the procedures described above in this chapter and to evaluate their achieved order of accuracy (as compared to the nominal order of accuracy). An approximation $\hat{F}$ of F has nominal order of accuracy n if

$$||\hat{F} - F||_l \sim \Delta^n \tag{2.78}$$

where $||\cdot||_l$ is any consistent l-norm over the codomains of $\hat{F}$ and F, and $\hat{F} \to F$ as $\Delta \to 0$. In practice, this is done by investigating the behaviour of

$$||\hat{F}(\phi) - F(\phi)||_l \tag{2.79}$$

as $\Delta \to 0$. In particular, choosing $l = 2$ gives global order of accuracy, while choosing $l = \infty$ gives the minimal local order of accuracy.

This procedure was conducted for the inviscid flux derivatives, viscous flux derivatives, solution sampling, and ghost-cell filling. The geometry used for this test was a sphere of radius r. The

flowfield is initialised with the following test function:

$$\begin{pmatrix} P \\ T \\ u_r \\ u_\theta \\ u_\phi \end{pmatrix} = \begin{pmatrix} 15 + \cos(r-1) + \frac{1}{6}(r-1)^3 \\ 15 + \cos(r-1) + \frac{1}{6}(r-1)^3 \\ 0 \\ 4\sin(r-1) + \frac{1}{2}(r-1)^2 \\ 0 \end{pmatrix}, \tag{2.80}$$

where (r, θ, ϕ) are the radius, azimuthal angle, and zenith angle in polar coordinates. These functions were chosen to match the boundary conditions in section 2.6 and to ensure that nominal order of accuracy was not increased artificially by using a test function with a zero coefficient in its Taylor series expansion.

Table 2.1: Interpolation and ghost-cell procedure convergence rates ($l = 2/l = \infty$).

Procedure	P	U	V	W	T
Interpolation	2.658/2.300	2.998/2.771	2.994/2.831	2.997/2.712	2.659/2.298
Ghost Cell	2.926/1.988	2.958/1.970	2.960/1.960	2.956/1.951	2.838/1.856

Table 2.2: Convective and viscous operator convergence rates ($l = 2/l = \infty$).

Procedure	Cont.	x-Mom.	y-Mom.	z-Mom.	Eng.
Convective	2.686/1.967	2.727/2.107	2.725/2.117	2.981/2.023	2.650/2.044
Viscous	N.A.	2.377/2.014	2.320/2.118	2.301/2.082	2.101/2.049

The convergence rates for the ghost-cell filling and solution sampling can be found in table 2.1, and the convergence rates for the inviscid and viscous operators can be found in table 2.2. Note that the boundary discretization is 2^{nd}-order at the immersed boundary.

2.10 Inviscid Stability Analysis

Note that this analysis has been adapted from ref. [5].

As discussed in section 2.2, the inviscid component of the compressible Navier-Stokes equations is a hyperbolic PDE that can be modelled by the one-dimensional scalar advection equation (2.81):

$$\frac{\partial \phi}{\partial t} + a\frac{\partial \phi}{\partial x} = 0, \tag{2.81}$$

where ϕ is a real-valued scalar transported at wave speed a. Many developments of classical numerical methods take a similar approximation as a starting point [88, 89].

The spatial operator $\partial(\cdot)/\partial x$ can be discretized as a linear operator D acting on the discrete solution ϕ. Here, D includes both the interior discretization and boundary conditions. Thus, (2.81) can be spatially discretized as

$$\frac{\partial \phi}{\partial t} + a\frac{1}{\Delta x} D\phi = 0. \tag{2.82}$$

For linear problems like this, homogeneous boundary conditions can be assumed without loss of generality.

Fully discretizing (2.82) in time leads to the system

$$\phi^{(n+1)} = A\phi^{(n)}, \tag{2.83}$$

where, for example, an explicit 4^{th}-order Runge-Kutta scheme approximates the update matrix A as a 4^{th}-order expansion of $\exp(-\gamma D)$:

$$e^{-\gamma D} \approx A = I - \gamma D + \frac{1}{2!}\gamma^2 D^2 - \frac{1}{3!}\gamma^3 D^3 + \frac{1}{4!}\gamma^4 D^4, \tag{2.84}$$

where $\gamma = a\Delta t/\Delta x$ is the Courant number, assumed to be 1. Classically, stability for the fully discretized system in (2.83) is ensured as long as the spectral radius satisfies (2.31). However, as shown by Trefethen [64] and in the boundary analysis by Brehm [90], the non-normality of the eigenvector matrix of A can result in finite-time instability of the discrete system in (2.83), even when condition (2.31) holds.

A more practical condition for finite-time stability, as explained in section 2.4.1, is given by

$$\text{dist}\Big(\Lambda_\varepsilon\{A\}, S\Big) = \mathcal{O}(\varepsilon, \Delta t), \tag{2.85}$$

where S represents the stability region, defined as

$$S = \big\{w \in \mathbb{C} : |w| \leq 1\big\}, \tag{2.86}$$

i.e. the unit disk in the complex plane, and Λ_ε denotes the ε-pseudospectrum of the operator A. This is expressed as the union of eigenvalues:

$$\Lambda_\varepsilon\{A\} = \bigcup_{||M||=\varepsilon} \Big\{\lambda \in \mathbb{C} : \det(\lambda I - A + M) = 0\Big\}, \tag{2.87}$$

for any perturbation matrix M with norm $||M|| = \varepsilon$.

Even in a linear context, fully capturing the stability properties of the proposed operators is challenging. Therefore, pseudospectral analysis is presented for several cases, each defined by the operator:

$$D = (I - B)D_c + B\Big(S_1 D^{(1),\pm} u + S_2 D^{(2),\pm} u\Big), \tag{2.88}$$

where $B = \mathrm{diag}(\alpha_0, \alpha_1, \ldots)$ controls the influence of the upwind operator, Dc is the 4^{th}-order centred operator reduced to 2^{nd}-order at the boundary, $S1/2 = \mathrm{diag}(\omega_0^{(1/2)}, \omega_1^{(1/2)}, \ldots)$ selects the finite difference stencils, and $D^{(1/2),\pm}$ are the upwind finite difference operators associated with the stencils from (2.50) and (2.51). The left-biassed operator D_u^+ is used when the wave speed a from (2.81) is positive, while the right-biassed operator D_u^- is used for negative a.

The following cases are examined for both $a > 0$ and $a < 0$:

$D1$: $B = I$, with $S1/2$ biassed in the first cell toward the domain's first face: $S_1 = \mathrm{diag}(0, 1/3, 1/3, \ldots)$ and $S_2 = \mathrm{diag}(1, 2/3, 2/3, \ldots)$,

D_2: Similar to D_1, but with $S_1 = \mathrm{diag}(1, 1/3, 1/3, \ldots)$ and $S_2 = \mathrm{diag}(0, 2/3, 2/3, \ldots)$,

D_3: Similar to D_1, but with $B = \mathrm{diag}(\exp(-j))$ to represent the marginally stable upwind operator using the three-point stencils from (2.58) and (2.59) while incorporating the centred operator.

This results in six operators for analysis.

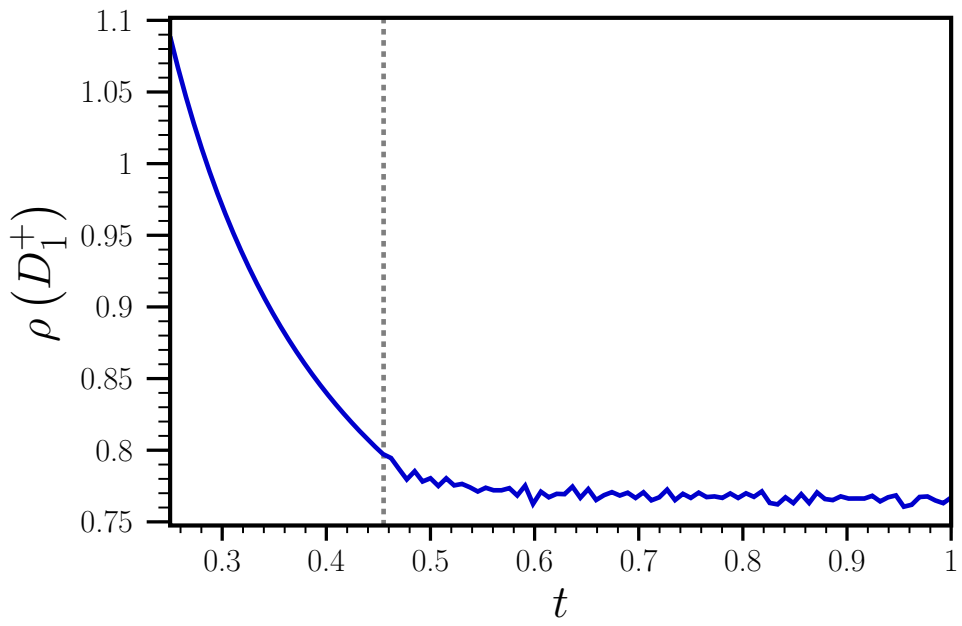


Figure 2.14: Variation of spectral radius of the operator D_1^+ with the stencil parameter t in (2.89), with eigenvalue saturation indicated.

The operators $D_1^{\pm}$ are designed to examine the stability of the full upwind operator when the 3-point candidate stencils in (2.58) and (2.59) are active. This also includes a brief analysis of the effect of reconstruction coefficients. Notably, the 3-point stencil in (2.58) appears biassed in the "wrong" direction, with a positive wave speed indicating a right-moving characteristic, yet the stencil is right-biassed. However, it will soon be shown that this configuration is stable with the appropriate choice of stencil coefficients.

After applying accuracy constraints for the desired order, the finite-difference coefficients C^+ from the 3-point stencil in (2.58) can be parameterized as:

$$C^+(t) = \left\{\frac{1}{2} - \frac{1}{6t}, \frac{1}{2} + \frac{1}{3t}, \frac{-1}{6t}\right\}, \quad t \in (0, 1), \tag{2.89}$$

where the constraint on t ensures the convexity of the stencil superposition under smooth flow conditions. The effect of this stencil parameter on the spectral radius of D_1^+ is shown in figure 2.14. Values of t below 0.25 are excluded, as the spectral radius increases exponentially. At $t \approx 0.5$, the spectral radius stabilises, as boundary treatment eigenvalues no longer dominate. Consequently,

$t = 1/2$ is chosen, leading to the final stencil:

$$C^{+} = \left\{ \frac{1}{6}, \frac{7}{6}, -\frac{1}{3} \right\}, \tag{2.90}$$

as given in (2.58). Similarly, the stencil from (2.59) can be parameterized as:

$$C^{-}(t) = \left\{ \frac{3}{2} + \frac{1}{3t}, -\frac{1}{2} - \frac{2}{3t}, \frac{1}{3t} \right\}, \tag{2.91}$$

but the spectral properties of the operator D_1^- are less sensitive to t. Thus, $t = 1/3$ is chosen for simplicity.

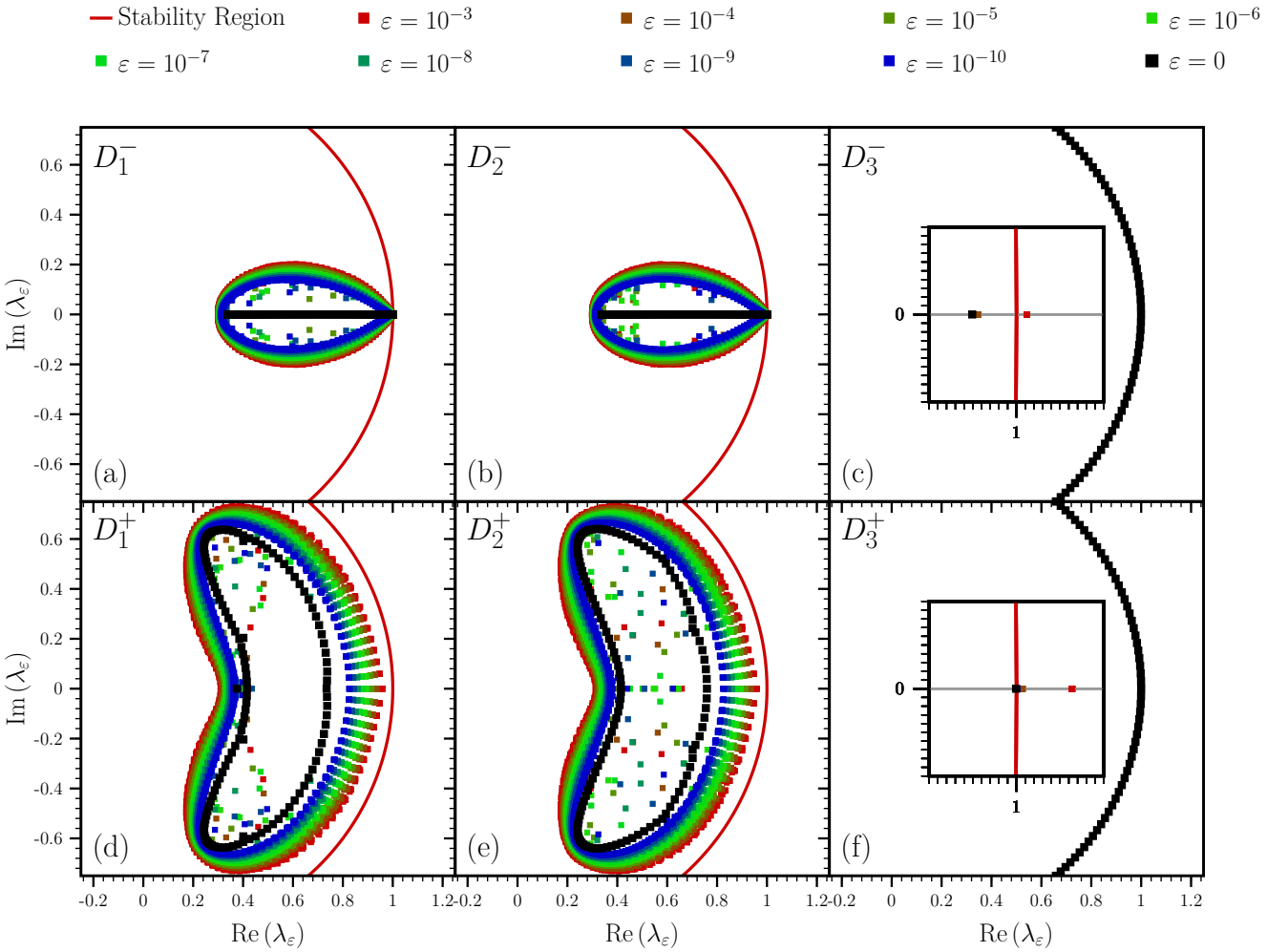


Figure 2.15: Eigenvalue spectrum and ε-pseudospectra of various linear operators, given in (2.88).

For the present analysis, the pseudospectra of $D_1^{\pm}$, $D_2^{\pm}$, and $D_3^{\pm}$ were computed using $N_x = 130$ spatial grid points with homogeneous boundary conditions. The spectrum of the perturbed opera-

tor in (2.87) was calculated for 5 random realisations of M for each $||M|| = \varepsilon \in \{10^{-3}, 10^{-4}, \ldots, 10^{-10}\}$. The pseudospectrum of each operator is approximated by the union of this finite collection of perturbed operators. Figure 2.15 shows the computed pseudospectra of all operators. The three operators associated with $a > 0$ have pseudospectra entirely within the stability region, indicating that no further analysis is needed to establish linear stability. However, the three operators associated with $a < 0$ are more marginally stable. For some chosen values of ε, the corresponding pseudospectra breach the stability region. The distance property from (2.85) for these operators is shown in figure 2.16. It is evident that while the pseudospectra extend beyond the stability region, they remain inside for approximately $\varepsilon < 10^{-5}$. The fact that there is a perturbation parameter that exists such that the full pseudospectrum is inside the stability region implies that the operator is stable, since this is a stronger condition than (2.85).

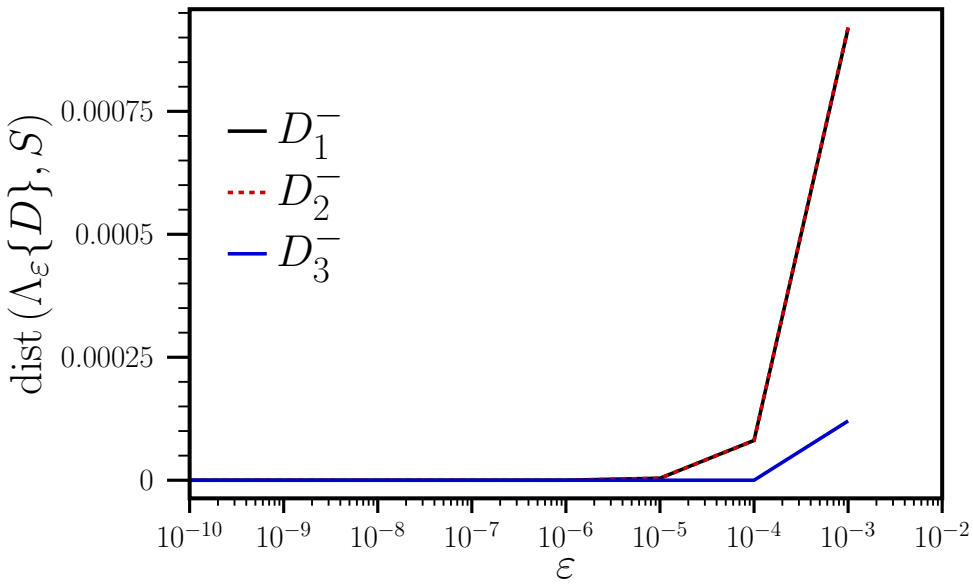


Figure 2.16: Distance of ε-pseudospectrum (from (2.85)) of various marginally-stable operators for small values of ε.

Note that the spectra in figure 2.15(c) and 2.15(f) correspond to the hybrid centred-upwind operator $D_3^{\pm}$. This operator is nearly a fully centred one, with its eigenvalues positioned almost entirely on the boundary of the stability region, although a small degree of dissipation is introduced via the time-integration operator.

2.11 Remarks

The development of an immersed boundary method is highly complex. There is a vast number of possible methods, the majority of which are not fit for practical application. To reiterate, each of these elements must be considered carefully in order to obtain high-quality results for turbulent flows at high Mach number:

- the method by which special cases in geometry handling are identified,
- the imposition of the numerical boundary condition as it relates to the local discretization,
- the relationship between modelling errors arising from the use of the wall model, discretization errors arising from the coarse near-wall grid, and the discretization errors introduced by the numerical treatment,
- the method by which the solution is interpolated onto sampling points, and
- the numerical discretization near the wall.

All of these aspects are required for a practical immersed boundary method.

Chapter 3: Wall Model for High Speed Flows

3.1 Background

WMLES has enjoyed success as a strategy for scale-resolving simulations since the 1970s, with initial work being performed by Deardorff [91]. This initial work investigated turbulent channel flow with a near-wall treatment that resembles the modern practice of WMLES, simply enforcing a frozen wall boundary condition that is computed explicitly using the law of the wall. An extension of this work by Schumann [92] modified Deardorff's approach by applying different filter scales in different areas of the computational domain. While classes of wall models have expanded greatly since this initial work, most implementations of WMLES have settled on some variation of the so-called "equilibrium" wall model, which works well for a large variety of low-speed flows, even outside the set of problems for which the underlying assumptions are valid [93].

For body-fitted grid approaches, some recent research on WMLES for high-speed flows has shown promise. Mettu and Subbareddy [94, 95] demonstrate that at relatively coarse grid spacing, WMLES heat transfer predictions for high-speed turbulent boundary layers can compare well with DNS results, a conclusion that may be said of a variety of studies [96, 97, 98, 99, 100]. One of the principal challenges of wall modelling for high speed boundary layers with heat transfer is accounting for compressibility effects.

At sufficiently high Mach number, boundary layer compressibility effects are realised both through the variation of fluid properties (like density and viscosity), and deviation of turbulent quantities (like turbulent heat transfer and shear stress) from the law of the wall. Attempts to account for these effects date back to the 1940s [101, 102], and are usually rooted in some attempt to 'localise' the fluid properties to recover the law of the wall. A classical example of such a transformation is that of van Driest [103], where the transformed velocity is computed as

$$u^{\mathrm{vd}} = \int_0^{u^+} \sqrt{\overline{\rho}/\rho_w}\, \mathrm{d}u^+, \tag{3.1}$$

where

$$u^+ = \frac{\widetilde{u}}{\sqrt{\tau/\rho_w}} \tag{3.2}$$

is the standard inner scaling.

Trettel and Larsson [26] note that the ability of (3.1) to transform the velocity profile to the law of the wall decreases as heat transfer increases, and introduce a corrective factor to the integral form (3.1) that yields significantly better results than the van Driest transformation. Since this work, a number of other efforts have been made to promote the law of the wall to compressible high-speed boundary layers. Griffin et al. [28] argue that a boundary layer transformation should treat viscous and turbulent shear stress, showing success in collapsing a range of profiles to the law of the wall. Volpiani et al. [104] use a data-driven approach to derive a transformation based on a tuned power law. Patel et al. [105] investigate the semi-local scaling in the damping function to create the semi-local wall model, which was extended later in Chen et al. [106], where further modifications to the scaling are made, as well as the turbulent Prandtl number formulation. Iyer and Malik [107] formulate an empirically-modified model by mixing various scaling laws.

As utilised in the vast majority of applications, these wall models are fundamentally an extension of incompressible turbulence modelling – in all cases, the action of the Reynolds shear stress within the viscous sublayer and buffer layer are modelled according to the incompressible law of the wall. The eventual success (or otherwise) of these models in predicting shear stress and heat transfer in the hypersonic flow regime requires such an approximately universal transformation exists.

A natural design process for compressible wall models is to attempt to reproduce the mean velocity and temperature profiles, coupled to the mean state computed from DNS [106, 107, 7, 108], often involving comparisons of momentum budgets in the wall model solution and the averaged outer solution. Modifications can be made to account for commutation errors, which are non-negligible for high Reynolds number flows [109], but this must be done carefully: accounting for resolved stresses effectively lowers the eddy viscosity near the wall and without proper solution coupling, this can be detrimental [110].

The observation that commutation errors in wall models are significant has yet to find its

way into wall modelling for high speed flows with significant heat transfer. The technique of comparing the wall model solution with mean profiles of compressible channel flow neglects this observation – it can be readily shown that even a wall model capable of predicting the perfect mean profile of DNS data will produce an error in the heat transfer prediction that scales with the square of the RMS fluctuation of temperature and, for sufficiently high Mach numbers [111], velocity. A predictive method for calculating heat transfer in more complex flow fields must take this into account. The aim of this chapter is decidedly not to make another attempt to formulate a closure for eddy viscosity or eddy conductivity for hypersonic flows, but rather to identify the importance of the unsteadiness within the wall model solution itself, demonstrate its significance, and formulate a technique to account for this unsteadiness in *a priori* analysis.

3.2 Overview of Wall Modelling for Compressible Boundary Layers

3.2.1 Reduction of the Governing Equations

The focus of wall modelling is to provide the correct shear stress and heat transfer at boundaries where a no-slip, diabatic wall is prescribed – this chapter does not consider adiabatic walls. Furthermore, as with many fundamental studies on wall modelling, this chapter will initially take a turbulent channel flow as a model problem. Shear stress and heat transfer are to be imposed as boundary conditions for the streamwise momentum and energy equations, respectively. For now, the streamwise momentum equation is considered:

$$\frac{\partial\left(\rho u\right)}{\partial t}+\frac{\partial\left(\rho u_i u\right)}{\partial x_i}+\frac{\partial p}{\partial x}=\frac{\partial\tau_{x,x_i}}{\partial x_i}+\psi_m. \tag{3.3}$$

The final term ψ_m in (3.3) is a forcing term introduced to prevent the solution from converging in time to zero velocity. Since the wall model is concerned with the steady-state behaviour of the solution to (3.3) an averaging procedure is considered. Averaging (3.3) in time and in the streamwise (x) and spanwise (z) directions, giving an equation for the mean profiles in the wall-normal (y) direction

$$\frac{\partial\left(\overline{\rho u v}\right)}{\partial y}=\frac{\partial\overline{\tau_{xy}}}{\partial y}+\overline{\psi_m}(y). \tag{3.4}$$

The quantities in (3.4) can be decomposed using the typical decompositions

$$\phi = \overline{\phi} + \phi' \tag{3.5}$$

and

$$\phi = \widetilde{\phi} + \phi'', \tag{3.6}$$

i.e. the Reynolds and Favre decompositions, respectively, where

$$\overline{\phi} = \frac{1}{L_x}\frac{1}{L_z}\lim_{T\to\infty}\int_{T_0}^{T}\int_0^{L_x}\int_0^{L_z}\phi(t,x,y,z)\,\mathrm{d}z\,\mathrm{d}x\,\mathrm{d}t \quad \text{and} \quad \widetilde{\phi} = \frac{\overline{\rho\phi}}{\overline{\rho}}. \tag{3.7}$$

With the viscous tensor definition

$$\overline{\tau_{xy}} = \overline{\mu\frac{\partial u}{\partial y}} = \overline{\mu}\frac{\partial \overline{u}}{\partial y} + \overline{\mu'\frac{\partial u'}{\partial y}} \approx \overline{\mu}\frac{\partial \overline{u}}{\partial y}, \tag{3.8}$$

the mean profile equation (3.4) becomes

$$\frac{\partial}{\partial y}\left(\overline{\mu}\frac{\partial \overline{u}}{\partial y} - \overline{\rho}\widetilde{u''v''}\right) + \overline{\psi_m}(y) = 0. \tag{3.9}$$

3.2.2 The Wall Model

Wall models typically solve an equation like (3.9) near solid walls with the aim of providing the correct mean wall shear stress. One of the commitments of wall modelling is that provision of the correct mean shear stress will dramatically improve the accuracy of calculations carried out on a significantly under-resolved grid. Because the shear stress term $\overline{\rho}\widetilde{u''v''}$ in (3.9) is difficult and expensive to resolve near the wall (and because numerical estimates of the mean flow gradient are inaccurate), wall model implementations generally choose to represent the unsteady shear stress term by way of an equivalent viscosity, satisfying the approximation

$$\mu_t\frac{\partial \widetilde{u}}{\partial y} \approx -\overline{\rho}\widetilde{u''v''}. \tag{3.10}$$

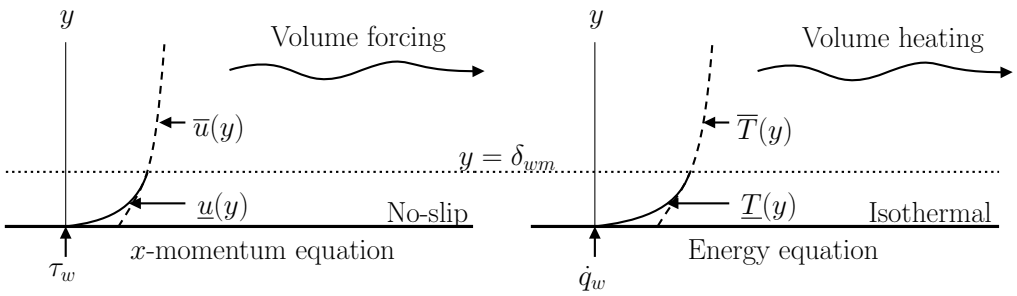


Figure 3.1: Schematic of wall model coupling to instantaneous interior solution.

This results in the most typical wall model implementation, given as

$$\frac{\partial}{\partial y}\left(\left(\underline{\mu}+\underline{\mu}_t\right)\frac{\partial \underline{u}}{\partial y}\right) = -\underline{\psi}_m, \tag{3.11}$$

which, for $\underline{\psi}_m = 0$, gives the so-called "equilibrium" wall model. Note that, for the rest of this chapter, any flow quantity ϕ that is present in the governing equations and that has an analogue in the wall model will be represented within the context of the wall model using the notation $\underline{\phi}$.

The wall model is coupled to the interior solution at a fixed distance δ_{wm}, where the instantaneous wall-parallel velocity, temperature, and density are taken as boundary conditions for (3.11), along with the physical no-slip and isothermal boundary conditions. The wall shear stress and wall heat flux are then passed back to the interior as

$$\tau_w = \underline{\mu}\frac{\partial \underline{u}}{\partial y} \quad \text{and} \quad \dot{q}_w = -\frac{\gamma R \underline{\mu}}{\Pr(\gamma-1)}\frac{\partial \underline{T}}{\partial y}. \tag{3.12}$$

This is crudely demonstrated in figure 3.1. Note that the choice of δ_{wm} is of critical importance in practice, as noted in [86] and many other studies.

3.2.3 The Energy Equation

The sections above have been concerned only with the momentum equation because there are fewer terms to contend with for turbulent channel flow. The energy conservation equation is given as

$$\frac{\partial E}{\partial t} + \frac{\partial}{\partial x_i}\left(u_i\left(E+p\right)\right) = \frac{\partial}{\partial x_i}\left(u_k \tau_{x_i,x_k} - \dot{q}_{x_i}\right) + \psi_e \tag{3.13}$$

The procedure that yields (3.9) from (3.3) can also be applied to (3.13), resulting in the averaged energy equation

$$\frac{\partial}{\partial y}\left(\overline{\rho}\widetilde{v''k''} + \frac{R}{\gamma-1}\overline{\rho}\widetilde{v''T''}\right) = \frac{\partial}{\partial y}\left(\overline{u_k\tau_{yx_k}} + \frac{\gamma R}{\gamma-1}\frac{\overline{\mu}}{\mathrm{Pr}}\frac{\partial\overline{T}}{\partial y}\right) + \overline{\psi_e}, \tag{3.14}$$

where the viscosity-fluctuation term in (3.14) has been ignored, assuming

$$\overline{\mu'\frac{\partial T'}{\partial y}} \approx 0. \tag{3.15}$$

This assumption is possibly not well justified.

Once again, the procedure of capturing fluctuations in the form of turbulent flow properties can be applied to (3.14), yielding the equilibrium wall model energy equation

$$\frac{\partial}{\partial y}\left(\frac{\gamma R}{\gamma-1}\left(\frac{\underline{\mu}}{\underline{\mathrm{Pr}}} + \frac{\underline{\mu}_t}{\underline{\mathrm{Pr}}_t}\right)\frac{\partial\underline{T}}{\partial y}\right) + \frac{\partial}{\partial y}\left(\left(\underline{\mu}+\underline{\mu}_t\right)\underline{u}\frac{\partial\underline{u}}{\partial y}\right) = -\underline{\psi}_e \tag{3.16}$$

3.2.4 Forcing Term

The analysis in this chapter deals with a particular use-case of wall models, detailed in this section. In practice, wall models are used in conjunction with LES, where the governing equations are formally cast in a spatially filtered sense. Using a typical subgrid-scale model eddy viscosity μ_{sgs}, the streamwise momentum equation becomes

$$\frac{\partial\left(\rho u\right)}{\partial t} + \frac{\partial\left(\rho u_i u\right)}{\partial x_i} + \frac{\partial p}{\partial x} = \frac{\partial}{\partial x_i}\left(\left(\mu+\mu_{\mathrm{sgs}}\right)\left(\frac{\partial u_i}{\partial x} + \frac{\partial u}{\partial x_i}\right)\right) + \frac{\partial}{\partial x}\left(\left(\beta+\beta_{\mathrm{sgs}}\right)\frac{\partial u_k}{\partial x_k}\right) + \psi_m, \tag{3.17}$$

where the flow quantities now formally represent values that have been filtered consistently with the definition of μ_{sgs}.

The forcing term ψ_m in (3.17) can be chosen in a variety of ways. Two typical forcing choices are

- *dynamic forcing*, where ψ_m is computed at every stage of the calculation using a closed-loop calculation targeting a specific mass flow rate, or

- *static forcing*, where ψ_m is set to a value that depends only on a target wall shear stress and flow variables.

This work focuses only on *static forcing*, where the wall shear stress is prescribed from a reference DNS. Given a particular value for the mean wall shear stress $\overline{\tau}_w$, it can be readily shown that the forcing term

$$\psi_m = \frac{\rho}{\rho_b}\frac{\overline{\tau}_w}{\delta} \tag{3.18}$$

with volume-averaged density ρ_b will yield a numerical solution with a mean wall shear stress *exactly* as prescribed. This exactness condition is independent of the numerical scheme – as long as it is discretely conservative – used to discretize (3.17), the subgrid-scale parameters, grid resolution, and most importantly, the details of the wall model implementation. It is this last point that makes the analysis of this forcing mode appealing for the work in this chapter.

The energy forcing term in (3.13) is such that the mean solution has exactly the specified mean wall heat transfer $\overline{\dot{q}}_w = \overline{\tau}_w U_b$, and is thus given by

$$\psi_e = \frac{\psi_m}{\overline{\tau}_w \overline{U}_b}, \tag{3.19}$$

where U_b is the bulk velocity given as

$$U_b = \int_V \rho u \, \mathrm{d}V \Big/ \int_V \rho \, \mathrm{d}V. \tag{3.20}$$

3.2.5 Compressible Boundary Layer Transformations

The definition of the eddy viscosity in (3.10) will clearly have a significant effect on the quality of any solution that employs the corresponding model. The aim of such a definition is to characterise the mean behaviour of a turbulent boundary layer profile under the present conditions. For compressible flows, such a characterisation is an active area of research, discussion of which can be found in [104, 28], and many others.

Considering first an incompressible flow at sufficiently high Reynolds number, most eddy vis-

cosity formulations are based on the mixing-length model and are of the form

$$\underline{\mu}_t = \underline{\mu}\underline{y}^+\kappa D(\underline{y}^+), \tag{3.21}$$

where D is a damping function designed to recover the incompressible law of the wall, e.g. the van Driest damping function given by

$$D(\underline{y}^+) = \left(1 - e^{-\underline{y}^+/A^+}\right)^2. \tag{3.22}$$

The form of the damping function in (3.22) is such that $D \approx 0$ near the wall, and that $D \approx 1$ away from the wall, particularly in the log-layer where the edge of the wall model domain ought to be. The transition between these two states is constructed to be at approximately $y^+ \approx A^+$, where A^+ is chosen heuristically to be the location between the buffer layer and the log layer (usually a value of 17). Note that near the wall, $D \approx 0$ together with (3.11) implies that

$$\frac{\partial u}{\partial y} \sim O(1), \tag{3.23}$$

and that, away from the wall,

$$\frac{\partial u}{\partial y} \sim O\left(\frac{1}{y^+}\right), \tag{3.24}$$

both of which are respectively consistent with the law of the wall assertions

$$u \sim y^+ \quad \text{and} \quad u \sim \log\left(y^+\right). \tag{3.25}$$

Typically, other formulations of the damping function are conceived by applying some kind of coordinate transformation to the wall-normal coordinate. As such, eddy viscosity wall models with eddy viscosity of the form

$$\underline{\mu}_t = \kappa\underline{\mu}T_m(y)\left(1 - e^{-T_d(y)/A^+}\right)^2, \tag{3.26}$$

with transformation functions T_m and T_d, are considered in the rest of this chapter.

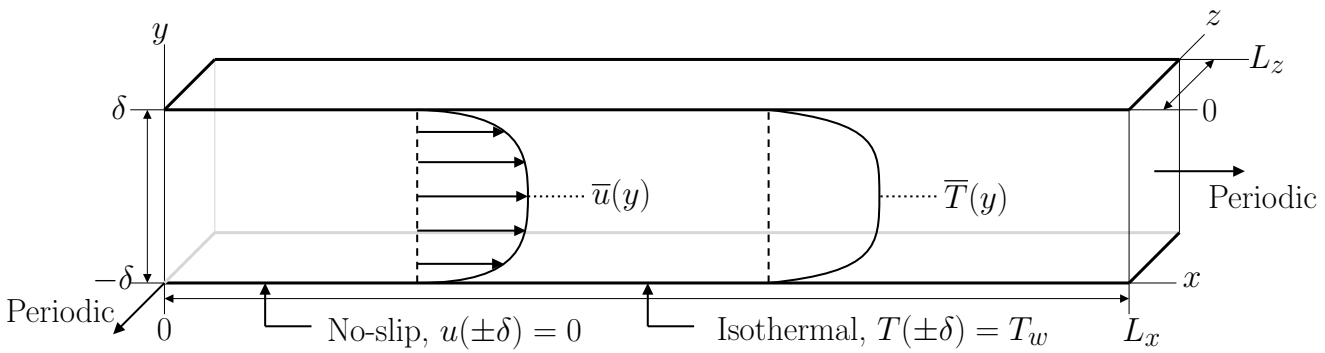


Figure 3.2: Schematic of turbulent channel test case.

3.2.6 Importance of the Turbulent Closure

The importance of the eddy viscosity model can be demonstrated by performing WMLES of a high speed channel flow, as per section 3.2.1. The physical domain lies within the spatial extent $(x, y, z) \in [0, L_x] \times [-\delta, \delta] \times [0, L_z]$ with periodic boundary conditions in the streamwise and spanwise directions. Isothermal no-slip walls are imposed at the solid walls located at $y = \pm\delta$. In this case, $\delta = 1$, $L_x = 4\pi$, $L_z = 2\pi$, and uniform grid spacing is used with 64 cells spanning the height of the channel. A CFL of 0.65 is used.

The turbulent channel domain is visualised in figure 3.2. Values of the mean wall shear stress and mean wall heat transfer rate are chosen to set the bulk Reynolds number

$$\mathrm{Re}_b = \frac{\rho_b U_b \delta}{\mu_w} \tag{3.27}$$

and bulk Mach number

$$M_b = \frac{U_b}{\sqrt{\gamma R T_w}} \tag{3.28}$$

at desired values. The preliminary demonstration case considered in this section has the nominal values

$$\mathrm{Re}_b = 20\,000 \quad \text{and} \quad M_b = 6.0. \tag{3.29}$$

For the sake of demonstration, two coordinate scaling functions $T_d(y)$ for (3.26) are considered, referred to as T_0 and T_1. In this case, T_0 is from [26] and T_1 uses a variation of the mixed scaling from [107]. It should be noted that these models were expressly chosen to illustrate the effect that

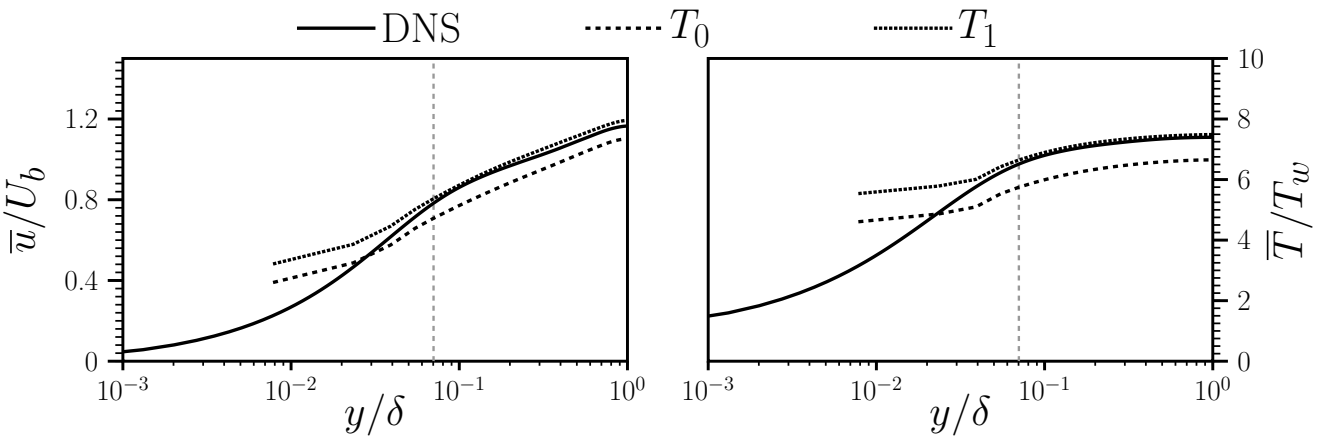


Figure 3.3: Comparison of wall model simulation results at $M_b = 6.0$ and $\mathrm{Re}_b = 20\,000$ using two coordinate scaling transformations. The coupling location is indicated with the vertical dashed line.

will be discussed hereafter, not to compare their ability to capture the flow physics.

It is apparent from figure 3.3 that the different coordinate transformations T_0 and T_1 used in the exponent of the damping function in (3.21) have introduced an offset in the velocity and temperature profiles in the resulting simulation. This becomes apparent when (3.9) is coupled to (3.11), considering the half-channel from $y = 0$ to $y = \delta$, which gives the system

$$\begin{aligned} \frac{\partial}{\partial y}\left(\overline{\mu}\frac{\partial \overline{u}}{\partial y} - \overline{\rho}\widetilde{u''v''}\right) + \overline{\psi_m}(y) = 0, \quad \overline{u}(\delta_{wm}) = \overline{u}_h, \qquad & \frac{\partial \overline{u}}{\partial y}(\delta) = 0 \\ \frac{\partial}{\partial y}\left(\overline{\left(\underline{\mu} + \underline{\mu}_t\right)\frac{\partial \underline{u}}{\partial y}}\right) = -\overline{\underline{\psi}_m} \quad \underline{\overline{u}}(0) = 0 \qquad & \underline{\overline{u}}(\delta_{wm}) = \overline{u}_h, \end{aligned} \tag{3.30}$$

where the interface velocity u_h is given by

$$u_h = \int_0^\delta \frac{\underline{\tau}_w}{\underline{\mu} + \underline{\mu}_t}\,\mathrm{d}y, \tag{3.31}$$

assuming an equilibrium wall model. By the conservation properties of (3.3) and the nature of the forcing term in (3.18), it becomes clear that the interface velocity is a function only of the wall model viscosity and the eddy viscosity. The interface velocity introduces an offset to the LES equation via the coupling boundary condition in (3.30).

An unusual feature of (3.30) is that the LES solution is not present for $y < \delta_{wm}$, indicating that, in the mean, that region of the velocity profile simply facilitates the mean interface velocity

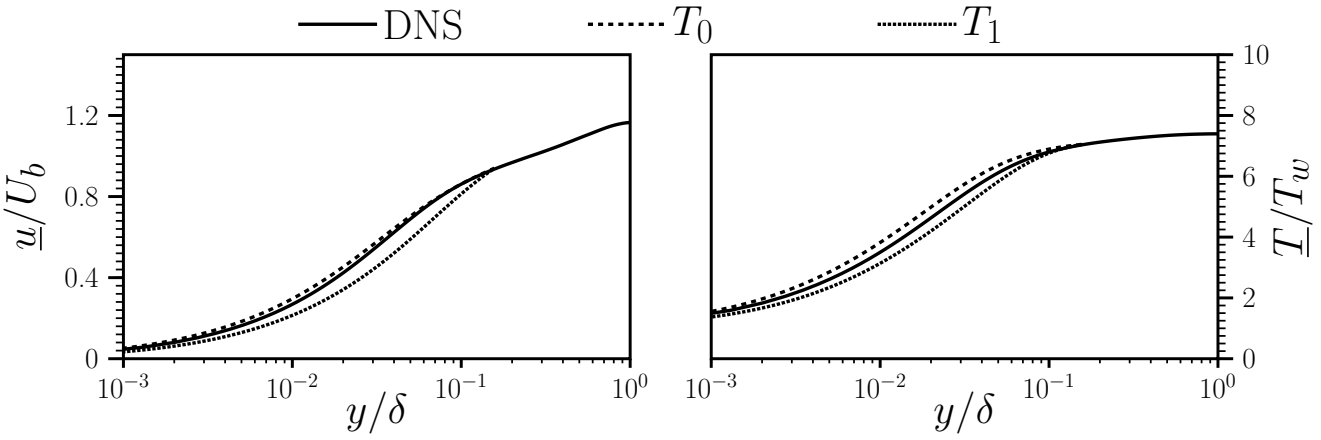


Figure 3.4: Comparison of wall model solutions using different coordinate transformations.

condition. Furthermore, for an "accurate" wall model, the velocity profile in the volume should match that of a DNS if the subgrid-scale model is sufficiently accurate, the volume grid is sufficiently resolved, and the Reynolds stress is sufficiently resolved at the coupling location. Note that the temperature profile is also shifted via the same mechanism, but the analysis is more complex without contributing to the discussion.

3.2.7 A-Priori Wall Model Evaluation

A natural first step in comparing the accuracy for the wall model formulations used in figure 3.3 is to perform *a priori* analysis without coupling the wall model to the LES equations (3.17). In this analysis, the wall model is solved in a standalone mode, where the mean velocity and temperature profiles are taken from a reference DNS (note that all channel DNS data is taken from [112]). The profiles are sampled within the log layer, and the sampled values are used as boundary conditions for (3.11) and (3.16). Figure 3.4 shows the result of this procedure for the conditions described in (3.29). At first glance, T_0 appears to perform somewhat better than T_1 in this regard, seemingly contradictory to the results presented in figure (3.3). At the very least, the velocity profile for T_0 nearly matches the DNS profile.

Figure 3.3 shows that the relative error introduced in the mean temperature is larger than that introduced in the velocity profile for the T_0 transformation. To investigate this difference, corresponding terms in the interior energy equation (3.14) and wall model energy equation (3.16)

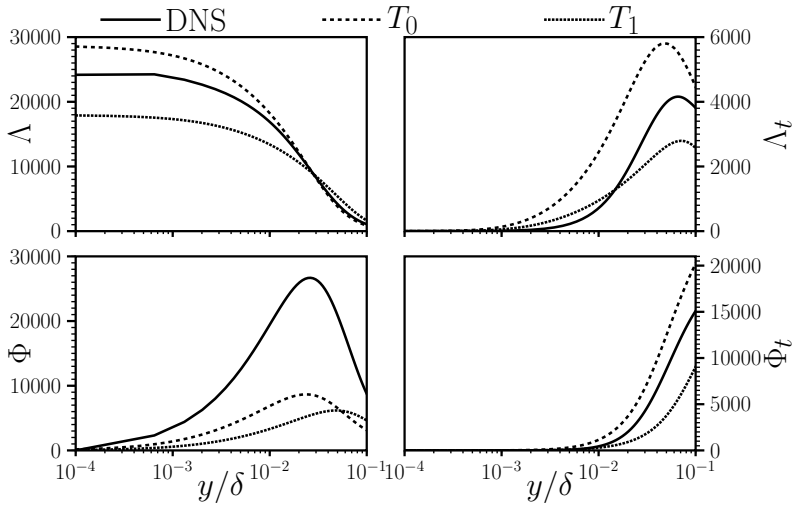


Figure 3.5: Comparison of budget terms for wall model coordinate transformations.

can be compared following [106], namely

$$\frac{\partial}{\partial y}\left(\Lambda + \Lambda_t + \Phi + \Phi_t\right) = \overline{\psi_e}, \quad \text{and} \quad \frac{\partial}{\partial y}\left(\underline{\Lambda} + \underline{\Lambda}_t + \underline{\Phi} + \underline{\Phi}_t\right) = \underline{\overline{\psi}_e}, \tag{3.32}$$

where

$$\Lambda = \frac{1}{\mathrm{Pr}}\frac{\gamma R}{(\gamma-1)}\overline{\mu\frac{\partial T}{\partial y}}, \quad \underline{\Lambda} = \frac{1}{\mathrm{Pr}}\frac{\gamma R}{(\gamma-1)}\underline{\mu}\frac{\partial \underline{T}}{\partial y} \tag{3.33}$$

$$\Lambda_t = -\frac{\gamma R}{(\gamma-1)}\widetilde{\overline{\rho}v''T''}, \quad \underline{\Lambda}_t = \frac{1}{\underline{\mathrm{Pr}}_t}\frac{\gamma R}{(\gamma-1)}\underline{\mu}_t\frac{\partial \underline{T}}{\partial y} \tag{3.34}$$

$$\Phi = \overline{u_k\tau_{yx_k}}, \quad \underline{\Phi} = \underline{\mu}\,\underline{u}\frac{\partial \underline{u}}{\partial y} \tag{3.35}$$

$$\Phi_t = -\widetilde{\overline{\rho}v''k''}, \quad \underline{\Phi}_t = \underline{\mu}_t\,\underline{u}\frac{\partial \underline{u}}{\partial y} \tag{3.36}$$

The budget terms (3.33) through (3.36) can be found in figure 3.5 for the DNS, T_0 and T_1. It is difficult to conclude which of T_0 or T_1 outperforms the other given the comparison budget terms. Unsurprisingly, the laminar diffusion Λ is better predicted by T_0, consistent with the overall favourable match with the mean temperature profile in figure 3.4. The laminar dissipation Φ is underpredicted by both models throughout the wall modelling layer, *i.e.*, $0 \leq y \leq \delta_{wm}$. Turbulent heat transfer Λ_t and turbulent kinetic energy dissipation Φ_t are both overpredicted by T_0 and underpredicted by T_1, with what appears to be a better overall match to the DNS data by T_0

(note the relative magnitudes of Λ_t and Φ_t).

Given the profiles in figure 3.4 and the budgets in figure 3.5, it is justifiable to conclude that T_0 should be expected to outperform T_1 for the conditions (3.29) of the reference DNS calculation. However, the profiles from the WMLES calculation in figure 3.3 do not appear to be consistent with this conclusion.

3.3 Unsteadiness Effects

3.3.1 Wall Model Idealisations

The purpose of sections 3.2.6 through 3.2.7 is to serve as a process for designing compressible wall models. A model may be conceived, such as one of the form given in (3.11), (3.16) and (3.10), analysed against reference DNS profiles and budget terms in a standalone implementation as in figures 3.4 and 3.5, and then tested and evaluated in a WMLES calculation as in figure 3.3. However, as demonstrated above, a favourable outcome for one wall model in the *a priori* analysis may not be reflected when carrying out a coupled WMLES calculation.

Careful analysis of the relevant equations makes this relatively clear. The wall model equations in (3.30) are averaged in time, meaning that the mean wall model solution is not described explicitly. Performing the Reynolds decomposition on the wall model equation in (3.30) gives

$$\frac{\partial}{\partial y}\left(\left(\underline{\overline{\mu}}+\underline{\overline{\mu}}_t\right)\frac{\partial \underline{\overline{u}}}{\partial y}\right) = -\underline{\overline{\psi}}_m - \frac{\partial}{\partial y}\left(\overline{\underline{\mu}'\frac{\partial \underline{u}'}{\partial y}}+\overline{\underline{\mu}'_t\frac{\partial \underline{u}'}{\partial y}}\right), \tag{3.37}$$

which can now be solved for $\underline{\overline{u}}$ provided the unsteady terms on the right-hand-side. Given that the eddy viscosity in (3.11) has been designed to account for unsteady shear stress already, it may be construed that the fluctuating eddy viscosity term in (3.37) effectively double-counts a certain portion of the turbulent shear stress, possibly leading to offsets in the resulting profile. Once again, it should be noted that the energy equation can be analysed in a similar way, but is not written explicitly here for brevity. A tempting method of analysis may be to compute the explicit unsteady terms in (3.37) and perform a budget comparison as per figure 3.5, but any argument as to whether or not one model performs better than another would still be polluted with the errors

in the other budget terms in (3.32).

The wall model, when considered from the point of view of the coupling procedure in (3.30), need not necessarily be constructed to be consistent with the averaged momentum equation (3.4). Generally, the wall model may be considered simply as some function of the interface condition that yields back a shear stress and heat transfer for the interior solution, opaquely written as

$$f_w = W(q_h), \tag{3.38}$$

where $f_w = (\tau_w, q_w)$ denotes the viscous wall flux and $q_h = (u_h, T_h, \rho_h)$ denotes the state at the coupling interface. The wall model component of (3.30) and the analogous energy equation is then simply

$$\overline{f}_w = \overline{W(q_h)}. \tag{3.39}$$

It it important to note that this is different than the boundary condition that is expected of a predictive model that would yield a perfect match in sections 3.2.6 through 3.2.7. Rather, the condition being evaluated in the standalone procedure from figure 3.4 is

$$\overline{f}_w = W(\overline{q_h}). \tag{3.40}$$

For the discussion that follows, a wall model is *perfect in the mean-flux sense* when, given *any* ground-truth solution q_{DNS}, (3.39) is satisfied as

$$\overline{f}_{w,\text{DNS}} = \overline{W\left(q_h\right)} \tag{3.41}$$

when

$$\overline{q_h} = \overline{q}_{h,\text{DNS}}. \tag{3.42}$$

Similarly, a model is *perfect in the mean-state sense* when (3.40) is satisfied as

$$\overline{f}_{w,\text{DNS}} = W(\overline{q}_{h,\text{DNS}}). \tag{3.43}$$

As per (3.37), the difference between the results in these models for a given condition can be isolated to the influence of unsteadiness.

For a particular condition, it is trivial to design wall models satisfying (3.39) and (3.40). Considering the conditions in (3.29) and the associated profile from figure 3.3, a simple linear scaling of the reference solution gives

$$\underline{\tau}_w = \frac{\underline{u}_h}{\overline{u}_{h,\mathrm{DNS}}}\overline{\tau}_{w,\mathrm{DNS}} \quad \text{and} \quad \underline{\dot{q}}_w = \frac{\underline{T}_h}{\overline{T}_{h,\mathrm{DNS}}}\overline{\dot{q}}_{w,\mathrm{DNS}}. \tag{3.44}$$

This is a simple model that is linear with respect to the coupling state, and taking the Reynolds decomposition readily yields

$$\overline{\underline{u}_h} = \overline{u}_{h,\mathrm{DNS}} \quad \text{and} \quad \overline{\underline{T}_h} = \overline{T}_{h,\mathrm{DNS}} \tag{3.45}$$

since the mean viscous wall fluxes match by construction. Note that a model similar to this was proposed by Schumann [92]. This constitutes satisfaction of (3.39). Producing a model that satisfies 3.40 requires setting

$$\overline{\underline{u}} = \overline{u}_{\mathrm{DNS}} \quad \text{and} \quad \overline{\underline{T}} = \overline{T}_{\mathrm{DNS}} \tag{3.46}$$

in (3.11) and (3.16) and then solving for the appropriate eddy viscosity and eddy conductivity. Doing so results in

$$\underline{\mu}_t = \frac{\underline{\psi}_m y + \underline{\tau}_w}{\partial \overline{u}_{\mathrm{DNS}}/\partial y} - \underline{\mu} \tag{3.47}$$

and

$$\frac{\underline{\mu}_t}{\mathrm{Pr}_t} = \frac{(\gamma-1)}{\gamma R}\frac{\underline{\psi}_e y + \underline{\dot{q}_w} - \left(\underline{\mu} + \underline{\mu}_t\right)\underline{u}\frac{\partial \underline{u}}{\partial y}}{\partial \overline{T}_{\mathrm{DNS}}/\partial y} - \frac{\underline{\mu}}{\mathrm{Pr}}. \tag{3.48}$$

Note that, for this work, the value of the turbulent Prandtl number Pr_t is fixed at 0.9. Hereafter, the model described by (3.44) will be referred to as P-MFC, and that described by (3.11) and (3.16) with (3.47) and (3.48) will be referred to as P-MSC. Clearly, the P-MFC and P-MSC models offer no predictive capability owing to their reliance on the availability of ground-truth reference data. However, comparison of these two distils the influence of unsteady terms such as those in (3.37).

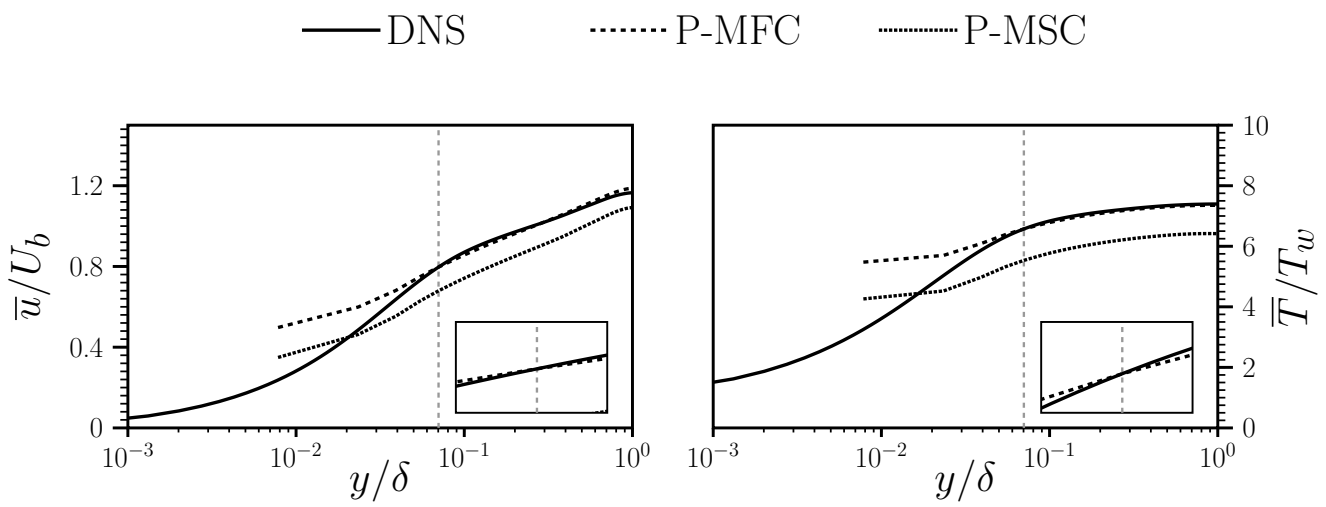


Figure 3.6: Comparison of wall model simulation results at $M_b = 6.0$ and $\mathrm{Re}_b = 20\ 000$ using P-MFC and P-MSC models. The coupling location is indicated with the vertical dashed line.

3.3.2 Investigation of Unsteadiness via Idealised models

For the conditions in (3.29), the results for coupled simulations using the P-MSC and P-MFC models are shown in figure 3.6.. The inset shows that, as expected from (3.45), the P-MFC model perfectly matches the reference profile at the interface location specified. Similarly to the results from the different coordinate transformations found in figure 3.3, there is an offset between the predicted velocity and temperature profiles. Notably, the slope of the mean profiles remains virtually constant. The fact that there is an offset between these two profiles at the sampling location directly shows the influence of the unsteady terms in the nonlinear model. The mismatch in the slope in the profiles can be mainly attributed to the influence of the subgrid scale model employed in (3.17).

For the sake of wall model design, the form of the P-MSC model in (3.47) and (3.48) is more appealing, aligning closely with physical investigations supporting compressible boundary layer and RANS turbulence modelling research. From this point of view, the presence of the unsteady terms in (3.37) are an artefact of the coupling procedure.

3.3.3 A Simple Correction for the Unsteady Terms

In an effort to bypass the artificial stress arising from the coupling procedure, a relation can be made between the P-MFC and P-MSC models. Taking (3.39) and (3.40) as a starting point, taking

an expansion about the mean state gives

$$\begin{aligned}\overline{W(q_h)} = \overline{W(\overline{q}_h + q'_h)} &= \overline{W(\overline{q}_h)} + \overline{q'_h \frac{\partial W(\overline{q}_h)}{\partial q_h}} + \frac{1}{2}\overline{q'_h q'_h \frac{\partial^2 W(\overline{q}_h)}{\partial q_h^2}} + O\left(\overline{q'_h q'_h q'_h}\right) \\ &= W(\overline{q}_h) + \frac{1}{2}\overline{q'_h q'_h}\frac{\partial^2 W(\overline{q}_h)}{\partial q_h^2} + O\left(\overline{q'_h q'_h q'_h}\right) \end{aligned} \tag{3.49}$$

More specifically, the expansion can be written as

$$\begin{aligned}\overline{\underline{\tau}_w(u_h, T_h, \rho_h)} - \underline{\tau}_w(\overline{u}_h, \overline{T}_h, \overline{\rho}_h) = &\frac{1}{2}\overline{u'_h u'_h}\frac{\partial^2 \underline{\tau}_w}{\partial u_h^2} + \overline{u'_h T'_h}\frac{\partial^2 \underline{\tau}_w}{\partial u_h \partial T_h} + \overline{u'_h \rho'_h}\frac{\partial^2 \underline{\tau}_w}{\partial u_h \partial \rho_h} \\ &\frac{1}{2}\overline{T'_h T'_h}\frac{\partial^2 \underline{\tau}_w}{\partial T_h^2} + \overline{T'_h \rho'_h}\frac{\partial^2 \underline{\tau}_w}{\partial T_h \partial \rho_h} \\ &+ \frac{1}{2}\overline{\rho'_h \rho'_h}\frac{\partial^2 \underline{\tau}_w}{\partial \rho_h^2} \\ &+ H, \end{aligned} \tag{3.50}$$

where H contains higher-order terms. A similar expansion can be written for the wall heat transfer, but has once again been omitted for brevity. The error in the mean wall shear stress scales directly with Reynolds-stress-like terms and with the Hessian of the wall model itself.

The individual terms in (3.50), normalised by the mean wall shear stress, can be written as

$$\begin{aligned} E_{uu} = \frac{1}{\overline{\tau}_w}\overline{u'_h u'_h}\frac{\partial^2 \underline{\tau}_w}{\partial u_h^2}, \quad E_{TT} = \frac{1}{\overline{\tau}_w}\overline{T'_h T'_h}\frac{\partial^2 \underline{\tau}_w}{\partial T_h^2}, \quad E_{TT} = \frac{1}{\overline{\tau}_w}\overline{\rho'_h \rho'_h}\frac{\partial^2 \underline{\tau}_w}{\partial \rho_h^2}, \\ E_{\rho u} = E_{u\rho} = \frac{1}{\overline{\tau}_w}\overline{u'_h \rho'_h}\frac{\partial^2 \underline{\tau}_w}{\partial u_h \partial \rho_h}, \quad E_{Tu} = E_{uT} = \frac{1}{\overline{\tau}_w}\overline{T'_h u'_h}\frac{\partial^2 \underline{\tau}_w}{\partial u_h \partial T_h} \\ E_{T\rho} = E_{\rho T} = \frac{1}{\overline{\tau}_w}\overline{T'_h \rho'_h}\frac{\partial^2 \underline{\tau}_w}{\partial \rho_h \partial T_h}, \end{aligned} \tag{3.51}$$

so the unsteady error is then given by

$$\varepsilon_\tau = \frac{1}{2}\left(E_{uu} + E_{TT} + E_{\rho\rho}\right) + \left(E_{\rho T} + E_{\rho u} + E_{uT}\right) + H, \tag{3.52}$$

where

$$\varepsilon_\tau = \frac{\overline{\tau_w(q_h)}}{\tau_w(\overline{q_h})} - 1 \tag{3.53}$$

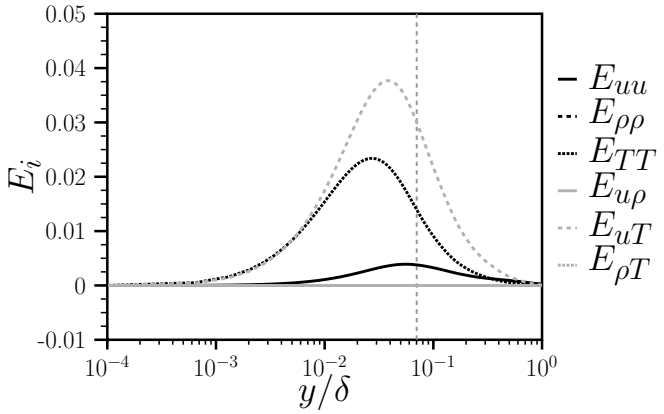


Figure 3.7: Individual unsteady shear stress error terms for the P-MSC model. The vertical dashed line indicates the wall model coupling location.

Figure 3.7 shows each of the individual error terms in (3.52). With each of the correlation terms computed from DNS data. It is immediately clear that $E_{\rho u}$, $E_{\rho T}$, and $E_{\rho\rho}$ are zero, but it should be noted that this is an artefact of the definition of the P-MSC model, in particular that the definition of the eddy viscosity in (3.47) no longer depends on the density at the interface location. By extension, this means that the mean wall shear stress does not depend on the fluid density directly. Observation of the nonzero terms indicate that E_{uu} is much less significant than the temperature fluctuation term E_{TT} or the velocity-temperature correlation term E_{uT}. Neither of these terms can be said to be significant in lower-speed flows without heat transfer. The expression (3.52) can also be re-written to express the error in heat transfer:

$$\varepsilon_{\dot{q}} = \frac{1}{2}\left(Q_{uu} + Q_{TT} + Q_{\rho\rho}\right) + \left(Q_{\rho T} + Q_{\rho u} + Q_{uT}\right) + H, \tag{3.54}$$

and these terms can be found in figure 3.8. The sensitivity in heat transfer for the P-MSC model depends primarily on the streamwise Reynolds stress $\overline{u'u'}$, with a small and notably negative contribution from the temperature fluctuations. The velocity-temperature correlation has a negligible contribution. It should be noted that since the P-MSC model is not truly a physical model, *i.e.* designed for the sole purpose of matching the mean state perfectly, relative contributions ought not to be too carefully interpreted with respect to physical aspects.

It is clear from figures 3.7 and 3.8 that the unsteady error will depend on the coupling location between the wall model and the interior solution. For the present flow configuration, it would be

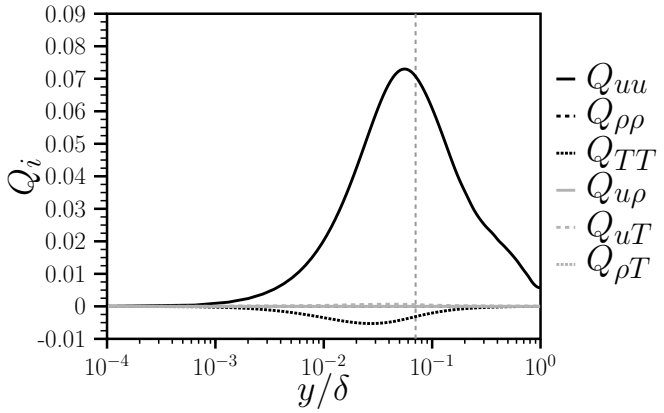


Figure 3.8: Individual unsteady heat transfer error terms for the P-MSC model. The vertical dashed line indicates the wall model coupling location.

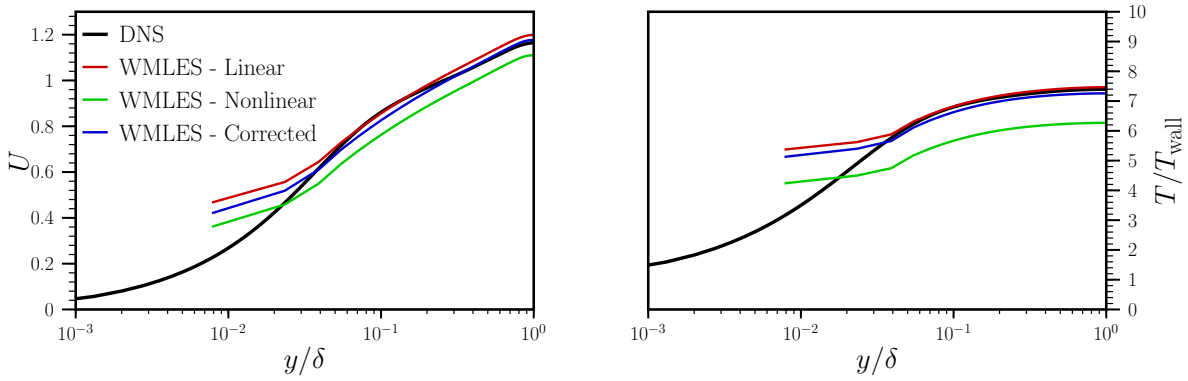


Figure 3.9: Corrected nonlinear model comparison.

undesirable to place the coupling location outside the region of peak error. Computing the error terms according to (3.52) and (3.54) for the given configuration gives relative errors

$$\varepsilon_\tau \approx 7\% \quad \text{and} \quad \varepsilon_{\dot{q}} \approx 4\%. \tag{3.55}$$

Clearly, this represents a non-negligible error for heat transfer and skin friction predictions. By correcting the shear stress and heat transfer at the wall by the amount predicted from the error analysis above, the P-MSC model provides better predictions for the velocity and temperature profiles, as seen in figure 3.9.

The corrected nonlinear model can be found for a series of varying conditions in figure 3.10. This shows that the error introduced from the nonlinear model is more significant at higher Mach and Reynolds numbers. For completeness, the original scaling T_0 is also included, although note

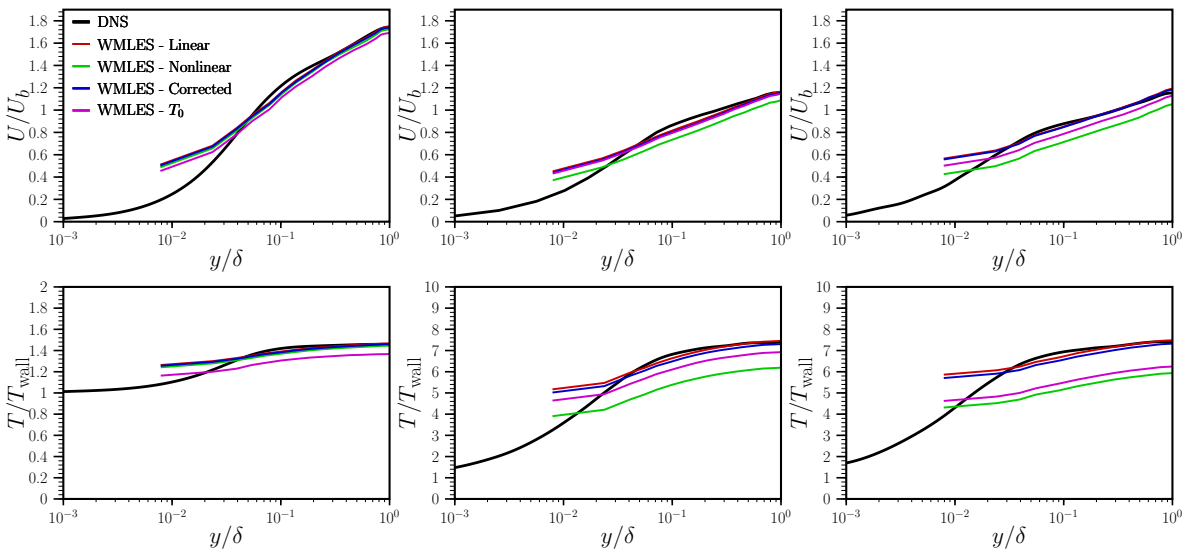


Figure 3.10: Corrected nonlinear model comparison for $M = 1$, $\mathrm{Re} = 5000$ (left), $M = 6$, $\mathrm{Re} = 20000$ (centre), and $M = 6$, $\mathrm{Re} = 40000$ (right).

that at the low Reynolds number case, the wall model is not necessarily expected to perform very well.

The results in figures 3.9 and 3.10 demonstrate that, for the nonlinear P-MSC model, a correction of the form (3.7) and (3.8) can improve results over the base model. This will be taken as a basis for the development of a more general model in the rest of this chapter.

3.4 Generalised Unsteady Wall Model Error Correction

The correction in section 3.3.3 works reasonably well for turbulent channel flow at statistical equilibrium. However, for more general flows, it is generally much more expensive and difficult to compute the wall model sensitivity terms than it is to compute the wall model itself. This primarily comes from the fact that, when computing derivatives using finite differences, the derivative estimates strongly rely on the convergence tolerance of the wall model ODE system. To obtain accurate sensitivity estimates, exceedingly small convergence residuals must be achieved. Furthermore, the wall model must be solved many times to provide the stencil points for the finite-difference approximation. This represents a significant computational cost when compared to the base wall model. A relatively simple solution to this problem may be to simply solve the wall model using the averaged state, but this section does not consider the averaging approach

and rather focuses on analysing a possible correction.. This section will derive a general error correction procedure for the error identified in this chapter.

3.4.1 Dual Numbers

This section briefly introduces the dual number system, which has been developed here as a slight variation of the dual number system demonstrated in ref. [113]. The dual numbers are introduced here because of the need for a mechanism by which to precisely compute the sensitivity of the wall model to the sample state.

Dual numbers of order N are hypercomplex numbers of the form

$$d = d_0 + d_1\varepsilon_1 + d_2\varepsilon_2 + \cdots + d_N\varepsilon_N, \tag{3.56}$$

where $d_i \in \mathbb{R}$, and with the product rule

$$\varepsilon_i\varepsilon_j = \frac{(i+j)!}{i!j!}\varepsilon_{i+j} \tag{3.57}$$

when $i + j \leq N$ and

$$\varepsilon_i\varepsilon_j = 0 \tag{3.58}$$

otherwise. Note that $\varepsilon_0 = 1$. For the sake of this work, $N = 2$ is considered, since the second-order sensitivity terms will be required., giving the multiplication table

$$\begin{pmatrix} \varepsilon_0 \\ \varepsilon_1 \\ \varepsilon_2 \end{pmatrix} \begin{pmatrix} \varepsilon_0 & \varepsilon_1 & \varepsilon_2 \end{pmatrix} = \begin{pmatrix} 1 & \varepsilon_1 & \varepsilon_2 \\ \varepsilon_1 & 2\varepsilon_2 & 0 \\ \varepsilon_2 & 0 & 0 \end{pmatrix}. \tag{3.59}$$

Dual numbers of order 2 then have the following properties:

$$(a_0 + a_1\varepsilon_1 + a_2\varepsilon_2) \pm (b_0 + b_1\varepsilon_1 + b_2\varepsilon_2) = (a_0 \pm b_0) + (a_1 \pm b_1)\varepsilon_1 + (a_2 \pm b_2)\varepsilon_2 \tag{3.60}$$

$$(a_0 + a_1\varepsilon_1 + a_2\varepsilon_2)(b_0 + b_1\varepsilon_1 + b_2\varepsilon_2) = (a_0b_0) + (a_0b_1 + a_1b_0)\varepsilon_1 + (a_0b_2 + 2a_1b_1 + a_2b_0)\varepsilon_2 \tag{3.61}$$

$$\begin{aligned}\frac{a_0 + a_1\varepsilon_1 + a_2\varepsilon_2}{b_0 + b_1\varepsilon_1 + b_2\varepsilon_2} &= (a_0/b_0) + \left(\frac{b_0a_1 - b_1a_0}{b_0^2}\right)\varepsilon_1 \\ &\quad + \left(\frac{b_0^2a_2 - 2b_0b_1a_1 - b_0b_2a_0 + 2b_1^2a_0}{b_0^3}\right)\varepsilon_2,\end{aligned} \tag{3.62}$$

with clear connections to the product and quotient differentiation rules.

For analytic functions, performing Taylor expansion gives the (exact) expression

$$f(a_0 + a_1\varepsilon_1 + a_2\varepsilon_2) = f(a_0) + (a_1\varepsilon_1 + a_2\varepsilon_2)f'(a_0) + \frac{1}{2}(a_1\varepsilon_1 + a_2\varepsilon_2)^2 f''(a_0)$$

$$f(a_0) + (a_1f'(a_0))\,\varepsilon_1 + \left(a_2f'(a_0) + a_1^2f''(a_0)\right)\varepsilon_2, \tag{3.63}$$

yielding the useful property for any analytic function

$$f(a_0 + \varepsilon_1) = f(a_0) + f'(a_0)\varepsilon_1 + f''(a_0)\varepsilon_2. \tag{3.64}$$

Note that $\varepsilon_1^2 = 2\varepsilon_2$ This property can readily be extended to an analytic multivariate function $f(x, y, z)$ as

$$\begin{aligned}&f(a_0 + a_1\varepsilon_1 + a_2\varepsilon_2, b_0 + b_1\varepsilon_1 + b_2\varepsilon_2, c_0 + c_1\varepsilon_1 + c_2\varepsilon_2) \\ &= f(a_0, b_0, c_0) \\ &+ (a_1f_x + b_1f_y + c_1f_z)\,\varepsilon_1 \\ &+ \left(a_2f_x + b_2f_y + c_2f_z + 2a_1b_1f_{xy} + 2a_1c_1f_{xz} + 2b_1c_1f_{yz} + a_1^2f_{xx} + b_1^2f_{yy} + c_1^2f_{zz}\right)\varepsilon_2.\end{aligned} \tag{3.65}$$

The expressions (3.63) and (3.64) differ from the equivalents in the number systems in refs. [113, 114] slightly in that the identity (3.63) is derived directly from the multiplication table

(3.59). This results in a different definition for the evaluation of transcendental functions but is useful in that the resulting identity for the multivariate case can more readily be applied to the approximations in this chapter.

3.4.2 Sensitivity Estimate

Setting $a_2 = b_2 = c_2 = 0$ in (3.65) results in the identity

$$\begin{aligned} f(a_0 + a_1\varepsilon_1, b_0 + b_1\varepsilon_1, c_0 + c_1\varepsilon_1) &= f(a_0, b_0, c_0) + (a_1 f_x + b_1 f_y + c_1 f_z)\,\varepsilon_1 \\ &+ \left(2a_1b_1 f_{xy} + 2a_1c_1 f_{xz} + 2b_1c_1 f_{yz} + a_1^2 f_{xx} + b_1^2 f_{yy} + c_1^2 f_{zz}\right)\varepsilon_2, \end{aligned} \tag{3.66}$$

which matches the form of the error terms (3.52) and (3.54). The unsteady correction for the wall model $\underline{\tau}_w(u_h, T_h, \rho_h)$ can then simply be found by solving in the 2^{nd}-order dual numbers, with the first-order dual coefficients set to the instantaneous fluctuation, resulting in the estimate

$$\begin{aligned} &\underline{\tau}_w(u_h + u_h'\varepsilon_1, T_h + T_h'\varepsilon_1, \rho_h + \rho_h'\varepsilon_1) \\ &= \underline{\tau}_w(u_h, T_h, \rho_h) + \left(u_h'\frac{\partial \underline{\tau}_w}{\partial u_h} + T_h'\frac{\partial \underline{\tau}_w}{\partial T_h} + \rho_h'\frac{\partial \underline{\tau}_w}{\partial \rho_h}\right)\varepsilon_1 \\ &+ \left(u_h'u_h'\frac{\partial^2 \underline{\tau}_w}{\partial u_h^2} + T_h'T_h'\frac{\partial^2 \underline{\tau}_w}{\partial T_h^2} + \rho_h'\rho_h'\frac{\partial^2 \underline{\tau}_w}{\partial \rho_h^2}\right. \\ &\left. + 2u_h'T_h'\frac{\partial^2 \underline{\tau}_w}{\partial u_h \partial T_h} + 2u_h'T_h'\frac{\partial^2 \underline{\tau}_w}{\partial u_h \partial T_h} + 2u_h'T_h'\frac{\partial^2 \underline{\tau}_w}{\partial u_h \partial T_h}\right)\varepsilon_2. \end{aligned} \tag{3.67}$$

With the notation

$$\underline{\tau}_w(u_h + u_h'\varepsilon_1, T_h + T_h'\varepsilon_1, \rho_h + \rho_h'\varepsilon_1) = \underline{\tau}_w + \underline{\tau}_w^{(1)}\varepsilon_1 + \underline{\tau}_w^{(2)}\varepsilon_2, \tag{3.68}$$

the corrected wall shear stress is then simply

$$\underline{\tau}_{w,\text{corrected}} = \underline{\tau}_w - \underline{\tau}_w^{(1)} - \frac{1}{2!}\underline{\tau}_w^{(2)} \quad \text{and} \quad \underline{q}_{w,\text{corrected}} = \underline{q}_w - \underline{q}_w^{(1)} - \frac{1}{2!}\underline{q}_w^{(2)} \tag{3.69}$$

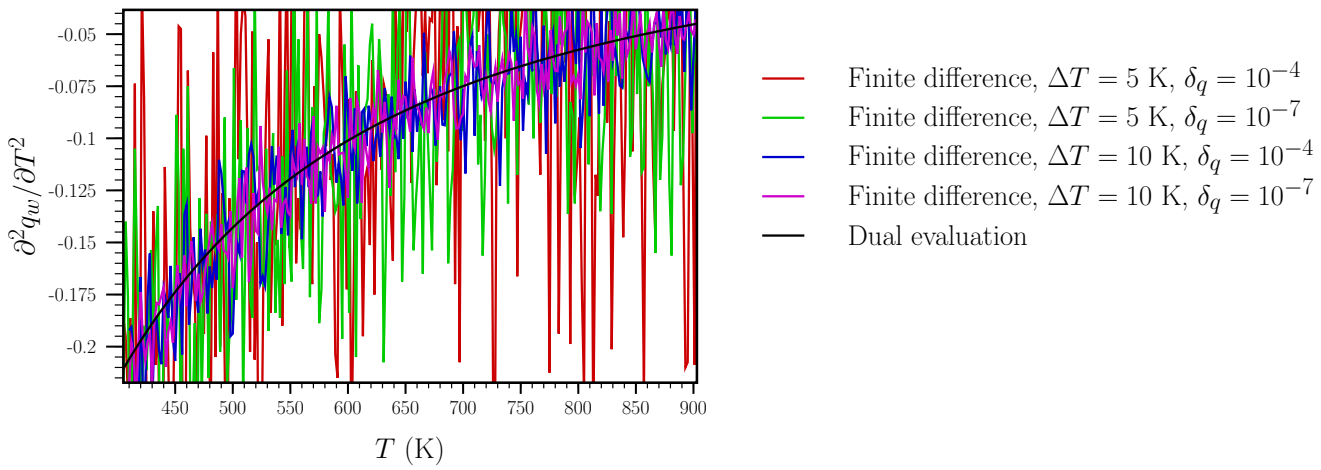


Figure 3.11: Comparison of finite-difference estimate and dual calculation of second derivative of heat transfer from ODE wall model.

Note that an N^{th}-order correction is possible simply by using N^{th}-order dual numbers and correcting as

$$\underline{\tau}_{w,\text{corrected}} = \underline{\tau}_w - \sum_{j=1}^{N} \frac{1}{j!}\underline{\tau}_w^{(j)} \quad \text{and} \quad \underline{q}_{w,\text{corrected}} = \underline{q}_w - \sum_{j=1}^{N} \frac{1}{j!}\underline{q}_w^{(j)} \tag{3.70}$$

To demonstrate the utility of dual numbers for the sensitivity evaluation, a comparison with numerical differentiation is shown in figure 3.11 at a range of temperature inputs at particular values of u_h and ρ_h. A finite-difference calculation was made with a step size of ΔT and the wall model ODE system was converged to within a tolerance of δ_q. Clearly, and intolerable level of noise is present in all finite-difference calculations. The finite-difference approximations are less accurate for smaller step sizes in temperature, indicating that numerical precision limits have been exceeded before convergence of the derivative estimate has been achieved. By contrast, the dual number estimate is smooth, which is to be expected given that it is an exact calculation. This shows that the dual number system employed here is a far more robust method of computing the sensitivity of the wall model, since there is no noise. This approach has the added value of having no finite difference parameters or error tolerances. This is consistent with other similar methods such as the complex step method [115]. Note that there is an additional computational cost for solving the wall model in dual numbers, but it is negligible compared to using a finite difference approximation.

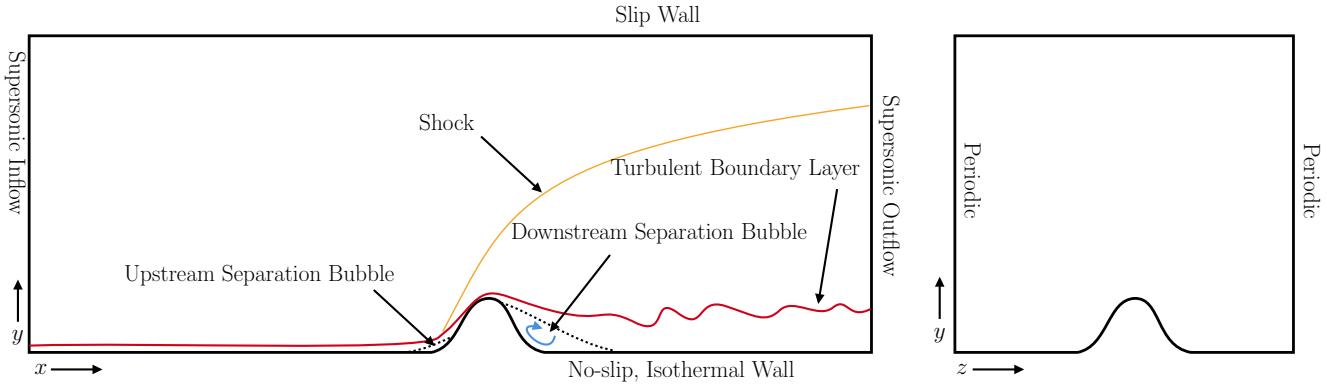


Figure 3.12: Diagram of $M = 4$ 3D bump flow test case.

3.5 Investigation for 3-Dimensional Flow Problem

3.5.1 Case Description

Until this point, the unsteadiness effects have been investigated in the context of hypersonic channel flow. While useful to illustrate the dynamics in question, channel flows are a poor representation of the broader set of flow problems that unsteady effects may be relevant for. In this section, unsteady effects are investigated for a more general hypersonic flow problem.

The test case under consideration is a flat plate with a large bump in the centre. The bump has been made sufficiently large to promote separation upstream and downstream, and the Reynolds number is chosen to have a sufficiently turbulent boundary layer downstream of the bump.

The setup for this case is shown in figure 3.12. The computational domain is $(x, y, z) \in [0, 5L] \times [0, 2L] \times [0, 0.5L]$. Grid spacing on the finest level is chosen to be $\Delta x = 6.5 \cdot 10^{-4} L$, $\Delta y = 6.5 \cdot 10^{-4} L$, and $\Delta z = 1.3 \cdot 10^{-3} L$. A boundary layer of thickness $\delta_0 = 10^{-2} L$ is imposed at the inflow boundary. The form of the bump is given by the expression

$$h(x, z) = a e^{-k((x-x_0)^2 + (z-z_0)^2)}, \tag{3.71}$$

where $x_0 = 2.5L$, $z_0 = 0.25L$, $a = 0.15L$, and $k = 335/L^2$.

Sutherland's law is used to set the viscosity, with a freestream temperature of 250 K. The freestream Mach number is 4 and the Reynolds number is $\mathrm{Re}_L = 31$ million. Figure 3.13 shows a slice of instantaneous streamwise velocity at $z = 0.25L$. At this grid resolution, the upstream

separation bubble is present. The downstream separation bubble exhibits low-frequency unsteadiness.

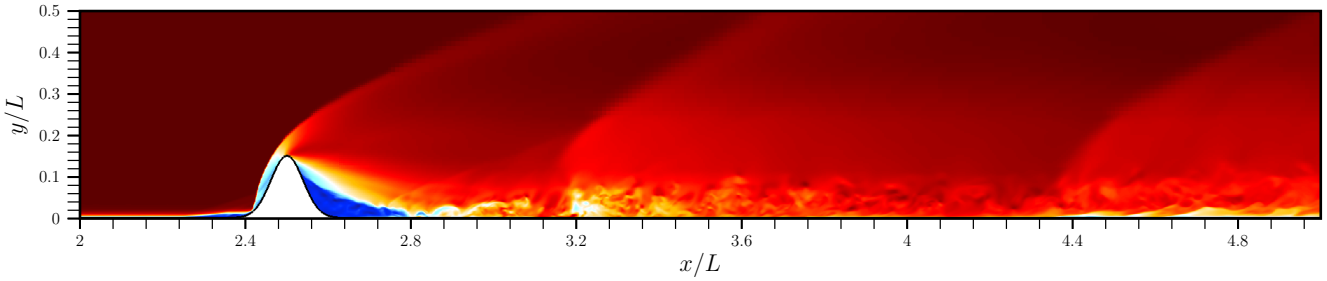


Figure 3.13: Visualisation of streamwise velocity for the 3D, $M = 4$ bump flow evaluation case.

Figure 3.14 shows instantaneous and time-averaged temperature, spanwise velocity, and streamwise velocity. The shock wave generated by the bump is reflected off of the periodic boundary, leading to a diamond-shaped flow topology pattern and unsteadiness in the upstream separation bubble.. This results in a series of shock wave-boundary layer interactions that are situated in the wake of the bump. Notably, the mean streamwise velocity near $x = 4.4L$ shows significant flow deceleration that is the result of interaction of the boundary layer with the reflected shock.

3.5.2 Unsteady Error Investigation

This test case was contrived to contain the flow features listed above as an investigation of the spatial distribution of wall-model unsteadiness effects for a more complex flow. The coupled simulation was performed without the unsteady correction, but unsteady wall model data was saved regularly. After the simulation was performed, the unsteady errors ε_τ and ε_q from (3.52) and (3.54) respectively are computed directly by computing the ageraged sample state and comparing the wall model evaluation against the mean viscous flux from the simulation. The surface distributions of ε_τ and ε_q for this case are shown in figure 3.15.

Over most of the surface shown in figure 3.15, the unsteady errors are relatively small, but not entirely insignificant. In regions of flow separation, the unsteady error become much more significant since both heat transfer and skin friction are reduced within separation bubbles. Generally, relative errors are larger in regions where the turbulent boundary layer interacts with shock waves. Note that along separation contours, skin friction is zero and thus the relative error is not

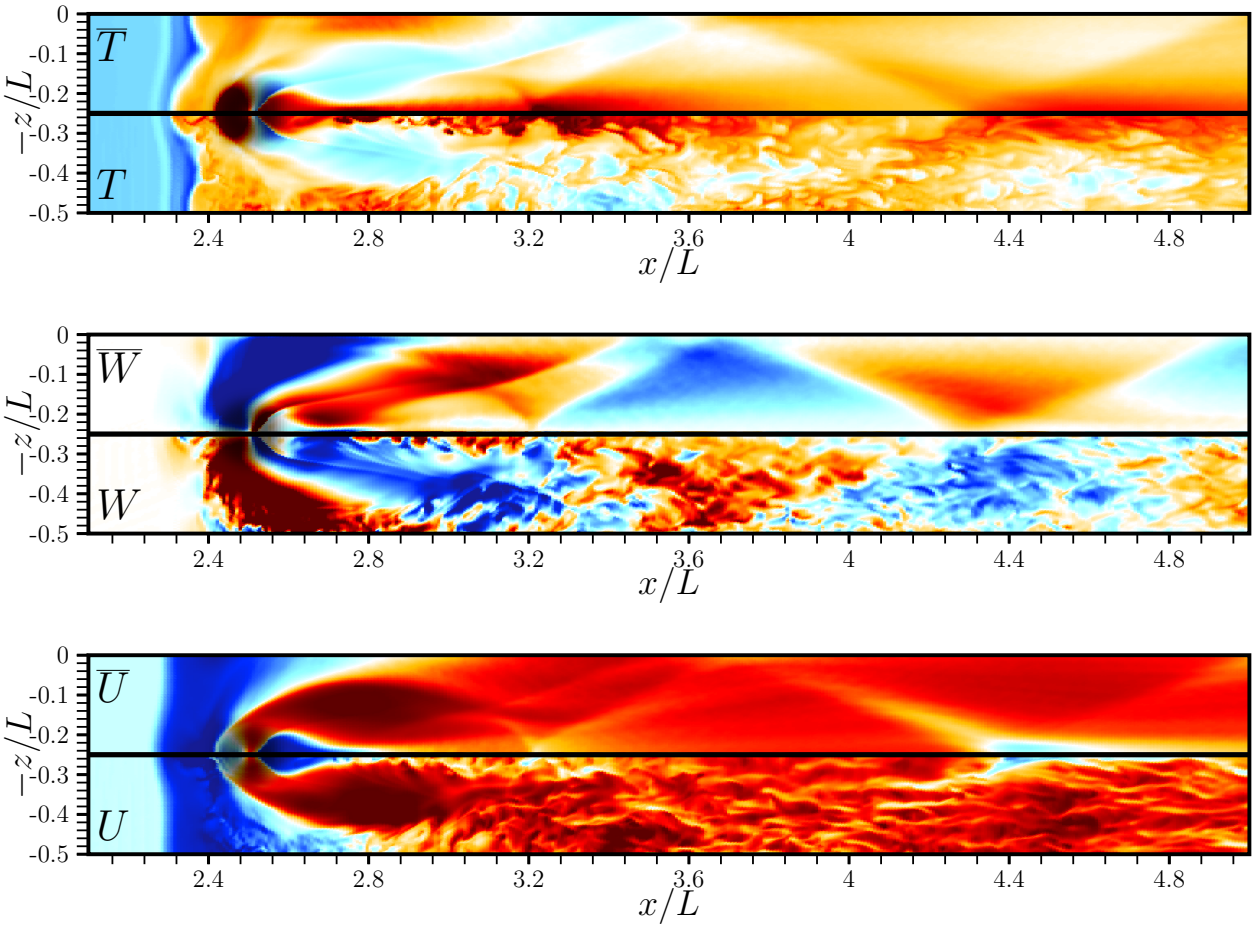


Figure 3.14: Flow topology visualisation for the hypersonic bump evaluation case.

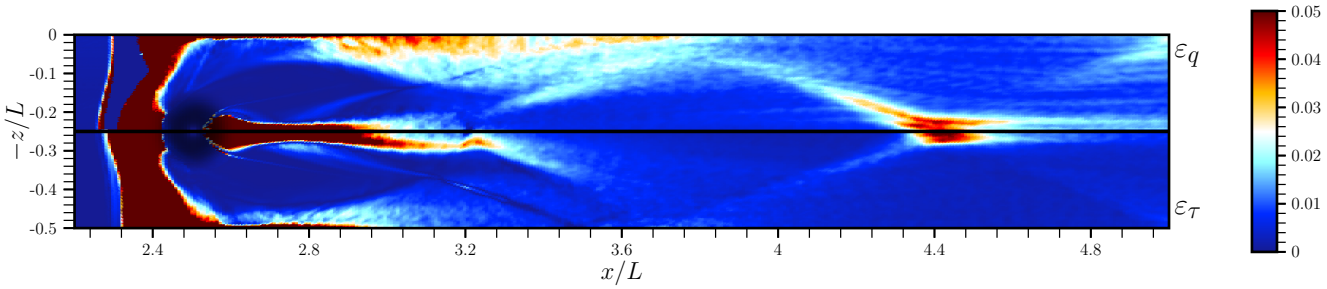


Figure 3.15: Surface distribution of unsteadiness errors for heat transfer and skin friction for the bump flow case.

well-defined.

Figure 3.16 shows the comparison between the corrected ($\varepsilon_{\bullet,\text{corrected}}$) and uncorrected ($\varepsilon_\bullet$) unsteady errors averaged over the surface data of the bump case. The error terms are computed

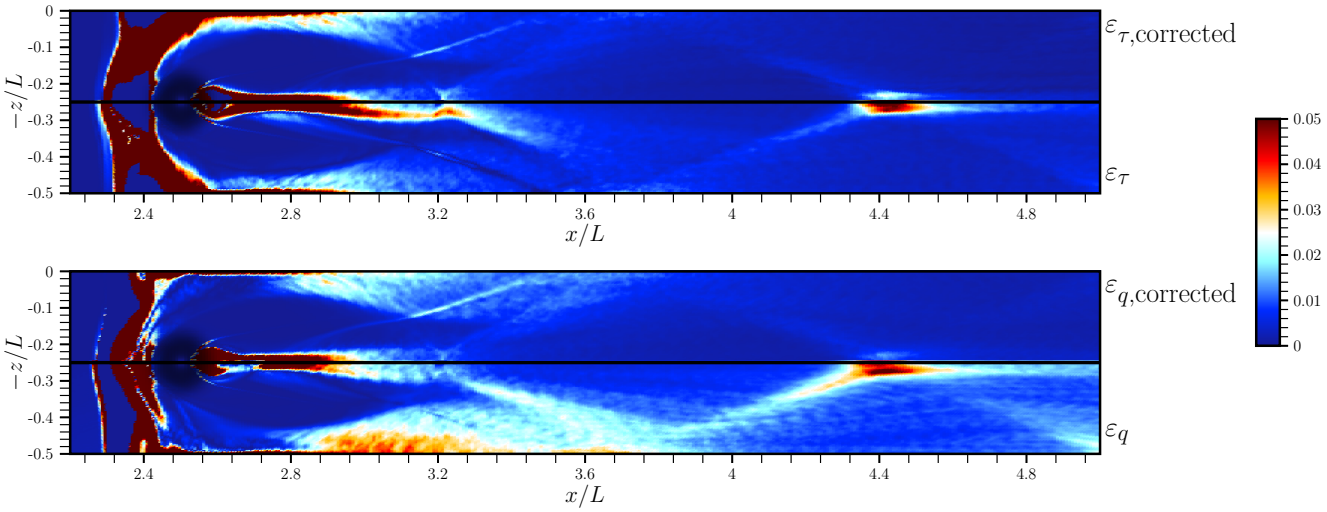


Figure 3.16: Comparison of unsteady uncorrected error and corrected error terms for heat transfer and skin friction.

for $\phi \in \{\underline{\tau}_w, \underline{q}_w\}$ as

$$\varepsilon_\phi = \frac{\overline{\phi(u_h, T_h, \rho_h)}}{\phi(\overline{u_h}, \overline{T_h}, \overline{\rho_h})} - 1 \quad \text{and} \quad \varepsilon_{\phi,\text{corrected}} = \frac{\overline{\phi(u_h, T_h, \rho_h) - \frac{1}{2!}\phi^{(2)}}}{\phi(\overline{u_h}, \overline{T_h}, \overline{\rho_h})} - 1. \tag{3.72}$$

While the use of the dual-number correction had modest improvements on the skin friction prediction, the reduction in unsteady error is more significant for the surface heat transfer. In particular, the most significant reduction is seen in regions of shock wave-boundary layer interaction within the turbulent region, especially near the decelerated region at the centre of the domain near $x/L = 4.4$. There are still unsteady errors present in both surface quantity predictions, but they are generally concentrated in regions of low-frequency unsteadiness where the time-average does not describe the behaviour of the flow.

3.6 Summary

Wall models are typically designed by comparing against DNS data *a priori*. However, models that perform favourably *a priori* may not perform well *a posteriori*. Two wall models were chosen in this section where performance evaluation in standalone mode, both by mean profile comparison and budget analysis, favoured one over the other, but evaluation with full WMLES reversed the comparison. It was shown that perfect performance in standalone mode does not imply

superior performance in a coupled simulation. This was done by deriving two models that perfectly match a DNS profile but have different performance in a coupled simulation. The difference between these two models was an error arising from the commutation of the wall model and time averaging operators. This error was demonstrated to be more significant for high Mach and Reynolds numbers, supported by analysis. It was shown that the unsteady error be corrected by estimating the commutation error via Taylor expansion, which significantly improved results. A simple procedure for estimating this error in live simulations was derived and it was shown that it was capable of significantly reducing commutation error for heat transfer in important flow regions.

Chapter 4: Computational Framework

In section 1.4, it was claimed that the use of block-structured Cartesian grids results in a computationally efficient framework. The IBM-WMLES approach generally requires a larger number of grid cells, since the tree-based mesh structure means that cells cannot be as efficiently distributed as they can be when making use of grid stretching. However, the regular grid spacing and block size allow for a much more compact representation of the flowfield that has much more favourable computational properties, namely memory locality and arithmetic intensity, than alternative approaches.

The aim of this chapter is to demonstrate that even with the tradeoff of having more grid cells, the use of IBM-WMLES, in combination with the computational framework outlined in the latter sections, allows for remarkable computational performance that is arguably comparable to the computational cost of RANS calculations.

4.1 Background

Increasing computational demand from artificial intelligence applications has led to broader availability and more competitive pricing of high-performance GPUs. Recent consumer-grade models, such as the NVIDIA RTX 4090, offer significant computational performance, achieving as high as 88 teraflops in single precision. This level of performance enables many simulation tasks that once required access to HPC infrastructure to be executed on a single high-end workstation. As a result, targeting GPU architectures has become a practical and efficient strategy for accelerating CFD simulations across a wide range of scales.

Recently, Pasquariello et al. [116] were able to achieve excellent performance for a finite-volume cut-cell solver by natively designing the solver for GPU architecture. By employing load balancing, using single precision, using a second-order centred flux, and employing best practices for GPU development, they were able to achieve similar results to a commercial WMLES solver on one-twelfth the number of GPUs. Wei et al. [117] present a two-dimensional GPU solver and highlight the importance of shared memory and the kernel configuration. Davis et al. [118] demonstrate the porting of a solver using a pre-built adaptive mesh refinement (AMR) library

and achieved promising scaling results up to large numbers of nodes, but also demonstrated that even computationally-intensive kernels sometimes achieve low arithmetic performance relative to the theoretical ceiling. Xue et al. [119] achieve a 16× speedup over a comparable CPU code using OpenACC, although require 16 GPUs for a grid size of 5 million cells. Jude et al. [120] present a Cartesian octree-based solver with speedups ranging between 3 and 12 depending on hardware.

Much of the existing work adapts CPU-based solvers for GPU execution, often out of pragmatic necessity. However, this can impose constraints on data layout and parallelism that limit performance on modern GPU hardware. GPU programming requires explicit management of parallelism, memory hierarchies, and synchronisation to avoid errors and performance traps. While low-level APIs (e.g. CUDA, HIP) expose fine-grained control, they increase development complexity. In this work, SPADE [121] is used, which is a performance portability framework written in C++20 developed by the author. SPADE simplifies kernel expression and memory management across CPU and GPU architectures. SPADE's modern template-metaprogramming constructs and backends for CUDA and HIP allow the user to write concise, portable code while retaining control over low-level optimisations. SPADE takes a GPU-first approach and is designed from the outset around GPU architecture. This enables more effective use of memory hierarchy, threading, and kernel structure without the compromises typical of CPU-to-GPU porting.

4.2 Test Problems

Two test problems are chosen for performance evaluation in this chapter:

1. the compressible Taylor-Green Vortex (TGV) with a mesh size of 384^3 cells (56M cells), and
2. external flow around the airfoil considered by Tamaki and Kawai [4] with a grid size of 60M cells.

Figure 4.2 shows the solenoidal dissipation rate of total kinetic energy for the compressible TGV probelm. The solenoidal dissipation is given by

$$\varepsilon_k = \frac{L^2}{ReU_0^2|\Omega|}\int_\Omega \frac{\mu(\mathrm{T})}{\mu_0}\boldsymbol{\omega}\cdot\boldsymbol{\omega}d\Omega, \tag{4.1}$$

where L is the reference length scale, Re is the Reynolds number, U_0 is the reference velocity, Ω is the domain volume, $\boldsymbol{\omega}$ is the vorticity, and μ_0 is the viscosity at the reference temperature. Note that Sutherland's law is used. See ref. [3] for more details.

The mesh for the airfoil test problem is depicted in figure 4.1. Two grid levels are used, each using the same topology. The coarse mesh contains 12^3 cells per block, and the fine mesh contains 16^3 cells per block. The coarse grid is used for performance evaluation, since the total grid size is closer to that of the TGV problem. The freestream Mach number is set to 0.15, the angle of attack is 13.3°, and the chord Reynolds number is 10 million. The finest grid spacing on the coarse mesh is $6 \times 10^{-4}c$, where c is the chord. A unit aspect ratio is used in the streamwise and wall-normal directions, with an aspect ratio of 5 used in the spanwise direction.

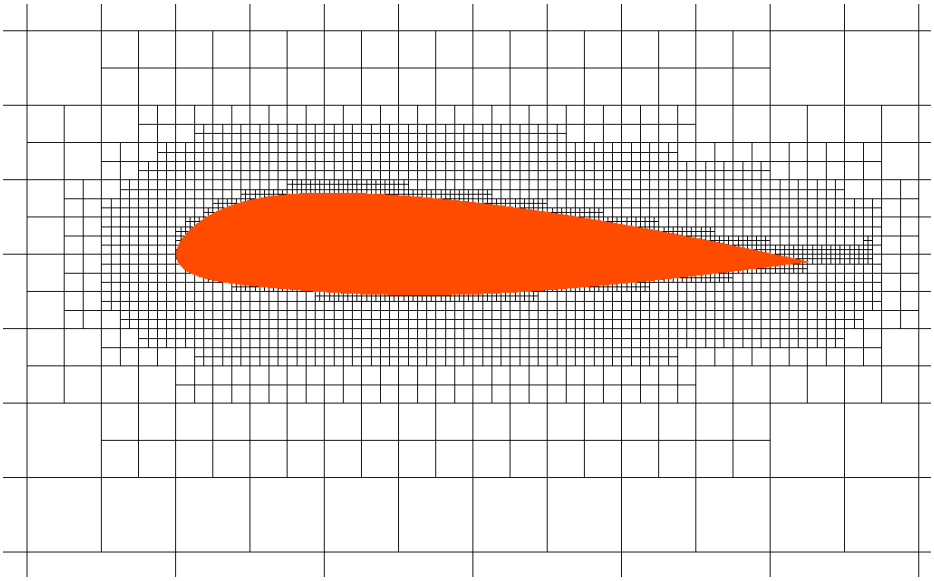


Figure 4.1: Airfoil test problem computational mesh. Grid blocks are shown.

Both problems are chosen to have a similar grid size that is large enough to use all of the computational resources of a single NVIDIA RTX4090. The same numerical treatment is used for each test problem. Problem 1 will be used to demonstrate the computational speedup of optimisations affecting calculations unrelated to the immersed boundary treatment or coarse-fine interfaces (CFIs), and problem 2 will be used otherwise.

Figure 4.3 shows validation of the surface pressure coefficient for problem 2.

Both of these comparisons are included to demonstrate that the optimised implementation is

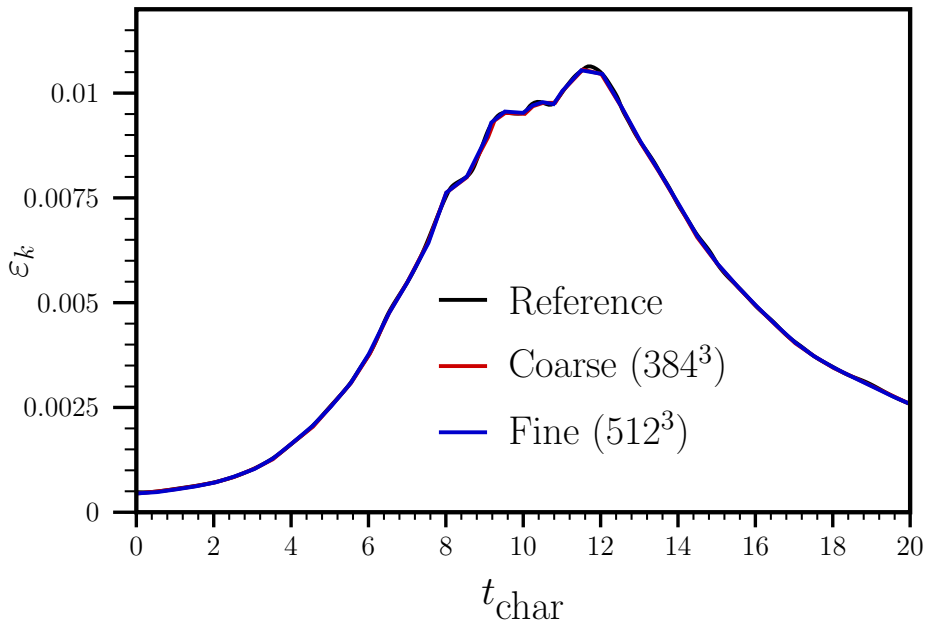


Figure 4.2: Validation for the Taylor-Green Vortex test problem: integrated solenoidal dissipation rate over time. Reference data from [3]. .

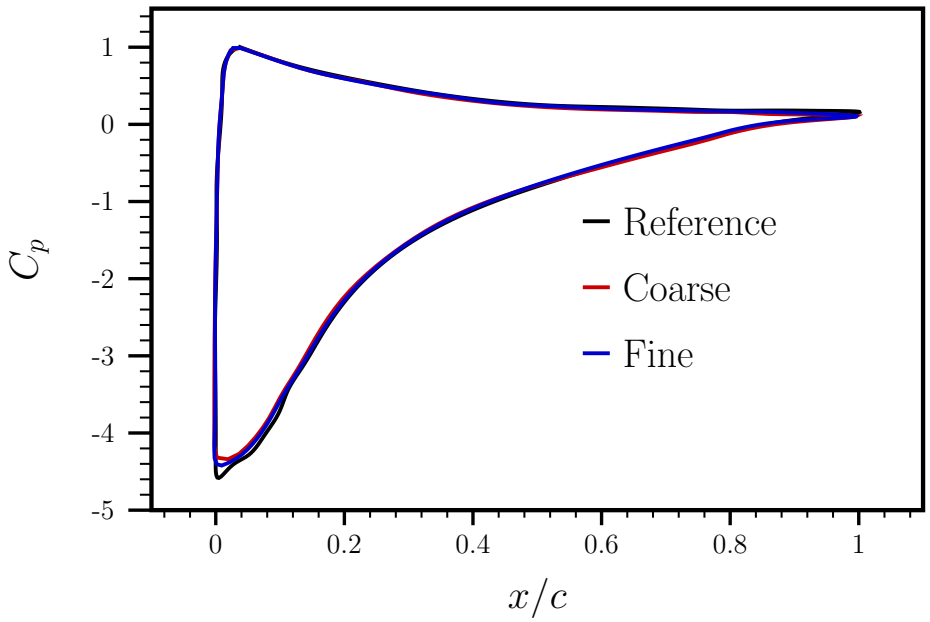


Figure 4.3: Validation for the airfoil test problem: surface pressure distribution. Reference data from [4].

minimally correct.

The numerical schemes used for these test problems are detailed in the rest of this thesis. Other than the presence of the immersed boundary and CFIs (and a small discrepancy in grid size), these two problems are identical from a computational perspective. In both cases, explicit

time integration is used, employing a classical 3-stage strong stability preserving Runge-Kutta (RK) scheme from Gottleib and Shu [122]:

$$k_0 = R\left(w^{(n)}\right) \tag{4.2a}$$

$$k_1 = R\left(w^{(n)} + \Delta t k_0\right) \tag{4.2b}$$

$$k_2 = R\left(w^{(n)} + \frac{1}{4}\Delta t k_0 + \frac{1}{4}\Delta t k_1\right) \tag{4.2c}$$

$$w^{(n+1)} = \frac{1}{6}\Delta t k_0 + \frac{1}{6}\Delta t k_1 + \frac{2}{3}\Delta t k_2. \tag{4.2d}$$

As a measure of absolute performance, millions of updates per second (MUPS) are used, where one update refers to advancing a single grid cell from timestep n to timestep $n+1$, which includes all boundary conditions, accumulations, and right-hand-side (RHS) computations. Performance will be expressed in MUPS when appropriate, and will be expressed in relative terms when considering incremental changes from specific optimisations. For the sections that follow, unless explicitly stated otherwise, all performance improvements are given for the Taylor-Green vortex test case.

4.3 GPU Architecture

The best performance for an implementation as described in this work will be on a GPU. With the popularity of artificial intelligence rapidly increasing (at the time of writing), high-performance GPU hardware is becoming widely available and ubiquitous on compute clusters and in consumer desktops and laptops.

High-performance CPU compute nodes generally have between 32 and 128 (e.g. AMD EPYC 7513 with 32 cores, Intel Xeon Platinum 8380 with 40 cores, Ampere Altra Max M128-30 with 128 cores, etc.) compute cores available for parallel processing, with hundreds of gigabytes (GB) of random access memory (RAM) available for storage. Memory access latency, even into the deep levels of main memory, is generally on the order of 10-20 clock cycles. By contrast, GPUs are equipped with many thousands of processing units capable of running in parallel, but have much less memory available (10-80 GB), and at a much greater latency cost of hundreds of clock cycles [123, 124]. However, by launching an enormous number of threads at once, GPUs are able to

hide the significant memory latency overhead behind large amounts of computation performed on previously fetched memory. These considerations generally lead to differing development principles in CPU and GPU implementations. A high-performance GPU implementation must carefully consider the amount of memory used, the amount of parallelism exposed in the algorithm of choice, and the way in which loaded memory is used.

GPU workloads are dispatched as kernels executed by many parallel threads, typically far more than the number of physical execution units. Threads are grouped into warps (or wavefronts), which are the smallest scheduling units. On NVIDIA hardware, warp size is fixed at 32 threads. Warps are further grouped into thread blocks, which execute on the same streaming multiprocessor and can share low-latency shared memory. Thread blocks support lightweight synchronisation primitives, enabling cooperation between threads. Since all threads in a warp execute instructions in lockstep, control flow divergence within a warp causes all branches to be executed serially, reducing efficiency.

To hide latencies, the GPU typically has compute pipelines that service requests from thread blocks that form a GPU kernel. When these thread blocks make requests for memory with high latency, other thread blocks can be swapped into the pipeline if they have memory values available for computation [125]. Generally, this rapid context-switching, coupled with the sheer volume of compute units available, is how GPU hardware so effectively speeds up large volumes of computation.

Within all simulations in this section, a mixed-precision approach is employed. All flowfield values are stored as single-precision, 32-bit floating point values. All calculations involving flowfield values are also performed in single precision. Coordinates, such as those describing the bounding boxes of grid blocks, surface geometry, and sample points are stored and computed using double precision. Generally, single precision has been observed to be sufficient for all simulations present in this thesis, with the exception of sharing of data between blocks at coarse-fine interfaces. In this instance, interpolation at the coarse-fine interface is computed using the extended precision emulation technique from Dekker [126].

4.4 Flux Divergence Calculation

The library underlying the solver in this work solves hyperbolic PDEs of the form

$$\frac{\partial T^{-1}(\mathbf{q})}{\partial t} + \frac{\partial \mathbf{F}_i(\mathbf{q})}{\partial x_i} = S(\mathbf{q}), \tag{4.3}$$

where $T: \mathbb{R}^n \to \mathbb{R}^n$ is an invertible transformation, $\mathbf{F}$ is the flux, and S is the source term. From a computational perspective, the flux divergence term $\partial \mathbf{F}_i(\mathbf{q})/\partial x_i$ represents the vast majority of the computational workload (i.e. floating point operations) as well as requiring a large memory throughput. This means that, in a naïve implementation, it is a computational bottleneck. This section will describe the approach taken towards calculating this term over the full domain. This chapter will ignore the source terms, as they are trivial to optimise.

In an effort to increase arithmetic intensity, all fluxes are combined into a single flux evaluation. By considering the fluxes as described in chapter 2, a complete domain of dependence of the full flux can be obtained. This domain of dependence drives the design of the algorithm in this section. Firstly, the inviscid flux required a stencil of four flowfield values in a line centred at each face. In order to compute the shock-sensing coefficient α, the full velocity gradient at the face must also be computed. For the viscous flux, the temperature and velocity gradients are required for the heat transfer, shear stress, and subgrid scale velocity. Additionally, the flow state must be averaged to the face for the viscous dissipation term and for the evaluation of the state-dependent viscous law. Taken together this produces the stencil shown in figure 4.4. The yellow point is centred at the face, and is where the flow state must be reconstructed via information at the red cells, and the gradient must be reconstructed from the flowfield values at the blue and red cells.

For reasons listed in section 4.8, the flux divergence calculation is performed over groups of $4 \times 4 \times 4$ (i.e., a thread block size of 64) cells to improve memory performance. When the stencil in figure 4.4 is placed at every cell face in a tile, the resulting shape represents the full domain of dependence for the flux divergence calculation. This domain of dependence is shown in figure 4.5. The full procedure is summarised in algorithm 2.

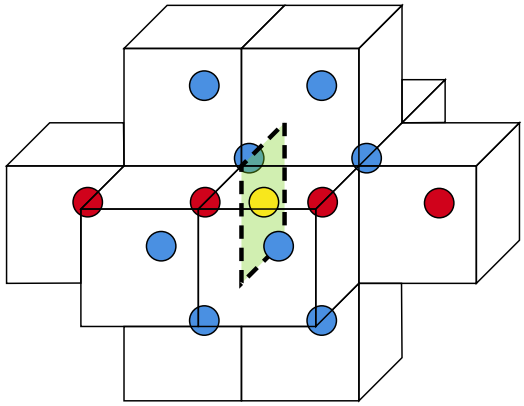


Figure 4.4: Stencil required for full flux calculation.

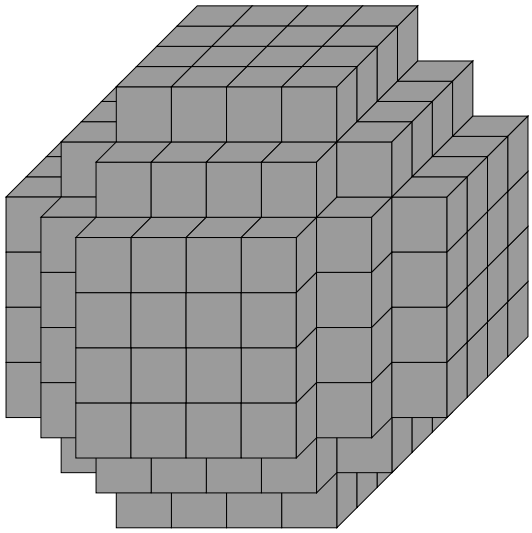


Figure 4.5: Stencil information required for full flux calculation over a single $4 \times 4 \times 4$ tile.

Algorithm 2 Per-thread update of flowfield values using shared memory

1: Compute global indices (i, j, k) for this thread
2: Declare `shared_flowfield[8][8][8]` in shared memory
3: **for** each of 8 thread-assigned points **do**
4: Load flowfield value from global memory
5: Store value into `shared_flowfield`
6: **end for**
7: Synchronise all threads
8: Load flowfield value at cell (i, j, k) from `shared_flowfield`
9: Transform to conservative variables
10: Compute RHS using neighboring values in `shared_flowfield`
11: Update conservative variable using computed RHS
12: Transform updated value back to primitive variables
13: Store result in new solution array

4.5 Memory Loading Patterns

As described previously, the manner in which memory is loaded and operated upon largely determines the performance of the resulting implementation. Furthermore, the performance coming from the loading pattern is tightly coupled with the memory map, which determines the way that flowfield values are arranged in memory. On GPUs, memory coalescence refers to the ability of threads within a warp to access consecutive memory locations in a single, aligned transaction. When memory accesses are coalesced, the hardware can combine these into a single memory operation, significantly improving effective bandwidth. In contrast, non-coalesced accesses result in multiple memory transactions per warp, increasing latency and underutilizing available bandwidth. This behaviour makes the layout of data in memory and the thread access pattern critical to performance. This is discussed in detail in section 4.8. Two loading patterns are considered a "balanced" loading pattern, where the total number of values loaded are minimised and spread across threads as evenly as possible, and an "octant" loading pattern, following a simplified algorithm. The balanced loading pattern is shown in figure 4.6.

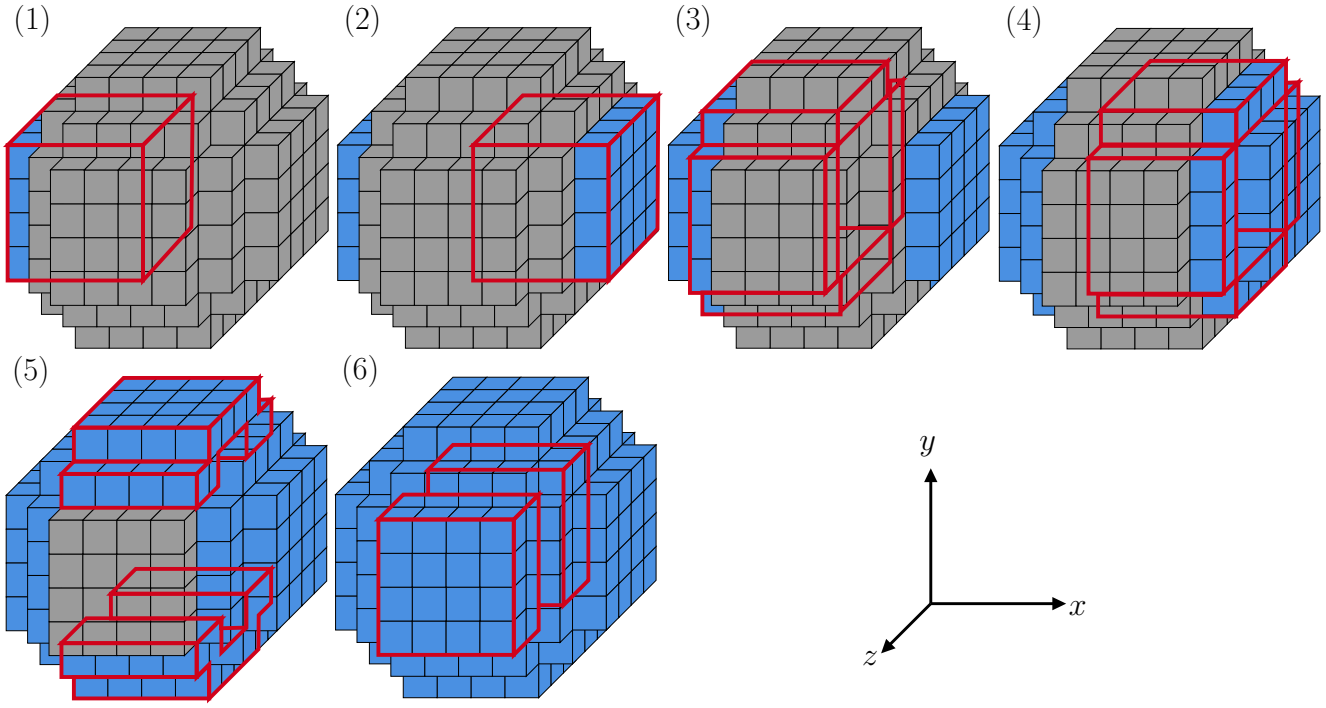


Figure 4.6: Balanced loading pattern in 6 stages using a thread block of 64 threads.

Using the balanced loading approach, memory is loaded in 6 stages. After the first four stages,

the data necessary for the fluxes in the x direction are available, and are thus calculated at this stage. Similarly, the y fluxes are calculated after stage 5, and the z fluxes after stage 6. The rationale behind the balanced loading approach is to interleave computation of the fluxes with the loading of flowfield data to allow for context-swapping on a finer scale, and to ensure that the minimal amount of data is loaded for the full flux computation at the expense of worse cache performance (see section 4.8) and minor logical divergence.

The loading pattern using the octant strategy is shown in figure 4.7. In contrast to the balanced loading pattern, all data is loaded first, and then all fluxes are computed and differentiated. This loading pattern has no logical divergence and better memory performance, but loads unnecessary memory.

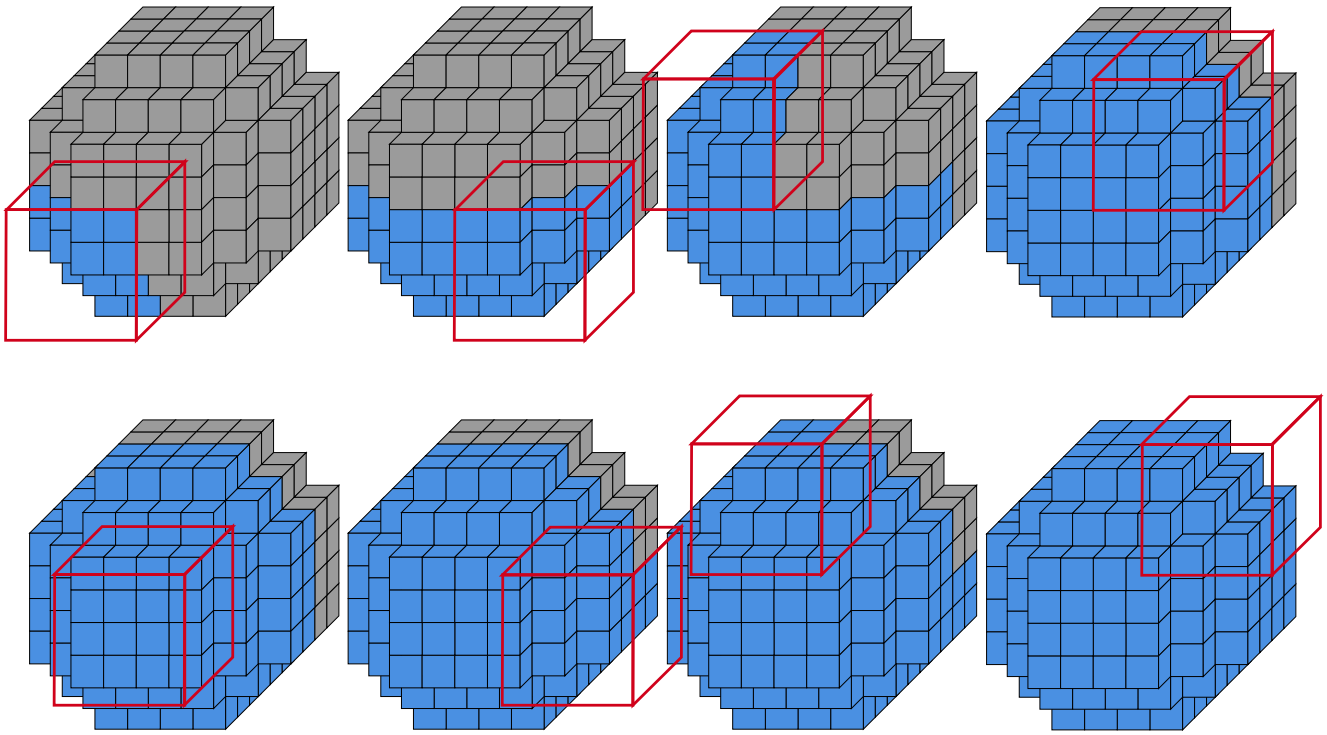


Figure 4.7: Octant loading pattern in 8 stages using a thread block of 64 threads.

Because the loading pattern and the choice of memory map are so tightly coupled, it is not possible to discuss their implications on performance separately. Therefore, the results from these two loading patterns can be found in section 4.8. For performance comparisons in the rest of this section, the octant loading pattern is used.

4.6 Use of Shared Memory

A shared memory array of $8 \times 8 \times 8$ is used to store flowfield values for the full domain of dependence over a $4 \times 4 \times 4$ tile. Once these values are available, the flux data structure is loaded from the shared memory block, including the calculation of face averages and gradients. When possible, face data is calculated directly from the stencil data rather than the shared memory array. The memory map used for the shared memory array also has an effect on performance, which is discussed further in section 4.8. The use of shared memory for the optimised implementation was tested against the same implementation, but fetching stencil values directly from global memory instead. The results of this test are shown in table 4.1, where the evaluated performance quantity is the wall-clock time taken by the flux divergence kernel for problem 1. Although this is a relatively simple aspect of the algorithmic implementation, the results demonstrate that there is a profound performance implication.

Table 4.1: Performance results for shared memory loading and global memory loading.

Description	**Improvement over baseline**	**MUPS**
Global memory (baseline)	0.0%	1491
Shared memory	107.0%	3087

4.7 Loop Unrolling

Similarly to section 4.6 another simple yet impactful optimisation is the use of compile-time looping for memory loading and the Cartesian direction for the fluxes. For the time taken to compute the flux divergence, the relative speedup given by loop unrolling is shown in table 4.2.

Table 4.2: Performance results for shared memory loading and global memory loading.

Description	**Improvement over baseline**	**MUPS**
Loop unrolling (baseline)	0.0%	296
No loop unrolling	944.0%	3087

The use of loop unrolling clearly has a profound impact on total performance of this kernel. Profile comparisons between the cases in table 4.2 suggest that with no loop unrolling, there is a significant increase in the number of warps stalled (i.e. awaiting data for further computation),

suggesting that the compiler was capable of re-ordering load instructions and compute instructions to allow for more context-switching. A full discussion of the implications of loop unrolling is too complex to be included here, and all implementations in this chapter use loop unrolling where possible.

4.8 Memory Map

4.8.1 Simplified Memory Model

As described previously, global memory accesses can have latencies of hundreds of clock cycles. Clearly, this means that the minimal amount of memory should be loaded in a kernel, barring pathologically high arithmetic intensity. When loading values from global memory, values are loaded sequentially in cache lines. When sequential threads of the same half-warp issue requests for values with sequential addresses in global memory, a single cache line can be loaded to service this memory request, since all values will be available locally. For strided memory accesses, a series of global loads will need to be made in order to service the range of requested addresses. This is illustrated in figure 4.8 for a warp size and cache line length of 16, and for access strides of 1, 2, and 3.

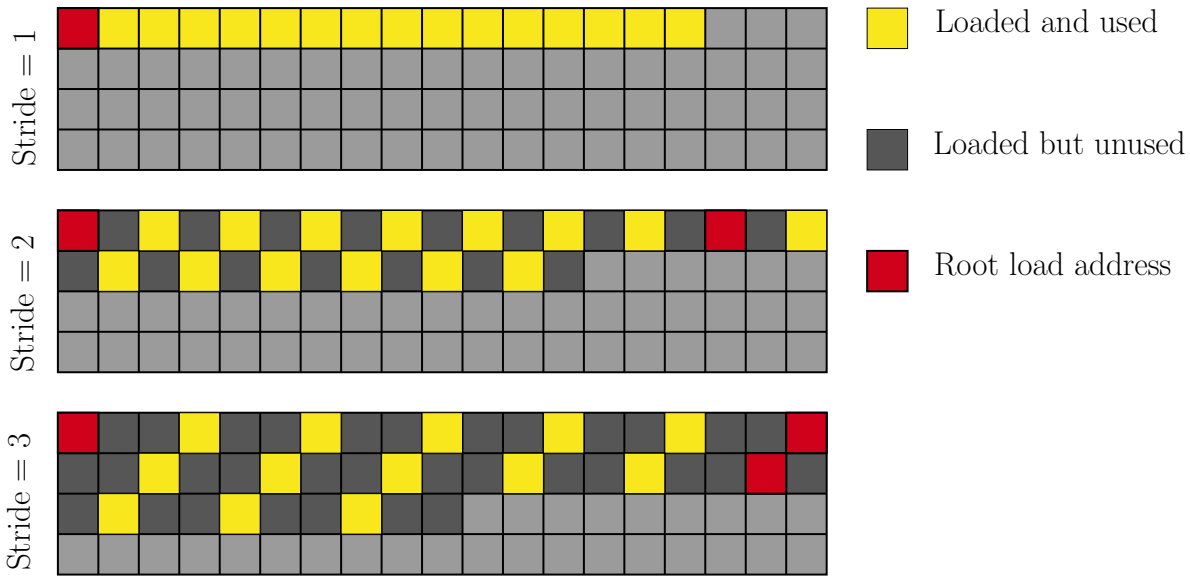


Figure 4.8: Simplified model of global memory loads for a warp size and cache size of 16.

While the description in this section is incomplete and simplified (see [125] for more details),

it will be used as an approximate model. for organising the address space of the flowfield and residual arrays. For the sake of the design of the present algorithm, global memory loads are modelled using a cache line length of 32 and a warp size of 32.

As mentioned in section 4.3, shared memory is a low-latency memory space available to all threads in a thread block simultaneously, and usually residing in the lowest-level cache in a reserved region that bypasses cache-hit checking. Shared memory is usually organised into banks that serialise access to different elements by individual threads. This is shown in figure 4.9.

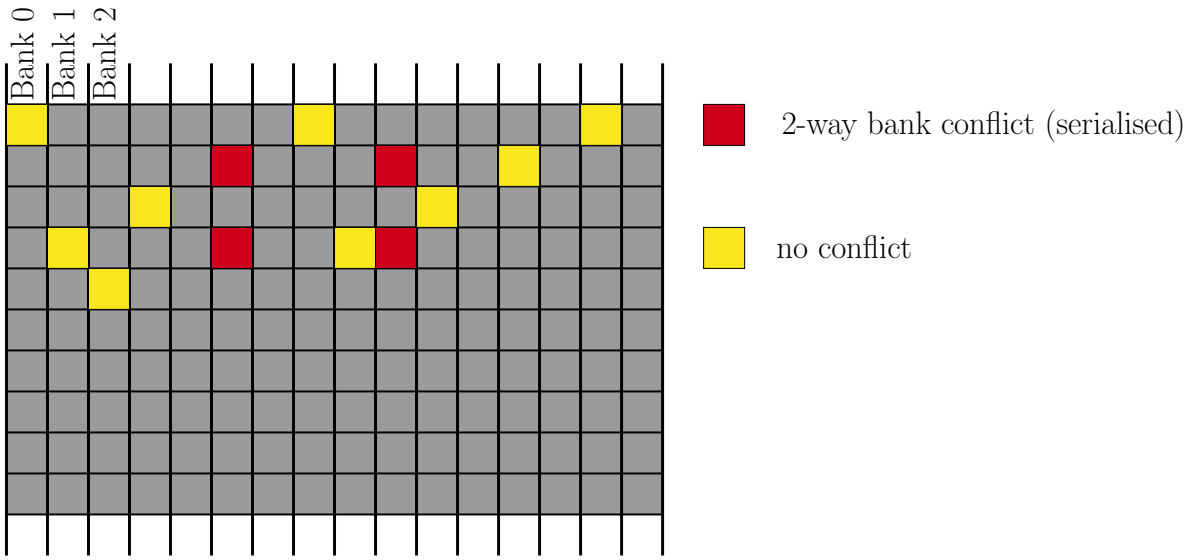


Figure 4.9: Simplified model of shared memory bank conflicts for a warp size of 16.

4.8.2 Memory Map Evaluation

The procedures described in Figures 4.8 and 4.9 are simple to model. The data loading pattern outlined in section 4.5 involves a specific sequence of warp memory requests to both global and shared memory. These requests can be evaluated based on the number of global memory cache lines fetched (for global memory transactions) and the occurrence of shared memory bank conflicts (for shared memory). Nevertheless, selecting a mapping from grid cells to memory addresses is a prerequisite. In practice, the choice of this memory mapping is influenced by both the algorithm in use and the specific data loading pattern. The mappings presented here and in section 4.5 were chosen after a series of experiments.

Figure 4.10 shows four memory maps under consideration for this work. Note that memory

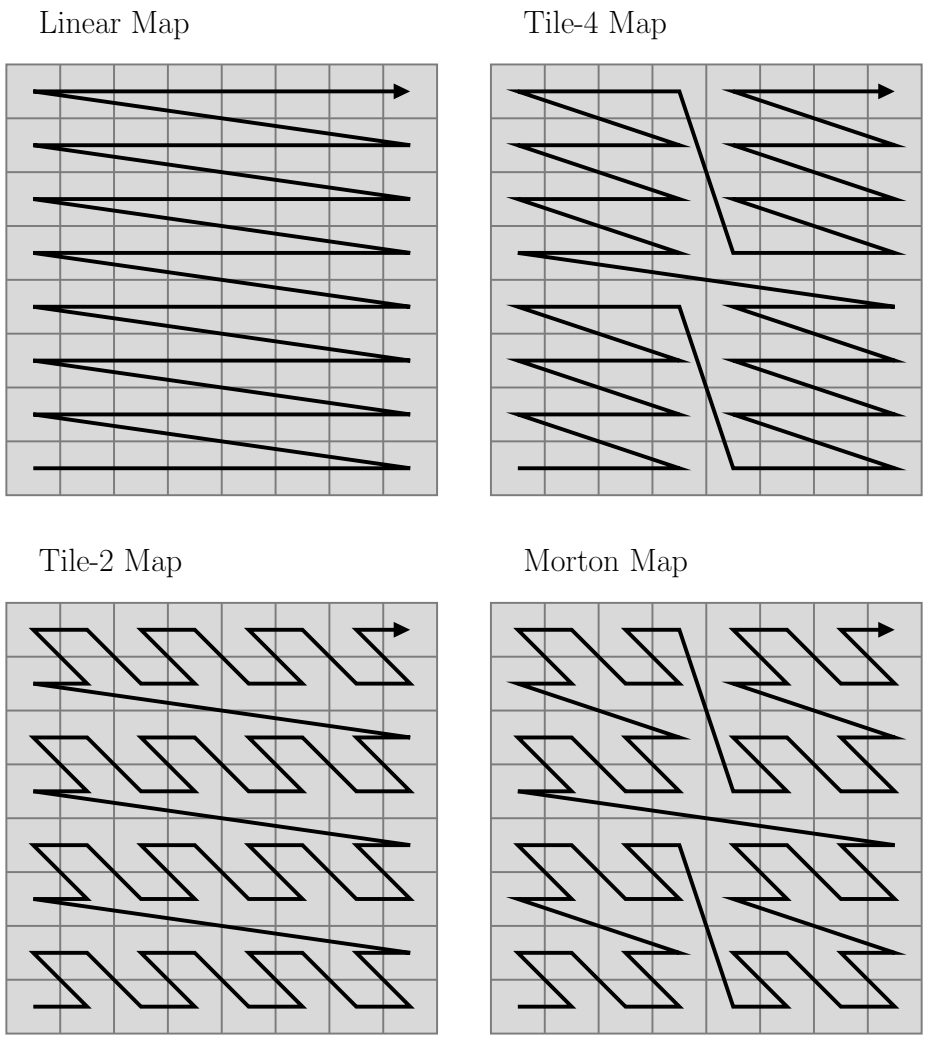


Figure 4.10: Comparison of linear (left), tile-4 (centre) and tile-2 (right) memory maps, showing order of memory addresses.

address offsets η are computed as

$$\text{Linear:} \quad \eta_L(v,i,j,k) = v + in_v + jn_v n_i + kn_v n_i n_j \tag{4.4}$$

$$\text{Tile-}n\text{:} \quad \eta_n(v,i,j,k) = \hat{i} + \hat{j}n + \hat{k}n^2 + vn^3 + \widetilde{i}n^3 n_v + \widetilde{j}n^3 n_v n_{\widetilde{i}} + \widetilde{k}n^3 n_v n_{\widetilde{i}} n_{\widetilde{j}}, \tag{4.5}$$

where i, j, k are the indices in each coordinate direction, v is the variable index, n_ϕ is the extent of the index ϕ, $\hat{\phi}$ is $\mod(\phi, n)$, and $\widetilde{\phi} = \phi/n$. In practice, the index modulo and division operations are implemented using bit shifting and bit masking. Note that the Morton-ordering map algorithm is well-known and thus is not expressed here. For this study, these four memory maps are used in various combinations for the shared memory pool in the flux divergence kernel, residual array,

and the flowfield array. Table 4.3 shows the various cases under consideration.

Table 4.3: Memory maps used in the eight evaluation cases.

Case ID	**Flowfield**	**Residual**	**Shared**	**Loading pattern**
LLL-B	Linear	Linear	Linear	Balanced
$T_2T_4T_2$-B	Tile-2	Tile-4	Tile-2	Balanced
$T_4T_4T_4$-B	Tile-4	Tile-4	Tile-4	Balanced
MT_4T_4-B	Morton	Tile-4	Tile-2	Balanced
LLL-O	Linear	Linear	Linear	Octant
$T_2T_4T_2$-O	Tile-2	Tile-4	Tile-2	Octant
$T_4T_4T_4$-O	Tile-4	Tile-4	Tile-4	Octant
MT_4T_4-O	Morton	Tile-4	Tile-2	Octant

The balanced loading pattern was considered since it loads the minimal amount of data necessary (4 elements per thread), essentially allowing for a more granular split between memory latency and computation. The octant loading pattern is simpler and has fewer logically-divergent branches and would be expected to have better memory performance, since the data loading for the flowfield array and the residual array is theoretically perfect.

Figure 4.11 shows a time history of global loads and shared memory bank conflicts for each of the flux divergence kernel strategies listed in table 4.3. Overall, the octant loading pattern shows better predicted memory behaviour using all memory maps. The balanced loading pattern demonstrates significant inefficiency before the first flux calculation, at which point it should be possible to mask memory latency behind computation. Among the octant loading pattern predictions, the linear map performs worst overall, with improved global memory predicted performance by the $T_4T_4T_4$-O configuration and the best shared memory predicted performance for $T_2T_4T_2$-O.

The performance of each of these strategies was tested for problem 1, and results are presented in table 4.4, normalised to the worst-performing implementation. All values are taken to be the average of 1000 iterations. It is clear that $T_2T_4T_2$-O gives the best performance, and is thus the implementation used for the final optimised version. Although the MT_4T_4-O is predicted to give favourable performance, its overall performance is similar to that of $T_2T_4T_2$-O. This appears to be because of worse performance during data exchanges, which are not included in the simplified model.

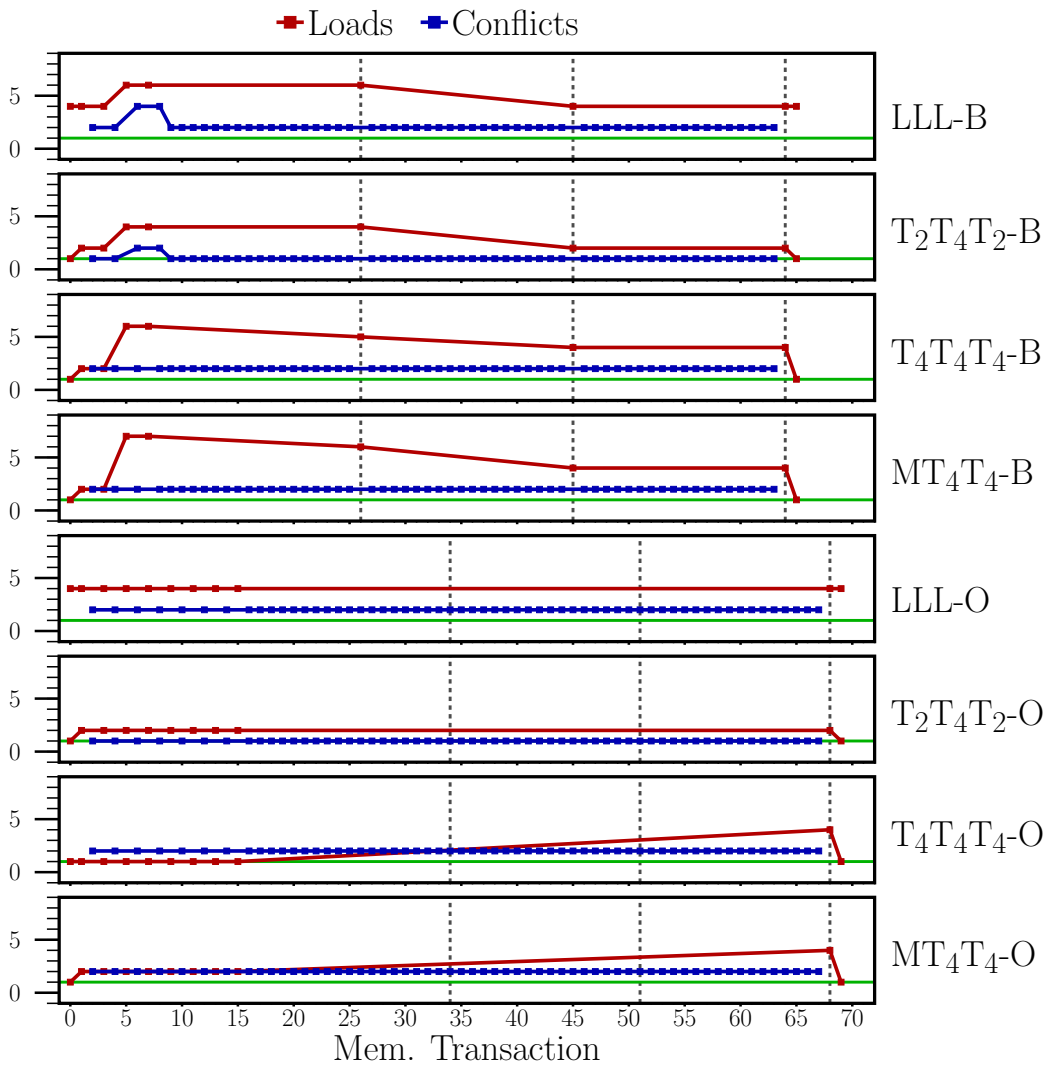


Figure 4.11: Per-transaction history of shared memory bank conflicts and global memory loads for the flux divergence kernel using balanced and octant loading patterns. The horizontal green line indicates perfect memory access under the present simplified model. Vertical dashed lines indicate the locations where fluxes are calculated.

4.9 Optimisation of Data Exchange

In order to compute fluxes at block boundaries, flowfield data from the neighbouring block must be loaded into a stencil. In a naïve implementation, this is usually done by over-allocating the number of cells in each block by half of the width of the flux stencil. As a result, there is additional storage required to allocate the flowfield values that are inside the exchange cells. While this cost is typically negligible for CPU, implementations (as there is a large pool of memory available), the amount of available global memory is generally much smaller on GPUs. For a block size of n^3, and using n_e layers of exchange cells, the memory storage efficiency ϕ (i.e. the ratio of volume

Table 4.4: Performance results for various loading pattern and memory map combinations. Absolute performance is reported in MUPS (millions of updates per second), normalised relative to the lowest-performing case.

Case ID	Improvement over baseline	MUPS
LLL-B (baseline)	0.0%	2674
$T_2T_4T_2$-B	19.0%	3182
$T_4T_4T_4$-B	5.0%	2807
MT_4T_4-B	−15.4%	2317
LLL-O	−0.5%	2661
$T_2T_4T_2$-O	40.7%	3760
$T_4T_4T_4$-O	10.8%	2961
MT_4T_4-O	23.5%	3302

cells to exchange cells) is given by

$$\phi = \frac{1}{\left(1 + 2\frac{n_e}{n}\right)^3}. \tag{4.6}$$

Assuming that $n_e = 2$, this is not an issue for large blocks of e.g. $n = 32$, which yields $\phi \approx 70\%$, but the efficiency drops to $\approx 13\%$ when $n = 4$. To enable the use of small blocks (which allow for more efficiently distributed mesh cells), a different strategy for data exchange must be employed. Furthermore, the data exchange kernels have high memory throughput with little computation, so there is a significant effect on overall performance.

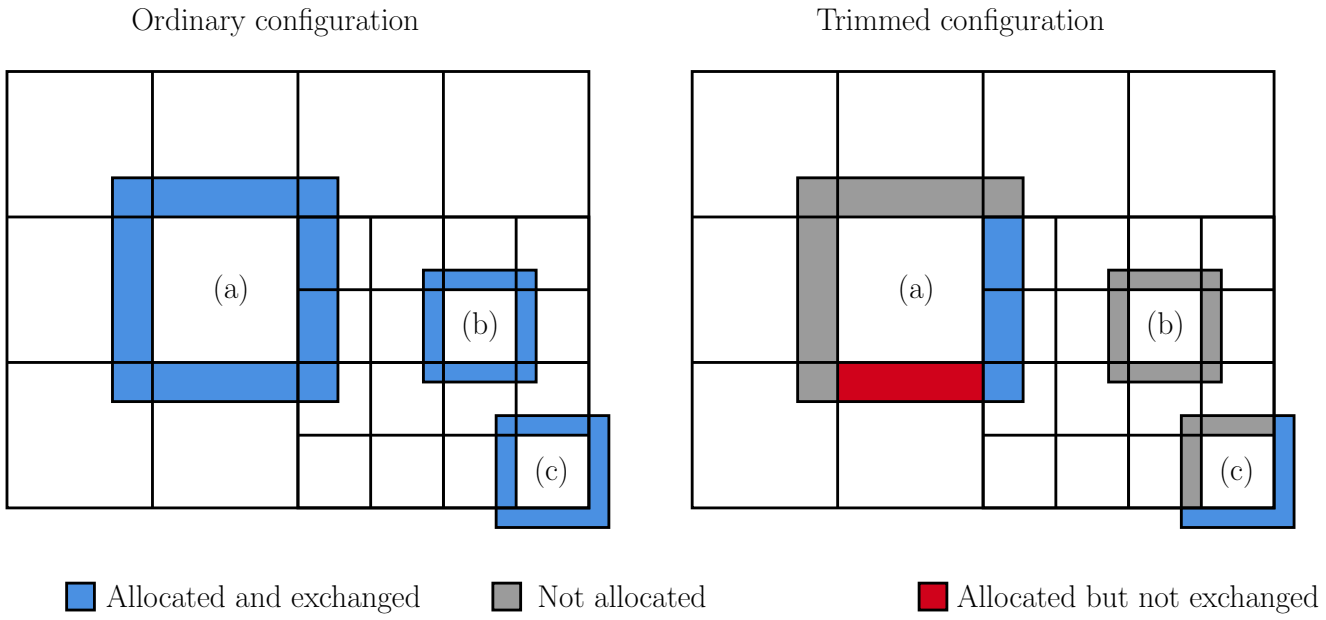


Figure 4.12: Comparison of ordinary/unoptimised (left) and trimmed/optimised (right) allocation of exchange cells for (a) a block with a coarse-fine interface, (b) within the volume mesh, and (c) at the domain boundaries.

Figure 4.12 shows the general strategy employed for handling allocation of exchange cells. For each block in the domain, exchange cells are allocated only if needed by proximity to a coarse-fine interface, a domain boundary, or a GPU partition boundary. This can result in some exchange cells that are allocated but not used for the calculation. This strategy requires a modification to the memory map implementation that allows for indices of exchange cells to be mapped to neighbouring blocks. That is, for the optimised implementation, $\eta(i, j + n, k)$ of a block b_0 maps to the same memory address as the index $\eta(i, j, k)$ for the neighbour block b_1 in the positive-j direction. Because this lookup is happening at the low-level offset calculation, this requires an additional few bytes of global memory to be loaded during the compute-heavy kernel, but balanced by the reduced cost from avoiding the full data exchange.

To further optimise the data exchanges, corner and edge data exchanges at CFIs are re-arranged to be direct injection exchanges from the faces allocated at the CFI. This is shown in figure 4.13. This results in a large reduction in the total number of coarse-to-fine and fine-to-coarse transactions, further optimising for smaller blocks.

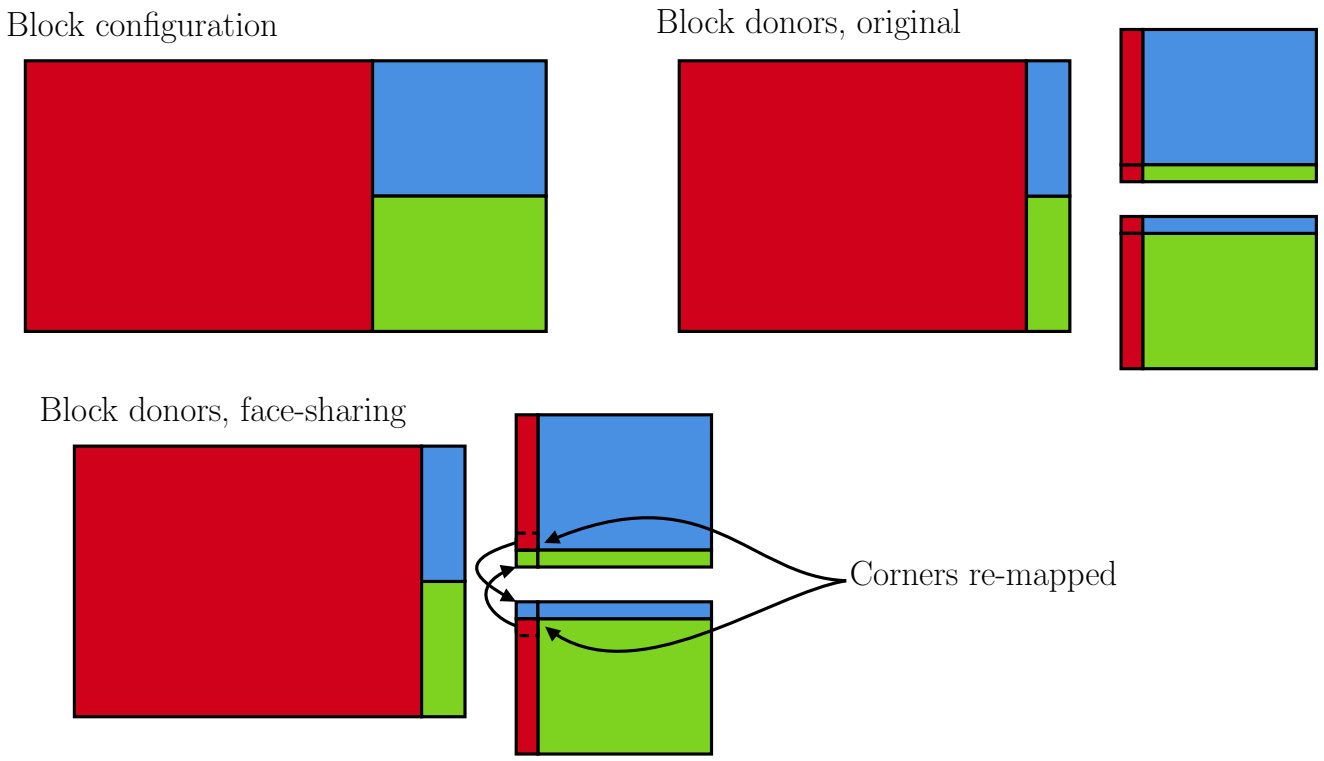


Figure 4.13: Illustration of the donor remapping, converting coarse-fine interface transactions into direct-injection transactions.

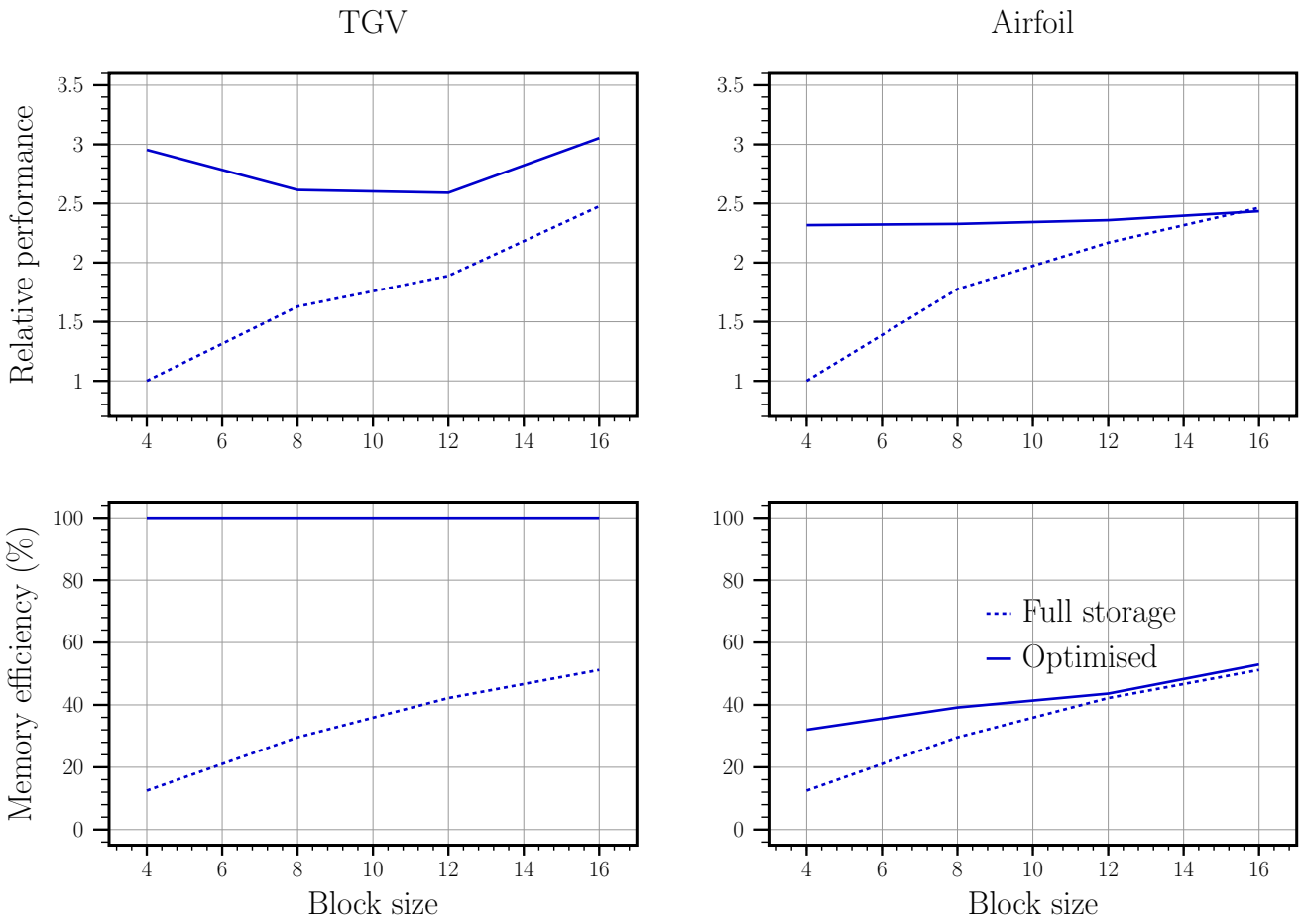


Figure 4.14: Memory efficiency and relative performance for test problems comparing naïve exchange mapping with the optimised memory map. Performance is taken as the total timestep time and normalised by the value taken with a block size of 4.

The memory storage efficiency and total performance as a function of block size for both the naïve memory map and the optimised one can be found in figure 4.14 for both test problems. Clearly, the optimised exchange mapping yields overall better performance and memory efficiency in both cases. The performance measure for the optimised implementation is relatively independent of block size when compared with the naïve one, with a noticeable loss for block sizes of 8 and 12 for the TGV case, likely due to worse performance in terms of the global memory loading having largely divergent memory addresses. The optimised exchange mapping also flattens the memory efficiency curve. Generally, there is still a negative performance impact of using smaller block sizes, but this is largely ameliorated by the use of the optimised exchanges.

4.10 Time Integration

In this section, explicit Runge-Kutta (RK) schemes are considered, and are expressed using the matrix

$$A = \begin{pmatrix} a_{0,0} & 0 & 0 & \dots & 0 \\ a_{1,0} & a_{1,1} & 0 & \dots & 0 \\ a_{2,0} & a_{2,1} & a_{2,2} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{N-1,0} & a_{N-1,1} & a_{N-1,2} & \dots & a_{N-1,N-1} \end{pmatrix} \in \mathbb{R}^{N \times N}, \tag{4.7}$$

where the final row denotes the residual accumulation step, and the remaining entries denote the RK coefficients. As an example, SSPRK3 scheme (4.2a) - (4.2d) is represented as

$$A = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{4} & \frac{1}{4} & 0 \\ \frac{1}{6} & \frac{1}{6} & \frac{2}{3} \end{pmatrix}. \tag{4.8}$$

Algorithm 3 gives a naïve implementation of the time integration. This implementation requires

Algorithm 3 Naïve implementation of explicit time integration scheme

Require: $w_0 = w^{(n)}$
Ensure: $w_1 = w^{(n+1)}$
1: $k_0 \leftarrow R(w_0)$
2: $c \leftarrow 1$
3: **while** $c \leq n_{\text{stage}}$ **do**
4: $\quad w_1 \leftarrow w_0 + \Delta t \sum_{i=0}^{c-1} a_{c-1,i} k_i$
5: $\quad$ **if** $c < n_{\text{stage}}$ **then**
6: $\quad\quad k_c \leftarrow R(w_1)$ ▷ Includes boundary treatment
7: $\quad$ **end if**
8: $\quad c \leftarrow c + 1$
9: **end while**

storage of the same number of copies of the residual array as there are stages in the explicit scheme, as well as two copies of the flowfield array. Furthermore, there are a significant number of kernels in use that do not balance memory and compute throughput well, notably the accumulation kernel in line 4, which requires loading a large amount of memory but doing very little computation on it.

The optimised implementation uses only a single copy of the residual array (k in algorithm 4) and two copies of the flowfield array. The implementation is detailed in algorithm 4. Note that the full residual calculation is separated into two components:

- $I(w)$, which computes, in the boundary cells (including the domain boundary), the difference between the residual computed with volume information and the residual computed with boundary information, and
- $D(w)$, which computes the ordinary residual.

The property $R(w) = I(w) + D(w)$ is maintained.

Algorithm 4 Optimised implementation of explicit time integration scheme

Require: $w_0 = w^{(n)}$
Ensure: $w_1 = w^{(n+1)}$
Ensure: $k = 0$
1: $c \leftarrow 0$
2: **while** $c \leq n_{\text{stage}}$ **do**
3: $\quad k \leftarrow k + I(w)$ ▷ Boundary treatment
4: $\quad w_1 \leftarrow w_1 + \Delta t k$
5: $\quad$ **for** $i \in \text{indices}(w)$ **do** ▷ Thread block
6: $\quad\quad w' \leftarrow w_0\big|_i$
7: $\quad\quad r \leftarrow D(w)\big|_i$
8: $\quad\quad r' \leftarrow k\big|_i$
9: $\quad\quad w' \rightarrow w' - r'$
10: $\quad\quad w' \rightarrow w' + a_c \Delta t (r + r')$
11: $\quad\quad k\big|_i \leftarrow z_c (r + r')$
12: $\quad\quad w_0\big|_i \leftarrow w'$
13: $\quad$ **end for**
14: $\quad c \leftarrow c + 1$
15: **end while**

The coefficients z_c are given as $\{1, 1/4, 0\}$ and a_c are given as $\{1, 1/4, 2/3\}$. Problem 1 was analysed with the naïve implementation and optimised implementation, the results of which can be found in table 4.5. Two versions of the naïve implementation are evaluated, one with separate update kernels, where the accumulation step (line 4 of algorithm 3) is implemented using allocation-free operator overloading, and another implementation where the accumulation step is a single kernel.

Table 4.5: Performance results for combined RHS and update kernels.

Case ID	Improvement over baseline	MUPS
Naïve, separate update kernels (baseline)	0.0%	755
Naïve, single update kernel	171.0%	2047
Optimised	309.0%	3087

The results in table 4.5 demonstrate the importance of ensuring that kernels perform all possible mathematical operations on available data. In the naïve implementation with separate kernels, memory is repeatedly loaded only to have almost no computational work done on it, and as a result, less than 10% of the execution time is spent calculating the most computation-heavy operations. The optimised kernel takes this one step further by fusing the update/accumulation and the flux divergence into a single kernel, along with all transformations from primitive to conservative variables.

4.11 Naïve/Optimised Performance Comparison

In this section, the naïve implementation and optimised implementation are compared as a whole. The two implementations are described in table 4.6. Note that neighbour and face sharing refer to the procedures in figures 4.12 and 4.13 respectively.

Table 4.6: Summary of differences between naïve and optimised implementation.

Implementation	Naïve	Optimised
Loop unrolling	not unrolled	unrolled
Stencil memory space	global memory	shared memory
Memory map	linear with exchange cells	flowfield: 2-tiling, neighbour/face sharing residual: 4-tiling
Loading pattern	independent threads	octant loading to shared memory
Time integration	separate kernels	single/fused kernel

As mentioned in section 4.2, all results in this section are taken from tests performed on a single NVIDIA RTX4090.

4.11.1 Problem 1 Results

The Taylor-Green Vortex problem was configured with a mesh of 384^3 cells with a total of 56M cells in total. The block size used is 16^3. There are no coarse-fine-interfaces or immersed-boundary

treatment, so this case represents a best-case scenario in terms of performance. Table 4.7 shows a summary of the performance results with a comparison between both implementations. Values were averaged over a duration of 1000 timesteps.

Table 4.7: Performance comparison for TGV test case.

Quantity	**Naïve**	**Optimised**
MUPS (see section 4.2)	163.6	3087
Timestep time (ms)	341.5	18.1
Speedup	1.0	18.9
Total required storage (GB)	6.1	3.2

The results in table 4.7 show a significant speedup from the optimisations in this chapter. Each of the kernels in the optimised implementation was profiled to compute the peak theoretical floating point operations per second (FLOPS) achieved over the full duration of the runtime. A weighted average was computed as

$$\overline{\phi} = \left(\sum_i \phi_i T_i \right) / \left(\sum_i T_i \right), \tag{4.9}$$

where ϕ_i is the FLOP rate of the i^{th} kernel and T_i is the duration of the i^{th} kernel. This quantity was evaluated to be $\overline{\phi} = 35.3\%$ for this test problem.

Figure 4.15 shows a roofline plot for the optimised kernel that computes solution updates and flux divergence simultaneously. All values were measured using NSight Compute. Because the solution is updated at the same time as the right-hand side is calculated, additional memory is loaded in this kernel, leading to an implementation that is marginally bandwidth bound with respect to the global DRAM bandwidth. To emphasise, this is a best-case scenario in terms of absolute performance, since there is no immersed boundary present in this case.

4.11.2 Problem 2 Results

The airfoil problem was configured with a grid of 60M cells in total. As with the previous section, a block size of 12^3 is used. The presence of the immersed boundary affects and the coarse-fine interfaces in the mesh has a significant impact on the performance, as shown in table 4.8. This is a more realistic estimate of the performance of the optimised solver under real conditions. Once

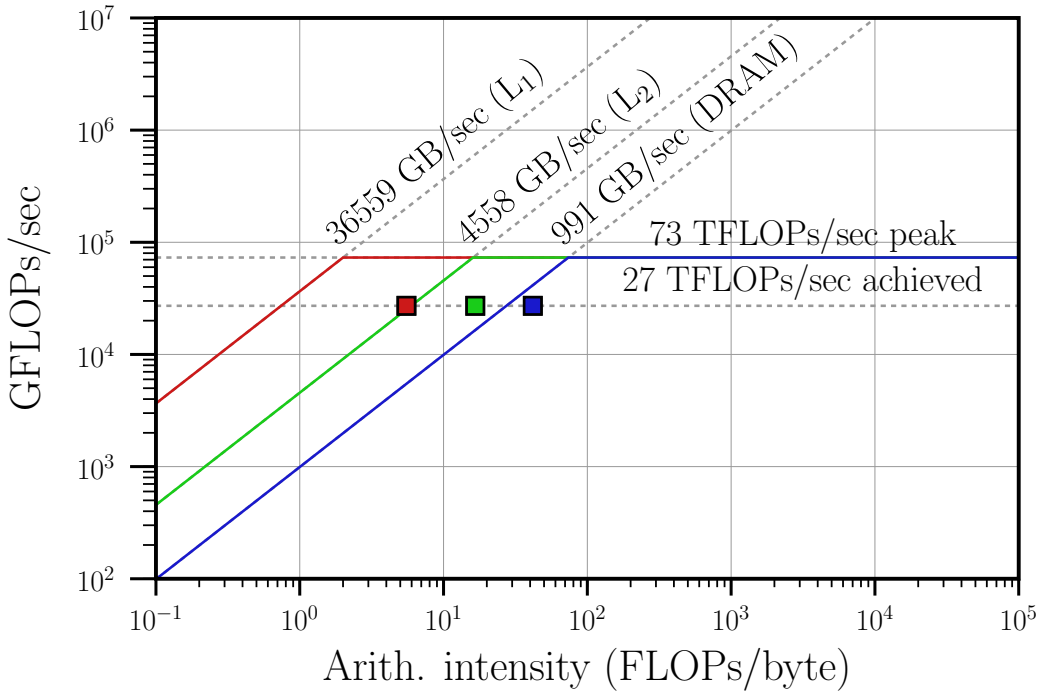


Figure 4.15: Hierarchical roofline plot for final optimised update flux divergence / solution update combined kernel.

again, values were averaged over a duration of 1000 timesteps. Total required storage includes information pertaining to the immersed boundary, but does not include the GPU runtime library overhead.

Table 4.8: Performance comparison for airfoil test case.

Quantity	**Naïve**	**Optimised**
MUPS (see section 4.2)	124.3	1658
Timestep time (ms)	449.5	36.9
Speedup	1.0	13.3
Total required storage (GB)	7.9	4.0

4.12 Remarks

To better contextualise the performance of the new GPU-first solver, a comparison is made between the original CPU implementation and the current GPU implementation for a full simulation consisting of 500,000 timesteps on a 60-million-cell domain.

The original CPU implementation required approximately 0.45 seconds per timestep when run across 340 CPU cores. This results in a total wall-clock time of 62.5 hours, or 21,250 core-hours.

At a typical cloud compute rate of \$0.0425 per core-hour, the total estimated cost of running the simulation on AWS is approximately \$903.

By contrast, the GPU implementation using a single NVIDIA RTX 4090 completes each timestep in 0.04 seconds, resulting in a total wall-clock time of 5.56 hours. The RTX 4090 has a nominal power draw of 450 W under full load. Assuming an electricity cost of \$0.34 per kWh (typical for London), the total energy cost is approximately 2.5 kWh $\times$ \$0.34 = \$0.85.

Table 4.9: Estimated Cost Comparison for 500,000 Timesteps

Metric	CPU (340 cores)	GPU (RTX 4090)
Timestep time	0.45 s	0.04 s
Wall-clock time	62.5 h	5.56 h
Total core-hours	21,250	–
Estimated cost (USD)	\$903	\$0.85

The cost and performance estimates assume ideal hardware utilisation. For the CPU run, full use of all 340 cores is assumed, which may overstate efficiency due to possible load imbalance. GPU estimates are based on consistent high utilisation of the RTX 4090 and do not account for full system power draw or thermal effects. Energy costs reflect only GPU consumption and exclude supporting components. These figures serve as comparative bounds rather than precise operational costs.

This analysis, although a rough estimate, shows that while the GPU simulation takes a fraction of the time and energy, it also incurs orders of magnitude lower cost. The analysis assumes that the user of the GPU implementation does not rent GPU time, but owns it. The GPU-first approach offers a highly cost-effective solution for large-scale CFD simulations.

Chapter 5: Validation

In this chapter, the IBM-WMLES framework developed in chapters 2 through 4 is tested for a series of problems involving turbulent flow at various Mach numbers. It should be noted that the present framework has also been validated for low Mach numbers in a series of other works to which the author contributed substantially, particularly refs [127, 128, 129]. However, as these are out of scope for this thesis, they are not discussed in this chapter. Note that many of the remarks on the results in this chapter are adapted from ref. [5].

5.1 Turbulent Channel Flow at $Re_\tau = 5200$

For basic validation, an incompressible turbulent channel flow was simulated using the current IBM-WMLES approach at $Re_\tau = 5200$ and compared against the DNS data by Lee & Moser [130]. This simulation was conducted in a quasi-compressible regime with a Mach number of $M = 0.2$. The bulk Reynolds number was specified as $Re_b = U_b\delta/\nu = 125,000$, where U_b represents the volume-averaged mean velocity in the streamwise direction, δ is the half-width of the channel, and ν is the kinematic viscosity. Initial conditions for this simulation (denoted by the subscript IC) are provided in table 5.1. The streamwise velocity was initialised with a laminar profile with a fluctuating velocity component (corresponding to the most unstable mode obtained via linear stability theory (LST) [131]) superimposed. For this and all following simulations in this work, the WALE model is employed [70]

Table 5.1: Initial conditions for the quasi-compressible turbulent channel flow simulation.

P_{IC}	T_{IC}	u_{IC}
101.327 kPa	300 K	$1.5U_b(1-y^2) + 0.2U_b a(x,y,z)$

In table 5.1, $a(x, y, z)$ is of the form

$$a(x,y,z) = \cos(\pi z)\left\{\sin(2\pi x)\sin(4\pi y) + \sin(4\pi x)\sin(8\pi y)\right\}, \tag{5.1}$$

which represents disturbances superimposed on the laminar mean flow. The freestream conditions

for this case are listed in table 5.2.

Table 5.2: Freestream flow conditions for the turbulent channel flow.

P_∞	T_∞	$u_\infty = U_b$	M_∞	Re_b
101.327 kPa	300 K	69.4377	0.2	125,000

The computational domain was discretized using a Cartesian grid with two levels of refinement, with the finer grid level positioned closer to the channel walls. The grid size and mesh spacing for the finest level of refinement are provided in table 5.3. Uniform discretization was applied across the domain to ensure consistent mesh spacing in both the streamwise and spanwise directions. The sampling point for data exchange between the wall model and the LES domain is set at a height of approximately 0.05δ, with some variations applied to study the impact of this location.

Table 5.3: Mesh details for the finest grid level used in the turbulent channel flow case.

Grid level	Total Grid Size	$\Delta x \times \Delta y \times \Delta z$	$\Delta x^+ \times \Delta y^+ \times \Delta z^+$	$L_x \times L_y \times L_z$
Coarse	1.9M	$\delta/20 \times \delta/80 \times \delta/20$	$270 \times 68 \times 270$	$12.8\delta \times 2\delta \times 4.8\delta$
Fine	8.9M	$\delta/30 \times \delta/128 \times 3\delta/100$	$173.3 \times 40.6 \times 156$	$12.8\delta \times 2\delta \times 4.8\delta$

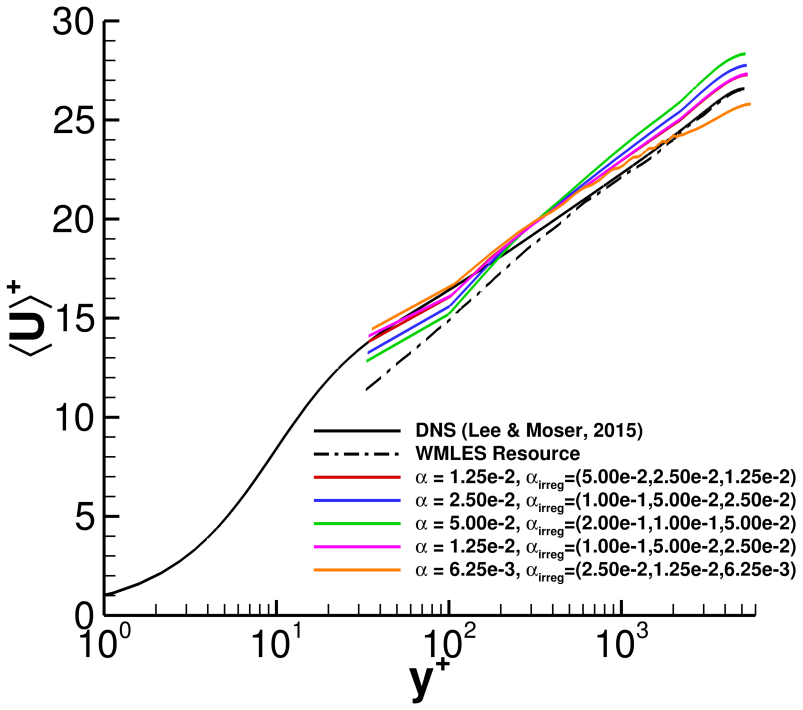
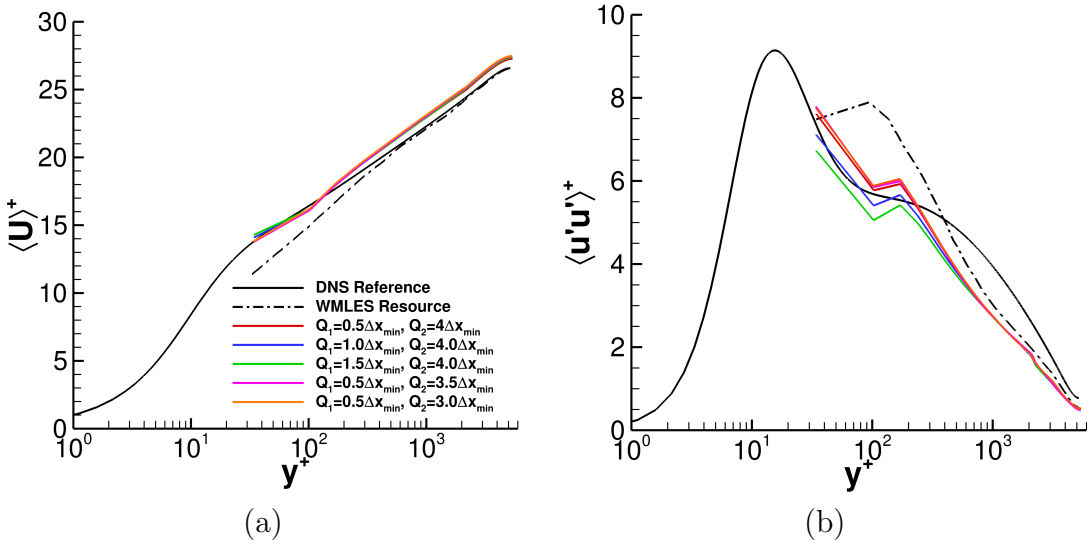


Figure 5.1: Mean velocity profiles for the turbulent channel for various levels of dissipation. Figure recreated from [5].

A series of numerical experiments were conducted to understand how the dissipation coefficients α (applied in the domain's interior) and α_{irreg} (applied within irregular cells) influence the results. Additional tests were performed by adjusting the positions of the interpolation points Q_1 (near) and Q_2 (far) as outlined in section 2.7. The simulations began with an initial interior dissipation coefficient of $\alpha = 0.00625$ and irregular dissipation coefficients of $\alpha_{\text{irreg}} = (0.025, 0.0125, 0.00625)$ (with the first value corresponding to the first irregular cell). The values of α and α_{irreg} were increased incrementally over a set of four simulations.

It was observed that when the dissipation coefficient α is kept low, specifically $\alpha < 0.0125$, the solution exhibits a significant amount of noise. Conversely, higher values of α and α_{irreg} cause the solution to deviate from the expected results. To reiterate, α_{irreg} is manually increased near the boundary for the purpose of stabilisation. Increasing α accelerates the flow toward the centreline while slowing it down near the wall, as depicted in figure 5.1. The best results were obtained with $\alpha = 0.0125$ and $\alpha_{\text{irreg}} = (0.05, 0.025, 0.0125)$. In another simulation, doubling the values of α_{irreg} in all cells showed no significant difference compared to the previous optimal case, as illustrated in figure 5.1. These findings suggest that dissipation in the interior domain has a more substantial impact on the results than dissipation in irregular cells. However, since numerical dissipation behaves differently in low-speed and high-speed flows (see, for example, [132]), these observations may not directly apply to high-speed flows, which are discussed in the subsequent sections.

The influence of both the location of the data exchange point between the wall model and the LES domain and the position of the second interpolation point used to impose boundary conditions at the immersed boundary were investigated. All cases were run using the optimal values for α and α_{irreg} determined earlier, specifically $\alpha = 0.0125$ and $\alpha_{\text{irreg}} = (0.05, 0.025, 0.0125)$. Adjusting either Q_1 or Q_2 did not significantly affect the mean velocity profiles or the Reynolds stress tensor components, a result consistent with the well-known approach of Kawai & Larsson [93]. The variations involved placing the data exchange point at $3\Delta x_{\text{min}}$, $3.5\Delta x_{\text{min}}$, and $4\Delta x_{\text{min}}$ from the wall, where Δx_{min} represents the minimum grid spacing in the wall-normal direction. Similarly, the location of the closer interpolation point Q_1 was varied between $0.5\Delta x_{\text{min}}$ and $1.5\Delta x_{\text{min}}$. The negligible differences in turbulent statistics, such as mean streamwise velocity



(a) (b)

Figure 5.2: Effect of the locations of the sampling point Q_2 between the wall-model and the immersed boundary and the interpolation point Q_1 on the turbulent statistics: (a) mean streamwise velocity and (b) streamwise normal component of the Reynolds stress tensor, $\langle u'u' \rangle$. Figure recreated from [5].

and the streamwise Reynolds stress component, demonstrate the relative insensitivity of these quantities to the positions of Q_1 and Q_2. These minor variations are illustrated in figures 5.2(a) and 5.2(b). In general, changing the location of Q_1 has a slightly greater impact on the Reynolds stresses near the wall, but these effects are limited to the Cartesian grid below the exchange point. As the centreline of the channel is approached, all profiles converge well.

Figures 5.3(a) and 5.3(b) present a grid convergence study using the optimal configuration of convective operators identified earlier. The grid used in previous tests was refined in all directions, as detailed in table 5.3. As shown in Figure 5.3(a), refining the grid significantly reduces the deviation of the WMLES results from the DNS solution [130] in the log region, indicating grid convergence. Figure 5.3(b) displays the streamwise-normal Reynolds stress ($\langle u'u' \rangle$) and Reynolds shear stress ($\langle u'v' \rangle$). While the WMLES results agree well with previous WMLES calculations, there is poor agreement with DNS results in the inner regions due to grid under-resolution, even after refinement. This agreement improves toward the channel centre but still shows notable discrepancies, especially for the $\langle u'u' \rangle$ component. This observation is consistent with literature. The $\langle u'u' \rangle$ component does not visibly decay to zero near the wall. This is likely because there is some noise present at the wall that is artificially increasing the fluctuation level. Values at the

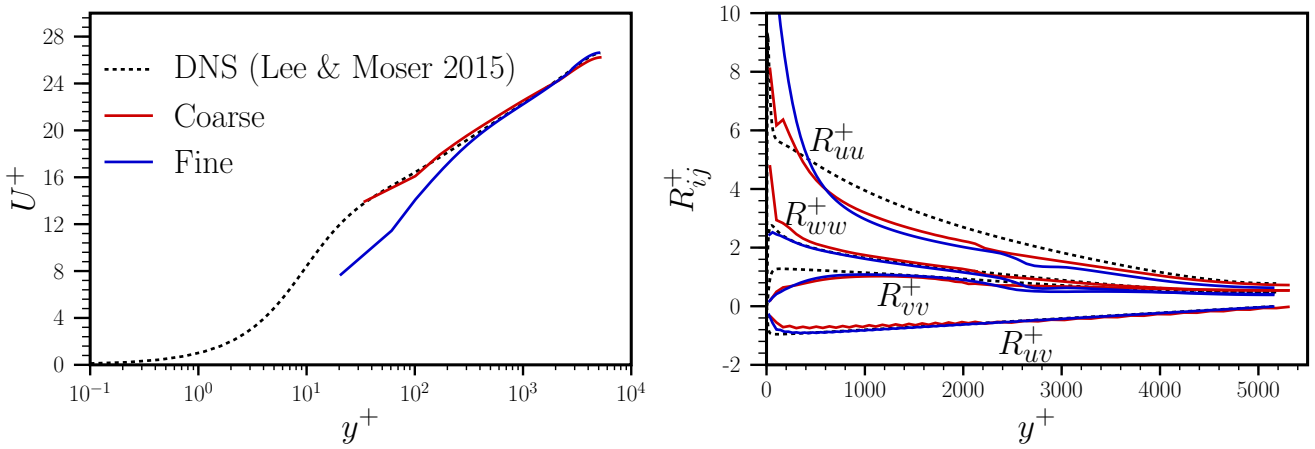


Figure 5.3: Left: inner-scaled mean profiles for average streamwise velocity. Right: components of the Reynolds stresses for the turbulent channel simulation.

wall are not visible since the value nearest the wall is taken from the first cell centre.

5.2 ONERA Wing at Transonic Conditions

The simulation presented here is used to validate the IBM-WMLES for a more complex, three-dimensional flow scenario. This particular flow includes the presence of shock waves, providing an initial evaluation of the shock-capturing capabilities of the current scheme. The geometry is the well-known ONERA wing test case under transonic flow conditions [133] and is visualised in figure 5.4. The flow conditions shown here are listed in table 5.4, and the computational grid parameters can be found in table 5.5

Table 5.4: Freestream flow conditions for the ONERA wing flow.

$M_\infty = 0.84$	$\mathrm{Re}_{c,\mathrm{root}}$	α
0.84	14.6×10^6	0

Table 5.5: Mesh parameters for the ONERA wing flow.

$\Delta x/\Delta y/\Delta z$	$\Delta x^+/\Delta y^+/\Delta z^+$	Grid size
$0.58 \times 10^{-3} c_{\mathrm{root}}$ / $0.58 \times 10^{-3} c_{\mathrm{root}}$ / $0.75 \times 10^{-3} c_{\mathrm{root}}$	105 / 105 / 135	223M

Figure 5.5 shows slices of surface pressure coefficient at each of the 7 slices indicated on the visualisation of the wing surface. Reference data is from the NASA turbulence modelling resource website. The results show good overall agreement with reference surface pressure data, although

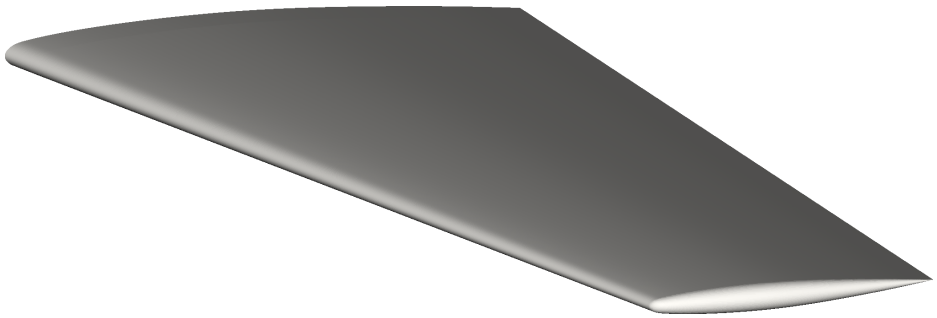


Figure 5.4: ONERA wing geometry.

notably position of the shock is accurately captured across all slices except for slice 3, where a slight mismatch occurs at $x/c = 0.3$.

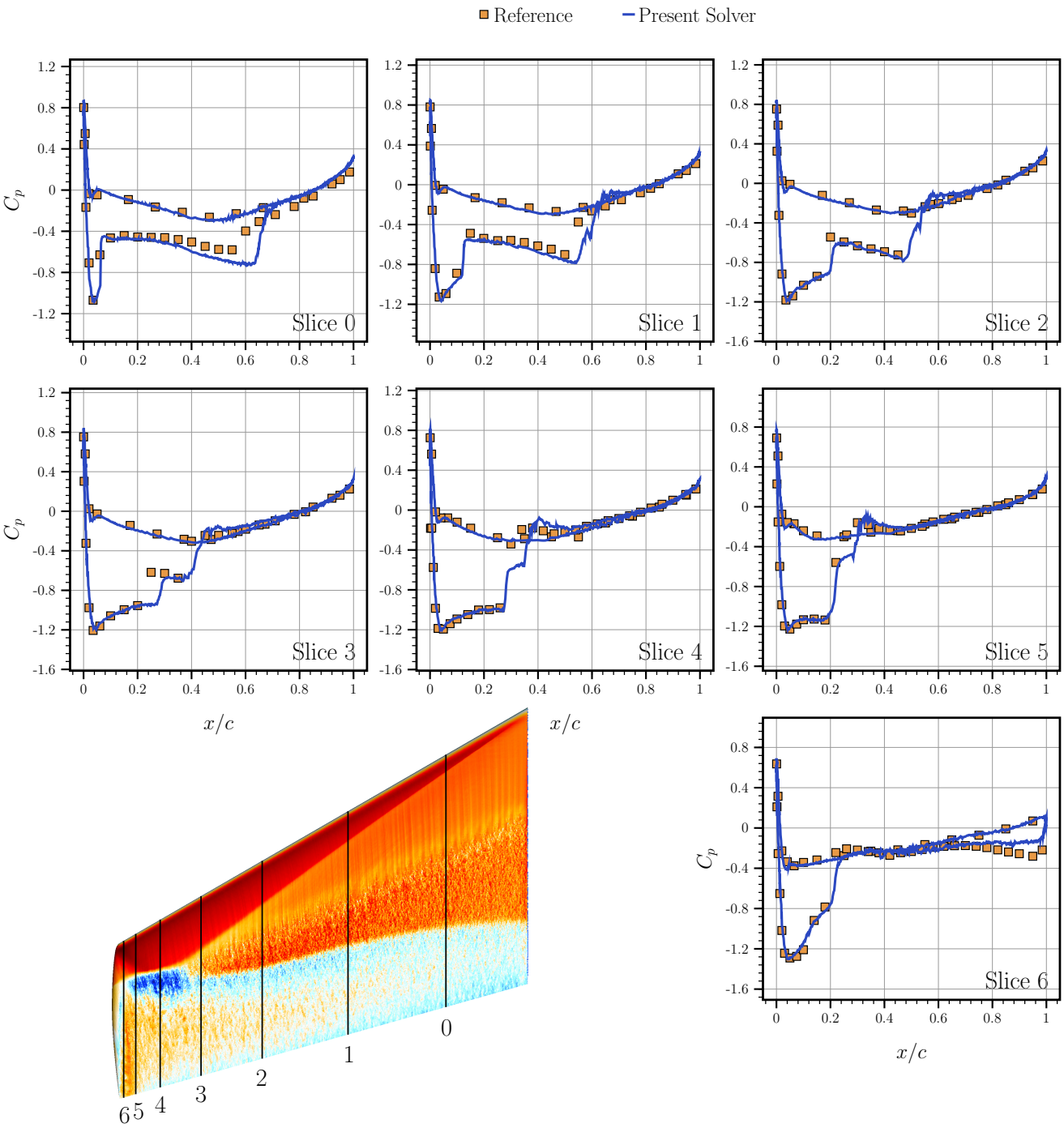


Figure 5.5: Slices of surface pressure coefficient at various spanwise slices with comparison against reference data from the NASA turbulence modelling resource. Bottom-left shows velocity input to wall model and includes location of spanwise slices.

5.3 Hypersonic Transitional Boundary Layer

A Mach 6 hypersonic transitional boundary layer is used as an initial test for the IBM-WMLES method in high-speed flows. This case also evaluates the method's performance for flows with an

extensive transitional region and examines the sensitivity of the transition location to different numerical parameters. It is important to note that the objective with this case is not to predict the boundary layer's transition location; instead, the transition sensor is calibrated to align with the reference transition location. The transition sensor applied in this study follows the approach in [94], where the dimensionless turbulent kinetic energy is calculated as:

$$Tr = \frac{\hat{\rho}\hat{k}}{\hat{\mu}|\hat{S}|}, \quad \text{with} \quad k = \widehat{u_i u_i} - \hat{u}_i \hat{u}_i, \tag{5.2}$$

The eddy viscosity in the wall model is neglected when the sensor value Tr is below a certain threshold $Tr_0 = 1.4$. A filtered quantity $\hat{\phi}$ is computed as

$$\hat{\phi}_{n+1} = \left(1 - \frac{\Delta t}{T}\right)\hat{\phi}_n + \frac{\Delta t}{T}\phi_n, \tag{5.3}$$

where $T = 1/\sqrt{S_{ij}S_{ij}}$.

The flow field examined involves a hypersonic transitional boundary layer at Mach 6 and a Reynolds number of 22 million. DNS data for this case is provided by Subbareddy and Candler [134], where transition was induced by introducing counter-rotating vortices at the inflow. Mettu and Subbareddy [94] conducted WMLES of this scenario using a body-fitted framework, applying inflow perturbations to the density field as described in (5.4) with $A = 0.001$. These perturbations, initially used by Sandham et al. [6], are given by:

$$\rho' = A\big(1 - e^{-(y/\delta_0^*)^3}\big)\rho_\infty \sum_{j=0}^{16} \cos\left(\frac{2\pi j z}{L_z} + \phi_j\right) \sum_{k=1}^{20} \sin(2\pi k f_0 t + \psi_k) = A\rho_\infty D(y)\Phi(z)\Psi(t). \tag{5.4}$$

For this case, $f_0 = 166$ kHz, ϕ_j, ψ_k are randomly generated phases, and the parameter δ_0^* is chosen to be $2.205 \times 10^{-4} \times L$. Instantaneous skin friction and Stanton number results are shown in figure 5.7. The expressions used to compute skin friction and Stanton number are given by

$$C_f = \frac{\tau_w}{\rho_\infty u_\infty^2/2} \quad \text{and} \quad St = \frac{-q_w}{\rho_\infty u_\infty C_p (T_w - T_r)}, \tag{5.5}$$

where τ_w is the wall shear stress, ρ_∞ is the reference density, u_∞ is the reference velocity, $C_p =$

$\gamma R/\gamma - 1$ is the specific heat, T_w is the wall temperature, and T_r is the recovery temperature, where $r = 0.89$ and

$$\frac{T_r}{T_\infty} = 1 + r\frac{\gamma - 1}{2}M^2. \tag{5.6}$$

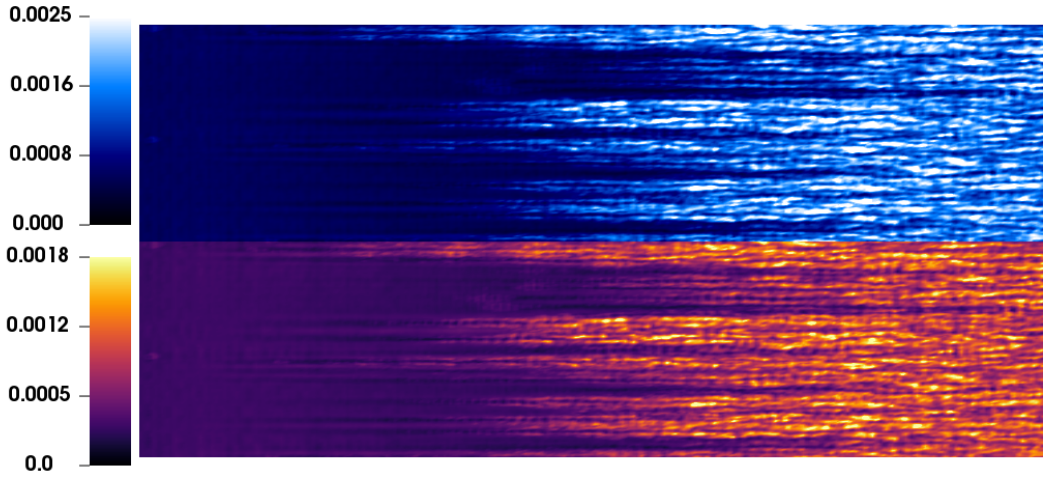


Figure 5.6: Instantaneous surface snapshots of C_f (top) and St (bottom) near the transitional region of the Mach-6 hypersonic transitional boundary layer. Figure recreated from [5].

Table 5.6: Flow conditions for the Mach-6 hypersonic transitional boundary layer

M_∞	Re_1	u_∞	P_∞	T_∞	T_w
6.0	2.2×10^7	1916.7 m/s	13491 Pa	254 K	300 K

Table 5.7 provides details on the near-wall mesh for this case. Note that the + units are calculated at the location of maximal C_f. The mesh is coarsened as it moves away from the wall, with the grid size doubling three times towards the free-stream. The entire boundary layer is resolved with consistent grid resolution.

This test case was configured for five configurations of the near-wall dissipation operator to evaluate its effect on solution quality. Each configuration involves three parameters: $\alpha_{\text{irreg,n}}$ for dissipation in the first three irregular cells in the wall-normal direction, $\alpha_{\text{irreg,t}}$ for tangential

Table 5.7: Grid details for the Mach-6 hypersonic transitional boundary layer flow.

Total Grid Size	$\Delta x/\delta_0 \times \Delta y/\delta_0 \times \Delta z/\delta_0$	$\Delta x^+ \times \Delta y^+ \times \Delta z^+$
$32M$ (coarse)	$1.14 \times 0.20 \times 0.55$	$156 \times 27 \times 75$
$45M$ (fine)	$0.9 \times 0.16 \times 0.45$	$123 \times 22 \times 61$

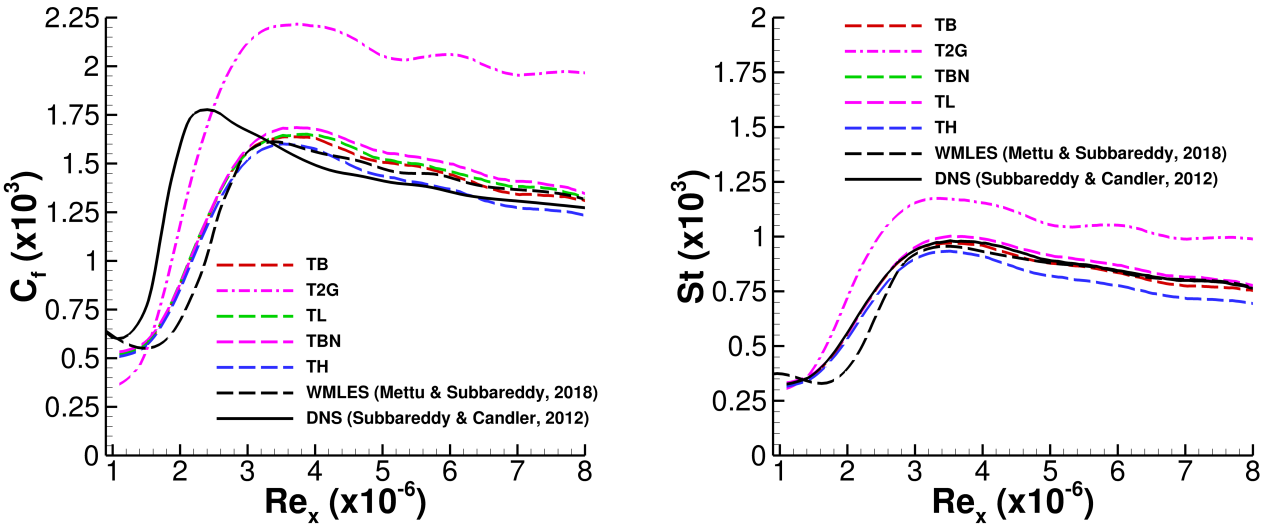


Figure 5.7: Skin friction (left) and Stanton number (right) along the surface of the Mach-6 hypersonic transitional boundary layer. Figure recreated from [5].

direction dissipation, and N_{ghost} for the number of ghost cells used in upwind flux calculations.

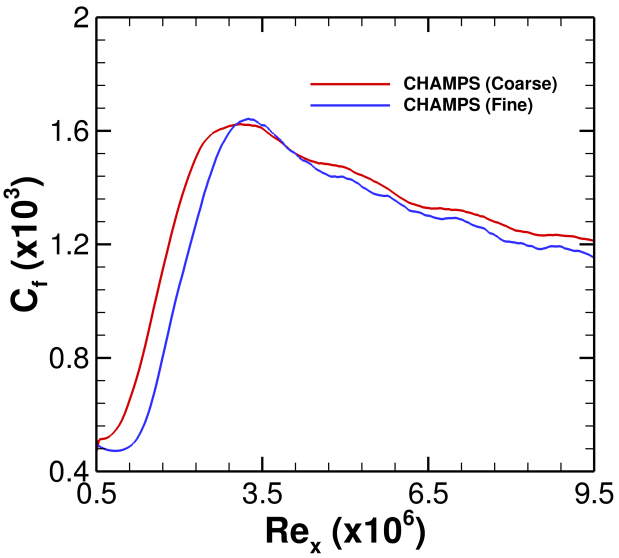


Figure 5.8: C_f for Mach-6 transitional flat plate evaluated at coarse and fine grid levels. Figure recreated from [5].

When $N_{\text{ghost}} = 2$, the upwind flux reconstruction in irregular cells is identical to the interior scheme. The first values of $\alpha_{\text{irreg,t}}$ and $\alpha_{\text{irreg,n}}$ refer to the cells closest to the wall. Case TB serves as a baseline, approximating the dissipation operator D_3 from section 2.10, applying dissipation only in the first three cells. Cases TL and TH use lower and higher dissipation coefficients, respectively,

Table 5.8: Numerical simulation parameters for the Mach-6 hypersonic transitional boundary layer.

Case Label	$\alpha_{\text{irreg,t}}$	$\alpha_{\text{irreg,n}}$	N_{ghost}
TB	$(0.4, 0.2, 0.1)$	$(0.4, 0.2, 0.1)$	1
T2G	$(0.4, 0.2, 0.1)$	$(0.4, 0.2, 0.1)$	2
TBN	$(0.4, 0.2, 0.1)$	$(0.1, 0.05, 0.025)$	1
TL	$(0.3, 0.15, 0.1)$	$(0.3, 0.15, 0.1)$	1
TH	$(0.5, 0.3, 0.1)$	$(0.5, 0.3, 0.1)$	1

to illustrate their impact. Case TBN examines the effect of varying normal dissipation coefficients, while T2G assesses the impact of adding a second ghost cell. All variations used a dissipation factor of $\alpha = 0.02$ in the interior.

Figure 5.6 shows instantaneous snapshots of skin friction and heat transfer data from case TB (see table 5.8). Transitional streaks, indicative of 2^{nd}-mode transition, are visible extending streamwise from approximately $Re_x \approx 2 \times 10^6$, consistent with observations in [134, 94]. Figure 5.7 shows the spanwise-averaged skin friction coefficient and Stanton number along the boundary layer. Results from case T2G indicate that using two ghost cell layers does not provide accurate results for the current scheme. The second layer of ghost cells was filled as described in section 2.6, but this approach may introduce significant errors due to the extrapolation over a larger distance than the sampling points span. An improved filling procedure might make this scheme viable, but all further simulations in this chapter use only a single ghost cell.

The baseline case TB shows reasonable agreement with [94], though heat transfer predictions are slightly higher than the DNS reference. Cases TB, TL, and TBN exhibit similar results, suggesting that the scheme is relatively insensitive to the distribution of tangential versus normal dissipation, although excessive dissipation, as seen in case TH, can degrade turbulent structures. Laminar surface quantities may not be accurately predicted by the wall model due to violations of wall modelling assumptions in this region.

A second grid resolution was used with configuration TB to assess grid convergence. Figure 5.8 illustrates the skin friction for the coarse and fine meshes. The peak skin friction changes less with grid refinement compared to variations due to different boundary dissipation operator configurations. The increased grid resolution caused a delay in transition location, showing that

the transition location is still sensitive to the mesh resolution.

5.4 Transitional Shock Wave Boundary Layer Interaction (SWBLI)

This test case replicates the conditions from [7, 94] for a transitional shock-wave boundary-layer interaction. It evaluates the performance of the IBM-WMLES under conditions that may not align with the wall-model assumptions. The case features a Mach 6 freestream at a Reynolds number of 6 million. Table 5.9 provides the test case parameters. The domain measures $362\delta_0 \times 26\delta_0 \times 47\delta_0$, with δ_0 being the laminar boundary layer thickness at the inflow. A 15 δ_0 outflow buffer is included. The incoming shock is specified using the Rankine-Hugoniot relations at the top boundary, with shock impingement occurring $158\delta_0$ from the inflow plate. The turn angle is 4°, leading to a shock angle of approximately 12.4°.

Table 5.9: Flow conditions for the Mach-6 transitional SWBLI

M_∞	Re_1	u_∞	P_∞	T_∞	T_w	δ_0/L
6.0	6.0×10^6	969.6 m/s	498.72 Pa	65 K	300 K	0.0011

Two variations of this test case were examined with different configurations. The first was conducted on a flat plate geometry using two grids: a coarse grid with approximately 2.1 million cells and a fine grid with about 4.6 million cells. This setup, with minimal immersed boundary effects, allows for a close comparison with reference data. The second variation involved rotating the geometry, introducing grid misalignment, to study the impact (and invariance) of the immersed boundary treatment at different misalignment levels. These cases will be detailed separately in the following sections.

5.4.1 Mach 6 SWBLI – Aligned Case

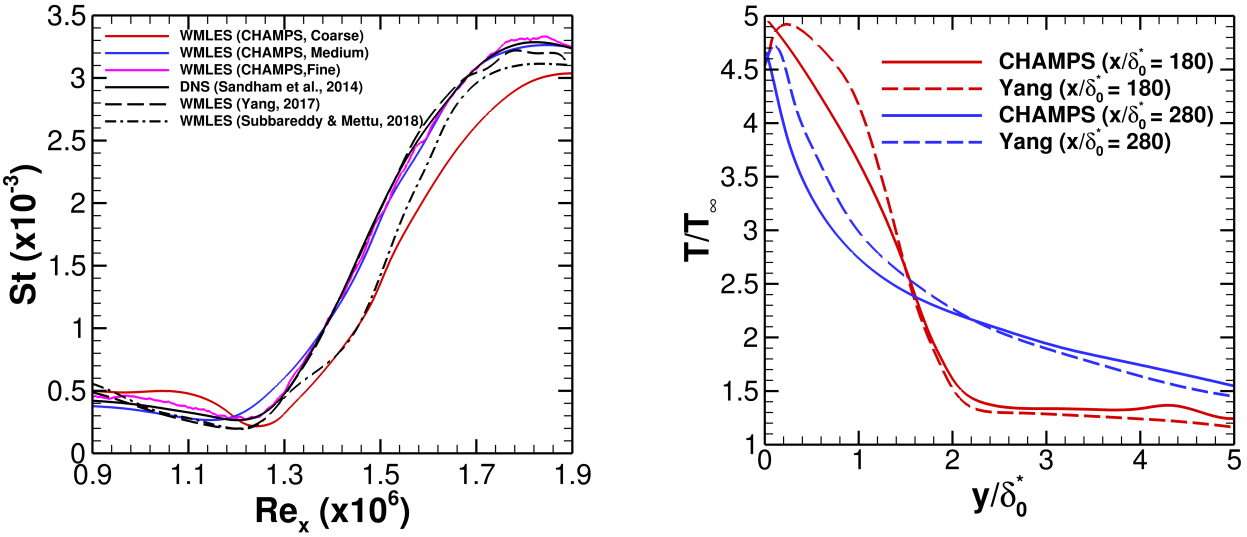
This test case poses several challenges for the IBM-WMLES framework. The thin laminar boundary layer upstream is perturbed by density fluctuations as described in (5.4). The boundary layer is significantly under-resolved, with approximately 4 grid cells covering its entire thickness at the inflow, making it difficult to accurately capture disturbance growth. Upstream of the shock incidence point, flow separation occurs, creating a separation bubble roughly $50\delta_0$ thick. Transition

happens shortly downstream of this bubble (see [135] for details). Table 5.10 provides details on the near-wall grid for this configuration. The grid topology is not kept constant, but the grid spacing is uniform throughout the boundary layer in each simulation.

Table 5.10: Grid details for the Mach-6 SWBLI case.

Total Grid Size	$\Delta x^+ \times \Delta y^+ \times \Delta z^+$
$2.1M$ (coarse)	$295 \times 65 \times 171$
$4.6M$ (medium)	$148 \times 50 \times 85$
$16M$ (fine)	$43 \times 35 \times 90$

Two reference quantities were evaluated in this case: surface heat transfer and mean temperature profiles. Surface heat transfer was compared with data from Mettu [94], Yang [7], and Sandham [6], while mean temperature profiles were compared with Yang [7] at $x = 180\delta_0$. Figure 5.9 shows that the fine mesh provides a good match to the DNS data, though it slightly over-predicts heat transfer near the end of the separation bubble. The coarse mesh exhibits less favourable agreement overall, but aligns well with other WMLES results from the centre of the separation bubble to the transitional region.



(a) (b)

Figure 5.9: (a) Stanton number and (b) temperature profile results for the Mach 6 SWBLI case. Figure recreated from [5].

Figure 5.9 shows that temperature profiles are reasonably consistent with those from [7]. However, within the wall-model sampling layer (approximately $y < 1.5\delta_0$, where δ_0 is the in-

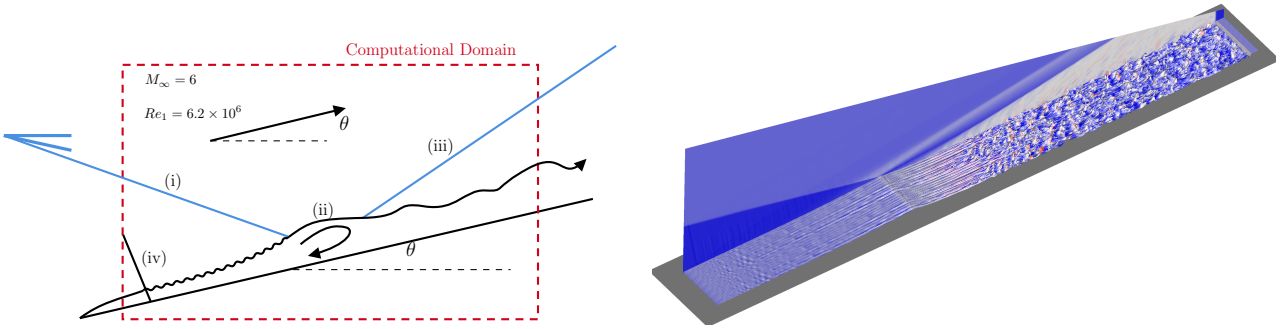
flow boundary layer thickness), errors arise from both the numerical treatment at the immersed boundary and the low resolution of the near-wall grid. For reference, the grid resolution in [7] is $\Delta x^+ \times \Delta y^+ \times \Delta z^+ = 40 \times 16 \times 23$.

5.4.2 Mach 6 SWBLI – Inclined Case

To evaluate the properties of the current IBM, the test case was adjusted to be rotationally invariant, introducing an angle of inclination as shown in figure 5.10(i). The problem setup was rotated accordingly. This modification required expanding the domain height by 50% compared to the previous case, resulting in a grid with approximately 10 million points. The cell aspect ratio was kept at approximately 1.0 to avoid issues related to large aspect ratios and misalignment. No grid optimisation was performed; the focus was solely on how misalignment affects the solution. Grid resolution details are provided in table 5.11.

Table 5.11: Grid details for the mesh inclined SWBLI case. The reported grid spacing applies to the blocks at the finest level of the grid.

δ_0/L	$\Delta x^+ \times \Delta y^+ \times \Delta z^+$	$L_x \times L_y \times L_z$
0.0011	$41 \times 40 \times 120$	$381\delta_0 \times 91\delta_0 \times 47\delta_0$



(a) (b)

Figure 5.10: (a) Schematic diagram for the inclined SWBLI case showing the: (i) incoming shock at the inflow, imposed using the Rankine-Hugoniot shock relations, (ii) shock-induced separation bubble, (iii) reflected shock, and (iv) density disturbance plane location, located $L/100$ from the inflow. (b) Instantaneous flow visualisation for the inclined SWBLI case at a 12° angle of inclination showing isosurfaces of temperature at $T = 200$K coloured by the streamwise component of velocity with a slice of instantaneous pressure on the rear plane. Figure recreated from [5].

Five angles are examined, as listed in table 5.12. The largest angle aligns the incoming shock wave with the grid. Due to the misalignment between the inflow plane and the geometry, applying

the density perturbations from (5.4) consistently required modifying the shape function. Thus, density perturbations are applied as a forcing term in the mass conservation equation:

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_i}{\partial x_i} = A\rho_\infty G(\hat{x})D(\hat{y})\Phi(z)\Psi'(t), \tag{5.7}$$

where $\hat{x}$ and $\hat{y}$ are rotated coordinates, and $G(x) = U_\infty \exp(-((x - x_0)/\Delta x)^2)/(\Delta x)$ is an approximate delta-function to localise fluctuations near the inflow. Freestream conditions are detailed in table 5.13. Note that $D(\hat{y})$, $\Phi(z)$, and $\Psi(t)$ are as in (5.4). The grid was refined twice to achieve the stated mesh resolution at the immersed boundary.

Table 5.12: Angles of inclination for each variation of the inclined SWBLI case

Label	$C0$	$C1$	$C2$	$C3$	$C4$
Inclination angle (degrees)	0	3	6	9	12

Table 5.13: Flow conditions for the inclined SWBLI test case

M_∞	Re_1	u_∞	P_∞	T_∞	T_w	δ_0
6.0	6.0×10^6	969 m/s	498 Pa	65 K	300 K	$1.1{\times}10^{-3}$ m

Figure 5.10(b) shows a visualisation of the flowfield the flow field for a 12° angle. Skin friction and surface heat transfer data for various angles are shown in figure 5.11. As seen in figure 5.11(a), heat transfer is well-matched across angles in the upstream region, though laminar heat transfer is marginally lower than reference values. This discrepancy could be due to disturbances being slightly shifted downstream, which reduces their development time, or due to the challenges of consistently imposing the laminar boundary layer with varying angles due to mesh resolution and the wall model's limitations for laminar flow. The current framework is not optimised for prediction of laminar boundary layers, as noted in other studies [58, 136, 137]. However, the transitional heat transfer peak downstream of the separation bubble is consistent across cases, within about 5%.

It is evident from the skin friction data in figure 5.11(b) that there is a small mismatch in the length of the separation bubble across different cases. Specifically, the separation bubble length tends to increase slightly with the angle of inclination. This trend is also visible in the streamwise

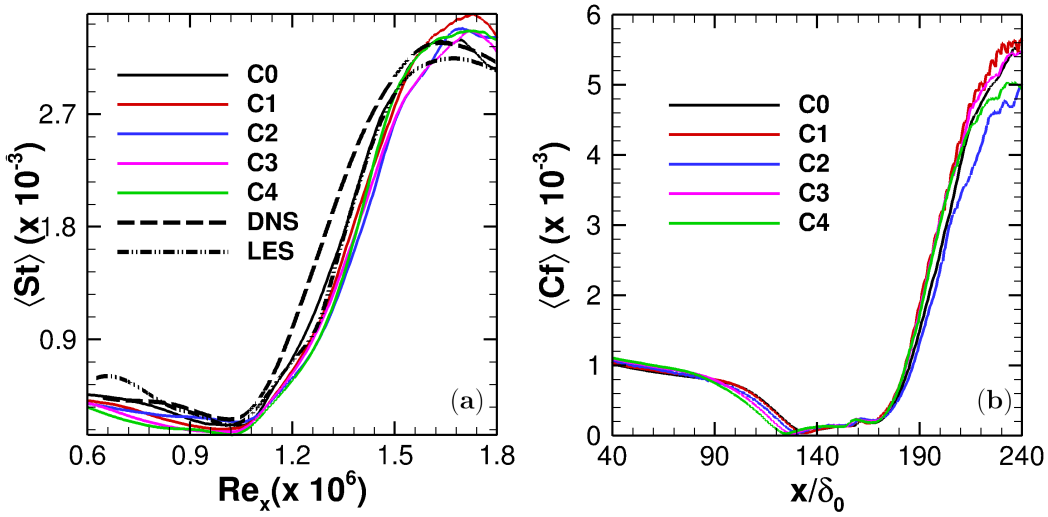


Figure 5.11: Comparison of surface quantities: (a) Stanton number St and (b) skin friction coefficient C_f, for the five different configurations of the SWBLI case. DNS reference from Sandham[6], and the reference LES from of Yang [7]. Figure recreated from [5].

component of the mean normalised velocity vector at the wall shown in figure 5.12(b). Despite consistent reattachment at the shock impingement location, the variation likely stems from the numerical treatment or other causes that require further investigation. Except for the slight mismatch in separation location, the pressure distributions in figure 5.12(a) show good agreement. However, there is noticeable numerical oscillation in the downstream pressure distribution and skin friction. This issue may be due to artefacts from the shock wave interacting with a coarse-fine grid interface, which disperses high-frequency acoustic waves onto the surface.

Figure 5.13 shows spanwise- and temporally-averaged plots of the temperature and pressure fields. The pressure and temperature contours agree well overall, though there are slight mismatches: one in the temperature within the separation bubble and another in the shape of the thermal boundary layer in the transitional region. The temperature mismatch in the separation bubble may be due to variations in the inflow boundary condition. Differences in the transitional region could arise from increased numerical dissipation in the streamwise direction due to the angle of inclination. When the geometry is aligned with the grid, irregular points have irregular flux differentiation stencils only in the wall-normal direction. Introducing a small angle of inclination makes some points irregular in both the wall-normal and streamwise directions, affecting the streamwise flux differentiation operator. This effect can be more pronounced in areas with

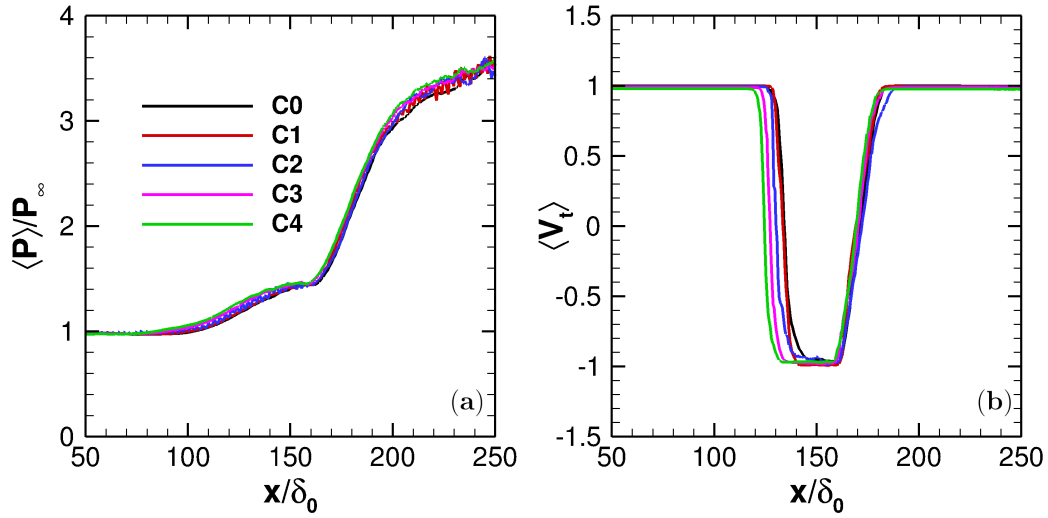
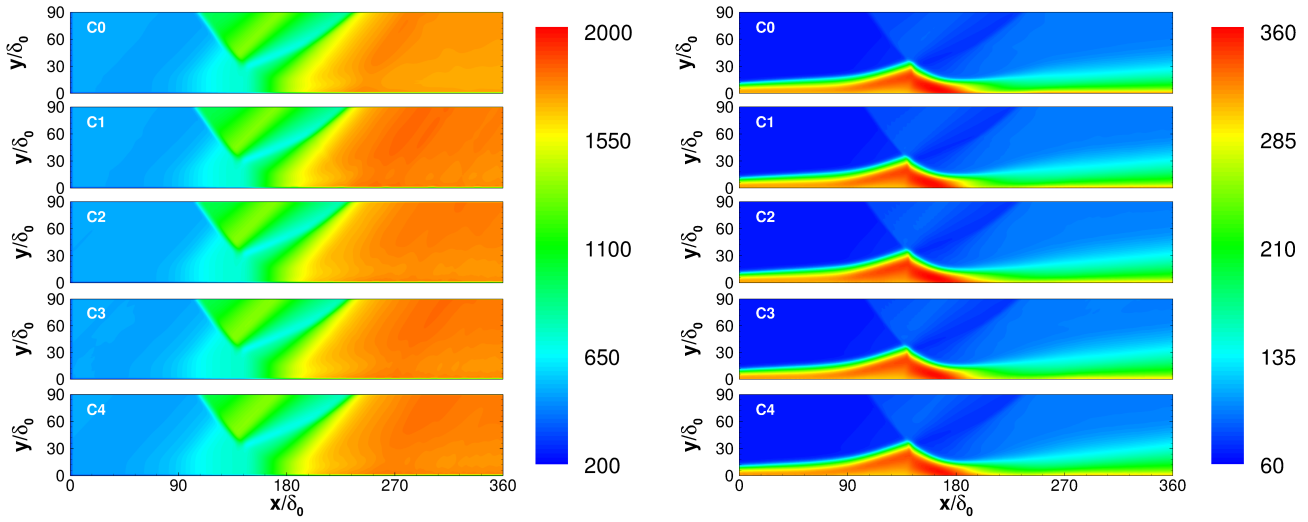


Figure 5.12: Comparison of surface quantities: (a) normalised mean surface pressure, $\langle P \rangle / P_\infty$ and (b) wall-tangential component of the unit velocity vector, $\langle V_t \rangle$ for the five different configurations of the SWBLI case. Figure recreated from [5].

significant streamwise flow variation.



(a) (b)

Figure 5.13: Colour contours of mean (a) pressure and (b) temperature for the five different configurations of the SWBLI case. Figure recreated from [5].

In figure 5.14, the spanwise- and temporally-averaged components of the Reynolds stress tensor are shown. The profiles for the streamwise normal Reynolds stress $\langle u'u' \rangle$ agree well overall. However, in the $C0$ case, a peak in $\langle u'u' \rangle$ is visible at the wall for a few cells, which is the most notable difference in the Reynolds stress profiles. This peak indicates variations in turbulent fluctuation

levels with increasing angle of inclination. A closer look at the inclined contours of $\langle u'u' \rangle$ shows periodic damping of velocity fluctuations, supporting the idea that the solution is influenced by irregular points in the streamwise direction.

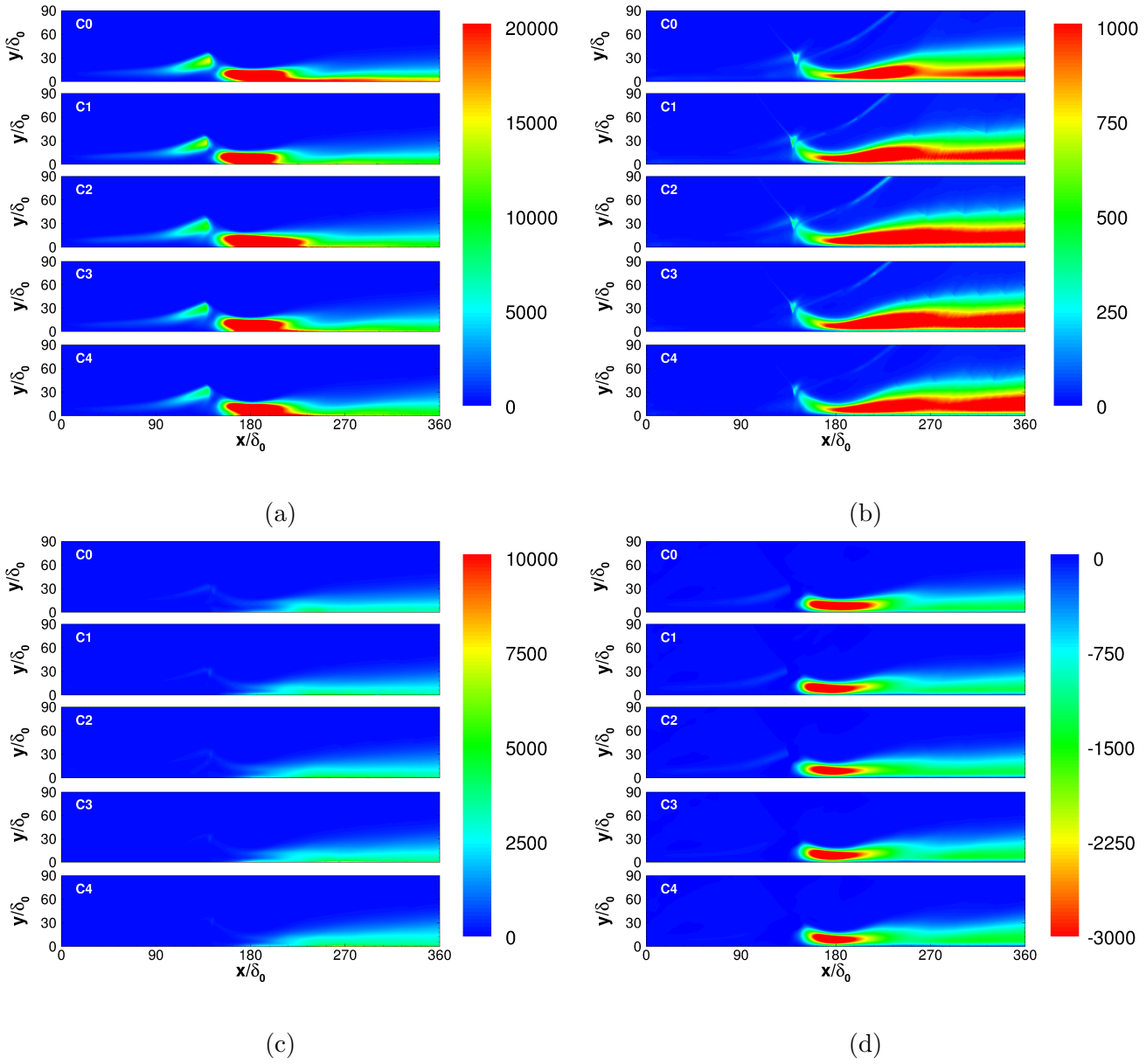
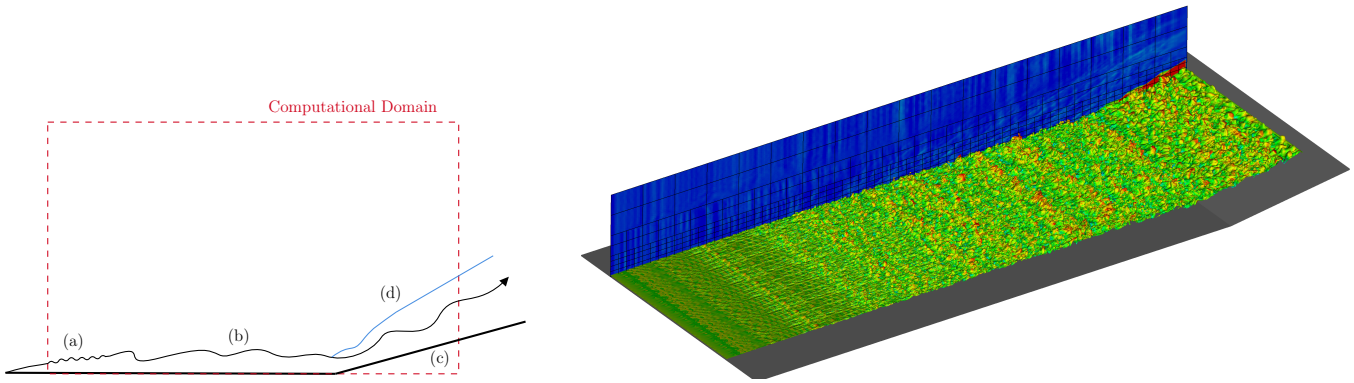


Figure 5.14: Colour contours of the nonzero Reynolds stress tensor components: (a) $\langle u'u' \rangle$, (b) $\langle v'v' \rangle$, (c) $\langle w'w' \rangle$, and (d) $\langle u'v' \rangle$. Figure recreated from [5].

The profiles of $\langle v'v' \rangle$ reveal an artefact near the boundary layer edge for cases $C1$ through $C4$, showing a characteristic spatial frequency. Investigation has suggested that this artefact arises from acoustic radiation reflected back towards the boundary layer due to the impedance between fine and coarse grid blocks. While these reflections have minimal impact on mean profiles, they

affect higher-order statistics.

5.5 Hypersonic Compression Ramp at Mach 7.2



(a) (b)

Figure 5.15: (a) Schematic diagram for the 8° compression corner case showing the: (i) perturbed laminar boundary layer inflow, (ii) transition and turbulent boundary layer, (iii) 8° compression ramp, and (iv) shock wave originating from the compression ramp. b) Instantaneous flow field visualisation for the 8° compression ramp test case showing isosurfaces of temperature at $T = 150$ K coloured by the streamwise velocity with a slice of instantaneous pressure superimposed with the block-structured Cartesian grid on the rear plane. Figure recreated from [5].

This test case builds on the work of Priebe & Martin[8], who performed DNS of a Mach 7.2 boundary layer impacting an 8° compression ramp (see figure 5.15(i)). At these conditions, the mean flow remains attached at the corner, but instantaneous flow separation occurs often due to a significant mean streamwise pressure gradient that violates some wall-model assumptions, such as convective balance and the wall-parallel assumption. A laminar boundary layer profile is prescribed at the inflow, with a density perturbation field ρ' applied to induce transition according to (5.4). For this case, $A = 10^{-3}$ and $f_0 = 17$ kHz. A boundary layer thickness of $\delta = 0.9\delta_{\text{ref}}$ is achieved $10\delta_{\text{ref}}$ upstream of the compression ramp. The flow conditions are detailed in table 5.14.

Table 5.14: Flow conditions for the 8° compression corner case

M_∞	Re_1	u_∞	P_∞	T_∞	T_w	δ_{ref}
7.2	1.89×10^7	1146 m/s	1341 Pa	62.9 K	340 K	5 mm

An instantaneous visualisation of the flow field for this case is shown in figure 5.15(ii). The wall-model coupling location is set to $0.11\delta_{\text{ref}}$. The solution was computed on two different meshes to assess the effect of mesh refinement, with grid details provided in table 5.15. The wall model

coupling location was not changed with the grid spacing. The coarse grid had a wall spacing coarser than the typical resolution recommended by Kawai & Larsson [93], with $\Delta y^+ \approx 80$.

Table 5.15: Grid details for the mesh used in the 8° compression corner case. The reported grid spacing applies to the blocks at the finest level of the grid.

Total Grid Size	$\Delta x^+ \times \Delta y^+ \times \Delta z^+$	$L_x \times L_y \times L_z$
11M	$160 \times 80 \times 190$	$68\delta_{\text{ref}} \times 9\delta_{\text{ref}} \times 10\delta_{\text{ref}}$
50M	$120 \times 40 \times 180$	$68\delta_{\text{ref}} \times 9\delta_{\text{ref}} \times 10\delta_{\text{ref}}$
89M	$116 \times 32 \times 171$	$68\delta_{\text{ref}} \times 9\delta_{\text{ref}} \times 10\delta_{\text{ref}}$

Surface pressure, heat transfer, and skin friction are compared to the reference DNS in figures 5.16, 5.17a, and 5.17b, respectively. Note that some level of noise is present due to the fact that averaging is performed over a triangulation that is not uniform in the spanwise direction. The equilibrium wall model used in the present method accurately captures the surface pressure distribution, as expected since pressure is only weakly affected by viscous effects. However, the flow exhibits statistically significant instantaneous separation at the corner. Figure 5.16 shows some disagreement in pressure distribution upstream of the compression corner for the coarse mesh, notably a slight overprediction of surface pressure. This early rise in surface pressure suggests that the coarse grid may not accurately capture the extent of instantaneous separation, affecting the pressure downstream, similar to the findings in section 5.4.1. For the fine grid, the pressure distribution is well captured, though the wall-to-freestream pressure ratio differs slightly from unity upstream of the corner, which is not considered significant.

Upstream of the compression ramp, both heat transfer and skin friction are well-predicted. Given previous validation of this method for transitional boundary layers [138], the upstream match is expected. The coarse mesh shows a premature drop in both quantities as the boundary layer approaches the compression corner, indicating an overprediction of instantaneous flow separation. The minimum value of the skin friction curve is higher than shown by DNS, possibly due to inconsistent instantaneous flow separation along the spanwise extent of the domain. In contrast, the fine mesh results align closely with DNS upstream of the compression corner, demonstrating that even with a streamwise grid resolution of $\Delta x^+ = 120$, the effects of flow separation are captured with reasonable accuracy.

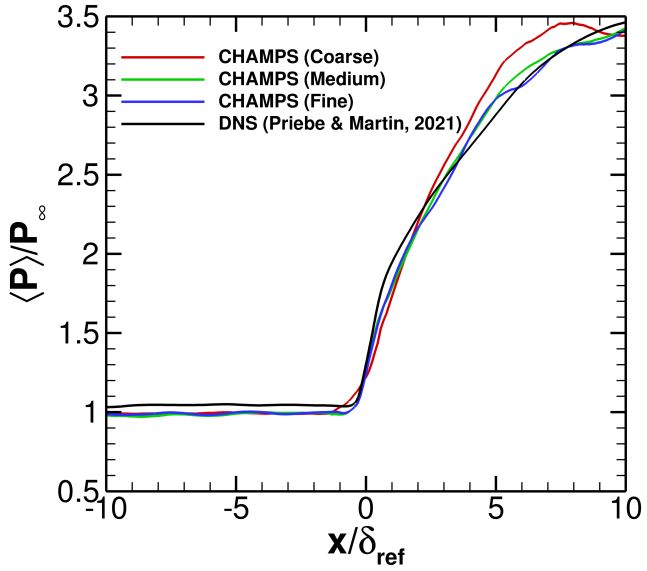


Figure 5.16: Comparison of normalised mean surface pressure, $\langle P \rangle / P_\infty$ for the 8° compression corner with the DNS from Priebe & Martin[8]. Note that $\langle \cdot \rangle$ designates an averaged quantity in time and the homogeneous direction. Figure recreated from [5].

Downstream of the compression corner, the DNS skin friction and heat transfer quickly adjust after compression through the shock, with the adjustment extent being negligible compared to the IBM-WMLES results. On the coarse mesh, there is an adjustment length of approximately $4\delta_{\text{ref}}$ before the post-shock surface quantities align with DNS data. In contrast, the adjustment region is smaller for the fine mesh, though a small adjustment to the DNS data still occurs after a jump in the surface quantities. The cause of this adjustment is unclear, but it may be due to an imbalance between the tangential pressure gradient and the convective terms that the wall model does not capture, even with a sufficiently fine grid resolution.

As mentioned in table 5.15, the coarse computational grid for this test case required approximately 11 million cells. In contrast, the reference DNS used a grid size of 96 million cells for the compression ramp domain and 138 million cells for the upstream boundary layer domain. Given that the present work includes the entire upstream boundary layer, the computational cost of the coarse simulation (considering only the spatial requirement) is less than 5% of the total cost of the DNS.

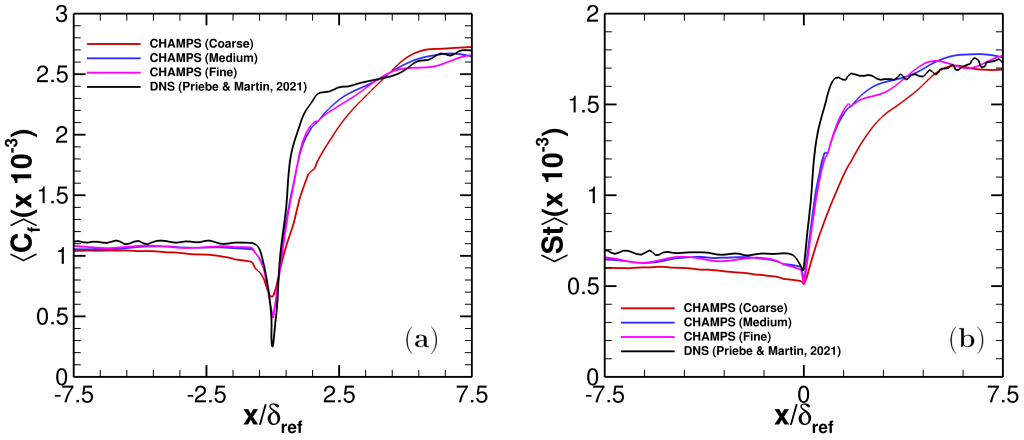


Figure 5.17: Comparison of surface quantities: (a) Stanton number, St and (b) skin friction coefficient for the 8° compression corner with the DNS from Priebe & Martin[8]. Figure recreated from [5].

Chapter 6: Applications

This chapter presents, in brief, a gallery of past and ongoing applications of the computational framework developed in this thesis. Because these are not canonical flow problems, a detailed discussion of comparisons with reference data is out of scope of this work. The test problems here are chosen to showcase application of the newly-developed IBM-WMLES to flow simulations that involve more complex geometry than those presented in chapter 5. These simulations demonstrate that the approach can be readily applied to practical flow problems that influence real-world vehicle designs.

6.1 Mach 4 Double Fin

This test problem contains similar challenges as those presented in sections 5.4.2 and 5.5, but includes three-dimensional aspects introduced by way of two vertically-oriented fins that compress the incoming flow, resulting in crossing shock waves that interact with the incoming turbulent boundary layer. The sharp corners present in this case introduce vortices that propagate into the channel between the two fins. The flowfield is visualised in figure 6.1, and the flow conditions are listed in table 6.1.

Table 6.1: Freestream conditions and grid parameters for the double fin case.

M_∞	Re_1	T_∞ (K)	P_∞ (Pa)	α_{fin}	$\Delta x/\delta \times \Delta y_{\text{min}}/\delta \times \Delta z/\delta$
4	80×10^6	70.24	9876	15.0°	$1/8 \times 1/16 \times 1/13$

Figure 6.2 shows a qualitative comparison against reference data of Gaitonde [9] of the mean shear stress near the entrance to the channel formed by the two fins. This case involves complex shock-boundary layer interactions, where the shock waves from the fins compress the incoming flow, creating high streamwise pressure gradients at the surface. Upstream vortices generated by the fin shocks propagate downstream, interacting with the boundary layer and inducing localised separation. A few key flow features are highlighted: the vortex in front of the shock emanating from the fin (yellow), the interaction of the two shocks with the incoming boundary layer (white),

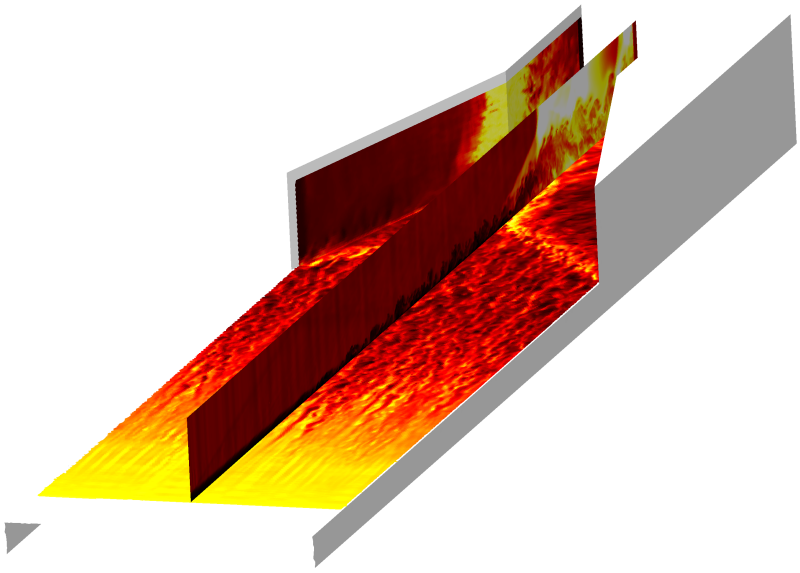


Figure 6.1: Flow over a double-fin configuration at $M = 4$. Surface data shows the wall model input velocity on the surface and the streamwise velocity is shown in the volume slice.

and the interaction of the corner flow vortices with the boundary layer near the centre of the channel.

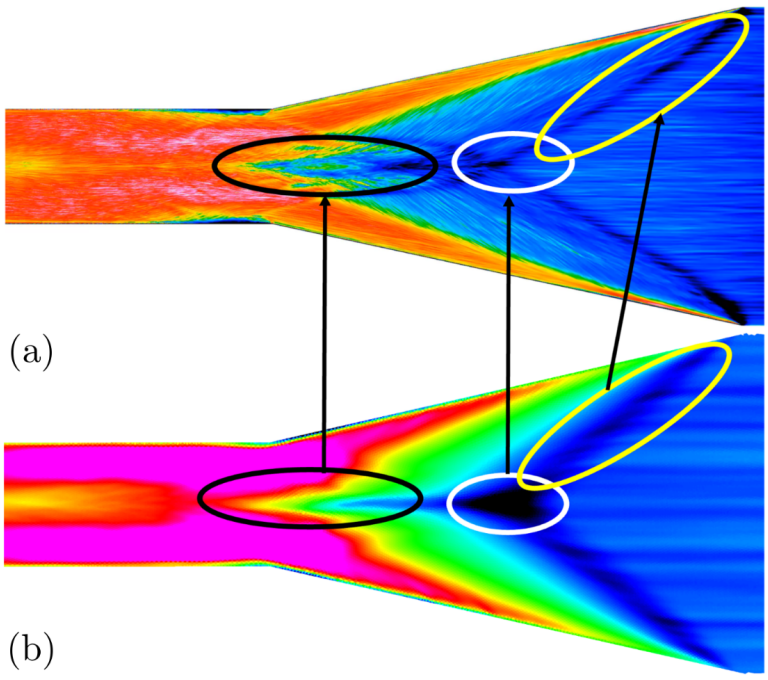


Figure 6.2: Qualitative comparison of flow features between (a) present framework and (b) reference data from ref. [9]. In both cases, the surface skin friction is depicted.

6.2 Mach 4 Streamtraced Engine

This case features an engine geometry that has been studied recently [139, 10]. This engine simulation highlights the ability of IBM-WMLES to handle complex geometries involving both convex and concave surfaces. Such configurations present unique challenges for computational methods where traditional mesh generation is required. The flow accelerates through the engine inlet, leading to compression and shock formation. These shock waves influence the boundary layer, creating pressure gradients along the surface. The simulation replicates these features, indicating the solver's ability to handle complex geometries and flow interactions. The freestream flow conditions used for this simulation are outlined in Table 6.2, and the engine inlet geometry is illustrated in Figure 6.3. A qualitative comparison between the simulation results and experimental data from [10] is shown in Figure 6.4. This comparison reveals that the simulation effectively captures the primary flow characteristics.

Table 6.2: Freestream conditions for the streamtraced engine case.

M_∞	Re_1	T_∞ (K)	P_∞ (Pa)	U (m/s)
4	50×10^6	70.0	6900	671

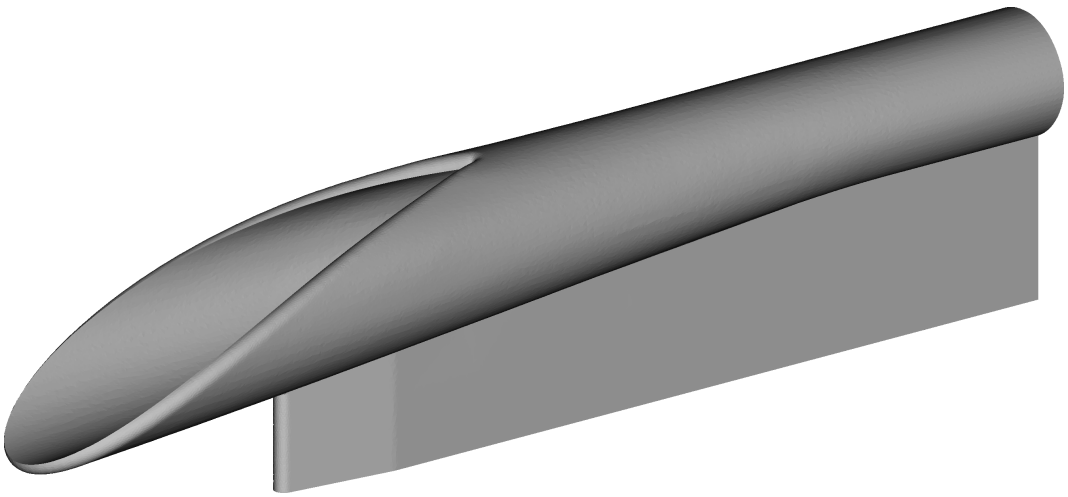


Figure 6.3: Surface geometry for the streamstraced engine.

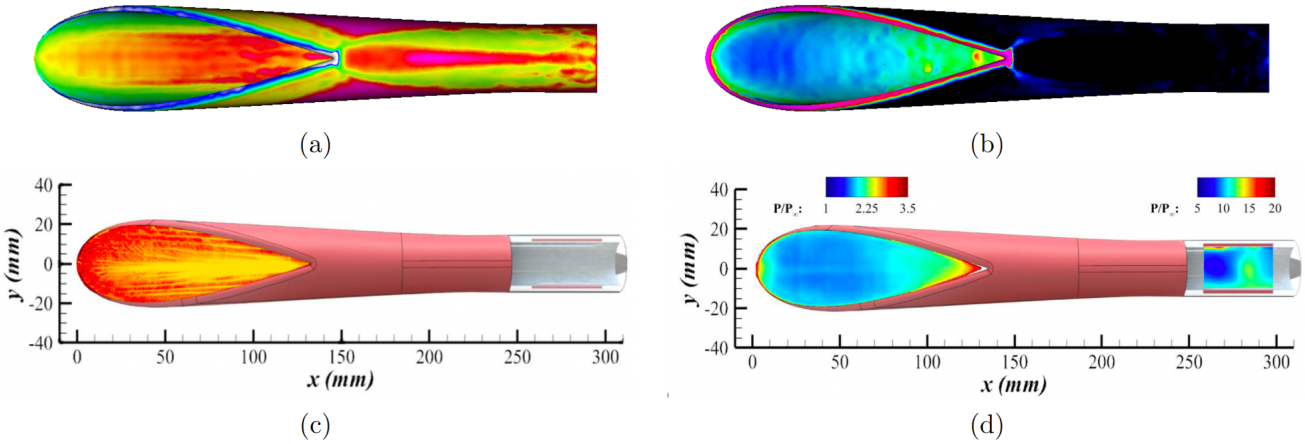


Figure 6.4: Qualitative comparison of (a) surface skin friction predicted by the current framework with (b) the experimental data of ref. [10] and (c) surface pressure from current framework with (d) experimental data.

6.3 Mach 3 Internal Ramjet Flow at Off-Design Conditions

This flow problem features internal flow through a ramjet duct at Mach 3 under off-design conditions. The inlet is long to facilitate the development of a turbulent boundary layer. The flow is then compressed slightly and then expanded again as it flows toward the main compression region. At the compression region, the duct is truncated to compress the flow, which would typically be to facilitate a combustion process by increasing air density. Finally, the flow is expanded through a diverging section, at which point it becomes supersonic and exits the computational domain. The flow geometry is shown in figure 6.5, and the flow conditions are shown in figure 6.3.

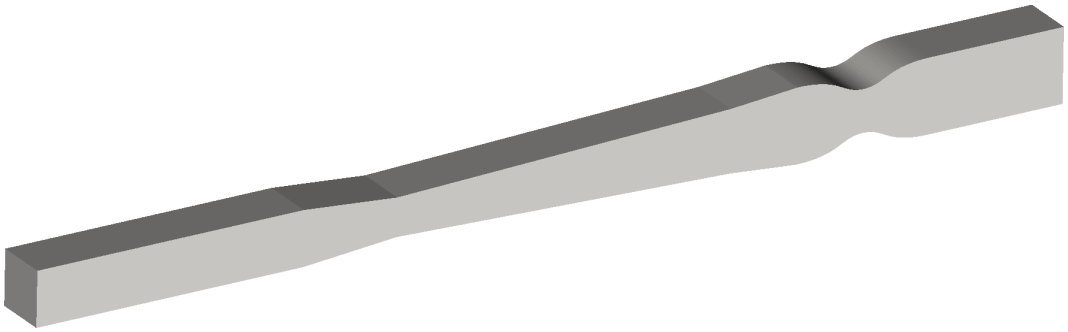


Figure 6.5: Ramjet flow geometry.

In total, 100 million grid cells are used. Running on a laptop with a single NVIDIA RTX 4090 Mobile, converged simulation results are obtained in less than 8 hours. The flow field is visualised

Table 6.3: Freestream conditions for the streamtraced engine case.

M_{inlet}	Re_L	L (m)	$T_{0,\text{inlet}}$ (K)	T_w (K)	P_∞ (Pa)
3	4.3×10^6	0.4 m	300	300	4000

in figure 6.6. For this simulation, the flow naturally transitions in the inlet region with no added disturbances. One of the main features of the flowfield is the bifurcation that occurs upstream of the final compression region: the backpressure experienced by the incoming turbulent flow causes the flow to bias to one side, resulting in a steady flowfield that is not symmetric.

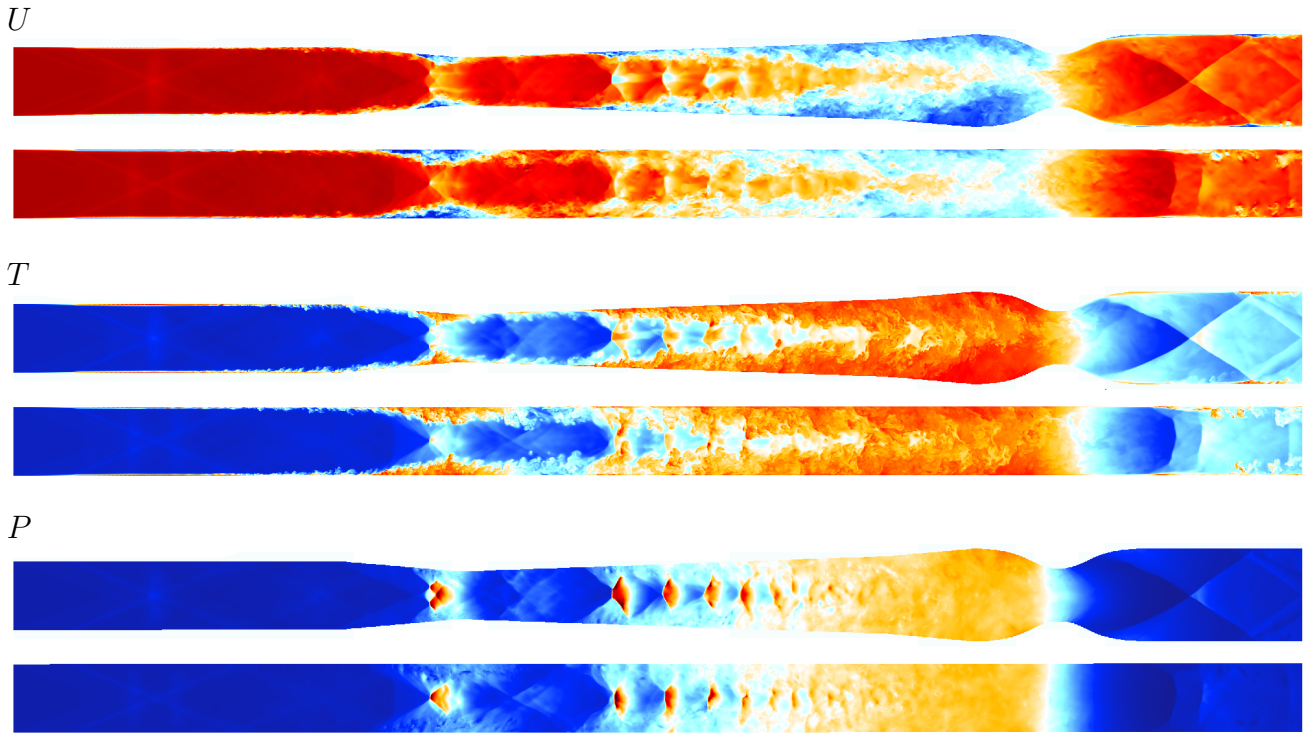


Figure 6.6: Visualisation of off-design Ramjet flow at Mach 3. Top to bottom: streamwise velocity (spanwise symmetry plane), streamwise velocity (wall-normal symmetry plane), temperature (spanwise symmetry plane), temperature (wall-normal symmetry plane), pressure (spanwise symmetry plane), and pressure (wall-normal symmetry plane).

6.4 BOLT Flight Vehicle at Mach 6

The BOLT flight geometry [12, 140, 141, 142] is simulated at Mach 6 to evaluate the capability of the IBM-WMLES to simulate flows with thin shock layers for blunted leading edges. This

geometry has been designed to study hypersonic transition, but the configuration here includes a blowing slot near the leading edge to induce rapid transition. Figure 6.7 depicts the geometry for this case, and tables 6.4 and 6.5 give the simulation parameters.

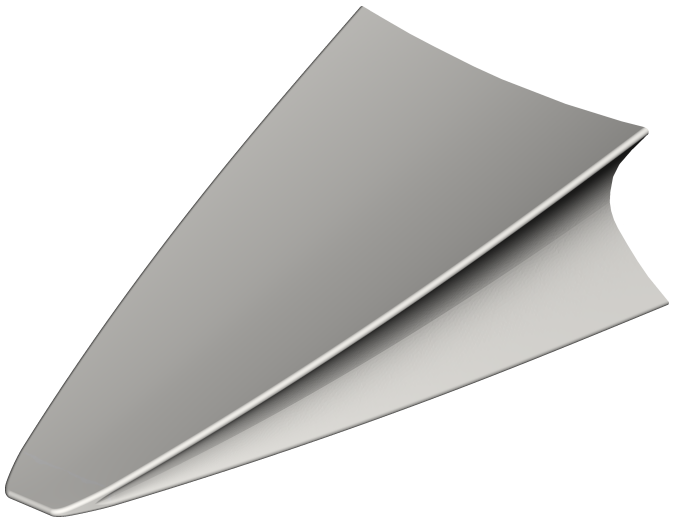


Figure 6.7: BOLT flight geometry.

Table 6.4: Freestream conditions for the BOLT flight geometry case.

M_∞	Re_L	L (m)	T_∞ (K)	T_w (K)	P_∞ (Pa)
6.04	6×10^6	1	213	300	2950

Table 6.5: Mesh parameters for the BOLT flight geometry case.

Δx / Δy / Δz	Num. Cells
$1.3 \times 10^{-4}L$ / $1.1 \times 10^{-4}L$ / $1.1 \times 10^{-4}L$	900 M

The Mach 6 simulation of the BOLT flight vehicle demonstrates the solver's ability to accurately handle hypersonic flows with shock standoff, a key feature over blunted leading edges. The shock layer remains detached from the surface and the solver captures the shock structure and boundary layer interactions with high fidelity.

The surface data for this case is visualised in figure 6.8. The blowing slot near the nose significantly affects the flowfield near the nose, but the boundary layer recovers approximately 30% of the way down the extent of the vehicle. Because the Mach number is 6 and the shock

standoff is included, the characteristic transformations (2.56) and (2.57) are included in the upwind flux reconstruction.

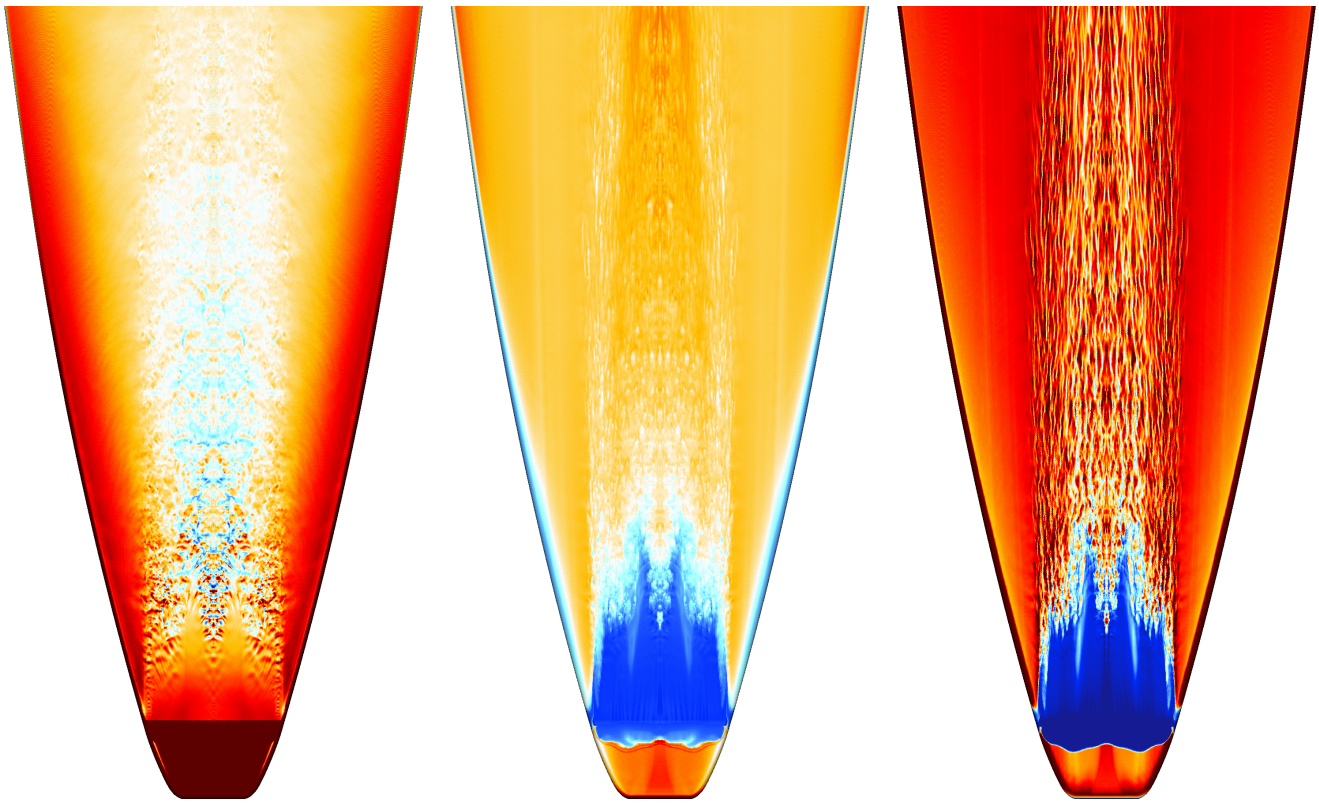


Figure 6.8: Surface data for Mach 6 flow over the BOLT flight vehicle. Left to right: pressure, temperature, and wall model sample velocity.

Chapter 7: Conclusion

7.1 Summary

A new computational framework has been developed for the simulation of turbulent flows around hypersonic vehicles. The framework combines immersed boundary methods and wall-modelled large-eddy simulation. While immersed boundary methods have had limitations in the past for high Reynolds number and high Mach number flows, a new ghost cell formulation enables the IBM-WMLES to be applied in the hypersonic regime. The new ghost cell method is predicated on four novel ideas:

- that a ghost-cell immersed boundary method can be parameterised and have its stability analysed via pseudospectral stability analysis,
- that the action of diffusive flux at the boundary is similar to that of an error in the wall model but can be ameliorated by carefully treating the ghost cell states with extrapolation using a second sampling point,
- that a stable upwind scheme can be formulated with apparently downwind biassing, and
- that the use of a second ghost cell in the vicinity of the boundary is significantly detrimental to the accuracy of heat flux predictions, thus giving rise to a ghost cell scheme that is higher-order but requires the use of only a single ghost cell.

The resulting scheme is robust and is able to capture relevant flow physics for Mach numbers greater than 5. This represents a step forward in the development of immersed boundary schemes, since applications to hypersonic flows have seen little research.

Commutation errors in the wall model were closely investigated as a source of error in heat transfer and skin friction predictions at high Mach numbers. Although they have been alluded to in a handful of studies, commutation errors have not been identified as a significant source of error for compressible flows. This thesis demonstrated that for high Mach numbers and high Reynolds

numbers, commutation errors can account for as much of the error in heat transfer and shear stress predictions as the choice of the turbulence model when compared against DNS data. These errors only get worse at higher Mach numbers and have been shown to be dependent on the wall model sensitivity as well as the covariance of the wall model boundary conditions. A method of correcting for commutation errors was derived based on Taylor series expansion around the mean state, which was shown to be useful for eliminating commutation errors for idealised models in turbulent channel flows at high Mach numbers. The correction method is then implemented via a dual-number sensitivity estimate, and it is demonstrated that commutation errors influenced are especially reduced in regions of the shock wave boundary layer interaction.

The newly-developed IBM-WMLES framework was evaluated for computational efficiency. A detailed analysis of the hardware performance considerations was performed, showing that careful selection of memory addressing, kernel fusion, and communication minimisation was able to speed simulations up by approximately 15× over a naïve implementation, and over 200× over an initial CPU implementation. It was shown that with this implementation and algorithm, the computational cost of IBM-WMLES simulations can be significantly competitive with RANS.

Finally, the present IBM-WMLES was validated for a range of canonical flows and common flow problems. It was shown that the method can robustly predict viscous surface quantities, even in the presence of complex flow physics like transition and shock wave boundary layer interactions. One of the most critical features of the present IBM-WMLES is its ability to give rotationally-invariant predictions for heat flux, which has not been demonstrated before.

7.2 Future Work

IBM-WMLES to hypersonic flows can readily be extended to include more flow physics. An example of this is fluid-structure interaction, where the solid vehicle boundary moves in time. Immersed boundary methods are well-suited for this, as they do not require alignment of the computational grid at boundaries. Flows with chemical reactions are a clear choice for extending this work to very high Mach numbers, although care will have to be taken for turbulent flows with chemistry.

References

[1] C. D. Argyropoulos and N. C. Markatos. Recent advances on the numerical modelling of turbulent flows. *Applied Mathematical Modelling*, 39, 2014.

[2] N. Ashton and W. van Noordt. Overview and summary of the first automotive cfd prediction workshop: Drivaer model. *SAE International Journal of Commercial Vehicles*, 2022.

[3] J. B. Chapelier, D. Lusher, W. van Noordt, C. Wenzel, T. Gibis, P. Mossier, A. Beck, G. Lodato, C. Brehm, M. Ruggeri, C. Scalo, and N. Sandham. Comparison of high-order numerical methodologies for the simulation of the supersonic Taylor–Green vortex flow. *Physics of Fluids*, 36(5), 2024.

[4] Y. Tamaki and S. Kawai. Wall-resolved large-eddy simulation of near-stall airfoil flow at $\mathrm{Re}_c = 10^7$. *AIAA Journal*, 61(2), 2023.

[5] W. van Noordt, S. Ganju, and C. Brehm. An immersed boundary method for wall-modeled large-eddy simulation of turbulent high-Mach-number flows. *Journal of Computational Physics*, 470, 2022.

[6] N. Sandham, E. Schülein, A. Wagner, S. Willems, and J. Steelant. Transitional shock-wave/boundary-layer interactions in hypersonic flow. *Journal of Fluid Mechanics*, 752, 2014.

[7] X. Yang, J. Urzay, S. Bose, and P. Moin. Aerodynamic heating in wall-modeled large-eddy simulationof high-speed flows. *AIAA Journal*, 2018.

[8] S. Priebe and M. P. Martín. Turbulence in a hypersonic compression ramp flow. *Physical Review Fluids*, 6(3), 2021.

[9] D. Gaitonde and J. Shang. Structure of a turbulent double-fin interaction at Mach 4. *AIAA Journal*, 33(12), 1995.

[10] E. Johnson, C. Jenquin, J. McCready, V. Narayanaswamy, and J. Edwards. Mach 4 Performance of a Hypersonic Streamtraced Inlet Part 1: Experimental Investigations. *AIAA Science and Technology Forum and Exposition, AIAA SciTech Forum 2022*, 2022.

[11] R. Hallion. The history of hypersonics: or, 'back to the future-again and again'. *43rd AIAA Aerospace Sciences Meeting and Exhibit*, 2005.

[12] B. Wheaton, C. Butler, G. McKiernan, and D. Berridge. Initial results from the bolt flight experiment. *AIAA SciTech Forum*, 2022.

[13] J. Walker. From columbia to discovery: Understanding the impact threat to the space shuttle. *International Journal of Impact Engineering*, 36, 2009.

[14] J. Slotnick, A. Khodadoust, J. Alonso, D. Darmofal, W. Gropp, E. Lurie, and D. Mavriplis. Cfd vision 2030 study: A path to revolutionary computational aerosciences. *NASA/CR–2014-218178*, 2014.

[15] R. Rogallo and P. Moin. Numerical simulation of turbulent flows. *Annual Review of Fluid Mechanics*, 1984.

[16] M. Brachet, D. Meiron, S. Orszag, B. G. Nickel, R. Morf, and U. Frisch. Small-scale structure of the taylor–green vortex. *Journal of Fluid Mechanics*, 1983.

[17] C. Roy and F. Blottner. Review and assessment of turbulence models for hypersonic flows. *Progress in Aerospace Sciences*, 2006.

[18] F. Nicoud, H. Toda, O. Cabrit, S. Bose, and J. Lee. Using singular values to build a subgrid-scale model for large eddy simulations. *Physics of Fluids*, 23, 2011.

[19] P. Moin, K. Squires, W. Cabot, and S. Lee. A dynamic subgrid-scale model for compressible turbulence and scalar transport. *Physics of Fluids*, 3, 1991.

[20] J. Bertin and R. Cummings. Fifty years of hypersonics: where we've been, where we're going. *Progress in Aerospace Sciences*, 39, 2003.

[21] X. Yang and K. Griffin. Grid-point and time-step requirements for direct numerical simulation and large-eddy simulation. *Physics of Fluids*, 33, 2021.

[22] J. Deardorff. A numerical study of three-dimensional turbulent channel flow at large Reynolds numbers. *Journal of Fluid Mechanics*, 41, 1970.

[23] J. Chawner, J. Dannenhoffer III, and N. Taylor. Geometry, mesh generation, and the cfd 2030 vision. *AIAA Aviation Forum*, 2016.

[24] J. Chawner and N. Taylor. Progress in geometry modeling and mesh generation toward the cfd vision 2030. *AIAA Aviation Forum*, 2019.

[25] B. J. Boersma and S. K. Lele. Large eddy simulation of compressible turbulent jets. *Center for Turbulence Research Annual Research Briefs*, 1999.

[26] A. Trettel and J. Larsson. Mean velocity scaling for compressible wall turbulence with heat transfer. *Physics of Fluids*, 28, 2016.

[27] C. Cheng and L. Fu. Mean temperature scalings in compressible wall turbulence. *Physics of Fluids*, 9, 2024.

[28] K. Griffin, L. Fu, and P. Moin. Velocity transformation for compressible wall-bounded turbulent flows with and without heat transfer. *Proceedings of the National Academy of Sciences*, 2021.

[29] C. Peskin. Numerical Analysis of Blood Flow in the Heart. *J. Comput. Phys.*, 25, 1977.

[30] C. S. Peskin. The Immersed Boundary Method. *Acta Numerica, Cambridge University Press*, 2002.

[31] D. Goldstein, R. Handler, and L. Sirovich. Modeling a Non–Slip Flow Boundary with an External Force Field. *Journal of Computational Physics.*, 105, 1993.

[32] H. Johansen and P. Collela. A Cartesian Grid Embedded Boundary Method for Poisson's Equation on Irregular Domains. *Journal of Computational Physics.*, 147, 1998.

[33] R. LeVeque, C. Peskin, and P. Lax. Solution of a two-dimensional cochlea model using transform techniques. *SIAM Journal of Applied Mathematics*, 40, 1985.

[34] R. Dillon, L. Fauci, A. Fogelson, , and D. Gaver III. Modeling biofilm processes using the immersed boundary method. *Journal of Computational Physics*, 129, 1996.

[35] P. Angot, C. H. Bruneau, , and P. Fabrie. A penalization method to take into account obstacles in incompressible viscous flows. *Numerical Mathematics*, 81, 1999.

[36] D. Clarke, M. Salas, and H. Hassan. Euler calculations for multielement airfoilsusing cartesian grids. *AIAA Journal*, 24, 1986.

[37] P. Frymier Jr., H. Hassan, and M. Salas. Navier-Stokes calculations using cartesian grids: I. laminar flows. *AIAA Journal*, 26, 1988.

[38] J. Melton, M. Berger, M. Aftosmis, and M. Wong. 3d applications of a cartesian grid euler method. *AIAA Paper 95-0853*, 1995.

[39] D. DeZeeuw and K. Powell. An adaptively refined cartesian mesh solver for the euler equations. *Journal of Computational Physics*, 104, 1993.

[40] J. Quirk. An alternative to unstructured grids for computing gas dynamic flows around arbitrarily complex two-dimensional bodies. *Computers & Fluids*, 23, 1994.

[41] M. Nemec, M. Aftosmis, and T. Pulliam. Aiaa aerospace sciences meeting and exhibit. *Computers & Fluids*, 2004.

[42] G. Iaccarino and R. Verzicco. Immersed boundary technique for turbulent flow simulations. *ASME Applied Mechanics Reviews*, 56, 2003.

[43] M. Bernardini, D. Modesti, and S. Pirozzoli. On the suitability of the immersed boundary method for the simulation of high-Reynolds-number separated turbulent flows. *Computers and Fluids*, 130, 2016.

[44] M. Tyagi and S. Acharya. Large eddy simulation of turbulent flows in complex and moving rigid geometries using the immersed boundary method. *International Journal for Numerical Methods in Fluids*, 48, 2005.

[45] P. Chang, C. C. Liao, H. W. Hsu, S. H. Liu, and C. A. Lin. Simulations of laminar and turbulent flows over periodic hills with immersed boundary method. *Computers & Fluids*, 2014.

[46] Y. Tamaki and T. Imamura. Turbulent flow simulations of the common research model using immersed boundary method. *AIAA Journal*, 56, 2018.

[47] P. Spalart and S. Allmaras. A one-equation turbulence model for aerodynamic flows. *Aerospace Sciences Meeting and Exhibit*, 1992.

[48] B. Shi, Z. Xu, and S. Wang. A non-equilibrium slip wall model for large-eddy simulation with an immersed boundary method. *AIP Advances*, 2022.

[49] M. Ma, W. Huang, C. Xu, and G. Cui. A hybrid immersed boundary/wall-model approach for large-eddy simulation of high-Reynolds-number turbulent flows. *International Journal of Heat and Fluid Flow*, 2021.

[50] S. Kang. An improved near-wall modeling for large-eddy simulation using immersed boundary methods. *International Journal for Numerical Methods in Fluids*, 2015.

[51] Z. Chen, S. Hickel, A. Devesa, J. Berland, and N. Adams. Wall modeling for implicit large-eddy simulation and immersed-interface methods. *Theoretical and Computational Fluid Dynamics*, 2013.

[52] F. De Vanna, G. Baldan, F. Picano, and E. Benini. On the coupling between wall-modeled les and immersed boundary method towards applicative compressible flow simulations. *Computers & Fluids*, 266, 2023.

[53] S. Cai, J. Jacob, and P. Sagaut. Immersed boundary based near-wall modeling for large eddy simulation of turbulent wall-bounded flow. *Computers & Fluids*, 259, 2023.

[54] H. Atmani, R. Zamansky, E. Climent, and D. Legendre. Stochastic wall model for turbulent pipe flow using immersed boundary method and large eddy simulation. *Computers & Fluids*, 239, 2022.

[55] X. I. A. Yang, J. Sadique, R. Mittal, and C. Meneveau. Integral wall model for large eddy simulations of wall-bounded turbulent flows. *Physics of Fluids*, 27(2), 02 2015.

[56] A. Tyliszczak and M. Ksiezyk. Large eddy simulations of wall-bounded flows using a simplified immersed boundary method and high-order compact schemes. *International Journal for Numerical Methods in Fluids*, 87(7), 2018.

[57] C. Chen, Z. Wang, L. Du, D. Sun, and X. Sun. Simulating unsteady flows in a compressor using immersed boundary method with turbulent wall model. *Aerospace Science and Technology*, 115, 2021.

[58] J. McQuaid, A. Zibitsker, B. Saikia, A. Martin, and C. Brehm. An Immersed Boundary Method for Hypersonic Viscous Flows. In *AIAA Scitech 2021 Forum*, AIAA SciTech Forum. American Institute of Aeronautics and Astronautics, 2021.

[59] S. Brahmachary, G. Natarajan, V. Kulkarni, N. Sahoo, V. Ashok, and V. Kumar. Role of solution reconstruction in hypersonic viscous computations using a sharp interface immersed boundary method. *Phys. Rev. E*, 103, 2021.

[60] T. Bridel-Bertomeu. Immersed boundary conditions for hypersonic flows using eno-like least-square reconstruction. *Computers & Fluids*, 215, 2021.

[61] A. Baskaya, M. Capriati, A. Turchi, T. Magin, and S. Hickel. Assessment of immersed boundary methods for hypersonic flows with gas–surface interactions. *Computers & Fluids*, 270, 2024.

[62] A. Baskaya, M. Capriati, D. Ninni, F. Bonelli, G. Pascazio, A. Turchi, T. Magin, and S. Hickel. Verification and validation of immersed boundary solvers for hypersonic flows with gas-surface interactions. In *AIAA AVIATION 2022 Forum*.

[63] L. Duan, X. Wang, and X. Zhong. A High–Order Cut–Cell Method for Numerical Simulation of Hypersonic Boundary–Layer Instability with Surface Roughness. *Journal of Computational Physics.*, 229, 2010.

[64] L. Trefethen and M. Embree. *Spectra and Pseudospectra – The Behavior of Nonnormal Matrices and Operators*. Princeton University Press, 2005.

[65] L. Grafakos. *Classical Fourier Analysis*. Springer, 2008.

[66] C. Brehm, M. Barad, J. Housman, and C. Kiris. A comparison of higher-order finite-difference shock capturing schemes. *Computers & Fluids*, 122, 2015.

[67] C. Bogey and C. Bailly. A family of low dispersive and low dissipative explicit schemes for flow and noise computations. *Journal of Computational Physics*, 194(1), 2004.

[68] N. Fleischmann, S. Adami, and N. Adams. Numerical symmetry-preserving techniques for low-dissipation shock-capturing schemes. *Computers & Fluids*, 189, 2019.

[69] L. Li, H. Wang, G. Zhao, M. Sun, D. Xiong, and T. Tang. An efficient low-dissipation hybrid central/weno scheme for compressible flows. *International Journal of Computational Fluid Dynamics*, 34(10), 2020.

[70] F. Nicoud and F. Ducros. Subgrid-Scale Stress Modelling Based on the Square of the Velocity Gradient Tensor. *Flow, Turbulence, and Combustion*, 62, 1999.

[71] F. Nicoud, H. Toda, O. Cabrit, S. Bose, and J. Lee. Using singular values to build a subgrid-scale model for large eddy simulations. *Physics of Fluids*, 23(8), 08 2011.

[72] A. Vreman. An eddy-viscosity subgrid-scale model for turbulent shear flow: Algebraic theory and applications. *Physics of Fluids*, 16(10), 10 2004.

[73] P. Chandrashekar. Kinetic energy preserving and entropy stable finite volume schemes for compressible Euler and Navier-Stokes equations. *Communications in Computational Physics*, 14(5), 2013.

[74] P. Subbareddy and G. Candler. A fully discrete, kinetic energy consistent finite-volume scheme for compressible flows. *Journal of Computational Physics*, 228(5), 2009.

[75] Y. Kuya, K. Totani, and S. Kawai. Kinetic energy and entropy preserving schemes for compressible flows by split convective forms. *Journal of Computational Physics*, 375, 2018.

[76] S. Pirozzoli. Generalized conservative approximations of split convective derivative operators. *Journal of Computational Physics*, 229(19), 2010.

[77] J. Crean, J. Hicken, D. Del Rey Fernàndez, D. Zingg, and M. Carpenter. Entropy-stable summation-by-parts discretization of the euler equations on general curved elements. *Journal of Computational Physics*, 356, 2018.

[78] M. Svärd and J. Nordström. Review of summation-by-parts schemes for initial–boundary-value problems. *Journal of Computational Physics*, 268, 2014.

[79] J. Nordström and T. Lundquist. Summation-by-parts in time. *Journal of Computational Physics*, 251, 2013.

[80] C. Brehm and H. Fasel. A novel concept for the design of immersed interface methods. *Journal of Computational Physics*, 242, 2013.

[81] F. Ducros, V. Ferrand, F. Nicoud, C. Weber, D. Darracq, C. Gacherieu, and T. Poinsot. Large-Eddy Simulation of the Shock/Turbulence Interaction. *Journal of Computational Physics*, 152(2), 1999.

[82] J. Blazek. *Computational Fluid Dynamics: Principles and Applications*. Elsevier, 2001.

[83] C.Shu and S. Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes. *Journal of Computational Physics*, 77(2), 1988.

[84] S. Bose and P. Moin. A dynamic slip boundary condition for wall-modeled large-eddy simulation. *Physics of Fluids*, 26, 2014.

[85] W. Gao, W. Zhang, W. Cheng, and R. Samtaney. Wall-modelled large-eddy simulation of turbulent flow past airfoils. *Journal of Fluid Mechanics*, 873, 2019.

[86] 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*, 3, 2016.

[87] C. Brehm, C. Hader, and H. Fasel. A locally stabilized immersed boundary method for the compressible Navier–Stokes equations. *Journal of Computational Physics*, 295, 2015.

[88] A. Harten, B. Engquist, S. Osher, and S. R. Chakravarthy. Uniformly high order accurate essentially non-oscillatory schemes iii. Contractor Report NASA–CR–178101, ICASE-86-22, Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, VA, 1986. Supported by NASA grants NAS1-17070, NAS1-18107; NSF DMS-85-03294; ARO DAAG29-85-K-0190.

[89] X. Liu, S. Osher, and T. Chan. Weighted essentially non-oscillatory schemes. *Journal of Computational Physics*, 115(1), 1994.

[90] C. Brehm. On consistent boundary closures for compact finite-difference weno schemes. *Journal of Computational Physics*, 334, 2017.

[91] J. Deardorff. A numerical study of three-dimensional turbulent channel flow at large Reynolds numbers. *Journal of Fluid Mechanics*, 41(2), 1970.

[92] U. Schumann. Subgrid scale model for finite difference simulations of turbulent flows in plane channels and annuli. *Journal of Computational Physics*, 18(4), 1975.

[93] S. Kawai and J. Larsson. Wall-modeling in large eddy simulation: Length scales, grid resolution, and accuracy. *Physics of Fluids*, 24(1), 2012.

[94] B. Mettu and P. Subbareddy. Wall modeled LES of compressible flows at non-equilibrium conditions. *2018 Fluid Dynamics Conference*, (Llm), 2018.

[95] B. Mettu and P. Subbareddy. Modeling non-equilibrium effects in wall modeled LES of high-speed flows. 2019.

[96] H. Xu, X. Yang, and P. Milani. Assessing Wall-Modeled Large-Eddy Simulation for Low-Speed Flows with Heat Transfer. *AIAA Journal*, 59(6), 2021.

[97] H. Owen, G. Chrysokentis, M. Avila, D. Mira, G. Houzeaux, R. Borrell, J. Cajas, and O. Lehmkuhl. Wall-modeled large-eddy simulation in a finite element framework. *International Journal for Numerical Methods in Fluids*, 92(1), 2020.

[98] R. Zangeneh. Wall-modeled Large-eddy Simulation of Hypersonic Turbulent Boundary-layers. In *AIAA Scitech 2021 Forum*, AIAA SciTech Forum. American Institute of Aeronautics and Astronautics, 2021.

[99] M. Adler, D. Gonzalez, L. Riley, and D. Gaitonde. Wall-Modeling Strategies for Large-Eddy Simulation of Non-Equilibrium Turbulent Boundary Layers. In *AIAA Scitech 2020 Forum*, AIAA SciTech Forum. American Institute of Aeronautics and Astronautics, 2020.

[100] Y. Kuwata and K. Suga. Wall-modeled large eddy simulation of turbulent heat transfer by the lattice Boltzmann method. *Journal of Computational Physics*, 433, 2021.

[101] F. Frankl and V. Voishel. Turbulent Friction in the Boundary Layer of a Flat Plate in a Two-Dimensional Compressible Flow at High Speeds. *NACA Technical Memorandum NACA-TM-1053*, 1943.

[102] F. Smith and R. Harrop. Turbulent Friction in the Boundary Layer of a Flat Plate in a Two-Dimensional Compressible Flow at High Speeds. *Royal Aircraft Establishment Technical Note 1759*, 1946.

[103] E. R. van Driest. Turbulent boundary layer in compressible fluids. *Journal of the Aeronautical Sciences*, 18(3), 1951.

[104] P. Volpiani, P. Iyer, S. Pirozzoli, and J. Larsson. Data-driven compressibility transformation for turbulent wall layers. *Physical Review Fluids*, 5, 2020.

[105] A. Patel, R. Pecnik, J. Peeters, S. Hickel, and M. Moghadam. Turbulence modulation by variable density and viscosity. *Center for Turbulence Research Annual Research Briefs*, 2016.

[106] P. Chen, Y. Lv, H. Xu, Y. Shi, and X. Yang. LES wall modeling for heat transfer at high speeds. *Physical Review Fluids*, 7(1), 2022.

[107] P. Iyer and M. Malik. Analysis of the equilibrium wall model for high-speed turbulent flows. *Physical Review Fluids*, 4, 7 2019.

[108] S. Bocquet, P. Sagaut, and J. Jouhaud. A compressible wall model for large-eddy simulation with application to prediction of aerothermal quantities. *Physics of Fluids*, 24(6), 2012.

[109] W Cabot and P. Moin. Approximate Wall Boundary Conditions in the Large-Eddy Simulation of High Reynolds Number Flow. *Flow, Turbulence and Combustion*, 63(1), 2000.

[110] G. Park and P. Moin. An improved dynamic non-equilibrium wall-model for large eddy simulation. *Physics of Fluids*, 26(1), 2014.

[111] X. I.A. Yang, J. Urzay, S. Bose, and P. Moin. Aerodynamic heating in wall-modeled large-eddy simulation of high-speed flows. *AIAA Journal*, 56(2), 2018.

[112] Y. Chen and C. Scalo. Trapped waves in supersonic and hypersonic turbulent channel flow over porous walls. *Journal of Fluid Mechanics*, 920, 2021.

[113] L. Szirmay-Kalos. Higher order automatic differentiation with dual numbers. *Periodica Polytechnica Electrical Engineering and Computer Science*, 65, 2021.

[114] R. Peón-Escalante, K. Cantún-Avila, O. Carvente, A. Espinosa-Romero, and Fe nu nuri. A dual number formulation to efficiently compute higher order directional derivatives. *Journal of Computational Science*, 76, 2024.

[115] J. Martins, P. Sturdza, and J. Alonso. The complex-step derivative approximation. *ACM Transactions on Mathematical Software*, 29, 2003.

[116] V. Pasquariello, Y. Bunk, S. Eberhardt, P. Huang, J. Matheis, M. Ugolotti, and S. Hickel. Gpu-accelerated simulations for evtol aerodynamic analysis. In *AIAA SciTech 2023 Forum*.

[117] F. Wei, L. Jin, J. Liu, F. Ding, and X. Zheng. GPU acceleration of a 2D compressible Euler solver on CUDA-based block-structured Cartesian meshes. *Journal of the Brazilian Society of Mechanical Sciences and Engineering Article*, 42, 2020.

[118] J. Davis, J. Shafner, D. Nichols, N. Grube, P. Martin, and A. Bhatele. Porting a computational fluid dynamics code with amr to large-scale gpu platforms. In *2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2023.

[119] W. Xue, C. Jackson, and C. Roy. An improved framework of gpu computing for cfd applications on structured grids using openacc. *Journal of Parallel and Distributed Computing*, 156, 2021.

[120] D. Jude, J. Sitaraman, and A. Wissink. An octree-based, cartesian cfd solver for helios on cpu and gpu architectures. In *AIAA Scitech 2021 Forum*.

[121] W. van Noordt. spade. `https://github.com/wvannoordt/spade`, 2023.

[122] S. Gottleib and C.W. Shu. Total-Variation-Diminishing Runge-Kutta Schemes. *Mathematics of Computation*, 67, 1998.

[123] K. Kim, S. Lee, and M. Kuk Yoon. Warped-preexecution: A gpu pre-execution approach for improving latency hiding. *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2016.

[124] M. Andersch, J. Lucas, M. Álvarez-Mesa, and B. Juurlink. On latency in gpu throughput microarchitectures. *2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2015.

[125] H. Wong, M. Papadopoulou, M. Sadooghi-Alvandi, and A. Moshovos. Demystifying gpu microarchitecture through microbenchmarking. *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2010.

[126] T. Dekker. A floating-point technique for extending the available precision. *Numerische Mathematik*, 18(3), 1971.

[127] S. Ganju, W. van Noordt, and C. Brehm. Wall-model informed ghost-cell reconstruction for immersed boundary large-eddy simulations. *Journal of Computational Physics (in review)*, 2024.

[128] S. Ganju, W. van Noordt, and C. Brehm. An Immersed Boundary Wall-Modeled Large-Eddy Simulation Approach for Low Speed Turbulent Flows. In *AIAA Aviation 2022 Forum*. 2022.

[129] W. van Noordt, J. McQuaid, J. Larsson, and C. Brehm. GPU-Accelerated Simulations of Turbulent Flows with the Immersed Boundary Method. In *AIAA Aviation 2024 Forum*. 2024.

[130] M. Lee and R. Moser. Direct numerical simulation of turbulent channel flow up to $Re_\tau \approx 5200$. *Journal of Fluid Mechanics*, 774, 2015.

[131] C. Brehm. Novel Immersed Interface Method for Solving the Incompressible Navier-Stokes Equations, 2011.

[132] H. Guillard and B. Nkonga. *On the Behaviour of Upwind Schemes in the Low Mach Number Limit: A Review*, volume 18. Elsevier B.V., 1 edition, 2017.

[133] V. Schmitt and F. Charpin. Pressure distributions on the onera-m6-wing at transonic Mach numbers. *Experimental Data Base for Computer Program Assessment. Report of the Fluid Dynamics Panel Working Group 04, AGARD AR-138*, 1979.

[134] P. Subbareddy and G. Candler. DNS of transition to turbulence in a hypersonic boundary layer. *41st AIAA Fluid Dynamics Conference and Exhibit*, 2011.

[135] L. Fu, M. Karp, S. Bose, P. Moin, and J. Urzay. Shock-induced heating and transition to turbulence in a hypersonic boundary layer. *Journal of Fluid Mechanics*, 909, 2020.

[136] J. McQuaid., A. Zibitsker, A. Martin, and C. Brehm. Heat Flux Predictions for High Speed Flows with an Immersed Boundary Method. In *AIAA Aviation 2021 Forum*, AIAA Aviation Forum. 2021.

[137] J. McQuaid. *Development of an Automated Volume Mesh Generation CFD Framework for Hypersonic Heat Flux Predictions*. PhD thesis, University of Maryland, College Park, Maryland, USA, 2024.

[138] S. Ganju, W. van Noordt, and C. Brehm. Progress in the Development of an Immersed Boundary Viscous-Wall Model for 3D and High-SpeedFlows. 2021.

[139] J. McCready, C. Hoppe, E. Johnson, J. Edwards, and V. Narayanaswamy. Mach 4 Performance of a Hypersonic Streamtraced Inlet – Part 2: Computational Results. *AIAA Science and Technology Forum and Exposition, AIAA SciTech Forum 2022*, 2022.

[140] C. Mullen and H. Reed. Analysis of the bolt flight geometry at off-nominal conditions. In *AIAA Scitech 2021 Forum*.

[141] H. Kostak and R. Bowersox. Preflight ground test analyses of the boundary layer transition (bolt) flight geometry. *Journal of Spacecraft and Rockets*, 58(1), 2021.

[142] B. Wheaton, D. Berridge, T. Wolf, R. Stevens, and B. McGrath. Boundary layer transition (bolt) flight experiment overview. In *2018 Fluid Dynamics Conference*.