
# A Second-Order Accurate Mach-Uniform Scheme for the Unsteady Compressible Navier-Stokes Equations on Orthogonal Unstructured Grids

David Jones

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

June 2023

**DECLARATION**

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed .....[redacted]..... (candidate)

Date .....29/6/23.....

**STATEMENT 1**

This thesis is the result of my own investigations, except where otherwise stated. Where correction services have been used, the extent and nature of the correction is clearly marked in a footnote(s). Other sources are acknowledged giving explicit references. A bibliography is appended.

Signed ...[redacted]..... (candidate)

Date .....29/6/23.....

**STATEMENT 2**

I hereby give consent for my thesis, if accepted, to be made available for electronic sharing and for inter-library loan (subject to the law of copyright) and (b) for the title and summary to be made available to outside organisations.

Signed [redacted]..... (candidate)

# Abstract

The disparity in the physics of high speed and low speed fluid flows has lead to the development of computational fluid dynamics methods for these flow conditions traditionally being considered separately. As a result there are many well-developed, efficient, and accurate schemes available that perform well in one case, but not the other. Nonetheless, there are many flow cases of engineering interest that contain both types of flow within the same domain. Examples include rotorcraft in hover, gas turbines at start-up, and propellers at take-off. There is therefore a need for a solver that performs well across a wide range of Mach numbers.

The development of a solver that remains efficient and accurate for flows for all speeds up to the high supersonic range is presented and discussed. The solver uses a staggered, segregated pressure correction approach, characteristic of low speed solvers. The pressure is corrected so as to enforce conservation of energy, providing strong coupling between the pressure and other flow variables at low speeds without losing efficiency at high speeds.

The modification of an existing two-dimensional Mach-uniform inviscid solver is first discussed and extensively validated. Then, the extension of the solver to viscous flow is considered. Finally the solver is extended to the case of unsteady flows with moving boundaries, which is achieved through the use of the Chimera method. The solver is shown to be efficient at all speeds in all three cases, and grid refinement studies are performed which demonstrate the second-order spatial accuracy of the solver. Additionally, a time step refinement study is performed for the unsteady solver, confirming that it achieves second-order temporal accuracy.

Recent advances in the generation and optimisation of unstructured grids have provided good quality orthogonal grids even for relatively complex geometries. The solver makes use of grids produced using these methods, resulting in a pressure correction that is both simple and robust.

# Contents

# Chapter 1

# Introduction

## 1.1 Background

Since its inception in the latter half of the twentieth century, computational fluid dynamics (CFD) has become an indispensible part of the design process in the aerospace, ground vehicle, and maritime industries, as well as finding applications in medicine, weather forecasting, oceanography, and computer graphics. This rapid growth has been enabled by an exponential increase in the available computing power, and this trend is expected to continue for some time [Sha20].

Historically, aerodynamic design projects were based largely on empirical investigations in wind tunnels and on test rigs. The process consisted of building a model from a design, running the tests, and using the results to guide modifications to the design. The process was repeated iteratively until an acceptable design was reached. Building, running, and maintaining the necessary facilities for executing a design process of this type required significant financial investment, and there were signficant development bottlenecks due to the serial nature of the testing process and the delay whilst models were fabricated and instrumented. In addition the instrumentation of the model alters the flow field, introducing some error into the results. In some cases this instrumentation error can be significant, such as when measuring the pressure at a point between stages in a compressor or turbine.

Including CFD in the design process has enabled many of the difficulties inherent in physical testing to be overcome

- For many applications good quality CFD simulations can be obtained relatively cheaply on a workstation or small cluster.

- If a reasonable amount of computational resources is available, several simulations can be run in parallel to investigate the effect of different flow conditions or design modifications. Even on smaller systems running CFD simulations in serial, these investigations can be completed much more quickly than would be possible with testing physical models in a wind tunnel.
- CFD studies provide the values of the flow variables across the whole flow field rather than at discrete points.
- Instrumentation error is avoided.

On the other hand, CFD cannot completely replace physical experiment. Developing a practical CFD model requires making numerous assumptions. It is not always clear that these assumptions are well-founded for a certain flow case, and so physical testing is needed to validate results from the CFD model and to guide the selection of model parameters.

There is no "universal" CFD solver which is optimal for all flow cases. Instead, different design choices made during development lend a given scheme to a certain group of flow cases at the expense of reduced efficiency and accuracy in others. One such design choice involves the Mach number, defined as the ratio of the flow velocity to the sonic velocity, $M = \frac{\|\boldsymbol{u}\|_2}{a}$. According to whether $M$ is greater than or less than some critical value (which is generally taken to be in the range 0.25–0.33), fluid flows can be categorised as high speed or low speed. In the former case the density, flow velocity, pressure, and temperature are all thermodynamic variables, and the requirement that mass, momentum, and energy are conserved yields a tightly coupled system of partial differential equations. In the latter case the fluid density is found to become approximately constant. The assumption that variations in the fluid density can be neglected then allows the governing equations to be drastically simplified, but also has profound implications for the physics expressed. Most importantly the pressure loses its status as a thermodynamic variable, and disturbances are found to propagate through the flow field instantaneously rather than at the sonic velocity as is the case in high speed flow. It is therefore possible to develop a highly efficient solver for low speed flows by neglecting density changes, but this solver will produce increasingly inaccurate results as the Mach number is raised above the critical Mach number. Conversely it is possible to develop an efficient and accurate solver for high speed flows by allowing the density to change and making use of the finite speed at which disturbances propagate. However the solver obtained in this way will be less efficient

at simulating low speed flows as negligible density changes are needlessly computed, and the significant difference in the flow physics will generally yield stiff systems of equations which can neither be solved efficiently nor accurately, if at all.

There are many flows of engineering interest that combine both high speed and low speed flows. For example, a rotorcraft in hover or low speed flight will have low speed flow over most of the domain, but the flow through the rotor will be high speed. Other examples include the flow through combustors in gas turbine engines, propeller aircraft during slow flight, and jet aircraft from start-up to take-off and climb. For these cases, CFD solvers designed for either high speed or low speed flows will not perform well in every region of the flow domain. A solver capable of performing well across a wide range of Mach numbers, from close to zero to the high supersonic range, is therefore required.

Attempts towards this goal have thus far been only partially successful. One popular methodology involves the extension of codes intended for computing high speed flows to the low speed case by preconditioning the governing equations to alleviate the increasing stiffness that occurs as the Mach number is reduced towards zero. The determination of a suitable preconditioning matrix for this purpose is not at all trivial and the subject of ongoing research [WSW02]. In addition, the resulting method is not truly Mach-uniform since the governing equations still become stiff in the low Mach number limit. A second methodology – the methodology followed in this work – involves the extension of an incompressible scheme to the compressible case. In the low Mach number limit the original incompressible scheme is recovered, hence this methodology can be expected to perform better than the first methodology in this limit. A third methodology that has gained some attention since the beginning of the work decribed in ths thesis is the extension of compressible solvers into the incompressible region through the use of complex convective flux reconstruction schemes such as the so-called rotated-hybrid Riemann solvers described in [HF23]. These methods do show promise, but are again likely to lose efficiency in the very-low-speed regime.

## 1.2 Objectives and Scope

The objective of the thesis is the development of a solver for the unsteady compressible Navier-Stokes equations that is accurate and efficient across a large range of Mach numbers, from incompressible flows to those at the boundary between the supersonic and hypersonic regimes. The development is broken down into three stages.

1. An existing steady inviscid Mach-uniform solver is modified to make use of the

special grid requirements imposed (these are discussed in the next chapter).

2. The solver is extended to allow the simulation of steady viscous flows.

3. The solver is extended to allow the simulation of unsteady viscous flows. This includes both those unsteady flows with fixed geometries and those with variable geometries.

As each of the above milestones is reached. Numerical testing is conducted to ensure the correctness, accuracy, and efficiency of the solver. In particular the solver is required to be second-order spatially accurate at each stage. In addition, the unsteady solver is required to be second-order temporally accurate.

At the time the work was undertaken, the generation of good quality two-dimensional grids meeting the grid requirements discussed in the next chapter was well-understood. However, the extension of these methods to complex three-dimensional geometries had not yet been completed. As a result, only two-dimensional geometries will be considered in the main body of the thesis. However the solver has a form that naturally lends itself to extension to three dimensions in a straightforward way, and so the possible future three-dimensional extension of the solver is very briefly considered in the final chapter.

## 1.3 Thesis Structure

The remainder of the thesis follows the chronological progression of the development of the solver, and is divided into six chapters. In addition, two appendices are included.

In Chapter 2, the theoretical foundations for the solver are developed. First, the integral form of the governing equations of unsteady compressible flow are derived from the basic conservation principles. Ancilliary equations are then discussed, including the ideal gas equation of state which is used to close the system of governing equations. Next, the special grid properties possessed by the grids to be used by the solver are described. Finally, some grid optimisation techniques with relevance to the development of the solver are briefly discussed.

In Chapter 3, the practical development of the solver begins. An existing steady inviscid Mach-uniform solver is discussed and adapted for the special grid requirements detailed in Chapter 2.

In Chapter 4, further development of the inviscid solver is undertaken. This includes consideration of time step selection, boundary conditions, and convergence acceleration

and testing, as well as the modification of convective flux discretisation and gradient limiting techniques for use with the solver. A grid refinement study is completed, which demonstrates the second-order spatial order of accuracy of the solver. Finally, a number of test cases are studied, to examine the correctness, accuracy, and convergence properties of the solver.

In Chapter 5, the solver is extended to the steady viscous case. This includes the addition of further terms to the governing equations, the reconstruction of the viscous and heat fluxes, the modification of the time step selection, and the implementation of new boundary conditions. The grid refinement study is then repeated, to examine the effect of these modifications on the spatial order of accuracy of the solver. Finally a number of test cases are presented and discussed.

In Chapter 6, the solver is extended to the unsteady viscous case. In order to achieve this, significant changes to the structure of the solver are required. The modification of the conservation principle to the case of variable geometry flows is discussed, and the dual time-stepping and chimera methods are motivated and presented. A pair of refinement studies are performed, to assess the spatial and temporal order of accuracy of the solver. Finally, test cases are used to demonstrate the properties of the finished solver.

In Chapter 7, the development of the solver is reviewed and the solver is compared against the objectives presented in this chapter. Finally, a number of possible further developments for the solver are suggested and briefly considered.

In Appendix A, some of the design choices mentioned in the main body of the thesis are discussed in more detail, including consideration of alternative options.

In Appendix B, some aspects of the code structure and implementation are briefly discussed.

# Chapter 2

# Governing Equations, Literature Review, and Grid Requirements

## 2.1 The Governing Equations

### 2.1.1 The Continuity Equation

The continuity equation is derived from the basic physical principle of conservation of mass, that mass cannot be created nor destroyed, only transported from one place to another. Consider the control volume of Figure 2.1. Conservation of mass applied to this control volume states that the rate of change of total mass of fluid inside the

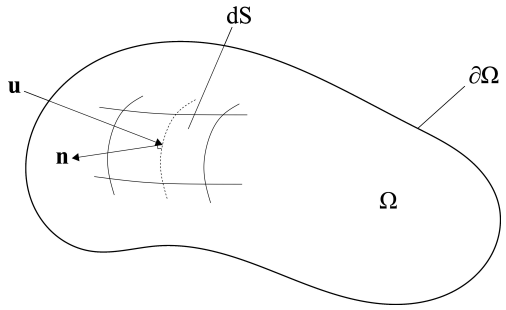



Figure 2.1: An arbitrary control volume in a compressible fluid flow for the derivation of the continuity equation.

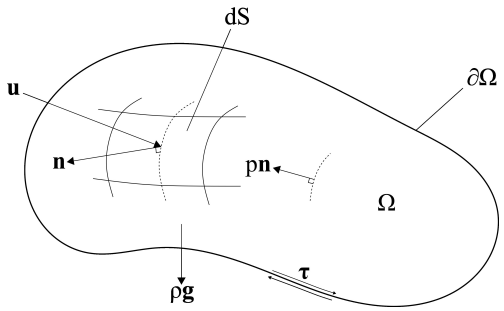



Figure 2.2: An arbitrary control volume in a compressible fluid flow for the derivation of the momentum and energy equations.

control volume is equal to the net rate at which mass is flowing into the control volume

$$\frac{\partial}{\partial t}\left\{\begin{array}{c}\text{Total mass in}\\ \text{control volume}\end{array}\right\} = \left\{\begin{array}{c}\text{Net rate of}\\ \text{flow of mass}\\ \text{into control volume}\end{array}\right\} \tag{2.1}$$

Each elemental volume $d\Omega$ has mass $\rho d\Omega$, where $\rho$ is the fluid density. The total mass of fluid inside the control volume at a given time is obtained by integrating over the control volume

$$\int_{\Omega} \rho d\Omega \tag{2.2}$$

Through an elemental patch $dS$ of the surface of the control volume, the rate of mass of fluid flow into the control volume is $-\rho\left(\boldsymbol{u}\cdot\boldsymbol{n}\right)dS$ for fluid velocity $\boldsymbol{u}$ and outwards unit normal $\boldsymbol{n}$. The net rate at which mass is flowing into the control volume is found by integrating over the surface of the control volume

$$-\oint_{\partial\Omega} \rho\left(\boldsymbol{u}\cdot\boldsymbol{n}\right)dS \tag{2.3}$$

Hence the continuity equation is

$$\frac{\partial}{\partial t}\int_{\Omega} \rho d\Omega + \oint_{\partial\Omega} \rho\left(\boldsymbol{u}\cdot\boldsymbol{n}\right)dS = 0 \tag{2.4}$$

### 2.1.2 The Momentum Equation

The momentum equation results from the application of Newton's second law to an arbitrary control volume. Referring to Figure 2.2, this can be stated as

$$\frac{\partial}{\partial t}\left\{\begin{array}{c}\text{Total}\\ \text{momentum in}\\ \text{control volume}\end{array}\right\}=\left\{\begin{array}{c}\text{Net rate of flow}\\ \text{of momentum into}\\ \text{control volume}\end{array}\right\}+\sum\left\{\begin{array}{c}\text{Forces on}\\ \text{control volume}\end{array}\right\} \tag{2.5}$$

Each elemental volume $d\Omega$ has momentum $\rho \boldsymbol{u} d\Omega$, hence the total momentum in the control volume at a given time is

$$\int_{\Omega} \rho \boldsymbol{u} d\Omega \tag{2.6}$$

Through each elemental patch $dS$ of the surface of the control volume, the rate of flow of momentum into the control volume is $-\rho\left(\boldsymbol{u}\cdot\boldsymbol{n}\right)\boldsymbol{u}ds$. Therefore, the net rate of flow of momentum into the control volume is

$$-\oint_{\partial\Omega} \rho\left(\boldsymbol{u}\cdot\boldsymbol{n}\right)\boldsymbol{u}dS \tag{2.7}$$

There are three forces acting on the control volume

1. Gravitational force $\rho\boldsymbol{g}$ acting on each elemental mass $\rho d\Omega$.
2. Pressure $-p\boldsymbol{n}ds$ acting on each elemental surface patch $dS$, where $p$ is the pressure.
3. Shear stress $\boldsymbol{\tau}\cdot\boldsymbol{n}dS$ acting on each elemental surface patch $dS$, where $\boldsymbol{\tau}$ is the viscous stress tensor. $\boldsymbol{\tau}$ can be defined as a linear map from a unit vector $\boldsymbol{c}$ to a vector giving the force per unit area on a surface perpendicular to $\boldsymbol{c}$ due to the action of viscous forces. $\boldsymbol{\tau}\cdot\boldsymbol{n} \coloneqq \boldsymbol{\tau}\left(\boldsymbol{n}\right)$ is then the normal force due to viscosity acting on the control volume surface at a given point. The viscous stress tensor is discussed further in Section 2.1.6.

The sum of forces acting on the control volume is therefore

$$\int_{\Omega} \rho\boldsymbol{g} - \oint_{\partial\Omega} p\boldsymbol{n}dS + \oint_{\partial\Omega} \boldsymbol{\tau}\cdot\boldsymbol{n}dS \tag{2.8}$$

and hence the momentum equation is

$$\frac{\partial}{\partial t}\int_{\Omega} \rho\boldsymbol{u}d\Omega + \oint_{\partial\Omega} \rho\left(\boldsymbol{u}\cdot\boldsymbol{n}\right)\boldsymbol{u}dS = \int_{\Omega} \rho\boldsymbol{g} - \oint_{\partial\Omega} p\boldsymbol{n}ds + \oint_{\partial\Omega} \boldsymbol{\tau}\cdot\boldsymbol{n}dS \tag{2.9}$$

Often the gravitational term is neglected, leaving

$$\frac{\partial}{\partial t}\int_{\Omega} \rho\boldsymbol{u}d\Omega + \oint_{\partial\Omega} \rho\left(\boldsymbol{u}\cdot\boldsymbol{n}\right)\boldsymbol{u}dS = -\oint_{\partial\Omega} p\boldsymbol{n}dS + \oint_{\partial\Omega} \boldsymbol{\tau}\cdot\boldsymbol{n}dS \tag{2.10}$$

### 2.1.3 The Energy Equation

The energy equation is a statement of the first law of thermodynamics, that energy can neither be created nor destroyed, only transported from one place to another or converted from one form to another. This can be written as

$$\frac{\partial}{\partial t}\left\{\begin{array}{c}\text{Total}\\ \text{energy in}\\ \text{control volume}\end{array}\right\} = \left\{\begin{array}{c}\text{Net rate of flow}\\ \text{of energy into}\\ \text{control volume}\end{array}\right\} + \left\{\begin{array}{c}\text{Rate of work}\\ \text{done on}\\ \text{control volume}\end{array}\right\} + \left\{\begin{array}{c}\text{Rate of heat}\\ \text{transferred to}\\ \text{control volume}\end{array}\right\} \tag{2.11}$$

In the case of the energy equation, the conserved quantity is the total energy $\rho E = \rho\left(e + \frac{1}{2}\boldsymbol{u}\cdot\boldsymbol{u}\right)$, where $e$ is the internal energy. Integrating over the control volume in Figure 2.2 as before yields the total energy in the control volume at a given time

$$\int_{\Omega} \rho E d\Omega \tag{2.12}$$

The rate of flow of total energy into the control volume through an elemental surface patch $dS$ is $-\rho E\left(\boldsymbol{u}\cdot\boldsymbol{n}\right)dS$, hence the net rate of flow of total energy into the control volume is

$$-\oint_{\partial\Omega} \rho E\left(\boldsymbol{u}\cdot\boldsymbol{n}\right)dS \tag{2.13}$$

The rate of work done on an elemental volume $d\Omega$ of the control volume by the gravitational force $\rho\boldsymbol{g}d\Omega$ is found by projecting the force in the direction of the velocity of the element

$$\rho\boldsymbol{g}\cdot\boldsymbol{n}d\Omega \tag{2.14}$$

Hence the rate of work done on the control volume $\Omega$ by the gravitational force is

$$\int_{\Omega} \rho\boldsymbol{g}\cdot\boldsymbol{n}d\Omega \tag{2.15}$$

Similarly, the rate of work done on an elemental surface patch of the control volume $ds$ by an arbitrary surface force $\boldsymbol{f}_s ds$ is found by projecting the force in the direction of the velocity of the surface element

$$\boldsymbol{f}_s\cdot\boldsymbol{n}dS \tag{2.16}$$

Hence the rate of work done on the control volume $\Omega$ by the pressure and viscous forces is

$$-\oint_{\partial\Omega} p\,(\boldsymbol{n}\cdot\boldsymbol{u})\,dS + \oint_{\partial\Omega} (\boldsymbol{\tau}\cdot\boldsymbol{u})\cdot\boldsymbol{n}dS \tag{2.17}$$

Finally, there are two forms of heat transfer between the control volume and its surroundings

1. A surface flux due to thermal conduction. Through an elemental surface patch $dS$, the rate of heat transfer into the control volume by conduction is given by Fourier's law of heat transfer

$$k\,(\nabla T\cdot\boldsymbol{n})\,dS \tag{2.18}$$

2. A volume source due to volumetric heating. For an elemental volume $d\Omega$, the rate of heat transfer to the control volume via volumetric heating is written

$$\dot{q}d\Omega \tag{2.19}$$

Hence the rate of heat transfer to the control volume by the surface flux and volume source is

$$\oint_{\partial\Omega} k\,(\nabla T\cdot\boldsymbol{n})\,dS + \int_{\Omega}\dot{q}d\Omega \tag{2.20}$$

The energy equation is therefore

$$\begin{aligned}\frac{\partial}{\partial t}\int_{\Omega}\rho E d\Omega + \oint_{\partial\Omega}\rho E\,(\boldsymbol{u}\cdot\boldsymbol{n})\,dS = \int_{\Omega}\rho\boldsymbol{g}\cdot\boldsymbol{n}d\Omega - \oint_{\partial\Omega} p\,(\boldsymbol{n}\cdot\boldsymbol{u})\,dS \\ + \oint_{\partial\Omega}(\boldsymbol{\tau}\cdot\boldsymbol{u})\cdot\boldsymbol{n}dS + \oint_{\partial\Omega} k\,(\nabla T\cdot\boldsymbol{n})\,dS + \int_{\Omega}\dot{q}d\Omega\end{aligned} \tag{2.21}$$

For the remainder of the thesis, the work done by the gravitational force and the heat transfer due to volumetric heating will be neglected, leaving

$$\begin{aligned}\frac{\partial}{\partial t}\int_{\Omega}\rho E d\Omega + \oint_{\partial\Omega}\rho E\,(\boldsymbol{u}\cdot\boldsymbol{n})\,dS = -\oint_{\partial\Omega} p\,(\boldsymbol{n}\cdot\boldsymbol{u})\,dS + \oint_{\partial\Omega}(\boldsymbol{\tau}\cdot\boldsymbol{u})\cdot\boldsymbol{n}dS \\ + \oint_{\partial\Omega} k\,(\nabla T\cdot\boldsymbol{n})\,dS\end{aligned} \tag{2.22}$$

This can be simplified further by introducing the *total enthalpy* $H = h + \frac{1}{2}\boldsymbol{u}\cdot\boldsymbol{u} = E + p/\rho$, where $h$ is the *enthalpy* [Bla01, p. 12]. This allows the second and third terms

of the energy equation to be combined, leaving

$$\frac{\partial}{\partial t}\int_{\Omega}\rho E d\Omega + \oint_{\partial\Omega}\rho H\left(\boldsymbol{u}\cdot\boldsymbol{n}\right)dS = \oint_{\partial\Omega}\left(\boldsymbol{\tau}\cdot\boldsymbol{u}\right)\cdot\boldsymbol{n}dS + \oint_{\partial\Omega}k\left(\nabla T\cdot\boldsymbol{n}\right)dS \tag{2.23}$$

### 2.1.4 The Ideal Gas Law and Additional Thermodynamic Relations

Before the energy equation can be transformed to the form used in the remainder of the thesis, a number of thermodynamic relations must be established. For the present work, the fluid will be assumed to be thermally and calorically perfect [RM92]. By definition, such a fluid satisfies the ideal gas law

$$p = \rho RT \tag{2.24}$$

where $R$ is the *characteristic gas constant*, as well as having constant *specific heat capacities* at constant volume $c_v$ and constant pressure $c_p$. These assumptions lead to a great number of other relations. Several of these are used in this thesis and are summarised below without much comment. Detailed discussions and derivations are available in the above reference.

The characteristic gas constant is the difference between the specific heats at constant pressure and at constant volume

$$R = c_p - c_v \tag{2.25}$$

and the enthalpy and internal energy can be written in terms of the temperature as

$$e = c_v T \tag{2.26}$$

$$h = c_p T \tag{2.27}$$

Therefore, the ideal gas law can be rewritten in either of the forms

$$\begin{aligned}\rho e &= \frac{p}{\gamma - 1}\\ \rho h &= \frac{\gamma p}{\gamma - 1}\end{aligned} \tag{2.28}$$

where $\gamma \coloneqq c_p/c_v$ is the *specific heat ratio*. Inserting these into the definitions of the

total energy and total enthalpy yields

$$\begin{aligned} \rho E &= \frac{p}{\gamma - 1} + \frac{\rho \boldsymbol{u} \cdot \boldsymbol{u}}{2} \\ \rho H &= \frac{\gamma p}{\gamma - 1} + \frac{\rho \boldsymbol{u} \cdot \boldsymbol{u}}{2} \end{aligned} \tag{2.29}$$

The sonic velocity $a$ can be calculated from [And10, p. 524]

$$a = \sqrt{\gamma R T} \tag{2.30}$$

or, using (2.25) and (2.27)

$$a = \sqrt{(\gamma - 1)\, h} \tag{2.31}$$

### 2.1.5 The Pressure-Based Form of the Energy Equation

In the solver presented here, the energy equation is used to update the pressure. The relations in the previous subsection can be used to transform the energy equation (2.23) into a pressure-based form. The result is

$$\begin{aligned} \frac{\partial}{\partial t} \int_\Omega \left\{ \gamma p + \frac{\gamma - 1}{2} (\rho \boldsymbol{u} \cdot \boldsymbol{u}) \right\} d\Omega + \oint_{\partial\Omega} \left\{ p + \frac{\gamma - 1}{2} (\rho \boldsymbol{u} \cdot \boldsymbol{u}) \right\} dS \\ = (\gamma - 1) \oint_{\partial\Omega} (\boldsymbol{\tau} \cdot \boldsymbol{u}) \cdot \boldsymbol{n} dS + (\gamma - 1) \oint_{\partial\Omega} k\, (\nabla T \cdot \boldsymbol{n})\, dS \end{aligned} \tag{2.32}$$

### 2.1.6 The Viscous Stress Tensor

In a two dimensional Cartesian coordinate system, the viscous stress tensor $\boldsymbol{\tau}$ is represented by the matrix

$$\boldsymbol{\tau} = \begin{bmatrix} \sigma_x & \tau_{xy} \\ \tau_{yx} & \sigma_y \end{bmatrix} \tag{2.33}$$

The physical interpretation of the components is depicted in Figure 2.3. The on-diagonal components $\sigma_x$ and $\sigma_y$ are the *direct* or *normal* stresses, and act to change the area of the elemental square without changing its interior angles. The off-diagonal components $\tau_{xy}$ and $\tau_{yx}$ are the *shear* stresses, and act to change the interior angles of the elemental square without changing its area. The viscous stress tensor is symmetric [DG95, p. 12], hence $\tau_{xy} = \tau_{yx}$.

Newton postulated that the shear stress is directly proportional to the velocity gradient. Later, Stokes derived the expressions for the components resulting from this

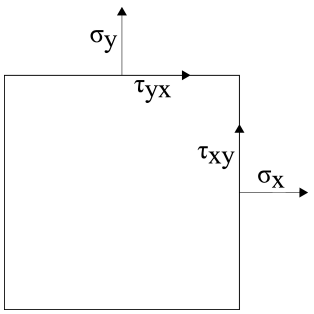



Figure 2.3: The physical significance of the components of the viscous stress tensor on an elemental square.

postulate [Bla01]

$$\begin{aligned} \sigma_x &= 2\mu\left(\frac{\partial u}{\partial x}\right) + \lambda\nabla\cdot\boldsymbol{u} \\ \sigma_y &= 2\mu\left(\frac{\partial v}{\partial y}\right) + \lambda\nabla\cdot\boldsymbol{u} \\ \tau_{xy} = \tau_{yx} &= \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \end{aligned} \tag{2.34}$$

where $\mu$ is the *dynamic viscosity*, $\lambda$ is the *second viscosity coefficient* or *bulk viscosity coefficient*, and $\nabla$ is the vector differential operator

$$\nabla \coloneqq \boldsymbol{i}\frac{\partial}{\partial x} + \boldsymbol{j}\frac{\partial}{\partial y} \tag{2.35}$$

where $\boldsymbol{i}$ and $\boldsymbol{j}$ are unit vectors in the $x$ and $y$ directions respectively.

Fluids for which Newton's postulate holds are referred to as *Newtonian*. Most commonly encountered engineering fluids are included in this category, such as water and air. Fluids that exhibit a more complex relationship between the shear stress and the velocity gradients (possibly including additional variables, such as time) are called *non-Newtonian*, and include blood and custard. Only Newtonian fluids will be considered in this thesis.

Stokes also introduced the postulate that the bulk viscosity coefficient is related to the dynamic viscosity coefficient by the simple relation

$$\lambda = -\frac{2\mu}{3} \tag{2.36}$$

For reasonable temperatures and pressures, this postulate is found to be in excellent agreement with experimental results [Bla01], and so will be assumed to hold in the

present work. The resulting expressions for the components of the viscous stress tensor are

$$\begin{aligned} \sigma_x &= 2\mu \left( \frac{\partial u}{\partial x} - \frac{1}{3} \nabla \cdot \boldsymbol{u} \right) \\ \sigma_y &= 2\mu \left( \frac{\partial v}{\partial y} - \frac{1}{3} \nabla \cdot \boldsymbol{u} \right) \\ \tau_{xy} = \tau_{yx} &= \mu \left( \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \end{aligned} \tag{2.37}$$

### 2.1.7 Dynamic Viscosity and Thermal Conductivity

The evaluation of the viscous and heat transfer terms in the momentum and energy equations requires the calculation of two additional thermodynamic variables, the dynamic viscosity $\mu$ and the thermal conductivity $k$.

For reasonable temperatures and pressures it is found experimentally that the dynamic viscosity and thermal conductivity of ideal gases are independent of pressure, and exhibit only a relatively weak dependence on the temperature. A common and reasonably accurate procedure is therefore to simply assume that the dynamic viscosity and thermal conductivity are constant throughout the domain, their values being obtained from tables of properties by assuming some reference state. This method is used in some of the numerical examples presented, as this is specified in the problem definition for the results taken from the literature for comparison. However, the solver developed in this thesis is intended primarily for flows with a wide variation in flow velocity, and therefore in temperature. It can therefore be expected that the exact dynamic viscosity and thermal conductivity have quite significant variations across the flow domain, and the assumption that they remain constant will be quite inaccurate. Better accuracy can be obtained for practically negligible additional computational effort through the use of a simple algebraic relation between the dynamic viscosity and the temperature due to Sutherland [Sut93], referred to as *Sutherland's law*. This can be written

$$\frac{\mu}{\mu_r} = \left( \frac{T}{T_r} \right)^{\frac{3}{2}} \frac{T_r + T_{s,\mu}}{T + T_{s,\mu}} \tag{2.38}$$

where $\mu_r$ and $T_r$ are the dynamic viscosity and temperature at some reference state and $T_{s,\mu}$ is a constant referred to as the *Sutherland temperature*. Sutherland's law is derived from the kinetic theory of gases and a simple model of inter-molecular force, so is valid for the standard assumptions of reasonable temperature and pressure.

For the present solver it turns out to be simpler to work with the enthalpy rather than the temperature, because the enthalpy is used in the calculation of other quantities

| | |
|---|---|
| $h_r$ | $2.745 \times 10^5\,\mathrm{J}$ |
| $\mu_r$ | $1.716 \times 10^{-5}\,\mathrm{Nsm}^{-2}$ |
| $k_r$ | $0.0241\,\mathrm{Wm}^{-1}\mathrm{K}^{-1}$ |
| $h_{s,\mu}$ | $1.116 \times 10^5\,\mathrm{J}$ |
| $h_{s,k}$ | $1.950 \times 10^5\,\mathrm{J}$ |

Table 2.1: Model constants used with Sutherland's law for the dynamic viscosity and thermal conductivity coefficients of air, from [Whi06, pp. 28–32]

and is therefore already computed. Sutherland's law is rewritten as a function of enthalpy using the relation (2.27), which results in

$$\frac{\mu}{\mu_r} = \left(\frac{h}{h_r}\right)^{\frac{3}{2}} \frac{h_r + h_{s,\mu}}{h + h_{s,\mu}} \tag{2.39}$$

where $h_{s,\mu}$ is referred to as the Sutherland *enthalpy* for dynamic viscosity.

By adjusting the model constants, an analgous expression is found that provides a reasonably accurate model for the thermal conductivity [Whi06, p. 30]

$$\frac{k}{k_r} = \left(\frac{h}{h_r}\right)^{\frac{3}{2}} \frac{h_r + h_{s,k}}{h + h_{s,k}} \tag{2.40}$$

This formula relies on the assumption that the Prandtl number (the ratio of momentum diffusion to thermal diffusion), defined by (5.9), is constant. As discussed in Section 5.2, this assumption is satisfied for an ideal gas at temperatures that are not too high

For air, suitable values for the model constants are shown in Table 2.1. A comparison between the predictions obtained from Sutherland's law using these constants and experimental values obtained from tables is shown in Figure 2.1. Clearly, Sutherland's law provides very good agreement with experiment over most of the range shown, although the predicted thermal conductivity is slightly too low at higher temperatures. A more accurate model might therefore be required when simulating high temperature flows such as those through combustion chambers and high pressure turbines in gas turbine engines.

### 2.1.8 The Governing Equations for Inviscid Flow

When modelling inviscid flow, an obvious simplification to the governing equations is that the terms involving the viscous stress tensor $\boldsymbol{\tau}$ can be neglected. Although not immediately obvious, the heat flux terms also drop out of the energy equation for

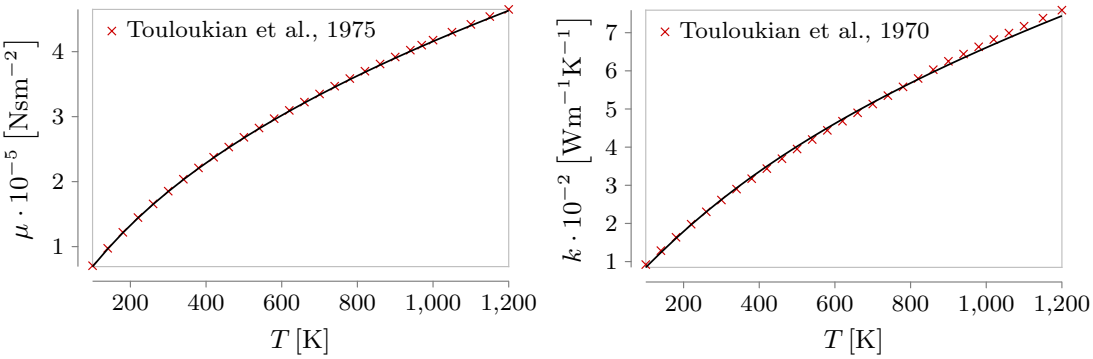



Figure 2.4: Plots of Sutherland's law predictions for the dynamic viscosity (left) and thermal conductivity (right) of air, compared to experimental data from [TSH75] and [TLS70] respectively.

inviscid flow. To see why this is the case, refer to the dimensionless form of the viscous energy equation (5.10). The heat flux term carries a factor of the reciprocal of the Reynolds number (the ratio of inertial to viscous forces, defined by (5.4)) multiplied by the Prandtl number. As mentioned previously, the Prandtl number can be assumed constant. In the limit as the viscous forces go to zero, then, the heat flux terms also go to zero and are therefore negligible in inviscid flow.

The governing equations of compressible inviscid flow are therefore

$$\begin{aligned} \frac{\partial}{\partial t}\int_\Omega \rho d\Omega + \oint_{\partial\Omega} \rho\left(\boldsymbol{u}\cdot\boldsymbol{n}\right) dS &= 0 \\ \frac{\partial}{\partial t}\int_\Omega \rho\boldsymbol{u} d\Omega + \oint_{\partial\Omega} \rho\left(\boldsymbol{u}\cdot\boldsymbol{n}\right)\boldsymbol{u} dS &= -\oint_{\partial\Omega} p\boldsymbol{n} dS \\ \frac{\partial}{\partial t}\int_\Omega \left\{\gamma p + \frac{\gamma-1}{2}\left(\rho\boldsymbol{u}\cdot\boldsymbol{u}\right)\right\} d\Omega + \oint_{\partial\Omega} \left\{p + \frac{\gamma-1}{2}\left(\rho\boldsymbol{u}\cdot\boldsymbol{u}\right)\right\} dS &= 0 \end{aligned} \tag{2.41}$$

## 2.2 Grid Layout and Requirements

In order to increase the accuracy and efficiency of the solver presented in the remainder of this thesis, certain grid requirements are imposed. In this section these requirements are described, and methods for achieving satisfactory grids are briefly discussed.

Typically, flow solvers for incompressible flows use a *staggered* approach, whereby scalar flow variables such as pressure and density are stored at the cell centroids whilst the normal flow velocity is stored at the edge centres [HW65]. This flow variable layout prevents the phenomenon known as *odd-even decoupling* [Hir07, p. 631], which can prevent convergence or cause convergence to an unphysical solution. This decoupling

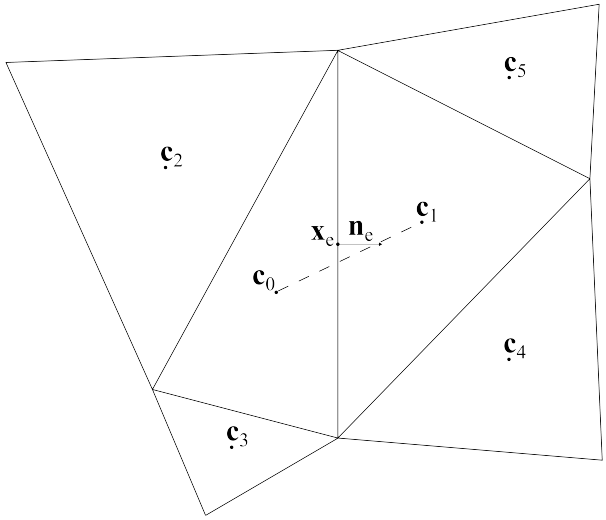



Figure 2.5: Example grid layout for a general staggered scheme with scalar flow variables stored at the cell centroids $\boldsymbol{c}_i$.

does not occur in compressible flow due to the strong density-flow velocity coupling. However, since the solver developed in the present work needs to be able to accurately compute incompressible flows, the staggered flow variable layout is adopted. This is discussed further in the next chapter.

An example grid layout for use with a staggered scheme is shown in Figure 2.5. It is frequently required to approximate the normal gradient of a cell centre scalar $\varphi$ at the centre of the edges. Referring to Figure 2.5, a first approximation would be the two-point difference

$$\left.\frac{\partial \varphi}{\partial \boldsymbol{n}_e}\right|_e \approx \frac{\varphi_1 - \varphi_0}{\|\boldsymbol{c}_1 - \boldsymbol{c}_0\|_2} \tag{2.42}$$

From the figure, three sources of error in this approximation are clear. Letting $\ell_{01}$ denote the line from $\boldsymbol{c}_0$ to $\boldsymbol{c}_1$, these are

1. $\ell_{01}$ does not pass through $\boldsymbol{x}_e$, hence the derivative is being approximated at the wrong point on the edge.

2. $\ell_{01}$ is not bisected by edge $e$, hence the derivative is being approximated at the wrong point on $\ell_{01}$.

3. $\ell_{01}$ is not parallel to $\boldsymbol{n}_e$, hence the derivative actually being approximated is the derivative with respect to a vector at a small angle to $\boldsymbol{n}_e$.

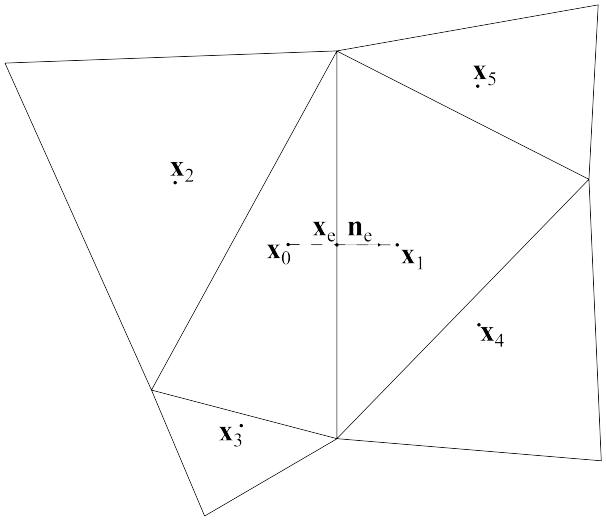



Figure 2.6: Example grid layout for a general staggered scheme with scalar flow variables stored at the cell circumcentres $\boldsymbol{x}_i$.

Due to these errors, the approximation (2.42) is only first-order accurate in general [FP02, p. 77]. On uniform grids all three errors vanish, and the approximation becomes second-order accurate. To achieve second-order accuracy on non-uniform grids, correction terms involving the values of $\varphi$ at the centroids of the surrounding cells $\{2, 3, 4, 5\}$ must be added. The result is a second-order approximation to the normal derivative with a six-point stencil. There is another source of error, due to the fact that the normal derivative is not constant along the line from $\boldsymbol{c}_0$ to $\boldsymbol{c}_1$. However this error is proportional to $1/\left\|\boldsymbol{c}_1 - \boldsymbol{c}_0\right\|_2^2$, and is therefore important only for schemes of third-order accuracy or higher.

In the solver presented in this thesis, the cell centre scalars are instead stored at the cell *circumcentres*. The result is a layout such as that shown in Figure 2.6 and a two-point normal derivative approximation

$$\left.\frac{\partial \varphi}{\partial \boldsymbol{n}_e}\right|_e \approx \frac{\varphi_1 - \varphi_0}{\left\|\boldsymbol{x}_1 - \boldsymbol{x}_0\right\|_2} \tag{2.43}$$

Clearly the sources of error enumerated above are zero with this layout, and the normal derivative approximation (2.43) is second-order accurate even on non-uniform grids. However, locating the scalar flow variables at the circumcentres introduces additional problems into the scheme. To see this, consider re-triangulating the same nodes used in the example layouts of Figure 2.5 and Figure 2.6. One possible result is shown in

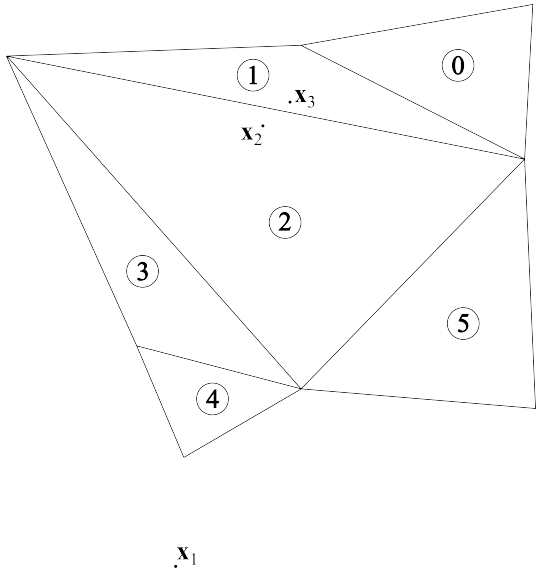



Figure 2.7: An alternative valid triangulation of the same vertices shown in Figure 2.5 and Figure 2.6, showing only the circumcentres $\boldsymbol{x}_i$ that are outside their corresponding triangle. Cell numbers are also shown

Figure 2.7.

Several issues are apparent in Figure 2.7

1. A cell need not contain its circumcentre, and in fact will not if (and only if) it possesses an interior angle of greater than 90°. This is the case for cells 1 and 3 in the figure. This is a serious problem numerically, because the value computed at the circumcentres depends on the values at the circumcentres of the cells immediately adjacent to the cell. If the cell circumcentre is not inside the cell, then the values of the scalars at that circumcentre are updated using information that is not local to the position of the circumcentre. For example, the flow variables at $\boldsymbol{x}_1$ in the figure depend on the values at the circumcentres of cells 0, 1, and 2, but not the cells actually near $\boldsymbol{x}_1$. This is likely to cause instability due to decoupling, as well as violating numerical conservation [Lan98, p. 187].

2. The vector from the circumcentre of one of the cells adjacent to an edge to the circumcentre of the other adjacent cell can be reversed relative to the vector between the corresponding cell centroids. This is the case for cells 2 and 3 in the figure.

3. The distance between the circumcentres of the two cells adjacent to an edge may be very small. This is again the case for cells 2 and 3 in the figure. This distance can even be zero, which occurs when the union of two adjacent cells is a square. The result is that the approximation (2.43) has a very small number (or zero) in the denominator, so that even extremely small changes to the values of $\varphi_0$ and $\varphi_1$ result in large (or infinite) changes to the value of the approximation. The result is solver instability that becomes increasingly more severe as the distances get progressively smaller.

The remainder of this section provides a brief review of the solution to these issues. Since grid generation is not the focus of this thesis, the focus will be on grid optimisation techniques that have specific consequences for the formulation of the solver.

### 2.2.1 The Delaunay Triangulation and Constrained Delaunay Triangulation

Comparing Figure 2.6 with Figure 2.7, it is clear that the problems described in the previous sub-section as well as the general quality of a grid are properties of the triangulation as well as of the node positions. A natural question is then how to obtain the "best" triangulation given a set of node positions. The answer obviously depends on the choice of quality metric. Inspection of Figure 2.7 suggests that mesh problems can be associated with triangles that contain small interior angles, and therefore the optimum triangulation should be that which maximises the minimum interior angle of the triangles belonging to the triangulation. The triangulation that possesses this property is the *Delaunay* triangulation [CDS13]. Given a set of node positions, the Delaunay triangulation is the (unique) triangulation that satisfies the *Delaunay criterion*. The Delaunay criterion is the requirement that every triangle belonging to a Delaunay triangulation has an inscribed circle containing none of the nodes.

The Delaunay triangulation is of limited use in CFD, because it does not respect the boundary definition. A related construction is the *constrained* Delaunay triangulation, which is the (unique) triangulation that satisfies the Delaunay criterion at all triangles that do not contain a boundary edge, with the remaining triangles completed simply by demanding that the boundary edges belong to the triangulation [FG08, p. 245]. This triangulation is the optimum (in the above sense) boundary-respecting triangulation for the interior cells, but not necessarily for the boundary cells. Reviews of modern approaches to constrained Delaunay grid generation and optimisation are available in

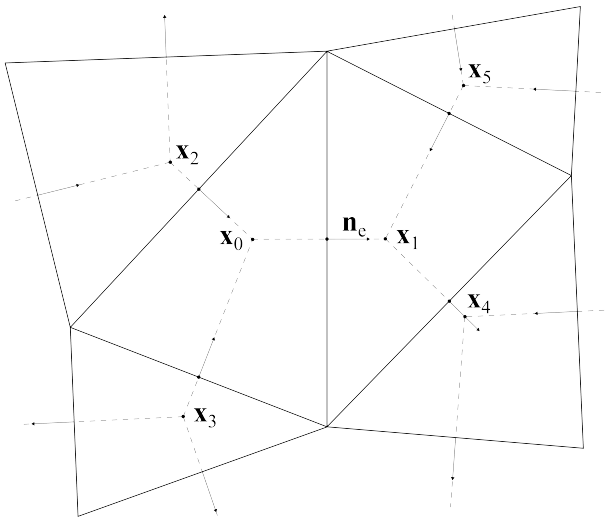



Figure 2.8: A Delaunay triangulation (solid lines) together with its dual Voronoi diagram (dashed lines)

[FG08] and [Lis10]. It is important to note that a unique constrained Delaunay triangulation always exists for a set of arbitrary points and edge constraints is guaranteed to exist in two-dimensional space [For18]. However, this is not the case for dimensions greater than two.

Each Delaunay triangulation has a dual *Voronoi diagram*, obtained by connecting the circumcentres of the Delaunay triangles [CDS13, p. 153]. An example is shown in Figure 2.8. As shown in the diagram, each Delaunay triangle corresponds to a unique Voronoi node, and each Delaunay edge corresponds to a unique Voronoi edge. These will be referred to as the *dual node* and *dual edge* of the triangle and edge respectively. Two key properties of the Delaunay-Voronoi pair are clear from the figure:

1. The Voronoi edges and Delaunay edges are mutually orthogonal.

2. If the Voronoi edges are extended to infinity in each direction then each Voronoi edge bisects the corresponding Delaunay edge.

The second-order accurate normal scalar derivative approximation (2.43) at a Delaunay edge is therefore obtained by using a finite difference approximation along the dual edge between the dual nodes of the cells adjacent to the edge.

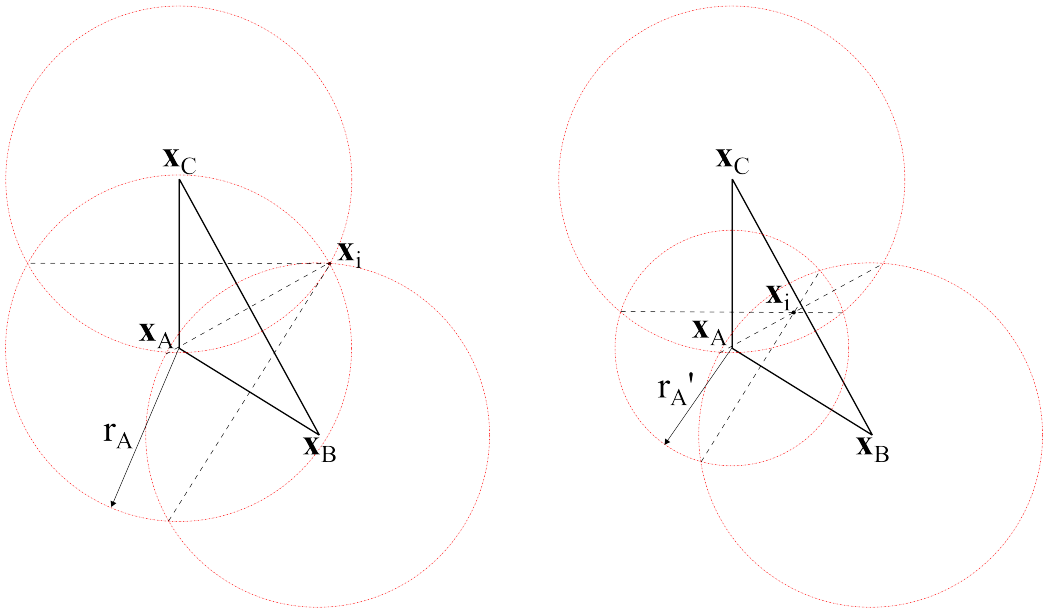



Figure 2.9: Left: Voronoi diagram in the neighbourhood of cell $i$, with dual node $\boldsymbol{x}_i$ well outside the cell. Right: The dual node has been moved inside cell $i$ by applying a weight to node $A$.

### 2.2.2 Mesh Optimisation

Several standard methods of optimisation are available. These techniques do not directly influence the design of the solver (other than making good quality grids available) and so are not discussed here. These techniques include Laplacian smoothing [FG08, p. 608], Lloyd's method [DG02], and edge flipping [FG08, p. 610].

One useful optimisation technique which does directly influence the development of the solver is the replacement of the cell circumcentres with alternative points at which to store the scalar flow variables. This is done through the use of the weighted Voronoi diagram [WHM17]. Initially, each node is assigned a weight of zero. For each cell, a circle is drawn centred at each vertex and passing through the cell circumcentre. For each Delaunay edge, a dual edge is placed by connecting the two points of intersection of the circles centred at the end nodes of the Delaunay edge. By construction the three dual edges of the three Delaunay edges of the cell will be the perpendicular bisector of their corresponding Delaunay edge, and will meet at the cell circumcentre. This is shown on the left of Figure 2.9. Since the node weights are all zero, this diagram coincides with the usual Voronoi diagram. Now let node $A$ have a non-negative

weight $0 \leq w_A < r_A$, and replace the circle centred at node $A$ with a circle of radius $r'_A \coloneqq r_A - w_A$. The dual edges are then constructed in the same way as before, by joining the points of intersection of each pair of node-centred circles. The result is shown on the right of Figure 2.9. Because the circle centred at node $A$ has been made smaller, the dual edges for edges $AB$ and $CA$ no longer bisect their associated edge, but they do remain perpendicular to them. Most importantly, if $w_A$ is large enough the point of intersection of the dual edges (the *dual node*) has been moved inside the element. If the scalars are stored at the dual nodes rather than the cell circumcentres, the problem noted earlier in this section with non-locality has been resolved. There are two drawbacks however. First, because the dual edges no longer pass through the midpoints of their associated Delaunay edges, the normal scalar gradient approximation (2.43) has been made less accurate. Also, because each node belongs to multiple triangles applying a non-zero weight to a node moves the dual nodes associated with several cells rather than just one. As a result, improving the quality of one cell by applying a node weight can reduce the quality of neighbouring cells. The optimisation of the node weights is therefore a global optimisation problem rather than a local one. If a grid has $N$ nodes then the optimisation problem is $2N$ dimensional for two dimensional grids and $3N$ dimensional for three dimensional grids. For larger grids this would be prohibitively expensive. For the present work, the reduced-order optimisation algorithm presented in [WHM13] was employed.

For every constrained Delaunay triangulation there exists a set of node weights such that every dual node lies inside its associated Delaunay cell. This must be the case since applying any positive weight to a node pulls the dual nodes corresponding to the attached cells towards the vertex itself, and therefore a large enough weight will pull all associated dual nodes onto the node itself, which lies inside every cell that has the node as a vertex by definition. This point is of little practical importance, however, as the quality of grids to which many large weights have been applied will be so poor as to preclude their use with the solver.

If the optimisation described in this section is used, the term "cell circumcentre" should be understood to mean "dual node corresponding to the cell" in the remainder of this thesis.

### 2.2.3 Cell Merging

Despite optimisation of the positions of the nodes and the cell centres, some short dual edges may remain. This can render the grid useless, as even a single very short dual edge

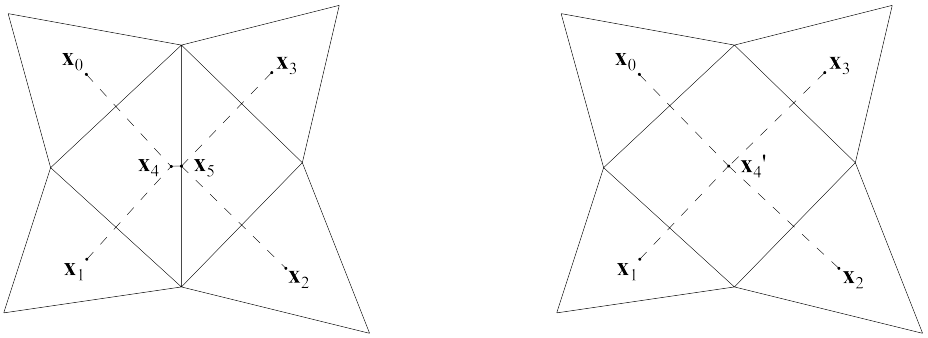



Figure 2.10: Left: a section of a grid with a very short dual edge. Right: the problematic dual edge is removed via cell merging.

can prevent convergence or cause solution blow-up. As a last resort the technique of *cell merging* can be used to remove short dual edges from the grid. In this process the edge associated with the short dual edge is removed, and its two adjacent cells are joined to form a single quadrilateral. A dual node associated with the quadrilateral must then be chosen, since quadrilaterals do not necessarily have a point which achieves the orthogonality results discussed in the previous section. The dual node could chosen to be the mean of the dual nodes corresponding to the original triangular cells, the position inside the cell which comes closest to achieving orthogonality between its bounding edges and their associated dual edges, or according to some other appropriate criterion. An example of cell merging is shown in Figure 2.10. The result is a *hybrid* grid, which contains both triangular and quadrilateral cells. It is less than ideal, because the quadrilateral cells formed in this manner have approximately twice the area of the surrounding triangular cells. It can also cause issues with the reconstruction of the flow variables. For example, in Section 3.8 a reconstruction is considered which is based at the node opposite an edge within the same cell. For quadrilateral cells, this choice is not unique.

Although cell merging was ultimately not required for any of the numerical examples presented in this thesis, it is important to keep it in mind. The solver may need to be extended to the case of hybrid grids in the future, as more complex geometries and extension of the solver to three dimensional flows are considered.

# Chapter 3

# Foundations of the Inviscid Solver

## 3.1 Introduction

In this chapter, the foundations of the inviscid solver are discussed. The starting point for the solver is the inviscid Mach-uniform method detailed in [WSW02]. Many of the concepts discussed in this chapter are introduced in that reference, which the reader is strongly encouraged to consult.

## 3.2 Grid Notation

In order to simplify the presentation of the solver, some aspects of the notation used to refer to different grid elements must first be discussed.

A typical edge is shown in Figure 3.1. The centre of the edge $i$ is $\boldsymbol{x}_i$. Each edge has a unit normal $\boldsymbol{n}_i$ – one of the the two possibilities is chosen arbitrarily. The edge unit tangent $\boldsymbol{t}_i$ is then defined so that $\{\boldsymbol{t}_i, \boldsymbol{n}_i\}$ forms a right-handed set. The edge end nodes are denoted $v_0$ and $v_1$, such that the edge unit tangent $\boldsymbol{t}_e$ points away from node $v_0$ towards node $v_1$. The cells bounded by the edge are denoted $\ell$ and $r$, such that the edge unit normal $\boldsymbol{n}_e$ points away from cell $\ell$ and into cell $r$. These adjacent cells are also referred to as the *left* and *right* adjacent cells respectively. The notation $e(d)$ is used to refer to the set of edges related to an arbitrary edge or cell $d$ (for example the edges bounding the cell $i$), and the subscript $e$ means an edge belonging to that set.

The circumcentres of the left and right cells are denoted $\boldsymbol{x}_\ell$ and $\boldsymbol{x}_r$ respectively. Similarly the flow variables at the circumcentres of these cells will be denoted by, for

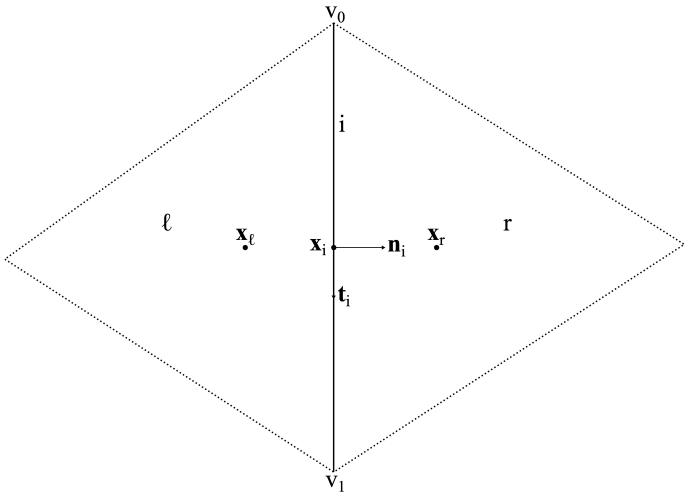



Figure 3.1: The notation used to refer to a grid edge $i$ and its associated objects and adjacent elements.

example $\varphi_\ell$ and $\varphi_r$. Note that this entails a minor abuse of notation, since $\ell$ and $r$ are being used to denote both the cell and its circumcentre. However, this should not cause confusion, as any flow variable associated with a cell will *always* be located at the circumcentre of that cell, unless otherwise noted. The same liberty is taken with regard to the edge notation. For example, $\varphi_e$ means the value of $\varphi$ at the centre of edge $e$.

## 3.3 Non-dimensionalisation

The non-dimensionalisation begins with the selection of reference values for the density, pressure, flow velocity magnitude, and characteristic length. The non-dimensional

density, flow velocity, and characteristic length are then defined by

$$\rho \coloneqq \frac{\tilde{\rho}}{\rho_r} \tag{3.1}$$

$$\boldsymbol{u} \coloneqq \frac{\tilde{\boldsymbol{u}}}{u_r} \tag{3.2}$$

$$\ell \coloneqq \frac{\tilde{\ell}}{\ell_r} \tag{3.3}$$

where $\varphi$ is the non-dimensional variable, $\tilde{\varphi}$ is the dimensional variable, and $\varphi_r$ is the chosen reference value. This notation will be adopted for the remainder of the thesis.

Using this non-dimensionalisation, the continuity equation (2.4) is found to remain invariant

$$\frac{\partial}{\partial t}\int_\Omega \rho d\Omega + \oint_{\partial\Omega} \rho\left(\boldsymbol{u}\cdot\boldsymbol{n}\right) dS = 0 \tag{3.4}$$

For many solvers, the non-dimensionalisation of the pressure is accomplished in the same manner

$$p \coloneqq \frac{\tilde{p}}{p_r} \tag{3.5}$$

To see why this should be avoided in the Mach-uniform case, consider the resulting non-dimensionalisation of the inviscid momentum equation

$$\frac{\partial}{\partial t}\int_\Omega \rho\boldsymbol{u} d\Omega + \oint_{\partial\Omega} \rho\boldsymbol{u}\left(\boldsymbol{u}\cdot\boldsymbol{n}\right) dS = -\frac{1}{\gamma M_r^2}\oint_{\partial\Omega} p\boldsymbol{n} dS \tag{3.6}$$

where $M_r$ denotes the reference Mach number, and the working fluid has been assumed to be is an ideal gas. The factor $1/\gamma M_r^2$ quickly becomes large as the Mach number is reduced. As this term is responsible for coupling the flow velocity to the pressure, the pressure-flow velocity coupling will become progressively weaker in the low Mach number limit.

An alternative definition of the non-dimensional pressure is

$$p \coloneqq \frac{\tilde{p} - p_r}{\rho_r u_r^2} \tag{3.7}$$

With this definition the non-dimensional inviscid momentum equation is invariant under the non-dimensionalisation

$$\frac{\partial}{\partial t}\int_\Omega \rho\boldsymbol{u} d\Omega + \oint_{\partial\Omega} \rho\boldsymbol{u}\left(\boldsymbol{u}\cdot\boldsymbol{n}\right) dS = -\oint_{\partial\Omega} p\boldsymbol{n} dS \tag{3.8}$$

This avoids the aforementioned problem with pressure–flow-velocity decoupling as $M_r \to 0$.

Next, the non-dimensionalisation of the energy equation is considered. Applying the non-dimensionalisation to (2.32) yields

$$\begin{aligned}\frac{\partial}{\partial t}\int_\Omega \left\{ M_r^2 p + \frac{\gamma-1}{2} M_r^2 \rho \left(\boldsymbol{u}\cdot\boldsymbol{u}\right)\right\} d\Omega \\ + \oint_{\partial\Omega} \left\{ \left(\boldsymbol{u}\cdot\boldsymbol{n}\right)\left[1 + \gamma M_r^2 p + \frac{\gamma-1}{2} M_r^2 \rho \left(\boldsymbol{u}\cdot\boldsymbol{u}\right)\right]\right\} dS = 0\end{aligned} \tag{3.9}$$

Note that the factor of the square of the reference Mach number has again appeared. However – unlike for the inviscid momentum equation, where it separated the pressure term from the flow velocity term – here it appears 'evenly' on all the flow variables. It can therefore be expected that it will not create a problem with flow variable decoupling. Numerical evidence supports this assertion.

Finally, the ideal gas equation of state is non-dimensionalised. Inserting the non-dimensional variables defined above yields

$$\frac{\left(\gamma-1\right) M_r^2}{u_r^2}\tilde{h} = \frac{1}{\rho}\left(1 + \gamma M_r^2 p\right) \tag{3.10}$$

If the non-dimensionalised enthalpy and reference enthalpy are defined by

$$\begin{aligned} h &\coloneqq \frac{\tilde{h}}{h_r} \\ h_r &\coloneqq \frac{u_r^2}{\left(\gamma-1\right) M_r^2}\end{aligned} \tag{3.11}$$

then the non-dimensionalised ideal gas equation of state is

$$h = \frac{1}{\rho}\left(1 + \gamma M_r^2 p\right) \tag{3.12}$$

Of course, there are several other relations that must be non-dimensionalised. Many of these follow the above non-dimensionalisation procedure without modification. For example, applying the non-dimensionalisation to the sonic velocity (2.31) yields the non-dimensional sonic velocity

$$a = \frac{1}{M_r}\sqrt{h} \tag{3.13}$$

Other relations, especially those required for the viscous solver, do require extension to the above non-dimensionalisation procedure. These extensions are discussed when

they become necessary.

## 3.4 Layout of the Primary Flow Variables

There are two common choices for the layout of the primary flow variables:

1. *Collocated*: the scalar flow variables and the momentum (or flow velocity) are solved for at the same locations. This is generally either at the centroids of the grid cells or at the grid nodes. This is the layout conventionally used by compressible solvers and is the simplest method.

2. *Staggered*: the scalar flow variables are solved for at the centres of the grid cells (normally at the centroids), whilst the normal momentum (or normal flow velocity) is solved for at the centres of the grid edges. This is the layout generally used for incompressible and pressure-based solvers, as it has been found to offer superior pressure-velocity coupling and damping of spurious modes in these cases compared to the collocated layout.

Since the main design objective of the solver is Mach-uniformity, a variant of the staggered layout will be employed. The expectation is that this will allow the solver to retain strong pressure-density-flow velocity coupling in the low Mach number limit, at the expense of somewhat greater complexity and computational cost.

Instead of the usual practice of locating the scalar variables at the centroids of the grid cells, advantage is taken of the special properties of the grid structure outlined in Section 2.2 by placing the scalar variables at the circumcentres of the grid cells (equivalently, at the nodes of the dual grid). The advantages of this choice are discussed in Section 3.8.

## 3.5 The Solution Process

The solver uses a discretised form of the governing equations to march from a guessed initial solution towards the steady-state solution. The actual discretisation of the governing equations is discussed in the next section. The solver proceeds in the following manner:

1. The initial conditions and boundary conditions are imposed. Usually, the initial conditions consist of applying the reference conditions or some of the boundary conditions across the grid.

2. The discretised continuity equation (3.21) is solved at the cell circumcentres, to compute the densities at the new time level, $\rho_i^{n+1}$.

3. The discretised normal momentum equation (3.26) is solved at the edge centres, to compute predictions to the normal momenta at the new time level, $(\mathbf{m}.\mathbf{n})_i^*$.

4. The discretised pressure correction equation (3.35) is solved at the cell circumcentres, to compute the pressure corrections at the new time level, $\delta p_i^{n+1}$.

5. The cell circumcentre pressure corrections are used to correct the pressures, to give the pressures at the new time level, $p_i^{n+1}$.

6. The cell circumcentre pressure corrections are used to correct the edge centre normal momenta, to give the normal momenta at the new time level, $(\mathbf{m}.\mathbf{n})_i^{n+1}$.

7. The equation of state is used to compute the cell circumcentre enthalpies, sonic velocities, and Mach numbers at the new time level.

8. The boundary conditions are updated.

9. Convergence is tested against some set of convergence criteria. If any of these tests fail, the process is repeated from step 2.

In the above, note that the continuity and normal momentum equations may be solved implicitly, semi-implicitly, or explicitly. However, the pressure correction equation is written in terms of the pressure corrections, and hence can *only* be solved implicitly.

## 3.6 Discretisation of the Governing Equations of Inviscid Flow

The governing equations are discretised using the usual finite-volume procedure. The method of lines is employed, with the time discretisation performed independently of the spatial discretisation. The flux terms are evaluated at the new time level. Since only the steady-state solution is of interest in this chapter, a simple foward Euler discretisation is used to perform the time discretisation. For an arbitrary flow variable $\varphi$ and grid location $i$

$$\left.\frac{\partial \varphi}{\partial t}\right|_i \approx \frac{\varphi_i^{n+1} - \varphi_i^n}{\Delta t_i} \tag{3.14}$$

The spatial discretisation involves the discretisation of two types of integrals: volume integrals and surface integrals. In the latter case, the discretisation is achieved by simply performing the integration over an appropriate control volume containing the location of the variable.

$$\oint_{\partial\Omega_i} \varphi \left(\boldsymbol{u} \cdot \boldsymbol{n}\right) dS = \sum_{e(i)} \int_e \varphi \left(\boldsymbol{u} \cdot \boldsymbol{n}\right) dl_e \tag{3.15}$$

where $e(i)$ ranges over the edges bounding the control volume for grid location $i$. It is important to note that at this stage the spatial discretisation is exact. In general, however, the exact variation of $\varphi$ along the edges of the control volume is not known – $\varphi$ is only available at discrete points. To complete the discretisation of the surface integral, the exact integral needs to be replaced using a numerical quadrature in terms of these discrete points. The simplest and least accurate procedure is to approximate the variation of $\varphi$ along the edge as being constant and equal to the value at the centre of the edge. Despite its simplicity, it can be shown to be sufficient for obtaining a second-order scheme. If the true variation of $\varphi$ along the edge is linear, the numerical integration is exact. This confirms that this surface integral discretisation is sufficiently accurate for a second-order-accurate scheme. The discrete surface integrals are therefore

$$\oint_{\partial\Omega_i} \varphi \left(\boldsymbol{u} \cdot \boldsymbol{n}\right) dS \approx \sum_{e(i)} \left\{\bar{l}_e \left(\boldsymbol{u} \cdot \boldsymbol{n}\right)_e \varphi_e\right\} \tag{3.16}$$

where $\bar{l}_e$ denotes the *signed edge length*. This is simply the edge length with a minus sign added if the edge unit normal points inwards, into the control volume. The value of $\varphi_e$ is interpolated – the interpolation procedure is discussed in Section 3.8 for the scalar flow variables and Section 3.9 for the momentum.

Next, the spatial discretisation of the volume integrals must be performed. This is again achieved using a numerical integration rule. As with the surface integrals, the simplest and least accurate rule is to assume the variation of $\varphi$ is constant across the volume and equal to the value at the centroid of the volume. This again exactly agrees with the true integral if the actual variation is linear, and so is sufficiently accurate for a second-order scheme. Usually, this integration scheme can be directly applied as $\varphi$ is directly solved for at the centroids of the control volumes. This is not the case here, however, as $\varphi$ is instead solved for at the circumcentres of the grid cells in the scalar case and at the centres of the edges in the case of the momentum, and in general these points do not coincide with the corresponding centroids. The

two options for handling this are discussed in Section A.2.1 of Appendix A. The option selected involves interpolating the primary flow variables at the centroids of the control volumes using the methods discussed in Section 3.8 and Section 3.9, then assuming the value acrosss the control volume is constant and equal to the interpolated value. The discretisation of the volume terms is therefore

$$\int_{\Omega_i} \varphi d\Omega \approx \Omega_i \varphi_{c(i)} \tag{3.17}$$

where $c(i)$ is the centroid of the control volume corresponding to grid location $i$.

Finally, the time level at which the spatial terms are evaluated must be discussed. Since the scheme is designed to be implicit, the numerical spatial terms should be evaluated at the new time level. For example, a surface integral term is evaluated as

$$\left( \oint_{\partial\Omega_i} \varphi \left( \boldsymbol{u} \cdot \boldsymbol{n} \right) dS \right)^{n+1} \approx \sum_{e(i)} \left\{ \bar{l}_e \left( \boldsymbol{u} \cdot \boldsymbol{n} \right)_e^{n+1} \varphi_e^{n+1} \right\} \tag{3.18}$$

However, there are two problems with this approach. First, the segregated design of the solver means that $(\boldsymbol{u} \cdot \boldsymbol{n})_e^{n+1}$ is not available until the end of the iteration, after the pressure corrections have been applied to the momentum predictions. Second, even if they were available, this would still lead to a non-linear system of equations that would be very expensive to solve. The spatial terms must therefore be linearised. Since only the steady-state solution is currently being sought, time-accuracy is of no concern. The simplest linearisation procedure, Picard linearisation, can therefore be used. In this procedure, the conserved variable being solved for is taken at the new time level, whilst the latest known values are used for the other variables. This yields

$$\left( \oint_{\partial\Omega_i} \varphi \left( \boldsymbol{u} \cdot \boldsymbol{n} \right) dS \right)^{n+1} \approx \sum_{e(i)} \left\{ \bar{l}_e \left( \boldsymbol{u} \cdot \boldsymbol{n} \right)_e' \varphi_e^{n+1} \right\} \tag{3.19}$$

The normal flow velocity $(\boldsymbol{u} \cdot \boldsymbol{n})'$ is computed using the latest known density and normal momentum at the edge centre. The interpolation of the edge centre densities for use in computing the edge centre flow velocities is discussed in Section 3.6.5.

### 3.6.1 The Continuity Equation

The non-dimensional continuity equation (3.4) is solved at the circumcentres of the grid cells, using the corresponding grid cell as the control volume. Applying the above

discretisation procedure, the following discretised continuity equation is obtained

$$\frac{\Omega_i \left(\rho_{c(i)}^{n+1} - \rho_{c(i)}^{n}\right)}{\Delta t_i} + \sum_{e(i)} \left\{ \bar{l}_e \left(\boldsymbol{u} \cdot \boldsymbol{n}\right)_e^{n+1} \rho_e^{n+1} \right\} = 0 \tag{3.20}$$

Applying Picard linearisation to the convection term yields

$$\frac{\Omega_i \left(\rho_{c(i)}^{n+1} - \rho_{c(i)}^{n}\right)}{\Delta t_i} + \sum_{e(i)} \left\{ \bar{l}_e \left(\boldsymbol{u} \cdot \boldsymbol{n}\right)_e' \rho_e^{n+1} \right\} = 0 \tag{3.21}$$

### 3.6.2 The Normal Momentum Equation

As previously mentioned, it is only the normal momentum that is solved for at the edge centres. At an edge centre $i$, the projection of the momentum equation (3.26) in the direction of the edge unit normal $\boldsymbol{n}_i$ is taken

$$\frac{\partial}{\partial t} \int_\Omega \left(\boldsymbol{m} \cdot \boldsymbol{n}_i\right) d\Omega + \oint_{\partial\Omega} \left(\boldsymbol{m} \cdot \boldsymbol{n}_i\right) \left(\boldsymbol{u} \cdot \boldsymbol{n}\right) dS = - \oint_{\partial\Omega} p \left(\boldsymbol{n} \cdot \boldsymbol{n}_i\right) dS \tag{3.22}$$

where the momentum has been written $\boldsymbol{m} \coloneqq \rho \boldsymbol{u}$. The resulting normal momentum equation is integrated over the control volume for each edge. Defining these control volumes requires more care than for the continuity equation. The natural choice is to define the control volume for each edge to be the union of the grid cells bounded by the edge (see Figure 3.2). However this creates a problem – the resulting control volumes overlap. It is well known that in order to accurately capture shocks a numerical scheme must respect the conservation properties of the governing equations, and that any change in the conserved quantity within the control volume must be due to the fluxes at the edges of the control volume and volume sources within the control volume only. Since the control volumes being used here overlap, this cannot be guaranteed, and it cannot be said that the numerical scheme is conservative. However, it is shown in [Wen01] that staggered schemes with this control volume definition do satisfy a somewhat weaker condition, referred to in that source as *generalised conservation*, and that the performance and accuracy of schemes that satisfy this condition are not strongly affected by the lack of true numerical conservation in practice. Numerical testing of this scheme supports that conclusion, as the scheme performs well even when strong shocks are present in the flow field.

The above control volume definition is only suitable for use with the interior edges. When solving the normal momentum equation at a boundary edge $b$, only the interior

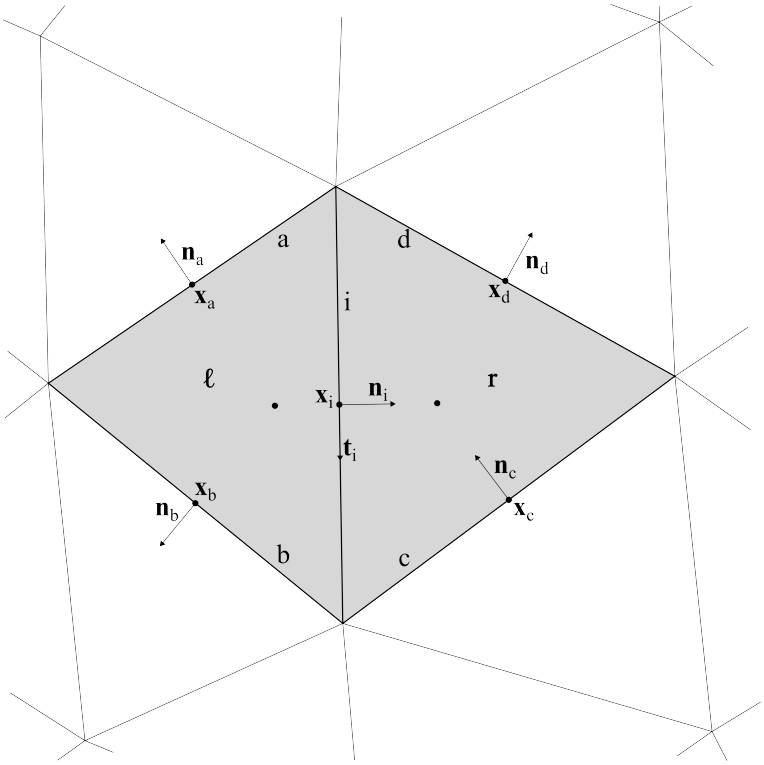



Figure 3.2: The stencil used for solving the normal momentum equation at edge $i$

adjacent cell is included in the stencil (see Figure 3.3).

Performing the above discretisation procedure for the normal momentum equation over the control volume corresponding to an edge $i$ results in the discretised normal momentum equation

$$\frac{\Omega_i \left[(\boldsymbol{m}\cdot\boldsymbol{n})^* - (\boldsymbol{m}\cdot\boldsymbol{n})^n\right]_{c(i)}}{\Delta t_i} + \sum_{e(i)} \left\{\bar{l}_e \left(\boldsymbol{u}\cdot\boldsymbol{n}\right)_e^* \left(\boldsymbol{m}_e\cdot\boldsymbol{n}_i\right)^*\right\} = T_{p_i} \tag{3.23}$$

where $e(i)$ ranges over the outer edges of the control volume for edge $i$, $T_{p_i}$ represents the pressure term, and

$$(\boldsymbol{m}\cdot\boldsymbol{n})_{c(i)} \coloneqq \boldsymbol{m}_{c(i)}\cdot\boldsymbol{n}_i \tag{3.24}$$

The two options for considered for the discretisation of the pressure term are discussed in Section A.2.2 of Appendix A. The selected method exploits the imposed

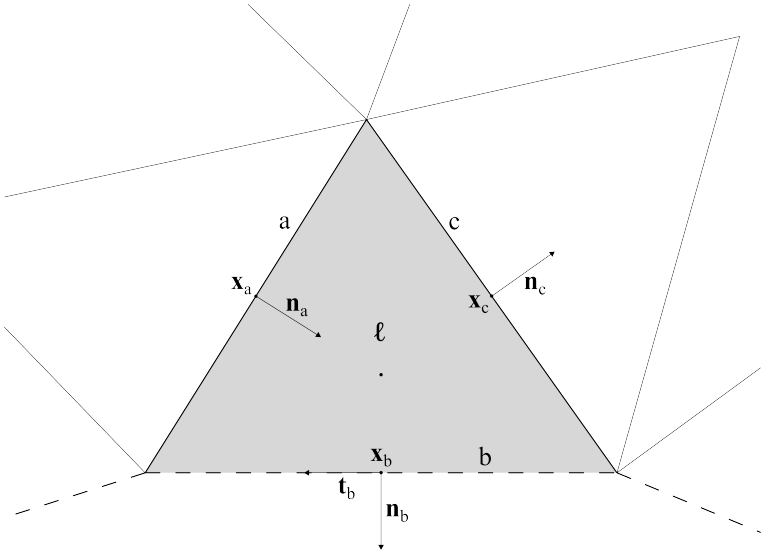



Figure 3.3: The stencil used for solving the normal momentum equation at boundary edge $b$. The boundary of the flow domain is represented by the dashed lines.

grid requirements, and takes the particularly simple form

$$T_{p_i} = \frac{\Omega_i \left(p_\ell - p_r\right)}{\|\boldsymbol{x}_\ell - \boldsymbol{x}_r\|_2} \tag{3.25}$$

where $\boldsymbol{x}$ denotes the circumcentre location and the subscripts $\ell$ and $r$ denote the left and right cells as described in Section 3.2.

Finally, applying Picard linearisation to the convection and pressure terms yields the discretised normal momentum equation

$$\frac{\Omega_i \left[(\boldsymbol{m}\cdot\boldsymbol{n})^* - (\boldsymbol{m}\cdot\boldsymbol{n})^n\right]_{c(i)}}{\Delta t_i} + \sum_{e(i)} \left\{\bar{l}_e \left(\boldsymbol{u}\cdot\boldsymbol{n}\right)'_e \left(\boldsymbol{m}_e\cdot\boldsymbol{n}_i\right)^*\right\} = \frac{\Omega_i \left(p_\ell - p_r\right)^n}{\|\boldsymbol{x}_\ell - \boldsymbol{x}_r\|_2} \tag{3.26}$$

### 3.6.3 The Momentum Correction

As described in Section 3.5, the normal momentum equation is solved for the predictions to the normal momenta at the new time level. Next the pressure correction equation, derived from the energy equation and described in the next section, is solved for the pressure corrections at the new time level. These pressure corrections are used to calculate the pressures at the new time level and to correct the normal momentum predictions. Therefore, the formula for the momentum correction in terms of the pressure corrections must be derived. This is both used to correct the momentum

predictions at the end of the iteration and inserted directly into the energy equation during the derivation of the pressure correction.

To derive the normal momentum correction, the discretised normal momentum equation is written at the prediction time level and at the new time level

$$\begin{aligned}
&\frac{\Omega_i\left(\boldsymbol{m}^{*}_{c(i)}-\boldsymbol{m}^{n}_{c(i)}\right)\cdot\boldsymbol{n}_i}{\Delta t_i}+\sum_{e(i)}\left\{\bar{l}_e\left(\boldsymbol{u}\cdot\boldsymbol{n}\right)'_e\left(\boldsymbol{m}_e\cdot\boldsymbol{n}_i\right)^{*}\right\}=-\Omega_i\nabla p\big|_i^{n}\cdot\boldsymbol{n}_i\\
&\frac{\Omega_i\left(\boldsymbol{m}^{n+1}_{c(i)}-\boldsymbol{m}^{n}_{c(i)}\right)\cdot\boldsymbol{n}_i}{\Delta t_i}+\sum_{e(i)}\left\{\bar{l}_e\left(\boldsymbol{u}\cdot\boldsymbol{n}\right)'_e\left(\boldsymbol{m}_e\cdot\boldsymbol{n}_i\right)^{n+1}\right\}=-\Omega_i\nabla p\big|_i^{n+1}\cdot\boldsymbol{n}_i
\end{aligned}\tag{3.27}$$

Subtracting the first from the second yields

$$\begin{aligned}
\frac{\Omega_i}{\Delta t}\left[\delta\left(\boldsymbol{m}\cdot\boldsymbol{n}\right)^{n+1}_{c(i)}\right]+\sum_{e(i)}\left\{\bar{l}_e\left(\boldsymbol{u}\cdot\boldsymbol{n}\right)'_e\left[\left(\boldsymbol{m}_e\cdot\boldsymbol{n}_i\right)^{n+1}-\left(\boldsymbol{m}_e\cdot\boldsymbol{n}_i\right)^{*}\right]\right\}\\
=-\Omega_i\nabla\left(p^{n+1}-p^{n}\right)\big|_i\cdot\boldsymbol{n}_i
\end{aligned}\tag{3.28}$$

where

$$\delta\left(\boldsymbol{m}\cdot\boldsymbol{n}\right)^{n+1}_{c(i)}\coloneqq\left(\boldsymbol{m}\cdot\boldsymbol{n}\right)^{n+1}_{c(i)}-\left(\boldsymbol{m}\cdot\boldsymbol{n}\right)^{*}_{c(i)}\tag{3.29}$$

is the normal momentum correction.

It can be seen that the normal momentum correction has two components: a first-order correction due to the effect of the pressure correction during the time step on the normal pressure gradient, and a second-order correction due to the effect of the normal momentum correction on the convective flux. As long as the normal momentum predictions are not too inaccurate, it is reasonable to assume that the normal momentum correction is dominated by the first-order correction. Additionally, the normal momentum correction tends to zero as the solution approaches the steady state, so small changes in the formula for the normal momentum correction can be expected to affect only convergence, not the obtained solution. Subject to numerical confirmation that convergence and stability are not unduly affected by this assumption, the second-order corrections can therefore be dropped. The resulting normal momentum correction is

$$\delta\left(\boldsymbol{m}\cdot\boldsymbol{n}\right)^{n+1}_{c(i)}=-\Delta t_i\nabla\delta p\big|_i^{n+1}\cdot\boldsymbol{n}_i\tag{3.30}$$

where $\delta p^{n+1}\coloneqq p^{n+1}-p^{n}$ is the pressure correction. It was found, in fact, that dropping the second-order corrections had little effect on the convergence properties of the inviscid solver. It might be necessary, of course, to revisit this assumption during

the development of the viscous and unsteady solvers.

If the grid growth rate is limited to allow area increases of less than about 20% between adjacent cells, the above arguments also justify the assumption that the normal momentum correction at the centre of edge $i$ is the same as the normal momentum correction at the centroid $c(i)$ of the associated control volume

$$\delta\left(\boldsymbol{m}\cdot\boldsymbol{n}\right)_i^{n+1} = -\Delta t_i \nabla\delta p\big|_i^{n+1}\cdot\boldsymbol{n}_i \tag{3.31}$$

Again, numerical testing found that making this assumption did not noticeably affect the convergence properties of the scheme on the grids used in numerical testing.

Finally, approximating the normal pressure correction gradient in the same manner as for the normal pressure gradient in the normal momentum equation (see 3.25) yields the normal momentum correction

$$\delta\left(\boldsymbol{m}\cdot\boldsymbol{n}\right)_i^{n+1} = \frac{\Delta t_i\left(\delta p_\ell - \delta p_r\right)}{\left\|\boldsymbol{x}_\ell - \boldsymbol{x}_r\right\|_2} \tag{3.32}$$

The derivation of the pressure correction equation requires not just the normal momentum correction, but the complete momentum correction. The above formula must therefore be extended by adding a tangential momentum correction. The derivation of the tangential momentum correction is not quite as clear, since the method does not include a tangential momentum equation. There are two reasonable choices for the tangential momentum correction, which are discussed in Section A.2.3 of Appendix A. The selected form for the tangential momentum correction yields the momentum correction

$$\delta\boldsymbol{m}_i^{n+1} = -\Delta t_i \nabla\delta p\big|_i^{n+1} \tag{3.33}$$

### 3.6.4 The Pressure Correction Equation

The pressure correction equation is derived from the non-dimensionalised energy equation (3.9). Applying the usual discretisation yields

$$\begin{aligned}\frac{\Omega_i M_r^2}{\Delta t_i}\left\{p_{c(i)}^{n+1} - p_{c(i)}^{n} + \frac{\gamma-1}{2}\left[\left(\frac{\boldsymbol{m}\cdot\boldsymbol{m}}{\rho}\right)_{c(i)}^{n+1} - \left(\frac{\boldsymbol{m}\cdot\boldsymbol{m}}{\rho}\right)_{c(i)}^{n}\right]\right\} \\ + \sum_{e(i)}\left\{\bar{l}_e\left(\boldsymbol{u}\cdot\boldsymbol{n}\right)_e^{n+1}\left[1+\gamma M_r^2 p + \frac{\gamma-1}{2}M_r^2\frac{\boldsymbol{m}\cdot\boldsymbol{m}}{\rho}\right]_{c(i)}^{n+1}\right\} = 0\end{aligned} \tag{3.34}$$

Next, the pressure correction $\delta p^{n+1}$ and momentum correction discussed in the previous section are inserted. This results in a highly non-linear equation with terms of second and third order in $\Delta t \nabla \delta p$. To linearise, $\Delta t \nabla \delta p$ is assumed small, and hence the higher-order terms can be neglected. This is true near the steady state, but is not necessarily well-founded in general. An alternative, more accurate method of handling this non-linearity is to use a multi-stage pressure correction procedure, involving second and possibly third pressure correction equations, which would greatly increase the complexity and computational cost of the scheme. Fortunately, this was not found to be necessary in practice. The linearised pressure correction equation is therefore

$$\begin{aligned}
&\frac{\Omega_i M_r^2}{\Delta t_i}\left[\delta p^{n+1} + \frac{\gamma-1}{2}\left(\frac{\boldsymbol{m}^*\cdot\boldsymbol{m}^*}{\rho^{n+1}} - \frac{\boldsymbol{m}\cdot\boldsymbol{m}}{\rho}\bigg|^n\right)\right]_{c(i)} \\
&\quad + \Omega_i M_r^2 (1-\gamma)\, \boldsymbol{u}^*_{c(i)} \cdot \nabla \delta p^{n+1}_{c(i)} \\
&\quad + \sum_{e(i)} \left\{ \bar{l}_e \left(\boldsymbol{u}\cdot\boldsymbol{n}\right)'_e \left[1 + \gamma M_r^2 \left(p^n + \delta p^{n+1}\right) + \frac{\gamma-1}{2} M_r^2 \cdot \frac{\boldsymbol{m}^*\cdot\boldsymbol{m}^*}{\rho^{n+1}}\right]_e \right\} \\
&\quad + (1-\gamma) M_r^2 \sum_{e(i)} \left\{ \bar{l}_e \left(\boldsymbol{u}\cdot\boldsymbol{n}\right)'_e \left(\frac{\Delta t}{\rho^{n+1}} \boldsymbol{u}^* \cdot \nabla \delta p^{n+1}\right)_e \right\} \\
&\quad + \sum_{e(i)} \left\{ \bar{l}_e \left(\frac{\Delta t_e}{\rho_e^{n+1}} \cdot \frac{\delta p^{n+1}_{c_\ell} - \delta p^{n+1}_{c_r}}{\|\boldsymbol{x}_{c_\ell} - \boldsymbol{x}_{c_r}\|_2}\right) \left[1 + \gamma M_r^2 p^n + \frac{\gamma-1}{2} M_r^2 \cdot \frac{\boldsymbol{m}^*\cdot\boldsymbol{m}^*}{\rho^{n+1}}\right]_e \right\} = 0
\end{aligned} \tag{3.35}$$

Two observations should be made here. The penultimate term represents the correction of the convected quantity due to the momentum correction. This term quickly becomes small as the steady state is approached, and is exactly zero as it is reached. Dropping this term therefore has no effect on the steady-state solution, and it is also likely that convergence would not be badly affected. Ultimately this was confirmed experimentally, and therefore the choice can be made to either include or exclude the penultimate term. The latter is chosen for the inviscid solver, but this point must be re-examined when considering the development of the viscous and unsteady solvers. The other observation is that the final term has a Possion-equation-like character, not unlike the Poisson equation solved for the pressure in incompressible solvers such as the SIMPLE algorithm [FP02]. This hints at the scheme becoming SIMPLE-like in the low Mach number limit.

### 3.6.5 Reconstruction of the Edge Centre Normal Flow Velocities

The convection terms of each of the governing equations contain terms of the form

$$(\boldsymbol{u} \cdot \boldsymbol{n})_e^n \, \varphi_e^{n+1} \tag{3.36}$$

for some edge $e$. In this solver, however, the edge centre normal flow velocity $(\boldsymbol{u} \cdot \boldsymbol{n})_e$ is unknown. Instead, the edge centre normal momentum $(\boldsymbol{m} \cdot \boldsymbol{n})_e$ and cell circumcentre densities are solved for. The normal flow velocity must therefore be calculated by dividing the normal momentum by an interpolated density $\rho_e$.

$$(\boldsymbol{u} \cdot \boldsymbol{n})_e = \frac{(\boldsymbol{u} \cdot \boldsymbol{n})_e}{\rho'_e} \tag{3.37}$$

The focus of this section is the interpolation of this density. It is important to note that the interpolation of this density is distinct from that used for the convected density solved for in the continuity equation. It is found in practice that using a central interpolation for this density rather than the upwind interpolation used for the convected density leads to a more stable scheme.

In [WSW02], a volume-weighted central interpolation is suggested

$$\rho'_e \approx \frac{\Omega_{c_r} \rho_{c_\ell} + \Omega_{c_\ell} \rho_{c_r}}{\Omega_{c_\ell} + \Omega_{c_r}} \tag{3.38}$$

In the case of the present solver, the location of the known densities at the cell circumcentres means that a simpler interpolation is possible

$$\rho'_e \approx \frac{1}{2} \rho_{c_\ell} + \frac{1}{2} \rho_{c_r} \tag{3.39}$$

In fact, testing both methods found very little difference. However, the former may be marginally better in the case of very poor quality grids, while the latter is more efficient since adjacent cell weights do not need to be stored (or calculated prior to each use).

### 3.6.6 Reconstruction of the Edge Centre Normal Scalar Gradients

Due to the layout of the flow variables and the properties possessed by the grids used with the solver, reconstruction of edge centre normal scalar gradients is trivial – this is in fact the major advantage of the strict grid requirements. With the layout shown in Figure 3.1, the normal gradient of the arbitrary scalar $\varphi$ at the centre of edge $e$ is

approximated by

$$\frac{\partial \varphi}{\partial \boldsymbol{n}_e} \equiv \nabla \varphi \big|_e \cdot \boldsymbol{n}_e \approx \frac{\varphi_r - \varphi_\ell}{\|\boldsymbol{x}_r - \boldsymbol{x}_\ell\|_2} \tag{3.40}$$

As $\varphi_\ell$ and $\varphi_r$ are located at the left and right adjacent cell circumcentres respectively, this approximation is exact for a linear variation of $\varphi$. As a result, this approximation is sufficient for a second-order-accurate scheme. This is in contrast to the usual situation - where the scalar primary flow variables are located at the cell centroids, and this simple two-point approximation only achieves second-order accuracy for uniform grids.

It is important to note that the above approximation is not sufficient for higher-than-second-order accuracy.

## 3.7 Reconstruction of the Convective Fluxes

A key consideration in the development of finite volume methods is the reconstruction of the edge centre convective fluxes appearing in the discretised governing equations presented in Section 3.6. In this section, the most basic method for accomplishing this reconstruction is presented. In the next chapter an more complex method is considered and shown to improve the performance of the solver.

### 3.7.1 Fully Upwind

The fully upwind method is by far the simplest upwind convective flux reconstruction method. The normal flow velocity computed according to Section 3.6.5 is used together with the convected flow variable taken solely from the upwind state

$$(\boldsymbol{u} \cdot \boldsymbol{n})_e \, \varphi_e \approx \hat{\mathcal{F}}_e \left(\varphi_L, \varphi_R\right) \coloneqq (\boldsymbol{u} \cdot \boldsymbol{n})_e \, \bar{\varphi}_e \tag{3.41}$$

where $\hat{\mathcal{F}}_e : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is the chosen numerical flux function at the edge centre, $\varphi_L$ and $\varphi_R$ are the values of the arbitrary flow variable from the left and right states respectively, and the reconstructed convected flow variable $\bar{\varphi}_e$ is

$$\bar{\varphi}_e \coloneqq \begin{cases} \varphi_L & \text{if } (\boldsymbol{u} \cdot \boldsymbol{n})_e \geq 0 \\ \varphi_R & \text{if } (\boldsymbol{u} \cdot \boldsymbol{n})_e < 0 \end{cases} \tag{3.42}$$

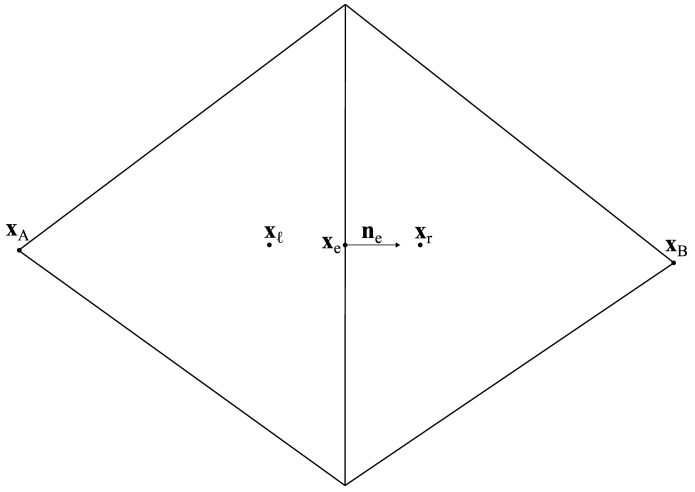



Figure 3.4: The layout and notation for performing scalar field reconstructions in the neighbourhood of the centre of a typical edge $e$.

## 3.8 Reconstruction of Scalar Fields

As already discussed, the scalar primary flow variables are obtained by solving the continuity and pressure correction (energy) equations at the circumcentres of the grid cells. However, solving these equations requires the (implicit) evaluation of these same flow variables at the centres of the edges bounding the grid cells. Therefore, a method of reconstructing scalar fields from the cell circumcentre values is necessary.

In this section, scalar field reconstruction is considered with reference to Figure 3.4. The focus is on piecewise polynomial reconstructions of first- and second-order, due to their relative simplicity and small reconstruction stencils. Higher-order polynomial reconstruction procedures are possible, but require larger stencils that are harmful to stability. It is worth mentioning that non-polynomial reconstruction methods are also possible, such as radial basis function (RBF) reconstructions [BW01]. An RBF method was tested with the present solver, but it found to have a detrimental effect on stability with no compensating benefits. However, a more thorough consideration of reconstructions of this type could be considered in future development of the solver.

### 3.8.1 Piecewise Constant Scalar Field Reconstruction

The simplest and least accurate method of piecewise polynomial scalar field reconstruction is to assume that the scalar field is constant across each cell and equal to the value at the cell circumcentre. That is, the left and right states for edge $e$ (see Figure 3.4)

are evaluated according to

$$\begin{aligned} \varphi_L &= \varphi_\ell \\ \varphi_R &= \varphi_r \end{aligned} \tag{3.43}$$

Despite its poor accuracy, this method of scalar reconstruction does have a major advantage, in that it is guaranteed to preserve monotonicity. This fact follows trivially as the reconstructed scalar is simply equal to the upwind scalar, and therefore can never be a new extremum. The implications of monotonicity preservation are discussed in [Lan98], where it is shown that monotonicity preservation prevents the generation of spurious oscillations in the neighbourhood of discontinuties such as shocks. The presence of discontinuities, even strong shocks, in the reconstruction stencil will therefore not degrade the stability of the scheme obtained using this scalar reconstruction method. However, as stated in the reference above, monotonicity preservation is neither a necessary nor a sufficient condition for the avoidance of these numerical oscillations in general.

### 3.8.2 Piecewise Linear Scalar Field Reconstruction

The accuracy of the reconstruction can be significantly improved by including linear corrections to the value at the cell circumcentres. Thus, the scalar field is assumed to have a linear variation across the reconstruction stencil. The left and right states for edge $e$ are therefore evaluated according to

$$\begin{aligned} \varphi_L &= \varphi_\ell + \overline{\nabla\varphi}\big|_L \cdot (\boldsymbol{x}_e - \boldsymbol{x}_\ell) \\ \varphi_R &= \varphi_r + \overline{\nabla\varphi}\big|_R \cdot (\boldsymbol{x}_e - \boldsymbol{x}_r) \end{aligned} \tag{3.44}$$

where $\overline{\nabla\varphi}$ is an appropriate approximation to the scalar gradient. Two choices for the approximation of this gradient are discussed in the following subsections. For the left state, these approximations correspond to a reconstruction based at the left adjacent cell circumcentre $\boldsymbol{x}_\ell$ and a reconstruction based at the vertex $\boldsymbol{x}_A$ of the left adjacent cell opposite the edge centre respectively.

Although the accuracy of the scalar field reconstruction is greatly improved by the inclusion of the lowest-order corrections compared to the piecewise constant scalar field reconstruction, the monotonicity of the scheme has been lost. This means that new extrema can be generated in the interior solution, and the aforementioned spurious numerical oscillations will typically appear near discontinuities. This will greatly affect the stability of the scheme near such discontinuities, and is likely lead to the solution

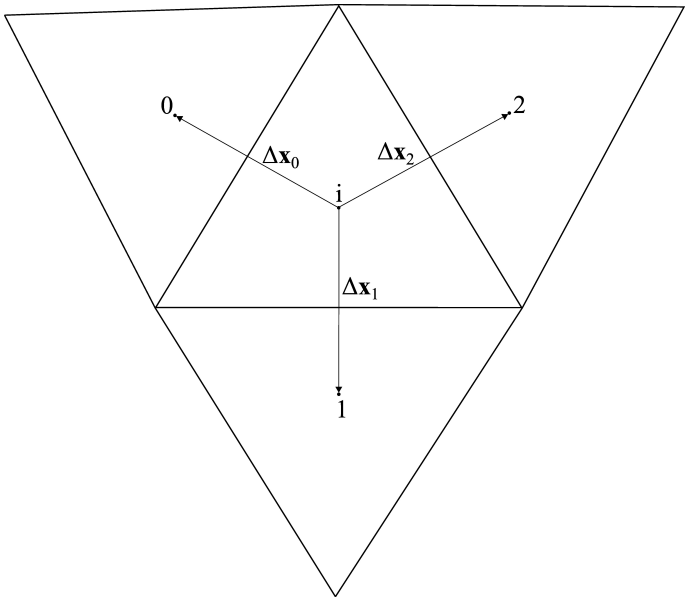



Figure 3.5: A typical stencil for a cell-based scalar gradient reconstruction based at the circumcentre of cell $i$.

process failing to convergence or even breaking up. To prevent this from happening, *gradient limiting* must be included. This effectively switches off the gradient terms where the solution is not smooth, preventing the generation of spurious extrema and the resulting numerical oscillations. This is discussed further in Section 4.4.

### 3.8.3 Cell-Based Scalar Gradient Reconstruction

In this scalar gradient reconstruction method, the reconstruction is based at the circumcentre of a grid cell, with the base cell itself and all adjacent cells included in the stencil (see Figure 3.5).

In the neighbourhood of the circumcentre of the base cell, $\boldsymbol{x}_i$, the scalar field $\varphi(\boldsymbol{x})$ is assumed to be approximated by the linear reconstruction polynomial $\bar{\varphi} : \mathbb{R}^2 \to \mathbb{R}$

$$\varphi(\boldsymbol{x}) \approx \bar{\varphi}(\Delta\boldsymbol{x}) \coloneqq \varphi(\boldsymbol{x}_i) + b\Delta x + c\Delta y \tag{3.45}$$

where $b$ and $c$ are unknown reconstruction coefficients to be determined and

$$\Delta\boldsymbol{x} \equiv \left\{ \begin{array}{c} \Delta x \\ \Delta y \end{array} \right\} \coloneqq \boldsymbol{x} - \boldsymbol{x}_i \tag{3.46}$$

is the position of the point at which the reconstruction polynomial is being evaluated

relative to the base cell circumcentre. As can be seen, the reconstruction coefficients approximate the gradient of the scalar field

$$\nabla\varphi\big|_i \approx \overline{\nabla\varphi}\big|_i \equiv \left\{ \begin{array}{c} b \\ c \end{array} \right\} \tag{3.47}$$

With the stencil constructed, the reconstruction coefficients can be determined by assembling a linear system of equations, expressing the equality of the reconstruction polynomial with the known scalar values at the circumcentres of the stencil cells other than the base cell, as

$$\left[ \begin{array}{cc} \Delta x_0 & \Delta y_0 \\ \Delta x_1 & \Delta y_1 \\ \Delta x_2 & \Delta y_2 \end{array} \right] \cdot \left\{ \begin{array}{c} b \\ c \end{array} \right\} = \left\{ \begin{array}{c} \varphi_0 - \varphi_i \\ \varphi_1 - \varphi_i \\ \varphi_2 - \varphi_i \end{array} \right\} \tag{3.48}$$

In general, this yields an over-determined system of equations, so it is the least-squares solution that is computed.

The elements of the system matrix in (3.48) are proportional to the grid spacing, and so the system matrix approaches the zero matrix as the grid spacing is reduced. This will cause numerical problems as progressively finer grids are used, as the determinant of the matrix tends to zero and that of its inverse grows without bound. To ameliorate this problem, the system can be replaced with a scaled system. A characteristic length $\ell_{\text{char}}$ is selected (distinct from the reference length $\ell_r$ used in the non-dimensionalisation of the solver). This characteristic length could be the square root of the area of the base cell, or the average distance from the base cell circumcentre to the other stencil cell circumcentres, for example. With the characteristic length selected, the positions and reconstruction coefficients are scaled according to

$$\begin{aligned} \Delta\widetilde{\boldsymbol{x}} &\coloneqq \frac{\Delta \boldsymbol{x}}{\ell_{\text{char}}} \\ \widetilde{b} &\coloneqq \ell_{\text{char}} b \\ \widetilde{c} &\coloneqq \ell_{\text{char}} c \end{aligned} \tag{3.49}$$

The scaled reconstruction polynomial is required to agree with the unscaled reconstruction polynomial

$$\varphi\left(\boldsymbol{x}_i\right) + b\Delta x + c\Delta y = \varphi\left(\boldsymbol{x}_i\right) + \widetilde{b}\Delta\widetilde{x} + \widetilde{c}\Delta\widetilde{y} \tag{3.50}$$

The unscaled reconstruction coefficients can therefore be obtained from

$$\left\{ \begin{array}{c} b \\ c \end{array} \right\} = \frac{1}{\ell_{\text{char}}} \left[ \begin{array}{cc} \Delta\widetilde{x}_0 & \Delta\widetilde{y}_0 \\ \Delta\widetilde{x}_1 & \Delta\widetilde{y}_1 \\ \Delta\widetilde{x}_2 & \Delta\widetilde{y}_2 \end{array} \right]^+ \cdot \left\{ \begin{array}{c} \varphi_0 - \varphi_i \\ \varphi_1 - \varphi_i \\ \varphi_2 - \varphi_i \end{array} \right\} \tag{3.51}$$

where $[\cdot]^+$ denotes the Moore-Penrose pseudoinverse (see, for example, [BG03, p. 40]).

Formally, (3.48) and (3.51) are identical. However, the improved numerical condition of the latter will prevent numerical instability from becoming an issue as the grid is refined.

Using the cell-based method of scalar gradient reconstruction, the left and right states for edge $e$ in Figure 3.4 are evaluated according to

$$\begin{aligned} \varphi_L &= \varphi_\ell + \overline{\nabla\varphi}\big|_\ell \cdot (\boldsymbol{x}_e - \boldsymbol{x}_\ell) \\ \varphi_R &= \varphi_r + \overline{\nabla\varphi}\big|_r \cdot (\boldsymbol{x}_e - \boldsymbol{x}_r) \end{aligned} \tag{3.52}$$

### 3.8.4 Node-Based Scalar Gradient Reconstruction

In this scalar gradient reconstruction method, the reconstruction is based at a grid node, with all adjacent cells included in the stencil (see Figure 3.6).

In some neighbourhood of the base node $\boldsymbol{x}_i$, the scalar field is assumed to be approximated by the linear reconstruction polynomial $\bar{\varphi} : \mathbb{R}^2 \to \mathbb{R}$ defined by

$$\varphi(\boldsymbol{x}) \approx \bar{\varphi}(\Delta\boldsymbol{x}) \coloneqq a + b\Delta x + c\Delta y \tag{3.53}$$

Note the important difference between this reconstruction polynomial and that used for cell-based scalar gradient reconstruction (3.45): in the node-based case, the value of the scalar field at the base point is not known, and so an additional unknown reconstruction coefficient must be added to the system. The approximation to the gradient of the scalar field remains the same, (3.47).

The linear system to be solved for the unknown reconstruction coefficients is now

$$\left[ \begin{array}{ccc} 1 & \Delta x_0 & \Delta y_0 \\ 1 & \Delta x_1 & \Delta y_1 \\ \vdots & \vdots & \vdots \\ 1 & \Delta x_5 & \Delta y_5 \end{array} \right] \cdot \left\{ \begin{array}{c} a \\ b \\ c \end{array} \right\} = \left\{ \begin{array}{c} \varphi_0 \\ \varphi_1 \\ \vdots \\ \varphi_5 \end{array} \right\} \tag{3.54}$$

As for the cell-based case this is normally an over-determined system, for which the

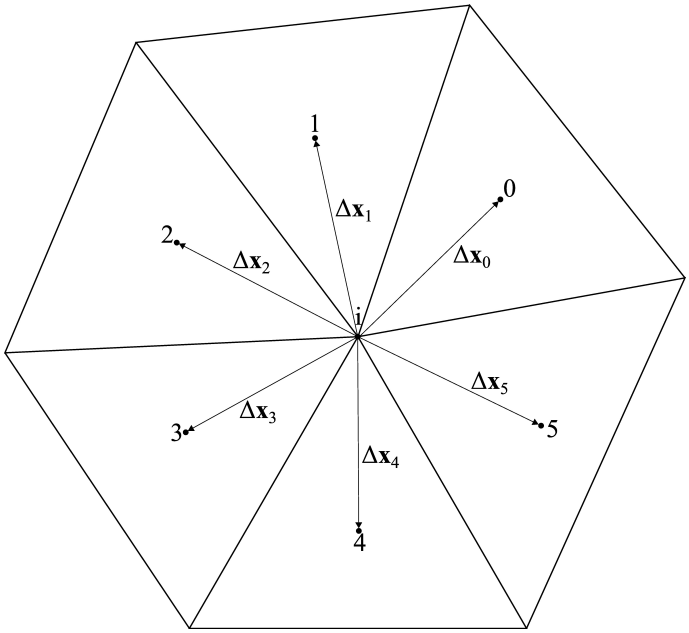



Figure 3.6: A typical stencil for a node-based scalar gradient reconstruction based at node $i$.

least-squares solution is computed.

To prevent numerical problems as the grid is refined, scaling is again applied. This follows the same procedure as (3.49), but note that the new reconstruction coefficient, $a$, is not scaled since it corresponds to the constant term. The unscaled reconstruction coefficients are therefore obtained from

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{\ell_{\text{char}}} & 0 \\ 0 & 0 & \frac{1}{\ell_{\text{char}}} \end{bmatrix} \begin{Bmatrix} a \\ b \\ c \end{Bmatrix} = \begin{bmatrix} 1 & \Delta\widetilde{x}_0 & \Delta\widetilde{y}_0 \\ 1 & \Delta\widetilde{x}_1 & \Delta\widetilde{y}_1 \\ \vdots & \vdots & \vdots \\ 1 & \Delta\widetilde{x}_5 & \Delta\widetilde{y}_5 \end{bmatrix}^{+} \cdot \begin{Bmatrix} \varphi_0 \\ \varphi_1 \\ \vdots \\ \varphi_5 \end{Bmatrix} \tag{3.55}$$

Using the node-based method of scalar gradient reconstruction, the left and right states for edge $e$ in Figure 3.4 are evaluated according to

$$\begin{aligned} \varphi_L &= \varphi_\ell + \overline{\nabla\varphi}\big|_A \cdot (\boldsymbol{x}_e - \boldsymbol{x}_\ell) \\ \varphi_R &= \varphi_r + \overline{\nabla\varphi}\big|_B \cdot (\boldsymbol{x}_e - \boldsymbol{x}_r) \end{aligned} \tag{3.56}$$

### 3.8.5 Comparison of the Node-Based and Cell-Based Scalar Gradient Reconstructions

Both the node-based and cell-based scalar reconstructions meet the requirements for a second-order scheme and so either could be used. However, each has a key advantage over the other. It is therefore likely that selecting one over the other will have a signficant effect on the properties of the solver. These differences can be seen by referring to Figure 3.7, which shows stencils for the continuity equation for a typical cell when using the node-based and cell-based scalar reconstructions. Clearly, cell-based scalar reconstruction has resulted in a more compact stencil. This is advantageous for efficiency, as it results in less fill-in of the sparse system, and for accuracy, as the information used for the reconstruction is more local to the reconstruction point. On the other hand, the stencil used for node-based scalar reconstruction is strictly upwind of the base cell, whereas the stencil used for cell-based scalar reconstruction includes a stencil cell that is downwind of the base cell. As a result, cell-based scalar reconstruction is likely to lead to a less stable scheme, possibily even requiring the addition of artificial dissipation for stabilisation.

Numerical examples were run with several combinations of node-based and cell-based scalar reconstruction. It was found that a satisfactory solver requires node-based scalar reconstruction for density reconstruction in the continuity equation, but that the pressure reconstruction in the pressure correction equation performs well with either the node-based or cell-based scalar reconstruction. The latter result is understandable since the pressure correction equation already includes gradient and Laplacian-like terms – there is therefore no additional downwind extension of the stencil due to the use of cell-based scalar reconstruction in the convection term. For the numerical examples presented in this thesis, cell-based scalar reconstruction is used for the pressure correction equation due to the improved numerical efficiency resulting from the smaller stencil. The authors of [VSW06] came to the same conclusion.

## 3.9 Reconstruction of the Momentum Field

In addition to reconstructing scalars at the edge centres, momentum reconstruction must also be performed. This is more complex that scalar field reconstruction, because only the normal component of momentum is known at the edge centres (the tangential component may also be specified at some boundary edges, see Section 4.5 for further details). The momentum field cannot therefore simply be reconstructed componentwise

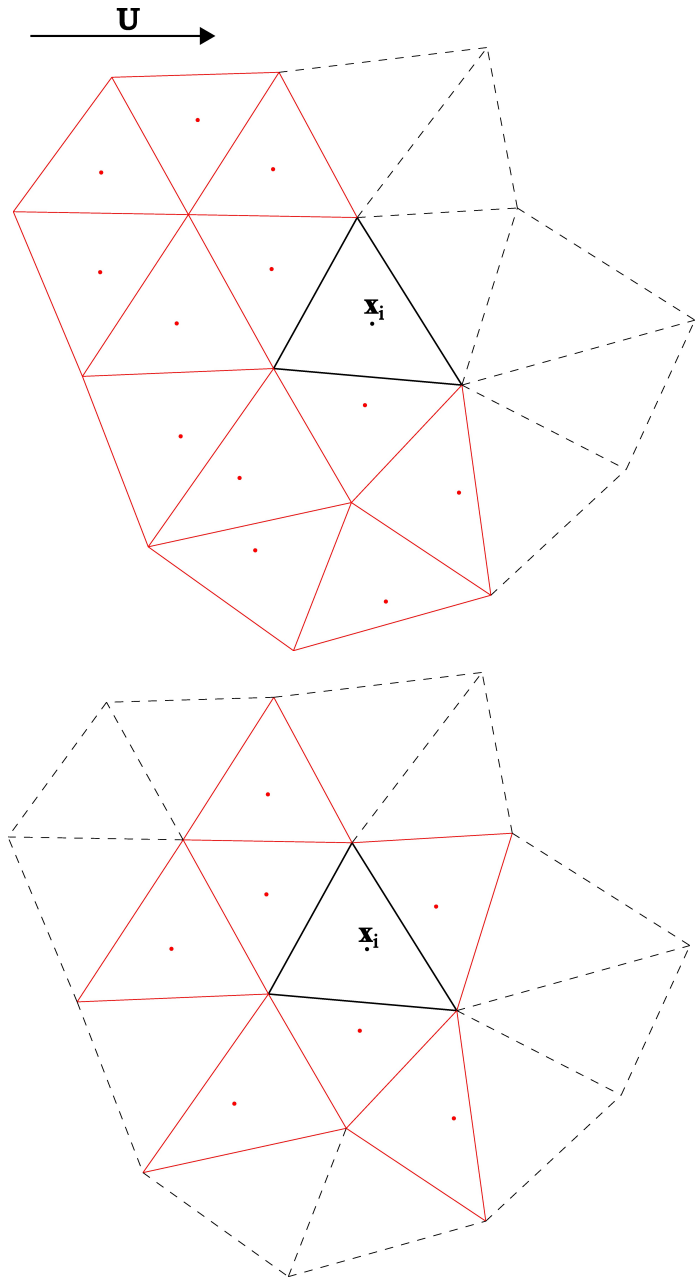



Figure 3.7: The stencil (shown in red) for the continuity equation at cell $i$ using node-based (top) and cell-based (bottom) piecewise linear scalar reconstruction. The velocity field is constant and left-to-right across the page.

as a $n$-tuple of independent scalar fields. Instead, methods that reconstruct the full momentum vector as a single object from the scattered momentum components are required. The resulting reconstructed momentum field is used for three purposes:

1. To implictly reconstruct the convected component of momentum at the edge centres in the normal momentum equation.
2. To explicitly reconstruct the tangential momentum at the edge centres for use by the boundary conditions, to calculate the edge centre Mach number etc.
3. To explicitly reconstruct the momentum at the cell circumcentres.

For the purpose of the present solver, first- and second-order reconstruction methods were considered. Higher-order reconstruction methods are of course possible, but were found to lead to excessively large stencils, difficulty choosing reconstruction thresholds, more complex gradient limiters and so forth. These are therefore left for potential future development, as are non-polynomial methods such as radial basis function reconstructions.

### 3.9.1 Piecewise Constant Momentum Field Reconstruction

As for the scalar case, piecewise constant momentum field reconstruction is the simplest and least accurate reconstruction procedure. Unlike the scalar case, however, it is non-trivial for the reason discussed above.

For cell $i$ in Figure 3.8, the stencil consists of the bounding edges $e(i) = \{a, b, c\}$. The reconstructed momentum field $\bar{\boldsymbol{m}}$ is assumed to be constant within each grid cell, with its value set by the requirement that the closest possible agreement with the normal components at the centres of the bounding edges is obtained (in the least-squares or minimum-norm sense). To achieve this, a reconstruction coefficient vector $\boldsymbol{\varepsilon}$ is computed for each bounding edge such that

$$\boldsymbol{m}_i \approx \bar{\boldsymbol{m}}_i \coloneqq \sum_{e(i)} \boldsymbol{\varepsilon}_e \left(\boldsymbol{m} \cdot \boldsymbol{n}\right)_e \tag{3.57}$$

To determine the reconstruction coefficients, three equations are obtained by demanding that the reconstructed momentum agrees with the normal momentum at the

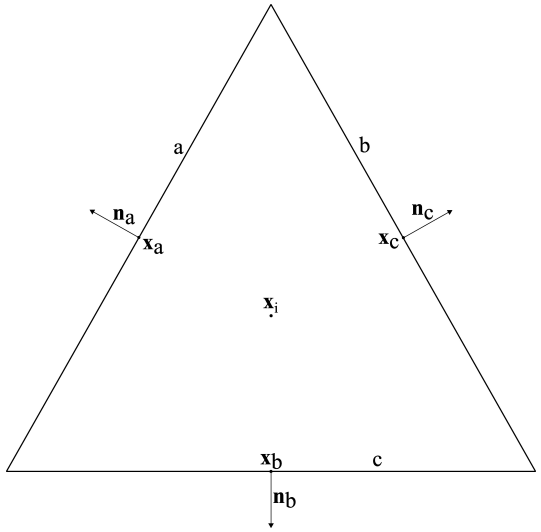



Figure 3.8: A typical stencil for a piecewise constant momentum reconstruction at the circumcentre of cell $i$.

centre of each stencil edge. For cell $i$ of Figure 3.8, these equations are

$$\begin{aligned}
(\boldsymbol{m}\cdot\boldsymbol{n})_a(\boldsymbol{\varepsilon}_a\cdot\boldsymbol{n}_a)+(\boldsymbol{m}\cdot\boldsymbol{n})_b(\boldsymbol{\varepsilon}_b\cdot\boldsymbol{n}_a)+(\boldsymbol{m}\cdot\boldsymbol{n})_c(\boldsymbol{\varepsilon}_c\cdot\boldsymbol{n}_a)&=(\boldsymbol{m}\cdot\boldsymbol{n})_a\\
(\boldsymbol{m}\cdot\boldsymbol{n})_a(\boldsymbol{\varepsilon}_a\cdot\boldsymbol{n}_b)+(\boldsymbol{m}\cdot\boldsymbol{n})_b(\boldsymbol{\varepsilon}_b\cdot\boldsymbol{n}_b)+(\boldsymbol{m}\cdot\boldsymbol{n})_c(\boldsymbol{\varepsilon}_c\cdot\boldsymbol{n}_b)&=(\boldsymbol{m}\cdot\boldsymbol{n})_b\\
(\boldsymbol{m}\cdot\boldsymbol{n})_a(\boldsymbol{\varepsilon}_a\cdot\boldsymbol{n}_c)+(\boldsymbol{m}\cdot\boldsymbol{n})_b(\boldsymbol{\varepsilon}_b\cdot\boldsymbol{n}_c)+(\boldsymbol{m}\cdot\boldsymbol{n})_c(\boldsymbol{\varepsilon}_c\cdot\boldsymbol{n}_c)&=(\boldsymbol{m}\cdot\boldsymbol{n})_c
\end{aligned} \tag{3.58}$$

This is an underdetermined system of three equations in six unknowns. The optimal (minimum-norm) solution can be found by computing the Moore-Penrose pseudoinverse of the row matrix of the edge unit normals. That is

$$\begin{bmatrix} \boldsymbol{\varepsilon}_a\cdot\boldsymbol{i} & \boldsymbol{\varepsilon}_a\cdot\boldsymbol{j} \\ \boldsymbol{\varepsilon}_b\cdot\boldsymbol{i} & \boldsymbol{\varepsilon}_b\cdot\boldsymbol{j} \\ \boldsymbol{\varepsilon}_c\cdot\boldsymbol{i} & \boldsymbol{\varepsilon}_c\cdot\boldsymbol{j} \end{bmatrix} = \begin{bmatrix} \boldsymbol{n}_a & \boldsymbol{n}_b & \boldsymbol{n}_c \end{bmatrix}^+ \tag{3.59}$$

The above method can be used both to reconstruct the momentum at the circumcentre of the cell and to reconstruct a component of the momentum at the centres of the edge. For the latter case, however, the stability can be improved by removing the edge itself from the stencil, and using only the other two edges bounding the cell. For edge $e$ in Figure 3.9, for example, edges $a$ and $b$ would be in the stencil for the left state and edges $c$ and $d$ would be in the stencil for the right state. This has the effect of increasing the upwind bias of the reconstruction.

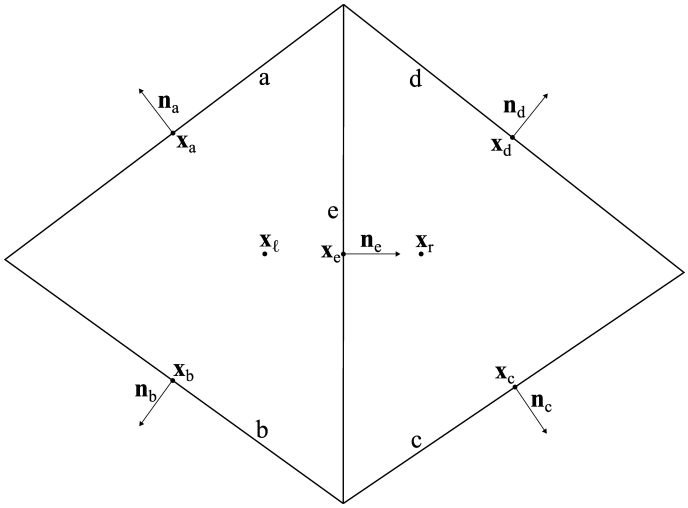



Figure 3.9: A pair of typical stencils for piecewise constant momentum reconstructions based at the centre of edge $e$. The left state reconstruction uses the normal momenta at $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$, whilst the right state reconstruction uses the normal momenta at $\boldsymbol{x}_c$ and $\boldsymbol{x}_d$.

With this alternate stencil definition, a similar solution to (3.59) is obtained, except that the system is now determinate. The reconstruction coefficients for the left state are therefore simply

$$\begin{bmatrix} \boldsymbol{\varepsilon}_a \cdot \boldsymbol{i} & \boldsymbol{\varepsilon}_a \cdot \boldsymbol{j} \\ \boldsymbol{\varepsilon}_b \cdot \boldsymbol{i} & \boldsymbol{\varepsilon}_b \cdot \boldsymbol{j} \end{bmatrix} = \begin{bmatrix} \boldsymbol{n}_a & \boldsymbol{n}_b \end{bmatrix}^{-1} \tag{3.60}$$

and similarly for the right state. Note that the matrix $\begin{bmatrix} \boldsymbol{n}_a & \boldsymbol{n}_b \end{bmatrix}$ is guaranteed to be non-singular due to the geometry of the stencil, as $\boldsymbol{n}_a$ and $\boldsymbol{n}_b$ will always be linearly independent if $a$ and $b$ are two distinct bounding edges of a valid triangle.

It is also possible to include additional components of the momentum where they are specified by the boundary conditions. For example, the tangential momentum along edge $a$, $(\boldsymbol{m} \cdot \boldsymbol{t})_a$, could be specified. This was found to be of little benefit for the piecewise constant reconstruction, however, whilst complicating the procedure somewhat. Hence this was not done for this reconstruction, but it was found to be crucial for the piecewise linear momentum field reconstruction, which is the topic of the next section.

A final comment regarding piecewise constant edge centre momentum reconstruction should be made. When solving the normal momentum equation for edge $a$ in Figure 3.9, for example, the left-state reconstruction of $m_e \cdot \boldsymbol{n}_a$ will be required. The reconstruction has a two-point stencil consisting of edges $a$ and $b$. It is therefore possible

to simply postulate $(m_e \cdot \boldsymbol{n}_a)_L \approx (\boldsymbol{m} \cdot \boldsymbol{n})_a$. Upon testing, this method was found to provide an increase in stability, presumably due to the increased diagonal dominance of the implicit operator of the normal momentum equation, with no noticeable difference in accuracy.

### 3.9.2 Piecewise Linear Momentum Field Reconstruction

The piecewise constant momentum field reconstruction described in the previous section is only sufficient to yield a first-order-accurate solver. In order to signficantly increase the accuracy of the momentum reconstruction, at the expense of larger reconstruction stencils, a piecewise linear reconstruction can be used. This level of vector reconstruction will also be employed in the next chapter, when the flow velocity gradients will need to be reconstructed for the approximation of the viscous terms. A general piecewise linear momentum field reconstruction involves assuming the momentum field $\boldsymbol{m}(\boldsymbol{x})$ is approximated by the reconstructing polynomial $\bar{\boldsymbol{m}}(\boldsymbol{x}) : \mathbb{R}^2 \to \mathbb{R}^2$ defined by

$$\boldsymbol{m}(\boldsymbol{x}) \approx \bar{\boldsymbol{m}}(\Delta\boldsymbol{x}) \coloneqq \left\{ \begin{array}{l} a_x + b_x \Delta x + c_x \Delta y \\ a_y + b_y \Delta x + c_y \Delta y \end{array} \right\} \tag{3.61}$$

where again $\Delta\boldsymbol{x}$ is the position vector of $\boldsymbol{x}$ relative to the reconstruction point $\boldsymbol{x}_i$, $\Delta\boldsymbol{x} \coloneqq \boldsymbol{x} - \boldsymbol{x}_i$. Note that, unlike for scalar field reconstructions, this is not simply a linear correction to a known value.

The reconstruction considered used here is based on the *divergence–free* piecewise linear vector reconstruction described in [Vid08]. In this method, the staggered layout of the solver is used to compute a sufficiently accurate approximation to the momentum divergence in the neighbourhood of the reconstruction point. For example, for a reconstruction based at the circumcentre of cell $i$ in Figure 3.8, a suitable approximation $\bar{d}_i$ to the momentum divergence is

$$\nabla \cdot \boldsymbol{m}\big|_i \approx \bar{d}_i \coloneqq \frac{1}{\Omega_i} \sum_{e(i)} \left\{ \bar{l}_e \left( \boldsymbol{m} \cdot \boldsymbol{n} \right)_e \right\} \tag{3.62}$$

The supplied approximation to the momentum divergence is used by the reconstruction procedure to reduce the number of unknown reconstruction coefficients from six to five. Whilst this may seem like a modest reduction, it has a profound effect in practice. Whilst the interior nodes of an ideal grid have six attached edges this is not achievable in general, and many interior nodes of a non-ideal grid have only five attached edges. Without the above reduction from six unknowns to five, a significant fraction of the

interior nodes will not have enough attached edges to form a complete stencil. The result would be that many of the interior node reconstruction stencils would need to be extended, with a negative effect on the accuracy and stability of the scheme. Following the discussion in Section 2.2, it is important to remember that it is possible for there to be less than five edges attached to an interior node. However, this was not encountered for any of the grids generated for the numerical examples considered in this thesis. As discussed later, the reduction in the number of unknowns is also greatly beneficial when performing momentum reconstructions along the grid boundaries.

The approximation to the momentum divergence $\bar{d}_i$ is inserted into (3.61) by setting it equal to the divergence of the reconstruction polynomial

$$\nabla \cdot \bar{\boldsymbol{m}}\left(\Delta \boldsymbol{x}\right) = b_x + c_y = \bar{d}_i \tag{3.63}$$

Let

$$b \coloneqq b_x - \frac{\bar{d}_i}{2} \tag{3.64}$$

which yields

$$\begin{aligned} b_x &= \frac{\bar{d}_i}{2} + b \\ c_y &= \frac{\bar{d}_i}{2} - b \end{aligned} \tag{3.65}$$

Inserting the new reconstruction coefficient $b$ into (3.61) leads to the reconstruction polynomial

$$\boldsymbol{m}\left(\boldsymbol{x}\right) \approx \bar{\boldsymbol{m}}\left(\Delta \boldsymbol{x}, \bar{d}_i\right) \coloneqq \left\{ \begin{array}{l} a_x + b\Delta x + c_x \Delta y \\ a_y + b_y \Delta x - b\Delta y \end{array} \right\} + \frac{\bar{d}_i}{2}\Delta \boldsymbol{x} \tag{3.66}$$

where $\bar{m} : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^2$ is the divergence-fixed reconstruction polynomial.

There are three sensible choices for the reconstruction locations. For reconstructing the momentum at the centre of an edge, the stencil could be based at the edge centre itself, at the circumcentre of the appropriate adjacent cell, or at the vertex of the appropriate adjacent cell that is not an end node of the edge (mirroring the reconstruction location for the node-based scalar reconstruction discussed in Section 3.8.4). The third option leads to all stencil edges being on the same side as the state being reconstructed, which avoids problems with downwind information being used inappropriately, and it is therefore the option selected. A typical left state stencil is shown in Figure 3.10.

In order to determine the five unknown reconstruction coefficients, the reconstruction polynomial is set equal to the known normal momentum components in the stencil.

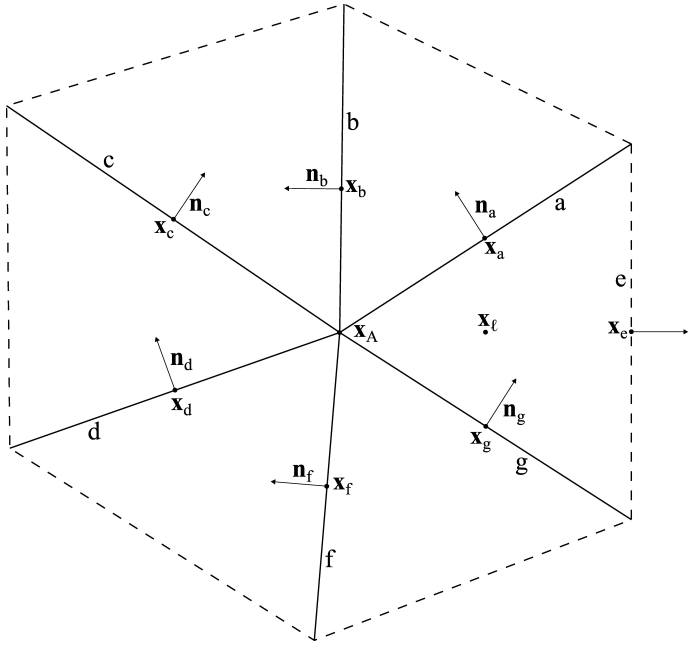



Figure 3.10: A typical stencil for the left state piecewise linear momentum reconstruction at the centre of edge $e$, based at node $A$. Only the solid edges are included in the stencil.

Letting $s$ denote an arbitrary stencil edge

$$\bar{\boldsymbol{m}}\left(\Delta\boldsymbol{x}_s, \bar{d}_i\right) \cdot \boldsymbol{n}_s = (\boldsymbol{m}\cdot\boldsymbol{n})_s \tag{3.67}$$

which yields the system of equations

$$\begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots \\ (\boldsymbol{n}_s\cdot\boldsymbol{i}) & (\boldsymbol{n}_s\cdot\boldsymbol{j}) & (\boldsymbol{n}_s\cdot\boldsymbol{i})\,\Delta x - (\boldsymbol{n}_s\cdot\boldsymbol{j})\,\Delta y & (\boldsymbol{n}_s\cdot\boldsymbol{i})\,\Delta y & (\boldsymbol{n}_s\cdot\boldsymbol{j})\,\Delta x \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{Bmatrix} a_x \\ a_y \\ b \\ c_x \\ b_y \end{Bmatrix}$$
$$= \begin{Bmatrix} \vdots \\ (\boldsymbol{m}\cdot\boldsymbol{n})_s \\ \vdots \end{Bmatrix} - \frac{\bar{d}_i}{2} \begin{Bmatrix} \vdots \\ (\Delta\boldsymbol{x}\cdot\boldsymbol{n})_s \\ \vdots \end{Bmatrix} \tag{3.68}$$

In addition, the system is augmented by adding an equation for each edge centre

tangential momentum component in the stencil that is specified by the boundary conditions. These equations are almost identical to those for the normal momentum components:

$$\begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots \\ (\boldsymbol{t}_s \cdot \boldsymbol{i}) & (\boldsymbol{t}_s \cdot \boldsymbol{j}) & (\boldsymbol{t}_s \cdot \boldsymbol{i})\,\Delta x - (\boldsymbol{t}_s \cdot \boldsymbol{j})\,\Delta y & (\boldsymbol{t}_s \cdot \boldsymbol{i})\,\Delta y & (\boldsymbol{t}_s \cdot \boldsymbol{j})\,\Delta x \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{Bmatrix} a_x \\ a_y \\ b \\ c_x \\ b_y \end{Bmatrix} = \begin{Bmatrix} \vdots \\ (\boldsymbol{m} \cdot \boldsymbol{t})_s \\ \vdots \end{Bmatrix} - \frac{\bar{d}_i}{2} \begin{Bmatrix} \vdots \\ (\Delta \boldsymbol{x} \cdot \boldsymbol{t})_s \\ \vdots \end{Bmatrix} \tag{3.69}$$

The reconstruction system is overdetermined in general, but is determinate when there are exactly five known momentum components in the stencil. For ease of reference, the system is rewritten as

$$\boldsymbol{M} \cdot \boldsymbol{r} = \boldsymbol{\chi} - \frac{\bar{d}_i}{2} \boldsymbol{g} \tag{3.70}$$

where $\boldsymbol{M}$ is called the *system matrix*, $\boldsymbol{r}$ is the *vector of reconstruction coefficients*, $\boldsymbol{\chi}$ is the *vector of known momentum components*, and $\boldsymbol{g}$ is the *vector of geometric knowns*.

With the system assembled, the vector of reconstruction coefficients is determined by computing the Moore-Penrose pseudoinverse of the system matrix. This can be done using either QR decomposition or singular value decomposition (SVD). The former procedure is significantly cheaper computationally but the latter has the advantage that it also provides the necessary information for computing the condition number of the reconstruction system, which in turn allows rank-deficient stencils to be identified. For the present solver rank deficient stencils can only occur at boundary nodes, and therefore the additional expense of SVD is only needed at boundary nodes. Rank deficiency is discussed in the next subsection.

With the pseudoinverse $\boldsymbol{M}^+$ computed, the vector of reconstruction coefficients can be written

$$\boldsymbol{r} = \boldsymbol{M}^+ \cdot \left( \boldsymbol{\chi} - \frac{\bar{d}_i}{2} \boldsymbol{g} \right) \tag{3.71}$$

Letting $\boldsymbol{M}^+_\mu$ denote row $\mu$ of the pseudoinverse, the pseudoinverse can be viewed as a

column vector of row vectors

$$\boldsymbol{M}^{+} \equiv \left\{ \begin{array}{c} \boldsymbol{M}_0^{+} \\ \boldsymbol{M}_1^{+} \\ \boldsymbol{M}_2^{+} \\ \boldsymbol{M}_3^{+} \\ \boldsymbol{M}_4^{+} \end{array} \right\} \tag{3.72}$$

and therefore the explicit expressions for the piecewise linear momentum reconstruction coefficients can be written as

$$\begin{aligned} a_x &= \boldsymbol{M}_0^{+} \cdot \boldsymbol{\chi} - \frac{\bar{d}_i}{2}(\boldsymbol{M}_0^{+} \cdot \boldsymbol{g}) \\ a_y &= \boldsymbol{M}_1^{+} \cdot \boldsymbol{\chi} - \frac{\bar{d}_i}{2}(\boldsymbol{M}_1^{+} \cdot \boldsymbol{g}) \\ b_x &= \boldsymbol{M}_2^{+} \cdot \boldsymbol{\chi} + \frac{\bar{d}_i}{2}(1 - \boldsymbol{M}_2^{+} \cdot \boldsymbol{g}) \\ b_y &= \boldsymbol{M}_4^{+} \cdot \boldsymbol{\chi} - \frac{\bar{d}_i}{2}(\boldsymbol{M}_4^{+} \cdot \boldsymbol{g}) \\ c_x &= \boldsymbol{M}_3^{+} \cdot \boldsymbol{\chi} - \frac{\bar{d}_i}{2}(\boldsymbol{M}_3^{+} \cdot \boldsymbol{g}) \\ c_y &= -\boldsymbol{M}_2^{+} \cdot \boldsymbol{\chi} + \frac{\bar{d}_i}{2}(1 + \boldsymbol{M}_2^{+} \cdot \boldsymbol{g}) \end{aligned} \tag{3.73}$$

As is the case with piecewise linear scalar field reconstruction, this reconstruction suffers from worsening ill-conditioning as the grid is refined and $\|\Delta \boldsymbol{x}_s\|_2 \to 0$. A similar use of scaling cures this problem. The characteristic length $\ell_{\text{char}}$ is chosen to be the mean average of the lengths of the edges included in the reconstruction stencil for the particular base point (note that a different characteristic length is therefore used for each base point). The scaled relative position vectors and reconstruction coefficients are then defined to be

$$\begin{aligned} \Delta \widetilde{\boldsymbol{x}} &\coloneqq \frac{\Delta \boldsymbol{x}}{\ell_{\text{char}}} \\ \widetilde{b} &\coloneqq \ell_{\text{char}} b \\ \widetilde{c_x} &\coloneqq \ell_{\text{char}} c_x \\ \widetilde{b_y} &\coloneqq \ell_{\text{char}} b_y \end{aligned} \tag{3.74}$$

The reconstruction coefficients $a_x$ and $a_y$ remain unscaled. The scaled approximation to the divergence is

$$\widetilde{d}_i \coloneqq \frac{1}{\widetilde{\Omega}} \sum_{e(i)} \left\{ (\boldsymbol{m} \cdot \boldsymbol{n})_e \, \widetilde{l}_e \right\} = \ell_{\text{char}} \bar{d}_i \tag{3.75}$$

However, the approximation to the divergence only appears in products with components of the vector of geometric knowns. The resulting terms are invariant under the scaling since

$$\widetilde{d}_i \left(\Delta\widetilde{\boldsymbol{x}} \cdot \boldsymbol{n}\right)_e = \ell_{\text{char}} \bar{d}_i \left(\frac{\Delta \boldsymbol{x}}{\ell_{\text{char}}} \cdot \boldsymbol{n}\right)_e = \bar{d}_i \left(\Delta \boldsymbol{x} \cdot \boldsymbol{n}\right)_e \tag{3.76}$$

The scaled reconstruction system therefore consists of equations of the form

$$\begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots \\ (\boldsymbol{n}_s \cdot \boldsymbol{i}) & (\boldsymbol{n}_s \cdot \boldsymbol{j}) & (\boldsymbol{n}_s \cdot \boldsymbol{i})\,\Delta\widetilde{x} - (\boldsymbol{n}_s \cdot \boldsymbol{j})\,\Delta\widetilde{y} & (\boldsymbol{n}_s \cdot \boldsymbol{i})\,\Delta\widetilde{y} & (\boldsymbol{n}_s \cdot \boldsymbol{j})\,\Delta\widetilde{x} \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{Bmatrix} a_x \\ a_y \\ \widetilde{b} \\ \widetilde{c}_x \\ \widetilde{b}_y \end{Bmatrix} = \begin{Bmatrix} \vdots \\ (\boldsymbol{m} \cdot \boldsymbol{n})_s \\ \vdots \end{Bmatrix} - \frac{\bar{d}_i}{2} \begin{Bmatrix} \vdots \\ (\Delta \boldsymbol{x} \cdot \boldsymbol{n})_s \\ \vdots \end{Bmatrix} \tag{3.77}$$

Finally, the unscaled reconstruction coefficients can be calculated from

$$\begin{aligned} a_x &= \hat{\boldsymbol{M}}_0^+ \cdot \boldsymbol{\chi} - \frac{\bar{d}_i}{2}\left(\hat{\boldsymbol{M}}_0^+ \cdot \boldsymbol{g}\right) \\ a_y &= \hat{\boldsymbol{M}}_1^+ \cdot \boldsymbol{\chi} - \frac{\bar{d}_i}{2}\left(\hat{\boldsymbol{M}}_1^+ \cdot \boldsymbol{g}\right) \\ b_x &= \hat{\boldsymbol{M}}_2^+ \cdot \boldsymbol{\chi} + \frac{\bar{d}_i}{2}\left(1 - \hat{\boldsymbol{M}}_2^+ \cdot \boldsymbol{g}\right) \\ b_y &= \hat{\boldsymbol{M}}_4^+ \cdot \boldsymbol{\chi} - \frac{\bar{d}_i}{2}\left(\hat{\boldsymbol{M}}_4^+ \cdot \boldsymbol{g}\right) \\ c_x &= \hat{\boldsymbol{M}}_3^+ \cdot \boldsymbol{\chi} - \frac{\bar{d}_i}{2}\left(\hat{\boldsymbol{M}}_3^+ \cdot \boldsymbol{g}\right) \\ c_y &= -\hat{\boldsymbol{M}}_2^+ \cdot \boldsymbol{\chi} + \frac{\bar{d}_i}{2}\left(1 + \hat{\boldsymbol{M}}_2^+ \cdot \boldsymbol{g}\right) \end{aligned} \tag{3.78}$$

where

$$\hat{\boldsymbol{M}}_\mu \coloneqq \begin{cases} \widetilde{\boldsymbol{M}}_\mu^+ & \mu < 2 \\ \frac{1}{\ell_{\text{char}}} \widetilde{\boldsymbol{M}}_\mu^+ & \text{otherwise.} \end{cases} \tag{3.79}$$

and $\widetilde{\boldsymbol{M}}_\mu^+$ denotes the rows of the pseudoinverse of the scaled reconstruction system.

Alternatively, the expressions for the reconstruction coefficients can be written in

the form

$$\begin{aligned} a_x &= \sum_{j(i)} M_{0j} m_j - \frac{\bar{d}_i}{2} g_0 \\ a_y &= \sum_{j(i)} M_{1j} m_j - \frac{\bar{d}_i}{2} g_1 \\ b_x &= \sum_{j(i)} M_{2j} m_j + \frac{\bar{d}_i}{2} (1 - g_2) \\ b_y &= \sum_{j(i)} M_{4j} m_j - \frac{\bar{d}_i}{2} g_4 \\ c_x &= \sum_{j(i)} M_{3j} m_j - \frac{\bar{d}_i}{2} g_3 \\ b_y &= -\sum_{j(i)} M_{2j} m_j + \frac{\bar{d}_i}{2} (1 + g_2) \end{aligned} \tag{3.80}$$

where $j(i)$ ranges over the momentum component equations specified in the stencil (rows of the system matrix), $m_j$ is the corresponding momentum component, $M_{\mu j} \coloneqq \left(\hat{\boldsymbol{M}}_\mu^+\right)_j$, and $g_\mu \coloneqq \left(\hat{\boldsymbol{M}}_\mu^+ \cdot \boldsymbol{g}\right)$.

**Rank-Deficient Stencils and Stencil Modification along Domain Boundaries**

The imposed grid requirements ensure that every interior node has at least five edges attached and that these edges are distributed fairly evenly around the node. As a result, the rank of the system (3.70) is guaranteed to be maximal. The situation is different along domain boundaries, where the stencil (which is constructed as the union of the edges attached to the base node) can fail to be of sufficient rank for two different reasons.

1. The stencil may yield fewer than five equations. In this case, the system is automatically rank-deficient since it is underdetermined. This situation cannot occur at interior nodes because the grid requirements mean that no interior angles can reach 90°, hence at least five edges are attached to every interior node.

2. Though the stencil may yield five or more equations, it may still be rank-deficient. An example of this is shown in Figure 3.11. These stencils occur when multiple stencil points are collinear. The result is that some information is multiply determined, leaving other information underdetermined. In the figure, for example, all information about $\boldsymbol{m} \cdot \boldsymbol{i}$ is located at the same $y$-coordinate. The

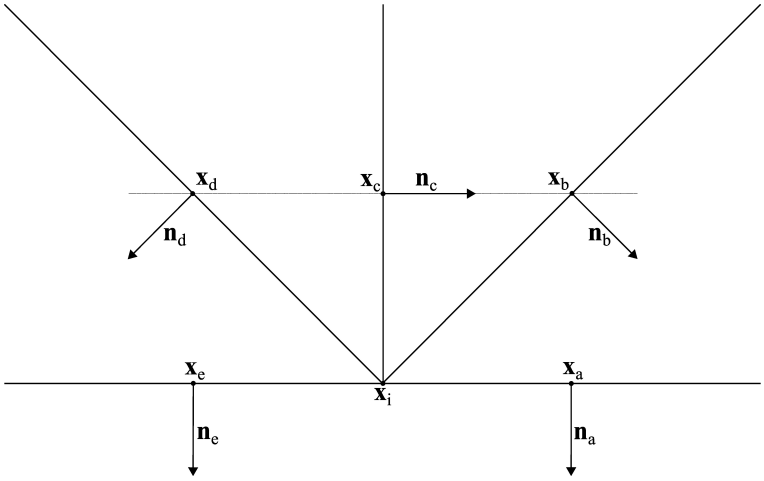



Figure 3.11: A rank-deficient stencil with five reconstruction equations. All information about $\boldsymbol{m} \cdot \boldsymbol{i}$ is located at the same $y$-coordinate, hence $\partial (\boldsymbol{m} \cdot \boldsymbol{i}) / \partial y$ cannot be determined.

result is that there is more than enough information to determine $\partial (\boldsymbol{m} \cdot \boldsymbol{i}) / \partial x$, but not enough information to determine $\partial (\boldsymbol{m} \cdot \boldsymbol{i}) / \partial y$. The fixing of $\nabla \cdot \boldsymbol{m}$ is unable to compensate, since it simply adds a relation between $\partial (\boldsymbol{m} \cdot \boldsymbol{j}) / \partial y$ and $\partial (\boldsymbol{m} \cdot \boldsymbol{i}) / \partial x$, both of which can already be determined. This situation cannot occur at interior nodes because in any situation where three attached edge centres are collinear, there must be at least three more edges attached to the node that have centres that are not collinear to the first three.

The solution is to extend rank-deficient stencils by adding new edges, which increases the number of equations in the system. This is done by adding any edges bounding a cell which is bounded by an edge already in the stencil, effectively adding a new layer to the stencil. Unfortunately, even after extending the stencil it may remain rank-deficient. Extending the stencil shown in Figure 3.11 results in the stencil shown in Figure 3.12. The extension has added new information about $\boldsymbol{m} \cdot \boldsymbol{i}$, but this new information is also collinear to the existing information, and so $\partial (\boldsymbol{m} \cdot \boldsymbol{i}) / \partial y$ still cannot be determined. It is therefore sometimes necessary to extend the stencil multiple times to obtain a satisfactory reconstruction.

Although extending rank-deficient stencils is necessary to obtain acceptable reconstruction systems, it has the undesirable effect of increasing numerical diffusion. The reason for this is obvious – a larger stencil allows information further away from the base node to influence the reconstructed momentum field. To limit this effect,

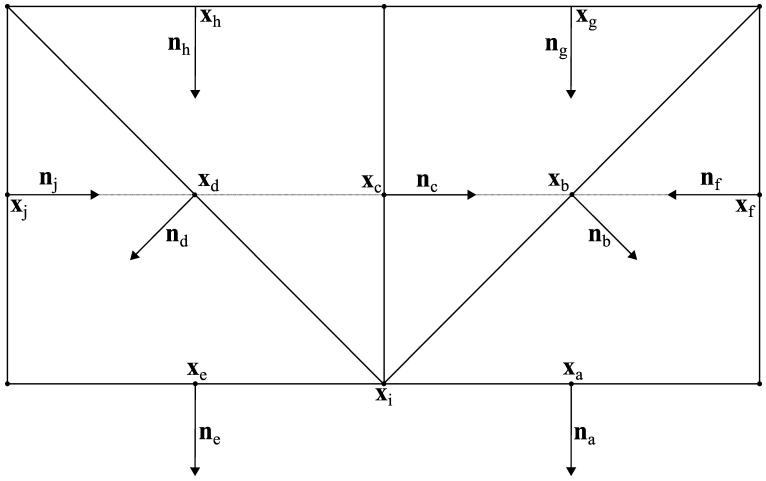



Figure 3.12: The result of extending the stencil of Figure 3.11 once. The new information about $\boldsymbol{m} \cdot \boldsymbol{i}$ is located at the same $y$-coordinate as the existing information, hence $\partial (\boldsymbol{m} \cdot \boldsymbol{i}) / \partial y$ still cannot be determined.

equations corresponding to those edges not in the original stencil can be weighted by some empirically-determined weight $w_M$. It was found that selecting $w_M = 1/100$ worked well for all the numerical cases considered. It is also possible to assign different weights to each layer of the stencil, but this was not found to be useful in practice.

The stencil shown in Figure 3.11 only results in a *formally* rank-deficient stencil if the three edge centres supplying information about $\boldsymbol{m} \cdot \boldsymbol{i}$ are exactly collinear (and the normals $\boldsymbol{n}_a$ and $\boldsymbol{n}_e$ are exactly perpendicular to the line of collinearity). However, consider the result of perturbing the position of $\boldsymbol{x}_c$ by some tiny distance $\delta \boldsymbol{x}_c = 10^{-20}$ in the positive y-direction. The system is no longer formally rank-deficient, but the determination of $\partial (\boldsymbol{m} \cdot \boldsymbol{i}) / \partial y$ is now based on a differencing of $\boldsymbol{m} \cdot \boldsymbol{i}$ involving a factor of $1/10^{-20} = 10^{20}$. A tiny change in one of the $\boldsymbol{m} \cdot \boldsymbol{i}$ values used in the differencing therefore results in a very large change in the resulting approximation to $\partial (\boldsymbol{m} \cdot \boldsymbol{i}) / \partial y$. This is a critical problem numerically, since even round-off error is likely to cause significant instability in the reconstruction. It is therefore required not only to extend stencils that are formally rank-deficient, but also those that are *numerically* rank-deficient. The correct test for rank-deficiency, then, is to compute the *condition number* of the assembled reconstruction system. For a system $\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{b}$, the condition number $\kappa$ is defined by

$$\kappa := \frac{\sup\limits_{\|\boldsymbol{x}\|_2 = 1} \|\boldsymbol{A} \cdot \boldsymbol{x}\|_2}{\inf\limits_{\|\boldsymbol{x}\|_2 = 1} \|\boldsymbol{A} \cdot \boldsymbol{x}\|_2} \tag{3.81}$$

A small perturbation $\delta\boldsymbol{b}$ in the vector of knowns induces a perturbation $\delta\boldsymbol{x}$ in the vector of unknowns. It can be shown [Nov22, p. 22] that $\delta\boldsymbol{x}$ is bounded by

$$\frac{\|\delta\boldsymbol{b}\|_2}{\|\boldsymbol{b}\|_2} \leq \kappa \frac{\|\delta\boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2} \tag{3.82}$$

The condition number can therefore be characterised as the maximum amplification applied to errors in the input provided to the system.

As mentioned above, the pseudoinverse of the reconstruction system is computed using SVD. Although this is computationally expensive, it has a key advantage in that it produces not only the pseudoinverse but also the set of singular values $s_i,\ i = \{0, ..., 4\}$ associated with the system matrix. This allows the condition number to be easily calculated via [Nov22, p. 22]

$$\kappa = \frac{\max\limits_i s_i}{\min\limits_i s_i} \tag{3.83}$$

Once the condition number has been computed, it is tested against some threshold $\kappa_{\max}$. If the condition number is too large, then the stencil is extended by one layer, and the process is repeated until a satisfactory stencil is obtained. A suitable value of $\kappa_{\max}$ has to be determined empirically. It was found that setting $\kappa_{\max} = 100/w_M$ produced good results for all the numerical examples considered in this thesis.

#### Approximation of the Momentum Divergence

As already discussed, the approximation of the momentum divergence supplied to the piecewise linear momentum field reconstruction procedure is obtained from a relatively simple approximation of the form (3.62). However, it turns out that the stability and accuracy of the scheme is strongly dependent on the selection of the control volume over which this approximation is computed, and thus this part of the reconstruction procedure needs to be carefully considered. This is done separately for each application of the momentum field reconstruction, and is discussed in Section 3.9.3.

#### Monotonicity

During the discussion of piecewise linear scalar field reconstruction in Section 3.8.2, it was pointed out that moving from piecewise constant reconstruction to piecewise linear reconstruction sacrificed the main benefit of the former – monotonicity preservation. The comments made in that discussion are also applicable to the case of piecewise linear momentum field reconstruction. Therefore it can be expected that new extrema will

be generated near discontinuities by this reconstruction, causing unphysical oscillations to appear in the solution. This is observed in practice to cause convergence problems and ultimately the complete breakdown of the solution process. The solution to this problem is the addition of *gradient limiting*, which is discussed in Section 4.4.

### 3.9.3 Edge Centre Piecewise Linear Momentum Reconstruction

The node-based piecewise linear momentum field reconstruction is used to reconstruct the momentum at the centres of the grid edges as well as at the circumcentres of the grid cells. The edge centre momentum reconstruction is used for two purposes: to implicitly reconstruct the convected normal momentum $\boldsymbol{m}_e \cdot \boldsymbol{n}_i$ when solving the normal momentum equation for edge $i$, and to explicitly reconstruct the edge centre tangential momenta $(\boldsymbol{m} \cdot \boldsymbol{t})_e$. The tangential momenta are used for computing any required edge centre secondary flow variables such as $(\boldsymbol{m} \cdot \boldsymbol{m})_e^*$. In turn, these are used when assembling the pressure correction equation and when updating the boundary conditions at the end of each iteration. The stencils used for these purposes are identical, and are constructed in a similar manner to those used in the node-based piecewise linear scalar field reconstruction – the reconstruction is based at the node opposite the target edge on the appropriate side. Refer back to Figure 3.10 for an example of a left state stencil.

There are two sensible choices for reconstructing the convected normal momentum $\boldsymbol{m}_e \cdot \boldsymbol{n}_i$ during the solution of the normal momentum equation for edge $i$. These are discussed in Section A.2.4 of Appendix A. The method selected is to reconstruct only the tangential component of momentum, retaining the stored value of the edge centre normal momentum. The convected normal momentum at the centre of edge $e$ is therefore approximated by

$$\boldsymbol{m}_e \cdot \boldsymbol{n}_i \approx (\boldsymbol{m}_e \cdot \boldsymbol{n}_e)(\boldsymbol{n}_e \cdot \boldsymbol{n}_i) + (\bar{\boldsymbol{m}}_e \cdot \boldsymbol{t}_e)(\boldsymbol{t}_e \cdot \boldsymbol{n}_i) \tag{3.84}$$

With the reconstruction procedure selected, all that remains is to identify the control volume used for approximating the momentum divergence. The four reasonable options are discussed in Section A.2.5 of Appendix A. The option that was ultimately selected involves defining the control volume to be the cell that has edge $e$ as a side and the base node as a vertex, as shown in Figure 3.13.

The second use for edge centre piecewise linear momentum reconstruction is to reconstruct the edge centre tangential momenta after solving the normal momentum equation and after correcting the momentum predictions. This is almost identical to

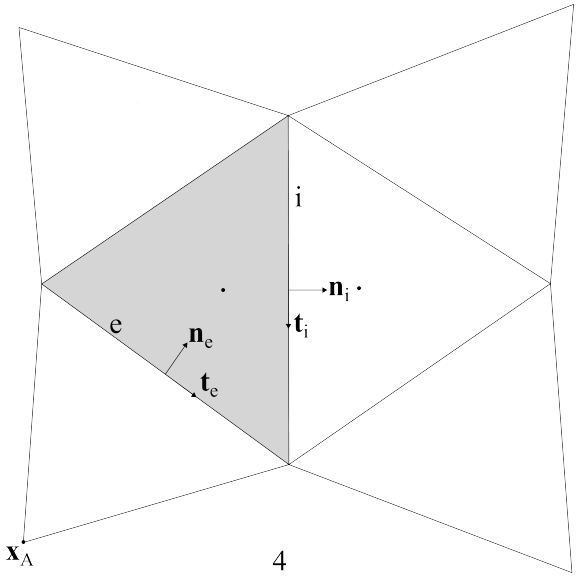



Figure 3.13: The definition of the control volume (represented by the shaded area) used to approximate the momentum divergence when performing piecewise linear reconstruction of the left state convected normal momentum $(\boldsymbol{m}_e \cdot \boldsymbol{n}_i)_L$ for edge $e$ in the normal momentum equation for edge $i$.

the reconstruction of the tangential momenta for use in approximating the convected quantities in the normal momentum equation, except for two differences. The first difference is that the reconstruction is now explicit rather than implicit, whilst the second is that the control volume used for approximating the momentum divergence is now not defined due to the lack of an edge $i$. The first difference requires no special consideration. For the second, there are two sensible choices for control volume used to approximate the momentum divergence. These are discussed in Section A.2.6 of Appendix A. The option

### 3.9.4 Cell Centre Piecewise Linear Momentum Reconstruction

Finally, the momentum must be reconstructed at the cell circumcentres. This is used explicitly in the pressure correction equation to compute $(\boldsymbol{m} \cdot \boldsymbol{m})_i^*$ at the cell circumcentres, and also to compute other secondary flow variables such as the local enthalpy, sonic velocity and Mach number. There are three reasonable choices for how to perform this reconstruction.

1. Construct a reconstruction system based at the circumcentre of each cell.

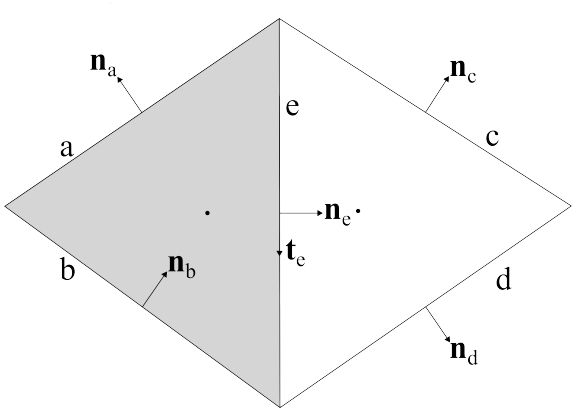



Figure 3.14: Definition of the control volume (represented by the shaded area) for use when performing explicit piecewise linear reconstruction of $(\boldsymbol{m} \cdot \boldsymbol{t})_e$.

2. Set the cell circumcentre momentum reconstruction as the mean average of the reconstructions obtained from the reconstruction systems based at each of the vertices of the cell.

3. Use some method to select one of the vertices of the cell, then reconstruct the cell circumcentre momentum using the reconstruction system based at that vertex.

The third option has a serious drawback in that there is no obvious reason to pick one vertex over another. There is perhaps utility in computing the explicit reconstruction at each vertex, then using the reconstruction that produces the momentum with the median norm. However, this seems artificial (this is not necessarily a disadvantage in general, as long as there is a reasonably strong theoretical or practical basis) and the reconstruction at some cell circumcentres would likely switch rapidly between vertices, which could introduce some instability to the scheme. Using a reconstruction based at the cell circumcentres and using the average of the values obtained from the node-based reconstructions were both found to perform reasonably well, but the former carries additional computational and storage requirements. The latter option is therefore likely the best choice. It is also possible to only use those vertices that are not located on the domain boundary, as the reconstruction stencils based at nodes lying on the domain boundary are generally larger and more complex than those based at the interior nodes. This is not noticeably beneficial in practice.

Unlike for the edge centre case, where there were multiple reasonable choices, there is an obvious choice for the control volume used to approximate the momentum divergence at the cell circumcentre – the cell itself.

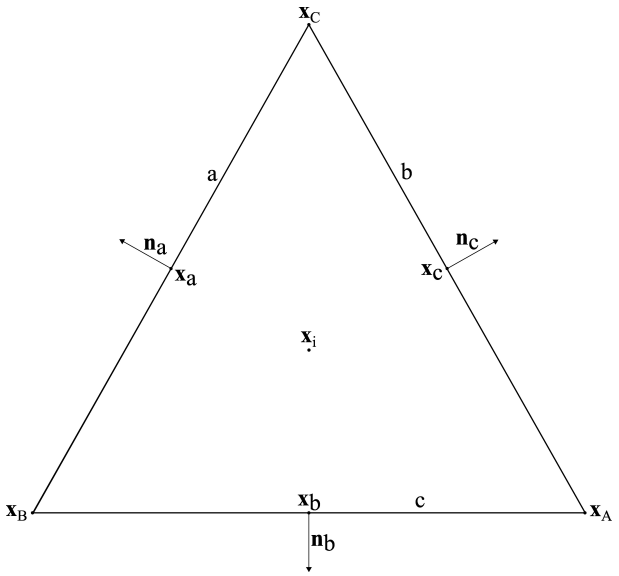



Figure 3.15: Layout for piecewise linear momentum reconstruction at the circumcentre of cell $i$.

The momentum reconstruction at the circumcentre of the cell $i$ in Figure 3.15 is therefore evaluated from

$$\boldsymbol{m}_i \approx \frac{1}{3}\left\{\bar{m}_A\left(\boldsymbol{x}_i - \boldsymbol{x}_A, \bar{d}_i\right) + \bar{m}_B\left(\boldsymbol{x}_i - \boldsymbol{x}_B, \bar{d}_i\right) + \bar{m}_C\left(\boldsymbol{x}_i - \boldsymbol{x}_C, \bar{d}_i\right)\right\} \tag{3.85}$$

where

$$\bar{d}_i \coloneqq \sum_{e(i)} \left\{\bar{l}_e \left(\boldsymbol{m} \cdot \boldsymbol{n}\right)_e\right\} \tag{3.86}$$

and $e(i) \coloneqq \{a, b, c\}$.

# Chapter 4

# Further Development of the Inviscid Solver

## 4.1 Introduction

In the previous chapter the adaptation of the solver of [WSW02] to utilise the special grid requirements imposed in this work was discussed, yielding the basic framework of the inviscid solver. In this chapter, the inviscid solver is completed and extended through consideraton of topics not discussed in the above reference.

## 4.2 Convergence Testing

The discrete residual of a conserved flow variable $\varphi$ at location $i$ and time level $n+1$ is defined to be

$$R_{\varphi,i}^{n+1} := \frac{\Omega_i \left(\varphi_i^{n+1} - \varphi_i^n\right)}{\Delta t_i} \tag{4.1}$$

and is a measure of how much the local solution changed during the iteration. Equivalently, the residual could be defined to be equal to the right-hand side of the corresponding conservation equation and characterised as the rate of generation or destruction of $\varphi$ within the control volume.

To obtain a convergence measure for the whole grid, the overall residual is defined to be the $\mathcal{L}_2$-norm of the residuals at all the relevant grid elements (over all the grid

cells for density and pressure, and over all the grid edges for normal momentum)

$$R_{\varphi}^{n+1} \coloneqq \sqrt{\sum_{j} \left( R_{\varphi,j}^{n+1} \right)^2} \tag{4.2}$$

The residual is unsatisfactory as a convergence measure in this form, however, as it has no clear interpretation in terms of progress towards convergence. Instead, a suitable measure of convergence is the number of orders of magnitude the residual has been reduced compared to its maximum during the first $M$ iterations

$$\Theta_{\varphi}^{n+1} \coloneqq \log_{10} \left[ \frac{R_{\varphi}^{n+1}}{\max\limits_{m \in [0,M]} R_{\varphi}^{m}} \right] \tag{4.3}$$

$\Theta_{\varphi}$ is referred to as the *convergence indicator* for $\varphi$. A suitable value of $M$ is determined empirically. For the inviscid solver, $M = 5$ was found to be a reasonable choice. The principal reason the convergence indicators are normalised by the maximum residuals over the first $M$ iterations, rather than simply being normalised by the initial residuals, is the fact that the solver is segregated rather than coupled. The initial residuals are therefore strongly dependent on the initial conditions, since it is possible, for example, for the imposed initial conditions to closely satisfy the continuity equation but not the other conservation equations. The initial density residual would then be abnormally small, resulting in an unreasonably large normalised density convergence indicator which would be of limited use in determining convergence. By instead taking the maximum residuals over the first $M$ iterations as the normalisation factors, this strong reliance of the convergence indicators on the initial conditions is ameliorated.

The solution is considered to be converged when the convergence indicators of all three primary flow variables fall below some threshold. For the numerical solutions obtained using the inviscid solver, this threshold was set to $-5.5$.

For some flow cases, aerodynamic coefficients can also be used for convergence testing. For example, when computing the flow over an aerofoil, the coefficients of lift, drag, and pitching moment can be monitored. This is of some use, as it can be directly related to the solution accuracy. For example, convergence can be continued until the aerodynamic coefficients are constant to a given number of significant figures over a specified number of iterations. For the applicable numerical examples in this chapter, this was used as an additional confirmation of convergence.

## 4.3 Reconstruction of the Convective Fluxes

The reconstruction of the edge centre convective fluxes appearing in the discretised governing equations presented in Section 3.6 is now reconsidered. There is a multitude of methods available in the literature for performing these reconstructions for use in CFD. Two such methods were adapted and tested for use with the solver, the simple fully upwind method presented in Section 3.7 and a significantly more complex alternative reconstruction. The primary purpose of doing this was to evaluate the effect of different choices of convective flux reconstruction on the stability of the solver.

Most methods of convective flux reconstruction, including both methods considered here, involve interpolation (or extrapolation) of the flow variables to the edge centre from either side of the edge. Following the notation introduced in Section 3.1, the interpolated information from the direction of the left (right) adjacent cell is referred to as the *left* (*right*) *state*. That is, the edge unit normal points towards the source of the information used to interpolate the right state. An arbitrary flow variable obtained from the left and right state is denoted $\varphi_L$ and $\varphi_R$ respectively – compare with the notation $\varphi_\ell$ and $\varphi_r$ used to denote the value of $\varphi$ at the left and right cell circumcentres. If the flow at the edge centre is in the direction of the edge unit normal, then the left state will consist of information interpolated primarily from upwind of the edge centre, and the right state will consist of information interpolated primarily from downwind of the edge centre. The left and right states could then be referred to as the *upwind* and *downwind* states respectively, and vice-versa. If the normal flow velocity at the edge centre is exactly zero, then the left state is defined to be "upwind".

Convective flux reconstruction methods can be broadly categorised into two groups:

1. *Central* methods put equal weight on information from the upwind and downwind states. This method is easy to implement and computationally cheap. This is especially true given the properties of the grids being used with the present solver, as second-order accuracy would be obtainable from two-point stencils. Unfortunately, this scheme is found to be unstable due to odd-even decoupling, and requires the addition of artificial (numerical) dissipation to stabilise it [Bla01, p. 95]. This artifical dissipation tends to smear shocks and boundary layers.

2. *Upwind* methods bias the reconstruction towards information from the upwind state, though the downwind state is not necessarily neglected completely. Methods of this type are more expensive and complex than central schemes, but are found to offer superior stability and accuracy, particularly in the presence of

shocks.

Since the key design requirement of the solver is good performance over a large range of Mach numbers, only upwind methods are considered further.

### 4.3.1 H-CUSP

The *Convective Upwind Split Pressure* (CUSP) method was first introduced in [Jam93]. The particular variant on which the procedure developed here is based is called the H-CUSP method, as it preserves total enthalpy [TMJ95]. Both the general CUSP and specific H-CUSP method use the collocated primary flow variable arrangement that is standard in compressible solvers, so some modifications are required before implementation in the present solver. These modifications are both simple and natural, however, making H-CUSP an ideal choice for inclusion in the present solver.

In the original formulation, the numerical flux function defined by the method is written as the average of the left and right fluxes minus a dissipation term

$$\hat{\mathcal{F}}_e\left(\varphi_L, \varphi_R\right) \coloneqq \frac{1}{2}\left[\hat{f}_e\left(\varphi_R\right) + \hat{f}_e\left(\varphi_L\right)\right] - \hat{\mathcal{D}}_e\left(\varphi_L, \varphi_R\right) \tag{4.4}$$

where $\hat{f}_e : \mathbb{R} \to \mathbb{R}$, $\hat{f}_e\left(\varphi_{L/R}\right) = \left(\boldsymbol{u}_{L/R} \cdot \boldsymbol{n}_e\right)\varphi_{L/R}$ is the one-sided numerical flux function and $\hat{\mathcal{D}}_e : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is the numerical dissipation function. In the original formulation, this dissipation term is defined as

$$\hat{\mathcal{D}}_e\left(\varphi_L, \varphi_R\right) \coloneqq \frac{1}{2}\left(\alpha^* a\right)_e\left(\varphi_R - \varphi_L\right) + \frac{1}{2}\beta_e\left[\left(\boldsymbol{u}_R \cdot \boldsymbol{n}_e\right)\varphi_R - \left(\boldsymbol{u}_L \cdot \boldsymbol{n}_e\right)\varphi_L\right] \tag{4.5}$$

where the upwinding coefficients $\left(\alpha^* a\right)_e$ and $\beta_e$ will be defined shortly.

In the present solver, the normal momenta are solved for at the edge centres. In order to apply the H-CUSP reconstruction directly, then, the momenta would have to be interpolated from the edge centres to the cell circumcentres, then the flow velocities calculated, and finally the flow velocities would have to be interpolated back to the edge centres. This is likely to lead to significant error and numerical dissipation. By contrast, the edge centre normal flow velocities are easily obtained using the procedure detailed in Section 3.6.5. This suggests that H-CUSP be modified using

$$\begin{aligned}
\hat{\mathcal{F}}_e\left(\varphi_L, \varphi_R\right) &\coloneqq \frac{1}{2}\left(\boldsymbol{u} \cdot \boldsymbol{n}\right)_e\left(\varphi_R + \varphi_L\right) - \hat{\mathcal{D}}_e\left(\varphi_L, \varphi_R\right) \\
\hat{\mathcal{D}}_e\left(\varphi_L, \varphi_R\right) &\coloneqq \frac{1}{2}\left(\alpha^* a\right)_e\left(\varphi_R - \varphi_L\right) + \frac{1}{2}\beta_e\left(\boldsymbol{u} \cdot \boldsymbol{n}\right)_e\left(\varphi_R - \varphi_L\right)
\end{aligned} \tag{4.6}$$

The upwinding coefficients $(\alpha^* a)_e$ and $\beta_e$ are calculated using the Roe-averaged flow variables [Roe81]

$$\begin{aligned} \tilde{\boldsymbol{u}}_e &\coloneqq \frac{\boldsymbol{u}_L\sqrt{\rho_L} + \boldsymbol{u}_R\sqrt{\rho_R}}{\sqrt{\rho_L} + \sqrt{\rho_R}} \\ \tilde{H}_e &\coloneqq \frac{H_L\sqrt{\rho_L} + H_R\sqrt{\rho_R}}{\sqrt{\rho_L} + \sqrt{\rho_R}} \\ \tilde{a}_e &\coloneqq \frac{\sqrt{\gamma-1}}{M_r}\sqrt{\tilde{H}_e - \frac{1}{2}\left\|\tilde{\boldsymbol{u}}_e\right\|_2^2} \end{aligned} \tag{4.7}$$

The eigenvalues $\Lambda^{\pm}$ of the approximate convective flux Jacobian are given by

$$\Lambda^{\pm} = \frac{\gamma+1}{2\gamma}\left(\boldsymbol{u}\cdot\boldsymbol{n}\right)_e \pm \sqrt{\left(\frac{\gamma-1}{2\gamma}\left(\boldsymbol{u}\cdot\boldsymbol{n}\right)_e\right)^2 + \frac{\tilde{a}_e^2}{\gamma}} \tag{4.8}$$

and the edge centre advection Mach number is calculated from

$$M_{n_e} \coloneqq \frac{\left(\boldsymbol{u}\cdot\boldsymbol{n}\right)_e}{\tilde{a}_e} \tag{4.9}$$

The upwinding coefficients are

$$\begin{aligned} \left(\alpha^* a\right)_e &\coloneqq \begin{cases} \left|\left(\boldsymbol{u}\cdot\boldsymbol{n}\right)_e\right| & \text{if } \beta_e = 0 \\ -\left(1+\beta_e\right)\Lambda^- & \text{if } \beta_e > 0 \ \wedge \ 0 < M_{n_e} < 1 \\ +\left(1-\beta_e\right)\Lambda^+ & \text{if } \beta_e < 0 \ \wedge \ -1 < M_{n_e} < 0 \\ 0 & \text{otherwise.} \end{cases} \\ \beta_e &\coloneqq \begin{cases} \max\left(0, \frac{\left(\boldsymbol{u}\cdot\boldsymbol{n}\right)_e + \Lambda^-}{\left(\boldsymbol{u}\cdot\boldsymbol{n}\right)_e - \Lambda^-}\right) & \text{if } 0 \le M_{n_e} < 1 \\ -\max\left(0, \frac{\left(\boldsymbol{u}\cdot\boldsymbol{n}\right)_e + \Lambda^+}{\left(\boldsymbol{u}\cdot\boldsymbol{n}\right)_e - \Lambda^+}\right) & \text{if } -1 \le M_{n_e} < 0 \\ \operatorname{sgn}\left(M_{n_e}\right) & \text{otherwise.} \end{cases} \end{aligned} \tag{4.10}$$

where

$$\operatorname{sgn}\left(\varphi\right) \coloneqq \begin{cases} +1 & \text{if } \varphi > 0 \\ -1 & \text{if } \varphi < 0 \\ 0 & \text{if } \varphi = 0 \end{cases} \tag{4.11}$$

though the third possibility cannot happen in (4.10).

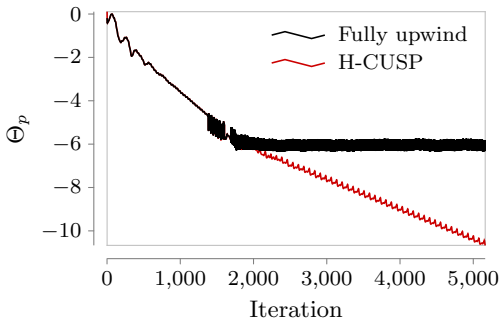



Figure 4.1: Pressure convergence indicator histories for the NACA 0012 at $M_r = 0.5$, $\alpha = 4°$ using the two convective flux reconstruction methods discussed in this section. All other parameters are the same for both cases.

### 4.3.2 Comparison of Methods

Compared to the fully upwind reconstruction method, the H-CUSP method presented here is more computationally expensive (per iteration). Several of the test cases for both the inviscid solver discussed in the current chapter and the viscous solver discussed in the next chapter were run with both methods, and it was found that using the H-CUSP method increased total runtime by about 10%. On the other hand, it was also found that the H-CUSP method lead to a more stable solver. For the NACA 0012 symmetric aerofoil discussed in Section 4.8.1 at a Mach number of $M_r = 0.5$ and angle of attack $\alpha = 4°$, convergence is found to stall when using the fully upwind method, but not when using the H-CUSP method. This is shown by the pressure convergence indicator history plots of Figure 4.1.

Additionally, in some cases it is found that the H-CUSP method leads to faster convergence. This tends to occur if the selected Courant number is in the upper end of the stable range. For the NACA 0012 aerofoil at $M_r = 0.8$, $\alpha = 1.25°$, and a Courant number of 5, the pressure convergence indicator history plot shown in Figure 4.2 results from the use of the two methods.

## 4.4 Gradient Limiting

As briefly described in Sections 3.8.2 and 3.9.2, the inclusion of linear terms in the flow variable field reconstruction polynomials means that the reconstruction polynomials are not monotonicity-preserving. As a result they allow the generation of spurious extrema in the interior of the solution domain, which leads to unphysical numerical oscillations in the neighbourhood of discontinuities (such as shocks and sharp corners in the boundary

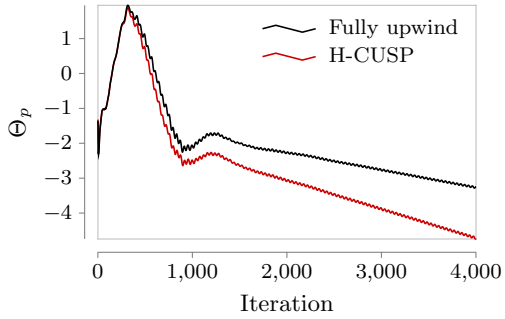



Figure 4.2: Pressure convergence indicator histories for the NACA 0012 at $M_r = 0.8$, $\alpha = 1.25°$ using the two convective flux reconstruction methods discussed in this section. All other parameters are the same for both cases.

of the flow domain) that can cause the solution process to stall or break down. The most popular method for controlling this phenomenon is including a *gradient limiter* (or a *flux limiter*, which is a closely related concept) in the reconstruction. In the ideal case, this essentially takes the form of a switch which 'turns off' the higher-order terms near discontinuities, without affecting the reconstruction in smooth regions of the flow domain. This ideal limiter would be impractical in a real scheme for two main reasons. First, the identification of smooth regions of the flow domain is not precise, and often uses semi-heuristic methods heavily based on numerical experimentation. Second, the sudden switch from a first-order scheme to a higher-order scheme would itself be a discontinuity in the solution, and may therefore cause unphysical oscillations.

For the present work, two limiters are considered. The first is the relatively simple limiter of Barth and Jespersen [BJ89]. Whilst simple and quite cheap to compute, it can cause convergence to stall, due to its non-differentiability [TA05]. A similar but more complex limiter function, Venkatakrishnan's limiter [Ven95], solves this problem, whilst also being less prone to activation in smooth regions of the flow field [Bla01, p. 166]. On the other hand, it is more expensive to compute. Both limiters were implemented in the solver.

The structure of the present scheme makes the design and implementation of gradient limiting slightly more difficult. Specifically, whilst the gradient limiting of the scalar field reconstruction schemes is similar to that for other schemes, the gradient limiting of the momentum field reconstruction requires some modification.

### 4.4.1 Gradient Limiting in Scalar Field Reconstructions

There is a wide array of gradient limiters for scalar field reconstructions described in the literature, but they all follow a similar procedure. First, a limiter function $\Psi_i$ is computed at the base $i$ of each gradient reconstruction. Then, the expression for the reconstructed scalar field (3.44) is modified by applying the limiter function to the reconstructed scalar gradient. The modified scalar field reconstruction is written

$$\begin{aligned} \varphi_L &= \varphi_\ell + \Psi_L \bar{\nabla\varphi}\big|_L \cdot (\boldsymbol{x}_e - \boldsymbol{x}_\ell) \\ \varphi_R &= \varphi_r + \Psi_R \bar{\nabla\varphi}\big|_R \cdot (\boldsymbol{x}_e - \boldsymbol{x}_r) \end{aligned} \tag{4.12}$$

The gradient limiter functions may take any value in the closed interval $[0, 1]$. From (4.12) it can be seen that when the gradient limiter is fully 'on' ($\Psi = 0$), the reconstruction is reduced to the first-order-accurate piecewise constant reconstruction of (3.43), whilst when the gradient limiter is fully 'off' ($\Psi = 1$), the second-order-accurate piecewise linear reconstruction is recovered.

In the present solver, both node-based and cell-circumcentre-based scalar field reconstructions are used. As a result, two slightly different procedures are required for computing the limiter functions. The differences between these procedures are noted as they occur. The computation of the node-based scalar gradient reconstruction limiters are described with reference to Figure 3.6, whilst the cell-based versions are described with reference to Figure 4.3. Both are written for an arbitrary scalar field $\varphi(\boldsymbol{x})$.

The procedure for the computation of the limiter at an element $i$ (which may be a node or a cell circumcentre) begins with the calculation of the minimum and maximum values of $\varphi$ in the stencil

$$\begin{aligned} \varphi_{\max} &\coloneqq \max\left(\varphi_i, \max_{j(i)} \varphi_j\right) \\ \varphi_{\min} &\coloneqq \min\left(\varphi_i, \min_{j(i)} \varphi_j\right) \end{aligned} \tag{4.13}$$

where $j(i)$ is the set of cells adjacent to the element $i$ (e.g. $j(i) = \{0, 1, 2, 3, 4, 5\}$ for the node-based example and $j(i) = \{0, 1, 2\}$ for the cell-based example). A measure of the reconstructed change in the value of $\varphi$ across the stencil is computed. For the node-based case, this is given by

$$\Delta_{2,j} = \nabla\varphi\big|_i \cdot \Delta\boldsymbol{x}_{i,j} \tag{4.14}$$

where $\Delta\boldsymbol{x}_{i,j} \coloneqq \boldsymbol{x}_j - \boldsymbol{x}_i$ is the vector from the base point $i$ to the adjacent cell $j$. For

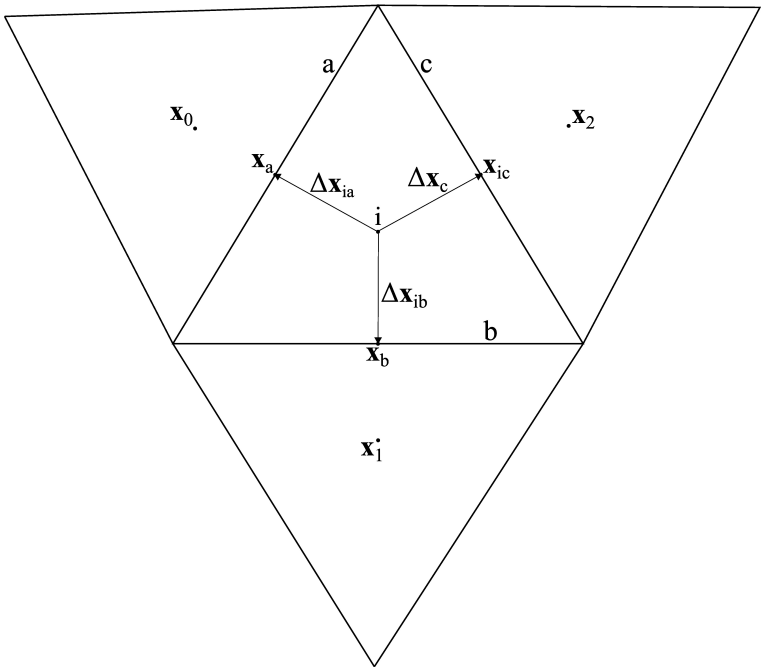



Figure 4.3: Layout for the computation of the piecewise linear scalar field reconstruction limiter function at the circumcentre of cell $i$.

the cell-based case, $\Delta_{2,j}$ is instead defined by

$$\Delta_{2,j} = \nabla\varphi\big|_i \cdot \Delta\boldsymbol{x}_{i,e(j)} \tag{4.15}$$

where $\Delta\boldsymbol{x}_{i,e(j)}$ is the vector from the circumcentre of cell $i$ to the centre of the edge $e(j)$ separating cell $i$ from cell $j$.

In fact, it is the reciprocal of $\Delta_{2,j}$ that will be used to compute the limiters. This can lead to a numerical issue [Ven93]. The problem occurs due to the limited floating-point precision of digital computers, which can cause instability and accumulating inaccuracy when dividing by a very small but nonzero number. This problem can be avoided by replacing $\Delta_{2,j}$ by

$$\Delta'_{2,j} \coloneqq \operatorname{sgn}(\Delta_{2,j})\,(\Delta_{2,j} + \omega) \tag{4.16}$$

where $\omega$ is a constant of order machine zero.

**Limiter of Barth and Jespersen**

A sub-limiter $\psi$ is computed for each adjacent cell

$$\psi_{i,j} = \begin{cases} \min\left(1, \frac{\varphi_{\max}-\varphi_i}{\Delta'_{2,j}}\right) & \text{if } \Delta_{2,j} > 0 \\ \min\left(1, \frac{\varphi_{\min}-\varphi_i}{\Delta'_{2,j}}\right) & \text{if } \Delta_{2,j} < 0 \\ 1 & \text{if } \Delta_{2,j} = 0 \end{cases} \tag{4.17}$$

before the limiter for element $i$ is evaluated by taking the minimum adjacent cell sublimiter

$$\Psi_i = \min_{j(i)} \psi_{i,j} \tag{4.18}$$

**Limiter of Venkatakrishnan**

Venkatakrishnan's limiter is somewhat more expensive to compute than the limiter of Barth and Jespersen, but has superior convergence properties. It achieves this in two ways.

First, the non-differentiable sub-limiter functions (4.17) of the limiter of Barth and Jespersen are replaced by the differentiable sub-limiters

$$\psi_{i,j} = \begin{cases} \frac{1}{\Delta_{2,j}}\left[\frac{\left(\Delta^2_{1,\max}+\varepsilon_i^2\right)\Delta_{2,j}+2\Delta^2_{2,j}\Delta_{1,\max}}{\Delta_{1,\max}+2\Delta^2_{2,j}+\Delta_{1,\max}\Delta_{2,j}+\varepsilon_i^2}\right] & \text{if } \Delta_{2,j} > 0 \\ \frac{1}{\Delta_{2,j}}\left[\frac{\left(\Delta^2_{1,\min}+\varepsilon_i^2\right)\Delta_{2,j}+2\Delta^2_{2,j}\Delta_{1,\min}}{\Delta_{1,\min}+2\Delta^2_{2,j}+\Delta_{1,\min}\Delta_{2,j}+\varepsilon_i^2}\right] & \text{if } \Delta_{2,j} < 0 \\ 1 & \text{if } \Delta_{2,j} = 0 \end{cases} \tag{4.19}$$

where

$$\begin{aligned} \Delta_{1,\max} &\coloneqq \varphi_{\max} - \varphi_i \\ \Delta_{1,\min} &\coloneqq \varphi_{\min} - \varphi_i \end{aligned} \tag{4.20}$$

and $\varepsilon_i$ is a damping parameter which is used to adjust the strength of the limiter function. The second method by which Venkatakrishnan's limiter achieves superior convergence properties is by scaling this damping parameter with the local grid geometry according to

$$\varepsilon_i^2 \coloneqq (Kh_i)^3 \tag{4.21}$$

where $K = [0, \infty)$ is a global parameter and $h_i$ is a measure of the size of the scalar reconstruction stencil. For the node-based case $h_i$ is the average distance from node $i$ to

each of the adjacent cell circumcentres is, and for the cell-based case $h_i$ is the square root of the area of cell $i$. The parameter $K$ can be set to zero to obtain the fully-limited case, or increased towards infinity to approach the unlimited case. Setting the parameter too low can result in convergence stalling [Bla01, p. 168], whilst setting the parameter too high can result in increasing overshoot at discontinuities, and ultimately the generation of spurious oscillations and solution break-up that the inclusion of gradient limiting is intended to prevent. In practice, $K = 5$ was found to be an appropriate choice for all the numerical examples presented in this thesis.

With the sub-limiters calculated, the limiter $\Psi_i$ is again calculated from (4.18).

### 4.4.2 Gradient Limiting in Momentum Field Reconstructions

As mentioned at the beginning of this section, the choice of edge centre normal momentum as a primary flow variable (and relegation of edge centre tangential momentum to a secondary flow variable) makes the limiting of the piecewise linear momentum field slightly more difficult – both because the limiter functions cannot be calculated using the formulae given above, and because the limiting cannot be applied directly to the gradient as in (4.12). A work-around to the first problem that is found to work well in practice is to set the node-based momentum limiter equal to a node-based scalar limiter computed for a scalar primary flow variable (in the case of the present solver, the node-based pressure limiter is available immediately or a node-based density limiter could be computed). There are two reasonable methods for solving the latter problem, which are discussed in Section A.3.1 of Appendix A. The method chosen was discussed in [VSW06] and involves computing both the piecewise constant momentum reconstruction $(\bar{\boldsymbol{m}}_e \cdot \boldsymbol{n}_i)^{(1)}$ and the piecewise linear momentum reconstruction $(\bar{\boldsymbol{m}}_e \cdot \boldsymbol{n}_i)^{(2)}$. The previously computed pressure gradient limiter is then used to compute a weighted average of the two reconstructions

$$(\bar{\boldsymbol{m}}_e \cdot \boldsymbol{n}_i) = (1 - \Psi_e)\,(\bar{\boldsymbol{m}}_e \cdot \boldsymbol{n}_i)^{(1)} + \Psi_e\,(\bar{\boldsymbol{m}}_e \cdot \boldsymbol{n}_i)^{(2)} \tag{4.22}$$

where $\Psi_e$ is the limiter at the base node for the reconstruction of $(\bar{\boldsymbol{m}}_e \cdot \boldsymbol{n}_i)^{(2)}$. The weighted average is such that in smooth regions of the flow field the unlimited second-order reconstruction is recovered, whilst near shocks the average is biased towards the first-order reconstruction.

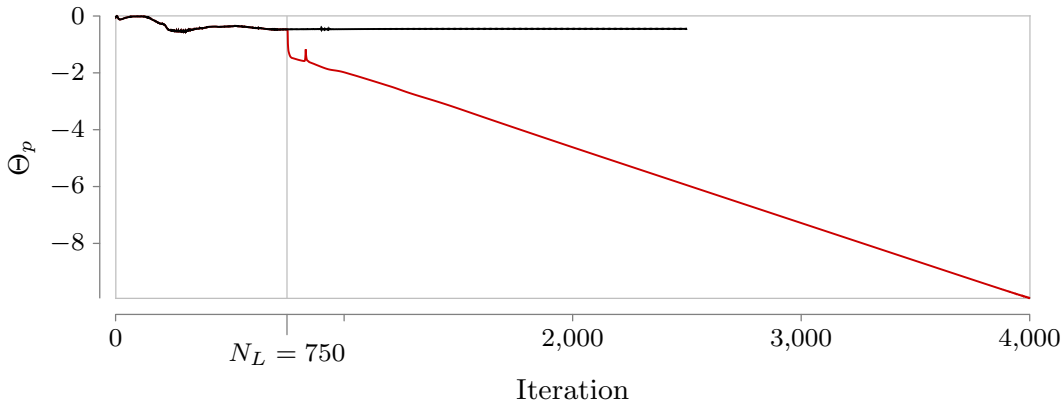



Figure 4.4: Pressure residual history for Venkatakrishnan's limiter with $K = 0$ on the supersonic NACA 0012 case discussed in Section 4.8.1, with $M_r = 1.5$ and zero angle of attack. The original limiter results in the residual history shown in black. Applying the historic modification with $N_L = 750$ results in the residual history shown in red.

### 4.4.3 Convergence Stalling and the Historic Modification

In practice Venkatakrishnan's limiter can cause convergence to stall. Increasing the parameter $K$ does not always alleviate this problem. Often it is found that increasing $K$ a small amount results in little improvement, whilst increasing $K$ a significant amount leaves the reconstructions insufficiently limited.

Several methods have been proposed to address this issue. For example, De Zeeuw [De 93] includes a user-specified parameter $N_L \in \mathbb{N}^+$. After $N_L$ iterations have been completed, the value of the limiters is frozen. This does work, but its success is quite senstive to the choice of $N_L$. If $N_L$ is chosen too small, new extrema may emerge, unlimited, after the freezing has occurred. If $N_L$ is chosen too large, instabilities may have already been allowed to grow too large, likely causing convergence issues. A similar procedure was proposed by Delanaye [Del96], the so-called *historic modification*. In this method the value of the limiter functions is no longer allowed to increase once $N_L$ iterations have elapsed, but they are allowed to decrease. This alleviates the first issue with De Zeeuw's method, and ensures that extrema are always adequately limited, regardless of the choice of $N_L$. Selecting too small a value for $N_L$ is still to be avoided, however, as it could lead to the limiters being applied more strongly than necessary, harming accuracy. In testing, it was found that the historic modification did indeed perform well in the present solver, with $N_L = 750$ appearing to be a good (though not universal) choice. An example pressure residual history with and without the historic modification applied is shown in Figure 4.4.

## 4.5 Boundary Conditions

The design and implementation of the boundary conditions was an area of particular difficulty in the development of the solver. The source of this difficulty is the combination of the compressible Euler equations with a staggered primary flow variable layout, which is rare in the literature. In addition, and in common with other solvers, the robustness and efficiency of the solver was found to be strongly dependent on the design and implementation of the boundary conditions.

Where possible the boundary conditions are imposed implicitly, by substituting them directly into the governing equations, in order to maintain stability. This can cause efficency and robustness problems with the linear solvers and preconditioners – the equations representing the boundary conditions are unscaled, whilst the equations corresponding to the interior points carry the factor $\Omega_i/\Delta t_i$. For this reason, the equations representing the boundary conditions are therefore weighted by the adjacent interior cell $\Omega/\Delta t$. This ensures that the boundary condition equations are scaled appropriately as the grid is refined and the time step varied. This scaling is used for all implicit boundary condition equations, but is omitted below for notational clarity.

It is important to note that order of accuracy requirements on boundary conditions are less strict than those of the interior discretisation. Specifically, a scheme with order of accuracy $p$ can use a boundary discretisation with order of accuracy $p-1$ without the order of accuracy of the scheme being negatively affected [Gus75].

### 4.5.1 Dummy Cells

There are two main methods for imposing boundary conditions. The first involves the modification of the scalar and vector reconstruction procedures and the computation of the fluxes along the boundaries of the flow domain. The other is the use of so-called *dummy cells* (also known as *ghost cells*). This involves the addition of one or more layers of cells to the outside of the flow domain, along the boundary edges. These cells are usually constructed simply by mirroring the interior cells across the boundary. Although this approach may result in overlapping dummy cells, this is not an issue since the conservation principles are not applied directly to the dummy cells.

The main advantage of dummy cells is that the conservation equations can be assembled at each cell without needing to test whether the stencil reaches the boundary of the flow domain. This increases computational efficiency, especially on modern processors with vector instructions [Bla01, p. 268] and out-of-order execution, due to the removal of conditional statements that would necessitate branch prediction.

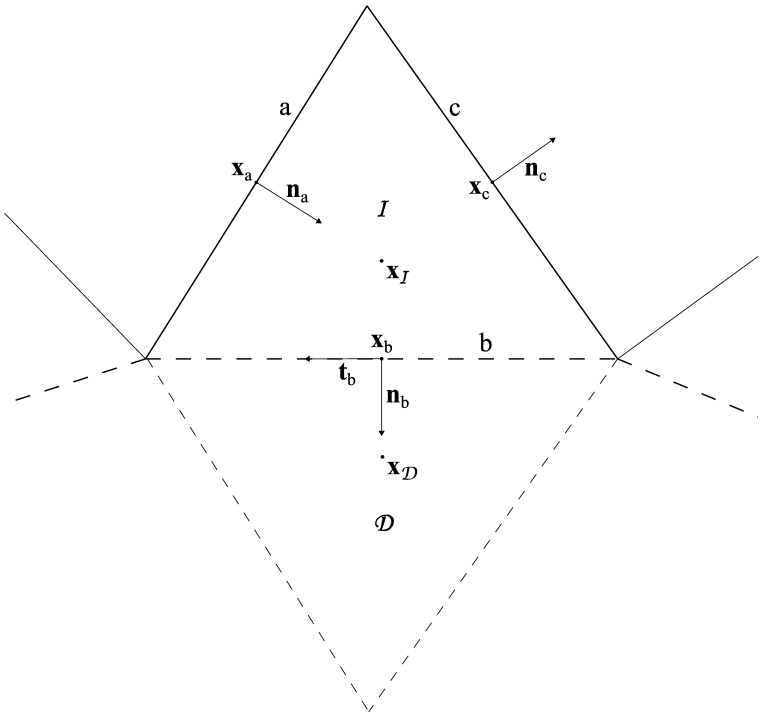



Figure 4.5: The grid layout used for discussion of the boundary conditions. $\mathcal{I}$ represents the interior cell adjacent to the boundary edge $b$, whilst $\mathcal{D}$ represents the adjacent dummy cell. Since the choice of edge unit normals between the two opposite directions is arbitrary, for notational convenience the boundary edge unit normals are defined such that they point outwards from the flow domain.

The boundary conditions are described with reference to Figure 4.5.

### 4.5.2 Farfield

In the theoretical analysis of external aerodynamics, the flow domain surrounding the geometry of interest is assumed to extend to infinity in all directions. The flow conditions imposed at infinity are called the *farfield* conditions. In CFD computations, however, this approach is not feasible because the computational domain must be compact (i.e. closed and bounded) so that it can be covered by a finite triangulation. The farfield must therefore be placed a reasonably large but finite distance from the geometry of interest. This creates a so-called *artificial boundary*. Along artificial boundaries, special boundary conditions must be imposed to simulate the theoretical situation of the flow domain extending to infinity – these are called *farfield* boundary

| | Subsonic | Supersonic |
|---|---|---|
| Inlet | 2 | 3 |
| Outlet | 1 | 0 |

Table 4.1: The number of flow variables that must be imposed at the farfield boundary.

conditions. In addition to the usual design requirements on boundary conditions in computational fluid dynamics, farfield boundary conditions should satisfy two additional constraints.

1. The cutting-off of the flow domain a finite distance from the geometry of interest must have a negligible effect on the obtained solution. As well as being considered during the choice of suitable farfield boundary conditions, this requirement also necessitates careful consideration of the choice of distance at which to place the farfield boundary.

2. Outgoing waves generated in the interior of the flow domain should be absorbed by the farfield boundary, rather than being reflected back into the interior. That is, the farfield boundary conditions should be *non-reflecting*. In the steady solver this is a only a matter of efficiency and stability rather than one of accuracy, since the outgoing waves decay as the steady state is approached. We can therefore expect incoming reflected waves to be absent from a converged solution, leaving the disruption of convergence as the only problem likely to be caused. This issue will take on a much greater importance in Chapter 6, when the development of the unsteady solver is considered.

The requirements placed on the design of the farfield boundary conditions allow some simplifying assumptions to be made. The requirement that the farfield boundary be placed sufficiently far from the geometry of interest means that the solution near the farfield boundary can be assumed to be nearly constant. The spatial gradients of the primary flow variables can be assumed to be practically negligible, especially in the tangential direction. Therefore, the flow at the farfield boundary can be assumed to be modelled sufficiently accurately by the one-dimensional Euler equations. Fortunately the structure and properties of these equations is well understood, and many of the farfield boundary conditions described in the literature exploit this. One result from this analysis is the number of conditions that must be imposed at the farfield boundary [Kre70]. Applied to the present solver, this information is summarised in Table 4.1. The remaining primary flow variables must be extrapolated from the interior solution.

Several sets of farfield boundary conditions were tested with the solver. The same issue discussed in Section 3.7 in the context of the reconstruction of the convective fluxes, that the literature for compressible finite volume methods is almost entirely focussed on collocated primary flow variable layouts, was again encountered. This issue meant that significant modification was required before implementation of the farfield boundary conditions in the solver. One such set of boundary conditions, first presented in [Whi83], was found to perform particularly well with the solver. Its modification and implementation is described in the following sections.

The implementation of the farfield boundary conditions requires all the farfield flow variables to be specified. These values are those the flow variables would take on an infinite distance from the geometry of interest, not the values actually imposed at the farfield boundary. In what follows, these values will be denoted by $\mathring{\rho}_\infty$, $\mathring{\boldsymbol{m}}_\infty$, and $\mathring{p}_\infty$, where $\mathring{\varphi}$ denotes a value imposed by the boundary conditions.

The description of the farfield boundary conditions is split into four sections corresponding to whether the local flow is supersonic or subsonic, and on whether it is inwards into the domain or outwards from the domain. The assignment of the boundary edges (and their attached dummy cells) to each of the four types is determined in the following way

1. If the farfield conditions are supersonic, then all boundary edges are also supersonic. At each boundary edge, $b$, $\mathring{\boldsymbol{m}}_\infty \cdot \boldsymbol{n}_b$ is computed. If this is greater than zero, then the edge is a supersonic outlet. Otherwise, it is a supersonic inlet.

2. If the farfield conditions are subsonic, then a linear interpolation is performed between the interior cell state and the farfield state

$$\begin{aligned} \boldsymbol{m}_{\text{int}} &\coloneqq \frac{1}{2}\left(\boldsymbol{m}_{\mathcal{I}} + \mathring{\boldsymbol{m}}_\infty\right) \\ M_{\text{int}} &\coloneqq \frac{1}{2}\left(M_{\mathcal{I}} + \mathring{M}_\infty\right) \end{aligned} \tag{4.23}$$

   where, as before, $M$ denotes the Mach number. The boundary edge is then subsonic if $M < 1$, supersonic if $M \geq 1$, an inlet edge if $\boldsymbol{m}_{\text{int}} \cdot \boldsymbol{n}_b \leq 0$, and an outlet edge if $\boldsymbol{m}_{\text{int}} \cdot \boldsymbol{n}_b > 0$ (recall that the boundary edge unit normals point outwards from the flow domain).

It is sometimes possible for the boundary flow to change type (locally) during the solution process. This is generally acceptable. It may cause convergence difficulties, however if the farfield flow is nearly tangential to the farfield boundary, in which case

the affected part of the boundary can oscillate rapidly between inlet to outlet. To prevent this phenomenon occuring, the shape of the farfield boundary can be modified to ensure that $\mathring{\boldsymbol{m}}_\infty$ is nowhere nearly tangential to the boundary. Notwithstanding the above, a boundary edge that is a supersonic inlet according solely to the farfield conditions should never become a different type. This would indicate that information is travelling in an unphysical direction, and likely means that the farfield is located too close to the geometry of interest or that the boundary conditions are poorly designed.

**Farfield, Supersonic Inlet**

At a subsonic inlet all eigenvalues of the convective flux Jacobian are negative (with respect to Figure 4.5, i.e. with $\boldsymbol{n}_b$ pointing outwards from the flow domain), meaning all the characteristic waves are travelling inwards into the domain. All primary and secondary flow variables at the centre of the inlet boundary edge and the circumcentre of its attached dummy cell are therefore set equal to the farfield conditions

$$\begin{aligned} \rho_{\mathcal{D}} &= \rho_b = \mathring{\rho}_\infty \\ \boldsymbol{m}_{\mathcal{D}} &= \boldsymbol{m}_b = \mathring{\boldsymbol{m}}_\infty \\ p_{\mathcal{D}} &= p_b = \mathring{p}_\infty \end{aligned} \tag{4.24}$$

The boundary condition on the pressure is imposed directly only during initialisation. During the solution process, it is instead imposed indirectly via the obvious condition on the pressure correction in the dummy cell

$$\delta p_{\mathcal{D}}^{n+1} = 0 \tag{4.25}$$

**Farfield, Supersonic Outlet**

At the supersonic outlet all eigenvalues of the convective flux Jacobian are positive, meaning all the characteristic waves are travelling outwards from the domain. As a result, no Dirichlet conditions can be imposed, and the scalar primary and secondary flow variables are simply extrapolated from the interior of the domain.

$$\begin{aligned} \rho_{\mathcal{D}} &= \rho_b = \rho_{\mathcal{I}} \\ p_{\mathcal{D}} &= p_b = p_{\mathcal{I}} \\ \delta p_{\mathcal{D}}^{n+1} &= \delta p_b^{n+1} = \delta p_{\mathcal{I}}^{n+1} \end{aligned} \tag{4.26}$$

Note that for the first-order extrapolation used here, this condition is identical to imposing the Neumann condition $\nabla\varphi|_b \cdot \boldsymbol{n}_b = 0$ at the boundary edge centre. This is not true when a higher-order extrapolation is employed. The above boundary conditions are simply substituted implicitly into the corresponding conservation equations. The normal momentum prediction at the outlet edge centre, $(\boldsymbol{m} \cdot n)_b^*$, does not require a boundary condition – it is obtained by solving the normal momentum equation using the modified control volume shown in Figure 3.3. The tangential momentum at the outlet edge centre is extrapolated using one of the reconstruction procedures discussed in Section 3.9. The momentum in the dummy cell is extrapolated using linear extrapolation from the boundary edge and adjacent interior cell momenta.

Note that a simple first-order extrapolation has been used from the interior state to the boundary edge centre, and then to the dummy cell circumcentre. As mentioned earlier in this section, a first-order boundary treatment is sufficient for second-order scheme. To obtain a formally third-order-accurate scheme, a more accurate extrapolation would be required. In fact, since the farfield is necessarily located a significant distance from the geometry of interest, retaining the first-order extrapolation presented here would likely remain sufficiently accurate in practice.

**Farfield, Subsonic Inlet**

The subsonic farfield boundary conditions are more complicated than for the supersonic case, for three reasons. Firstly, they require some characteristic variables to be extrapolated from the interior of the flow domain, whilst the remainder are obtained from the farfield state. The result is that the values of the primary flow variables at the centre boundary edge and at the circumcentre of its adjacent dummy cell are no longer the same, and the staggered layout of the present solver becomes relevant. Secondly, most implementations of farfield boundary conditions in the literature use the flow velocity as a primary flow variable, whilst the present solver uses the momentum. Finally, compressible farfield boundary conditions in the literature are almost always designed for use with a collocated solver, whereas the present solver uses a staggered layout of the primary flow variables. The original version of the boundary conditions presented here, written for a collocated solver, can be found in [Bla01, p. 278].

The boundary conditions involve a pressure-based extrapolation from the interior

state

$$\begin{aligned} p_b &= \frac{p_{\mathcal{I}} + \mathring{p}_\infty}{2} + \frac{\mathring{\rho}_\infty \mathring{a}_\infty}{2} \boldsymbol{n}_b \left( \boldsymbol{u}_{\mathcal{I}} - \mathring{\boldsymbol{u}}_\infty \right) \\ \rho_b &= \mathring{\rho}_\infty + \frac{p_b - \mathring{p}_\infty}{a_{\mathcal{I}}^2} \\ \boldsymbol{u}_b &= \mathring{\boldsymbol{u}}_\infty + \boldsymbol{n}_b \frac{p_b - \mathring{p}_\infty}{\mathring{\rho}_\infty \mathring{a}_\infty} \end{aligned} \tag{4.27}$$

where $a$ is the sonic velocity. The actual boundary conditions imposed implicitly on the governing equations are obtained via the linear extrapolation formula $\varphi_{\mathcal{D}} = 2\varphi_b - \varphi_{\mathcal{I}}$ and some basic algebraic manipulation

$$\begin{aligned} \rho_{\mathcal{D}}^{n+1} + \rho_{\mathcal{I}}^{n+1} &= 2\mathring{\rho}_\infty + \frac{1}{\mathring{a}_\infty} \left[ \frac{p_{\mathcal{I}}^n - \mathring{p}_\infty}{\mathring{a}_\infty} + \mathring{\rho}_\infty \boldsymbol{n}_b \cdot \left( \boldsymbol{u}_{\mathcal{I}}^n - \mathring{\boldsymbol{u}}_\infty \right) \right] \\ \left( \boldsymbol{m} \cdot \boldsymbol{n} \right)_b^* - \frac{\rho_b^{n+1}}{2\rho_{\mathcal{I}}^{n+1}} \boldsymbol{m}_{\mathcal{I}}^* \cdot \boldsymbol{n}_b &= \rho_b^{n+1} \mathring{\boldsymbol{u}}_\infty \cdot \boldsymbol{n}_b + \frac{\rho_b^{n+1} \left( p_{\mathcal{I}}^n - \mathring{p}_\infty \right)}{2\mathring{\rho}_\infty \mathring{a}_\infty} - \frac{\rho_b}{2} \mathring{\boldsymbol{u}}_\infty \cdot \boldsymbol{n}_b \\ \delta p_{\mathcal{D}}^{n+1} - \frac{\Delta t_{\mathcal{I}}}{\rho_{\mathcal{I}}^{n+1}} \nabla \delta p_{\mathcal{I}}^{n+1} &= 0 \end{aligned} \tag{4.28}$$

The term $\boldsymbol{m}_{\mathcal{I}}^* \cdot \boldsymbol{n}_b$ is reconstructed implicitly using one of the procedures described in Section 3.9. When required, the tangential momentum at the boundary edge centre, $\left( \boldsymbol{m} \cdot \boldsymbol{t} \right)_b$, is simply set equal to the farfield value

$$\left( \boldsymbol{m} \cdot \boldsymbol{t} \right)_b = \mathring{\boldsymbol{m}}_\infty \cdot \boldsymbol{n}_b \tag{4.29}$$

Note that the primary flow variables are treated in line with the segregated solution procedure:

- When solving the continuity equation for the density at the circumcentre of the dummy cell, density is taken at the new time level $n+1$ whilst momentum and pressure are taken at the old time level $n$.
- When solving the normal momentum equation for the boundary edge centre normal momentum, the density is taken at the new time level $n+1$, the momentum is taken at the prediction time level $*$, and the pressure is taken at the old time level $n$.
- When solving the pressure correction equation for the pressure correction at the circumcentre of the dummy cell, all primary flow variables are taken at the new time level $n+1$.

**Farfield, Subsonic Outlet**

The farfield subsonic outlet boundary conditions are quite similar to the subsonic inlet conditions. However, the signs of the eigenvalues of the convective flux Jacobian are reversed, and so there is only one ingoing characteristic, with the remainder outgoing. Hence, only one primary flow variable is taken from the farfield conditions, with the remainder extrapolated from the interior state. The boundary conditions are [Bla01, p. 279]

$$\begin{aligned} p_b &= \mathring{p}_\infty \\ \rho_b &= \rho_\mathcal{I} + \frac{\mathring{p}_\infty - p_\mathcal{I}^n}{\mathring{a}_\infty^2} \\ \boldsymbol{u}_b &= \boldsymbol{u}_\mathcal{I} + \boldsymbol{n}_b \frac{p_\mathcal{I}^n - \mathring{p}_\infty}{\mathring{\rho}_\infty \mathring{a}_\infty} \end{aligned} \tag{4.30}$$

As for the inlet case, the actual boundary conditions imposed on the governing equations are found by linear extrapolation

$$\begin{aligned} \rho_\mathcal{D}^{n+1} - \rho_\mathcal{I}^{n+1} &= \frac{2\left(\mathring{p}_\infty - p_\mathcal{I}^n\right)}{\mathring{a}_\infty^2} \\ p_\mathcal{D} &= 2p_b - p_\mathcal{I}^n \\ \delta p_\mathcal{D}^{n+1} + \delta p_\mathcal{I}^{n+1} &= 0 \end{aligned} \tag{4.31}$$

The normal momentum prediction at the outlet edge centre, $(\boldsymbol{m} \cdot \boldsymbol{n})_b^*$, is obtained directly by solving the normal momentum equation. The tangential momentum at the outlet edge centre is extrapolated from the interior state. The momentum in the dummy cell is extrapolated using linear extrapolation in the usual manner.

**Farfield Vortex Correction**

According to the basic theory of subsonic Euler flows, lift is generated by the production of circulation about the lifting surface [And10]. As there is no viscous diffusion in an Euler flow, the production of circulation disturbs the farfield conditions arbitrarily far from the geometry of interest. However, the deviation from the non-lifting conditions is inversely proportional to the distance from the lifting surface. In order to use non-lifting farfield boundary conditions without noticeably affecting the flow region of interest, then, the farfield must be placed sufficiently far from the geometry of interest that the deviation is negligible. In [Bla01, p. 281] this distance is found by numerical experiment to exceed a hundred chords, even for flow cases involving only modest amounts of lift. This indicates that very large grids are necessary for accurate simulation of lifting flows

in external aerodynamics.

A better solution to the problem of lift generation disturbing the farfield conditions was presented in [Usa83]. In this method, the farfield conditions used for imposing the boundary conditions on each farfield edge are locally corrected by assuming that a compressible vortex is placed at the centre of lift of each lifting surface contained in the flow domain. As the true centres of lift can change during the solution process, it is the usual practice to instead centre the vortex on the quarter-chord point of the lifting surface, $\boldsymbol{x}_{1/4}$. The output from the procedure is a simple correction to be applied to the local values of each of the farfield primary flow variables, and so multiple lifting surfaces can easily be handled by simply superimposing vortices centred on each.

The strength of each vortex is determined by numerically integrating the pressure over the lifting surface, calculating the lift coefficient $c_L$, and applying the Kutta–Joukowski theorem

$$\Gamma = \frac{1}{2} \left\| \mathring{\boldsymbol{u}}_\infty \right\|_2 \ell_c c_L \tag{4.32}$$

where $\Gamma$ is the dimensionless circulation and $\ell_c$ is the chord length. The corrections are updated at the beginning of each time step. The actual numerical integration procedure used to calculate the lift coefficient is not critical to the success of the procedure, a simple first-order numerical integration is found to be sufficient.

The corrections to the farfield flow conditions at location $\boldsymbol{x}_i$ on the farfield boundary due to the lifting surface $\beta$ are most easily calculated in polar coordinates

$$\begin{aligned} r_{\beta,i} &= \sqrt{\left\| \boldsymbol{x}_i - \boldsymbol{x}_{\beta,1/4} \right\|_2} \\ \theta &= \tan^{-1} \left( \frac{\left( \boldsymbol{x}_i - \boldsymbol{x}_{\beta,1/4} \right) \cdot \boldsymbol{j}}{\left( \boldsymbol{x}_i - \boldsymbol{x}_{\beta,1/4} \right) \cdot \boldsymbol{i}} \right) \end{aligned} \tag{4.33}$$

where $\boldsymbol{x}_{\beta,1/4}$ is the location of the quarter-chord point of lifting surface $\beta$. The corrections are then

$$\begin{aligned} \delta \boldsymbol{u}_{\beta,i} &= \frac{\Gamma \sqrt{1 - \mathring{M}_\infty^2}}{2\pi r_{\beta,i} \left[ 1 - M_\infty^2 \sin^2 (\theta - \alpha) \right]} \left\{ \begin{array}{c} \sin\theta \\ -\cos\theta \end{array} \right\} \\ \delta p_{\beta,i} &= \left( \left( \mathring{p}_\infty^+ \right)^{\frac{\gamma-1}{\gamma}} + \frac{\gamma - 1}{\gamma} \cdot \frac{\mathring{\rho}_\infty \left\| \delta \boldsymbol{u}_{\beta,i} \right\|_2^2}{\left( \mathring{p}_\infty^+ \right)^{1/\gamma}} \right)^{\frac{\gamma}{\gamma-1}} - \mathring{p}_\infty^+ \\ \delta \rho_{\beta,i} &= \mathring{\rho}_\infty \left[ \left( \frac{\hat{p}_{\beta,i}}{\mathring{p}_\infty^+} \right)^{1/\gamma} - 1 \right] \end{aligned} \tag{4.34}$$

where $p^+ \coloneqq p + \frac{1}{\gamma M_r^2}$ is the dimensionless absolute pressure. The farfield flow state $(\mathring{\rho}_\infty, \mathring{\boldsymbol{u}}_\infty, \mathring{p}_\infty)$ used in (4.27), (4.28), (4.30), and (4.31) is replaced by the corrected state at the centre of the boundary edge

$$\begin{aligned} \rho^*_{\infty,b} &= \mathring{\rho}_\infty + \sum_\beta \delta\rho_{\beta,i} \\ \boldsymbol{u}^*_{\infty,b} &= \mathring{\boldsymbol{u}}_\infty + \sum_\beta \delta\boldsymbol{u}_{\beta,i} \\ p^*_{\infty,b} &= \mathring{p}_\infty + \sum_\beta \delta p_{\beta,i} \end{aligned} \tag{4.35}$$

where the summations range over the individual lifting surfaces.

It is shown in [Bla01, p. 281] that applying the farfield vortex correction allows the farfield distance to be reduced to around 25 chords without significant loss in accuracy. In fact, the corrections presented above include only the lowest-order corrections to the farfield flow conditions due to the presence of lifting surfaces. It is shown in [GDT85] that including the next-lowest-order corrections allows the farfield distance to be reduced even further, perhaps even to only five chords.

For supersonic flow, only a small portion of the farfield boundary is within the physical region of influence of the lifting surface. Correcting the farfield flow conditions around the whole of the farfield boundary due to the flow at the geometry of interest would fail to respect the physics, and can therefore be expected to yield unphysical results. For this reason, the farfield vortex correction should only be applied when the farfield flow is subsonic.

### 4.5.3 Pressure Boundary Conditions

When modelling interior flows, subsonic farfield boundary conditions can no longer be used for two reasons.

1. The geometry of interest may be close to the inlet or outlet boundaries, and the assumption of one-dimensional flow at the boundary is no longer valid.

2. The only path from the inlet to the outlet is through the geometry of interest, so the flow conditions at outlet are necessarily dependent on entropy production in the geometry of interest regardless of the distance to the outlet boundary. Hence the downstream flow state is both different to the upstream flow state and dependent on the solution. There is therefore no farfield state to impose at the

outlet. In subsonic inviscid flow this is also true of the inlet boundary, since any upstream-travelling waves should continue travelling upstream with no diffusion.

For supersonic flow, the local flow state at the depends only on the upstream flow state. Hence, supersonic inlets have all variables imposed and supersonic outlets have all variables extrapolated from the interior state. The boundary conditions used in interior supersonic flows are therefore identical to those used for farfield supersonic flows.

**Subsonic Pressure Inlet**

The procedure presented here is based on a method orginally developed for use with turbomachinery simulations [HC], but it was found to perform well with the present solver for the interior flow cases considered. The total enthalpy $\mathring{h}_{o,b}$, total pressure $\mathring{p}_{o,b}$, and inlet flow angle $\mathring{\theta}_b$ are specified at the inlet edge centres. The total enthalpy and total pressure are non-dimensionalised in the same manner as their static counterparts

$$\begin{aligned} h_{o,b} &= \frac{\tilde{h}_{o,b}}{h_r} = h_b + \frac{\gamma - 1}{2} M_r^2 \left\| \boldsymbol{u} \right\|_2^2 \\ p_{o,b} &= \frac{\tilde{p}_{o,b} - p_r}{\rho_r u_r^2} \end{aligned} \tag{4.36}$$

where $h_r$ is given by (3.11).

As for the farfield case, one characteristic variable must be interpolated from the interior state. For this purpose, the outgoing Riemann invariant is selected

$$\mathcal{R}^- = \boldsymbol{u}_\mathcal{I} \cdot \boldsymbol{n}_b - \frac{2a_\mathcal{I}}{\gamma - 1} \tag{4.37}$$

The sonic velocity at the boundary can then be calculated from

$$a_b = \frac{-\mathcal{R}\left(\gamma - 1\right)}{\left(\gamma - 1\right)\cos^2\theta_\mathcal{I} + 2} \left\{ 1 + \cos\theta_\mathcal{I} \sqrt{\frac{\left[\left(\gamma - 1\right)\cos^2\theta_\mathcal{I} + 2\right] a_o^2}{\left(\gamma - 1\right)\left(\mathcal{R}^-\right)^2} - \frac{\gamma - 1}{2}} \right\} \tag{4.38}$$

where $a_o$ is the interior state stagnation sonic velocity

$$a_o^2 = a_\mathcal{I}^2 + \frac{\gamma - 1}{2} \left\| \boldsymbol{u}_\mathcal{I} \right\|_2^2 \tag{4.39}$$

and $\theta_{\mathcal{I}}$ is the interior state flow angle

$$\cos\theta_{\mathcal{I}} = -\frac{\boldsymbol{u}_{\mathcal{I}} \cdot \boldsymbol{n}_b}{\|\boldsymbol{u}_{\mathcal{I}}\|_2} \tag{4.40}$$

The remaining flow variables are determined using the non-dimensionalised form of the standard flow equations [Bla01, p. 284]

$$\begin{aligned}
h_b &= \mathring{h}_{o,b}\left(\frac{a_b^2}{a_o^2}\right) \\
p_b &= \left(\mathring{p}_{o,b} + \frac{1}{\gamma M_r^2}\right)\left(\frac{h_b}{\mathring{h}_{o,b}}\right)^{\frac{\gamma}{\gamma-1}} - \frac{1}{\gamma M_r^2} \\
\rho_b &= \frac{1}{h_b}\left(1 + \gamma M_r^2 p_b\right) \\
\boldsymbol{u}_b &= \frac{1}{M_r}\sqrt{\frac{2\left(\mathring{h}_{o,b} - h_b\right)}{\gamma - 1}}\left\{\begin{array}{c} \cos\mathring{\theta}_b \\ \sin\mathring{\theta}_b \end{array}\right\}
\end{aligned} \tag{4.41}$$

These boundary conditions are imposed explicitly, at the beginning of each iteration. The flow variables in the dummy cell are obtained by linear extrapolation from the boundary edge and interior cell flow variables. When solving the pressure correction equation, the dummy cell pressure correction is set to zero.

### Subsonic Pressure Outlet

At the subsonic pressure outlet, only the outlet pressure $\mathring{p}_e$ is specified. It is generally acceptable to simply use the same conditions as for the farfield subsonic outlet (4.30), but this can perform quite poorly if the pressure outlet is quite close to the geometry of interest. Instead, a split-pressure extrapolation based on the AUSM+ convective flux splitting [Lio96] is found to lead to significantly improved performance.

The exit sonic velocity and splitting advection Mach number are calculated from

$$\begin{aligned}
a_b &= \frac{\hat{a}_b^2}{\max\left(\hat{a}_b, |\boldsymbol{u}_{\mathcal{I}} \cdot \boldsymbol{n}_b|\right)} \\
M_{\mathcal{S}} &= \frac{1}{a_b}\boldsymbol{u}_{\mathcal{I}} \cdot \boldsymbol{n}_b
\end{aligned} \tag{4.42}$$

where

$$\hat{a}_b^2 = \frac{2a_{o,\mathcal{I}}^2}{\gamma + 1} \tag{4.43}$$

The split pressures are calculated from

$$\begin{aligned}\mathcal{P}^{+}\left(M_{\mathcal{S}}\right) &= \frac{1}{4}\left(M_{\mathcal{S}}+1\right)^{2}\left(2-M_{\mathcal{S}}\right)+\frac{3}{16}M_{\mathcal{S}}\left(M_{\mathcal{S}}^{2}-1\right)^{2}\\ \mathcal{P}^{-}\left(M_{\mathcal{S}}\right) &= \frac{1}{4}\left(M_{\mathcal{S}}-1\right)^{2}\left(2+M_{\mathcal{S}}\right)-\frac{3}{16}M_{\mathcal{S}}\left(M_{\mathcal{S}}^{2}-1\right)^{2}\end{aligned} \tag{4.44}$$

Finally, the boundary edge centre pressure is

$$p_{\flat} = \mathcal{P}^{+}\left(M_{\mathcal{S}}\right)p_{\mathcal{I}}+\mathcal{P}^{-}\left(M_{\mathcal{S}}\right)\mathring{p}_{e} \tag{4.45}$$

The normal momentum equation is solved for the normal momentum at the exit edge centre, whilst the the tangential momentum is extrapolated from the interior state. Finally, the density is extrapolated in a similar way as for the subsonic farfield outlet, except $p_{\infty}$ is replaced by the exit pressure $\mathring{p}_{e}$ and the reference sonic velocity is instead taken from the interior state. The dummy cell circumcentre flow variables are obtained by extrapolation from the outlet edge centre and the adjacent interior cell circumcentre. The actual conditions imposed on the governing equations are then

$$\begin{aligned}\rho_{\mathcal{D}}^{n+1}-\rho_{\mathcal{I}}^{n+1} &= \frac{\left[2\mathcal{P}^{+}\left(M_{\mathcal{S}}\right)-1\right]p_{\mathcal{I}}^{n}+2\mathcal{P}^{-}\left(M_{\mathcal{S}}\right)\mathring{p}_{e}}{\left(a_{\mathcal{I}}^{n}\right)^{2}}\\ \delta p_{\mathcal{D}}^{n+1}-\left[2\mathcal{P}^{+}\left(M_{\mathcal{S}}\right)-1\right]\delta p_{\mathcal{I}} &= 0\end{aligned} \tag{4.46}$$

where the split pressures $\mathcal{P}^{\pm}\left(M_{\mathcal{S}}\right)$ are calculated explicitly.

Since the subsonic outlet boundary might be located quite close to the geometry of interest, it is possible for backflow to occur. Even if this is not present in the solution, it may still happen during convergence to steady state. It is therefore important for the possibility of backflow to be handled robustly, or solution break-down is likely to result. This can be done by specifying a backflow total enthalpy. Unfortunately there is no universal procedure for choosing the backflow total enthalpy. A sensible value must be guessed, then adjusted if necessary. The backflow flow angle is assumed to be normal to the boundary, and the backflow total pressure is taken to be equal to the specified exit static pressure. Then, the subsonic pressure inflow conditions described in the previous subsection are imposed. Backflow at the subsonic pressure outlet was not encountered when testing the inviscid solver, but did occur when testing the viscous solver described in the next chapter. The backflow procedure described here was found to perform well.

#### Outlet with Specified Exit Mach Number

Occasionally, interior fluid dynamics problems have an outlet for which the exit Mach number is specified rather than the exit pressure. This is the case for the numerical example presented in Section 4.8.4, an engine inlet with the Mach number specifed at the end of the inlet. The procedure employed to impose this condition is quite similar to the procedure used at subsonic pressure outlets, except the prescribed exit pressure is varied to drive the specified target exit Mach number towards the specified value.

At the start of each iteration, the exit total pressure corresponding to each outlet boundary edge is calculated by assuming no loss of total pressure from the adjacent interior cell to the exit and applying the isentropic flow relations [And10, p. 529]. This allows the exit static pressure to be written in terms of the adjacent interior cell static pressure

$$p_e = \left(p_{\mathcal{I}} + \frac{1}{\gamma M_r^2}\right)\left[\frac{1+\frac{\gamma-1}{2}M_{\mathcal{I}}^2}{1+\frac{\gamma-1}{2}M_e^2}\right]^{\frac{\gamma}{\gamma-1}} - \frac{1}{\gamma M_r^2} \tag{4.47}$$

With the exit pressure calculated, the boundary conditions are identical to those imposed at a subsonic pressure outlet, except the exit pressure is updated using the specified exit Mach number at the beginning of each iteration.

### 4.5.4 Symmetry Plane

The symmetry plane boundary is defined by two conditions: the normal momentum at the boundary is zero (in subsonic flow this is equivalent to the requirement that the boundary is a streamline), and the normal gradients of the primary flow variables at the boundary are zero. The first of these conditions is imposed directly, by setting the normal momentum at the boundary edge to zero. Since the dummy cells result from mirroring the adjacent interior cell across the boundary edge, the second condition can easily be imposed by setting the dummy cell circumcentre flow variables equal to the values in the adjacent interior cell

$$\begin{aligned} \rho_{\mathcal{D}}^{n+1} - \rho_{\mathcal{I}}^{n+1} &= 0 \\ (\boldsymbol{m}\cdot\boldsymbol{n})_b^* &= 0 \\ \delta p_{\mathcal{D}}^{n+1} - \delta p_{\mathcal{I}}^{n+1} &= 0 \end{aligned} \tag{4.48}$$

The tangential momentum at the boundary is extrapolated from the interior state. The component of the dummy cell momentum tangential to the boundary is obtained by linear extrapolation, whilst the component of the dummy cell momentum normal

to the wall is found by 'reflecting' from the interior state

$$\begin{aligned} \boldsymbol{m}_{\mathcal{D}} \cdot \boldsymbol{t}_b &= 2\left(\boldsymbol{m} \cdot \boldsymbol{t}\right)_b - \boldsymbol{m}_{\mathcal{I}} \cdot \boldsymbol{t}_b \\ \boldsymbol{m}_{\mathcal{D}} \cdot \boldsymbol{n}_b &= -\boldsymbol{m}_{\mathcal{I}} \cdot \boldsymbol{n}_b \end{aligned} \tag{4.49}$$

Note that whilst this view of the dummy cell normal momentum is useful conceptually, it is completely equivalent to the usual linear extrapolation.

### 4.5.5 Slip Wall

In inviscid flows there are no viscous stresses between the wall and the fluid, and so the only condition on the flow velocity at the wall boundary is that its component normal to the boundary is zero. This also obviously implies that the normal momentum normal to the wall is zero.

$$\left(\boldsymbol{m} \cdot \boldsymbol{n}\right)_b = \left(\boldsymbol{u} \cdot \boldsymbol{n}\right)_b = 0 \tag{4.50}$$

The remainder of the flow variables at the boundary and in the adjacent dummy cells are obtained by extrapolation from the interior state. As discussed earlier, a second-order-accurate scheme requires only a first-order-accurate boundary treatment. Therefore, a suitable set of scalar slip wall boundary conditions is

$$\begin{aligned} \rho_{\mathcal{D}} &= \rho_b = \rho_{\mathcal{I}} \\ p_{\mathcal{D}} &= p_b = p_{\mathcal{I}} \end{aligned} \tag{4.51}$$

The set of conditions actually imposed on the governing equations is therefore

$$\begin{aligned} \rho_{\mathcal{D}}^{n+1} - \rho_{\mathcal{I}}^{n+1} &= 0 \\ \left(\boldsymbol{m} \cdot \boldsymbol{n}\right)_b^* &= 0 \\ \delta p_{\mathcal{D}}^{n+1} - \delta p_{\mathcal{I}}^{n+1} &= 0 \end{aligned} \tag{4.52}$$

Note that this is identical to the boundary conditions imposed at symmetry planes, (4.48). This is no longer true if a higher-order extrapolation is used.

The tangential momentum at the wall is extrapolated from the interior state as usual, whilst the dummy cell momentum is obtained from (4.49).

In [Lio96] it is noted that this set of boundary conditions can generate short-wave oscillations, impeding the convergence of the solver. It is suggested that the pressure

at the wall instead be set using the characteristic equation

$$p_b = p_{\mathcal{I}} + a_{\mathcal{I}} \boldsymbol{m}_{\mathcal{I}} \cdot \boldsymbol{n}_b \tag{4.53}$$

The dummy cell pressure is then obtained using linear extrapolation

$$p_{\mathcal{D}} = 2p_b - p_{\mathcal{I}} = p_{\mathcal{I}} + 2a_{\mathcal{I}} \boldsymbol{m}_{\mathcal{I}} \cdot \boldsymbol{n}_b \tag{4.54}$$

The pressure correction equation for the dummy cell circumcentre is then

$$\delta p_{\mathcal{D}}^{n+1} - \delta p_{\mathcal{I}}^{n+1} - 2a_{\mathcal{I}}^{n} \delta \boldsymbol{m}_{\mathcal{I}}^{n+1} \cdot \boldsymbol{n}_b = 0 \tag{4.55}$$

Although it is stated in [Lio96] that this pressure boundary condition is intended to suppress short-wave oscillations at the wall in combination with the AUSM+ convective flux reconstruction scheme presented in that reference, it is found that it also performs quite well with the H-CUSP scheme used in the present solver.

## 4.6 Stability and Efficiency

### 4.6.1 Selection of the Optimum Time Step

For the steady inviscid solver, the issue of time accuracy can be neglected, and the selection of the time step is focused solely on efficiency and stability considerations. Several methods for calculating the time step were tested. It was found that a procedure described in [Bla01, page 188-189] (and originally introduced in [MJ90]) resulted in the most robust and efficient scheme. For a grid point $i$ with control volume area $\Omega_i$:

1. Calculate the lengths of the projections of the control volume onto the x and y axes

$$\begin{aligned} \Delta S_{x_i} &= \max_{v(i)} (\boldsymbol{x}_v \cdot \boldsymbol{i}) - \min_{v(cv)} (\boldsymbol{x}_v \cdot \boldsymbol{i}) \\ \Delta S_{y_i} &= \max_{v(i)} (\boldsymbol{x}_v \cdot \boldsymbol{j}) - \min_{v(cv)} (\boldsymbol{x}_v \cdot \boldsymbol{j}) \end{aligned} \tag{4.56}$$

   where $v(i)$ denotes the set of vertices of the control volume, and $\boldsymbol{x}_v$ denotes the position of node $v$.

2. Compute the approximate spectral radii of the convective flux Jacobian

$$\begin{aligned} \hat{\Lambda}_{c_i}^{x} &= (|\boldsymbol{u}.\boldsymbol{i}|) + a_i) \, \Delta S_{x_i} \\ \hat{\Lambda}_{c_i}^{y} &= (|\boldsymbol{u}.\boldsymbol{j}|) + a_i) \, \Delta S_{y_i} \end{aligned} \tag{4.57}$$

3. Compute the maximum local time step

$$\Delta t_i = \sigma \frac{\Omega_i}{\hat{\Lambda}^x_{c_i} + \hat{\Lambda}^y_{c_i}} \tag{4.58}$$

where $\sigma$ is the Courant number

$$\sigma \coloneqq \frac{u_r \Delta t}{\Delta x} \tag{4.59}$$

The maximum time step is computed for each grid cell, before the smallest maximum time step across all grid cells is selected as the global time step. An additional advantage of the above method of time step selection is that it can be easily extended to viscous flows and unsteady flows, as will prove useful in the following chapters.

### 4.6.2 Local Time Stepping

The above method of time step selection has a severe disadvantage, in that the time step across the whole grid is set according to the single element requiring the smallest time step. Since only steady-state solutions are being considered here, time accuracy is unimportant and there is no requirement for the solution to evolve at the same rate across the whole domain. Convergence can therefore be dramatically accelerated by the use of local time steps instead of a single global time step. That is, the maximum local time step calculated for a cell using (4.58) is used when solving the corresponding conservation equations. Local convergence is then limited only by the local flow conditions and grid geometry, allowing significantly faster convergence in most regions of the grid. This techique is especially useful for the present solver, which is designed to compute flows that contain regions of drastically different Mach numbers. With global time stepping, the convergence to steady-state in low Mach number regions would be several orders of magnitude slower than necessary.

The inclusion of local time stepping in the scheme does present a difficulty, due to the staggered arrangement of the primary flow variables. Not only are local time steps required for cells, but local time steps must also be computed for edges, for use both in the solution of the normal momentum equation and in the application of the normal momentum correction. The same calculation procedure above could in principle be used to calculate these edge-local time steps, but this will lead to a kind of 'decoupling' between the cell-local time steps and the edge-local time steps due to the difference in control volume definition between the cells and the edges. This decoupling would cause the solver to favour convergence in the normal momenta rather than the scalar primary

flow variables, or vice-versa. Indeed, testing of this strategy found that it leads to a scheme for which selection of an appropriate Courant number was strongly case-specific. The solution employed was simply to define the edge-local time steps to be the average of the local time steps of the adjacent cells in the case of interior edges, and equal to the local time step of the adjacent interior cell in the case of boundary edges. This lead to a scheme with an apparently optimum Courant number that was not strongly dependent on the grid geometry. It is possible, however, that this procedure causes edge-local time steps to be somewhat too large compared to the adjacent cell-local time steps. Partly to address this possibility, the use of *under-relaxation* was considered – this is the topic of the next subsection.

### 4.6.3 Under-Relaxation

Since the scheme is segregated, it is possible that a change in one of the primary flow variables during a given time step may "overpower" the changes in the other two primary flow variables during that time step, and thereby have a destabilising influence on the overall scheme. This would harm the convergence of the scheme, and could possibly cause solution break-up. In order to prevent this, underrelaxation is introduced. Three separate underrelaxation parameters are defined, one each for density, normal momentum and pressure correction. For a general scalar $\varphi$ and corresponding underrelaxation parameter $\alpha_\varphi$, the underrelaxed value of $\varphi$ at the new time level is set according to

$$\varphi^{n+1} = \varphi^n + \alpha_\varphi \left( \varphi^{\text{calc}} - \varphi^n \right) \tag{4.60}$$

where $\varphi^{\text{calc}}$ is the originally computed value of the scalar, and $0 < \alpha_\varphi \leq 1$. Although the underrelaxation can be applied explicitly (i.e. at the end of each time step), it was found that substituting (4.60) directly into the continuity equation and normal momentum equation further improved efficiency. For example, the underrelaxed continuity equation is

$$\frac{\Omega_i \rho_i^{n+1}}{\Delta t \alpha_\rho} + \sum_{e(i)} \bar{l_e} (\boldsymbol{u} \cdot \boldsymbol{n})_e^n \rho_e^{n+1} = \frac{\Omega_i \rho_i^n}{\Delta t \alpha_\rho} \tag{4.61}$$

It can be seen that the underrelaxation effectively scales the time step. The additional efficiency gain from applying the underrelaxation implicitly results from the increased diagonal dominance of the system matrix associated with the effective reduction in the time step. The increased diagonal dominance of the system significantly improves the efficiency of the preconditioner and iterative solver used to solve the linear systems, and these comprise the majority of the computational work required by the solver.

For the pressure correction, it was found that the explicit underrelaxation method lead to faster convergence of the solution than the implicit underrelaxation method, although the precise reason for this is uncertain. However, it is likely due, at least in part, to the fact that the pressure correction (which includes the underrelaxed time step as a factor) is present in all terms, and hence applying the pressure correction underrelaxation implicitly affects not just the diagonal dominance of the system, but also the balance between the various terms in the pressure correction equation. For example, increasing the the implicit underrelaxation factor may change the system from being dominated by the convection term to being dominated by the Laplacian term. An explicit underrelaxation procedure is therefore used for the pressure corrections – after solving the pressure correction equation, the computed pressure correction $\delta p^{comp}$ is replaced by

$$\delta p^{n+1} = \alpha_{\delta p} \delta p^{comp} \tag{4.62}$$

## 4.7 Grid Refinement Study

The penultimate step in the development of the inviscid solver is a grid refinement study, which will allow the asymptotic accuracy of the solver to be approximated. Ultimately, this will confirm the second-order spatial accuracy of the inviscid solver.

### 4.7.1 Procedure

The grid refinement study first requires the definition of an appropriate measure $h$ of grid coarseness. For uniform grids, there is an obvious choice – $h$ is simply set equal to the meshwidth $\Delta r$. On non-uniform grids it is possible to set $h$ to the average meshwidth, but the result is that the meshwidth in the farfield region has an undesirable influence on $h$. In fact, moving the farfield further away from the geometry of interest whilst retaining the same grid growth rate would result in a larger value for $h$, demonstrating the undue influence on $h$ of the meshwidth in regions of the grid far away from the geometry of interest. A better choice for $h$ is the average meshwidth along the wall defining the geometry of interest

$$h \coloneqq \frac{1}{N} \sum_{b \in \Gamma'_w} \ell_b \tag{4.63}$$

where $\Gamma'_w$ is the discretised form of the flow domain boundary of interest (i.e. the set of edges forming the discretisation of the boundary). For the *NACA 0012 aerofoil* case

of Section 4.8.1, this is the aerofoil surface, whilst for the *channel with bump* case of Section 4.8.3, this is the bottom wall containing the bump.

The grid refinement study also requires the definition of a one-parameter family of grids parametrised by the grid coarseness $h$. One way of achieving this is to define the target meshwidth $\Delta r(x, y)$ at the geometry of interest to be a function of $h$, whilst keeping the geometry, grid growth rate, farfield distance and so forth the same regardless of $h$. For a given (deterministic) meshing algorithm, once this procedure is fixed a grid can be identified uniquely by the value of $h$ used to produce it.

In what follows, the exact solution of the Euler equations with the prescribed boundary conditions will be written $\hat{\varphi}(x, y)$ for an arbitary flow variable $\varphi$, whilst the obtained approximate solution to the discretised equations on grid $i$ with coarseness $h_i$ will be denoted $\varphi_{h_i}(x, y)$. The ordered set of approximate solutions on grids with reducing coarseness then forms a sequence $\{\varphi_{h_i}(x, y)\}_{i \in \mathbb{N}}$. Note that, formally, $\varphi_{h_i}(x, y)$ is only defined at discrete points. However, it can be extended to the whole of the flow domain using the interpolation methods discussed in Section 3.8 and Section 3.9.

It is important to note at this point that finite volume methods do not involve solving for pointwise values of $\varphi$, but instead the average of $\varphi$ over each grid cell. Hence, instead of comparing $\varphi_{h_i}(x, y)$ to $\hat{\varphi}(x, y)$, $\varphi_{h_i}(x, y)$ should properly be compared to

$$\hat{\varphi}^*(x_j, y_j) = \frac{1}{\Omega_j} \int_{\Omega_j} \hat{\varphi}(x, y)\, d\Omega_j \tag{4.64}$$

where $(x_j, y_j)$ is the coordinates of the centroid of the grid cell $\Omega_j$ containing the point of interest. The error in the approximate solution at the centroid of that cell can then be defined as

$$\varepsilon_{h_i}{}^{\varphi}(x_j, y_j) \coloneqq \frac{1}{\Omega_j} \int_{\Omega_j} \hat{\varphi}(x, y)\, d\Omega_j - \varphi_{h_i}(x_j, y_j) \tag{4.65}$$

However, as discussed in Section 3.6, approximating the integral in (4.65) by the value of $\hat{\varphi}(x, y)$ at the centroid of the cell is sufficiently accurate for a second-order scheme. Similarly, the same approximation yields a second-order-accurate approximation to the error, and so is sufficiently accurate for performing a grid refinement study on schemes of first- and second-order [Lev02, p. 140]. A significantly simpler and completely sufficient definition for the error in the approximate solution is therefore

$$\varepsilon_{h_i}{}^{\varphi}(x, y) \coloneqq \hat{\varphi}(x, y) - \varphi_{h_i}(x, y) \tag{4.66}$$

The grid refinement study begins with the selection of a suitable geometry and

appropriate boundary conditions. Next, the solution of the governing equations is obtained on progressively finer grids and compared to the exact solution to compute the error fields across each grid. An appropriate norm is then used to evaluate the error in the solution obtained on each grid. Finally, the reduction in the error as the $h$ is reduced is assessed. This produces two critical pieces of information about the solver.

1. The solver is *convergent*. A numerical method is convergent if produces a sequence of approximate solutions that approach (converge to) the true solution of the exact equations as the grid is refined. That is, a convergent method satisfies

$$\lim_{i\to\infty} \left\{ {\varepsilon_{h_i}}^{\varphi} (x,y) \right\} \equiv \lim_{h\to 0} \left\{ \hat{\varphi}(x,y) - \varphi_{h_i}(x,y) \right\} = 0 \tag{4.67}$$

   for all $(x, y)$.

2. The *order of convergence* of the solver. That is, the rate at which the solution of the discretised equations approaches the solution of the exact equations as the grid is refined. For a solver with order of convergence $p$, the error in the approximate solution is proportional to $h^p$ for sufficiently small $h$.

There is a problem with the above procedure that renders it impossible to perform in practice: for all but the simplest flow cases, the exact solution is not available. Instead, the exact solution must be substituted by the best available approximation to the exact solution; for a solver of practical use this is simply the approximate solution obtained on the finest grid. A numerical method is then required to produce a sequence of discrete solutions that converge as the grid is refined. That is, there exists a $\varphi_{h_\infty}(x,y)$ such that

$$\lim_{i\to\infty} \left\{ \varphi_{h_i}(x,y) \right\} = \varphi_{h_\infty}(x,y) \tag{4.68}$$

In some sense $\varphi_{h_\infty}(x,y)$ can be thought of as the discrete solution on an infinitely fine grid, though this is obviously not rigourously meaningful. For this to be true, it must also be true that for sufficiently large $N \in \mathbb{N}^+$

$$\lim_{i\to N} \left| {\hat{\varepsilon}_{h_i}}^{\varphi} (x,y) \right| \approx 0 \tag{4.69}$$

where

$${\hat{\varepsilon}_{h_i}}^{\varphi} \coloneqq \varphi_{h_N}(x,y) - \varphi_{h_i}(x,y) \tag{4.70}$$

is the approximation to the error on grid $i$ relative to the solution obtained on grid $N$.

Hence, the convergence of the solver can be studied by examining the convergence of the discrete solutions to the discrete solution on some sufficiently fine grid, rather than to the discrete solution on an infinitely fine grid. The actual procedure to be used to perform the grid refinement study is therefore

1. A suitable geometry and a set of appropriate boundary conditions are selected.
2. The solution of the governing equations is obtained on progressively finer grids and compared to the approximate solution on the finest such grid to compute the approximate error fields across each grid.
3. An appropriate norm is used to evaluate the approximate error $E(h)^{\varphi}$ in a flow variable $\varphi$ the solution obtained on each grid.
4. The reduction in the approximate error as the $h$ is reduced is assessed.

The final issue to be discussed is the assessment of the reduction in the approximate error as $h$ is reduced. The approximate error $E(h)^{\varphi}$ can be expanded as a Taylor series in the grid coarseness

$$E(h)^{\varphi} = a_0 + a_1 h + a_2 \frac{h^2}{2!} + a_3 \frac{h^3}{3!} + \mathcal{O}\left(h^4\right) \tag{4.71}$$

where the $a_i$ are unknown constants and $g(h) = \mathcal{O}(h^4)$ means $\lim_{h\to 0}\left(g(h)/h^4\right)$ is bounded.

In order for the scheme to be consistent, the constant error term $a_0$ must be zero, otherwise the error $E(h)^{\varphi}$ cannot possibly tend to zero as the grid is refined. For a higher-order scheme it is expected, additionally, that at least one of the other $a_j$ is also zero, leaving a Taylor series of the form

$$E(h)^{\varphi} = a_p h^p + a_{p+1} h^{p+1} + a_{p+2} h^{p+2} + \mathcal{O}\left(h^{p+3}\right) \tag{4.72}$$

for some $p \in \mathbb{N}^+$. For sufficiently small $h$ only the leading term in this series will make a significant contribution to the error, leaving

$$E(h)^{\varphi} \approx a_p h^p \tag{4.73}$$

$p$ is therefore the required approximation to the order of accuracy of the inviscid solver. Some trivial algebraic manipulation yields

$$\ln E(h)^{\varphi} \approx p \ln h + c \tag{4.74}$$

where $c \coloneqq \ln a_p$. The approximation to the order of accuracy of the solver is therefore obtained by plotting $\ln E(h)$ against $\ln h$ for several values of $h$, finding the region of the scatter plot where $h$ is sufficiently small for the above manipulation to be justified (this region is called the *asymptotic region*, and is the region where the scatter plot becomes approximately linear), and using linear regression on that region to determine the gradient.

### 4.7.2 Results

The grid refinement study was performed on the geometry of the first test case presented in Section 4.8, the NACA 0012 aerofoil. The flow conditions chosen were $M_r = 0.5$, $\alpha = 4°$. The simple geometry allows reliable and smooth refinement of the mesh, whilst also containing relatively steep gradients in the primary flow variables. In addition the lack of shocks leads to only modest activation of the limiters, which would otherwise lead to significant suppression of the second-order terms.

The geometry of both the aerofoil and farfield used for the convergence study was identical to that used for the subsonic case in Section 4.8.1. The target meshwidth at the farfield was set to 32 chords, with the maximum $\partial h / \partial \left\| \boldsymbol{x} \right\|$ across the grid limited to 0.15, where $h\left(\boldsymbol{x}\right)$ is the target meshwidth at $\boldsymbol{x}$.

As briefly mentioned in the previous subsection, a one-parameter family of grids is required. This is turn requires a one-parameter family of functions representing the target meshwidth at each point on the surface of the aerofoil. Let the parameter represent the maximum target meshwidth on the surface of the aerofoil, and denote it by $h_{\max}$. A suitable formula for the target meshwidth at any other point on the aerofoil was found to be

$$h\left(\kappa\right) = h_{\max} \left(\frac{5}{8}\right)^{\frac{|\kappa|}{|\kappa|_{\max}}} \tag{4.75}$$

where $\kappa$ is the curvature of the aerofoil surface at a point on it, and $|\kappa|_{\max}$ is the maximum unsigned curvature over the whole aerofoil surface. The singularity in the curvature at the trailing edge is removed by setting the curvature there equal to the maximum curvature found elsewhere on the aerofoil. Finally, Laplacian smoothing is applied to the target meshwidth distribution.

The coarsest grid was obtained using $h_{\max} = 1 \times 10^{-2}$ and had $11,072$ cells. The finest grid was obtained using $h_{\max} = 1.75 \times 10^{-4}$ had $343,486$ cells, and the solution obtained on this grid was the reference solution against which the pressure errors on the aerofoil surface were computed for each of the coarser grids. This was done by

creating a piecewise cubic polynomial interpolant on the aerofoil surface for each grid, interpolating to the aerofoil surface nodes of the coarsest grid, and then computing the errors and the norms. Four norms were computed:

1. The $\mathcal{L}_1$ norm
$$\|\boldsymbol{\varepsilon}_\varphi\|_{\mathcal{L}_1} \coloneqq \sum_{i=0}^{N_b-1} |\varepsilon_{\varphi_i}| \tag{4.76}$$
where $N_b$ is the number of boundary nodes on the coarsest grid, $\boldsymbol{\varepsilon}_\varphi \in \mathbb{R}^{N_b}$ is the vector of $\varphi$ errors at the coarsest grid nodes, and $\varepsilon_{\varphi_i}$ is the $\varphi$ error at node $i$.

2. The (pointwise) $\mathcal{L}_2$ norm
$$\|\boldsymbol{\varepsilon}_\varphi\|_{\mathcal{L}_2} \coloneqq \sqrt{\sum_{i=0}^{N_b-1} \varepsilon_{\varphi_i}^2} \tag{4.77}$$

3. The $\mathcal{L}_\infty$ norm
$$\|\boldsymbol{\varepsilon}_\varphi\|_{\mathcal{L}_\infty} \coloneqq \max_{0\leq i<N_b} \varepsilon_{\varphi_i}^2 \tag{4.78}$$

4. The (functional) $\mathcal{L}_2$ norm
$$\|\boldsymbol{\varepsilon}_\varphi\|_{\mathcal{F}_2} \coloneqq \sqrt{\int_{\Gamma_b} E_\varphi(s)^2 ds} \tag{4.79}$$
where $E_\varphi(s)$ is a piecewise cubic interpolant of $\varepsilon_{\varphi_i}$ over the aerofoil surface $\Gamma_b$.

The results are summarised in Table 4.2. With the errors calculated, the natural logarithm of the meshwidth is plotted against the natural logarithm of each of the error norms. The result is shown in Figure 4.6.

As expected, the plots have two regions. For $\ln h$ greater than about $-7.75$, the curves are not closely approximated by straight lines. In this region, the assumptions that lead to (4.74) are not valid, and so there is no reason why the curves should be approximately linear. On the other hand, when $\ln h$ is less than about $-7.75$, the curves do become approximately linear. This is the aforementioned asymptotic region. In this region, the meshwidth is sufficiently small that the trucation error is completely dominated by the leading term, $a_p h^p$. (4.74) is therefore valid, and so by using linear regression on the data points in the asymptotic region the sought-after approximation to the spatial accuracy of the solver, $p$, can be determined. The best-fit lines resulting from this linear regression are shown on Figure 4.6, and the calculated approximations to $p$ according to each of the above norms are shown on the left of Table 4.3. Clearly,

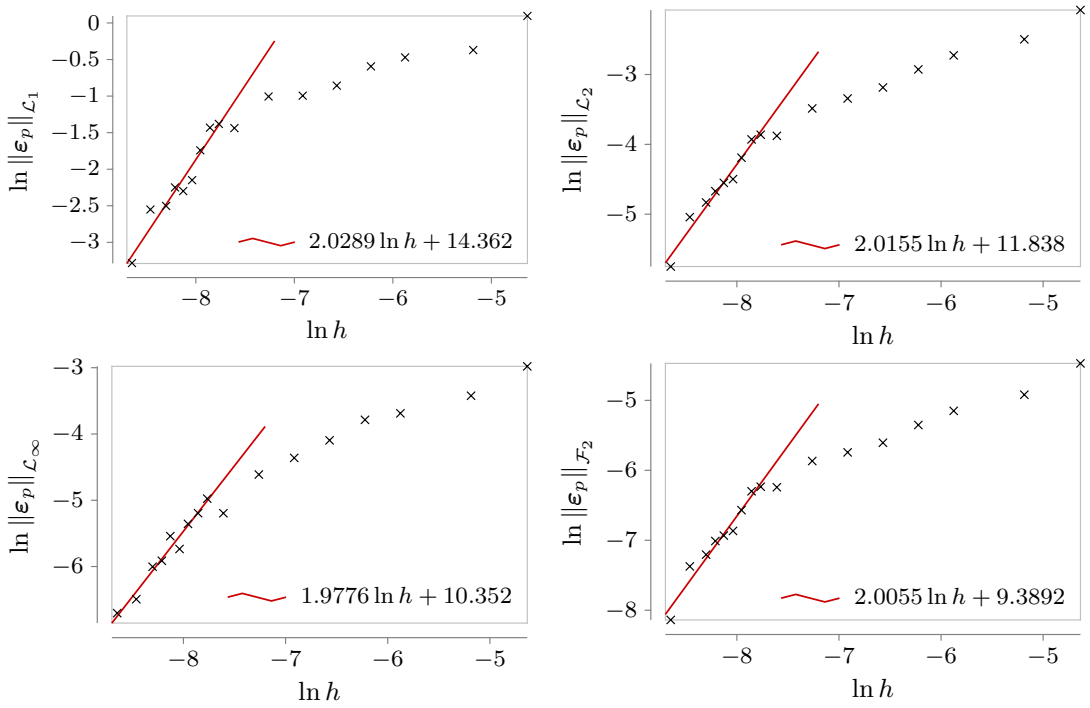



Figure 4.6: The natural logarithm of the meshwidth against the natural logarithms of the pressure error in each of the norms, together with best-fit lines placed using linear regression on the nine data points with the lowest values of the meshwidth.

| $h$ | $N_{\text{cells}}$ | $\|\boldsymbol{\varepsilon}_p\|_{\mathcal{L}_1}$ | $\|\boldsymbol{\varepsilon}_p\|_{\mathcal{L}_2}$ | $\|\boldsymbol{\varepsilon}_p\|_{\mathcal{L}_\infty}$ | $\|\boldsymbol{\varepsilon}_p\|_{\mathcal{F}_2}$ |
|---|---|---|---|---|---|
| 9.71E-03 | 11072 | 1.10E+00 | 1.25E-01 | 5.08E-02 | 1.14E-02 |
| 5.60E-03 | 15246 | 6.90E-01 | 8.24E-02 | 3.26E-02 | 7.30E-03 |
| 2.81E-03 | 24513 | 6.24E-01 | 6.55E-02 | 2.50E-02 | 5.80E-03 |
| 1.98E-03 | 31886 | 5.52E-01 | 5.35E-02 | 2.27E-02 | 4.74E-03 |
| 1.40E-03 | 41424 | 4.25E-01 | 4.13E-02 | 1.66E-02 | 3.68E-03 |
| 9.92E-04 | 56623 | 3.70E-01 | 3.53E-02 | 1.28E-02 | 3.20E-03 |
| 7.02E-04 | 77860 | 3.66E-01 | 3.06E-02 | 9.93E-03 | 2.83E-03 |
| 4.96E-04 | 107452 | 2.37E-01 | 2.07E-02 | 5.54E-03 | 1.95E-03 |
| 4.24E-04 | 124758 | 2.51E-01 | 2.10E-02 | 6.90E-03 | 1.96E-03 |
| 3.87E-04 | 135652 | 2.39E-01 | 1.97E-02 | 5.54E-03 | 1.84E-03 |
| 3.51E-04 | 148924 | 1.75E-01 | 1.51E-02 | 4.72E-03 | 1.40E-03 |
| 3.23E-04 | 160644 | 1.17E-01 | 1.11E-02 | 3.23E-03 | 1.04E-03 |
| 2.95E-04 | 175708 | 1.00E-01 | 1.06E-02 | 3.92E-03 | 9.75E-04 |
| 2.72E-04 | 190096 | 1.05E-01 | 9.37E-03 | 2.70E-03 | 9.02E-04 |
| 2.48E-04 | 207276 | 8.19E-02 | 7.97E-03 | 2.47E-03 | 7.43E-04 |
| 2.12E-04 | 241416 | 7.80E-02 | 6.46E-03 | 1.51E-03 | 6.28E-04 |
| 1.75E-04 | 288904 | 3.75E-02 | 3.19E-03 | 1.23E-03 | 2.92E-04 |

Table 4.2: The computed error norms.

| | |
|---|---|
| $\|\boldsymbol{\varepsilon}_p\|_{\mathcal{L}_1}$ | 2.0289 |
| $\|\boldsymbol{\varepsilon}_p\|_{\mathcal{L}_2}$ | 2.0155 |
| $\|\boldsymbol{\varepsilon}_p\|_{\mathcal{L}_\infty}$ | 1.9776 |
| $\|\boldsymbol{\varepsilon}_p\|_{\mathcal{F}_2}$ | 2.0055 |

| | |
|---|---|
| $\|\boldsymbol{\varepsilon}_\rho\|_{\mathcal{L}_1}$ | 1.9468 |
| $\|\boldsymbol{\varepsilon}_\rho\|_{\mathcal{L}_2}$ | 1.9488 |
| $\|\boldsymbol{\varepsilon}_\rho\|_{\mathcal{L}_\infty}$ | 2.1476 |
| $\|\boldsymbol{\varepsilon}_\rho\|_{\mathcal{F}_2}$ | 1.9518 |

Table 4.3: The computed approximations to the spatial accuracy of the inviscid solver, using the pressure and density errors on the aerofoil surface and each of the four norms defined previously.

all four norms produced approximations to the spatial accuracy that are very nearly two. This confirms that the inviscid solver achieves second-order spatial accuracy, as claimed.

Finally, an additional confirmation of the second-order spatial accuracy of the inviscid solver can be obtained by repeating the above, using the density on the aerofoil surface in place of the pressure. The results are shown on the right of Table 4.3.

## 4.8 Presentation and Discussion of Numerical Results

The accuracy, robustness, and performance of the inviscid Mach-uniform scheme is now demonstrated by application to several well-known test cases. Several of the test cases

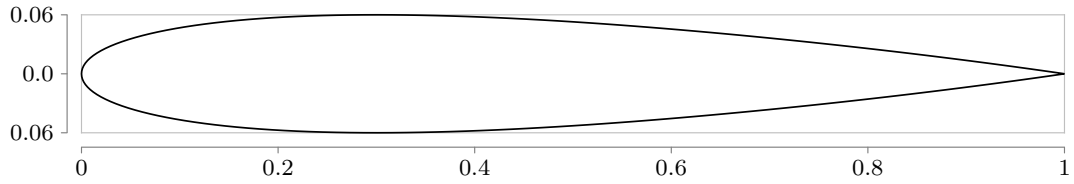



Figure 4.7: The geometry of the NACA 0012 Aerofoil.

are taken from the 1986 GAMM workshop [Der+89], as it provides a large number of computed solutions for comparison to the present solver.

### 4.8.1 NACA 0012 Aerofoil

This is a classic test case consisting of inviscid flow at various Mach numbers over a simple symmetric aerofoil. The aerofoil geometry is shown in Figure 4.7. For all cases slip wall boundary conditions are imposed on the aerofoil walls, and farfield boundary conditions are imposed on the farfield. The chord length is denoted $c$, and all lengths reported are nondimensionalised by dividing by the chord length.

To create the farfield geometry, two circular arcs (representing the inlet and outlet parts of the farfield boundary) were joined, creating an interior angle of 110° at the top and bottom which ensures that the farfield flow is not nearly tangential to the boundary at any point. This prevents the issue discussed in Section 4.5.2, where convergence can be harmed by farfield boundary edges that are nearly tangential to the farfield flow rapidly switching type between inlet and outlet as the solution progresses. The resulting farfield boundary being approximately 70 chords from the aerofoil in the streamwise direction and 127 chords from the aerofoil in the transverse direction.

Three flow cases were considered, corresponding to subsonic, transonic, and supersonic flow. A different grid was generated for each, to allow for refinement at the shocks generated in the latter two cases.

#### Subsonic

This test case involves flow that is well into the compressible range, at $M_\infty = 0.63$, but which is everywhere subsonic. There are therefore significant density gradients, but no shocks. The chosen angle of attack of 2° results in a small amount of lift being generated, but no noticeable separation.

For this case the target meshwidth on the surface of the aerofoil varied from $4.42 \times 10^{-3}$ to $5.66 \times 10^{-4}$ according to the local boundary curvature. This resulted

| | $c_L$ | $c_D$ | $c_M$ |
|---|---|---|---|
| C1 [Ang+89] | 0.336 | 0.0004 | -0.0023 |
| C6 [Dad89] | 0.3332 | 0.00043 | -0.00296 |
| C11 [HK89] | 0.3291 | 0.0007 | -0.0021 |
| C14 [LS89] | 0.3327 | 0.0003 | -0.0028 |
| C15 [ML89] | 0.33417 | 0.000087 | -0.001818 |
| C17 [PLA89] | 0.3323 | 0.0045 | 0.0051 |
| C20 [TWV89] | 0.3335 | 0.00003 | - |
| Present solver | 0.33514 | 0.00085 | -0.00237 |

Table 4.4: Computed aerodynamic coefficients for the NACA 0012 aerofoil at $M_\infty = 0.63$, $\alpha_\infty = 2°$, compared to those provided in contributions to the GAMM workshop [Der+89].

in 410 nodes being generated on the aerofoil surface. The selected target growth rate of 1.15 resulted in $7,828$ nodes and $15,246$ cells across the whole domain. The grid geometry in the neighbourhood of the leading edge of the aerofoil is shown on the left of Figure 4.8. The Courant number used for this case was 10, resulting in convergence to $\Theta = -5.5$ in $4,704$ iterations.

As shown in Table 4.4, the computed aerodynamic coefficients agree well with the results from the GAMM workshop. Additionally, the contour plots in Figure 4.8 and surface plots in Figure 4.9 show good agreement.

**Transonic**

This test case is in the high transonic range at $M_\infty = 0.85$, with an angle of attack of $\alpha_\infty = 1°$. Isolated shocks appear on the top and bottom surfaces of the aerofoil, but there is no bow shock.

For this case the target meshwidth on the surface of the aerofoil varied from $4.0 \times 10^{-3}$ to $6.0 \times 10^{-3}$ according to the local boundary curvature. This resulted in 398 points being generated on the aerofoil surface. For this test case, oblique shocks on the top and bottom surface of the aerofoil were expected. The approximate location of these shocks was found by solving on a coarse grid. Line sources were then used to refine the grid in the regions were the shocks were predicted, to better resolve the strong gradients in the flow variables. The selected target growth rate of 1.15 resulted in $14,214$ points and $28,030$ cells across the whole domain. A section of the grid used, showing the line source refinement, is given on the left of Figure 4.10. The Courant number used for this case was 5, resulting in convergence to $\Theta_\varphi = -5.5$ in $7,878$ iterations.

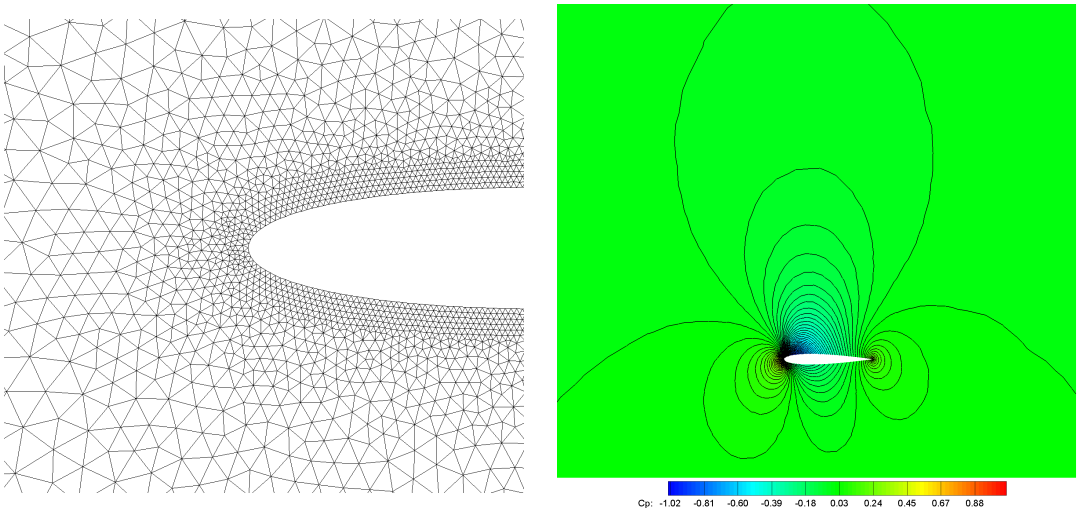



Figure 4.8: Left: a close-up of the grid used for the NACA 0012 aerofoil at $M_\infty = 0.63$, $\alpha_\infty = 2°$. Right: the resulting coefficient of pressure contours.

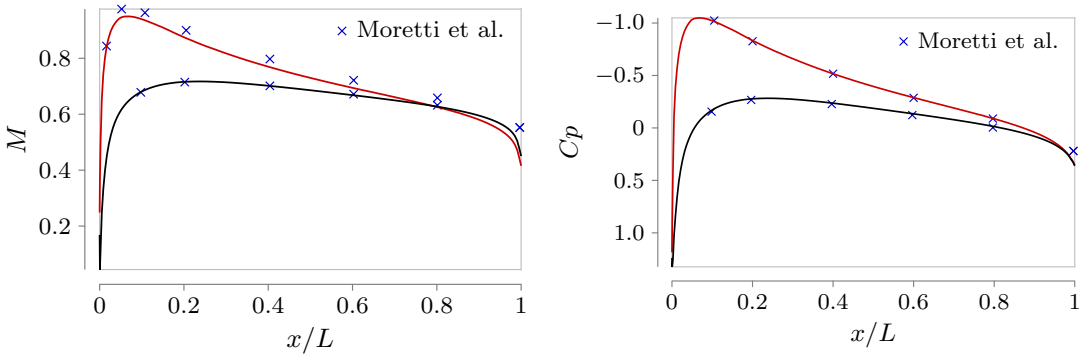



Figure 4.9: Mach number and coefficient of pressure plots along the top wall (red line) and bottom wall (black line) on the NACA 0012 aerofoil at $M_\infty = 0.63$, $\alpha_\infty = 2°$, compared to results from [ML89].

| | $c_L$ | $c_D$ | $c_M$ | $x_U$ | $x_L$ |
|---|---|---|---|---|---|
| C1 [Ang+89] | 0.374 | 0.0565 | -0.1208 | 0.81:0.87 | 0.59:0.63 |
| C2 [BM89] | 0.3721 | 0.0566 | 0.1317 | 0.84:0.86 | 0.60:0.63 |
| C4 [Cha+89] | 0.334 | 0.0547 | 0.0965 | 0.81:0.84 | 0.60:0.63 |
| C5 [CV89] | 0.3569 | 0.0555 | -0.1195 | 0.81:0.86 | 0.59:0.63 |
| C6 [Dad89] | 0.3572 | 0.0596 | -0.1323 | 0.84:0.93 | 0.62:0.70 |
| C7 [DDF89] | 0.3847 | 0.0585 | -0.122 | 0.84:0.89 | 0.56:0.62 |
| C8 [EA89] | 0.3860 | 0.0523 | -0.1077 | 0.82:0.85 | 0.58:0.60 |
| C8 [Der+89] | 0.3965 | 0.0507 | -0.1130 | - | - |
| C11 [HK89] | 0.3565 | 0.0582 | -0.1209 | 0.84:0.89 | 0.63:0.67 |
| C13 [KR89] | 0.330 | 0.0528 | -0.104 | 0.78:0.87 | 0.57:0.67 |
| C14 [LS89] | 0.3697 | 0.0575 | -0.1264 | 0.84:0.89 | 0.61:0.63 |
| C15 [ML89] | 0.3692 | 0.05426 | -0.1286 | 0.86 | 0.62 |
| C17 [PLA89] | 0.3586 | 0.0514 | -0.1170 | 0.83:0.86 | 0.56:0.59 |
| C18 [SM89] | 0.363 | 0.0562 | -0.130 | 0.81:0.87 | 0.59:0.64 |
| C18 [Der+89] | 0.357 | 0.0578 | -0.128 | - | - |
| C20 [TWV89] | 0.3793 | 0.0576 | -0.1290 | 0.86:0.87 | 0.62:0.64 |
| Present solver | 0.3386 | 0.0639 | -0.1205 | 0.871 | 0.675 |

Table 4.5: Computed aerodynamic coefficients and shock locations for the NACA 0012 aerofoil at $M_\infty = 0.85$, $\alpha_\infty = 1°$, compared to those provided in contributions to the GAMM workshop [Der+89]. $x_U$ and $x_L$ refer to the $x/c$ positions of the shocks on the upper and lower surfaces of the aerofoil respectively. Where the computed shock is quite thick, the approximate range of $x/c$ values covering the shock is provided instead of a single location.

This is another test case from the 1987 GAMM workshop. The editors noted that there were "striking discrepancies" among the computed aerodynamic coefficients, which are tabulated in Table 4.5. Evidentally, the present solver predicts values of coefficient of lift and coefficient of pitching moment that lie within the range of predictions presented by contributors to the workshop. However, the predicted coefficient of drag is somewhat larger than that calculated by any of the contributors. Since the flow is inviscid, the only source of drag in the exact solution should be the shocks. Comparing the contour and surface plots presented in Figures 4.10 and 4.11 to those provided by the contributors suggests a possible reason: the present solver achieves significantly better shock resolution. This is likely due to the use of a superior convective flux reconstruction scheme and limiter than were available at the time of the workshop as well as a finer grid, especially in the region of the shocks. The present solver also predicts that the shocks occur towards the higher end of the locations predicted by the workshop contributors.

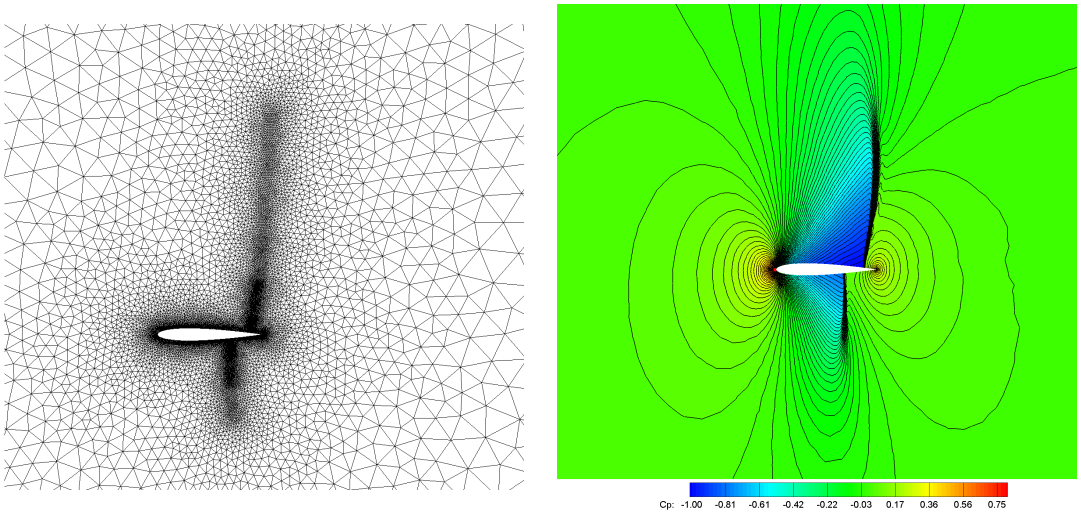



Figure 4.10: Left: a section of the grid used for the NACA 0012 aerofoil at $M_\infty = 0.85$, $\alpha_\infty = 1°$. Right: the resulting coefficient of pressure contours.

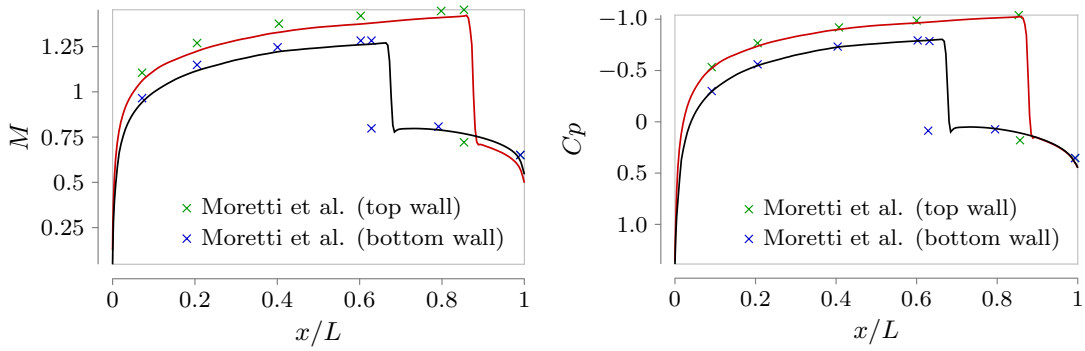



Figure 4.11: Mach number and coefficient of pressure plots along the top wall (red line) and bottom wall (black line) on the NACA 0012 aerofoil at $M_\infty = 0.85$, $\alpha_\infty = 1°$, compared to results from [ML89].

### Supersonic

This test case is above the transonic range at $M_\infty = 1.5$ with zero angle of attack. The shocks have detached from the aerofoil and travelled upstream, yielding a bow shock. There is also a pair of oblique shocks attached to the trailing edge.

For this case the target meshwidth on the surface of the aerofoil varied from $5.6 \times 10^{-3}$ to $7.13 \times 10^{-3}$ according to the local boundary curvature. This resulted in 359 points being generated on the aerofoil surface. As for the transonic case the locations of the shocks was predicted by first obtaining a solution on a coarse grid, then line sources were used to refine the grid in those regions. Since the Mach number is now higher, the gradients at the shocks are expected to be sharper. This is especially true of the bow shock, since it is perpendicular to the flow. The selected target growth rate of 1.15 resulted in $20,036$ points and $39,713$ cells across the whole domain. A section of the grid used, showing the line source refinement, is given on the left of Figure 4.12. The Courant number used for this case was 1.5, resulting in convergence to $\Theta_\varphi = -5.5$ in $7,027$ iterations.

The results are shown in Figures 4.12, 4.13, and 4.14, and compare favourably to those presented in [BWM15] and [BWM15]. For example, the computed coefficient of drag of 0.1013 is close to the value of 0.09630 given in [BWM15]. The computed coefficient of lift was $-6.1673 \times 10^{-4}$ compared to the theoretical value of 0 for a symmetric aerofoil at zero angle of attack. In addition, the contour plots demonstrate very crisp resolution of the shocks.

Since inviscid solutions are currently being considered the only source of entropy production should be shocks. However, there will also be non-physical entropy production due to numerical diffusion. Figure 4.13 provides a visual confirmation that this spurious entropy production is small compared to the physical entropy production at the shocks. At the ends of the shocks, there appears to be a sudden loss of resolution on the entropy contours. However, this is simply where the shock grid refinement terminates.

An additional confirmation of the quality of the results is obtained from the surface plots of Mach number and coefficient of pressure in Figure 4.14. Since the geometry and boundary conditions are exactly symmetrical about the chord line of the aerofoil, the top and bottom wall surface plots should coincide. This has clearly been achieved, apart from at the trailing edge of the aerofoil, where the non-differentiability of the geometry causes a drop in accuracy of the flow variable reconstructions.

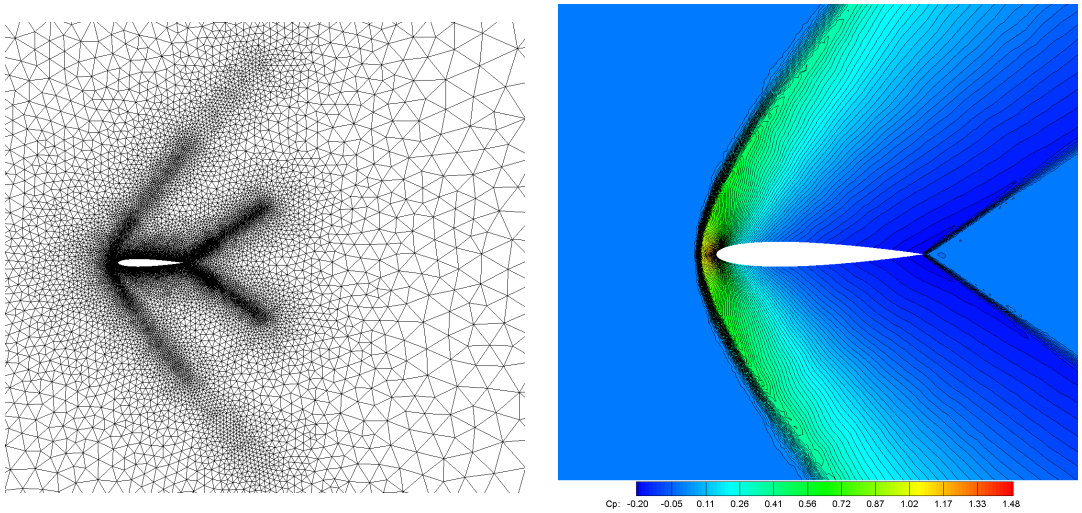



Figure 4.12: Left: a section of the grid used for the NACA 0012 aerofoil at $M_\infty = 1.5$, $\alpha_\infty = 0°$. Right: the resulting coefficient of pressure contours.

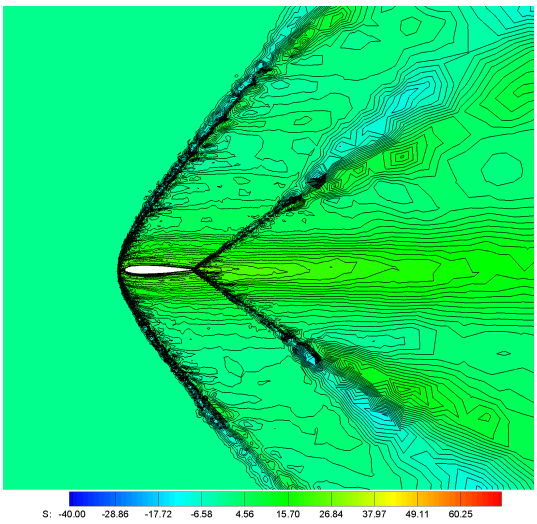



Figure 4.13: Entropy contours for the NACA 0012 aerofoil at $M_\infty = 1.5$, $\alpha_\infty = 0°$.

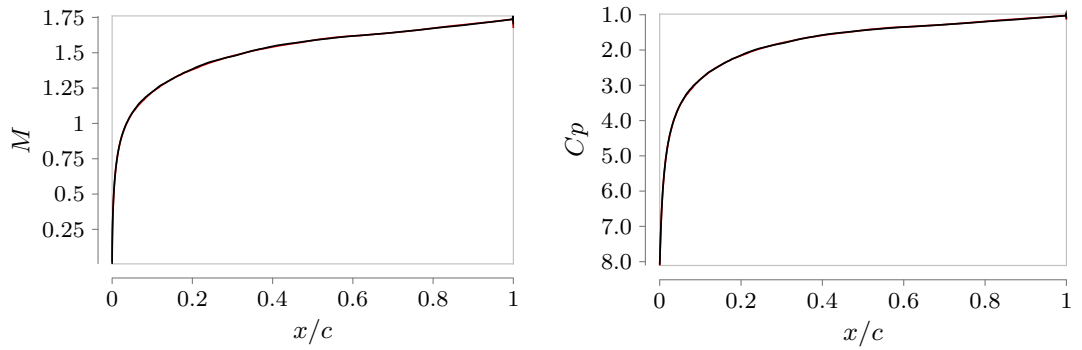



Figure 4.14: Mach number and coefficient of pressure plots along the top wall (red line) and bottom wall (black line) on the NACA 0012 aerofoil at $M_\infty = 1.5$, $\alpha_\infty = 0°$. The top wall values are barely visible due to the symmetry of the solution.

### 4.8.2 Quasi-Incompressible Flow Over a Right Circular Cylinder

The second test case is the quasi-incompressible ($M_r = 0.1$) flow over a right circular cylinder of unit diameter. This is a particularly useful test of the low Mach number performance of the solver due to the existence of an analytic solution in the zero Mach number limit. In addition the true solution is symmetric about the line perpendicular to the farfield flow through the centre of the cylinder. As a result, the deviation from this symmetry in the computed results provides a useful indication of the amount of numerical dissipation present.

Slip wall boundary conditions were imposed on the cylinder wall, and farfield boundary conditions are imposed on the farfield. The farfield definition remains the same as that used for the NACA 0012 aerofoil case in the previous subsection. The target meshwidth on the cylinder wall was set to $4.25 \times 10^{-3}$, with a point source at the trailing edge reducing this to $3.75 \times 10^{-3}$. This resulted in 740 points on the cylinder wall. The target growth rate was set to 1.12, which resulted in $16,746$ nodes and $32,684$ cells across the complete grid. A section of the grid is shown on the left of Figure 4.15. The selected Courant number of 25.0 yielded convergence to $\Theta_\varphi = -5.5$ in $3,541$ iterations. The results are shown on the right of Figure 4.15 and in Figure 4.16. Both figures show that very good symmetry about the transverse axis has been achieved, as is expected in inviscid incompressible flow where the geometry has this symmetry. This indicates that numerical diffusion and unphysical entropy generation are both low. Figure 4.16 also shows good agreement between the computed results and the analytic solution for incompressible flow. There is minor disagreement at the leading and trailing edge stagnation points. At the leading edge stagnation point, the pressure

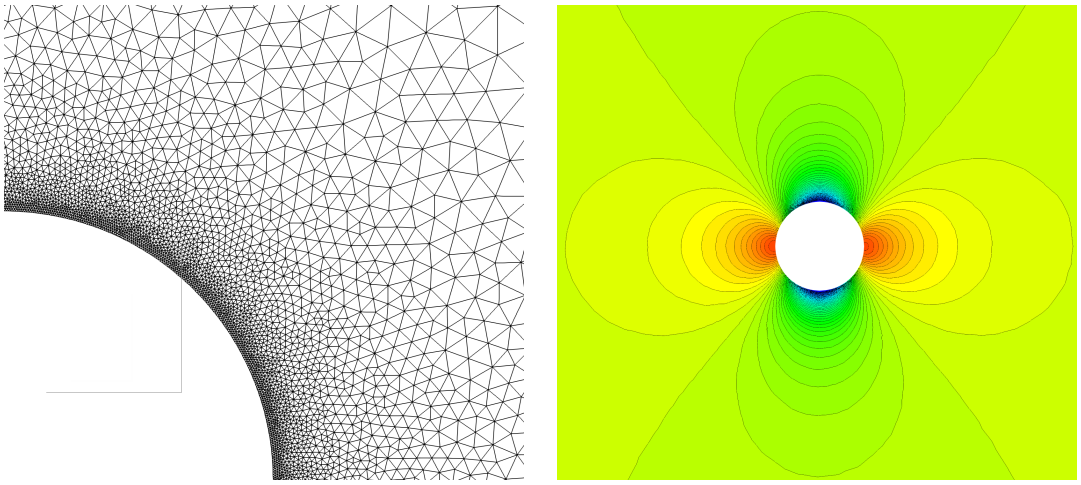


Figure 4.15: Left: Section of the mesh used for the right circular cylinder case. Right: Computed coefficient of pressure contours.

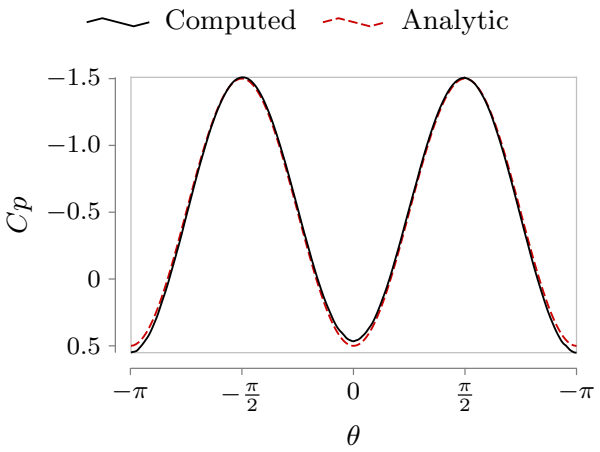



Figure 4.16: Plot of the computed coefficient of pressure on the top surface of the cylinder in quasi-incompressible flow against the analytic results for incompressible flow.

is higher than the analytic value, largely because a very slight density increase occurs in the quasi-incompressible calculation. Conversely, at the trailing edge stagnation point the pressure is lower than the analytic value, primarily due to total pressure loss from numerical diffusion.

### 4.8.3 Channel with Bump

The third test case is the well-studied channel flow over a sinusoidal bump. The bump length, channel width, distance from inlet to bump leading edge, and distance from bump trailing edge to outlet are equal and denoted by $L$, and all lengths reported are non-dimensionalised by dividing by $L$.

#### Subsonic

Despite being a particularly simple flow case, subsonic channel flow over a 10% bump provides some useful qualitative information on the numerical diffusion associated with the solver. Since the geometry is symmetrical about a line through the centre of the bump in the y-direction and no shocks nor physical diffusion are present in the flow, the exact solution will be symmetrical and reversible. Hence, any deviation from symmetry in the computed flow variable contours must be due to numerical diffusion.

Slip wall boundary conditions are imposed on the top and bottom walls, and pressure inlet and pressure outlet boundary conditions are imposed at the inlet and outlet respectively. The target meshwidth on the boundary varied from $9.375 \times 10^{-3}$ at the points of highest curvature to $2.8125 \times 10^{-2}$ at the points of zero curvature. There are discontinuities in the curvature of the geometry at the leading edge and trailing edge of the bump. At these, point sources were used to reduce the targe meshwidth to 0.01 to help ensure that the inevitably large gradents are properly resolved. This resulted in 326 boundary nodes, of which 108 were on the top wall and 179 were on the bottom wall, including 89 on the bump itself. The target growth rate of 1.07 results in $8,889$ nodes and $14,893$ cells across the whole grid. The grid is shown in Figure 4.17. The selected Courant number was 10.0, which resulted in convergence to $\Theta_\varphi = -5.5$ in $2,215$ iterations.

The results are shown in Figures 4.18 and 4.19. The expected symmetry in the solution is seen to be well achieved, especially in the coefficient of pressure contours. Visually, this symmetry compares favourably to the corresponding contour plots in [WSW02] and [DLP93]. In addition, the surface plots show good quantitative agreement with the results presented in [DLP93]. For example, the authors report Mach

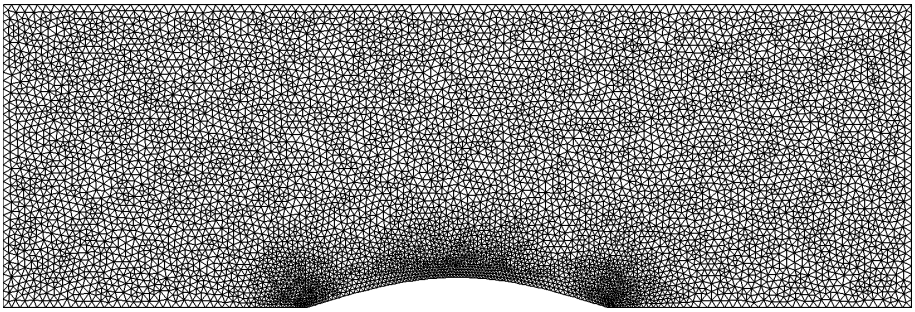


Figure 4.17: The grid used for the channel with 10% bump and $M_{\text{inlet}} = 0.5$.

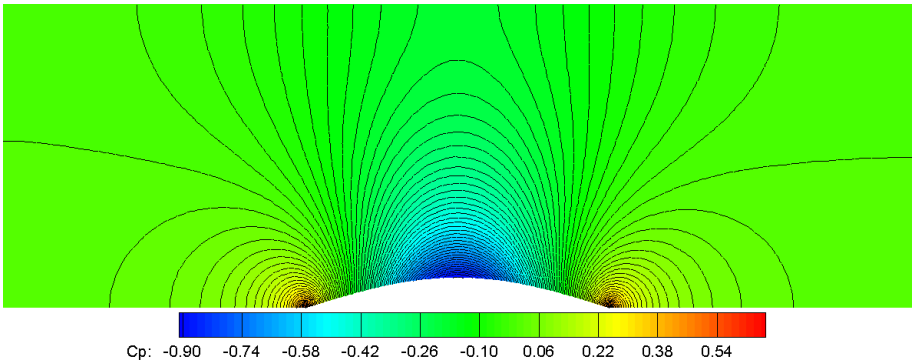



Figure 4.18: Coefficient of pressure contours for the channel with 10% bump and $M_{\text{inlet}} = 0.5$.

number peaks on the bottom and top walls of 0.72 and 0.54 respectively, compared to the values of 0.719 and 0.547 presented here.

### Transonic

Transonic channel flow over the 10% bump is now considered. It is worth mentioning an interesting phenomenon that was encountered when the inlet Mach number was at the top end of the transonic range, $0.8 \lesssim M_{\text{inlet}} < M_{\text{crit}}$, where $M_{\text{crit}}$ is some critical Mach number. At the beginning of convergence, an attached oblique shock forms on the bump. As convergence progresses the shock moves upstream all the way to the inlet, becoming normal in the process. The shock then remains in this position as convergence continues. This phenomenon was also noted in [DLP93]. The authors stated that this behaviour is essentially that of a choked nozzle and reported a value for the critical Mach number of $M_{\text{crit}} = 1.37$. The useful range for testing the solver with this geometry therefore excludes the range of inlet Mach numbers $0.8 \lesssim M_{\text{inlet}} \lesssim 1.37$, apart from obtaining confirmation that the present solver exhibits the same behaviour.

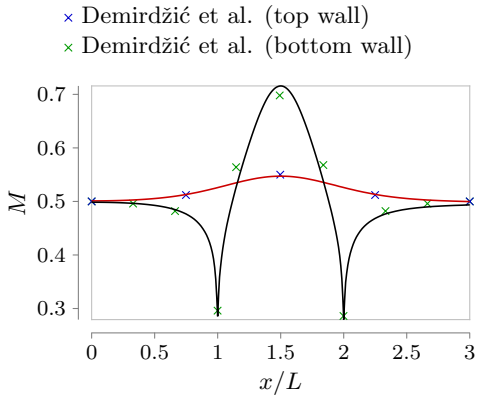



Figure 4.19: Mach number plots along the bottom wall (solid black line) and top wall (dashed red line) for the channel with 10% bump and $M_{\text{inlet}} = 0.5$. Some data points reported in [DLP93] are shown for comparison with the present solver.

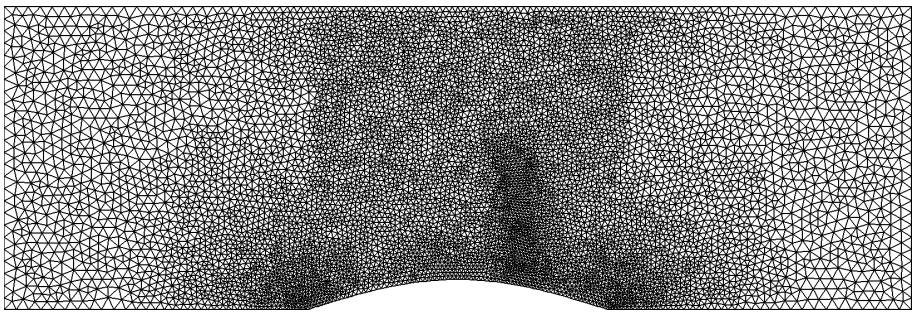


Figure 4.20: The grid used for the channel with 10% bump and $M_{\text{inlet}} = 0.675$.

This was indeed the case.

The boundary conditions imposed are the same as those used for the subsonic case. The target meshwidth on the boundary varied from $1.25 \times 10^{-2}$ at the points of highest curvature to $4.0 \times 10^{-2}$ at the points of zero curvature. Point sources are again used at the leading and trailing edges. One (approximately normal) shock was expected on the bump, and so line sources were used to ensure proper resolution. A polygonal source was also used to slightly reduce the meshwidth in the neighbourhood of the bump where the highest gradients were expected. This resulted in 325 boundary points, of which 114 were on the top wall and 164 were on the bottom wall, including 80 on the bump itself. The selected target growth rate of 1.07 resulted in $8,889$ nodes and $17,450$ cells across the whole grid. The grid is shown in Figure 4.20. . The selected Courant number was 7.5, which resulted in convergence to $\Theta_\varphi = -5.5$ in $3,504$ iterations.

The inlet Mach number was set to $M_{\text{inlet}} = 0.675$, which results in a single, station-

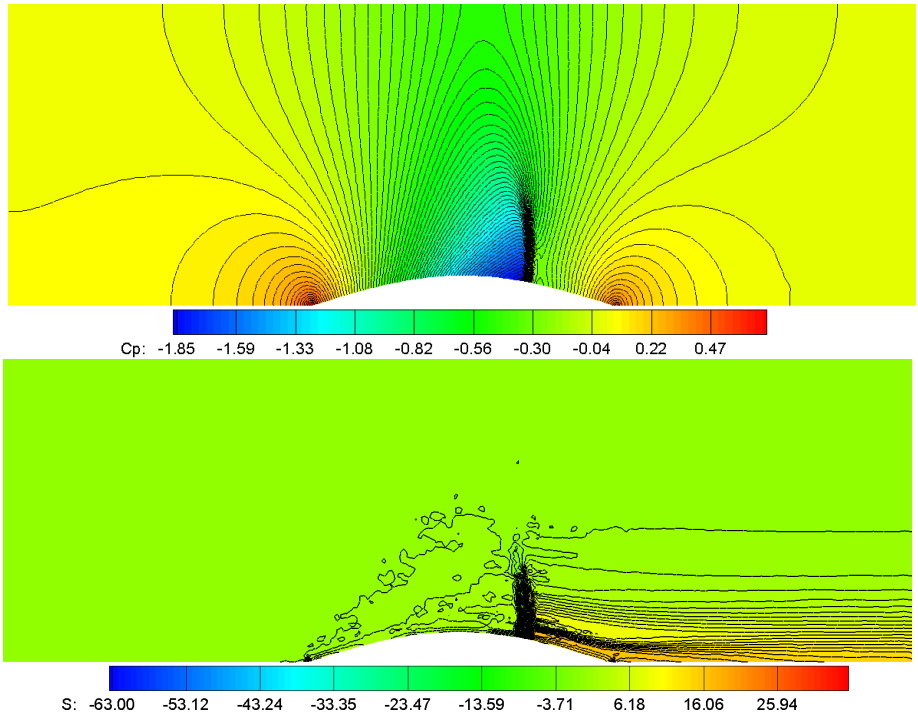



Figure 4.21: Mach number (top) and entropy (bottom) contours for the channel with 10% bump and $M_{\text{inlet}} = 0.675$.

ary attached shock on the bump at $x \approx 1.7076$. The results are shown in Figures 4.21 and 4.22. In theory, inviscid flows generate entropy only at shocks [RG14, p. 242]. The entropy contours are in reasonably close agreement with this theoretical principle, with virtually all of the entropy generation appearing at the shock. Practical solvers do always have some degree of numerical dissipation across the solution, which is reflected in the existence of non-zero entropy upstream of the shock. This non-zero entropy is small compared to the entropy downstream of the shock however, demonstrating that the numerical dissipation inherent in the solver is small (at least for this test case). The selected Courant number was 7.5, which resulted in convergence to $\Theta_\varphi = -5.5$ in $3,745$ iterations.

Reasonably close agreement is achieved with the results presented in [DLP93] and [PC93]. The shock locations along the bottom wall of $x \approx 1.71$ and $x \approx 1.72$ reported in those sources compared to $x \approx 1.7076$ found here is particularly encouraging. In addition, the contour and surface plots can be seen to be very similar.

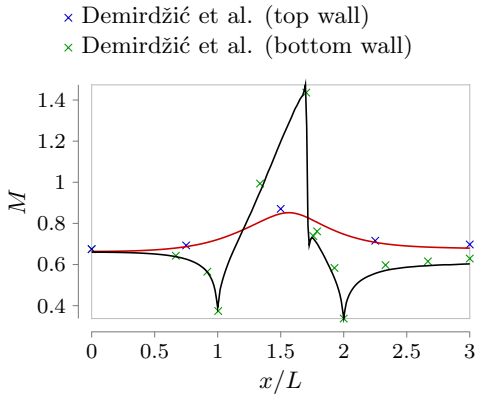



Figure 4.22: Mach number plots along the bottom wall (solid black line) and top wall (dashed red line) for the channel with 10% bump and $M_{\text{inlet}} = 0.675$. Some data points reported in [DLP93] are shown for comparison with the present solver.

### Supersonic

For this case the bump is reduced in height to 4% of the channel height. Selecting an inlet Mach number of 1.65 produces a pair of shocks, at the leading and trailing edges of the bump. The trailing edge shock points directly into the outlet, but the leading edge shock hits the top wall and is reflected.

Slip wall boundary conditions are imposed at the top and bottom walls, and supersonic pressure inlet and supersonic pressure outlet conditions are imposed at the inlet and outlet respectively. The target meshwidth on the boundary varied from $1.408 \times 10^{-2}$ at the points of highest curvature to $2.65 \times 10^{-2}$ at the points of zero curvature. Point sources were once again placed at the leading and trailing edges of the bump. Line sources were used to improve the resolution of the expected oblique shocks, and were again placed by trial and error. This resulted in 373 boundary points, of which 126 were on the top wall and 155 were on the bottom wall, including 66 on the bump itself. The selected target growth rate of 1.07 resulted in $10,761$ nodes and $21,147$ cells across the whole grid. The grid is shown in Figure 4.23. The selected Courant number was 2.0, which resulted in convergence to $\Theta_\varphi = -5.5$ in $3,086$ iterations.

The results are shown in Figures 4.24 and 4.25, and compare quite well with results presented in [DLP93] and [DA10]. In those sources, the reflected shock hits the outlet boundary at $y = 0.11$ and $y = 0.10$ respectively, compared to $y = 0.123$ in the results presented here.

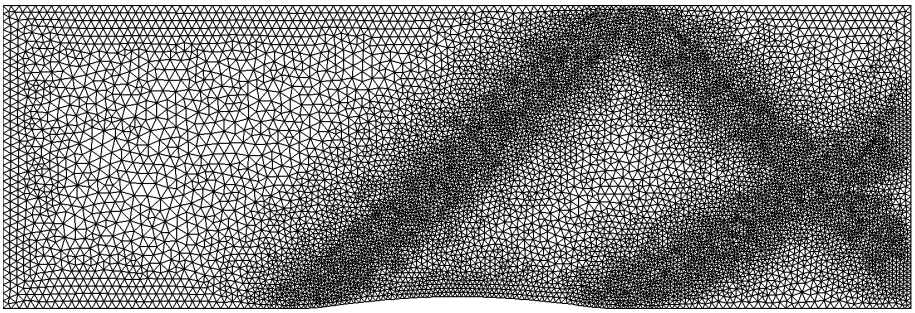


Figure 4.23: The grid used for the channel with 4% bump and $M_{\text{inlet}} = 1.65$.

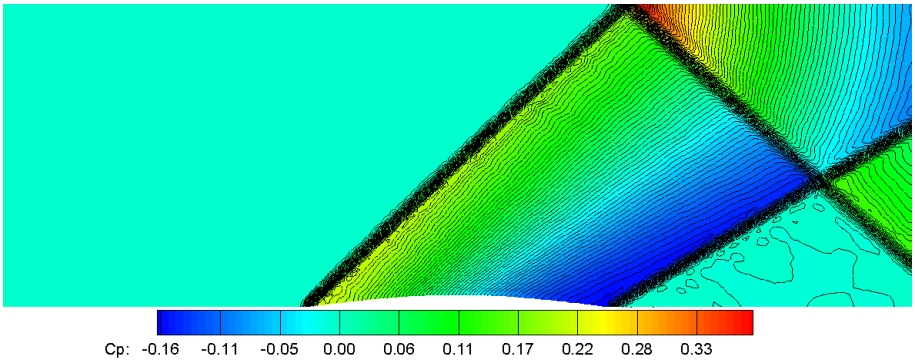



Figure 4.24: Coefficient of pressure contours for the channel with 4% bump and $M_{\text{inlet}} = 1.65$.

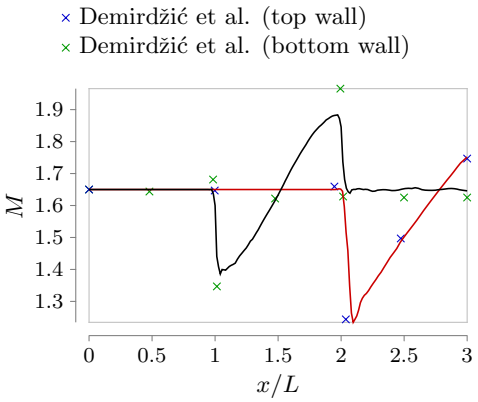



Figure 4.25: Mach number plots along the bottom wall (solid black line) and top wall (dashed red line) for the channel with 4% bump and $M_{\text{inlet}} = 1.65$. Some data points reported in [DLP93] are shown for comparison with the present solver.

### 4.8.4 Engine Inlet

The final test case is the engine inlet from the GAMM workshop [Der+89]. It was selected due to the significantly increased complexity in both the geometry and the resulting fluid dynamics and shock structure. There is a lambda shock at the entrance, consisting of a strong bow-type shock and an attached weak shock. The lengths reported are nondimensionalised by the height of the inlet opening, which is 5.901 units. However, it should be noted that the contributions to the workshop use different non-dimensionalisations.

Farfield boundary conditions are imposed at the farfield ($M_\infty = 2$), slip wall boundary conditions are imposed at the top wall and hub of the inlet, pressure outlet with specified exit Mach number boundary conditions are imposed at the end of the engine inlet ($M_{\text{exit}} = 0.27$), and symmetry plane boundary conditions are imposed on the symmetry plane. Due to the relative complexity of the geometry and the resulting flow field, a signficantly finer grid was used compared to the previous test cases. The target meshwidth varied according to curvature from $7.2 \times 10^{-2}$ to $9.6 \times 10^{-2}$ inside the inlet and $7.2 \times 10^{-2}$ to 10.2 outside the inlet. At the cusp of the inlet, the target meshwidth was set to $7.2 \times 10^{-2}$. Line sources and a polygonal source were placed by trial and error to improve the resolution of the lambda shock on the bump at the inlet entrace. This resulted in 127 nodes on the top surface of the inlet top wall, 230 on the bottom surface of the inlet top wall, 20 on the outlet inside the engine inlet, and 286 on the symmetry plane. The target growth rate was set to 1.07 inside the inlet and 1.15 outside. This resulted in $38,006$ nodes and $75,081$ cells across the grid as a whole.

The results are shown on the right of Figures 4.26 and in 4.27. Clearly, the expected lambda shock structure is properly and sharply resolved. There was significant disagreement between the workshop participants about the correct location of the shocks, particularly the weak shock. These are shown in Table 4.6, together with the results from the present solver. Despite this disagreement, there is very close agreement between the results using the present solver and those of a relative outlier amongst the contributions, [Mor+89]. It should be emphasised that this agreement is coincidental, the present solver being an entirely unrelated development.

| | $x_s$ | $x_w$ |
|---|---|---|
| C1 [Ang+89] | $-3.7 : -2.9$ | $-0.85 : 1.4$ |
| C2 [BM89] | $-2.7 : -2.4$ | $4.6 : 5.3$ |
| C4 [Cha+89] | $-3.7 : -2.2$ | $0.0 : 1.0$ |
| C15 [ML89] | $-3.7 : -2.9$ | $-0.85 : 1.4$ |
| C16 [Mor+89] | $-1.4$ | $2.0$ |
| C18 [SM89] | $-3.4 : -2.9$ | $-1.4 : 0.17$ |
| Present solver | $-1.02$ | $1.83$ |

Table 4.6: Computed strong and weak shock positions ($x_s$ and $x_w$ respectively) along the bottom wall of the engine inlet compared to those provided in contributions to the GAMM workshop [Der+89]. Note the values for the workshop contributors were obtained by manual digitisation from the published results.

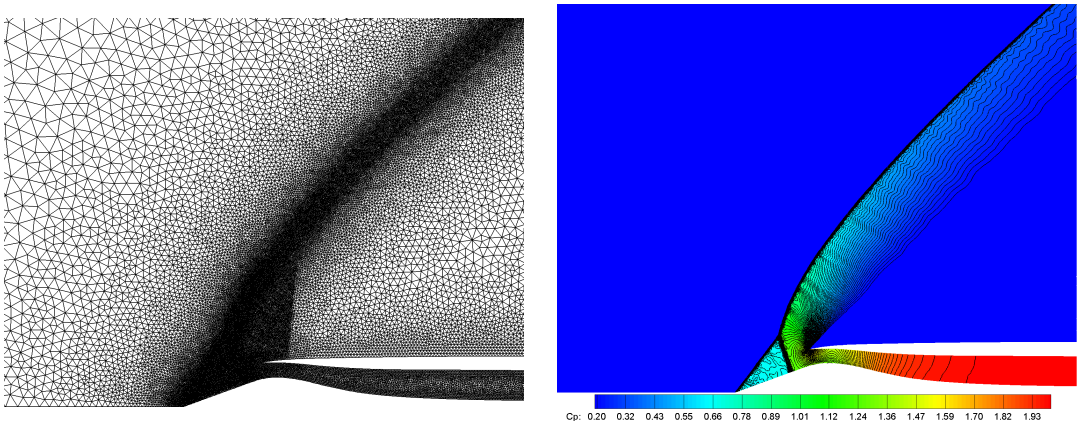



Figure 4.26: Left: a close-up of the grid used for the engine inlet. Right: the resulting coefficient of pressure contours.

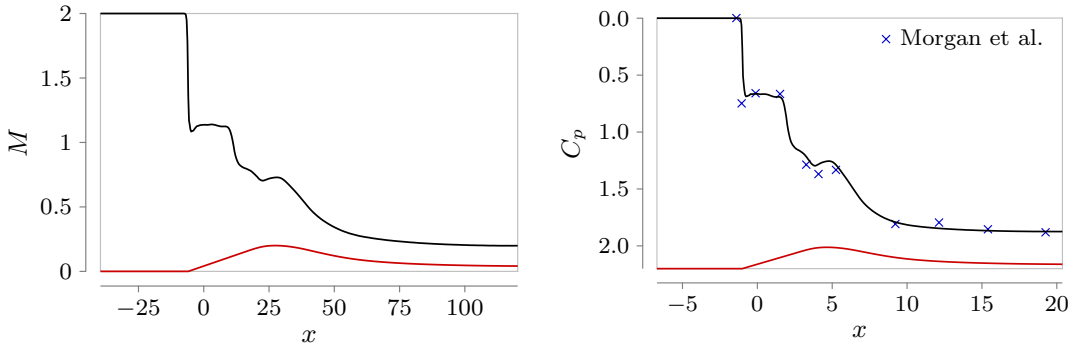



Figure 4.27: Mach number and coefficient of pressure plots (black lines) along the symmetry plane and bottom wall of the engine inlet (geometry shown as red lines), compared to experimental results from [Mor+89].

# Chapter 5

# Development of the Viscous Solver

## 5.1 Introduction

The extension of the solver to viscous flows consists largely of incorporating additional surface fluxes into the momentum and energy equations, with the continuity equation remaining the same as for the inviscid case. The additional surface fluxes model two new phenomena

1. Momentum diffusion due to shear stresses. This adds one additional term to the momentum equation and one additional term to the energy equation.

2. Thermal diffusion due to temperature gradients. This adds one additional term to the energy equation.

The addition of the new surface fluxes requires several aspects of the solver to be modified. In particular, the calculation of the optimum time step must be revised and new boundary conditions must be designed and implemented.

## 5.2 Non-Dimensionalisation

As discussed briefly in Chapter 2, the viscous stresses in the flow are described by the viscous stress tensor

$$\tilde{\boldsymbol{\tau}} = \begin{bmatrix} \tilde{\sigma}_x & \tilde{\tau}_{xy} \\ \tilde{\tau}_{yx} & \tilde{\sigma}_y \end{bmatrix} \tag{5.1}$$

where, as before, the tildes indicate dimensional variables. The fluid is assumed to be Newtonian, and therefore explicit formulae for the components of the viscous stress tensor are (see Section 2.1.6)

$$\begin{aligned} \tilde{\sigma}_x &= 2\tilde{\mu}\left(\frac{\partial \tilde{u}}{\partial \tilde{x}} - \frac{1}{3}\tilde{\nabla}\cdot\tilde{\boldsymbol{u}}\right) \\ \tilde{\tau}_{xy} = \tilde{\tau}_{yx} &= \tilde{\mu}\left(\frac{\partial \tilde{u}}{\partial \tilde{y}} + \frac{\partial \tilde{v}}{\partial \tilde{x}}\right) \\ \tilde{\sigma}_y &= 2\tilde{\mu}\left(\frac{\partial \tilde{v}}{\partial \tilde{y}} - \frac{1}{3}\tilde{\nabla}\cdot\tilde{\boldsymbol{u}}\right) \end{aligned} \tag{5.2}$$

Non-dimensionalising the viscous stresses requires the introduction of a new dimensionless variable, the *dimensionless dynamic viscosity*

$$\mu \coloneqq \frac{\tilde{\mu}}{\mu_r} \tag{5.3}$$

and the definition of a new dimensionless group, the *Reynolds number*

$$R_e \coloneqq \frac{\rho_r u_r \ell_r}{\mu_r} \tag{5.4}$$

Physically, the Reynolds number represents the ratio of inertial to viscous forces [And10, p. 38] and is ubiquitous in the physics of viscous flows.

Following the non-dimensionalisation procedure of Section 3.3 then yields the non-dimensionalised viscous stresses

$$\begin{aligned} \sigma_x &= \frac{2\mu}{R_e}\left(\frac{\partial u}{\partial x} - \frac{1}{3}\nabla\cdot\boldsymbol{u}\right) \\ \tau_{xy} = \tau_{yx} &= \frac{\mu}{R_e}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \\ \sigma_y &= \frac{2\mu}{R_e}\left(\frac{\partial v}{\partial y} - \frac{1}{3}\nabla\cdot\boldsymbol{u}\right) \end{aligned} \tag{5.5}$$

The inclusion of the factor $1/R_e$ in the definition of the dimensionless viscous stresses is not necessary, but slightly simplifies the resulting governing equations. For convenience, the projection of the viscous stress tensor in the normal direction is written

$$\boldsymbol{\eta} \coloneqq \boldsymbol{\tau}\cdot\boldsymbol{n} = \left\{ \begin{array}{c} \sigma_x n_x + \tau_{xy} n_y \\ \tau_{xy} n_x + \sigma_y n_y \end{array} \right\} \tag{5.6}$$

where $n_x \coloneqq \boldsymbol{n}\cdot\boldsymbol{i}$ and $n_y \coloneqq \boldsymbol{n}\cdot\boldsymbol{j}$ are the components of the unit normal in the $x$ and

$y$ directions respectively.

The non-dimensionalised viscous momentum equation (2.10) is then

$$\frac{\partial}{\partial t}\int_{\Omega}\rho\boldsymbol{u}d\Omega+\oint_{\partial\Omega}\rho\boldsymbol{u}\left(\boldsymbol{u}\cdot\boldsymbol{n}\right)dS=-\oint_{\partial\Omega}p\boldsymbol{n}dS+\oint_{\partial\Omega}\boldsymbol{\eta}dS \tag{5.7}$$

which is simply the non-dimensionalised inviscid momentum equation (3.8) with an additional term representing the viscous momentum flux.

It is important to keep in mind that whilst the viscous momentum equation looks invariant under the non-dimensionalisation, it is not. The appearance of the Reynolds number is simply hidden inside the definition of the dimensionless viscous stress tensor.

The dimensionless viscous energy equation requires the definition of another dimensionless variable, the *dimensionless thermal conductivity*

$$k\coloneqq\frac{\tilde{k}}{k_r} \tag{5.8}$$

and the introduction of another dimensionless group, the *Prandtl number*

$$P\coloneqq\frac{\mu c_p}{k} \tag{5.9}$$

The Prandtl number represents the ratio of momentum diffusion to thermal diffusion. If the temperature range is not too extreme, and unlike the Reynolds number, the Prandtl number can be assumed to be a property of the fluid rather than of the flow. For air in the ideal gas region but below about 600 °C, it is generally sufficiently accurate to assume the constant value $P=P_r=0.71$ [And10, p. 818].

The dimensionless viscous energy equation is found to be

$$\begin{aligned}&\frac{\partial}{\partial t}\int_{\Omega}\left\{M_r^2p+\frac{\gamma-1}{2}M_r^2\rho\left(\boldsymbol{u}\cdot\boldsymbol{u}\right)\right\}d\Omega\\&\quad+\oint_{\partial\Omega}\left\{\left(\boldsymbol{u}\cdot\boldsymbol{n}\right)\left[1+\gamma M_r^2p+\frac{\gamma-1}{2}M_r^2\rho\left(\boldsymbol{u}\cdot\boldsymbol{u}\right)\right]\right\}dS\\&\quad=\left(\gamma-1\right)M_r^2\oint_{\partial\Omega}\left(\boldsymbol{\eta}\cdot\boldsymbol{u}\right)ds-\frac{1}{R_eP_r}\oint_{\partial\Omega}k\left(\nabla h\cdot\boldsymbol{n}\right)dS\end{aligned} \tag{5.10}$$

As with the viscous momentum equation, the viscous energy equaton can be obtained from the inviscid energy equation by adding additional terms; for the viscous energy equation two must be added. These terms represent the work done on the control volume by the viscous stresses and the net heat flux out of the control volume due to the temperature gradient.

For the test cases where constant dynamic viscosity and thermal conductivity are not assumed, they are obtained from the enthalpy using Sutherland's formula (2.39) and (2.40). In dimensionless form, these are given by

$$\begin{aligned} \mu &= \mu_0 \left(\frac{h}{h_0}\right)^{\frac{3}{2}} \frac{h_0 + h_{s,\mu}}{h + h_{s,\mu}} \\ k &= k_0 \left(\frac{h}{h_0}\right)^{\frac{3}{2}} \frac{h_0 + h_{s,k}}{h + h_{s,k}} \end{aligned} \tag{5.11}$$

where $\mu_0$, $k_0$, $h_0$, $h_{s,\mu}$, and $h_{s,k}$ are dimensionless forms of the gas-specific constants described in Section 2.1.7. As can be seen by comparing these relations to (2.39) and (2.40), Sutherland's law is invariant under the non-dimensionalisation (other than the replacement of the model constants).

## 5.3 Discretisation of the Governing Equations of Viscous Flow

### 5.3.1 The Viscous Normal Momentum Equation

The viscous normal momentum equation for an arbitrary edge $i$ is obtained by taking the inner product of the viscous momentum equation (5.7) with the edge unit normal $\boldsymbol{n}_i$

$$\frac{\partial}{\partial t}\int_\Omega (\boldsymbol{m}\cdot\boldsymbol{n}_i)\,d\Omega + \oint_{\partial\Omega} (\boldsymbol{m}\cdot\boldsymbol{n}_i)\,(\boldsymbol{u}\cdot\boldsymbol{n})\,dS = -\oint_{\partial\Omega} p\,(\boldsymbol{n}\cdot\boldsymbol{n}_i)\,dS + \oint_{\partial\Omega} (\boldsymbol{\eta}\cdot\boldsymbol{n}_i)\,dS \tag{5.12}$$

The discretisation of all but the final term has already been discussed in Section 3.6.2, and results in

$$\begin{aligned} \frac{\Omega_i\left[(\boldsymbol{m}\cdot\boldsymbol{n})^* - (\boldsymbol{m}\cdot\boldsymbol{n})^n\right]_{c(i)}}{\Delta t_i} + \sum_{e(i)} \left\{\bar{l}_e\,(\boldsymbol{u}\cdot\boldsymbol{n})'_e\,(\boldsymbol{m}_e\cdot\boldsymbol{n}_i)^*\right\} &= \frac{\Omega_i\,(p_\ell - p_r)^n}{\|\boldsymbol{x}_\ell - \boldsymbol{x}r\|_2} \\ &\quad + \oint_{\partial\Omega} (\boldsymbol{\eta}\cdot\boldsymbol{n}_i)\,dS \end{aligned} \tag{5.13}$$

All that remains is the discretisation of the viscous term

$$\oint_{\partial\Omega} (\boldsymbol{\eta}\cdot\boldsymbol{n}_i)\,dS \tag{5.14}$$

The surface integral is handled in the same way as for the convection term, and the viscous term is treated implicitly (i.e. the normal projection of the viscous stress tensor is approximated at the prediction time level)

$$\oint_{\partial\Omega} (\boldsymbol{\eta} \cdot \boldsymbol{n}_i)\, dS \approx \sum_{e(i)} \left\{ \bar{l}_e \left( \boldsymbol{\eta}_e^* \cdot \boldsymbol{n}_i \right) \right\} \tag{5.15}$$

The approximation of the edge centre normal projection of the viscous stress tensor $\boldsymbol{\eta}_e$ will be discussed in Section 5.4. The result will be an approximation of the form

$$\boldsymbol{\eta}_e^* \approx \mu_e \sum_{j(e)} \left\{ \boldsymbol{\zeta}_j u_j^* \right\} + \mu_e \boldsymbol{\pi}_e \bar{d}_e^* \tag{5.16}$$

where $\mu_e$ is the dynamic viscosity at the centre of edge $e$, $j(e)$ ranges over the edge flow velocity component specifications included in the reconstruction stencil, $u_j$ is the corresponding flow velocity component, $\boldsymbol{\zeta}_j, \boldsymbol{\pi}_e \in \mathbb{R}^2$ are reconstruction coefficient vectors, and

$$\bar{d}_e^* \coloneqq \frac{1}{\Omega_e} \sum_{d(e)} \left\{ \bar{l}_d \left( \boldsymbol{u} \cdot \boldsymbol{n} \right)_d^* \right\} \tag{5.17}$$

is an approximation to the flow velocity divergence at the centre of edge $e$ using an appropriate control volume $\Omega_e$. The selection of $\Omega_e$ and the specifics of the reconstruction are left to Section 5.4.

Since the unknowns are the edge centre normal momenta rather than the edge centre normal flow velocities, the above approximation must be rewritten in terms of the momenta. This is done using $\boldsymbol{u}_e \equiv \frac{1}{\rho_e'} \boldsymbol{m}_e$, where $\rho_e'$ is the same interpolated edge centre density used to compute $(\boldsymbol{u} \cdot \boldsymbol{n})_e'$ for the convection term. The approximation to the normal projection of the viscous stress tensor is therefore rewritten

$$\boldsymbol{\eta}_e^* \approx \mu_e \sum_{j(e)} \left\{ \frac{m_j^*}{\rho_j'} \boldsymbol{\zeta}_j \right\} + \mu_e \boldsymbol{\pi}_e \bar{d}_{e,m}^* \tag{5.18}$$

with

$$\bar{d}_{e,m}^* \coloneqq \frac{1}{\Omega_e} \sum_{d(e)} \left\{ \frac{\bar{l}_d}{\rho_d'} \left( \boldsymbol{m} \cdot \boldsymbol{n} \right)_d^* \right\} \tag{5.19}$$

Inserting the approximation to $\boldsymbol{\eta}_e$ into (5.14) yields the discretised viscous term

$$\sum_{e(i)} \left\{ \bar{l}_e \mu_e \left[ \sum_{j(e)} \left( \frac{m_j^*}{\rho_j'} \boldsymbol{\zeta}_j \cdot \boldsymbol{n}_i \right) + \frac{(\boldsymbol{\pi}_e \cdot \boldsymbol{n}_i)}{\Omega_e} \sum_{d(e)} \left( \frac{\bar{l}_d}{\rho_d'} (\boldsymbol{m} \cdot \boldsymbol{n})_d^* \right) \right] \right\} \tag{5.20}$$

Hence the discretised viscous normal momentum equation is

$$\begin{aligned} &\frac{\Omega_i \left[ (\boldsymbol{m} \cdot \boldsymbol{n})^* - (\boldsymbol{m} \cdot \boldsymbol{n})^n \right]_{c(i)}}{\Delta t_i} + \sum_{e(i)} \left\{ \bar{l}_e \left( \boldsymbol{u} \cdot \boldsymbol{n} \right)_e' \left( \boldsymbol{m}_e \cdot \boldsymbol{n}_i \right)^* \right\} = \frac{\Omega_i \left( p_\ell - p_r \right)^n}{\left\| \boldsymbol{x}_\ell - \boldsymbol{x} r \right\|_2} \\ &\quad + \sum_{e(i)} \left\{ \bar{l}_e \mu_e^n \left[ \sum_{j(e)} \left( \frac{m_j^*}{\rho_j'} \boldsymbol{\zeta}_j \cdot \boldsymbol{n}_i \right) + \frac{(\boldsymbol{\pi}_e \cdot \boldsymbol{n}_i)}{\Omega_e} \sum_{d(e)} \left( \frac{\bar{l}_d}{\rho_d'} (\boldsymbol{m} \cdot \boldsymbol{n})_d^* \right) \right] \right\} \end{aligned} \tag{5.21}$$

where the kinematic viscosity $\mu_e$ is treated explicitly.

### 5.3.2 The Viscous Pressure Correction Equation

The viscous pressure correction equation is obtained by discretising the non-dimensionalised viscous energy equation (5.10). The inviscid terms are discretised in the same way as for the inviscid pressure equation (3.35), yielding

$$\begin{aligned} &\frac{\Omega_i M_r^2}{\Delta t_i} \left[ \delta p^{n+1} + \frac{\gamma - 1}{2} \left( \frac{\boldsymbol{m}^* \cdot \boldsymbol{m}^*}{\rho^{n+1}} - \frac{\boldsymbol{m} \cdot \boldsymbol{m}}{\rho} \Big|^n \right) \right]_{c(i)} \\ &+ \Omega_i M_r^2 \left( 1 - \gamma \right) \boldsymbol{u}_{c(i)}^* \cdot \nabla \delta p_{c(i)}^{n+1} \\ &+ \sum_{e(i)} \left\{ \bar{l}_e \left( \boldsymbol{u} \cdot \boldsymbol{n} \right)_e' \left[ 1 + \gamma M_r^2 \left( p^n + \delta p^{n+1} \right) + \frac{\gamma - 1}{2} M_r^2 \cdot \frac{\boldsymbol{m}^* \cdot \boldsymbol{m}^*}{\rho^{n+1}} \right]_e \right\} \\ &+ \left( 1 - \gamma \right) M_r^2 \sum_{e(i)} \left\{ \bar{l}_e \left( \boldsymbol{u} \cdot \boldsymbol{n} \right)_e' \left( \frac{\Delta t}{\rho^{n+1}} \boldsymbol{u}^* \cdot \nabla \delta p^{n+1} \right)_e \right\} \\ &+ \sum_{e(i)} \left\{ \bar{l}_e \left( \frac{\Delta t_e}{\rho_e^{n+1}} \cdot \frac{\left( \delta p_{c_\ell} - \delta p_{c_r} \right)^{n+1}}{\left\| \boldsymbol{x}_{c_\ell} - \boldsymbol{x}_{c_r} \right\|_2} \right) \left[ 1 + \gamma M_r^2 p^n + \frac{\gamma - 1}{2} M_r^2 \cdot \frac{\boldsymbol{m}^* \cdot \boldsymbol{m}^*}{\rho^{n+1}} \right]_e \right\} \\ &= \left( \gamma - 1 \right) M_r^2 \oint_{\partial \Omega} \left( \boldsymbol{\eta} \cdot \boldsymbol{u} \right) dS - \frac{1}{R_e P_r} \oint_{\partial \Omega} k \left( \nabla h \cdot \boldsymbol{n} \right) dS \end{aligned} \tag{5.22}$$

Completing the discretisation requires the discretisation of the viscous stress term

$$\left( \gamma - 1 \right) M_r^2 \oint_{\partial \Omega} \left( \boldsymbol{\eta} \cdot \boldsymbol{u} \right) dS \tag{5.23}$$

and the heat flux term

$$-\frac{1}{R_e P_r}\oint_{\partial\Omega} k\left(\nabla h\cdot\boldsymbol{n}\right)dS \tag{5.24}$$

**The Viscous Stress Term**

The discretisation of the surface integral follows the usual prescription, resulting in

$$\left(\gamma-1\right)M_r^2\oint_{\partial\Omega}\left(\boldsymbol{\eta}\cdot\boldsymbol{u}\right)dS\approx\left(\gamma-1\right)M_r^2\sum_{e(i)}\left\{\bar{l}_e\left(\boldsymbol{\eta}\cdot\boldsymbol{u}\right)_e\right\} \tag{5.25}$$

As with the other terms, the product $\left(\boldsymbol{\eta}\cdot\boldsymbol{u}\right)_e$ is evaluated at the new time level. To accomplish this, $\boldsymbol{\eta}_e^{n+1}$ and $\boldsymbol{u}_e^{n+1}$ are each expanded as the sum of their values at the prediction time level plus a correction term

$$\begin{aligned}\left(\boldsymbol{\eta}\cdot\boldsymbol{u}\right)_e^{n+1}&=\left(\boldsymbol{\eta}_e^{*}+\delta\boldsymbol{\eta}_e^{n+1}\right)\left(\boldsymbol{u}_e^{*}+\delta\boldsymbol{u}_e^{n+1}\right)\\&=\boldsymbol{\eta}_e^{*}\cdot\boldsymbol{u}_e^{*}+\boldsymbol{u}_e^{*}\cdot\delta\boldsymbol{\eta}_e^{n+1}+\boldsymbol{\eta}_e^{*}\cdot\delta\boldsymbol{u}_e^{n+1}+\delta\boldsymbol{\eta}_e^{n+1}\cdot\delta\boldsymbol{u}_e^{n+1}\end{aligned} \tag{5.26}$$

It will be shown that the corrections $\delta\boldsymbol{\eta}_e^{n+1}$ and $\delta\boldsymbol{u}_e^{n+1}$ are linear combinations of the pressure corrections in the stencil. As a result, the term $\delta\boldsymbol{\eta}_e^{n+1}\cdot\delta\boldsymbol{u}_e^{n+1}$ is quadratic in the pressure corrections. There are two suitable methods for handling this non-linearity, which are discussed in Section A.4.1 of Appendix A. The selected method is to linearise by simply neglecting the terms of higher than first order in the pressure correction. With this assumption, (5.26) becomes

$$\left(\boldsymbol{\eta}\cdot\boldsymbol{u}\right)_e^{n+1}\approx\boldsymbol{\eta}_e^{*}\cdot\boldsymbol{u}_e^{*}+\boldsymbol{u}_e^{*}\cdot\delta\boldsymbol{\eta}_e^{n+1}+\boldsymbol{\eta}_e^{*}\cdot\delta\boldsymbol{u}_e^{n+1} \tag{5.27}$$

The flow velocity correction is calculated from the momentum correction (3.33)

$$\delta\boldsymbol{u}_e^{n+1}=-\frac{\Delta t_e}{\rho_e'}\nabla\delta p\big|_e^{n+1} \tag{5.28}$$

The normal flow velocity correction is obtained using the usual two-point normal scalar gradient approximation

$$\delta\left(\boldsymbol{u}\cdot\boldsymbol{n}\right)_e^{n+1}=-\frac{\Delta t_e}{\rho_e'}\nabla\delta p\big|_e^{n+1}\cdot\boldsymbol{n}_e\approx\frac{\Delta t_e\left(\delta p_\ell-\delta p_r\right)^{n+1}}{\rho_e'\left\|\boldsymbol{x}_\ell-\boldsymbol{x}_r\right\|_2} \tag{5.29}$$

The correction to the normal projection of the viscous stress tensor $\delta\boldsymbol{\eta}_e^{n+1}$ is eval-

uated by inserting the flow velocity correction into (5.16)

$$\begin{aligned}\delta\boldsymbol{\eta}_e^{n+1} \approx \mu_e \sum_{j(e')} \left\{ \boldsymbol{\zeta}_j \frac{\Delta t_j \left(\delta p_{j,\ell} - \delta p_{j,r}\right)^{n+1}}{\rho'_j \left\|\boldsymbol{x}_{j,\ell} - \boldsymbol{x}_{j,r}\right\|_2} \right\} \\ + \frac{\mu_e \boldsymbol{\pi}_e}{\Omega_e} \sum_{d(e)} \left\{ \bar{l}_d \frac{\Delta t_d \left(\delta p_{d,\ell} - \delta p_{d,r}\right)^{n+1}}{\rho'_d \left\|\boldsymbol{x}_{d,\ell} - \boldsymbol{x}_{d,r}\right\|_2} \right\}\end{aligned} \tag{5.30}$$

where the subscripts $j,\ell$ and $j,r$ refer to the cells on the left and right side of edge $j$ respectively, and the range $j(e')$ on the first sum includes only the normal flow velocity components in the stencil. Any other components that are specified in the stencil, such as inlet edge tangential flow velocities, are fixed by the boundary conditions and hence do not contribute to the correction. Note that this expression is much simpler (and yields a much smaller stencil for the viscous pressure correction equation) than would be the case had the orthogonality condition discussed in Section 2.2 not been imposed, since the two-point correction (5.29) would need to be replaced by a reconstruction also involving neighbouring cells.

Substituting the above results into (5.25) yields the discretised viscous stress term

$$\begin{aligned}&(\gamma-1)\, M_r^2 \sum_{e(i)} \left\{ \bar{l}_e \left(\boldsymbol{\eta}\cdot\boldsymbol{u}\right)_e^* \right\} \\ &+ (\gamma-1)\, M_r^2 \sum_{e(i)} \left\{ \bar{l}_e \mu_e \sum_{j(e')} \left[ \left(\boldsymbol{\zeta}_j \cdot \boldsymbol{u}_e^*\right) \frac{\Delta t_j \left(\delta p_{j,\ell} - \delta p_{j,r}\right)^{n+1}}{\rho'_j \left\|\boldsymbol{x}_{j,\ell} - \boldsymbol{x}_{j,r}\right\|_2} \right] \right\} \\ &+ (\gamma-1)\, M_r^2 \sum_{e(i)} \left\{ \frac{\bar{l}_e \mu_e \left(\boldsymbol{\pi}_e \cdot \boldsymbol{u}_e^*\right)}{\Omega_e} \sum_{d(e)} \left[ \bar{l}_d \frac{\Delta t_d \left(\delta p_{d,\ell} - \delta p_{d,r}\right)^{n+1}}{\rho'_d \left\|\boldsymbol{x}_{d,\ell} - \boldsymbol{x}_{d,r}\right\|_2} \right] \right\} \\ &+ (\gamma-1)\, M_r^2 \sum_{e(i)} \left\{ -\bar{l}_e \frac{\Delta t_e}{\rho'_e} \boldsymbol{\eta}_e^* \cdot \nabla \delta p \Big|_e^{n+1} \right\}\end{aligned} \tag{5.31}$$

**The Heat Flux Terms**

Discretising the surface integral as previously results in

$$-\frac{1}{R_e P_r} \oint_{\partial\Omega} k \left(\nabla h \cdot \boldsymbol{n}\right) dS \approx -\frac{1}{R_e P_r} \sum_{e(i)} \left\{ \bar{l}_e k_e \left(\nabla h \cdot \boldsymbol{n}\right)_e \right\} \tag{5.32}$$

Since the pressure correction equation is being treated implicitly, this term is evaluated at the new time level. This is accomplished by using the ideal gas equation

of state (3.12) to rewrite the heat conduction term as

$$-\frac{1}{R_e P_r}\sum_{e(i)}\left\{\bar{l}_e k_e\left[\nabla\left(\frac{1+\gamma M_r^2 p^{n+1}}{\rho^{n+1}}\right)\cdot\boldsymbol{n}\right]_e\right\} \tag{5.33}$$

The pressure is then written as the sum of its value at the old time level plus the pressure correction, $p^{n+1} = p^n + \delta p^{n+1}$. This allows the heat conduction term to be decomposed into the sum of an explicit term and an implicit term

$$\begin{aligned}&-\frac{1}{R_e P_r}\sum_{e(i)}\left\{\bar{l}_e k_e\left[\nabla\left(\frac{1+\gamma M_r^2 p^{n}}{\rho^{n+1}}\right)\cdot\boldsymbol{n}\right]_e\right\}\\ &-\frac{\gamma M_r^2}{R_e P_r}\sum_{e(i)}\left\{\bar{l}_e k_e\left[\nabla\left(\frac{\delta p^{n+1}}{\rho^{n+1}}\right)\cdot\boldsymbol{n}\right]_e\right\}\end{aligned} \tag{5.34}$$

Note that as the steady state is approached $\delta p \to 0$, so the implicit term tends to zero and the heat conduction term is represented by the explicit term only.

The obvious way to evaluate the normal gradients in the heat transfer terms is by using the same two-point approximation that has been used throughout for the normal momentum correction and flow velocity correction terms. This yields the discretised heat conduction terms

$$\begin{aligned}&+\frac{\gamma M_r^2}{R_e P_r}\sum_{e(i)}\left\{\frac{\bar{l}_e k_e}{\|\boldsymbol{x}_{e,\ell}-\boldsymbol{x}_{e,r}\|_2}\left[\left(\frac{\delta p}{\rho}\right)_{e,\ell}^{n+1}-\left(\frac{\delta p}{\rho}\right)_{e,r}^{n+1}\right]\right\}\\ &\frac{1}{R_e P_r}\sum_{e(i)}\left\{\frac{\bar{l}_e k_e}{\|\boldsymbol{x}_{e,\ell}-\boldsymbol{x}_{e,r}\|_2}\left[\left(\frac{1+\gamma M_r^2 p^{n}}{\rho^{n+1}}\right)_{e,\ell}-\left(\frac{1+\gamma M_r^2 p^{n}}{\rho^{n+1}}\right)_{e,r}\right]\right\}\end{aligned} \tag{5.35}$$

However, this discretisation can lead to convergence problems when a Dirichlet specification of the boundary enthalpy is used, especially on coarse grids or those with sharp boundary corners. In these cases, an alternative approximation of the heat fluxes is found to be somewhat superior, and is discussed in Section 5.5.

### The Complete Discretised Viscous Pressure Correction Equation

Collecting the above results yields the complete discretised viscous pressure correction equation

$$\frac{\Omega_i M_r^2}{\Delta t_i}\left[\delta p^{n+1}+\frac{\gamma-1}{2}\left(\frac{\boldsymbol{m}^*\cdot\boldsymbol{m}^*}{\rho^{n+1}}-\frac{\boldsymbol{m}\cdot\boldsymbol{m}}{\rho}\bigg|^{n}\right)\right]_{c(i)}$$

$$
\begin{aligned}
&+ \Omega_i M_r^2 (1-\gamma)\, \boldsymbol{u}^*_{c(i)} \cdot \nabla \delta p^{n+1}_{c(i)} \\
&+ \sum_{e(i)} \left\{ \bar{l}_e \left( \boldsymbol{u} \cdot \boldsymbol{n} \right)'_e \left[ 1 + \gamma M_r^2 \left( p^n + \delta p^{n+1} \right) + \frac{\gamma - 1}{2} M_r^2 \cdot \frac{\boldsymbol{m}^* \cdot \boldsymbol{m}^*}{\rho^{n+1}} \right]_e \right\} \\
&+ (1-\gamma) M_r^2 \sum_{e(i)} \left\{ \bar{l}_e \left( \boldsymbol{u} \cdot \boldsymbol{n} \right)'_e \left( \frac{\Delta t}{\rho^{n+1}} \boldsymbol{u}^* \cdot \nabla \delta p^{n+1} \right)_e \right\} \\
&+ \sum_{e(i)} \left\{ \bar{l}_e \left( \frac{\Delta t_e}{\rho_e^{n+1}} \cdot \frac{\left( \delta p_{c_\ell} - \delta p_{c_r} \right)^{n+1}}{\left\| \boldsymbol{x}_{c_\ell} - \boldsymbol{x}_{c_r} \right\|_2} \right) \left[ 1 + \gamma M_r^2 p^n + \frac{\gamma - 1}{2} M_r^2 \cdot \frac{\boldsymbol{m}^* \cdot \boldsymbol{m}^*}{\rho^{n+1}} \right]_e \right\} \qquad (5.36) \\
&= (\gamma - 1) M_r^2 \sum_{e(i)} \left\{ \bar{l}_e \left( \boldsymbol{\eta} \cdot \boldsymbol{u} \right)^*_e \right\} \\
&+ (\gamma - 1) M_r^2 \sum_{e(i)} \left\{ \bar{l}_e \mu_e \sum_{j(e')} \left[ \left( \boldsymbol{\zeta}_j \cdot \boldsymbol{u}^*_e \right) \frac{\Delta t_j \left( \delta p_{j,\ell} - \delta p_{j,r} \right)^{n+1}}{\rho'_j \left\| \boldsymbol{x}_{j,\ell} - \boldsymbol{x}_{j,r} \right\|_2} \right] \right\} \\
&+ (\gamma - 1) M_r^2 \sum_{e(i)} \left\{ \frac{\bar{l}_e \mu_e \left( \boldsymbol{\pi}_e \cdot \boldsymbol{u}^*_e \right)}{\Omega_e} \sum_{d(e)} \left[ \bar{l}_d \frac{\Delta t_d \left( \delta p_{d,\ell} - \delta p_{d,r} \right)^{n+1}}{\rho'_d \left\| \boldsymbol{x}_{d,\ell} - \boldsymbol{x}_{d,r} \right\|_2} \right] \right\} \\
&+ (\gamma - 1) M_r^2 \sum_{e(i)} \left\{ -\bar{l}_e \frac{\Delta t_e}{\rho'_e} \boldsymbol{\eta}^*_e \cdot \nabla \delta p \Big|_e^{n+1} \right\} \\
&+ \frac{\gamma M_r^2}{R_e P_r} \sum_{e(i)} \left\{ \frac{\bar{l}_e k_e}{\left\| \boldsymbol{x}_{e,\ell} - \boldsymbol{x}_{e,r} \right\|_2} \left[ \left( \frac{\delta p}{\rho} \right)^{n+1}_{e,\ell} - \left( \frac{\delta p}{\rho} \right)^{n+1}_{e,r} \right] \right\} \\
&+ \frac{1}{R_e P_r} \sum_{e(i)} \left\{ \frac{\bar{l}_e k_e}{\left\| \boldsymbol{x}_{e,\ell} - \boldsymbol{x}_{e,r} \right\|_2} \left[ \left( \frac{1 + \gamma M_r^2 p^n}{\rho^{n+1}} \right)_{e,\ell} - \left( \frac{1 + \gamma M_r^2 p^n}{\rho^{n+1}} \right)_{e,r} \right] \right\}
\end{aligned}
$$

If the alternative approximation to the heat transfer fluxes is used instead, the final two terms are replaced by (5.34).

## 5.4 Reconstruction of the Viscous Stresses

### 5.4.1 Reconstruction of the Viscous Stress Tensor at the Nodes

From (5.5), the non-dimensionalised viscous stress tensor at an arbitrary node $i$ is represented by the symmetric matrix

$$
\boldsymbol{\tau}_i \coloneqq \begin{bmatrix} \sigma_x & \tau_{xy} \\ \tau_{xy} & \sigma_y \end{bmatrix}_i = \frac{\mu_i}{R_e} \begin{bmatrix} 2\frac{\partial u}{\partial x}\big|_i - \frac{2}{3}\nabla \cdot \boldsymbol{u}|_i & \frac{\partial u}{\partial y}\big|_i + \frac{\partial v}{\partial x}\big|_i \\ \frac{\partial u}{\partial y}\big|_i + \frac{\partial v}{\partial x}\big|_i & 2\frac{\partial v}{\partial y}\big|_i - \frac{2}{3}\nabla \cdot \boldsymbol{u}|_i \end{bmatrix} \qquad (5.37)
$$

Hence reconstructing the viscous stresses amounts to reconstructing the flow velocity gradients. This can be achieved using the methods presented for reconstructing the momentum gradients in Section 3.9.2, by simply replacing the momentum components by the corresponding flow velocity components and the approximation to the momentum divergence with an approximation to the flow velocity divergence. The resulting approximation is sufficient for a second-order spatially accurate scheme, since the reconstructed viscous stresses are exact when the primary flow variable fields are linear. Reusing the momentum reconstruction in this manner is advantageous for efficency and memory usage, since no new set of reconstruction coefficients would need to be computed for the flow velocity gradient reconstructions.

The expressions for the approximations to the flow velocity gradients are therefore obtained from the expressions for the piecewise linear momentum reconstruction expressions (3.80) by applying the above modifications. The result is

$$\begin{aligned}
\left.\frac{\partial u}{\partial x}\right|_i &\approx \sum_{j(i)} M_{2j} u_j + \frac{\bar{d}_i}{2}(1 - g_2) \\
\left.\frac{\partial v}{\partial x}\right|_i &\approx \sum_{j(i)} M_{4j} u_j - \frac{\bar{d}_i}{2} g_4 \\
\left.\frac{\partial u}{\partial y}\right|_i &\approx \sum_{j(i)} M_{3j} u_j - \frac{\bar{d}_i}{2} g_3 \\
\left.\frac{\partial v}{\partial y}\right|_i &\approx -\sum_{j(i)} M_{2j} u_j + \frac{\bar{d}_i}{2}(1 + g_2)
\end{aligned} \tag{5.38}$$

where $j(i)$ ranges over the momentum (flow velocity in this case) component equations specified in the stencil, $u_j$ is the corresponding flow velocity component, and

$$\bar{d}_i \coloneqq \frac{1}{\Omega_i} \sum_{d(i)} \left\{ \bar{l}_d \left( \boldsymbol{u} \cdot \boldsymbol{n} \right)_d \right\} \tag{5.39}$$

is now an approximation to the flow velocity divergence using a suitable control volume $\Omega_i$, the selection of which is discussed in Section 5.4.3.

Clearly, the expressions in (5.38) consist of sums of weighted sums of the flow velocity components with constant multiples of the approximation to the flow velocity divergence. Since the definition of $\boldsymbol{\tau}_i$ in (5.37) consists of sums of these, the reconstruction of $\boldsymbol{\tau}_i$ will itself consist of a sum of the weighted sum of the flow velocity components with a constant multiple of the approximation to the flow velocity gradient. However,

the coefficients of the terms in the weighted sum and the coefficient of the divergence term will be second-order tensors rather than scalars. The reconstruction of the viscous stress tensor can therefore be written in the form

$$\boldsymbol{\tau}_i \approx \mu_i \left( \sum_{j(i)} \{ \boldsymbol{Z}_j u_j \} + \boldsymbol{P}_i \bar{d}_i \right) \tag{5.40}$$

where $\boldsymbol{Z}_j$ and $\boldsymbol{P}_i$ are approximation coefficient tensors.

All that remains is to handle the terms involving the exact flow velocity divergence $\nabla \cdot \boldsymbol{u}|_i$ in (5.37). This will amount to modifying the definition of the approximation coefficient tensors $\boldsymbol{Z}_j$ and $\boldsymbol{P}$ in (5.40). The two possible methods for achieving this are

1. Expand the flow velocity divergence as

$$\nabla \cdot \boldsymbol{u}|_i = \left.\frac{\partial u}{\partial x}\right|_i + \left.\frac{\partial v}{\partial y}\right|_i \tag{5.41}$$

   and reconstruct as for the other flow velocity gradients

2. Substitute the already-computed flow velocity divergence approximation used for reconstructing the flow velocity gradients

$$\nabla \cdot \boldsymbol{u}|_i \approx \bar{d}_i \tag{5.42}$$

In fact, for the specific piecewise linear flow velocity field reconstruction procedure used here, these choices lead to exactly the same approximation to $\boldsymbol{\tau}_i$. This is a simple consequence of imposing (3.63) (i.e. setting the divergence of the reconstruction polynomial equal to the approximation to the divergence). Inserting (5.38) into (5.37) and comparing to (5.40) yields explicit expressions for the approximation coefficient tensors

$$\begin{aligned} \boldsymbol{Z}_j &= \frac{1}{R_e} \begin{bmatrix} 2M_{2j} & M_{3j} + M_{4j} \\ M_{3j} + M_{4j} & -2M_{2j} \end{bmatrix} \\ \boldsymbol{P} &= \frac{1}{R_e} \begin{bmatrix} \frac{1}{3} - g_2 & -\frac{1}{2}(g_3 + g_4) \\ -\frac{1}{2}(g_3 + g_4) & \frac{1}{3} + g_2 \end{bmatrix} \end{aligned} \tag{5.43}$$

### 5.4.2 Reconstruction of the Normal Projection of the Viscous Stress Tensor at the Edge Centres

In fact, the discretised normal momentum and pressure correction equations do not require the reconstruction of the viscous stress tensor at the nodes, but instead require its normal projection at the edge centres. The edge centre viscous stress tensor must therefore be interpolated from the reconstructed nodal values. Due to the elliptic character of the viscous stress terms in the Navier-Stokes equations, this is best achieved using an average of gradients approach [Bla01, p. 169]. For an arbitrary edge $i$ this amounts to selecting two nodes, say $n_\alpha$ and $n_\beta$, and setting

$$\boldsymbol{\tau}_i = \frac{1}{2}\left(\boldsymbol{\tau}_{n_\alpha} + \boldsymbol{\tau}_{n_\beta}\right) \tag{5.44}$$

The selection of the stencil nodes $n_\alpha$ and $n_\beta$ is obviously important for the accuracy and stability of the scheme, and is discussed in Section A.4.2 of Appendix A. The use of the end nodes of the target edge for this purpose was ultimately selected as it can be done at all edges, including boundary edges, and because the centre of the target edge always lies at the mean position of its end nodes regardless of the local grid quality.

The interpolated viscous stress tensor is

$$\boldsymbol{\tau}_i \approx \frac{\mu_i}{2}\left[\left(\sum_{j(\alpha)}\{\boldsymbol{Z}_j u_j\} + \boldsymbol{P}_\alpha \bar{d}_i\right) + \left(\sum_{j(\beta)}\{\boldsymbol{Z}_j u_j\} + \boldsymbol{P}_\beta \bar{d}_i\right)\right] \tag{5.45}$$

For brevity, this is rewritten

$$\boldsymbol{\tau}_i \approx \mu_i\left(\sum_{j(i)}\left\{\bar{\boldsymbol{Z}}_j u_j\right\} + \bar{\boldsymbol{P}}_i \bar{d}_i\right) \tag{5.46}$$

where $j(i)$ ranges over the elements of $j(\alpha)$ followed by the elements of $j(\beta)$, $\bar{\boldsymbol{Z}}_j \coloneqq \frac{1}{2}\boldsymbol{Z}_j$, and $\bar{\boldsymbol{P}}_i \coloneqq \frac{1}{2}\left(\boldsymbol{P}_\alpha + \boldsymbol{P}_\beta\right)$.

Finally, the normal projection of the viscous stress tensor at the centre of edge $i$ is obtained by taking the inner product of (5.46) with the edge unit normal $\boldsymbol{n}_i$. The result is

$$\boldsymbol{\eta}_i \approx \mu_i\left(\sum_{j(i)}\left\{\boldsymbol{\zeta}_j u_j\right\} + \boldsymbol{\pi}_i \bar{d}_i\right) \tag{5.47}$$

where $\boldsymbol{\zeta}_j \coloneqq \bar{\boldsymbol{Z}}_j \cdot \boldsymbol{n}_i$ and $\boldsymbol{\pi}_i \coloneqq \bar{\boldsymbol{P}}_i \cdot \boldsymbol{n_i}$ are the reconstruction coefficients for $\boldsymbol{\eta}_i$.

Explicit formulae for the reconstruction coefficients can be obtained from (5.43) and the above definitions. The result is

$$\begin{aligned} \boldsymbol{\zeta}_j &= \frac{1}{R_e} \left\{ \begin{array}{c} n_x M_{2j} + \frac{n_y}{2} \left( M_{3j} + M_{4j} \right) \\ \frac{n_x}{2} \left( M_{3j} + M_{4j} \right) - n_y M_{2j} \end{array} \right\} \\ \boldsymbol{\pi}_i &= \frac{1}{2R_e} \left\{ \begin{array}{c} n_x \left( \frac{1}{3} - g_2 \right) - \frac{n_y}{2} \left( g_3 + g_4 \right) \\ -\frac{n_x}{2} \left( g_3 + g_4 \right) + n_y \left( \frac{1}{3} + g_2 \right) \end{array} \right\} \end{aligned} \tag{5.48}$$

### 5.4.3 Approximation of the Flow Velocity Divergence

The reconstruction of the viscous stresses at the centre of an edge $i$ requires the evaluation of an approximation to the flow velocity divergence of the form 5.39, which in turn requires the definition of an appropriate control volume $\Omega_i$.

The normal momentum equation is considered first. The normal momentum equation control volume for an interior arbitrary edge $i$ is shown in Figure A.3. To evaluate the normal momentum equation, the normal projection of the viscous stress tensor must be reconstructed at each of the outer bounding edges $\{a, b, c, d\}$. Each of these reconstructions requires a control volume to be defined for the approximation of the flow velocity divergence. Consider the selection of this control volume for edge $b$. Since central discretisation is being used for the reconstruction of the viscous stresses, there are three possible choices for the control volume used to evaluate the flow velocity divergence. These are shown in Figure 5.1. Of these three options note that the first option is identical to the control volume used for evaluating the normal momentum equation for edge $i$, whilst the second option is the same as the control volume used for approximating the momentum divergence when reconstructing the normal momentum equation convection term at the centre of edge $b$. Both of these options produce very similar results, so either choice is acceptable. The third option, however, leads to a scheme with poor stability. The first option was chosen since it leads to the same flow velocity divergence approximation for all outer bounding edges of the normal momentum equation control volume, and therefore is the cheapest to construct and simplest to implement.

For the normal momentum equation at a boundary edge $i$, the only sensible choice for the control volume for approximating the flow velocity divergence is to use the single interior cell adjacent to edge $i$ as the control volume. As for the interior edge case, this is the same as the normal momentum control volume for edge $i$. Any other choice of control volume would lead to a large, possibly non-convex (in fact, star-shaped) control volume, which is likely to be harmful to stability and accuracy (especially on

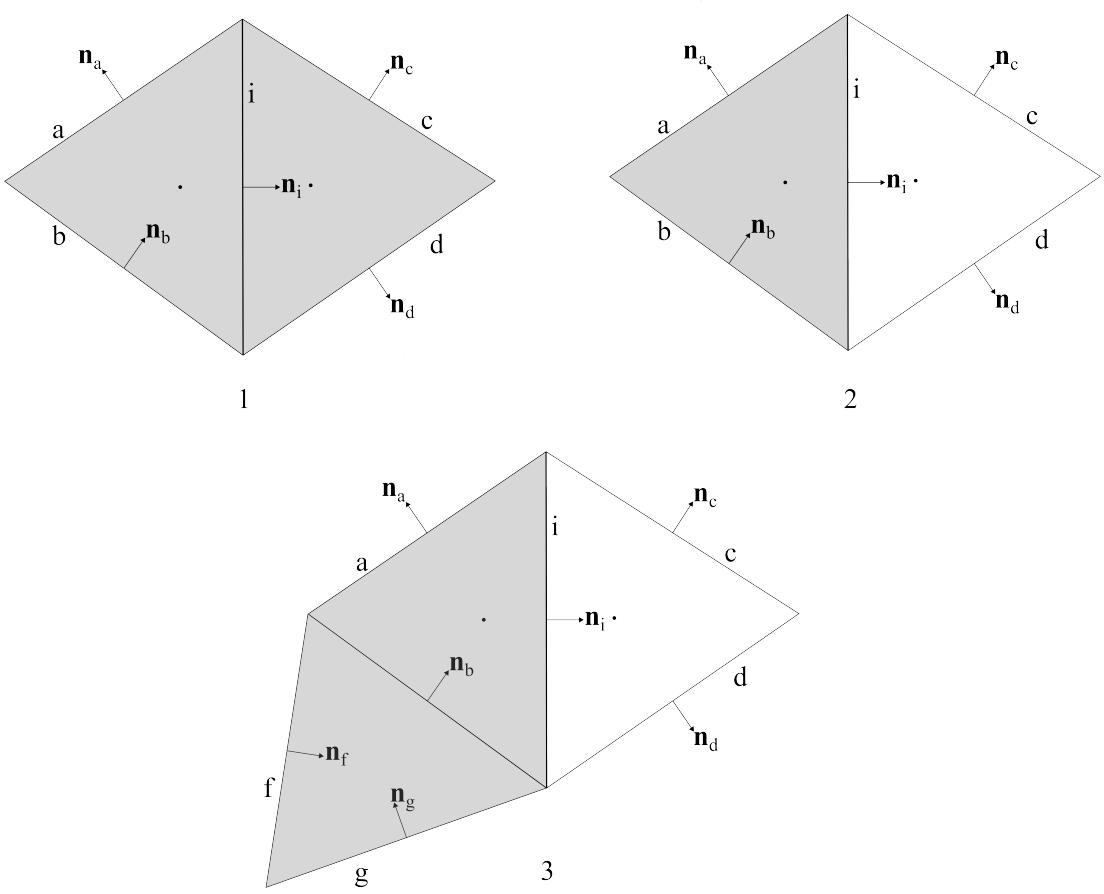



Figure 5.1: The three possible control volume definitions (represented by the shaded areas) for approximating the flow velocity divergence in piecewise linear viscous stress reconstructions at the centre of edge $b$ when solving the normal momentum equation for an interior edge $i$.

poor quality grids).

Finally, for the pressure correction equation at an arbitrary cell $i$, the control volume for approximating the flow velocity divegence is set equal to the pressure correction equation control volume (i.e. consisting only of cell $i$ itself). It may be possible to use the union of cell $i$ together with all adjacent interior cells as the control volume instead, but this would be non-convex for most cells which would again likely be harmful to stability and accuracy.

## 5.5 Approximation of the Heat Fluxes

The evaluation of the heat fluxes requires the approximation of terms of the form

$$\nabla\varphi\big|_i \cdot \boldsymbol{n}_i \tag{5.49}$$

where $i$ is the centre of an arbitrary edge and $\varphi$ is some function of the primary flow variables.

As discussed in Section 5.3.2, the obvious way to approximate these terms is via the same two-point approximation used for the normal pressure and pressure correction gradients

$$\nabla\varphi\big|_i \cdot \boldsymbol{n}_i \approx \frac{\varphi_r - \varphi_\ell}{\|\boldsymbol{x}_r - \boldsymbol{x}_\ell\|_2} \tag{5.50}$$

However it was noted that this approximation does not always perform well in the neighbourhood of fixed-temperature walls, especially when the grid is coarse or poor quality or where the fixed-temperature wall contains sharp corners such as at the trailing edge of an aerofoil. For these cases, an alternative approximation is presented in this section.

The alternative approximation method uses the Green-Gauss method [Bla01, p. 160], which is quite similar to the method used to approximate the flow velocity divergence used in the reconstruction of the viscous stresses (5.39). The Green-Gauss theorem implies

$$\int_{\Omega_i} \nabla\varphi d\Omega = \oint_{\partial\Omega_i} \varphi \boldsymbol{n} dS \tag{5.51}$$

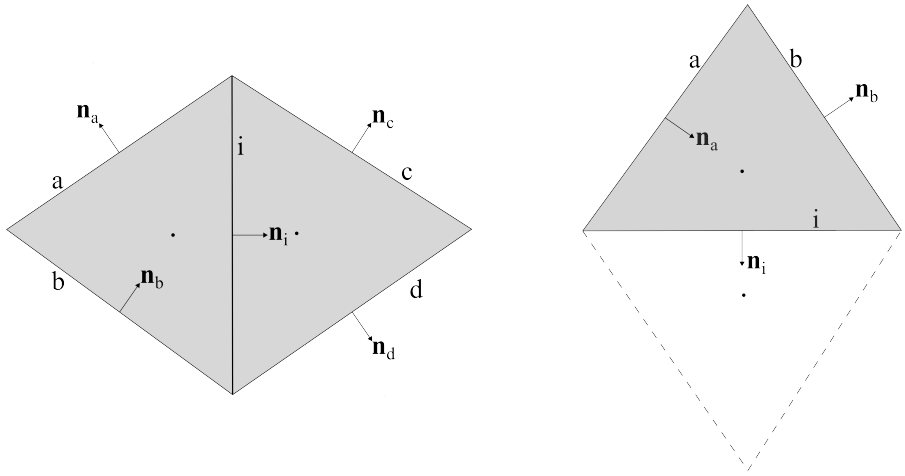



Figure 5.2: The control volume definitions (represented by the shaded areas) for approximating the heat fluxes at an arbitrary edge $i$ when $i$ is an interior edge (left) and a boundary edge (right). The dashed lines represent a dummy cell.

which yields the approximation

$$\begin{aligned} \nabla\varphi\big|_i &\approx \frac{1}{\Omega_i}\oint_{\partial\Omega_i}\varphi\boldsymbol{n}dS \\ &\approx \frac{1}{\Omega_i}\sum_{e(i)}\left\{\bar{l}_e\varphi_e\boldsymbol{n}_e\right\} \end{aligned} \tag{5.52}$$

Here, the edge centre value $\varphi_e$ of the cell circumcentre scalar flow variable $\varphi$ can be simply obtained as the average of the values at the circumcentres of the cells adjacent to edge $e$. Hence the approximation is

$$\nabla\varphi\big|_i \approx \frac{1}{2\Omega_i}\sum_{e(i)}\left\{\bar{l}_e\left(\varphi_{e,\ell}+\varphi_{e,r}\right)\boldsymbol{n}_e\right\} \tag{5.53}$$

All that remains is the selection of the control volume $\Omega_i$ for the arbitrary edge $i$. The obvious choice is to use the same control volume used when solving the normal momentum equation, see Figure 5.2 As can be easily be deduced from the figure, this requires a ten-point stencil for the pressure correction equation at cell $j$ when all edges of $j$ are interior edges, or an eight-point stencil when one edge of $j$ is a boundary edge. This method of approximating the heat fluxes therefore adds no additional cells to the pressure correction stencil, since all cells in the heat flux approximation stencils are already in the viscous stress reconstruction stencils.

This approximation increases the size of the stencil used for evaluating the heat flux at each individual edge, but does not increase the size of the stencil used to solve the pressure correction equation for each cell. It is therefore no more expensive than the original discretisation when solving the pressure correction equation, and is only marginally more expensive when assembling the pressure correction equation. It was therefore used for all numerical examples considered in the remainder of this thesis. Interestingly, using the same approximation for the normal pressure and pressure correction gradients showed no clear benefit even on very poor quality grids, perhaps because these gradients are generally significantly less severe than the gradients involved in the heat flux terms.

Using the approximation discussed in this section, the heat flux terms become

$$\begin{aligned} &- \frac{1}{R_e P_r} \sum_{e(i)} \left\{ \frac{\bar{l}_e k_e}{2\Omega_i} \sum_{f(e)} \left[ \bar{l}_f \left( \upsilon_{f,\ell}^{n+1} + \upsilon_{f,r}^{n+1} \right) (\boldsymbol{n}_f \cdot \boldsymbol{n}_e) \right] \right\} \\ &- \frac{\gamma M_r^2}{R_e P_r} \sum_{e(i)} \left\{ \frac{\bar{l}_e k_e}{2\Omega_i} \sum_{f(e)} \left[ \bar{l}_f \left( \left( \frac{\delta p^{n+1}}{\rho^{n+1}} \right)_{f,\ell} + \left( \frac{\delta p^{n+1}}{\rho^{n+1}} \right) f, r \right) (\boldsymbol{n}_f \cdot \boldsymbol{n}_e) \right] \right\} \end{aligned} \tag{5.54}$$

where

$$\upsilon^{n+1} := \frac{1 + \gamma M_r^2 p^n}{\rho^{n+1}} \tag{5.55}$$

which is equal to the enthalpy at the prediction time level, and is therefore the most up-to-date value of the enthalpy available when assembling the pressure correction equation.

## 5.6 Additional Boundary Conditions for Viscous Flows

### 5.6.1 No-Slip Walls

The treatment of solid walls in viscous flows is significantly more complex than in inviscid flows, due to the appearance of two new phenomena. The first is the generation of shear stresses in the neighbourhood of the wall by friction between the wall and the fluid. The other is the possibility of a heat flux through the wall, due to a non-zero normal temperature gradient at the interface between the wall and the fluid. A very common assumption for solid walls in viscous flows is that the layer of fluid immediately adjacent to the wall is brought completely to rest with respect to the wall, such that there is no relative motion. This is referred to as the *no-slip* condition [And10, p. 65].

Good agreement has been found between this assumption and experimental results, except in the case of rarefied gases with large mean free paths [Whi06, p. 46] where some slip is seen to occur even in the layer of fluid immediately adjacent to the wall. These types of flow are beyond the scope of the current work, hence the no-slip condition will be assumed throughout. Combined with the no-penetration condition, the resulting conditions on the flow velocity and momentum are

$$\boldsymbol{u}_b = \boldsymbol{m}_b = 0 \tag{5.56}$$

The flow velocity in the dummy cell is then obtained by 'reflecting' from the interior state

$$\boldsymbol{u}_{\mathcal{D}} = -\boldsymbol{u}_{\mathcal{I}} \tag{5.57}$$

Note that by 'reflection' it is meant that the component of the flow velocity tangential to the wall is reversed as well as the normal component. As for the situation at slip walls, these are identical to the values obtained by application of linear extrapolation from the wall and interior cell values.

An additional condition is required at no-slip walls. This is usually a Neumann or Dirichlet condition on the temperature or enthalpy. The former is frequently the specification that there is no heat transfer through the wall (i.e. that the wall is adiabatic), whilst the latter is simply the imposition of a fixed temperature or enthalpy at the wall. These two cases are discussed separately. Other specifications are possible, such as the specification of a fixed non-zero heat flux through the wall, but are not considered further in the present work.

#### Adiabatic No-Slip Walls

In this case, the normal temperature gradient at the wall is zero. Using (2.27) and the assumption that the gas is calorically perfect (i.e. that $c_p$ is constant) allows this to be expressed in terms of the enthalpy

$$\nabla h\big|_b \cdot \boldsymbol{n}_b = 0 \tag{5.58}$$

Using the usual two-point approximation to normal gradients of cell-centroidal scalars results in

$$h_{\mathcal{D}} = h_{\mathcal{I}} \tag{5.59}$$

The final condition imposed at adiabatic no-slip walls is that of zero normal pressure

gradient at the wall, as is the case for slip walls. This is again enforced by setting

$$p_{\mathcal{D}} = p_{\mathcal{I}} \tag{5.60}$$

The ideal gas equation of state (3.12) then implies

$$\rho_{\mathcal{D}} = \rho_{\mathcal{I}} \tag{5.61}$$

It then follows from this and (5.57) that the momentum in the dummy cell is

$$\boldsymbol{m}_{\mathcal{D}} = -\boldsymbol{m}_{\mathcal{I}} \tag{5.62}$$

The set of boundary conditions actually imposed on the governing equations is

$$\begin{aligned} \rho_{\mathcal{D}}^{n+1} - \rho_{\mathcal{I}}^{n+1} &= 0 \\ (\boldsymbol{m} \cdot \boldsymbol{n})_b^* &= 0 \\ \delta p_{\mathcal{D}}^{n+1} - \delta p_{\mathcal{I}}^{n+1} &= 0 \end{aligned} \tag{5.63}$$

**No-Slip Walls with Specified Wall Temperature**

This condition can be imposed simply by setting the temperature at the centre of the boundary edge. However, since the gas was assumed to be calorically perfect it can instead be imposed by specifying the enthalpy at the wall $\mathring{h}_w$

$$h_b = \mathring{h}_w \tag{5.64}$$

Linear extrapolation from the specified wall enthalpy and the interior cell centre yields the enthalpy in the dummy cell

$$h_{\mathcal{D}} = 2\mathring{h}_w - h_{\mathcal{I}} \tag{5.65}$$

Zero normal pressure gradient at the wall is again imposed by setting

$$p_{\mathcal{D}} = p_{\mathcal{I}} \tag{5.66}$$

The ideal gas equation of state (3.12) is then used to calculate the density in the dummy cell

$$\rho_{\mathcal{D}} = \frac{1 + \gamma M_r^2 p_{\mathcal{D}}}{2\mathring{h}_w - h_{\mathcal{I}}} \tag{5.67}$$

A relation between the interior cell and dummy cell densities can be obtained by applying $p_{\mathcal{D}} = p_{\mathcal{I}}$ and the ideal gas equation of state

$$\rho_{\mathcal{D}} \left( 2\mathring{h}_w - h_{\mathcal{I}} \right) - \rho_{\mathcal{I}} h_{\mathcal{I}} = 0 \tag{5.68}$$

This is used when solving the continuity equation, but the density must then be updated at the end of the iteration when the enthalpy has been updated.

Finally, the dummy cell momentum is calculated from

$$\boldsymbol{m}_{\mathcal{D}} = -\rho_{\mathcal{D}} \boldsymbol{u}_{\mathcal{I}} \tag{5.69}$$

The set of boundary conditions actually imposed on the governing equations is

$$\begin{aligned} \rho_{\mathcal{D}}^{n+1} \left( 2\mathring{h}_w - h_{\mathcal{I}}^{n} \right) - \rho_{\mathcal{I}}^{n+1} h_{\mathcal{I}}^{n} &= 0 \\ (\boldsymbol{m} \cdot \boldsymbol{n})_b^* &= 0 \\ \delta p_{\mathcal{D}}^{n+1} - \delta p_{\mathcal{I}}^{n+1} &= 0 \end{aligned} \tag{5.70}$$

### 5.6.2 Modifications to the Momentum Reconstruction Procedure along the Boundaries

The condition on the flow velocity at the wall suggests modifications to the piecewise linear momentum reconstrution that improve its accuracy and efficiency. First, inserting the condition on the flow velocity at the wall directly into the reconstruction system allows two of the unknowns to be eliminated. This is done simply by setting $a_x = a_y = 0$ in (3.66). The reconstruction system then consists of linear equations of the form

$$\begin{aligned} \begin{bmatrix} \vdots & \vdots & \vdots \\ [(\boldsymbol{c}_s \cdot \boldsymbol{i})\,\Delta x - (\boldsymbol{c}_s \cdot \boldsymbol{j})\,\Delta y] & (\boldsymbol{c}_s \cdot \boldsymbol{i})\,\Delta y & (\boldsymbol{c}_s \cdot \boldsymbol{j})\,\Delta x \\ \vdots & \vdots & \vdots \end{bmatrix} \begin{Bmatrix} b \\ c_x \\ b_y \end{Bmatrix} \\ = \begin{Bmatrix} \vdots \\ (\boldsymbol{m} \cdot \boldsymbol{c})_s \\ \vdots \end{Bmatrix} - \frac{\bar{d}_i}{2} \begin{Bmatrix} \vdots \\ (\Delta \boldsymbol{x} \cdot \boldsymbol{c})_s \\ \vdots \end{Bmatrix} \end{aligned} \tag{5.71}$$

where $\boldsymbol{c}_s$ is either the edge unit normal or the edge unit tangent. It is then possible that the reconstruction system will have sufficient rank even if there are only three edges attached to a particular wall node. Another advantage of this modification is that it

guarantees that the boundary conditions imposed on the momentum along no-slip wall boundaries are exactly satisfied at the wall nodes. This modification is used for all the numerical examples presented in the remainder of the thesis.

Another modification can be derived by first transforming the piecewise linear reconstruction polynomial to the basis formed from the wall unit tangent $\boldsymbol{t}_w$ and unit normal $\boldsymbol{n}_w$. This yields

$$\bar{m}\left(\boldsymbol{x}\right) = \left\{ \begin{array}{c} b_t\left(\boldsymbol{x}\cdot\boldsymbol{t}_w\right) + c_t\left(\boldsymbol{x}\cdot\boldsymbol{n}_w\right) \\ b_n\left(\boldsymbol{x}\cdot\boldsymbol{t}_w\right) + c_n\left(\boldsymbol{x}\cdot\boldsymbol{n}_w\right) \end{array} \right\} \tag{5.72}$$

Next, observe that the same condition is imposed on the momentum along the boundary. Therefore, the spatial derivatives of the momentum in the tangential direction are zero

$$\frac{\partial \boldsymbol{m}\cdot\boldsymbol{t}_w}{\partial \boldsymbol{t}_w} = \frac{\partial \boldsymbol{m}\cdot\boldsymbol{n}_w}{\partial \boldsymbol{t}_w} = 0 \tag{5.73}$$

Imposing these conditions on the reconstruction polynomial yields

$$\bar{m}\left(\boldsymbol{x}\right) = \left\{ \begin{array}{c} c_t\left(\boldsymbol{x}\cdot\boldsymbol{n}_w\right) \\ c_n\left(\boldsymbol{x}\cdot\boldsymbol{n}_w\right) \end{array} \right\} \tag{5.74}$$

However, since the momentum divergence at the wall is

$$\nabla\cdot\boldsymbol{m}_w = \frac{\partial \boldsymbol{m}\cdot\boldsymbol{t}_w}{\partial \boldsymbol{t}_w} + \frac{\partial \boldsymbol{m}\cdot\boldsymbol{n}_w}{\partial \boldsymbol{n}_w} \quad = \frac{\partial \boldsymbol{m}\cdot\boldsymbol{n}_w}{\partial \boldsymbol{n}_w} \tag{5.75}$$

the reconstruction polynomial can be replaced by

$$\bar{m}\left(\boldsymbol{x}\right) = \left\{ \begin{array}{c} c_t\left(\boldsymbol{x}\cdot\boldsymbol{n}_w\right) \\ \bar{d}\left(\boldsymbol{x}\cdot\boldsymbol{n}_w\right) \end{array} \right\} \tag{5.76}$$

where $\bar{d}$ is the approximation to the momentum divergence given by (3.62).

The second modification reduces the number of unknown coefficients to one, but the resulting solver is unstable. This is likely due to the fact that the momentum reconstruction based at the wall nodes is used to reconstruct the momentum at the circumcentres of the cells and centres of the edges adjacent to the wall, which can be far enough away that the conditions 5.73 are not close to being met. However this modification is likely to be of use when considering turbulent flows at reasonably high Reynolds numbers, as the required grids are signficantly finer near the walls (at least in the normal direction).

## 5.7 Stability and Efficiency

### 5.7.1 Modification of the Optimum Time Step

In Section 4.6.1, a procedure for the computation of the optimium time step for the inviscid solver was presented, which was based on approximating the spectral radius of the convective flux Jacobian. For the Navier-Stokes equations this procedure is no longer sufficient, as the *viscous* flux Jacobian must also be taken into account. For this purpose, the procedure of [Bla01, pp. 188–189] was adapted for the laminar case. The resulting procedure is

1. Calculate the lengths of the projections of the control volume on the x and y axes

$$\begin{aligned} \Delta S_{x_i} &= \max_{v(i)} (\boldsymbol{x}_v \cdot \boldsymbol{i}) - \min_{v(cv)} (\boldsymbol{x}_v \cdot \boldsymbol{i}) \\ \Delta S_{y_i} &= \max_{v(i)} (\boldsymbol{x}_v \cdot \boldsymbol{j}) - \min_{v(cv)} (\boldsymbol{x}_v \cdot \boldsymbol{j}) \end{aligned} \tag{5.77}$$

   where $v(i)$ denotes the set of vertices of the control volume, and $\boldsymbol{x}_v$ denotes the position of node $v$.

2. Compute the approximate spectral radii of the convective flux Jacobian

$$\begin{aligned} \hat{\Lambda}^x_{c_i} &= (|\boldsymbol{u}.\boldsymbol{i}|) + a_i) \, \Delta S_{x_i} \\ \hat{\Lambda}^y_{c_i} &= (|\boldsymbol{u}.\boldsymbol{j}|) + a_i) \, \Delta S_{y_i} \end{aligned} \tag{5.78}$$

3. Compute the approximate spectral radii of the viscous flux Jacobian

$$\begin{aligned} \hat{\Lambda}^x_{v_i} &= \max\left(\frac{4}{3\rho}, \frac{\gamma}{\rho}\right) \cdot \frac{\mu \left(\Delta S_{x_i}\right)^2}{\Omega_i} \\ \hat{\Lambda}^y_{v_i} &= \max\left(\frac{4}{3\rho}, \frac{\gamma}{\rho}\right) \cdot \frac{\mu \left(\Delta S_{y_i}\right)^2}{\Omega_i} \end{aligned} \tag{5.79}$$

4. Compute the maximum local time step

$$\Delta t_i = \sigma \frac{\Omega_i}{\left(\hat{\Lambda}^x_{c_i} + \hat{\Lambda}^y_{c_i}\right) + \frac{C}{R_e}\left(\hat{\Lambda}^x_{v_i} + \hat{\Lambda}^y_{v_i}\right)} \tag{5.80}$$

   The parameter $C$ can be tuned as necessary, but $C = 4$ is often used.

## 5.8 Grid Refinement Study

In order to analyse the effect of adding the viscous terms on the order-of-accuracy of the solver, the grid refinement study is now repeated for a pair of viscous test cases. The first is solved on a series of grids with non-uniform meshwidth, whilst the second is an interior flow on grids with uniform meshwidth. The procedure is identical to that presented for the inviscid solver in Section 4.7.1.

### 5.8.1 Results - Non-Uniform Grid

The test case chosen for this grid refinement study was the right circular cylinder in low Reynolds number, weakly compressible flow. This test case is explored further in Section 5.9.2, but for the purposes of the grid refinement study the flow conditions were fixed at $M_r = 0.25$, $R_e = 25$, $P_r = 0.71$. This test case was selected as it provides steep but smooth gradients in the primary flow variables and no points of non-differentiability in the geometry.

The target meshwidth was kept constant along the cylinder wall. The farfield was placed 70 cylinder diameters from the geometry to ensure that the wake-farfield interaction had minimal effect on the solution at the wall. The target meshwidth at the farfield was set to 32.0, and the target growth rate across the grid was limited to 1.12 in order to obtain good grid quality across the grids at the expense of somewhat high total cell counts.

The required one-parameter family of grids is very simple to obtain in this case, since the target meshwidth at the wall $h_w$ is the same at all points. The parameter is therefore identified with the target meshwidth at the wall. The coarsest grid had $h_w = 1.063 \times 10^{-2}$, which resulted in $95,000$ cells; the finest grid had $h_w = 2.136 \times 10^{-4}$, which resulted in $433,820$ cells. The solution on the finest grid was again used as the reference solution, to which the solutions on the coarser grids were compared.

As for the grid refinement study performed on the inviscid solver the reference solution at the wall was interpolated using a piecewise cubic interpolant, allowing the approximate error in the wall pressures to be calculated at each wall node on the coarser grids. By the use of the four norms (4.76)-(4.79), error estimates for each grid were then calculated. The results are shown in Table 5.1. Next, the natural logarithm of the meshwidth is plotted against the natural logarithm of each of the error norms, as shown in Figure 5.3. Finally, linear regression is used on the data points judged to be in the asymptotic region of the convergence plot, producing the approximations to the spatial accuracy of the viscous solver shown on the left of Table 5.2. The process can

| $h$ | $N_{\text{cells}}$ | $\|\varepsilon_p\|_{\mathcal{L}_1}$ | $\|\varepsilon_p\|_{\mathcal{L}_2}$ | $\|\varepsilon_p\|_{\mathcal{L}_\infty}$ | $\|\varepsilon_p\|_{\mathcal{F}_2}$ |
|---|---|---|---|---|---|
| 1.06E-03 | 95000 | 5.97E-01 | 5.35E-02 | 8.71E-03 | 7.00E-03 |
| 7.97E-04 | 123658 | 5.18E-01 | 4.64E-02 | 7.79E-03 | 6.07E-03 |
| 6.64E-04 | 146378 | 4.82E-01 | 4.36E-02 | 7.03E-03 | 5.70E-03 |
| 5.31E-04 | 180830 | 3.84E-01 | 3.48E-02 | 5.72E-03 | 4.55E-03 |
| 4.70E-04 | 203862 | 2.72E-01 | 2.45E-02 | 4.22E-03 | 3.20E-03 |
| 3.98E-04 | 239598 | 2.75E-01 | 2.51E-02 | 4.17E-03 | 3.28E-03 |
| 3.49E-04 | 272166 | 1.36E-01 | 1.23E-02 | 1.97E-03 | 1.61E-03 |
| 3.24E-04 | 291932 | 2.15E-01 | 1.97E-02 | 3.35E-03 | 2.57E-03 |
| 2.99E-04 | 314482 | 1.32E-01 | 1.26E-02 | 2.52E-03 | 1.65E-03 |

Table 5.1: The computed error norms for the right circular cylinder case.

| | | | |
|---|---|---|---|
| $\|\varepsilon_p\|_{\mathcal{L}_1}$ | 2.0357 | $\|\varepsilon_\rho\|_{\mathcal{L}_1}$ | 2.1593 |
| $\|\varepsilon_p\|_{\mathcal{L}_2}$ | 1.8934 | $\|\varepsilon_\rho\|_{\mathcal{L}_2}$ | 2.0337 |
| $\|\varepsilon_p\|_{\mathcal{L}_\infty}$ | 1.3122 | $\|\varepsilon_\rho\|_{\mathcal{L}_\infty}$ | 1.3662 |
| $\|\varepsilon_p\|_{\mathcal{F}_2}$ | 1.8984 | $\|\varepsilon_\rho\|_{\mathcal{F}_2}$ | 2.0427 |

Table 5.2: The computed approximations to the spatial accuracy of the viscous solver for the right circular cylinder case, using the pressure and density errors on the cylinder surface and each of the four norms defined previously.

be repeated for the density at the wall, which produces the results shown on the right of the same table.

As can be seen in Table 5.2, second order spatial accuracy is achieved in the $\mathcal{L}_1$, $\mathcal{L}_2$, and $\mathcal{F}_2$ norms. However, in the $\mathcal{L}_\infty$ norm the spatial accuracy is found to be superlinear, but not nearly quadratic. These results imply that the viscous solver is second-order spatially accurate, but not uniformly so [Car92], [Lan98]. This situation is shared by many of the schemes examined in that reference. In practice this is not likely to be problematic, since the parts of the solution field where the lack of uniform second-order convergence becomes important are also the parts of the solution field (e.g. in the neighbourhood of shocks and non-differentiable geometry) where the limiters will be activated and second-order spatial accuracy is deliberately abandoned.

### 5.8.2 Results - Uniform Grid

The test case chosen for the second grid refinement study was the compressible Couette flow between two parallel plates. The reason for selecting this test case is that, in addition to allowing the use of a uniform grid, it is one of the few compressible viscous flow cases for which an analytic solution is known [Ill50]. The geometry is

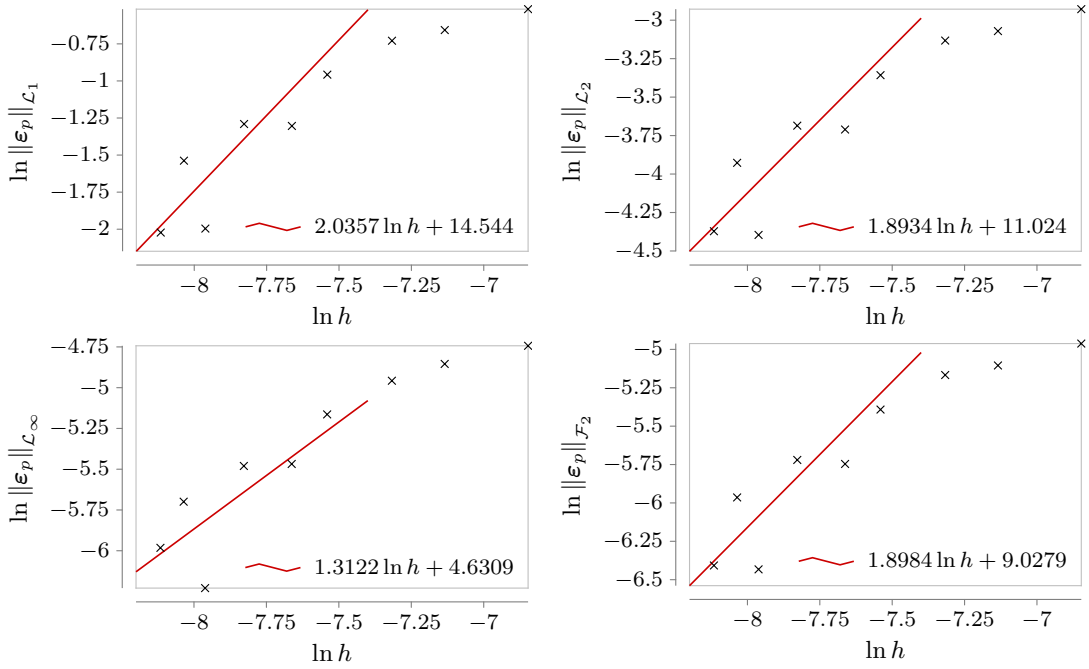



Figure 5.3: The natural logarithm of the meshwidth against the natural logarithms of the pressure error in each of the norms for the right circular cylinder case, together with best-fit lines placed using linear regression on the four data points with the lowest values of the meshwidth.

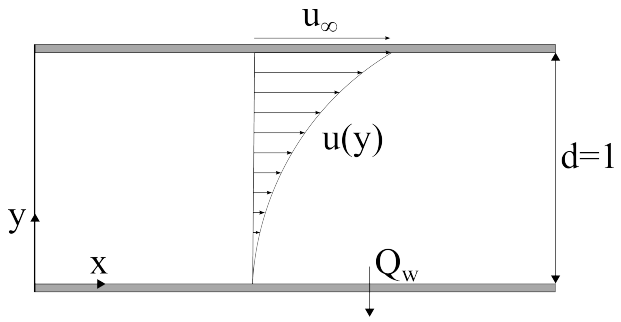



Figure 5.4: The geometry for compressible Couette flow.

shown in Figure 5.4. The distance between inlet and outlet was set to ten channel heights, to ensure that the solution at the centre was not strongly dependent on the implementation of the inlet and outlet boundary conditions.

The subscript $w$ is used to denote conditions at the lower wall, whilst the subscript $\infty$ is used to denote conditions at the top wall. The conditions at the top wall are used as the reference conditions. The bottom wall is adiabatic, $Q_w = 0$, and stationary, $u_w = 0$. The top wall is held at constant temperature $T_\infty$ and moves at speed $u_\infty$ in the positive x-direction. The conditions at inlet and outlet are set equal to the analytic values, to yield fully developed flow through the channel. Since the flow is fully developed, the analytic solution is independent of $x$ [LR57, p. 306]. In addition, the pressure $p$ and shear stress $\tau_{xy}$ are found to be constant throughout the flow. The $y$-velocity is everywhere zero. The dynamic viscosity is assumed to vary linearly with the temperature

$$\frac{\mu}{\mu_\infty} = \frac{T}{T_\infty} \tag{5.81}$$

With this assumption, the non-dimensional $x$-velocity $u$ at a given height $y$ is given implicitly as the unique root of

$$\frac{P_r M_r^2 (\gamma - 1)}{6} u^3 - \left(1 + \frac{P_r M_r^2 (\gamma - 1)}{2}\right) u + \left(1 + \frac{P_r M_r^2 (\gamma - 1)}{3}\right) y = 0 \tag{5.82}$$

that satisfies $0 \leq u \leq 1$. Once the $x$-velocity is determined, the enthalpy and density can be calculated from

$$\begin{aligned} h &= 1 + \frac{P_r M_r^2 (\gamma - 1)}{2} \left(1 - u^2\right) \\ \rho &= \frac{1}{h} \end{aligned} \tag{5.83}$$

The latter of these equations results from the combination of the non-dimensionalised

| $h$ | $N_{\text{cells}}$ | $\|\boldsymbol{\varepsilon}_p\|_{\mathcal{L}_1}$ | $\|\boldsymbol{\varepsilon}_p\|_{\mathcal{L}_2}$ | $\|\boldsymbol{\varepsilon}_p\|_{\mathcal{L}_\infty}$ | $\|\boldsymbol{\varepsilon}_p\|_{\mathcal{F}_2}$ |
|---|---|---|---|---|---|
| 2.50E-02 | 54252 | 5.94E-01 | 2.59E-02 | 3.15E-02 | 1.16E-03 |
| 1.43E-02 | 167058 | 2.57E-01 | 4.67E-03 | 2.56E-02 | 5.08E-04 |
| 1.25E-02 | 218814 | 2.13E-01 | 3.24E-03 | 2.34E-02 | 3.51E-04 |
| 1.11E-02 | 276444 | 1.80E-01 | 2.42E-03 | 1.00E-02 | 2.80E-04 |
| 1.05E-02 | 308006 | 1.83E-01 | 2.42E-03 | 1.38E-02 | 2.94E-04 |
| 1.00E-02 | 341806 | 1.72E-01 | 1.97E-03 | 1.46E-02 | 2.23E-04 |

Table 5.3: The computed error norms for the compressible Couette case.

| | |
|---|---|
| $\|\boldsymbol{\varepsilon}_p\|_{\mathcal{L}_1}$ | 0.9236 |
| $\|\boldsymbol{\varepsilon}_p\|_{\mathcal{L}_2}$ | 2.0709 |
| $\|\boldsymbol{\varepsilon}_p\|_{\mathcal{L}_\infty}$ | 2.143 |
| $\|\boldsymbol{\varepsilon}_p\|_{\mathcal{F}_2}$ | 1.7652 |

| | |
|---|---|
| $\|\boldsymbol{\varepsilon}_\rho\|_{\mathcal{L}_1}$ | 0.9564 |
| $\|\boldsymbol{\varepsilon}_\rho\|_{\mathcal{L}_2}$ | 2.0187 |
| $\|\boldsymbol{\varepsilon}_\rho\|_{\mathcal{L}_\infty}$ | 1.3431 |
| $\|\boldsymbol{\varepsilon}_\rho\|_{\mathcal{F}_2}$ | 1.5713 |

Table 5.4: The computed approximations to the spatial accuracy of the viscous solver for the compressible Couette case, using the pressure and density errors along the line slice $(5, 0) \rightarrow (5, 1)$ and each of the four norms defined previously.

equation of state (3.12) with the fact that the pressure is everywhere equal to the reference pressure.

The meshwidth was constant across each individual grid. The coarsest grid had 40 edges along its inlet and $54,252$ cells, whilst the finest grid had 100 edges along its inlet and $341,806$ cells. The error in the converged solutions was evaluated by interpolating the solution to thirty equally-spaced stations along the line from (5, 0) to (5, 1) and comparing to the analytic solution. The resulting error norms are shown in Table 5.3, and the natural logarithm of the meshwidth is plotted against the natural logarithm of the error norms in Figure 5.5. Best-fit lines were placed on the scatter plots using linear regression on the four data points with the smallest meshwidths, producing the approximations to the order-of-accuracy of the solver shown in Table 5.4.

The results show that second-order spatial accuracy has been achieved in the $\mathcal{L}_2$ and $\mathcal{F}_2$ norms. For the $\mathcal{L}_\infty$ norm second-order accuracy is achieved for the pressure but only superlinear accuracy is achieved for the density. In the $\mathcal{L}_1$ norm, only first-order accuracy is achieved. These results suggest that the convergence to the analytic solution is second-order accurate "in the mean" but not pointwise [Lan98]. This is likely due to the boundary conditions, since the exact analytic solution is imposed at the inlet and outlet, but not on the no-slip walls. The interpolation of the interior flow variables to the wall is only first-order in the present scheme, hence the solution at points on the wall only achieves linear convergence to the analytic solution. The

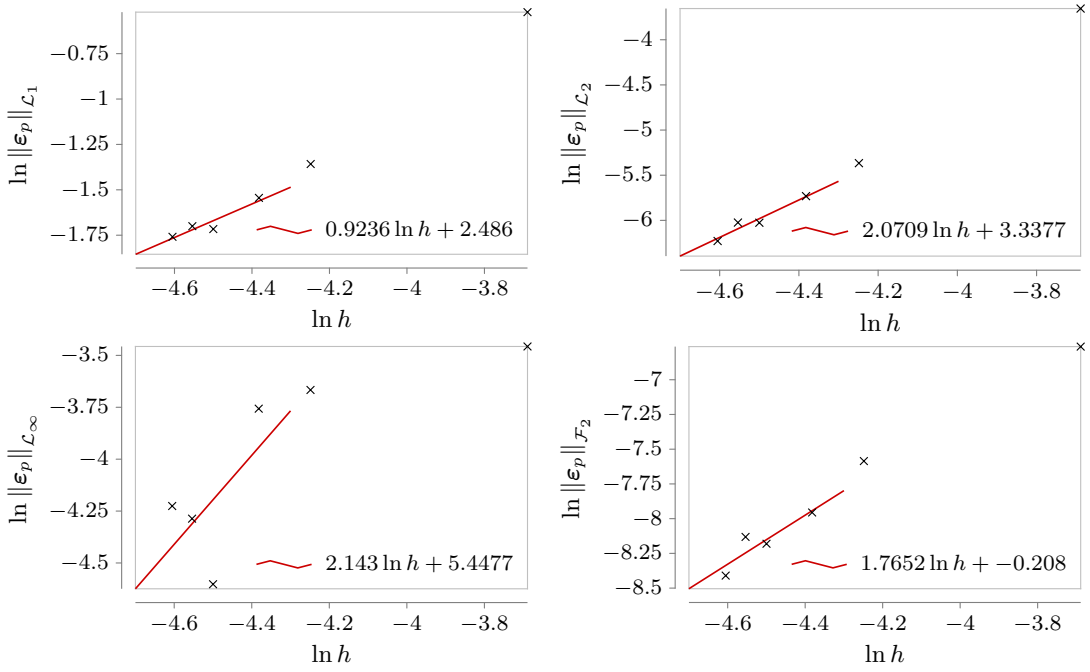



Figure 5.5: The natural logarithm of the meshwidth against the natural logarithms of the pressure error in each of the norms for the compressible Couette case, together with best-fit lines placed using linear regression on the four data points with the lowest values of the meshwidth.

use of a more accurate wall boundary condition would alleviate this, but at the cost of increased cost and reduced stability.

## 5.9 Presentation and Discussion of Numerical Results

Several viscous flow cases are now considered, in order to test the accuracy and efficiency of the viscous solver. All of these cases contain either shocks or wakes, and therefore line source refinement is used to ensure proper resolution of these. To place the line sources, a solution is first obtained on a coarse, unrefined grid of approximately $8,000$ to $10,000$ cells. The line sources are then placed over the shocks and wakes on this solution and used to generate the fine grids. After obtaining the solution on the fine grid, the line source location is checked and updated if the shocks or wakes have moved significantly. This correction is done iteratively if necessary, though this was not required for any of the cases considered here.

The coarse grid solutions used to place the line sources serve a dual purpose. It was found that using the coarse grid solutions as the initial conditions when solving on the fine grids lead to significantly better convergence properties, even when taking into account that the initial error is smaller. This is probably due to the faster decay of the lower frequency error waves on coarser grids. This property of the solver is shared with many other CFD solvers, and can be exploited more fully through the use of multigrid [Bra77]. Though this is beyond the scope of the current work, it is a very promising future direction of development for the solver and is briefly discussed in the final chapter.

### 5.9.1 NACA 0012 Aerofoil

Viscous flow over the NACA 0012 aerofoil is first considered. This is a particularly useful test case as numerical and experimental results are available in the literature, and hence allow the accuracy of the solver to be analysed. The results used for comparison here are those provided in contributions to the 1985 GAMM workshop [Bri+87a]. For both test cases involving the NACA 0012 the dynamic viscosity and thermal conductivity were assumed to remain constant and equal to the farfield values across the whole flow domain, in order to match the problem specification for the workshop. As for the inviscid case, all reported lengths are non-dimensionalised by the aerofoil chord length.

In simulations of exterior viscous flows at low Reynolds numbers, formation of a

substantial wake is usually expected. Therefore, the downstream section of the farfield used for the inviscid case was linearly stretched by a factor of two to allow space for the wake to be properly resolved without significant wake-farfield interaction.

**Transonic**

The transonic test case is the first case from the GAMM workshop. The Mach number, angle of attack, and Reynolds number are 0.8, 10°, and 73 respectively. The wall temperature is set equal to the farfield total temperature. Despite the high angle of attack, the flow remains largely attached to the top surface of the aerofoil and laminar.

For this case the target meshwidth on the aerofoil surface varied from $2.5 \times 10^{-3}$ at the points of highest curvature to $3 \times 10^{-3}$ at the points of lowest curvature. A point source was placed at the trailing edge, where the boundary is non-differentiable, to reduce the meshwidth to $1 \times 10^{-3}$. Line sources were placed using the method described above to properly capture the wake. The target growth rate was limited to 1.15 across the grid. This resulted in 737 boundary nodes, of which 690 were on the aerofoil wall. The complete grid contained $22,570$ nodes and $44,403$ cells. The section of the grid near the aerofoil is shown on the left of Figure 5.6. The selected Courant number of 8.0 lead to convergence to $\Theta = -5.5$ in $4,102$ iterations.

The results are shown as iso-Mach contours on the right of Figure 5.6 and as a surface skin friction coefficient plot against a solution from the literature in Figure 5.7, where the skin friction coefficient is defined by

$$C_f \coloneqq \frac{\tau_w}{\frac{1}{2}\rho_\infty \|\boldsymbol{u}\|^2} \tag{5.84}$$

where

$$\tau_w \coloneqq \boldsymbol{n}_w \cdot \boldsymbol{\tau}_w \cdot \boldsymbol{t}_w = \mu_w \left( \frac{\partial \boldsymbol{u} \cdot \boldsymbol{n}_w}{\partial \boldsymbol{t}_w} + \frac{\partial \boldsymbol{u} \cdot \boldsymbol{t}_w}{\partial \boldsymbol{n}_w} \right) \tag{5.85}$$

is the shear stress at the wall. Note that there is a sign ambiguity inherent in this definition, since the unit tangent to the aerofoil wall can be defined to either point from leading edge to trailing edge or from trailing edge to leading edge. In this thesis, the convention followed is that the wall unit tangent is defined so that the wall unit normal points outwards from the flow domain (with the unit tangent and the unit normal forming a right-handed set). Thus a positive value of $C_f$ on the top surface of the aerofoil and a negative value of $C_f$ on the bottom surface both correspond to skin friction that is doing work against the mean flow (when the mean flow velocity vector forms an acute angle with the positive $x$ direction). The contributors to the GAMM

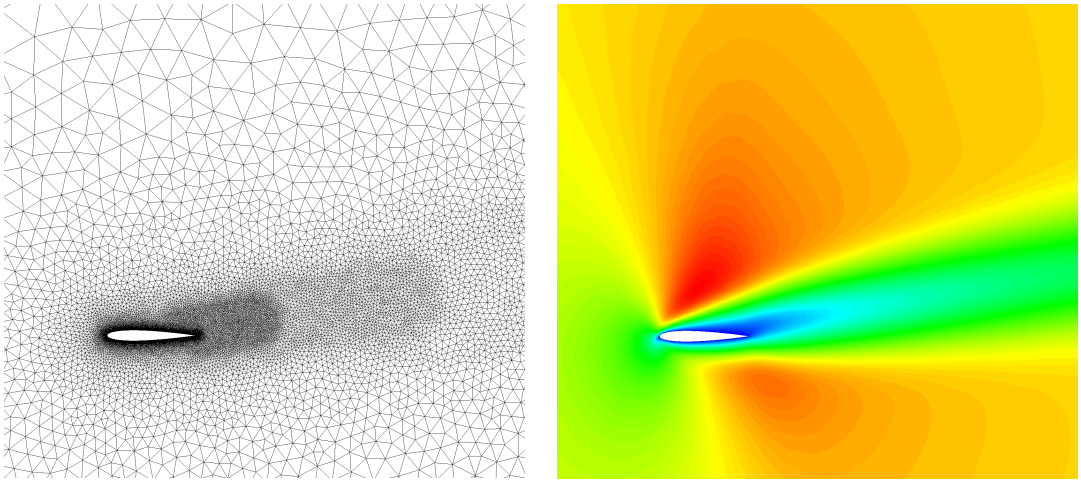


Figure 5.6: Left: a section of the grid used for the NACA 0012 aerofoil at $M_\infty = 0.8$, $\alpha_\infty = 10°$, $R_e = 73$. Right: the resulting Mach number contours.

workshop did not all follow the same convention.

The results show quite close agreement with the published results. Note the skin friction coefficient diverges somewhat at the trailing edge. This is likely caused by the non-differentiability of the geometry and consequently the shear stress either being zero or infinite at the trailing edge. In a real fluid flow, it is driven to zero (this is the origin of the *Kutta condition*, see [And10]). The resolution of the solutions presented at the GAMM workhop are too coarse to capture the same effect if this is indeed a feature of the respective solvers.

Finally, the computed lift, drag, and pitching moment coefficients are given in Table 5.5 together with those from the GAMM workshop. The computed lift coefficient is in good agreement with the other solutions, whilst the drag coefficient and pitching moment coefficients are at the lower end of the ranges of the other solutions.

### Supersonic

The supersonic test case is the third test case from the GAMM workshop. The Mach number, angle of attack, and Reynolds number are 2.0, 10°, and 106 respectively. The wall temperature is set equal to the farfield total temperature. The flow again remains attached to the aerofoil and fully laminar.

For this case, the target meshwidth on the aerofoil surface varied linearly from $2.5 \times 10^{-3}$ at the points where the boundary curvature was highest $3 \times 10^{-3}$ at the points of lowest boundary curvature. A point source was placed at the trailing edge to reduce

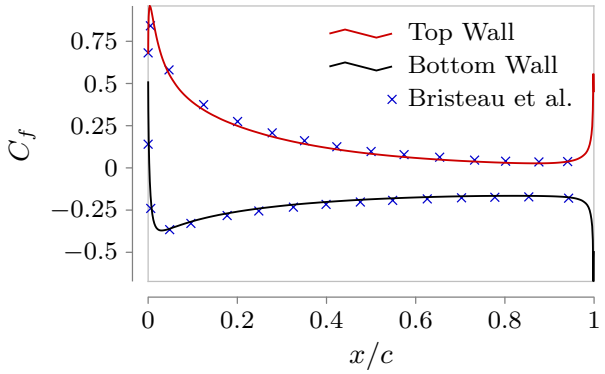



Figure 5.7: Skin friction coefficient on the surface of the NACA 0012 aerofoil at $M_\infty = 0.8$, $\alpha_\infty = 10°$, $R_e = 73$, compared to the results given in [Bri+87b].

| | $c_L$ | $c_D$ | $c_M$ |
|---|---|---|---|
| 2 [Ang87] | 0.5916 | 0.58 | 0.233 |
| 3 [Bri+87b] | 0.551 | 0.608 | 0.216 |
| 4 [Cam87] | 0.5465 | 0.6280 | 0.2118 |
| 5 [GJM87] | 0.6657 | 0.2522 | - |
| 6 [Haa87] | 0.5959 | 0.6377 | -0.05119 |
| 7 [KVB87] | 0.5584 | 0.6568 | 0.1912 |
| 8 [Kor87] | 0.5380 | 0.7046 | -0.2377 |
| 9 [MBR87] | 0.5242 | 0.6276 | 0.2019 |
| 10 [SMN87] | 0.5268 | 0.6447 | 0.2050 |
| 11 [SDN87] | 0.6127 | 0.6377 | 0.2453 |
| Present solver | 0.5814 | 0.5009 | -0.05592 |

Table 5.5: Computed aerodynamic coefficients for the NACA 0012 aerofoil at $M_\infty = 0.8$, $\alpha_\infty = 10°$, $R_e = 73$, compared to those provided in contributions to the GAMM workshop [Bri+87a].

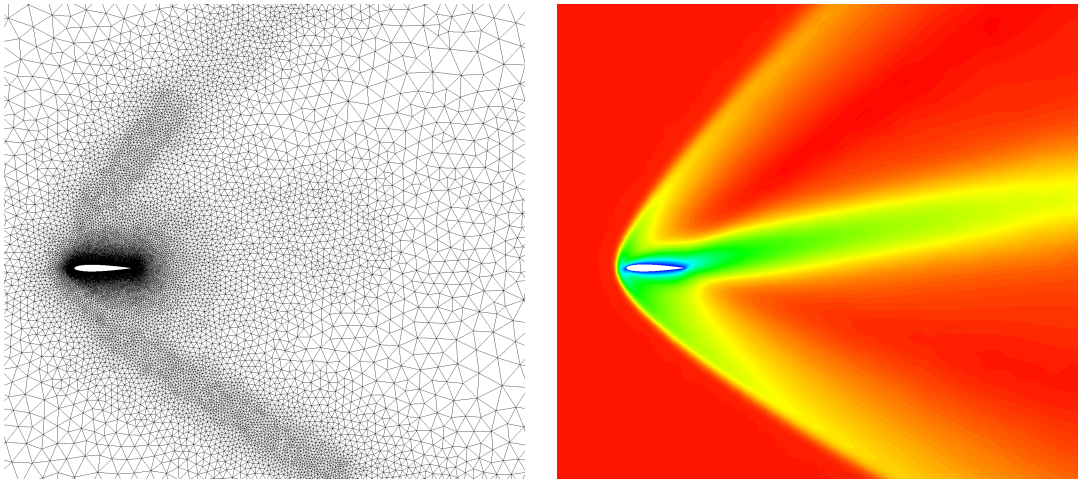


Figure 5.8: Left: a section of the grid used for the NACA 0012 aerofoil at $M_\infty = 2$, $\alpha_\infty = 10°$, $R_e = 106$. Right: the resulting Mach number contours.

the meshwidth to $1 \times 10^{-3}$ to ensure good resolution despite the non-differentiability of the geometry. Due to the high Mach number and angle of attack a detached bow shock is expected as well as a substantial wake, and line sources are used to refine the grid in these regions as usual. The resulting grid had 745 boundary nodes, of which 697 were on the aerofoil surface. The selected target growth rate of 1.15 lead to $35,140$ nodes and $69,535$ cells across the grid. A section of the grid is shown on the left of Figure 5.8. The selected Courant number of 2 lead to convergence to $\Theta = -5.5$ in $2,829$ iterations. The historic modification described in Section **??** was applied to the limiters at iteration $1,250$ to prevent convergence stalling.

The results are again shown as iso-Mach contours on the right of Figure 5.8 and as a surface skin friction coefficient plot in Figure 5.9. The skin friction coefficient plot again shows good agreement between the solver and the displayed solution from the literature. However, the skin friction coefficient again diverges somewhat at the trailing edge, for the same reason as discussed in the transonic case.

Finally, the computed lift, drag, and pitching moment coefficients are given in Table 5.6 together with those from the GAMM workshop. The lift and drag coefficients are in good agreement with the other solutions, whilst the pitching moment coefficient is slightly below the bottom of the range of the other solutions. It is important to note that reducing the resolution at the trailing edge causes a signficant increase in the drag coefficient. For example, removing the trailing edge point source increased the computed drag coefficient to 0.6932, because the skin friction divergence visible at the

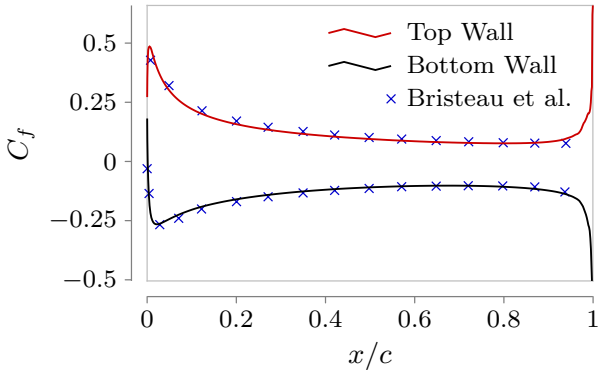



Figure 5.9: Skin friction coefficient on the surface of the NACA 0012 aerofoil at $M_\infty = 2$, $\alpha_\infty = 10°$, $R_e = 106$, compared to the results given in [Bri+87b].

| | $c_L$ | $c_D$ | $c_M$ |
|---|---|---|---|
| 2 [Ang87] | 0.367 | 0.412 | 0.1817 |
| 3 [Bri+87b] | 0.331 | 0.443 | 0.171 |
| 4 [Cam87] | 0.3289 | 0.4826 | 0.1698 |
| 5 [GJM87] | 0.3716 | 0.1851 | - |
| 6 [Haa87] | 0.3063 | 0.4772 | -0.05061 |
| 9 [MBR87] | 0.3261 | 0.4771 | 0.1652 |
| 10 [SMN87] | 0.3173 | 0.4910 | 0.1640 |
| 11 [SDN87] | 0.4059 | 0.4791 | 0.2044 |
| Present solver | 0.3410 | 0.4554 | -0.05977 |

Table 5.6: Computed aerodynamic coefficients for the NACA 0012 aerofoil at $M_\infty = 2$, $\alpha_\infty = 10°$, $R_e = 106$, compared to those provided in contributions to the GAMM workshop [Bri+87a].

trailing edge in Figure 5.9 then occurs over a much larger portion of the aerofoil. This highlights the importance of achieving a fine mesh in the region of non-differentiable boundary points when generating meshes for use with the solver presented in this thesis.

### 5.9.2 Right Circular Cylinder

The second test case considered for the viscous solver is low Reynolds number flow over a right circular cylinder. Although the geometry is very simple, the range of resulting flow phenomena is surprisingly rich. The exact behaviour obviously depends on the Mach number, but follows a similar pattern across quite a wide range of speeds. At low Reynolds numbers a large, stable wake forms. As the Reynolds number increases vortices begin to form on the cylinder. At some (weakly Mach-number-dependent)

critical Reynolds number laminar shedding of the vortices begins, and the flow becomes unsteady. As the Reynolds number is increased further still, transition to turbulent flow eventually occurs. Turbulent flow is beyond the scope of this thesis, but the behaviour of the flow over the right circular cylinder at lower Reynolds numbers is of interest. Three flow cases were considered, with Reynolds numbers of 20, 40, and 50. This covers the range of flow behaviours from the development of attached vortices to the onset of laminar shedding. For this case, all reported lengths are non-dimensionalised by the diameter of the cylinder.

The wall geometry is simply a unit circle centred at the origin, whilst the farfield geometry is the same as for the NACA 0012 aerofoil case considered previously. The same grid was used for all selected values of the Reynolds number. The target mesh-width on the cylinder surface was constant and set to $4.25 \times 10^{-3}$. In order to ensure proper resolution of the wake, line sources were placed in the streamwise direction downwind of the cylinder, as shown in Figure 5.10. The target growth rate across the grid was set to 1.12 to ensure sufficient grid quality. The resulting grid had 799 boundary nodes, of which 740 were placed on the wall. In total, the complete grid had $29,810$ nodes and $58,821$ cells. For all three cases, a Courant number of 7 was selected. The iterations were continued until the computed coefficient of drag was constant to four significant figures over a period of at least 500 iterations. This required $2,136$ iterations for the $R_e = 20$ case and $3,410$ iterations for the $R_e = 40$ case. For the $R_e = 50$ case convergence to this criterion was achieved, in $3,631$ iterations, but it was noted in the lift coefficient plot shown in Figure 5.11 that there was significant unsteadiness in the solution. This suggests that the critical Reynolds number has been exceeded and laminar vortex shedding is beginning. This is studied further in the next chapter, when the solver is extended to the case of unsteady flows.

For this case, the dynamic viscosity and thermal conductivity were allowed to vary according to Sutherland's laws (2.39) and (2.40). The Mach number was set to 0.5 to produce significant density and pressure gradients without shocks. The resulting coefficient of pressure and x-velocity contours are shown in Figure 5.12, whilst Figure 5.13 shows the coefficient of pressure and coefficient of skin friction on the cylinder wall for each tested value of the Reynolds number. Finally, a comparison of the computed coefficient of drag for each of the cases with results from a DNS study [CT15] is presented in Table 5.7.

At $R_e = 20$, the attached vortices have begun to develop, and the flow is fully symmetric. As the Reynolds number is increased to 40, the vortices become more pronounced, though the symmetry of the flow field is maintained. At $R_e = 50$ the

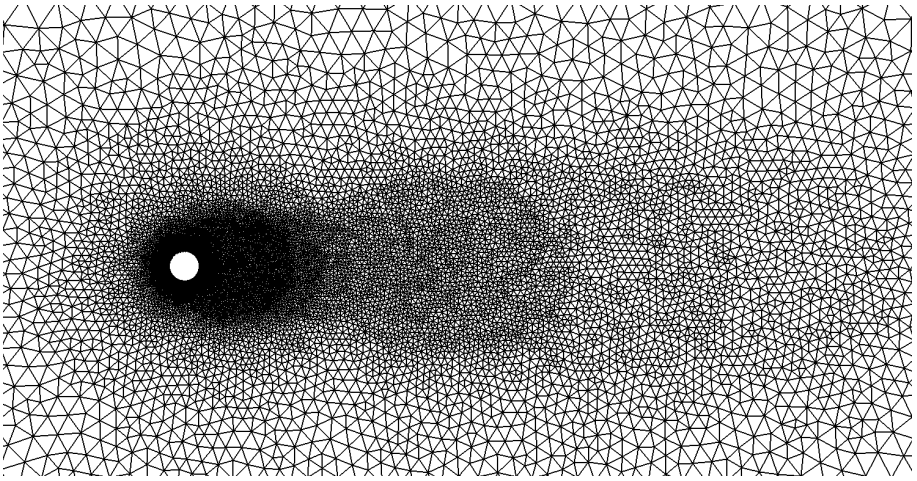


Figure 5.10: A section of the grid used for computing the flow over the right circular cylinder, showing the wake refinement.

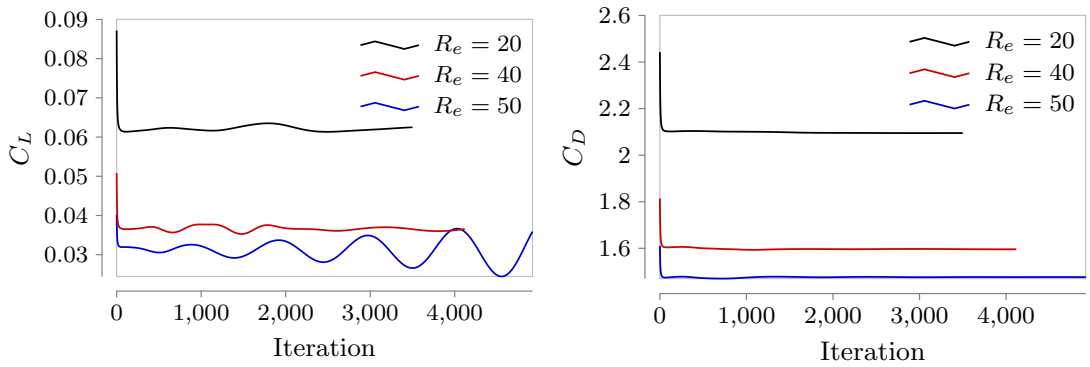



Figure 5.11: Time histories of the coefficient of lift and coefficient of drag for the right circular cylinder at $M_r = 0.5$ for the indicated values of the Reynolds number.

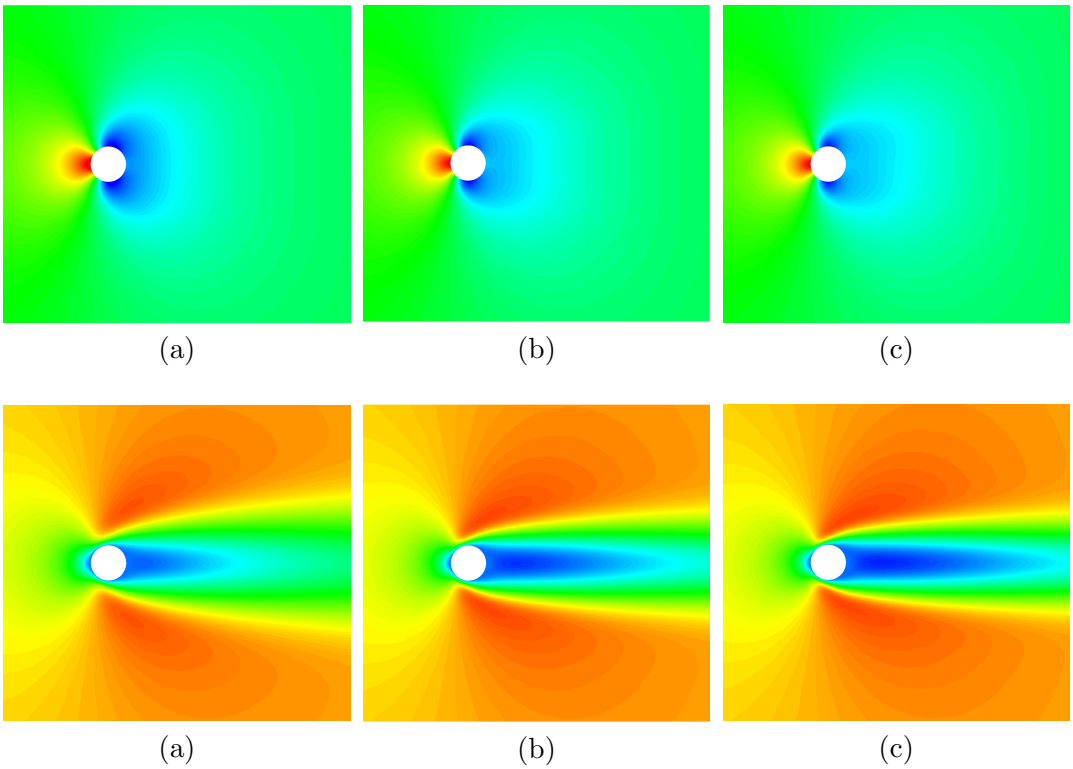



Figure 5.12: Coefficient of pressure contours (top row) and $\boldsymbol{u} \cdot \boldsymbol{i}$ contours (bottom row) around the right circular cylinder at $M_\infty = 0.5$ for (a) $R_e = 20$, (b) $R_e = 40$, and (c) $R_e = 50$ showing the development of vortices on the cylinder wall.

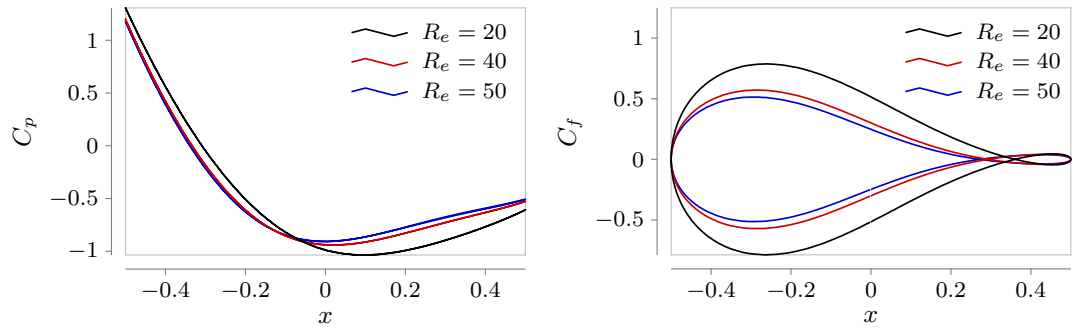



Figure 5.13: Surface plots for the right circular cylinder at $M_r = 0.5$ with Reynolds numbers of 20, 40, and 50. Left: coefficient of pressure. Right: coefficient of skin friction.

| $R_e$ | Present Solver | [CT15] |
|---|---|---|
| 20 | 2.095 | 2.23 |
| 40 | 1.596 | 1.68 |
| 50 | 1.476 | $1.56 \pm 2.03 \times 10^{-5}$ |

Table 5.7: The computed values of the coefficient of drag compared to those resulting from the DNS study presented in [CT15].

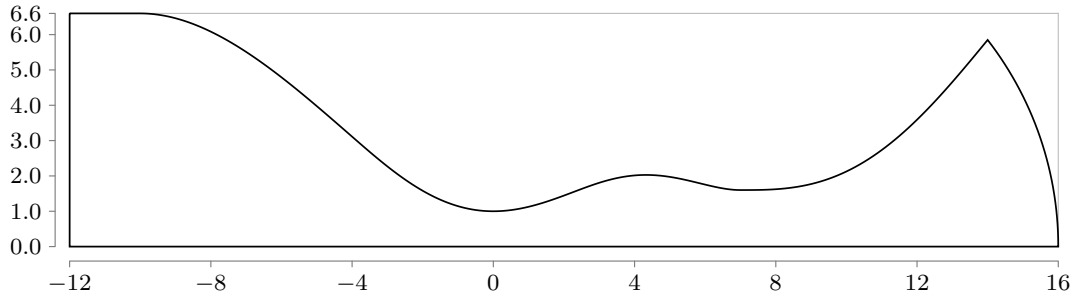



Figure 5.14: The geometry of the double-throated nozzle.

beginning of vortex shedding is presumed to have been reached based on the convergence plots shown previously. However the effect of this is not visible on the pressure and $x$-velocity contours shown in Figure 5.12. Similarly, seemingly exact symmetry in the pressure and skin friction coefficients on the top and bottom walls is shown in Figure 5.13 and the drag coefficient on the cylinder was constant (to four significant figures). This agrees with the results from the DNS study shown in Table 5.7, where the coefficient of drag only varied in the sixth significant figure. In the same table the computed values of the drag coefficient can be seen to be in reasonable agreement with the published results, with values about 5% lower than the published results.

### 5.9.3 Double-Throated Nozzle

The final test case for the viscous solver is the double-throated nozzle from the 1985 GAMM workshop [Bri+87a]. Despite the relatively simple geometry (see Figure 5.14), this is a particularly challenging case. At inlet the flow is in the low incompressible range, with a Mach number below 0.1. The flow is accelerated to the sonic velocity at the first throat, then to the high supersonic range through the second. The wide range of Mach numbers involved and the fact that relatively small changes in the Reynolds number leads to quite different flow features makes this an ideal case for testing the properties of the Mach-uniform viscous solver. For this case, reported lengths are non-dimensionalised by the half-height of the first throat.

Three cases were considered, with the sole difference being the value of the Reynolds number, which in this case was defined as

$$R_{e_0} \coloneqq \frac{\rho_{\text{res}} a_{\text{res}} L_{\text{throat}}}{\mu_{\text{res}}} \tag{5.86}$$

where $\varphi_{\text{res}}$ refers to the reservoir (inlet stagnation) conditions and $L_{\text{throat}}$ is the half-height of the first throat. Despite the significant variation of temperature across the flow domain, the problem specification for the workshop required that the dynamic viscosity and thermal conductivity be assumed constant across the domain. They were therefore used to set the values of the Reynolds number and Prandtl number to their prescribed values, whilst the remaining boundary conditions were taken to be the same for all the cases considered. The required Prandtl number was not stated in the workshop notes, so the value of 0.71 was assumed as this is the value commonly used for air across most of the ideal gas range. The left-most and right-most boundaries shown in Figure 5.14 had pressure inlet and pressure outlet boundary conditions imposed, whilst the bottom boundary was a symmetry plane and the top boundary was a no-slip wall with its temperature set equal to the reservoir total temperature.

The same basic grid was used for all three cases, with line source refinement being added to improve the resolution of the shocks in the higher Reynolds number cases. The target meshwidth on the no-slip wall varied linearly from $4.5 \times 10^{-2}$ to $9 \times 10^{-2}$ depending on the local curvature, so that the finest resolution was placed at the throats. The target meshwidth along the inlet, outlet, and symmetry plane boundaries was set to $8.92 \times 10^{-2}$. A single line source was used to reduce the meshwidth across the width of the first throat to $3 \times 10^{-2}$ in order to properly resolve the normal shock expected there in all three flow cases. Since a thick boundary layer is expected along the no-slip wall, the target growth rate within 0.15 units of the wall was set to zero. Elsewhere in the nozzle the target growth rate was kept small, at 1.07, since the flow variable gradients are quite high across most of the grid.

One difficulty that was experienced with this case was determining when convergence had been achieved. It was found that the convergence indicators defined in Section 4.2 remained stubbornly high even when the actual changes in the flow variables were small. This is likely because the initial changes in the flow variables was small, meaning that the residuals were being normalised by small values. In all three cases, convergence was double-checked by ensuring that the valuess of the primary flow variables on the no-slip wall and symmetry plane were no longer changing appreciably over several hundred iterations.

$R_e = 100$

Since no shocks are expected other than the normal shock at the first throat, the grid used for this case was the basic grid described above. The grid had $1,019$ boundary nodes, of which 740 were placed on the no-slip wall. Across the grid as a whole there were $22,792$ nodes and $44,563$ cells. The grid is shown in Figure 5.15. The selected Courant number of 3.5 resulted in convergence in $6,869$ iterations.

The results are shown as an iso-Mach contour plot in Figure 5.16, as a plot of the skin friction coefficient along the top wall, and as a plot of the Mach number along the symmetry plane. The latter two plots also show some data points from a representative contribution to the GAMM workshop. The skin friction coefficient along the top wall shows exceptionally good agreement with the workshop case. The Mach number along the symmetry axis also shows good agreement in the region before the first throat.

The area of largest disagreement between the computed solution and the reference solution is the Mach number along the symmetry axis in the region between the second throat and the outlet. This discrepancy is apparent for all three Reynolds numbers tested, though it is clearly largest when the Reynolds number is smallest – compare Figure 5.16, Figure 5.19, and Figure 5.22. There are several possible reasons for this. The most important are likely to be

- The grid used to obtain the solution presented here has significantly smaller meshwidth and grid growth rate in the neighbourhood of the centre-line between the second throat and the outlet, likely resulting in significantly less numerical diffusion. This is likely to be of most consequence in the low Reynolds number cases, since the resulting larger velocity gradients would be better resolved by the current solver.

- Unlike the current solver, the reference solution used a cell-centred arrangement for all primary flow variables. Therefore application of the boundary conditions on the normal momentum and flow velocity could only be accomplished indirectly.

- The present solver uses more recent convective flux reconstruction and gradient limiting schemes.

$R_e = 400$

When the Reynolds number is increased to 400, a small lambda shock appears on the symmetry plane between the first and second throats. Line sources were placed using

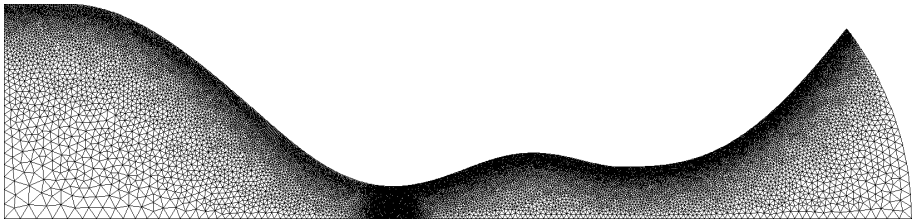


Figure 5.15: The grid used for the double-throated nozzle at $R_e = 100$.

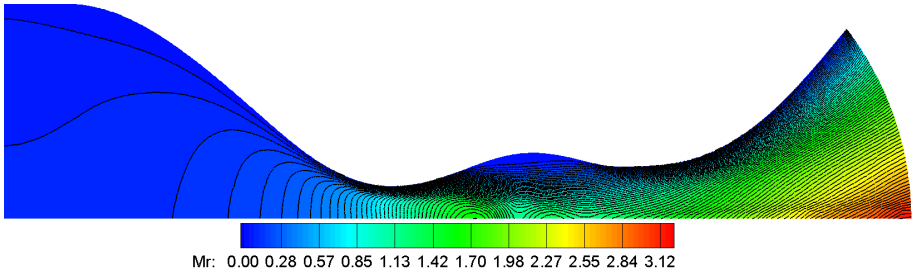



Figure 5.16: Mach number contours on the double-throated nozzle at $R_e = 100$.

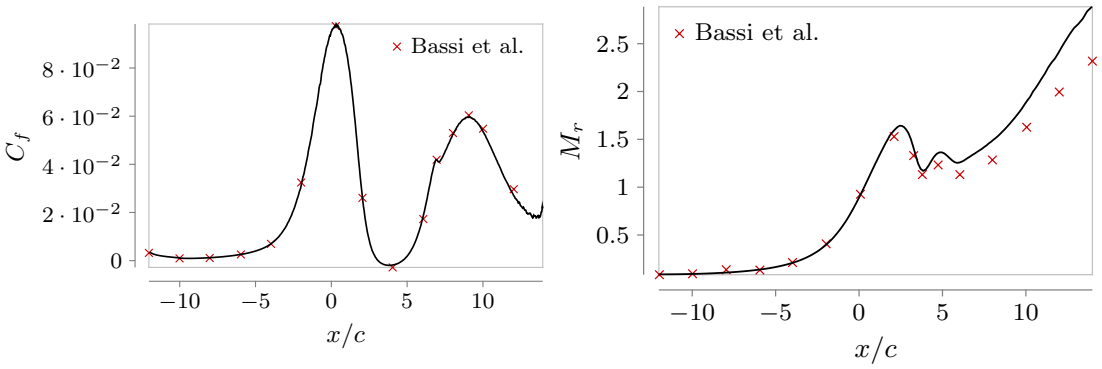



Figure 5.17: Surface plots for the double-throated nozzle at $R_e = 100$, compared to the results given in [Bas+87]. Left: skin friction coefficient on the top wall. Right: Mach number along the centre line

a solution on a significantly coarsened version of the basic grid described above, to ensure that the Lambda shock was properly resolved. Apart from the additional line source refinement, the grid remains the same in as for that used for the $R_e = 100$ case. This resulted in 740 nodes on the no-slip wall and $1,124$ boundary nodes in total. Across the grid, there were $28,208$ nodes and $55,290$ cells. The complete grid is shown in Figure 5.18. The selected Courant number of 3.5 resulted in convergence in $3,615$ iterations.

The results are shown in Figure 5.19 and Figure 5.20. These show good agreement with the reference case, with the lambda shock structure matching that predicted by the workshop contributions. Another area of good agreement between the present solvers and the contributions to the workshop is in the prediction of the separation and reattachment of the boundary layer from the top wall of the nozzle. This can be inferred from the surface plot of skin friction coefficient, where a negative value corresponds to reversed flow on the wall. The location of this reversed flow closely matches that given in each of the workshop contributions.

Figure 5.18: The grid used for the double-throated nozzle with $R_e = 400$.

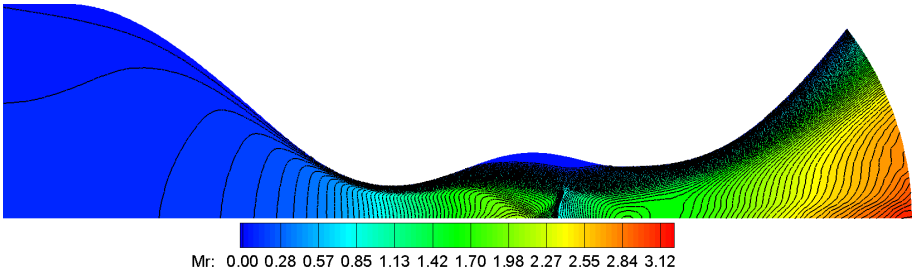



Figure 5.19: Mach number contours on the double-throated nozzle with $R_e = 400$.

$R_e = 1,600$

As the Reynolds number is further increased to $1,600$ the shock structure becomes even more complex. A new oblique shock is formed in front of the region of separated flow between the two throats. This shock reflects from the symmetry plane and interacts with the boundary layer along the top wall of the. The result is a series of oblique shocks reflecting through the second throat. The usual line source method is used to increase the resolution at these shocks. The resulting grid had $1,014$ boundary nodes, of which 621 were placed on the no-slip wall. The complete grid had $26,502$ nodes and $51,988$ cells, and is shown in Figure 5.21. The selected Courant number of 2.5 resulted in convergence in $3,601$ iterations.

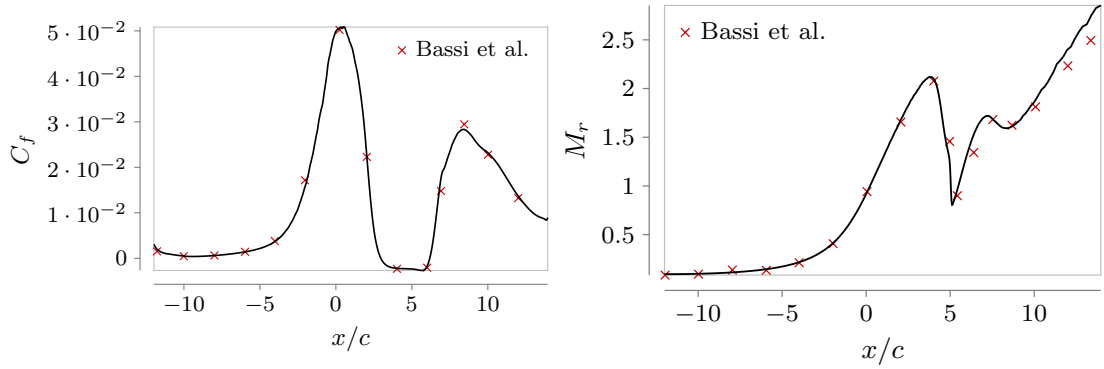



Figure 5.20: Surface plots for the double-throated nozzle at $R_e = 400$, compared to the results given in [Bas+87]. Left: skin friction coefficient on the top wall. Right: Mach number along the centre line

The results are shown as an iso-Mach contour plot in Figure 5.22, and as surface plots of the top wall skin friction coefficient and the Mach number along the centre line on the left and right of Figure 5.23 respectively. As with the lower Reynolds number cases, the agreement with the reference solutions is very good. In particular the shock is crisply resolved, and its location and reflection angles closely match that predicted by the workshop contributions. However the shock is much better resolved in the solution presented here compared to the solutions provided to the workshop, and an additional reflection from the symmetry plane is visible. This is likely due largely to the finer grid and second-order spatial accuracy yielding lower numerical diffusion. The small area of reversed flow on the top wall of the nozzle also agrees well with the reference solution.

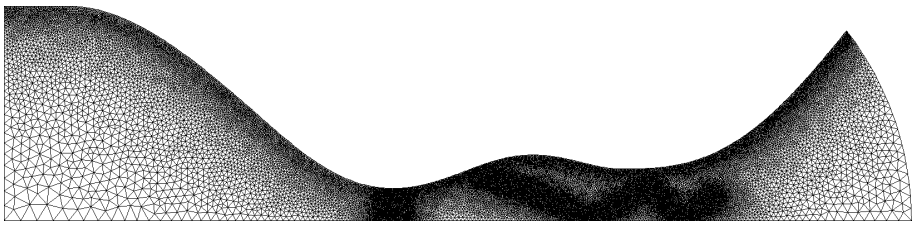


Figure 5.21: The grid used for the double-throated nozzle with $R_e = 1600$.

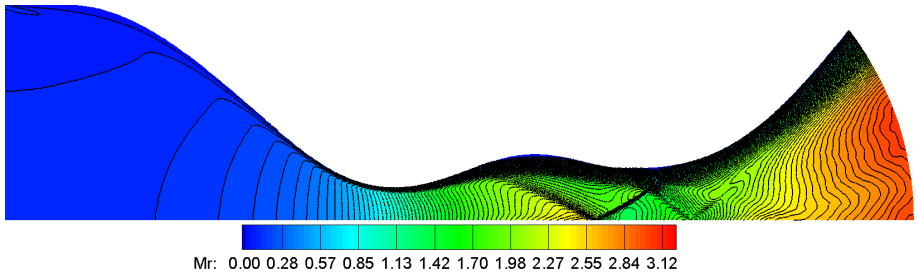



Figure 5.22: Mach number contours on the double-throated nozzle with $R_e = 1600$.

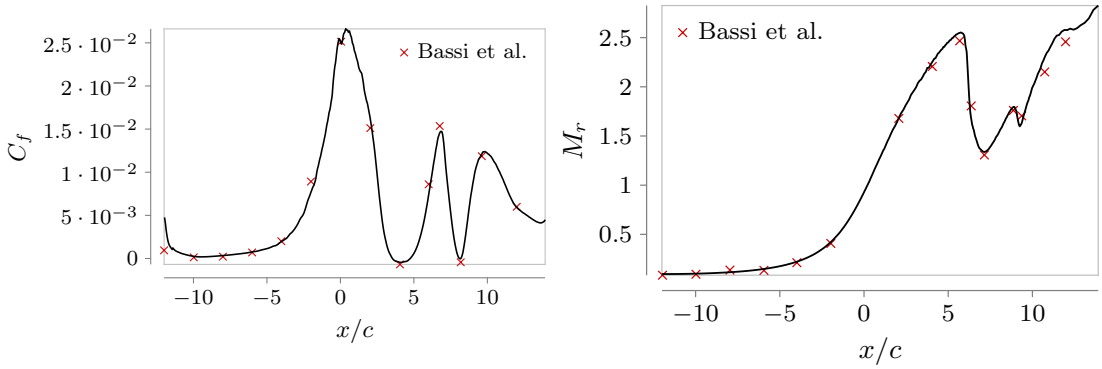



Figure 5.23: Surface plots for the double-throated nozzle at $R_e = 1600$, compared to the results given in [Bas+87]. Left: skin friction coefficient on the top wall. Right: Mach number along the centre line

# Chapter 6

# Development of the Unsteady Solver

## 6.1 Introduction

Many flows of interest in physics and engineering are inherently unstable. These flows can be broadly categorised into two distinct types. In the first type, the geometry of the flow domain is fixed, and instabilities arise directly from the flow physics. One example of this type of flow has already been encountered – the laminar vortex shedding that occurs behind a right circular cylinder at sufficiently high Reynolds numbers. Another common example of this type, one that is ubiquitous in real engineering flows, is the behaviour of turbulent boundary layers. The second type of unsteady flow includes flows where the geometry is not fixed. In these, instabilities arise both from the deformation of the flow domain and from the flow physics. Examples of these type of flows are those around rotating helicopter and compressor blades, variable geometry air intakes, and wings during the deployment or stowage of flaps and slots.

In this chapter, the solver is extended to allow accurate and efficient computation of both types of unsteady flow, beginning with the fixed geometry case. For the variable geometry case, the grid requirements outlined earlier present some difficulties. To overcome these, significant changes to the structure of the solver are reqired.

## 6.2 Consideration of the Physical and Computational Time Scales

For the steady solver, the time step was set as large as possible and allowed to vary across the grid. This is obviously advantageous for the efficiency of the steady solver, since the number of time steps to reach the steady state is minimised.

Neither of these techniques are possible for the unsteady case. Local time stepping would lead to the loss of time-accuracy, since different parts of the flow would simultaneously be in different stages of their transient evolution. This would result in the solution at some points being influenced by the solution at neighbouring points at later times and vice-versa. It is not possible to set the time step too large either, as this would mean that physical features at shorter time scales would not be correctly resolved. For laminar vortex shedding, for example, the time step must be significantly shorter than the period of the shedding to ensure correct resolution. The relationship between the physical time scales and the time step are best understood by defining the *physical* Courant number

$$\sigma_{\text{phys}} \coloneqq \frac{u_{\text{phys}} t_{\text{phys}}}{\ell_{\text{phys}}} \tag{6.1}$$

where $u_{\text{phys}}$, $t_{\text{phys}}$, and $\ell_{\text{phys}}$ are appropriately selected physical flow velocity, time, and length scales of the flow of interest. The physical Courant number can then compared to the *computational* Courant number discussed in Section 4.6.1. Roughly speaking, the computational Courant number must be significantly smaller than the physical Courant number for the correct resolution of the flow features of interest.

If the physical Courant number is much greater than unity, then the computational Courant number can also be much greater than unity, and the efficiency of the implicit scheme presented in this thesis will be retained. Fortunately the bulk of the flow cases of engineering interest fall into this category, such as the flows through compressors, turbines and pumps, combustion chambers, and helicopter rotors. If, on the other hand, the physical Courant number is of order unity then the computational Courant number is so severely limited that there is no advantage to the use of an implicit scheme, since the Courant-Friedrichs-Lewy (CFL) condition [Hir07, p. 191] is automatically satisfied in these cases. As a result an explicit scheme can be used with a time step just as large as would be possible with an implicit scheme. Since implicit schemes are much more computationally expensive per time step than explicit schemes, an explicit scheme would be much more efficient in these cases. Flow cases meeting this condition are less common, but do still include several of practical interest, such as in aeroacoustics as

well as with eddy-resolving approaches such as DNS and LES [Bla01, p. 212].

## 6.3 The Time Derivative and Higher-Order Temporal Accuracy

Previously, the temporal derivative has been discretised using the backward Euler method

$$\left.\frac{\partial\varphi}{\partial t}\right|_i^{n+1} = \frac{\varphi_i^{n+1} - \varphi_i^n}{\Delta t} + \mathcal{O}\left(\Delta t\right) \tag{6.2}$$

Clearly the truncation error is $\mathcal{O}\left(\Delta t\right)$, and hence the approximation is first-order accurate.

To obtain a second-order accurate approximation to the temporal derivative, begin with the Taylor series for $\varphi_i^n$ about $\varphi_i^{n+1}$

$$\begin{aligned} \varphi_i^n &= \varphi_i^{n+1} - \left.\frac{\partial\varphi}{\partial t}\right|_i^{n+1}\Delta t + \left.\frac{\partial^2\varphi}{\partial t^2}\right|_i^{n+1}\frac{\Delta t^2}{2!} + \mathcal{O}\left(\Delta t^3\right) \\ \implies \left.\frac{\partial\varphi}{\partial t}\right|_i^{n+1} &= \frac{\varphi_i^{n+1} - \varphi_i^n}{\Delta t} + \left.\frac{\partial^2\varphi}{\partial t^2}\right|_i^{n+1}\frac{\Delta t}{2!} + \mathcal{O}\left(\Delta t^2\right) \end{aligned} \tag{6.3}$$

An expression for the second derivative can be obtained by substituting $\varphi \mapsto \partial\varphi/\partial t$ into (6.2)

$$\left.\frac{\partial^2\varphi}{\partial t^2}\right|_i^{n+1} = \frac{\left.\frac{\partial\varphi}{\partial t}\right|_i^{n+1} - \left.\frac{\partial\varphi}{\partial t}\right|_i^{n}}{\Delta t} + \mathcal{O}\left(\Delta t\right) = \frac{\varphi_i^{n+1} - 2\varphi_i^n + \varphi_i^{n-1}}{\Delta t} + \mathcal{O}\left(\Delta t\right) \tag{6.4}$$

Finally, substituting this result into (6.3) yields

$$\left.\frac{\partial\varphi}{\partial t}\right|_i^{n+1} = \frac{3\varphi_i^{n+1} - 4\varphi_i^n + \varphi_i^{n-1}}{2\Delta t} + \mathcal{O}\left(\Delta t^2\right) \tag{6.5}$$

which is second-order accurate since it has truncation error $\mathcal{O}\left(\Delta t^2\right)$. It is sometimes referred to as the *backward* method [GW22] or the *three-level* scheme. Second-order temporal accuracy is therefore obtained by replacing the first-order accurate approximation to the temporal derivative (6.2) with the second-order accurate approximation (6.5).

Using the new discretisation of the temporal derivative adds one additional complication. During the first time step the approximation to the temporal derivative

is

$$\left.\frac{\partial \varphi}{\partial t}\right|_i^1 \approx \frac{3\varphi_i^1 - 4\varphi_i^0 + \varphi_i^{-1}}{2\Delta t} \tag{6.6}$$

This presents a problem: whilst $\varphi_i^0$ is specified by the initial conditions, $\varphi_i^{-1}$ is not available. Three possible solutions are

1. Assume that $\varphi_i^{-1} = \varphi_i^0$. This is the simplest procedure, but leads to an approximation to the temporal derivative that is very inaccurate.

2. Impose an initial condition on the time derivative $\left.\frac{\partial \varphi}{\partial t}\right|_i^0$. This is the most accurate procedure, but it is not clear how to select an appropriate value to impose in general.

3. Revert to the first-order temporal discretisation for the first time step only. This is straightforward to implement and is more accurate than the first option, though it is less accurate than the second.

The third option was selected as it offers better accuracy than the first without the requirement to impose initial conditions on the derivatives of the flow variables.

## 6.4 Dual Time Stepping

In this section, the linearisation method utilised by the steady solver is re-examined and modified to obtain a linearisation suitable for use with the unsteady solver. For this purpose, let $\boldsymbol{\varphi} \coloneqq \{\varphi_\mu\}$, $(\mu = 0, \ldots, q)$ denote the set of primary flow variables at some point. An arbitrary primary flow variable $\varphi_\mu$ is updated by solving a conservation equation of the form

$$\frac{\partial}{\partial t}\int_\Omega \varphi_\mu d\Omega = -\oint_{\partial\Omega} \boldsymbol{F}_S^\mu(\boldsymbol{\varphi}, \boldsymbol{n})\, dS + \int_\Omega \boldsymbol{F}_V^\mu(\boldsymbol{\varphi})\, d\Omega \tag{6.7}$$

where $\boldsymbol{F}_S^\mu : \mathbb{R}^{q+1} \times \mathbb{R}^2 \to \mathbb{R}$ is the surface flux function and $\boldsymbol{F}_V^\mu : \mathbb{R}^{q+1} \to \mathbb{R}$ is the volume source function. Applying the spatial discretisation discussed previously for an arbitrary grid location $i$ yields the differential equation

$$\Omega \left.\frac{d\varphi_\mu}{dt}\right|_i^{n+1} = -R_{\mu,i}\left(\boldsymbol{\varphi}^{n+1}\right) \tag{6.8}$$

where the *residual* $R_{\mu,i}$ is defined by

$$R_{\mu,i}\left(\boldsymbol{\varphi}^{n+1}\right) \coloneqq \sum_{e(i)} \bar{l}_e \left\{ \boldsymbol{F}^{\mu}_{S,e}\left(\boldsymbol{\varphi}^{n+1}, \boldsymbol{n}_e\right)\right\} - \Omega_i \boldsymbol{F}^{\mu}_{V,i}\left(\boldsymbol{\varphi}^{n+1}\right) \tag{6.9}$$

Note that this involves an abuse of notation since $\boldsymbol{\varphi}$ is now being used to refer to the sets of primary flow variables at all locations. However, this should not cause confusion and serves to significantly simplify the presentation.

The residual is a non-linear function of the primary flow variables, and must therefore be linearised before (6.8) can be solved. For the steady solver, this was done using Picard linearisation. In this linearisation method, the primary flow variable being solved for is treated implicitly, with the remaining flow variables being treated explicitly

$$R_{\mu,i}\left(\boldsymbol{\varphi}^{n+1}\right) \approx R_{\mu,i}\left(\bar{\boldsymbol{\varphi}}^{n+1}_{\mu}\right) \tag{6.10}$$

where

$$\bar{\boldsymbol{\varphi}}^{n+1}_{\mu} \coloneqq \left\{\varphi'_0, \ldots, \varphi'_{\mu-1}, \varphi^{n+1}_{\mu}, \varphi'_{\mu+1}, \ldots \varphi'_q\right\} \tag{6.11}$$

and $\varphi'_\nu$ denotes the latest known value of $\varphi_\nu$. As discussed previously the solver follows a segregated approach, meaning that the primary flow variables $\{\varphi_\mu\}$ are updated sequentially. Hence when (6.7) is solved for $\varphi^{n+1}_\mu$, the latest known values of the primary flow variables are $\varphi^{n+1}_0, \ldots \varphi^{n+1}_{\mu-1}, \varphi^{n}_{\mu}, \ldots \varphi^{n}_q$. Picard linearisation therefore consists of the approximation

$$R_{\mu,i}\left(\boldsymbol{\varphi}^{n+1}\right) \approx R_{\mu,i}\left(\varphi^{n+1}_0, \varphi^{n+1}_{\mu}, \varphi^{n}_{\mu+1}, \ldots \varphi^{n}_q\right) \tag{6.12}$$

Inserting this approximation into (6.8) yields

$$\Omega \frac{d\varphi_\mu}{dt}\bigg|_i^{n+1} + R_{\mu,i}\left(\bar{\boldsymbol{\varphi}}^{n+1}_{\mu}\right) = \hat{R}_{\mu,i}\left(\boldsymbol{\varphi}^{n+1}, \bar{\boldsymbol{\varphi}}^{n+1}_{\mu}\right) \approx 0 \tag{6.13}$$

where $\hat{R}_{\mu,i}$ is defined by

$$\hat{R}_{\mu,i}\left(\boldsymbol{\varphi}^{n+1}, \bar{\boldsymbol{\varphi}}^{n+1}_{\mu}\right) \coloneqq R_{\mu,i}\left(\bar{\boldsymbol{\varphi}}^{n+1}_{\mu}\right) - R_{\mu,i}\left(\boldsymbol{\varphi}^{n+1}\right) \tag{6.14}$$

This is called the *unsteady residual* for reasons that will soon become clear. The unsteady residual represents the error introduced into the scheme by the linearisation of the residual, and this will not be zero in general. As the steady state is approached, however, $\varphi^n_\nu \to \varphi^{n+1}_\nu$ for all $\nu \in \{0, \ldots, q+1\}$, and therefore $\hat{R}_{\mu,i}\left(\boldsymbol{\varphi}^{n+1}, \bar{\boldsymbol{\varphi}}^{n+1}_{\mu}\right) \to$

$\hat{R}_{\mu,i}\left(\boldsymbol{\varphi}^{n+1}, \boldsymbol{\varphi}^{n+1}\right) = 0$. Picard linearisation therefore has no influence on the accuracy of the steady solver, though it is presumed to affect the convergence to steady state. For the unsteady solver, however, the steady state is never reached and therefore the unsteady residual will remain large. Applying Picard linearisation in the same form is therefore likely to introduce unacceptably large errors into the unsteady solver. It is also probable that the maximum time step for which the solver converges to the correct solution would be severely limited.

To obtain more accurate and stable unsteady solutions, $\bar{\boldsymbol{\varphi}}_\mu^{n+1}$ must be corrected to solve

$$\hat{R}_{\mu,i}\left(\boldsymbol{\varphi}^{n+1}, \bar{\boldsymbol{\varphi}}_\mu^{n+1}\right) = 0 \tag{6.15}$$

This is a non-linear equation with the obvious solution $\bar{\boldsymbol{\varphi}}_\mu^{n+1} = \boldsymbol{\varphi}^{n+1}$, but $\boldsymbol{\varphi}^{n+1}$ is unknown *a priori*. Two common classes of methods for solving equations of this type are Newton-type methods and time stepping methods. A method of the latter type which has been employed with great success in computational fluid dynamics, and which is applied here, is the *dual time stepping* procedure of Jameson [Jam91]. The approximation to the vector of primary flow variables at the new time level $\bar{\boldsymbol{\varphi}}_\mu^{n+1}$ is corrected incrementally by stepping forwards in so-called *pseudotime*

$$\left.\frac{d\varphi_\mu}{d\tilde{t}}\right|_i^{(m+1)} + \hat{R}_{\mu,i}\left(\boldsymbol{\varphi}^{n+1}, \bar{\boldsymbol{\varphi}}_\mu^{(m+1)}\right) = 0 \tag{6.16}$$

where $(m)$ is the pseudotime level, $\tilde{t}$ is the pseudotime variable, and

$$\bar{\boldsymbol{\varphi}}_\mu^{(m+1)} = \left\{\varphi_0^{(m+1)}, \ldots, \varphi_\mu^{(m+1)}, \varphi_{\mu+1}^{(m)}, \ldots \varphi_q^{(m)}\right\} \tag{6.17}$$

is the latest approximation to $\boldsymbol{\varphi}^{n+1}$.

For clarity the time variable $t$ referring to actual time (as opposed to pseudotime) will be referred to as the *physical* time.

As with the time steps in the steady solver, the stepping in pseudotime is segregated. At the time level $n$, the procedure is therefore

1. $\bar{\boldsymbol{\varphi}}_0^{(0)}$ is set equal to $\boldsymbol{\varphi}^n$ and $m$ is initialised to zero.
2. For each $\mu = \{0, \ldots, q\}$, (6.16) is solved to calculate $\bar{\boldsymbol{\varphi}}_\mu^{(m+1)}$.
3. $\bar{\boldsymbol{\varphi}}_q^{(m+1)}$ is tested against some convergence criterion. If the criterion is not met, $m$ is incremented by one and the procedure continues at step 2. Otherwise $\boldsymbol{\varphi}^{n+1} = \bar{\boldsymbol{\varphi}}_q^{(m+1)}$ to the required accuracy and the physical time step is complete.

In effect, during each physical time step a steady problem is constructed by freezing the physical time derivative, with the flow variables at the new physical time level being approximated by the flow variables at the latest pseudotime level. Each of these steady problems is then solved in an analogous way to how the steady problems in the previous two chapters were solved, by stepping forwards in pseudotime. The major additional complication introduced in this chapter, that of temporal accuracy, is therefore not a concern with respect to pseudotime and it is completely sufficient to discretise the pseudo-temporal term using the first-order accurate backward Euler method. Hence the discretisation used is

$$\left.\frac{d\varphi_\mu}{d\tilde{t}}\right|_i^{(m+1)} \approx \frac{\varphi_{\mu,i}^{(m+1)} - \varphi_{\mu,i}^{(m)}}{\Delta\tilde{t}_i} \tag{6.18}$$

In particular, since accuracy in pseudotime does not need to be maintained there is no need for a uniform pseudotime step size across the grid. The method of local time stepping described in Section 4.6.2 can therefore be used to significantly accelerate convergence to the steady state. In addition, the method of under-relaxation discussed in Section 4.6.3 can be applied, leading to the approximation to the pseudotime derivative

$$\left.\frac{d\varphi_\mu}{d\tilde{t}}\right|_i^{(m+1)} \approx \frac{\varphi_{\mu,i}^{(m+1)} - \varphi_{\mu,i}^{(m)}}{\alpha_{\varphi_\mu}\Delta\tilde{t}_i} \tag{6.19}$$

where $\alpha_{\varphi_\mu}$ is the under-relaxation parameter corresponding to the flow variable $\varphi_\mu$.

A suitable convergence criterion can be obtained simply by modifying (4.3) to use the unsteady residual in place of the residual

$$\tilde{\Theta}_\varphi^{(m+1)} \coloneqq \log_{10}\left[\frac{\tilde{R}_\varphi^{(m+1)}}{\max\limits_{k\in[0,K]} \tilde{R}_\varphi^{(k)}}\right] \tag{6.20}$$

where $K \in \mathbb{N}^+$ is a user-defined parameter. For the test cases in the remainder of this thesis, $K$ was set to five.

Finally, note that (6.16) is not dimensionally homogeneous. The pseudotime derivative has dimensions $\varphi/T$, whilst the unsteady residual has dimensions $L^2\cdot\varphi/T$. Varying the grid size will therefore alter the balance between the respective terms, meaning that the selection of the optimum pseudotime step will likely be unduly affected by the grid size. A more robust scheme can be obtained by scaling the pseudotime derivative by a factor proportional to the square of the grid size. The obvious candidate is the area

of the control volume associated with the grid point $i$. Equation (6.16) is therefore replaced by

$$\Omega_i \frac{d\varphi_\mu}{d\tilde{t}}\bigg|_i^{(m+1)} + \hat{R}_{\mu,i}\left(\boldsymbol{\varphi}^{n+1}, \bar{\boldsymbol{\varphi}}_\mu^{(m+1)}\right) = 0 \tag{6.21}$$

## 6.5 Variable Geometry and Moving Grids

As previously stated a large number of the flows of interest in engineering involve not only unsteadiness deriving from the flow physics, but also unsteadiness resulting from variable geometry. There are several methods for modelling these flow cases. The method selected for the particular solver developed in this thesis is presented in Section 6.7. In preparation, several issues that complicate the requisite implementation of moving grids into the solver must be discussed. For this purpose, and looking ahead to Section 6.7, the following assumptions are placed on the grid movement

1. The grid movement is differentiable with respect to time.

2. Grid elements such as edges and cells may undergo rigid motions due to the movement, but will not be deformed. Note that since the control volumes and stencils used by the solver are unions of these, this condition implies that the control volumes and stencils are not deformed by the grid movement.

These conditions are equivalent to the requirement that the allowed motions are combinations of smooth translations and rotations in the plane.

The requirement that the grid movement is differentiable with respect to time means that geometrical quantities such as edge unit normals and cell circumcentres can be discretised in the same manner as the flow variables. Each geometrical quantity then becomes a discrete function of the physical time step. Hence, for example, the unit normal to edge $i$ at physical time level $n+1$ is written $\boldsymbol{n}_i^{n+1}$.

For the present work only pre-determined ("forced") grid movement is considered. The grid is therefore updated at the beginning of each time step, and does not change during the pseudotime stepping. After the solution at time level $n$ has been determined, the grid is advanced to time level $n+1$ and the pseudotime stepping to advance the solution to time level $n+1$ begins.

The variation of the geometrical quantities with respect to time has a particular implication for the status of the edge centre normal momentum as a primary flow variable. Since $\boldsymbol{n}_i^n \neq \boldsymbol{n}_i^{n+1}$ in general, the edge centre normal momentum is effectively a different primary flow variable at each time step. It is therefore no longer reasonable

to use discretisations such as

$$\left.\frac{\partial\left(\boldsymbol{m}\cdot\boldsymbol{n}\right)}{\partial t}\right|_{i}^{n+1} \approx \frac{\left(\boldsymbol{m}\cdot\boldsymbol{n}\right)_{i}^{n+1}-\left(\boldsymbol{m}\cdot\boldsymbol{n}\right)_{i}^{n}}{\Delta t} \tag{6.22}$$

This issue is solved by using the momentum reconstructions discussed in Section 3.9 to reconstruct the edge centre tangential momenta at the required time level. When solving at time level $n+1$, the reconstructed normal and tangential momenta for a given edge $i$ at a given time level $p$ are used to calculate the normal momentum relative to the new grid geometry according to

$$\boldsymbol{m}_{i}^{p}\cdot\boldsymbol{n}_{i}^{n+1}=\left(\boldsymbol{m}\cdot\boldsymbol{n}\right)_{i}^{p}\left(\boldsymbol{n}^{p}\cdot\boldsymbol{n}^{n+1}\right)+\left(\boldsymbol{m}\cdot\boldsymbol{t}\right)_{i}^{p}\left(\boldsymbol{t}^{p}\cdot\boldsymbol{n}^{n+1}\right) \tag{6.23}$$

Note that this transformation is only necessary if the grid containing edge $i$ has been rotated, since translations do not affect the edge centre unit normals.

The approximations to the temporal derivative of the normal momentum are then

$$\left.\frac{\partial\left(\boldsymbol{m}\cdot\boldsymbol{n}\right)}{\partial t}\right|_{i}^{n+1} \approx \frac{\left(\boldsymbol{m}\cdot\boldsymbol{n}\right)_{i}^{n+1}-\boldsymbol{m}_{i}^{n}\cdot\boldsymbol{n}_{i}^{n+1}}{\Delta t} \tag{6.24}$$

for first-order temporal accuracy, and

$$\left.\frac{\partial\left(\boldsymbol{m}\cdot\boldsymbol{n}\right)}{\partial t}\right|_{i}^{n+1} \approx \frac{3\left(\boldsymbol{m}\cdot\boldsymbol{n}\right)_{i}^{n+1}-4\boldsymbol{m}_{i}^{n}\cdot\boldsymbol{n}_{i}^{n+1}+\boldsymbol{m}_{i}^{n-1}\cdot\boldsymbol{n}_{i}^{n+1}}{2\Delta t} \tag{6.25}$$

for second-order temporal accuracy.

In Sections 3.8, 3.9, and 5.4, methods for reconstructing the scalar and vector flow variables and the viscous stress tensor were discussed. One of the stated advantages of the selected methods is that the relatively expensive (pseudo)inversion of the reconstruction systems only needs to be performed once, with the reconstruction coefficients being stored and re-used throughout the time stepping. For moving grids this is no longer possible, as the reconstructions do not remain valid once the geometric quantities have changed. However, the limitation of the grid movement to rigid motions means the reconstruction coefficients do not need to be recalculated from scratch because much more cheaply computed transformations can be derived. These are discussed in Section 6.7.2.

### 6.5.1 Modification of the Conservation Principle for Moving Grids

When a control volume is not at rest with respect to the selected coordinate system, the net rate of flow of $\varphi$ into the control volume through an elemental surface patch is not proportional to the projection of the flow velocity normal to that patch. Instead, it is proportional to the normal projection of the *difference* between the flow velocity and the velocity of the surface patch. Hence if $\boldsymbol{v}$ denotes the velocity of a given elemental surface patch of the control volume then the correct conservation equation for a conserved quantity $\varphi$ is

$$\frac{\partial}{\partial t}\int_{\Omega}\rho\varphi d\Omega + \oint_{\partial\Omega}\rho\varphi\left[(\boldsymbol{u}-\boldsymbol{v})\cdot\boldsymbol{n}\right]dS = 0 \tag{6.26}$$

## 6.6 The Discretised Governing Equations of Unsteady Viscous Flow

Having discussed the grid movement and modification of the conserved variable fluxes in the previous section, the three-level discretisation of the physical time derivative in Section 6.3, and the dual time stepping method in Section 6.4, the complete discretised governing equations for unsteady viscous flow can now be presented.

### 6.6.1 The Continuity Equation

$$\begin{aligned}\Omega_i\left[\frac{1}{\Delta\tilde{t}_i}+\frac{3}{2\Delta t}\right]\rho_{c(i)}^{(m+1)} + \sum_{e(i)}\left\{\bar{l}_e\left(\Delta\boldsymbol{u}\cdot\boldsymbol{n}\right)_e^{(m')}\rho_e^{(m+1)}\right\} \\ = \frac{\Omega_i\rho_{c(i)}^{(m)}}{\Delta\tilde{t}_i} + \frac{2\Omega_i\rho_{c(i)}^{n}}{\Delta t} - \frac{\Omega_i\rho_{c(i)}^{n-1}}{2\Delta t}\end{aligned} \tag{6.27}$$

where the superscript $(m')$ denotes the latest known value and

$$\Delta\boldsymbol{u}^{(m')} \coloneqq \boldsymbol{u}^{(m')} - \boldsymbol{v}^{n+1} \tag{6.28}$$

is the convecting normal flow velocity as discussed in the previous subsection. Note that the unit normals and element positions are now functions of time.

### 6.6.2 The Normal Momentum Equation

$$\begin{aligned}
&\Omega_i\left[\frac{1}{\Delta\tilde{t}_i}+\frac{3}{2\Delta t}\right](\boldsymbol{m}\cdot\boldsymbol{n})_{c(i)}^{(m*)}+\sum_{e(i)}\left\{\bar{l}_e\left(\Delta\boldsymbol{u}\cdot\boldsymbol{n}\right)_e^{(m')}(\boldsymbol{m}\cdot\boldsymbol{n})_e^{(m*)}\right\}\\
&\qquad\qquad-\sum_{e(i)}\left\{\bar{l}_e\mu_e^{(m)}\sum_{j(e)}\left(\frac{m_j^{(m*)}}{\rho_j(m')}\left(\boldsymbol{\zeta}_j\cdot\boldsymbol{n}_i\right)^{n+1}\right)\right\}\\
&-\sum_{e(i)}\left\{\bar{l}_e\mu_e^{(m)}\frac{\left(\boldsymbol{\pi}_e^{n+1}\cdot\boldsymbol{n}_i\right)^{n+1}}{\Omega_e}\sum_{d(e)}\left(\frac{\bar{l}_d}{\rho_d(m')}(\boldsymbol{m}\cdot\boldsymbol{n})_d^{(m*)}\right)\right\}\\
&=\frac{\Omega_i\left(p_\ell-p_r\right)^{(m)}}{\|\boldsymbol{x}_\ell-\boldsymbol{x}_r\|_2^{n+1}}+\frac{\Omega_i\left(\boldsymbol{m}\cdot\boldsymbol{n}\right)_{c(i)}^{(m)}}{\Delta\tilde{t}_i}+\frac{2\Omega_i\left(\boldsymbol{m}^n\cdot\boldsymbol{n}^{n+1}\right)_{c(i)}}{\Delta t}\\
&\qquad\qquad-\frac{\Omega_i\left(\boldsymbol{m}^{n-1}\cdot\boldsymbol{n}^{n+1}\right)_{c(i)}}{2\Delta t}
\end{aligned}\tag{6.29}$$

where $(m*)$ denotes the prediction to the flow variable at pseudotime step $(m+1)$.

### 6.6.3 The Pressure and Momentum Corrections

For the unsteady solver, the pressure correction becomes the change in the pressure from the current pseudotime level to the next pseudotime level. The pressure is therefore written

$$p_i^{(m+1)}\coloneqq p_i^{(m)}+\delta p_i^{(m+1)}\tag{6.30}$$

The definition of the momentum correction changes accordingly

$$\delta\boldsymbol{m}_i^{(m+1)}\coloneqq\boldsymbol{m}_i^{(m+1)}-\boldsymbol{m}_i^{(m*)}=-\Delta\tilde{t}_i\nabla\delta p\big|_i^{(m+1)}\tag{6.31}$$

Note that the pseudotime step has replaced the physical time step in this definition.

### 6.6.4 The Pressure Correction Equation

$$\begin{aligned}
&\Omega M_r^2\left[\frac{1}{\Delta\tilde{t}_i}+\frac{3}{2\Delta t}\right]\delta p^{(m+1)}+\frac{\Omega_i M_r^2}{2\Delta t}\left[3p_i^{(m)}-4p_i^n+\frac{1}{2}p_i^{n-1}\right]\\
&+\frac{\Omega_i M_r^2\left(\gamma-1\right)}{4\Delta t}\left[3\frac{(\boldsymbol{m}\cdot\boldsymbol{m})^{(m)}}{\rho^{(m+1)}}-4\left(\frac{\boldsymbol{m}\cdot\boldsymbol{m}}{\rho}\right)^n+\frac{1}{2}\left(\frac{\boldsymbol{m}\cdot\boldsymbol{m}}{\rho}\right)^{n-1}\right]\\
&+\frac{3\Omega_i M_r^2\Delta\tilde{t}_i\left(1-\gamma\right)}{2\Delta t}\boldsymbol{u}^{(m')}\cdot\nabla\delta p_i^{(m+1)}
\end{aligned}$$

$$
\begin{aligned}
&+\sum_{e(i)}\left\{\bar{l}_e\left(\Delta\boldsymbol{u}\cdot\boldsymbol{n}\right)_e^{(m')}\left[1+\gamma M_r^2\left(p^{(m)}+\delta p^{(m+1)}\right)+\frac{\gamma-1}{2}M_r^2\cdot\frac{\left(\boldsymbol{m}\cdot\boldsymbol{m}\right)^{(m)}}{\rho^{(m+1)}}\right]_e\right\}\\
&+\left(1-\gamma\right)M_r^2\sum_{e(i)}\left\{\bar{l}_e\left(\Delta\boldsymbol{u}\cdot\boldsymbol{n}\right)_e^{(m')}\left(\frac{\Delta t}{\rho^{n+1}}\boldsymbol{u}^{(m*)}\cdot\nabla\delta p^{(m+1)}\right)_e\right\}\\
&+\sum_{e(i)}\left\{\bar{l}_e\left(\frac{\Delta\tilde{t}_e}{\rho_e^{(m+1)}}\cdot\frac{\delta p_{c_\ell}^{(m+1)}-\delta p_{c_r}^{(m+1)}}{\left\|\boldsymbol{x}_{c_\ell}-\boldsymbol{x}_{c_r}\right\|_2^{n+1}}\right)\left[1+\gamma M_r^2p^{(m)}+\frac{\gamma-1}{2}M_r^2\cdot\frac{\left(\boldsymbol{m}\cdot\boldsymbol{m}\right)^{(m)}}{\rho^{(m+1)}}\right]_e\right\}\\
&=\left(\gamma-1\right)M_r^2\sum_{e(i)}\left\{\bar{l}_e\left(\boldsymbol{\eta}\cdot\boldsymbol{u}\right)_e^{(m*)}\right\}\\
&+\left(\gamma-1\right)M_r^2\sum_{e(i)}\left\{\bar{l}_e\mu_e^{(m)}\sum_{j(e')}\left[\left(\boldsymbol{\zeta}_j\cdot\boldsymbol{u}_e^{(m*)}\right)\frac{\Delta t_j\left(\delta p_{j,\ell}-\delta p_{j,r}\right)^{(m+1)}}{\rho_j^{(m')}\left\|\boldsymbol{x}_{j,\ell}-\boldsymbol{x}_{j,r}\right\|_2^{n+1}}\right]\right\}\\
&+\left(\gamma-1\right)M_r^2\sum_{e(i)}\left\{\frac{\bar{l}_e\mu_e^{(m)}\left(\boldsymbol{\pi}_e\cdot\boldsymbol{u}_e^{(m*)}\right)}{\Omega_e}\sum_{d(e)}\left[\bar{l}_d\frac{\Delta t_d\left(\delta p_{d,\ell}-\delta p_{d,r}\right)^{(m+1)}}{\rho_d^{(m')}\left\|\boldsymbol{x}_{d,\ell}-\boldsymbol{x}_{d,r}\right\|_2^{n+1}}\right]\right\}\\
&+\left(\gamma-1\right)M_r^2\sum_{e(i)}\left\{-\bar{l}_e\frac{\Delta t_e}{\rho_e^{(m')}}\boldsymbol{\eta}_e^{(m*)}\cdot\nabla\delta p\big|_e^{(m+1)}\right\}\\
&+\frac{\gamma M_r^2}{R_eP_r}\sum_{e(i)}\left\{\frac{\bar{l}_ek_e^{(m)}}{\left\|\boldsymbol{x}_{e,\ell}-\boldsymbol{x}_{e,r}\right\|_2^{n+1}}\left[\left(\frac{\delta p}{\rho}\right)_{e,\ell}^{(m+1)}-\left(\frac{\delta p}{\rho}\right)_{e,r}^{(m+1)}\right]\right\}\\
&+\frac{1}{R_eP_r}\sum_{e(i)}\left\{\frac{\bar{l}_ek_e^{(m)}}{\left\|\boldsymbol{x}_{e,\ell}-\boldsymbol{x}_{e,r}\right\|_2^{n+1}}\left[\left(\frac{1+\gamma M_r^2p^{(m)}}{\rho^{(m+1)}}\right)_{e,\ell}-\left(\frac{1+\gamma M_r^2p^{(m)}}{\rho^{(m+1)}}\right)_{e,r}\right]\right\}
\end{aligned}
\tag{6.32}
$$

## 6.7 The Chimera Method

As previously discussed, the requirement of orthogonality extends the computational effort required for generation and optimisation of the grid. Simulating flow on variable geometries would require remeshing at each time step, multiplying this increase in grid preparation time by the number of time steps. The overall algorithm would then be very inefficient, severely limiting the complexity of the geometry, the grid fineness, or the time over which the unsteady solution is simulated. To overcome this, the *chimera* method is used (see, for example, [TJS03] or [ZNG08]).

The chimera method (also known as the *overset grids* method) replaces the single grid representing the flow domain by multiple independent overlapping grids. Hole-cutting algorithms are used to 'disable' parts of grids overlapping boundary holes or

other grids, and interpolation is used to transfer flow information between the various grids. Originally the chimera method was used in the context of the generation of structured grids for stationary complex geometries. In that role the method allows complex geometries to be broken down into multiple simple parts, which are then amenable to otherwise prohibitively difficult structured grid generation.

The chimera method can be generalised by allowing the individual grids to undergo rigid motions with respect to one another, and therefore the flow boundaries contained in the grids to undergo rigid motions with respect to one another. This generalisation allows the bulk of flow cases involving variable geometry to be simulated. It does not allow for the simulation of those cases involving the non-rigid deformation of flow boundaries, such as the flight of birds, ornithopters, or ram-air parachutes, but these cases are of little relevance to compressible CFD.

### 6.7.1 Basic Structure

Before describing the structure of the chimera method, several terms need to be defined. Each geometry consists of one *outer* boundary, which has the flow domain contained within its interior, and zero or more *inner* boundaries, which have the flow domain contained within their exterior. For example, the NACA 0012 geometry considered in the previous chapters has the farfield boundary as its outer boundary and the aerofoil wall as its sole inner boundary.

A set of chimera grids (hereafter referred to as a *grid set*) consists of a *base grid*, which fills the interior of the outer boundary (including the interior of any inner boundaries), and one or more *component grids*, which lay on top of the base grid and are contained within its outer boundary at all times. The base grid and component grids are individually referred to as *grids*. Each boundary is contained in exactly one grid. Typically each inner boundary is contained in its own component grid, but for the purposes of the present work it is only necessary that inner boundaries that move relative to one another are placed on separate component grids. The base grid remains stationary, whereas the component grids are either stationary or undergo rigid motions with respect to the base grid and one another.

Attached to each node, edge, and cell is a tag recording the status of that element as *active*, *interpolation*, or *disabled*. At a point in the time stepping, grid elements that are covered by another grid are *disabled* and do not participate in the solution process. Grid elements that are not covered by another grid are *active* and participate in the solution process as normal. Separating the active elements from the disabled elements

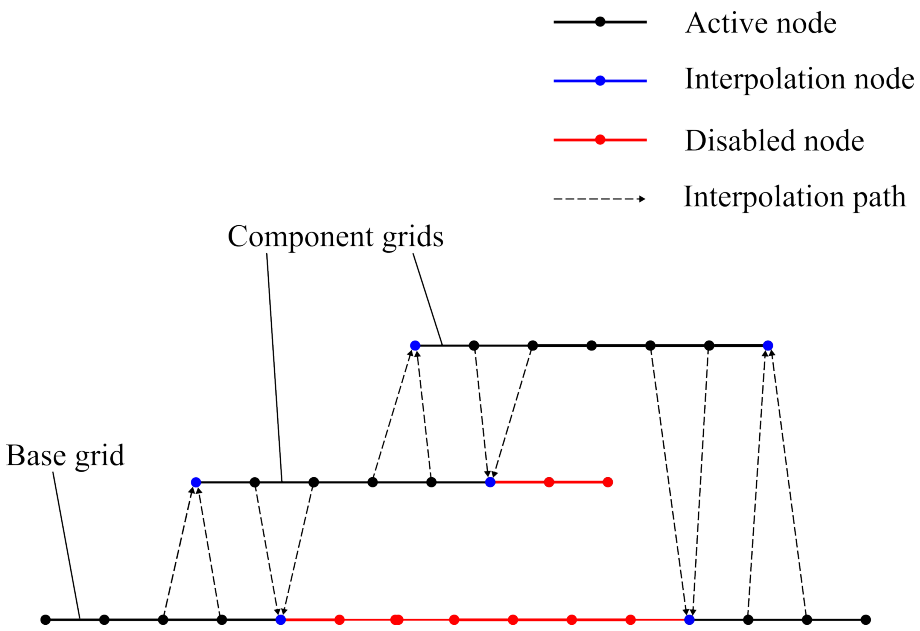



Figure 6.1: Relationship between the base grid and two component grids in a one dimensional chimera grid set.

are rows of *interpolation* elements at which the flow variables and their gradients are determined by interpolation from other grids in the set. Through this interpolation, the solution on the various grids in the set are coupled. This basic structure is shown for the one dimensional case in Figure 6.1.

Incorporating the chimera method into the solver requires the addition of several new algorithms, as well as the modification of the scalar, vector, and viscous stress reconstruction methods, as well as changes to the overall solver algorithm. Each of these ingredients is discussed in the remaining parts of this section.

### 6.7.2 The Element-Finding Algorithm

A key component of the chimera method is the element-finding algorithm, which finds the elements of a given grid (the *test grid*) in the neighbourhood of a specified point (the *test point*). This is used to determine whether a given grid element is covered by another grid, and to determine the correct grid element from which to interpolate flow variables. The efficiency of the element-finding algorithm is of moderately importance, since it is called at least once per grid node per time step. It is an order of magnitude less important than the efficiency of the solver algorithm itself, though, since it is not called during the pseudotime stepping within each time step.

The element-finding algorithm is based on a nearest-node search. If the cell containing a test point is required instead, this can be accomplished simply by testing the cells in some neighbourhood of the nearest node. If the grid is assumed to be of good quality (no cells of high aspect ratio), this neighbourhood can simply consist of the cells having the nearest node as a vertex.

To begin, each of the grids is enclosed by an axis-aligned box (the *grid box*). Checking whether a test point lies within an axis-aligned box can be done in at most four floating point comparisons, allowing most test points that lie outside the test grid to be rejected at little cost. Even if the test point lies within the grid box, it is still possible that the test point does not lie within the test grid, and this must be tested if necessary. If a test grid cell containing the test point is being sought, this is not an issue since the test point will be found not to lie in a cell in the neighbourhood of the nearest node.

Having filtered out the majority of test points that do not lie in the test grid, the next step is to actually search for the nearest node. This is done by organising the nodes of the test grid into a *quadtree*, a hierarchical structure based on a recursive division of the grid box [Ber+00]. Using this structure there is an efficient algorithm for finding the nearest node to a test point [Sam82]. Additionally, by giving each grid in the set its own quadtree, the movement of the component grids can be easily handled without expensive updates of the structure. This is done by endowing each grid with a local coordinate system in which the grid is at rest, storing the current translation and rotation of the component grids relative to the base grid, and transforming the test point to local coordinates before searching for the nearest node.

### 6.7.3 The Boundary Hole-Cutting Algorithm

The boundary hole-cutting algorithm identifies and disables those grid elements that lie inside an inner boundary contained in one of the other grids in the set. A boundary hole-cutting algorithm would seem to be unnecessary, since each inner boundary is contained in a component grid, and therefore each boundary hole will be contained in a component grid hole. It is necessary to identify the elements disabled because they lie within a boundary hole, however, because they can never be reactivated for use as interpolation source or target elements, whereas those elements disabled because they are covered by an element belonging to another grid can be. Little needs to be said about the boundary hole-cutting algorithm itself. Since the boundary curves of interest are discrete, all boundary holes are polygons. Testing whether a point lies

within a polygon is well covered in the computational geometry literature (see [ORo98] for example). In the same way that grid boxes are used to cheaply reject most test points not lying within a test grid, each inner boundary is contained within an axis-aligned box (the *boundary box*) which allows most test points not lying within an inner boundary to be rejected in less than four floating point comparisons. The complete boundary hole-cutting algorithm is shown in Algorithm 6.1 and Algorithm 6.2.

```
1  Function IsNodeInBoundaryHole(x_i, C)
       input  : test node position x_i, grid C containing the test node
       output: whether x_i is inside a boundary hole
2      foreach component_grid ≠ C do
3          if x_i is not inside the grid box then go to next component_grid
4          foreach inner_boundary belonging to component_grid do
5              if x_i is not inside the boundary box then go to next inner_boundary
6              if IsPointInsideBoundary(x_i, inner_boundary) then return true
7          end
8      end
9      return false;
```

**Algorithm 6.1:** Function for testing whether a given grid node lies within a boundary hole.

```
1  Subroutine CutBoundaryHoles()
2      foreach grid belonging to the grid set do
3          foreach node in grid do
4              if IsNodeInBoundaryHole(node,grid) then
5                  node(← disabled)
6                  foreach edge attached to node do edge ← disabled
7                  foreach cell attached to node do cell ← disabled
8              end
9          end
10     end
```

**Algorithm 6.2:** The boundary hole-cutting algorithm.

Several methods exist in the literature for testing whether a given point is inside a given polygon. The method actually used in this work was the ray-casting algorithm [ORo98, p. 239].

Nodes that are identified as being inside a boundary hole are disabled and cannot be used as interpolation source nodes. However, they may be reactivated later on in the time stepping if they move back into the flow domain. Effectively, disabled nodes are temporarily removed from the grid. If a given node is removed, then edges that have

that node as an end point and cells that have that node as a vertex are no longer valid grid elements. Therefore, the boundary hole-cutting algorithm concludes by disabling edges and cells that are attached to any of the disabled nodes.

### 6.7.4 The Component Grid Hole-Cutting Algorithm

The component grid hole-cutting algorithm is somewhat more involved than the boundary hole-cutting algorithm since an element in one grid that is covered by an element in another grid is not necessarily disabled, because one of the overlapping grid elements must remain active. Some method for determining a precedence in the grid elements is therefore required, to determine which element is disabled and which is kept active.

To begin, the requirement that base grid elements have a lower precedence than all component grid elements is imposed. If this were not the case, component grids would automatically be cut back from the target size that was selected by the user during mesh generation. In deciding precedence between component grid elements, the key requirement is that no component grid be cut back too close to an inner boundary. To achieve this, a variant of Dijkstra's algorithm (see [MS08, p. 196]) is used to assign a weight to each node in a component grid, defined to be the minimum number of edges connecting the node to an inner boundary in the component grid. Weights of edges and cells are defined to be the minimum end node or vertex weight respectively. Precedence between component grid elements is given to the element that has the lowest "distance" to an inner boundary in the above sense. To break ties unambiguously the component grids are also assigned arbitrary indices, the element belonging to the component grid with the higher index being given precedence.

Referring to Figure 6.1, it is clear that it is not sufficient to simply disable grid nodes that are covered by a cell on another grid with a higher precedence. Instead, there must be some degree of overlap between the first few layers of active cells on each grid. To achieve this, a given number of rows of cells are simply reactivated after cutting the component grid holes.

The complete component grid hole-cutting algorithm is shown in Algorithm 6.3, Algorithm 6.4, and Algorithm 6.5.

### 6.7.5 Scalar, Vector, and Viscous Stress Reconstruction Procedures

The scalar, vector, and viscous stress reconstruction procedures discussed in Section 3.8, Section 3.9, and Section 5.4 respectively must now be reconsidered.

The first issue that needs addressing is the selection of the node or cell at which each

```
1  Function IsComponentGridNodeCovered(test_node, C)
       input  : test_node, component grid C containing the test node
       output: whether test_node is inside a component grid hole
2      foreach component_grid ≠ C do
3          if test_node is not inside the grid box then go to next component_grid
4          // Search component_grid for a covering cell
5          if FindCoveringCell(test_node) is successful then
6              if tag(covering_cell) = disabled then go to next component_grid
7              if precedence(covering_cell) < precedence(test_node) then go to
                next component_grid
8              if precedence(covering_cell) = precedence(test_node) and
                index(C) > index(component_grid) then go to next
                component_grid
9              return true
10         end
11     end
12     return false
```

**Algorithm 6.3:** Function for testing whether a given component grid node lies within a component grid hole and should therefore be *disabled*.

```
1  Function IsBaseGridNodeCovered(test_node)
       input  : test_node
       output: whether test_node is inside a component grid hole
2      foreach component_grid do
3          if test_node is not inside the grid box then go to next component_grid
4          // Search component_grid for a covering cell
5          if FindCoveringCell(test_node) is successful then
6              if tag(covering_cell) = disabled then go to next component_grid
7              return true
8          end
9      end
10     return false
```

**Algorithm 6.4:** Function for testing whether a given base grid node lies within a component grid hole and should therefore be *disabled*.

```
1  Subroutine CutComponentGridHoles()
2      foreach component_grid belonging to the grid set do
3          foreach node in component_grid do
4              if IsComponentGridNodeCovered(node,component_grid) then
5                  node ← disabled
6                  foreach edge attached to node do edge ← disabled
7                  foreach cell attached to node do cell ← disabled
8              end
9          end
10         Set first m layers of disabled nodes, edges, and cells to active
11         Set next n layers of disabled nodes, edges, and cells to interpolation
12     end
13     foreach node in base_grid do
14         if IsBaseGridNodeCovered(node) then
15             node ← disabled
16             foreach edge attached to node do edge ← disabled
17             foreach cell attached to node do cell ← disabled
18         end
19         Set first m layers of disabled nodes, edges, and cells to active
20         Set next n layers of disabled nodes, edges, and cells to interpolation
21     end
```

**Algorithm 6.5:** The component grid hole-cutting algorithm .

reconstruction is based (hereafter referred to as the *base node*). In piecewise linear node-based scalar reconstruction, for example, the scalar gradient is reconstructed at the node immediately upwind of the target edge. When the chimera method is employed, however, the base node might be an interpolation node with attached *disabled* cells. In this case the reconstruction will be incorrect because the flow variables at the (circum)centres of *disabled* grid elements are not updated (or, if the *disabled* grid elements are simply left out of the reconstruction stencil, the stencil may be rank deficient and require extending beyond its usual size). Instead, the element-finding algorithm presented in Section 6.7.2 is used to select an appropriate *active* node in one of the grids overlapping the upwind node at which to base the reconstruction instead. This node will be referred to as the *source node*. The same is done for the cell-based reconstructions and vector reconstructions. For the piecewise linear vector reconstructions, an approximation to the momentum divergence is evaluated by selecting a suitable source cell and applying (3.62). When combining this with the chimera method, it was found that the most robust scheme was obtained by using a source cell from the same grid as the source node. Choosing a cell from the same grid as the base node lead to a scheme that was unacceptably grid dependent, with unpredictable solution break-up. No change is required to the base node selection for the viscous stress reconstruction procedure becaue it is only required at the end nodes of edges in a control volume for the normal momentum equation, rather than at the nodes upwind of those edges. Therefore, viscous stress reconstruction base nodes never have *disabled* edges in their stencils, provided that at least two layers of interpolation elements are added during Algorithm 6.5. Note that if a component grid is moving during the simulation, then the selection of an appropriate source node or cell will need to be repeated at the beginning of each time step. If the grids are assumed to be moving sufficiently slowly, the update only needs to be performed every $m$ iterations, but this is more useful in the case of an explicit solver.

The other required modification to the reconstruction procedures is the update of the reconstruction coefficients following grid movement. As previously mentioned, only rigid motions of the component grids is considered in the current work. The gradient reconstructions are manifestly invariant under translations, since only relative displacements within the same component grid are considered during the calculation of the reconstruction coefficients, and hence only rotations need to be considered. For this purpose, attach a local Cartesian coordinate system $(x', y')$ to each grid. Each local coordinate system remains stationary relative to the grid to which it is attached. The base grid is stationary, and hence its local coordinate system can be

assumed to coincide with the global Cartesian coordinate system $(x, y)$. Consider a scalar reconstruction based at an element belonging to a component grid. When the component grid is rotated, its local coordinate system rotates along with it. The scalar gradient reconstruction remains valid after the rotation, but its components are calculated with respect to the rotated local coordinate system. In order to obtain the components of the reconstructed scalar gradient in the global coordinate system, the local coordinates must be rotated in the opposite direction to the rotation of the component grid. Therefore, the coefficients of the scalar gradient reconstructions (3.51) and (3.55) are updated according to [AWH13]

$$\begin{Bmatrix} a \\ b \\ c \end{Bmatrix} \mapsto \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \begin{Bmatrix} a \\ b \\ c \end{Bmatrix} \tag{6.33}$$

The piecewise constant momentum reconstruction is updated in an almost identical way. The reconstruction coefficient vectors of (3.57) are updated according to

$$\begin{Bmatrix} \boldsymbol{\varepsilon}_e \cdot \boldsymbol{i} \\ \boldsymbol{\varepsilon}_e \cdot \boldsymbol{j} \end{Bmatrix} \mapsto \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{Bmatrix} \boldsymbol{\varepsilon}_e \cdot \boldsymbol{i} \\ \boldsymbol{\varepsilon}_e \cdot \boldsymbol{j} \end{Bmatrix} \tag{6.34}$$

The update of the piecewise linear momentum reconstructions is more complex than for the scalar gradient reconstructions since a vector field is being transformed rather than a scalar field. Not only do the gradients of the components of the vector need to be transformed, but the components themselves need to be updated. The transformation matrix in the above must therefore be applied twice, which results in the reconstruction coefficients being transformed according to

$$\begin{aligned} \begin{Bmatrix} a_x \\ a_y \end{Bmatrix} &\mapsto \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{Bmatrix} a_x \\ a_y \end{Bmatrix} \\ \begin{Bmatrix} b_x \\ b_y \\ c_x \\ c_y \end{Bmatrix} &\mapsto \begin{bmatrix} \cos^2\theta & -\sin\theta\cos\theta & -\sin\theta\cos\theta & \sin^2\theta \\ \sin\theta\cos\theta & \cos^2\theta & -\sin^2\theta & -\sin\theta\cos\theta \\ \sin\theta\cos\theta & -\sin^2\theta & \cos^2\theta & -\sin\theta\cos\theta \\ \sin^2\theta & \sin\theta\cos\theta & \sin\theta\cos\theta & \cos^2\theta \end{bmatrix} \begin{Bmatrix} b_x \\ b_y \\ c_x \\ c_y \end{Bmatrix} \end{aligned} \tag{6.35}$$

Finally, the viscous stress reconstruction systems need to be updated. Since the viscous stress is a tensor, coordinate transformations of the type discussed above also apply to the viscous stress reconstruction systems. However, unlike for the scalar and

vector reconstruction systems, the initial setup of the viscous stress reconstruction systems is computationally cheap because there is no need to invert reconstruction matrices or build stencils – the reconstruction coefficients are calculated directly from the momentum reconstruction coefficients. The viscous stress reconstuction systems can therefore be updated simply by calculating the reconstruction coefficients from scratch after the momentum reconstruction systems have been updated, without a loss of efficiency.

### 6.7.6 Modifications to the Solver Algorithm

Once the modifications to the grid and reconstruction systems described above have been implemented, only relatively minor changes are required to the solver algorithm itself.

The first set of modifications concerns the grid movement. At the beginning of each physical time step, the position of each of the component grids needs to be updated. Since only rigid motions of the component grids are considered in the present work, this simply involves applying a linear transformation to the positions of the component grid nodes. Edge centres, unit normals, and unit tangents, and cell circumcentres are then updated according to the new node positions. Edge lengths, dual edge lengths, and cell areas are invariant under rigid motions and do not need to be updated. The reconstruction systems are then updated, as discussed in the previous section. Finally, the edge centre normal momenta at the previous one or two time steps (according to whether first-order or second-order temporal discretisation is being used respectively) are updated according to (6.23).

Next, the primary flow variables at the *interpolation* edge centres and cell circumcentres must be interpolated from other grids in the grid set. This is done implicitly, by replacing the row of the relevant conservation equation corresponding to the edge or cell with an equation expressing the interpolation. The source node or cell for the interpolation is found using the element finding algorithm of Section 6.7.2. For an *interpolation* element $i$ with an interpolation source element src, the equation added to the conservation equation system can simply be

$$\varphi_i^{(m+1)} - \varphi_{\text{src}}^{(m+1)} - \nabla\varphi\big|_{\text{src}}^{(m+1)} \cdot (\boldsymbol{x}_i - \boldsymbol{x}_{\text{src}}) = 0 \tag{6.36}$$

This is not optimal for the iterative solvers used, however, because the diagonal element of the implicit operator is unity rather than $\mathcal{O}\left(\Omega_i / \left(\Delta t_i + \Delta\tilde{t}_i\right)\right)$ as is the case for the rows corresponding to the implicit operator. It is therefore better to use the weighted

interpolation equation

$$\frac{\Omega_i}{\Delta t_i + \Delta\tilde{t}_i}\left[\varphi_i^{(m+1)} - \varphi_{\text{src}}^{(m+1)} - \nabla\varphi\big|_{\text{src}}^{(m+1)} \cdot (\boldsymbol{x}_i - \boldsymbol{x}_{\text{src}})\right] = 0 \tag{6.37}$$

The interpolation is slightly different for the cell circumcentre pressures as the unknowns in the conservation equation are the pressure corrections rather than the actual pressures. Using $p_i^{(m+1)} = p_i^{(m)} + \delta p_i^{(m+1)}$ the interpolation equation for the pressure correction equation is

$$\begin{aligned}\frac{\Omega_i}{\Delta t_i + \Delta\tilde{t}_i}&\left[\delta p_i^{(m+1)} - \delta p_{\text{src}}^{(m+1)} - \nabla\delta p\big|_{\text{src}}^{(m+1)} \cdot (\boldsymbol{x}_i - \boldsymbol{x}_{\text{src}})\right] \\ &= \frac{\Omega_i}{\Delta t_i + \Delta\tilde{t}_i}\left[p_{\text{src}}^{(m)} - p_i^{(m)} + \nabla p\big|_{\text{src}}^{(m)} \cdot (\boldsymbol{x}_i - \boldsymbol{x}_{\text{src}})\right]\end{aligned} \tag{6.38}$$

The update of the secondary flow variables must then be performed. This follows in the obvious manner from the considerations above. For example, if the edge centre tangential momenta are being updated using the node-based piecewise linear momentum reconstruction, a base node that is set to *interpolation* is replaced by an interpolation source node.

As the component grids move around the base grid, the tag (*active*, *interpolation*, or *disabled*) attached to each element may change from one type to another. Most of these tag changes are straightforward. However, *disabled* elements do not have their flow variables updated, so the changes from *disabled* to *interpolation* and from *disabled* to *active* require special attention. The former transition simply requires that all the flow variables are interpolated after the transition. The latter transition, however, has problems that cannot be remedied easily, since the time discretisation requires not only the flow variables at the latest physical time level but also those at the previous one or two physical time levels. It is possible to interpolate the flow variables at the previous time levels, but so as to avoid difficulties for the current work it is instead required that the physical time step is small enough that the component grids do not move far enough over two time steps for this type of transition to occur. In practice this is hardly a restriction as long as the layer of interpolation elements separating the *active* elements from the *disabled* elements is large enough (testing found two layers to be sufficient for all the test cases), and as long as the grid sizes roughly match where there are *interpolation* elements in one grid overlapping *active* elements in another. This restriction is also likely to be beneficial for the accuracy and robustness of the flow variable reconstructions.

## 6.8 Modifications to the Boundary Conditions for Unsteady Flows

Apart from being evaluated at the pseudotime levels rather than the physical time levels, there is little change in most of the boundary conditions described in the previous chapters. One set of boundary conditions that does change non-trivially when applying the solver to unsteady problems is the boundary conditions imposed along walls, since the velocity of the wall will not necessarily be zero. These changes are now discussed.

### 6.8.1 Moving No-Slip Wall Boundary Conditions

If the wall boundary is not stationary, then imposing the no-slip wall condition (5.56) on the flow velocity would violate the no-slip and no-penetration conditions. Instead, the fluid flow velocity at the wall must be set equal to the velocity of the wall $\boldsymbol{v}_w$.

$$\boldsymbol{u}_b = \mathring{\boldsymbol{v}}_w \tag{6.39}$$

Instead of simply reflecting the interior cell flow velocity, linear extrapolation is used to obtain the dummy cell centre flow velocity

$$\boldsymbol{u}_{\mathcal{D}} = 2\mathring{\boldsymbol{v}}_w - \boldsymbol{u}_{\mathcal{I}} \tag{6.40}$$

The momentum at the dummy cell circumcentre is also obtained by linear extrapolation

$$\boldsymbol{m}_{\mathcal{D}} = 2\boldsymbol{m}_b - \boldsymbol{m}_{\mathcal{I}} \tag{6.41}$$

However, the momentum at the wall edge is not known *a priori*. It can be written in the form

$$\boldsymbol{m}_b = \rho_b \mathring{\boldsymbol{v}}_w \tag{6.42}$$

where the wall density $\rho_b$ is obtained by linear interpolation from the adjacent interior cell and dummy cell. This results in the treatment of the wall momentum being different for adiabatic no-slip walls and non-adiabatic walls with specified wall temperature.

### Adiabatic No-Slip Wall

At adiabatic no-slip walls, the enthalpy, pressure, and density are all treated in exactly the same way as in the stationary case

$$\begin{aligned} h_{\mathcal{D}} &= h_{\mathcal{I}} \\ p_{\mathcal{D}} &= p_{\mathcal{I}} \\ \rho_{\mathcal{D}} &= \rho_{\mathcal{I}} \end{aligned} \tag{6.43}$$

This results in the wall density being equal to the interior cell circumcentre density. Therefore, the dummy cell circumcentre momentum is

$$\boldsymbol{m}_{\mathcal{D}} = 2\rho_{\mathcal{I}}\mathring{\boldsymbol{v}}_w - \boldsymbol{m}_{\mathcal{I}} \tag{6.44}$$

and the set of boundary conditions actually imposed on the governing equations is

$$\begin{aligned} \rho_{\mathcal{D}}^{(m+1)} - \rho_{\mathcal{I}}^{(m+1)} &= 0 \\ (\boldsymbol{m}\cdot\boldsymbol{n})_b^* &= \rho_{\mathcal{I}}^{(m+1)}\left(\mathring{\boldsymbol{v}}_w\cdot\boldsymbol{n}_b\right)^{n+1} \\ \delta p_{\mathcal{D}}^{(m+1)} - \delta p_{\mathcal{I}}^{(m+1)} &= 0 \end{aligned} \tag{6.45}$$

### No-Slip Walls with Specified Wall Temperature

In the case of a non-adiabatic no-slip wall where the wall temperature is specified, the dummy cell circumcentre density is related to the interior cell density by (5.68). The wall edge centre density is then

$$\rho_w = \frac{\rho_{\mathcal{I}}\mathring{h}_w}{2\mathring{h}_w - h_{\mathcal{I}}} \tag{6.46}$$

The wall edge centre momentum can then be written

$$\boldsymbol{m}_b = \frac{\rho_{\mathcal{I}}\mathring{h}_w}{2\mathring{h}_w - h_{\mathcal{I}}}\mathring{\boldsymbol{v}}_w \tag{6.47}$$

Finally, the set of boundary conditions imposed directly on the governing equations is

$$\begin{aligned} \rho_{\mathcal{D}}^{(m+1)} \left( 2\mathring{h}_w - h_{\mathcal{I}}^{(m)} \right) - \rho_{\mathcal{I}}^{(m+1)} h_{\mathcal{I}}^{(m)} &= 0 \\ (\boldsymbol{m} \cdot \boldsymbol{n})_b^{(m*)} &= \frac{\rho_{\mathcal{I}}^{(m+1)} \mathring{h}_w}{2\mathring{h}_w - h_{\mathcal{I}}^{(m)}} \left( \mathring{\boldsymbol{v}}_w \cdot \boldsymbol{n}_b \right)^{(m+1)} \\ \delta p_{\mathcal{D}}^{(m+1)} - \delta p_{\mathcal{I}}^{(m+1)} &= 0 \end{aligned} \tag{6.48}$$

### 6.8.2 Moving Slip Wall Boundary Conditions

If the unsteady solver is used to compute unsteady inviscid flows, then the slip wall conditions of Section 4.5.5 may need to be imposed. These need to be modified in very similar way to the no-slip wall conditions just discussed, except only the component of the wall velocity normal to the wall is relevant. As for the steady solver, the density at the wall edge centre and dummy cell centre are set equal to the interior cell centre density

$$\rho_{\mathcal{D}} = \rho_b = \rho_{\mathcal{I}} \tag{6.49}$$

The normal flow velocity at the wall edge centre is calculated from

$$(\boldsymbol{m} \cdot \boldsymbol{n})_b = \rho_{\mathcal{I}} \boldsymbol{v}_w \cdot \boldsymbol{n}_b \tag{6.50}$$

whilst the tangential flow velocity at the wall edge centre is extrapolated from the interior state using the methods discussed in Section 3.9. The momentum at the dummy cell circumcentre is then calculated by linear extrapolation from the wall edge centre and interior cell circumcentre

$$\boldsymbol{m}_{\mathcal{D}} = 2\rho_{\mathcal{I}} \left( \boldsymbol{v}_w \cdot \boldsymbol{n}_b \right) \boldsymbol{n}_b + 2 \left( \boldsymbol{m} \cdot \boldsymbol{t} \right)_b - \boldsymbol{m}_{\mathcal{I}} \tag{6.51}$$

The set of boundary conditions directly imposed on the governing equations at moving slip wall boundaries is

$$\begin{aligned} \rho_{\mathcal{D}}^{(m+1)} - \rho_{\mathcal{I}}^{(m+1)} &= 0 \\ (\boldsymbol{m} \cdot \boldsymbol{n})_b^{(m)} &= \rho_{\mathcal{I}}^{(m+1)} \left( \mathring{\boldsymbol{v}}_w \cdot \boldsymbol{n}_b \right)^{n+1} \\ \delta p_{\mathcal{D}}^{(m+1)} - \delta p_{\mathcal{I}}^{(m+1)} &= 0 \end{aligned} \tag{6.52}$$

## 6.9 Spatial and Temporal Refinement Studies

The refinement study for the unsteady solver is significantly more complex than that performed for the inviscid and viscous solvers. This is partly due to the exact solution becoming a function of time as well as spatial position (and therefore the approximate solution and truncation error also becoming functions of time), but is also due to the added requirement of evaluating the *temporal* order of accuracy (i.e. the rate at which the error in the approximate solution reduces as the time step is reduced) of the solver as well as the spatial order of accuracy. In fact, this necessitates two separate refinement studies: one with the time step being held constant whilst the grid is refined spatially, and the other with the spatial discretisation being held constant whilst the time step is refined. The first of these refinement studies will demonstrate that the unsteady solver retains the second-order spatial accuracy of the inviscid and viscous solvers, whilst the second will demonstrate that the unsteady solver possesses second-order temporal accuracy.

### 6.9.1 Procedure

For the unsteady solver, the flow domain is discretised both in space and in (physical) time. The resulting structure will be referred to here simply as the *discretisation*. The discretisation is much simplified by fact that the physical time step remains constant both in space and in time. As a result, the discretisation can be viewed as a series of slices representing the domain at each discrete physical time step. Each slice consists of a grid resembling those considered during the grid refinement studies for the inviscid and viscous solvers in Section 4.7 and Section 5.8 respectively, although the grids for the unsteady solver may be Chimera grids.

The first task is the choice of appropriate measures of spatial coarseness $h$ and temporal coarseness $q$ of the discretisation. Due to the use of the Chimera method and the specific requirements placed on the Chimera grids, the meshwidth

$$h \coloneqq \frac{1}{N} \sum_{b \in \Gamma'_w} \ell_b \tag{6.53}$$

as defined in Section 4.7.1 can be used without change, since it is the same regardless of the time level at which it is evaluated. The physical time step is the obvious choice for the temporal coarseness, and therefore $q = \Delta t$.

Next a two-parameter family of discretisations must be defined, with the first

parameter representing the spatial coarseness and the second representing the temporal coarseness. That discretisations can be uniquely parametrised by temporal coarseness is a trivial consequence of the use of a constant physical time step. The parametrisation by the spatial coarseness follows in a way analogous to that described in Section 4.7.1, by having the target meshwiths at the geometry of interest be functions of the grid coarseness parameter $h$ with all other meshing parameters being kept constant. Therefore, given strictly decreasing ordered sets of $N$ spatial coarseness values $\{h_N\}$ and $M$ temporal coarseness values $\{q_M\}$, a discretisation can be uniquely specified by the pair $(h_i, q_j)$ for $i \leq N$ and $j \leq M$.

A suitable measure of the error in the approximate solution must now be defined. Let $\hat{\varphi}(x, y, t)$ denote the exact solution of the unsteady Navier-Stokes equations for a given set of auxiliary data, $\varphi_{h_i,q_j}(x, y, t)$ denote the approximate solution obtained on the discretisation parametrised by spatial coarseness $h_i$ and temporal coarseness $q_j$, and ${\varepsilon_{h_i,q_j}}^{\varphi}(x, y, t)$ denote the corresponding error. Following the discussion justifying the definition (4.66) of the error in the approximate solution for an arbitrary flow variable $\varphi$ for the steady solver, the error in the approximate unsteady solution can be defined to be

$$ {\varepsilon_{h_i,q_j}}^{\varphi}(x, y, t) \coloneqq \hat{\varphi}(x, y, t) - \varphi_{h_i,q_j}(x, y, t) \tag{6.54} $$

Unfortunately, for all but the simplest flow cases $\hat{\varphi}(x, y, t)$ is unknown. As a result, an approximation to the error must instead be used in practice. This approximation is obtained by replacing the exact solution $\hat{\varphi}(x, y, t)$ in (6.54) with the most accurate approximate solution, which is presumed to be the approximate solution obtained on the discretisation with the lowest spatial coarseness $h_N$ and temporal coarseness $q_M$. The error is therefore approximated by

$$ {\varepsilon_{h_i,q_j}}^{\varphi}(x, y, t) \approx \varphi_{h_N,q_M}(x, y, t) - \varphi_{h_i,q_j}(x, y, t) \tag{6.55} $$

Using an appropriate norm, the vector of errors in a given flow variable $\varphi$ at each of the points on the geometry of interest at each time level can be reduced to an error indicator $E(h_i, q_i)^{\varphi}$ in $\varphi$ for the particular discretisation $(h_i, q_i)$. This error can be expanded as a Taylor series, which yields

$$ \begin{aligned} E(h,q)^{\varphi} =& a_0 + a_h h + a_t t + a_{h^2} h^2 + a_{hq} hq + a_{q^2} q^2 \\ &+ a_{h^3} h^3 + a_{h^2q} h^2 q + a_{hq^2} hq^2 + a_{q^3} q^3 + \cdots \end{aligned} \tag{6.56} $$

As for the steady solver, for the unsteady solver to be consistent the constant term

must be zero in order for the error to go to zero in the limit as the spatial and temporal coarseness are reduced to zero. The error is then

$$\begin{aligned} E(h,q)^{\varphi} =& a_h h + a_t t + a_{h^2} h^2 + a_{hq} hq + a_{q^2} q^2 \\ &+ a_{h^3} h^3 + a_{h^2 q} h^2 q + a_{hq^2} hq^2 + a_{q^3} q^3 + \cdots \end{aligned} \tag{6.57}$$

At this point, it is clear that there are two methods for carrying out the refinement study:

1. Obtain approximate solutions for various values of the spatial and temporal coarseness, and plot the resulting error values as a surface in $(h, q, E)$-space. The shape of this plane is then studied as the finest discretisation is approached along the $h$- and $q$-coordinate directions. This method would be very time consuming because many approximate solutions would need to be computed.

2. Effectively perform two separate refinement studies. The first would study the shape of the error curve as the spatial coarseness is reduced whilst the temporal coarseness is set to a small constant, and the second would study the shape of the error curve as the temporal coarseness is reduced whilst the spatial coarseness is set to a small constant.

The second method is obviously superior as it provides all the necessary information without the need for computing essentially superfluous approximate solutions.

**Spatial Refinement Study**

First, the temporal coarseness is set to a small constant $q_M$, and the Taylor expansion of the error (6.57) is reordered into the form

$$\begin{aligned} E\left(h, q_M\right)^{\varphi} = \left(a_q q_M\right) &+ \left(a_h + a_{hq} q_M + a_{hq^2} q_M^2 + \cdots\right) h \\ &+ \left(a_{h^2} + a_{h^2 q} q_M + a_{h^2 q^2} q_M^2 + \cdots\right) h^2 \\ &+ \left(a_{h^3} + a_{h^3 q} q_M + a_{h^3 q^2} q_M^2 + \cdots\right) h^3 + \cdots \end{aligned} \tag{6.58}$$

For the purposes of the spatial refinement study only, it is now assumed that the solver is second-order accurate in time. This reduces the error expansion to

$$\begin{aligned} E\left(h, q_M\right)^{\varphi} = &\left(a_h + a_{hq^2} q_M^2 + \cdots\right) h \\ &+ \left(a_{h^2} + a_{h^2 q^2} q_M^2 + \cdots\right) h^2 \\ &+ \left(a_{h^3} + a_{h^3 q^2} q_M^2 + \cdots\right) h^3 + \cdots \end{aligned} \tag{6.59}$$

It is then assumed that the solver is $p^{\text{th}}$-order spatially accurate, yielding

$$\begin{aligned} E\left(h, q_M\right)^{\varphi} = &\left(a_{h^p} + a_{h^p q^2} q_M^2 + \cdots\right) h^p \\ &+ \left(a_{h^{p+1}} + a_{h^{p+1} q^2} q_M^2 + \cdots\right) h^{p+1} + \cdots \end{aligned} \tag{6.60}$$

For sufficiently small meshwidth $h$ the leading term dominates the error expansion, hence

$$E\left(h, q_M\right)^{\varphi} \approx a_{h^p} h^p + a_{h^p q^2} h^p q_M^2 \tag{6.61}$$

If the temporal coarseness $q$ is of the same order as the meshwidth (or lower), then $h^p q_M^2$ is of the same order as $h^{p+2}$ and can also be neglected. Therefore

$$E\left(h, q_M\right)^{\varphi} \approx a_{h^p} h^p \tag{6.62}$$

Finally, taking the natural logarithm of both sides and rearranging yields

$$\ln E\left(h, q_M\right)^{\varphi} \approx p \ln h + C \tag{6.63}$$

where $C \coloneqq \ln a_{h^p}$. As a result, as long as the temporal coarseness is constant and sufficiently small the spatial refinement study takes exactly the same form as the grid refinement studies presented in the previous chapters.

**Temporal Refinement Study**

The theoretical basis for the temporal refinement study follows in exactly the same way as for the spatial refinement study. The result is that if the spatial coarseness is a constant $h_N$ which is sufficiently small then the error is well approximated by

$$E\left(h_N, q\right)^{\varphi} \approx a_{q^r} q^r \tag{6.64}$$

where $r$ is the temporal order of accuracy of the solver. This implies

$$\ln E\left(h_N, q\right)^{\varphi} \approx r \ln q + D \tag{6.65}$$

where $D \coloneqq \ln a_{q^r}$. The temporal refinement study is therefore completely analogous to the spatial refinement study, except the temporal coarseness is reduced with the spatial coarseness remaining constant.

### 6.9.2 Results

The test case chosen for the refinement studies was that of a cylinder with sinusoidal oscillation transverse to the farfield flow direction with $M_r = 0.25$, $R_e = 240$, and $P_r = 0.71$. The relatively low Mach number ensures little limiter activation, whilst the Reynolds number is well above the critical Reynolds number so that laminar vortex shedding will occur and there will be significant unsteadiness in the flow. The parameters of the oscillaton of the cylinder were: phase $\psi = 0°$, amplitude $\|\boldsymbol{A}\| = 1$, and *Strouhal number* $\mathrm{St} = 0.005$, where the Strouhal number

$$\mathrm{St} \coloneqq \frac{f\ell_r}{u_r} \tag{6.66}$$

is a dimensionless group describing the characteristic frequency $f$ of the oscillation. The non-dimensional integration time was set to $t_{\mathrm{int}} = 20$, where the time is non-dimensionalised by dividing by the reference time $t_r \coloneqq \ell_r/u_r$. Hence, the integration time covers one tenth of the period of oscillation of the cylinder. Note that since the characteristic frequency is the reciprocal of the characteristic period, the Strouhal number is simply the reciprocal of the physical Courant number defined previously (6.1).

The Chimera grid set consisted of a base grid containing the same farfield geometry used for the exterior viscous flow cases considered in the previous chapter and a single component grid centred on the cylinder. The target radius of the component grid was set to four cylinder diameters. The farfield target meshwidth was set to 32 cylinder diameters, and the target grid growth rate was set to 1.12 to ensure good grid quality. Line sources were placed in the streamwise direction downwind of the cylinder to provide adequate resolution of the wake. The target meshwidth at each point along the line sources was kept the same during the grid refinement process, so that only the target meshwidth at the cylinder was refined. The selected target meshwidth on the cylinder wall was kept constant with respect to angular position. As usual, the physical time step was kept constant across the grid and throughout the integration period.

#### Spatial Refinement Study

The results for the spatial refinement study are shown in Table 6.1 and Figure 6.2. The results are very similar to that for the steady viscous case: second-order accuracy has been achieved for the unsteady viscous solver in the $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{F}_2$ norms, whilst superlinear convergence has been achieved in the $\mathcal{L}_\infty$ norm. The interpretation remains

| $h$ | $N_{\text{cells}}$ | $\|\boldsymbol{\varepsilon}_p\|_{\mathcal{L}_1}$ | $\|\boldsymbol{\varepsilon}_p\|_{\mathcal{L}_2}$ | $\|\boldsymbol{\varepsilon}_p\|_{\mathcal{L}_\infty}$ | $\|\boldsymbol{\varepsilon}_p\|_{\mathcal{F}_2}$ |
|---|---|---|---|---|---|
| 5.08E-04 | 216813 | 9.18E+04 | 1.42E+02 | 7.46E-01 | 4.54E+00 |
| 4.30E-04 | 249887 | 1.00E+05 | 1.64E+02 | 1.00E+00 | 5.22E+00 |
| 3.75E-04 | 281100 | 1.01E+05 | 1.66E+02 | 8.50E-01 | 5.29E+00 |
| 3.20E-04 | 322851 | 7.77E+04 | 1.31E+02 | 6.77E-01 | 4.19E+00 |
| 2.93E-04 | 348839 | 5.83E+04 | 1.01E+02 | 6.23E-01 | 3.23E+00 |

Table 6.1: The computed error norms for the spatial refinement study.

| $q$ | $\|\boldsymbol{\varepsilon}_p\|_{\mathcal{L}_1}$ | $\|\boldsymbol{\varepsilon}_p\|_{\mathcal{L}_2}$ | $\|\boldsymbol{\varepsilon}_p\|_{\mathcal{L}_\infty}$ | $\|\boldsymbol{\varepsilon}_p\|_{\mathcal{F}_2}$ |
|---|---|---|---|---|
| 2.50E-03 | 1.34E+05 | 1.76E+02 | 5.06E-01 | 8.76E+00 |
| 2.00E-03 | 7.05E+04 | 1.22E+02 | 7.53E-01 | 6.04E+00 |
| 1.80E-03 | 6.65E+04 | 1.16E+02 | 7.47E-01 | 5.73E+00 |
| 1.60E-03 | 5.92E+04 | 1.03E+02 | 6.41E-01 | 5.12E+00 |
| 1.25E-03 | 2.95E+04 | 5.85E+01 | 4.73E-01 | 2.89E+00 |

Table 6.2: The computed error norms for the temporal refinement study.

the same, that the unsteady viscous solver is second-order spatially accurate but not uniformly so. This is not a concern in practice for the same reason discussed in the grid refinement study for the steady viscous solver.

#### Temporal Refinement Study

The results for the temporal refinement study are shown in Table 6.2 and Figure 6.3. The results show that the unsteady viscous solver achieves second-order temporal convergence in the $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{F}_2$ norms, but only superlinear temporal convergence in the $\mathcal{L}_\infty$ norm.

## 6.10 Presentation and Discussion of Numerical Results for Unsteady Flows with Stationary Geometries

### 6.10.1 Right-Circular Cylinder

In Section 5.9.2 viscous flow over a right circular cylinder was considered for Mach number 0.5 and Reynolds numbers 20, 40, and 50. The first two of these yielded stationary solutions with attached vortices. However, the $R_e = 50$ case was shown to lead to very small-scale instability as the vortices began to shed from the cylinder in a laminar fashion. As the Reynolds number is increased further, the vortex shedding increases in frequency and amplitude [Whi06, p. 10] and therefore the flow will be

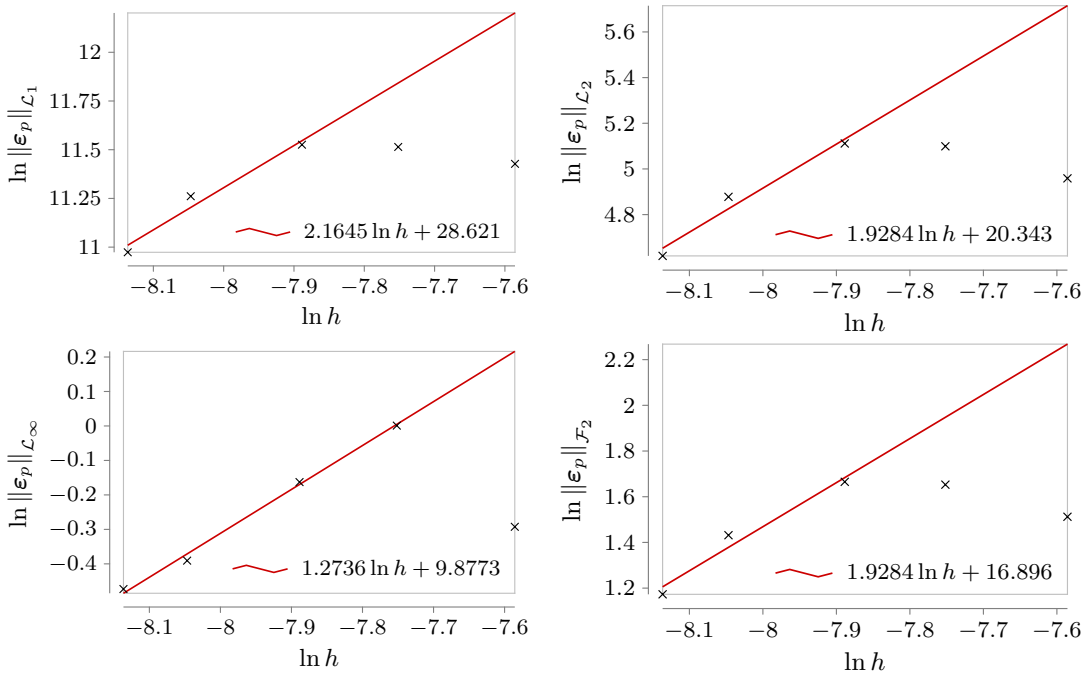



Figure 6.2: The natural logarithm of the meshwidth against the natural logarithms of the pressure error in each of the norms, together with best-fit lines placed using linear regression on the three data points with the lowest values of the meshwidth.

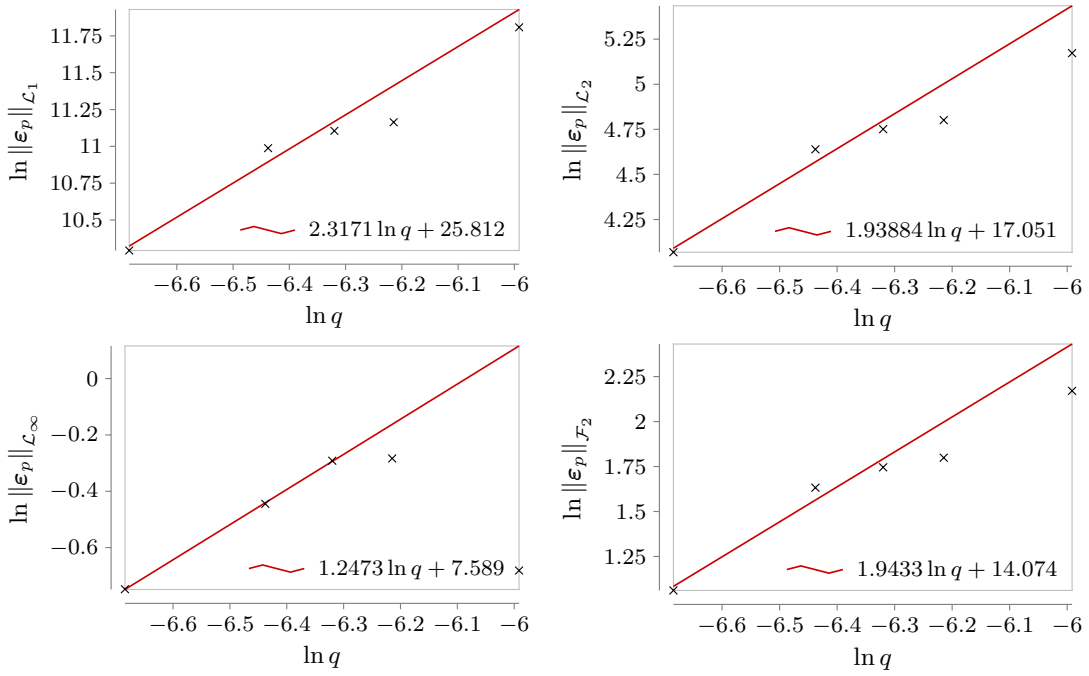



Figure 6.3: The natural logarithm of the physical time step against the natural logarithms of the pressure error in each of the norms, together with best-fit lines placed using linear regression on the three data points with the lowest values of the physical time step.

even less steady. This is therefore a useful first test of the unsteady solver. The Reynolds number must still be kept relatively low, however, to avoid the transition to turbulent flow that begins at approximately $R_e = 150$ [Lie66]. Three cases will be considered, with Reynolds numbers of 50, 75, and 100, with all other flow parameters remaining the same as for the steady viscous case considered previously. The repetition of the $R_e = 50$ case is intended to provide a comparison between the steady and unsteady solvers. As the Reynolds number is increased, the solver should predict laminar vortex shedding of increasing amplitude and modestly increasing Strouhal number [CT15]. The wall temperature was set equal to the farfield total temperature, whilst the dynamic viscosity and thermal conductivity were set equal to the farfield values across the grid. For these cases, reported lengths are non-dimensionalised by the diameter of the cylinder.

The same grid was used as for the steady viscous cases in Section 5.9.2. In [CT15] the authors reported Strouhal numbers of 0.114, 0.144, and 0.161 for the three selected cases, which correspond to physical Courant numbers of 8.78, 6.94, and 6.21 respectively. The non-dimensional physical time step was set at 0.2 (where the time step is non-dimensionalised by dividing by the reference time $\ell_r/u_r$) so that each period of the shedding would be divided into approximately $30-45$ physical time steps (under the assumption that the predicted shedding frequency would be reasonably similar to the published results). The same physical time step was used for all three cases. For each physical time step, the pseudotime stepping was continued until the convergence indicator (6.20) reached $-3$. The Courant number of the pseudotime stepping

$$\tilde{\sigma} \coloneqq \frac{u_r \tilde{\Delta} t_i}{\Delta x_i} \tag{6.67}$$

was set to 15, which resulted in convergence of the dual time stepping to $\tilde{\Theta} = -3$ in an average of 29, 42, and 49 pseudotime steps per physical time step for the $R_e = 50$, $R_e = 75$, and $R_e = 100$ cases respectively. The results produced by the steady viscous solver for the $R_e = 50$ case were used as the initial conditions for all three cases. The computed time-averages of the lift and drag coefficients were $7.224 \times 10^{-3}$ and $6.222 \times 10^{-2}$ respectively.

The results are shown as contour plots of the coefficient of pressure and $x$-velocity in Figure 6.4, as line plots of the deviations of the coefficients of lift and drag from the time-averaged values in Figure 6.5, and in Table 6.3.

The growth of the amplitude of the laminar vortex shedding from very small to large as the Reynolds number is increased from 50 to 100 is well-captured in the contour

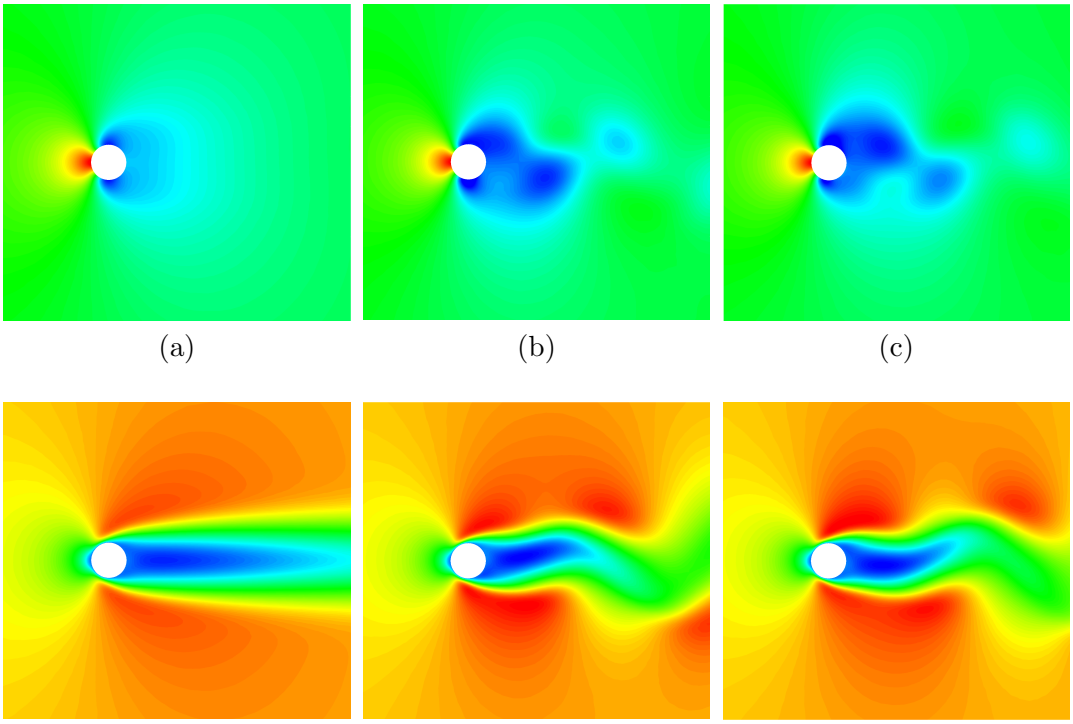



Figure 6.4: Instantaneous coefficient of pressure contours (top row) and instantaneous $\boldsymbol{u} \cdot \boldsymbol{i}$ contours (bottom row) around the right circular cylinder at $M_\infty = 0.5$ for (a) $R_e = 50$, (b) $R_e = 75$, and (c) $R_e = 100$.

plots. One notable feature of the computed solutions is that the $R_e = 50$ case took far longer (by a factor of ten) to become periodic, i.e. to reach constant values of the time-averages of the aerodynamic coeffiients and constant values of the instantaneous fluctuations from the time-averages.

Table 6.3 shows that many of the predictions of the solver are in reasonable agreement with the published results. For example, the time-averages of the coefficient of drag are within 5% for all three values of the Reynolds number. However, there is significant disagreement in some of the other parameters. For example, the computed Strouhal number for the $R_e = 50$ case is 10.5% lower than the published value. This is likely due to the proximity to the critical Reynolds number. The Reynolds number is only just large enough for shedding to occur, hence the solution is very sensitive to numerical diffusion. The numerical diffusion is obviously much lower for the DNS study, since the grids used in the DNS study had approximately 1.6 million cells compared to the $58,821$ used in the present work. For the higher Reynolds number cases, the

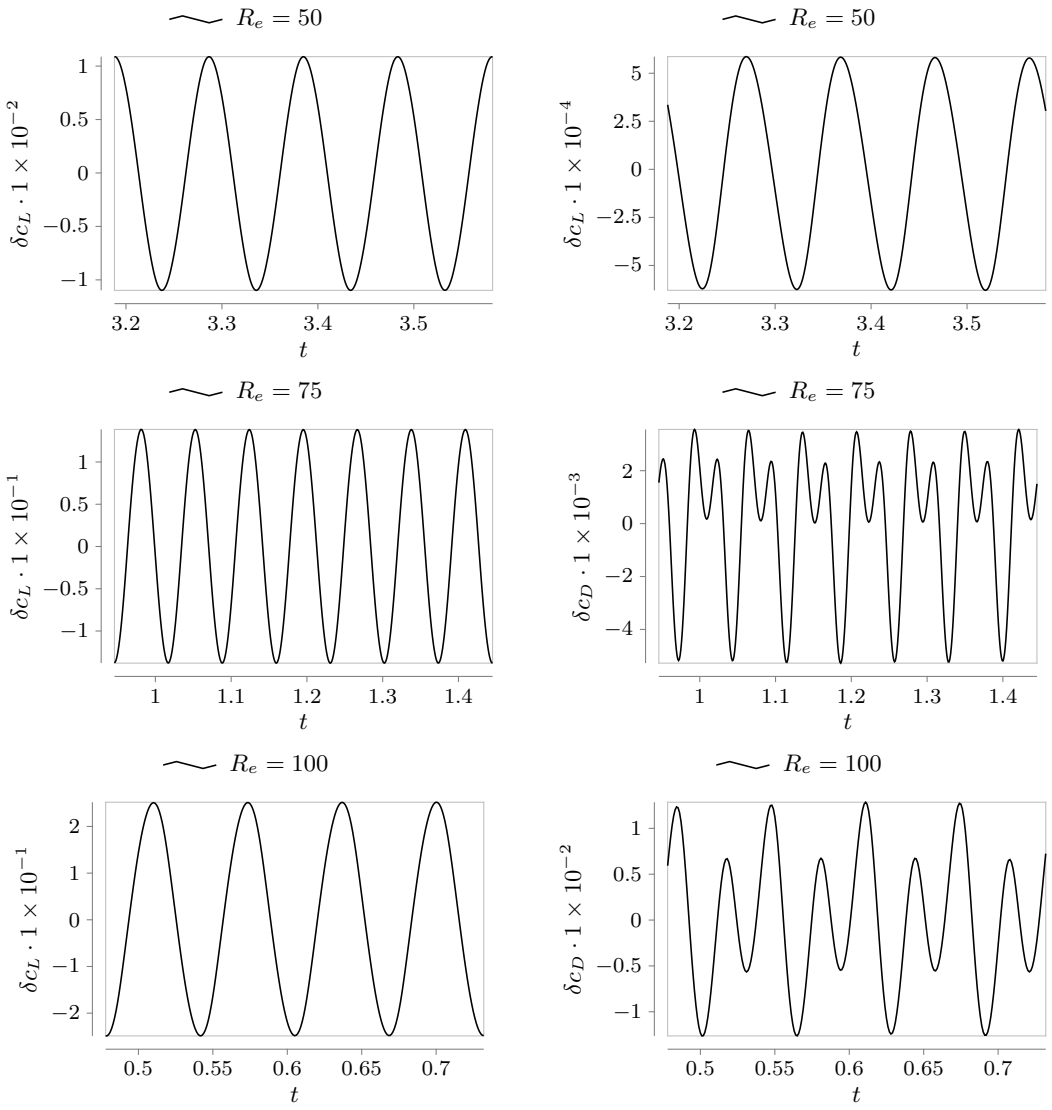



Figure 6.5: Time histories of the deviations of the lift and drag coefficients from the time-averaged values for the right circular cylinder at $M_\infty = 0.5$.

| $R_e$ | Parameter | Present Solver | [CT15] |
|---|---|---|---|
| 50 | $\overline{c_L}$ | $3.15 \times 10^{-2}$ | 0 |
| | $\lvert\delta c_L\rvert$ | $1.10 \times 10^{-2}$ | $1.44 \times 10^{-2}$ |
| | $\overline{c_D}$ | 1.485 | 1.56 |
| | $\lvert\delta c_D\rvert$ | $6.36 \times 10^{-4}$ | $2.03 \times 10^{-5}$ |
| | St | 0.102 | 0.114 |
| 75 | $\overline{c_L}$ | $2.80 \times 10^{-2}$ | 0 |
| | $\lvert\delta c_L\rvert$ | 0.139 | 0.190 |
| | $\overline{c_D}$ | 1.45 | 1.53 |
| | $\lvert\delta c_D\rvert$ | $5.27 \times 10^{-3}$ | $2.13 \times 10^{-3}$ |
| | St | 0.140 | 0.144 |
| 100 | $\overline{c_L}$ | $2.46 \times 10^{-2}$ | 0 |
| | $\lvert\delta c_L\rvert$ | 0.252 | 0.352 |
| | $\overline{c_D}$ | 1.43 | 1.54 |
| | $\lvert\delta c_D\rvert$ | $1.29 \times 10^{-2}$ | $1.01 \times 10^{-2}$ |
| | St | 0.157 | 0.161 |

Table 6.3: The computed values of the time-averaged lift and drag coefficients, maximum instantaneous lift and drag coefficient perturbations, and Strouhal numbers for the right circular cylinder at $R_e = 50$, $R_e = 75$, and $R_e = 100$ compared to the results of the DNS study presented in [CT15].

discrepancy in the computed Strouhal number is reduced to below 3%

### 6.10.2 NACA 0012 Aerofoil

Transonic unsteady flow over the NACA 0012 aerofoil is now considered. The case is taken from the 1985 GAMM workshop [Bri+87a], and has the farfield Mach number, angle of attack, and Reynolds number set to 0.8, 0° and 10,000 respectively, whilst the temperature of the aerofoil wall was set equal to the farfield total temperature. The result is unsteady vortex shedding from the trailing edge that somewhat resembles those seen in the right circular cylinder case. The dynamic viscosity and thermal conductivity were constant and equal to the farfield values throughout the flow domain. For this case, reported lengths are non-dimensionalised by the chord length of the aerofoil.

The specified target meshwidth on the aerofoil wall varied from $6.22 \times 10^{-3}$ in the highest curvature regions and at the trailing edge to $8.48 \times 10^{-3}$ in the regions of lowest curvature. Line sources were used to refine the grid downstream of the trailing edge to ensure sufficient resolution of the unsteady wake. The target meshwidth at the farfield was set to 32 chords. The resulting grid had 386 boundary points, of which 328 were placed on the aerofoil wall. The selected target growth rate of 1.12 yielded 23,061

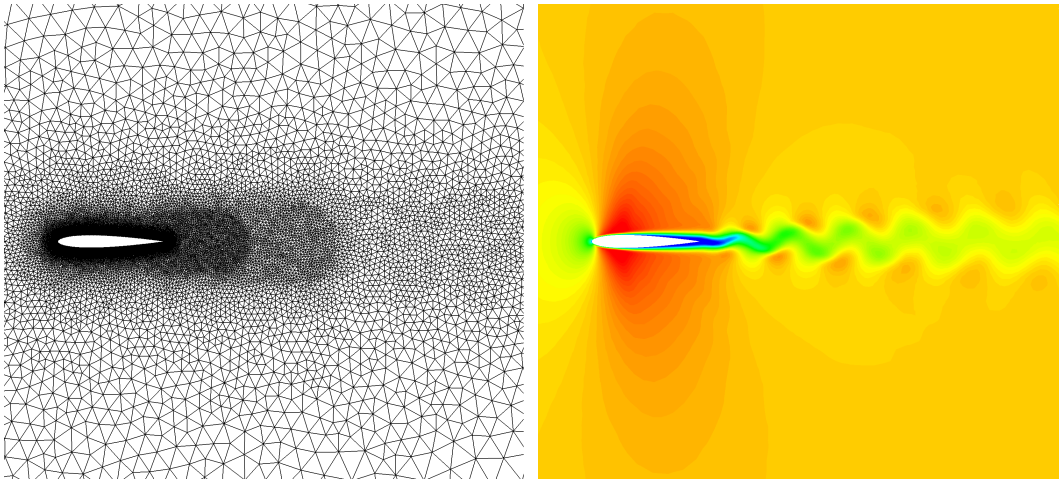


Figure 6.6: Left: a section of the grid used for computing the flow over the NACA 0012 at $M_r = 0.8$, $\alpha = 0°$, $R_e = 10,000$. Right: the resulting $\boldsymbol{u} \cdot \boldsymbol{i}$ contours

nodes and 45,736 cells in the complete grid. A section of the grid is shown on the left of Figure 6.6.

The physical time step was again selected using the shedding Strouhal number published in the literature. In this case the value used was 0.22 from [PA18], but note that this Strouhal number is based on the aerofoil thickness rather than the aerofoil chord. Since the amplitude of the shedding is considerably lower than that from the right circular cylinder, fewer physical time steps per period were deemed necessary. The non-dimensional physical time step was therefore set to $4 \times 10^{-2}$, corresponding to approximately 15 time steps per period. The Courant number of the pseudotime stepping was set to 7, which resulted in convergence to $\tilde{\Theta} = -3$ in an average of 26 pseudotime steps per physical time step. The initial conditions were obtained by performing a few hundred iterations with the steady solver before switching to the unsteady solver.

The results are shown as a contour plot of instantaneous $x$-velocity on the right of Figure 6.6 and as line plots of the time history of the lift and drag coefficients in Figure 6.7. In Table 6.4, the computed Strouhal number is compared with published results, and shows a disagreement of approximately 13%. The computed time-averaged lift and drag coefficients were $-7.22 \times 10^{-3}$ and $6.22 \times 10^{-2}$ respectively.

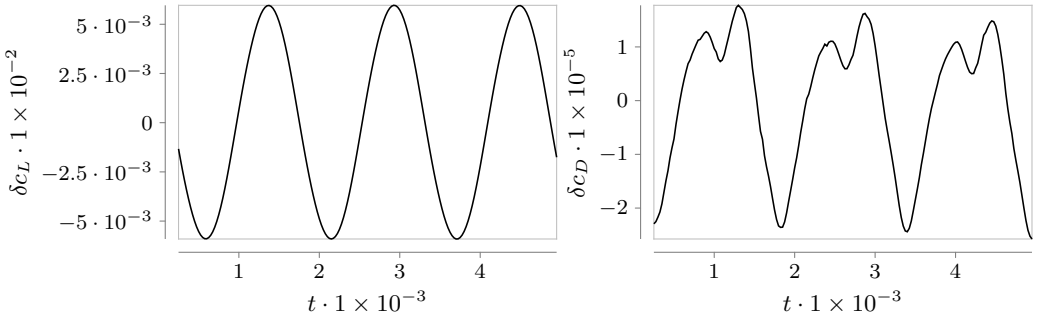



Figure 6.7: Time histories of the deviations of the lift and drag coefficients from the time-averaged values for the NACA 0012 aerofoil at $M_r = 0.8$, $\alpha = 0°$, $R_e = 10,000$.

| Solver | St |
|---|---|
| NS [PA18] | 0.221 |
| ES-BGK [PA18] | 0.218 |
| Present solver | 0.191 |

Table 6.4: Comparison of computed Strohal number (based on aerofoil thickness) for the NACA 0012 aerofoil at $M_r = 0.8$, $\alpha = 0°$, $R_e = 10,000$ with published results from two solvers from [PA18].

## 6.11 Presentation and Discussion of Numerical Results for Unsteady Flows with Moving Geometries

### 6.11.1 Oscillating Right Circular Cylinder

For this test case, the quasi-incompressible flow (the Mach number was set to $M_r = 0.1$) over an oscillating right circular cylinder was considered. The Reynolds number was set to 144, which leads to laminar vortex shedding from a stationary right circular cylinder of the type considered in Section 6.10.1. In this case, however, the cylinder is forced to oscillate sinusoidally in the transverse direction with an amplitude of 12% of the cylinder diameter at the Strouhal frequency of the shedding from a stationary cylinder with the same flow parameters. The Strouhal number of the forced oscillation was set at 0.1777 to match the value given in [GR74]. The position of the point at angular position $\theta$ on the surface of the cylinder at a given time $t$ is therefore given by

$$\boldsymbol{x}(\theta, t) = \left\{ \begin{array}{c} 0.5\cos\theta \\ 0.5\sin\theta + 0.12\sin(2\pi f t) \end{array} \right\} \tag{6.68}$$

where $f = 0.1777$ is the non-dimensional frequency, which is equal to the Strouhal number. In [GR74], it is noted that when the frequency of the forced oscillation is sufficiently close to the natural frequency of the laminar vortex shedding, the shedding frequency is "locked" to match the oscillation frequency. As for the previous cylinder cases the cylinder wall temperature was set equal to the farfield total temperature, and the dynamic viscosity and thermal conductivity were constants equal to their farfield values. Reported lengths are again non-dimensionalised by the diameter of the cylinder.

To account for the increased complexity of the flow field, the mesh used was somewhat finer than that used for the stationary cylinder considered previously. The target meshwidth was therefore set to $2.2 \times 10^{-3}$ on the cylinder surface, and line sources were again used for wake refinement. The target growth rate was set to 1.12. The grid set consisted of the base grid together with a single component grid containing the cylinder boundary. The target meshwidth at the farfield was set to 32 cylinder diameter. The base grid had 59 boundary points, 14,298 nodes, and 28,538 cells. The component grid had 1,428 points on the cylinder wall. The target radius of the component grid was set to four cylinder diameters. A section of the resulting grid set is shown on the left of Figure 6.8.

Since the frequency of the laminar vortex shedding is expected to be locked to the frequency of the oscillation, the physical time step can be selected based solely on the period of the forced oscillation. Since a more complex flow field than the stationary case is likely, the target number of physical time steps per oscillation period was set to 80, which corresponds to a non-dimensional physical time step of $2 \times 10^{-3}$. The Courant number of the pseudotime stepping was set to 4, which resulted in convergence to $\tilde{\Theta} = -3$ in an average of 63 pseudotime steps per physical time step. The solution computed by the steady solver for the $R_e = 50$ case was again used as the initial conditions for this case.

The results are shown on the right of Figure 6.8 as instantaneous $x$-velocity contours, and in Figure 6.9 as plots of the time-average of the velocity magnitude along transverse lines at several axial stations in the wake behind the oscillating cylinder against published experimental results. The former shows that the laminar vortices shed from the oscillating cylinder are well-resolved. The line plots show some areas of reasonable agreement between the solver and the experimental results. For example, there is good agreement in the time-average of the velocity magnitude at the centre of the wake and at two cylinder diameters from the centre of the wake in the transverse direction. However, the solver predicts that the velocity magnitude returns to its farfield value more quickly than the experimental results. This is likely due to the lack

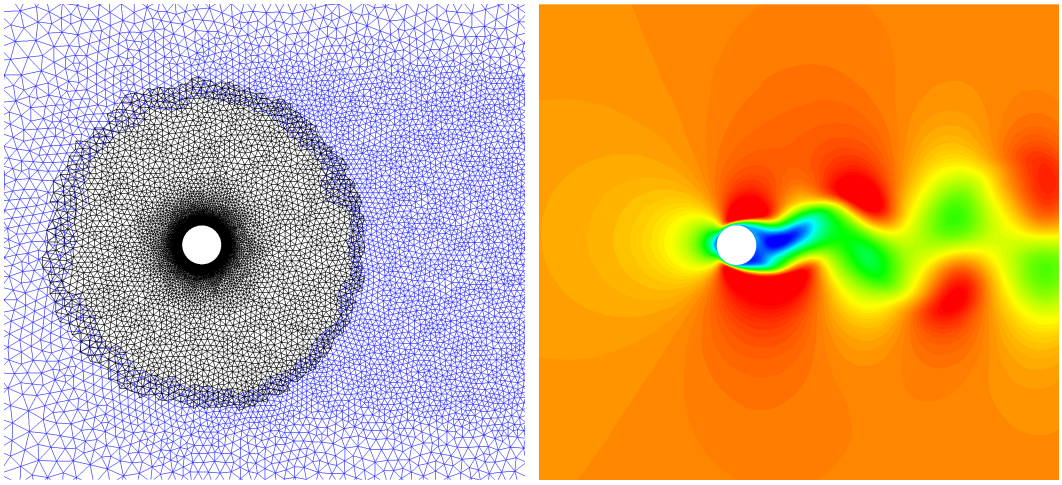


Figure 6.8: Left: a section of the grid set used for computing the flow over the oscillating cylinder at $M_r = 0.1$, $\alpha = 0°$, $R_e = 10,000$. The base grid is shown in blue, the component grid is shown in black, and elements initially tagged *disabled* are not shown. Right: the resulting $\boldsymbol{u} \cdot \boldsymbol{i}$ contours

of a turbulence model in the solver, which results in reduced momentum mixing and therefore less "spread" in the wake. This explanation is supported by the fact that the predictions of the present solver agree more closely with the reference results at locations further downstream of the cylinder, where significant decay of the turbulence will have occurred. In the final chapter, the possibility of adding turbulence models to the solver is very briefly discussed.

### 6.11.2 Dynamic Stall of the NACA 0012 Aerofoil

For the final test case, the dynamic stall behaviour of the NACA 0012 aerofoil flow is investigated. The Mach number and Reynolds number are set to 0.15 and 10,000 respectively. Initially the aerofoil is at an angle of attack of $\alpha = 5°$. The aerofoil pitches up about the quarter-chord point to a maximum angle of attack of $\alpha = 25°$, then returns to the initial of attack. The variation of angle of attack with respect to time follows the sinusoidal relation

$$\alpha(t) = \alpha_{\text{mean}} + \alpha_{\text{max}} \sin(2\pi f t + \phi) \tag{6.69}$$

where $\alpha_{\text{mean}} = 5°$, $\alpha_{\text{max}} = 10°$, $\phi = \frac{3\pi}{2}$, and $f$ is the pitching frequency. The pitching frequency is set to the value corresponding to a Strouhal number of $\frac{1}{2\pi}$. The temperature at the aerofoil wall is set equal to the farfield total temperature, whilst the

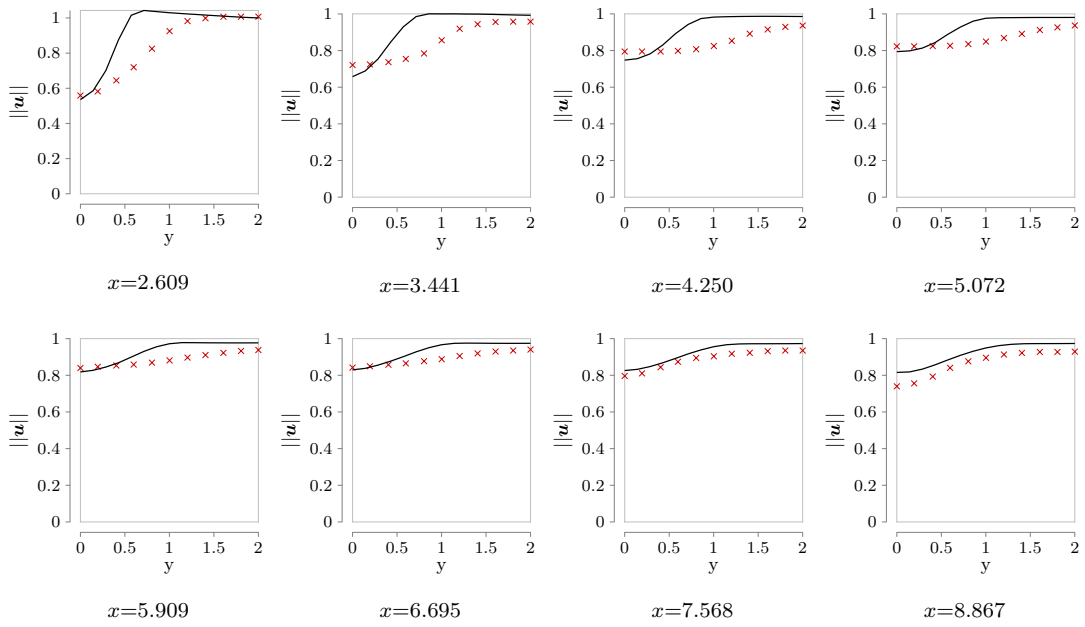



Figure 6.9: Plots of the time average of the non-dimensional velocity magnitude along transverse lines in the wake behind the oscillating cylinder at the indicated axial stations, against experimental results from [GR74] (shown as scatter points). The line $y = 0$ corresponds to the centre of the cylinder when no transverse displacement is applied.

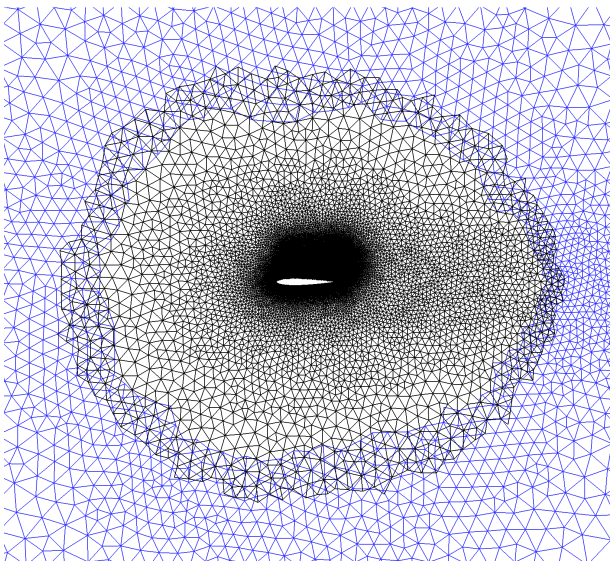


Figure 6.10: A section of the grid set used for the pitching NACA 0012 aerofoil after cutting the boundary and component grid holes. The base grid is shown in blue, the component grid is shown in black, and elements initially tagged *disabled* are not shown.

dynamic viscosity and thermal viscosity are constant and equal to their farfield values. Reported lengths are non-dimensionalised by the aerofoil chord.

Due to the large angle of attack, the flow is expected to be more complex than for the stationary geometry case considered in Section 6.10.2. As a result, a finer grid was obtained. The target meshwidth varied from $2.1 \times 10^{-3}$ to $3.2 \times 10^{-3}$ according to the boundary curvature, with the target meshwidth at the trailing edge set to the minimum value. As usual, line sources were used to increase the spatial resolution in the region of the expected wake. In addition, sources were used to improve the resolution in the neighbourhood of the top surface of the aerofoil, in the region where the stall behaviour of interest is expected. The grid set consisted of the base grid and one component grid, which contained the aerofoil boundary. The component grid was centered on the leading edge of the aerofoil, and had a target radius of four chords, so that the gradients of the flow variables were not too severe in the interpolation region. The target meshwidth at the farfield was set to 32 chords, which resulted in 67 boundary points on the farfield boundary. The component grid had 648 points on the aerofoil wall. The target growth rate across the grid set was set to 1.12. This lead to the base grid having $10,080$ nodes and $20,091$ cells, and the component grid having $25,211$ nodes and $49,641$ cells. A section of the grid set is shown in Figure 6.10.

The non-dimensional physical time step was set to $6.5 \times 10^{-4}$, which again cor-

responds to approximately 80 time steps per period (of the pitching motion). The Courant number of the pseudotime stepping was set to 12, which produced convergence to $\tilde{\Theta} = -3$ in an average of 103 pseudotime steps per physical time step. The initial conditions were obtained by performing a few hundred iterations with the steady solver (with the aerofoil at its initial angle of attack of 5°) before switching to the unsteady solver.

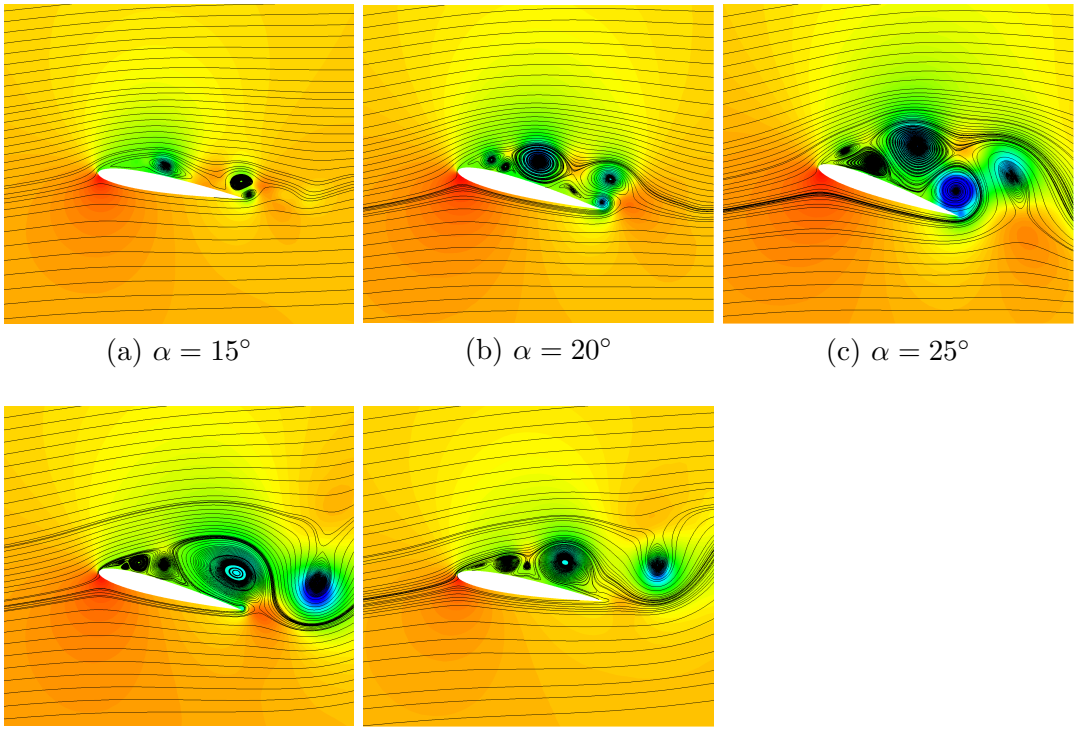
The results are shown as contours of instantaneous coefficient of pressure contours with overlaid streamlines in Figure 6.11 and as line plots of the loci of the lift and drag coefficients during one pitching cycle of the aerofoil in Figure 6.12. The former clearly shows the development of the well-known phenomenon of *hysteresis*, by which the loss of lift and increase of drag that occurs at stall persists even after the angle of attack of the aerofoil is reduced below the stall angle (see, for example, [McC82]). With reference to Figure 6.11, this phenomenon proceeds in the following way:

(a) As the angle attack increases, small vortices begin develop on the upper surface of the aerofoil.

(b), (c) As the angle of attack increases further the vortices continue to grow, and new vortices develop.

(d), (e) As the angle of attack reduces again, the vortices that have built up on the upper surface of the aerofoil do not disappear immediately. Instead they persist in-place, until the angle of attack has reduced sufficiently that the vortices detach from the aerofol and are swept downstream.

The same effect is shown in Figure 6.12, where the coefficient of lift and coefficient of drag of the aerofoil are plotted against the angle of attack during one pitching cycle. Also shown is published results from [SV08]. The plots are in good qualitative agreement. There are several areas of disagreement, however. For example, the published data shows a relatively stable gap in the coefficient of lift between the increasing angle of attack phase and the decreasing angle of attack phase, whereas the present solver predicts a much greater amount of fluctuation. Running the simulation over several cycles suggests a possible reason for the discrepancies between the published results and the results presented here – each cycle produces significantly different vortex structures, suggesting that the exact quantitative solution is strongly dependent on the initial conditions imposed. It is also likely that there are similarly strong dependencies on the exact discretisation employed as well on the grid and the selected time step.



(a) $\alpha = 15°$ (b) $\alpha = 20°$ (c) $\alpha = 25°$

(d) $\alpha = 20°$ (e) $\alpha = 15°$

Figure 6.11: Instantaneous coefficient of pressure contours and streamlines on the NACA 0012 in dynamic stall at the specified angles of attack, in chronological order from (a) to (e).

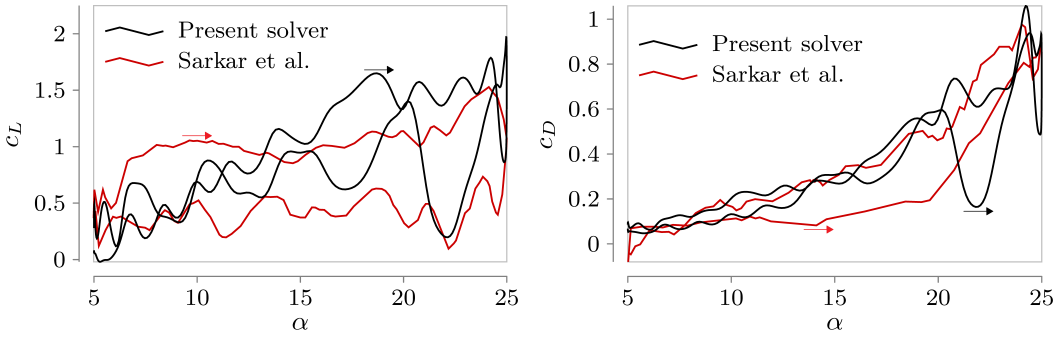



Figure 6.12: Hysteresis loops of the lift and drag coefficients for the NACA 0012 aerofoil in dynamic stall, compared to digitised results from [SV08]. Arrows indicate the direction of increasing time.

# Chapter 7

# Conclusion and Final Remarks

## 7.1 Achievements

The key achievements of the work presented in this thesis are now recapitulated.

**Inviscid Solver**

- An existing inviscid Mach-uniform method was selected and modified to exploit the special properties demanded of the grids to be used with the solver.
- A novel variant of the H-CUSP convective flux reconstruction method for the staggered arrangement of the primary flow variables employed by the solver was derived and implemented. Compared to the fully upwind reconstruction method, it was found to improve stability and convergence.
- Scalar and vector reconstruction schemes were considered. For each, several design choices needed to be made. These choices were carefully considered, before theoretically appropriate options were practically tested.
- The popular gradient limiting schemes of Barth and Jespersen [BJ89] and of Venkatakrishnan [Ven95] were modified to obtain forms suitable for use with a staggered scheme.
- A time step selection procedure [MJ90] from the literature was adapted for use with a staggered compressible solver.
- The inviscid solver was shown to be second-order spatially accurate.

- The inviscid solver was validated by comparison of numerical results against those published in the literature. Many of the results are in excellent agreement with the published results. Where this is not the case, there is generally also found to be significant disagreement between the published results.

**Viscous Solver**

- The inviscid normal momentum and pressure correction equations were extended to include viscous stress and heat flux terms. These made extensive use of the imposed grid requirements. These allowed a much simpler, more compact expression for the correction to the viscous stress tensor and a two-point approximation to the heat flux to be employed.
- A method for reconstructing the viscous stresses using the momentum field reconstruction procedure discussed in Chapter 3 was derived.
- Modifications to the vector reconstruction scheme to improve the accuracy and efficiency of the reconstruction along no-slip walls was introduced.
- The viscous solver was shown to be second-order spatially accurate, but not uniformly.
- The viscous solver was validated through the use of numerical examples. The first showed excellent agreement between the solver and published results. The second showed reasonable agreement with a published DNS study and the appearance of laminar vortex shedding at approximately the correct Reynolds number. The third numerical example showed excellent agreement with published results, particularly in the calculation of the skin friction coefficient along the no-slip wall.

**Unsteady Solver**

- The solver algorithm was extended through the addition of a variant of the dual time stepping procedure, to allow time-accurate unsteady solutions despite the need for linearisation.
- The approximation to the (physical) time derivative was replaced by a second-order accurate approximation, and the continuity, normal momentum, and pressure correction equations were modified accordingly.

- A variant of the chimera method was developed and implemented.
- The wall boundary conditions were modified to allow the modelling of moving boundaries.
- The unsteady solver was shown to be second-order spatially and temporally accurate, but not uniformly.
- The unsteady solver was validated through numerical examples. The results were compared against a mixture of computational and experimental results from the literature, and showed reasonable agreement. In particular, the computed time-averaged drag coefficients of the cylinder in the laminar shedding regime where close to the results of a published DNS study. The robustness of the unsteady solver was also demonstrated through the successful simulation of the dynamic stall behaviour of an aerofoil in the quasi-incompressible regime.

## 7.2 Suggestions for Future Development

Finally, several directions of possible further research and development of the solver are suggested and briefly discussed.

### 7.2.1 Extension to Three Dimensions

The most obvious extension of the solver is an increase in the number of spatial dimensions from two to three. Due to the use of the normal momentum as one of the primary flow variables rather than the grid-aligned momentaums, this would not require the addition of another conservation equation. The only major change required to the solver would be the modification of the scalar, vector, and viscous stress reconstructions. Of these, reconstruction of cell-centre scalars in three dimensions is well covered in the literature [Bla01], whilst reconstruction of viscous stresses in three dimensions using the method discussed in Chapter 5 follows easily once the vector field reconstruction is accomplished, simply by adding additional components to the viscous stress tensor and an additional term to the flow velocity divergence in the definition of the on-diagonal components. Only the extension of the vector reconstruction to three dimensions then remains, and presents the biggest challenge. Extending the same momentum reconstruction procedure presented in this thesis to the three-dimensional case results in a reconstruction polynomial with twelve unknown coefficients, of which one can be eliminated by imposing a condition on the divergence of the reconstruction

polynomial in an analogous way to (3.63). As a result, each node-based reconstruction stencil must contain at least eleven faces. In order to avoid large reconstruction stencils and the resulting inaccuracy and instability, it is desirable that all interior nodes have enough attached faces to form a maximal rank reconstruction stencil without additional faces being required. Obtaining three dimensional grids that satisfy this requirement is difficult in practice, and it is not clear that it is possible in general. This problem must be overcome in order to successfully extend the solver to three dimensional geometries, either through improvements in grid generation and optimisation algorithms or through modification of the momentum reconstruction procedure.

### 7.2.2 Multigrid

In analysing CFD solvers, it is found that the low frequency errors present in the solution at a particular iteration or pseudotime step decay more slowly than the high frequency errors [Bra77]. In the introduction to Section 5.9, it was briefly mentioned that decay of the low frequency errors, and therefore the convergence of the scheme, can be accelerated through the use of *multigrid*. In the most basic variant of this method, the grid on which the solution is to be obtained is accompanied by several grids of the same geometry of increasing coarseness. Convergence is started on the finest grid. After several iterations, the solution is restricted to the next-coarser grid and convergence continues. Through this process the low frequency errors become higher frequency (because the error waves fit within fewer cells on the coarser grid than on the finer grid), and are therefore expected to decay faster. The process of restricting to a coarser grid can be done several times. Once satisfactory decay of the lowest frequency errors has been achieved, the solution is progressively interpolated back up the grid hierarchy to the finest grid. Numerous variants of this basic approach are described in the literature.

The multigrid method as described above is also known as *geometric* multigrid, and is not easily applied to the current solver due to the somewhat restrictive grid requirements. Instead, the newer *algebraic* multigrid [Fal06] would be suitable. This follows the same basic principle, except the restriction and interpolation operations are carried out directly on the implicit operator of the linear systems without reference to the underlying grid structure.

As seen in the examples presented throughout this thesis, the number of iterations required for convergence is somewhat high. This is largely due to the solver being segregated rather than coupled, which reduces the maximum Courant number. Algebraic

multigrid should significantly reduce the required number of iterations for convergence, at only modest or moderate computational cost. The solver presented in this thesis would therefore likely benefit greatly from the addition of multigrid.

### 7.2.3 Third- and Higher-Order Spatial Accuracy

A third-order spatially accurate method requires piecewise quadratic reconstruction of the scalar and vector primary flow variables. Piecewise quadratic reconstruction of cell-centre scalar flow variables is well-covered in the literature, and requires at least six points in the reconstruction stencil to uniquely determine the six reconstruction coefficients. Similarly, piecewise quadratic reconstruction of staggered vector fields is discussed in [Vid08], and requires at least eleven edges in the reconstruction stencil to uniquely determine the twelve reconstruction coefficients (one reconstruction coefficient is eliminated by the condition on the divergence of the reconstruction polynomial). In general, neither of these requirements are met by the nodes and cells of the two dimensional grids considered in this thesis. The required reconstruction stencils would therefore need to be extended beyond the cells and edges immediately adjacent to the base node or cell. As well as stability problems caused by the large reconstruction stencils, the limiting of quadratic reconstruction polynomials is also not straightforward. Extension to even higher orders of accuracy is possible in principle, although the aforementioned problems will become even more acute.

### 7.2.4 Turbulence Modelling

In this work, laminar flow was assumed. The great majority of flows of practical interest in engineering, however, include regions of turbulent flow [Dav15]. This is certainly true of the motivating examples given when discussing the need for a Mach-uniform solver during the introduction to this thesis: rotorcraft in hover or low-speed flight, flow through combustors in gas turbine engines, propeller aircraft during slow flight, and jet aircraft from start-up to take-off and climb. The inclusion of turbulence modelling in the solver is therefore a priority in future development of the solver. There are hundreds of turbulence models described in the literature [Pop00]. Careful research and testing would be required in the selection of a suitable turbulence model (or turbulence models), and there is therefore little that can be said about the issues surrounding the addition of turbulence models to the solver here. One issue that will be problematic regardless of which turbulence model is selected, however, is the generation of suitable boundary layer meshes near walls. Typically, high-aspect-ratio prismatic elements are

placed in these regions so that sufficient resolution across the thickness of the boundary layer is obtained without an excessive number of cells being generated in the tangential direction. However, this approach conflicts with the special grid requirements outlined in Chapter 3. This will likely be the major obstacle in the extension of the solver to turbulent flow.

# Bibliography


[And10] J.D. Anderson. *Fundamentals of Aerodynamics*. Fifth. McGraw-Hill, 2010. ISBN: 978-0-0707-0012-3.

[Ang+89] F Angrand et al. "Implicit Euler calculations using a Galerkin finite element approximation on adapted non-structured meshes." In: *Numerical Simulation of Compressible Euler Flows*. Vieweg, 1989, pp. 54–75.

[Ang87] F Angrand. "Viscous perturbation for the compressible Euler equations. Application to the numerical simulation of compressible viscous flows". In: *Numerical Simulation of Compressible Navier-Stokes Flows*. Vieweg, 1987, pp. 69–84.

[AWH13] G.B. Arfken, H.J. Weber, and F.E. Harris. *Mathematical Methods for Physicists*. Seventh. 2013. ISBN: 978-0-12-384654-9.

[Bas+87] F. Bassi et al. "Solution of the compressible Navier-Stokes equations for a double throat nozzle". In: *Numerical Simulation of Compressible Navier-Stokes Flows*. Vieweg, 1987, pp. 237–254.

[Ber+00] M. de Berg et al. *Computational Geometry: Algorithms and Applications*. Second Revised. Berlin: Springer-Verlag, 2000. ISBN: 3-540-65620-0.

[BG03] A. Ben-Israel and T.N.E. Greville. *Generalised inverses: Theory and applications*. Second. New York, NY, USA: Springer, 2003.

[BJ89] T. Barth and D. Jespersen. "The design and application of upwind schemes on unstructured meshes." In: *27th Aerospace Sciences Meeting*. 1989. DOI: 10.2514/6.1989-366.

[Bla01] J. Blasek. *Computational Fluid Dynamics: Principles and Applications*. Oxford, UK: Elsevier, 2001. ISBN: 0-08-043009-0.

[BM89] M. Borrel and J.L. Montagne. “Upwind second-order unsteady scheme.” In: *Numerical Simulation of Compressible Euler Flows*. Vieweg, 1989, pp. 76–87.

[Bra77] A. Brandt. “Multi-level adaptive solutions to boundary-value problems.” In: *Mathematics of Computation* 31.138 (1977), pp. 333–390.

[Bri+87a] M.O. Bristeau et al., eds. *Numerical Simulation of Compressible Navier-Stokes Flows*. Vol. 18. Notes on Numerical Fluid Mechanics. Braunschweig, Germany: Vieweg, 1987. ISBN: 3-528-08092-2.

[Bri+87b] M.O. Bristeau et al. “Solution of the compressible Navier-Stokes equations by least-squares and finite element methods”. In: *Numerical Simulation of Compressible Navier-Stokes Flows*. Vieweg, 1987, pp. 85–104.

[BW01] A. Beckert and H. Wendland. “Multivariate interpolation for fluid-structure-interaction problems using radial basis functions.” In: *Aerospace Science and Technology* 5.2 (Feb. 2001), pp. 125–134.

[BWM15] A. Balan, M. Woopen, and G. May. “Hp-adaptivity on anisotropic meshes for hybridized discontinuous Galerkin scheme.” In: *22nd AIAA Computational Fluid Dynamics Conference*. 2015. DOI: 10.2514/6.2015-2606.

[Cam87] L. Cambier. “Computation of viscous transonic flows using an unsteady type method and a zonal grid refinement technique”. In: *Numerical Simulation of Compressible Navier-Stokes Flows*. Vieweg, 1987, pp. 105–122.

[Car92] J. Carroll. “Sufficient conditions for uniformly second-order convergent schemes for stiff initial-value problems”. In: *Computers and Mathematics with Applications* 24.10 (1992), pp. 105–116. ISSN: 0898-1221. DOI: 10.1016/0898-1221(92)90023-B.

[CDS13] S.W. Cheng, T.K. Dey, and J.R. Shewchuk. *Delaunay Mesh Generation*. Boca Raton, USA: CRC Press, 2013. ISBN: 978-1-58488-730-0.

[Cha+89] F. Chalot et al. “Calculation of two-dimensional compressible Euler flows with a new Petrov-Galerkin finite element method.” In: *Numerical Simulation of Compressible Euler Flows*. Vieweg, 1989, pp. 88–104.

[CT15] D. Canuto and K. Taira. “Two-dimensional compressible viscous flow around a circular cylinder”. In: *Journal of Fluid Mechanics* 785 (2015), pp. 349–371. DOI: 10.1017/jfm.2015.635.

[CV89] V. Couaillier and J.P. Veuillot. "Multigrid scheme for the Euler equations." In: *Numerical Simulation of Compressible Euler Flows*. Vieweg, 1989, pp. 105–121.

[DA10] M.H. Djavareshkian and M.H. Abdollahi Jahdi. "Shock-capturing method using characteristic-based dissipation filters in pressure-based algorithm." In: *Acta Mechanica* 209.99 (2010).

[Dad89] A. Dadone. "Computation of transonic steady flows using a modified lambda formulation." In: *Numerical Simulation of Compressible Euler Flows*. Vieweg, 1989, pp. 122–137.

[Dav15] P.A. Davidson. *Turbulence: An Introduction for Scientists and Engineers*. second. UK: Oxford University Press, 2015. ISBN: 978-0-18-872259-5.

[DDF89] A. Dervieux, J.A. Desideri, and F. Fezoui. "Euler calculations by upwind finite element methods and adaptive mesh algorithms." In: *Numerical Simulation of Compressible Euler Flows*. Vieweg, 1989, pp. 138–156.

[De 93] D.L. De Zeeuw. "A quadtree-based adaptively refined Cartesian-grid algorithm for solution of the Euler equations." PhD thesis. University of Michigan, 1993.

[Del96] M. Delanaye. "Polynomial reconstruction finite volume schemes for the compressible Euler and Navier-Stokes equations on unstructured adaptive grids." PhD thesis. Universite de Liege, 1996.

[Der+89] A. Dervieux et al., eds. *Numerical Simulation of Compressible Euler Flows*. Vol. 26. Notes on Numerical Fluid Mechanics. Braunschweig, Germany: Vieweg, 1989. ISBN: 3-528-07626-7.

[DG02] Q. Du and M. Gunzburger. "Grid generation and optimization based on centroidal Voronoi tessellations." In: *Applied Mathematics and Computation* 133.2 (2002), pp. 591–607. DOI: 10.1016/S0096-3003(01)00260-0.

[DG95] C.R. Doering and J.D. Gibbon. *Applied Analysis of the Navier-Stokes Equations*. UK: Cambridge University Press, 1995. ISBN: 0-521-44557-4.

[DLP93] I. Demirdžić, Ž. Lilek, and M. Perić. "A collocated finite volume method for predicting flows at all speeds." In: *International Journal for Numerical Methods in Fluids* 16.12 (June 1993), pp. 1029–1050.

[EA89] A. Ecer and H.U. Akay. “Computation of steady Euler equations using finite element method.” In: *Numerical Simulation of Compressible Euler Flows*. Vieweg, 1989, pp. 157–174.

[Fal06] R.D. Falgout. “An introduction to algebraic multigrid.” In: *Computing in Science and Engineering* 8.6 (Nov. 2006), pp. 24–33.

[FG08] P.J. Frey and P.L. George. *Mesh Generation: Application to Finite Elements*. second. UK: Wiley, 2008. ISBN: 978-1-84821-029-5.

[For18] S. Fortune. “Voronoi Diagrams and Delaunay Triangulations”. In: *Handbook of Discrete and Computational Geometry*. Ed. by J.E. Goodman, J. O’Rourke, and C.D. Tóth. CRC Press, 2018. ISBN: 978-1-4987-1139-5.

[FP02] J.H. Ferziger and M. Perić. *Computational Methods for Fluid Dynamics*. third. Germany: Springer-Verlag, 2002. ISBN: 3-540-42074-6.

[Gam+95] E. Gamma et al. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995. ISBN: 0-201-63361-2.

[GDT85] M.B. Giles, M. Drela, and W.T. Thompkins. “Newton solution of direct and inverse transonic Euler equations.” In: *AIAA 7th Computational Physics Conference*. Cincinnati, OH, USA, 1985.

[GJM87] F. Grasso, A. Jameson, and L. Martinelli. “A multistage multigrid method for the compressible Navier-Stokes equations.” In: *Numerical Simulation of Compressible Navier-Stokes Flows*. Vieweg, 1987, pp. 123–138.

[GR74] O.M. Griffin and S.E. Ramberg. “The vortex street wakes of vibrating cylinders.” In: *Journal of Fluid Mechanics* 66.3 (1974), pp. 553–576. DOI: 10.1017/S002211207400036X.

[Gus75] B. Gustafsson. “The convergence rate for difference approximations to mixed initial boundary value problems.” In: *Mathematics of Computation* 29.130 (Apr. 1975), pp. 396–406.

[GW22] C.J. Greenshields and H.G. Weller. *Notes on Computational Fluid Dynamics: General Principles*. CFD Direct Limited, 2022. ISBN: 978-1-3999-2078-0.

[Haa87] W. Haase. “Solutions of the Navier-Stokes equations for sub- and supersonic flows in rarefied gases.” In: *Numerical Simulation of Compressible Navier-Stokes Flows*. Vieweg, 1987, pp. 139–157.

[HC] D. Holmes and S. Connell. "Solution of the 2D Navier-Stokes equations on unstructured adaptive grids." In: *AIAA 9th Computational Fluid Dynamics Conference*. Buffalo, NY, USA. DOI: 10.2514/6.1989-1932.

[HF23] J. Holman and J. Fürst. "Rotated-hybrid Riemann solver for all-speed flows." In: *Journal of Computational and Applied Mathematics* 427 (2023), pp. 115–129. DOI: https://doi.org/10.1016/j.cam.2023.115129.

[Hir07] C. Hirsch. *Numerical Computation of Internal and External Flows*. Second. Vol. 1. Elsevier, 2007. ISBN: 978-0-7506-6594-0.

[HK89] P.W. Hemker and B. Koren. "A non-linear multigrid method for the steady Euler equations." In: *Numerical Simulation of Compressible Euler Flows*. Vieweg, 1989, pp. 175–195.

[HW11] G. Hager and G. Wellein. *Introduction to High Performance Computing for Scientists and Engineers*. CRC Press, 2011. ISBN: 978-1-4398-1192-4.

[HW65] F.H. Harlow and J.E. Welch. "Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface." In: *The Physics of Fluids* 8.12 (Dec. 1965), pp. 2182–2189. DOI: 10.1063/1.1761178.

[Ill50] C.R. Illingworth. "Some soltuions of the equations of flow of a viscous compressible fluid". In: *Mathematical Proceedings of the Cambridge Philosophical Society* 46.3 (1950), pp. 469–478. DOI: 10.1017/S0305004100025986.

[Jam91] A. Jameson. "Time dependent calculations using multigrid, with applications to unsteady flows past airfoils and wings." In: *AIAA 10th Computational Fluid Dynamics Conference* (1991).

[Jam93] A. Jameson. "Artifical diffusion, upwind biasing, limiters and their effect on accuracy and multigrid convergence in transonic and hypersonic flow." In: *11th Computational Fluid Dynamics Conference*. 93-3559. 1993. DOI: 10.2514/6.1993-3359.

[Kor87] W. Kordulla. "Using an unfactored implicit predictor-corrector method." In: *Numerical Simulation of Compressible Navier-Stokes Flows*. Vieweg, 1987, pp. 165–182.

[KR89] B. Krouthén and A. Rizzi. "Numerical solutions to the Euler equations for the 1986 GAMM workshop." In: *Numerical Simulation of Compressible Euler Flows*. Vieweg, 1989, pp. 197–226.

[Kre70] H.O. Kreiss. "Initial boundary value problems for hyperbolic systems." In: *Communications on Pure and Applied Mathematics* 23.3 (1970), pp. 277–298. DOI: 10.1002/cpa.3160230304.

[KVB87] D. Kalfon, D. Volpert, and O. Brocard. "A finite element method for solving Navier-Stokes equations." In: *Numerical Simulation of Compressible Navier-Stokes Flows*. Vieweg, 1987, pp. 158–164.

[Lan87] S. Lang. *Calculus of Several Variables*. Third. New York, US: Springer-Verlag, 1987. ISBN: 0-387-96405-3.

[Lan98] C.B. Laney. *Computational Gasdynamics*. Cambridge, UK: Cambridg University Press, 1998. ISBN: 978-0-521-62558-6.

[Lev02] R.J. Leveque. *Finite-Volume Methods for Hyperbolic Problems*. Cambridge, UK: Cambridge University Press, 2002. ISBN: 0-521-00924-3.

[Lie66] J.H. Lienhard. *Synopsis of lift, drag, and vortex frequency data for rigid circular cylinders.* Tech. rep. 300. Pullman, Washington, US: Technical Extension Service, Washington State University, 1966.

[Lio96] M.S. Liou. "Approximate Riemann solvers, parameter vectors, and difference schemes." In: *Journal of Computational Physics* 129.2 (1996), pp. 364–382.

[Lis10] V.D. Liseikin. *Grid Generation Methods*. second. Springer, 2010. ISBN: 978-90-481-2912-6.

[LR57] H.W. Liepmann and A. Roshko. *Elements of Gasdynamics*. John Wiley and Sons, 1957. ISBN: 0-471-53460-9.

[LS89] A. Lerat and J. Sidès. "Implicit transonic calculations without artificial viscosity or upwinding." In: *Numerical Simulation of Compressible Euler Flows*. Vieweg, 1989, pp. 227–250.

[MBR87] B. Müller, T. Berglind, and A. Rizzi. "Implicit central difference simulation of compressible Navier-Stokes flow over a NACA0012 airfoil." In: *Numerical Simulation of Compressible Navier-Stokes Flows*. Vieweg, 1987, pp. 183–200.

[McC82] W.J. McCroskey. "Unsteady airfoils." In: *Annual Review of Fluid Mechanics* 14.1 (1982), pp. 285–311. DOI: 10.1146/annurev.fl.14.010182.001441.

[MJ90] D. J. Mavriplis and A. Jameson. "Multigrid solution of the Navier-Stokes equations on triangular meshes." In: *AIAA Journal* 28.8 (1990), pp. 1415–1425. DOI: 10.2514/3.25233.

[ML89] G. Moretti and A. Lippolis. "Transonic airfoil and intake calculations." In: *Numerical Simulation of Compressible Euler Flows*. Vieweg, 1989, pp. 250–274.

[Mor+89] K. Morgan et al. "A finite element scheme for the Euler equations." In: *Numerical Simulation of Compressible Euler Flows*. Vieweg, 1989, pp. 275–291.

[MS08] K Mehlhorn and P. Sanders. *Algorithms and Data Structures: The Basic Toolbox*. Berlin, Germany: Springer-Verlag, 2008. ISBN: 978-3-540-77977-3.

[Nov22] K. Novak. *Numerical Methods for Scientific Computing*. Second. Arlington, Virginia, USA: Equal Share Press, 2022. ISBN: 9798985421804.

[ORo98] J. O'Rourke. *Computational Geometry in C*. Cambridge, UK.: Cambridge University Press, 1998. ISBN: 978-0521649766. DOI: 10.1017/CBO9780511804120.

[PA18] C. Pekardan and A. Alexeenko. "Rarefaction effects for transonic airfoil flows at low Reynolds numbers." In: *AIAA Journal* 56.2 (2018), pp. 765–779. DOI: 10.2514/1.J056051.

[PC93] Dartzi Pan and Jen-Chieh Cheng. "A second-order upwind finite-volume method for the Euler solution on unstructured triangular meshes." In: *International Journal for Numerical Methods in Fluids* 16.12 (June 1993), pp. 1079–1098.

[PLA89] M. Pandolfi, F. Larocca, and T.T. Ayele. "A contribution to the numerical prediction of transonic flows." In: *Numerical Simulation of Compressible Euler Flows*. Vieweg, 1989, pp. 292–308.

[Pop00] S.B. Pope. *Turbulent Flows*. Cambridge, UK.: Cambridge University Press, 2000. ISBN: 978-0-521-59886-6.

[Rei07] J. Reinders. *Intel Threading Building Blocks: Outfitting C++ for Multi-Core Processor Parallelism*. O'Reilly, 2007. ISBN: 978-0-596-51480-8.

[RG14] A.I. Ruban and J.S.B. Gajjar. *Fluid Dynamics Part 1: Classical Fluid Dynamics*. St. Ives, UK.: Oxford University Press, 2014. ISBN: 978-0-19-968173-0.

[RM92] G. Rogers and Y. Mayhew. *Engineering Thermodynamics: Work and Heat Transfer*. Fourth. Pearson, 1992. ISBN: 978-0-582-04566-8.

[Roe81] P. Roe. "Approximate Riemann solvers, parameter vectors, and difference schemes." In: *Journal of Computational Fluids* 43.2 (1981), pp. 357–372.

[Sam82] H. Samet. "Neighbor finding techniques for images represented by quadtrees." In: *Computer Graphics and Image Processing* 18.1 (1982), pp. 37–57. DOI: 10.1016/0146-664X(82)90098-3.

[SDN87] Y. Secretan, G. Dhatt, and D. Nguyen. "Compressible viscous flow around a NACA-0012 airfoil." In: *Numerical Simulation of Compressible Navier-Stokes Flows*. Vieweg, 1987, pp. 219–236.

[Sha20] J. Shaif. "The future of computing beyond Moore's law." In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 378.2166 (2020). DOI: 10.1098/rsta.2019.0061.

[SM89] N. Satofuka and K. Morinishi. "Solution of compressible Euler flows using rational Runge-Kutta time stepping scheme." In: *Numerical Simulation of Compressible Euler Flows*. Vieweg, 1989, pp. 309–330.

[SMN87] N. Satofuka, K. Morinishi, and Y. Nishida. "Numerical solution of two-dimensional compressible Navier-Stokes equations using rational Runge-Kutta method." In: *Numerical Simulation of Compressible Navier-Stokes Flows*. Vieweg, 1987, pp. 201–218.

[Sut93] W. Sutherland. "LII. The viscosity of gases and molecular force". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 36.223 (1893), pp. 507–531. DOI: 10.1080/14786449308620508.

[SV08] S. Sarkar and K. Venkatraman. "Influence of pitching angle on the dynamic stall behavior of a symmetric aerofoil." In: *European Journal of Mechanics - B/Fluids* 27.3 (2008), pp. 219–238. DOI: 10.1016/j.euromechflu.2006.07.004.

[TA05] S. Tu and S. Aliabadi. "A slope limiting procedure in discontinuous Galerkin finite element for gasdynamics applications." In: *International Journal of Numerical Analysis and Modelling* 2 (Jan. 2005).

[TJS03] H.S. Tang, S.C. Jones, and F. Sotiropoulos. "An overset-grid method for 3D unsteady incompressible flow." In: *Journal of Computational Physics* 191.2 (2003), pp. 567–600. DOI: 10.1016/S0021-9991(03)00331-0.

[TLS70] Y.S. Touloukian, P.E. Liley, and S.C. Saxena. *Thermophysical Properties of Matter*. Vol. 3. New York, NY, USA: Plenum Publishing Corporation, 1970. ISBN: 0-306-67023-2.

[TMJ95] S. Tatsumi, L. Martinelli, and A. Jameson. "A new high resolution scheme for compressible viscous flow with shocks." In: *33rd Aerospace Sciences Meeting and Exhibit*. 95-0466. 1995. DOI: 10.2514/6.1995-466.

[TSH75] Y.S. Touloukian, S.C. Saxena, and P. Hestermans. *Thermophysical Properties of Matter*. Vol. 11. New York, NY, USA: Plenum Publishing Corporation, 1975. ISBN: 0-306-67031-3.

[TWV89] J.L. Thomas, R.W. Walters, and B. Van Leer. "Implicit finite-volume algorithms for the flux-split Euler equations." In: *Numerical Simulation of Compressible Euler Flows*. Vieweg, 1989, pp. 331–347.

[Usa83] W.J. Usab. "Embedded mesh solutions of the Euler equation using a multiple-grid method." PhD thesis. Massachusetts Institute of Technology, 1983.

[Ven93] V Venkatakrishnan. "On the accuracy of limiters and convergence to steady state solutions". In: *31st Aerospace Sciences Meeting*. 1993. DOI: 10.2514/6.1993-880.

[Ven95] V Venkatakrishnan. "Convergence to steady state solutions of the Euler equations on unstructured grids with limiters". In: *Journal of Computational Physics*. 118.1 (1995), pp. 120–130.

[Vid08] D. Vidović. "Polynomial reconstruction of staggered unstructured vector fields." In: *Theoretical Applied Mechanics* 36.2 (2008), pp. 85–99.

[VSW06] D. Vidović, A. Segal, and P. Wesseling. "A superlinearly convergent Mach-uniform finite volume method for the Euler equations on staggered unstructured grids". In: *Journal of Computational Physics* 217 (2006), pp. 27–294. DOI: 10.1016/j.jcp.2006.01.031.

[Wen01] I. Wenneker. "Conservation properties of a new unstructured staggered scheme." In: *Computers and Fluids* 32 (2001), pp. 139–147.

[Whi06] F.M. White. *Viscous Fluid Flow*. Third. New York, USA: McGraw Hill, 2006. ISBN: 978-0071244930.

[Whi83] D.L. Whitfield. *Three-dimensional unsteady Euler equation solutions using flux vector splitting*. Tech. rep. CR-173254. NASA, 1983.

[WHM13] S. Walton, O. Hassan, and K. Morgan. "Reduced order mesh optimisation using proper orthogonal decomposition and a modified cuckoo search." In: *International Journal for Numerical Methods in Engineering* 93.3 (2013), pp. 527–550. DOI: 10.1002/nme.4400.

[WHM17] S. Walton, O. Hassan, and K. Morgan. "Advances in co-volume mesh generation and mesh optimisation techniques." In: *Computers and Structures* 181 (2017), pp. 70–88. DOI: 10.1016/j.compstruc.2016.06.009.

[WSW02] I. Wenneker, A. Segal, and P. Wesseling. "A Mach-uniform unstructured staggered grid method". In: *International Journal for Numerical Methods in Fluids* 40 (2002), pp. 1209–1235. DOI: 10.1002/fld.417.

[Y03] Saad Y. *Iterative Methods for Sparse Linear Systems*. Second. Society of Industrial and Applied Mathematics, 2003. ISBN: 978-0-898-715347.

[ZNG08] X. Zhang, S. Ni, and He. G. "A pressure-correction method and its application on an unstructured Chimera grid." In: *Computers and Fluids* 37.8 (2008), pp. 993–1010.

# Appendix A

# Design Choices

## A.1 Introduction

At various points in the main body of the thesis, design choices need to be made. These are mentioned only briefly in the main body, with fuller discussions included here.

## A.2 Chapter 3 - Foundations of the Inviscid Solver

### A.2.1 Approximation of the Volume Integrals

In Section 3.6, it is pointed out that the decision not to store the primary flow variables at the centroids of the control volume means that the simplest volume integral approximation method of assuming that the value of the primary flow variable is constant across the control volume and equal to the stored value no longer reproduces the exact value of the integral when the primary flow variable varies linearly across the domain. As a result, this approximation method is no longer sufficient for a formally second order accurate scheme. There are two methods for handling this error

1. If the cells are close to being equilateral triangles, the centroids of the control volumes will be 'near' the point where the value of $\varphi$ is actually stored. Thus, the loss of accuracy by ignoring the distinction between the two points could be expected to be small.

2. A simple linear interpolation of $\varphi$ to the control volume centroid using a gradient reconstruction of the type discussed in Section 3.8 and Section 3.9 could be used to approximate the value at the centroid in a way that is exact in the case of a linear variation. This has the disadvantage of increasing the size of the stencil

from a single element to the element and its surrounding elements, though it can be expected that most (if not all) of these elements are already in the stencil due to the convection terms.

In practice it was found that even for good-quality grids an appreciable drop in accuracy occurred using the first option. On the other hand, there was found to be little or no increase in stencil size and computational cost using the second option, and so this was the solution used for the numerical examples.

### A.2.2 Discretisation of the Pressure Term in the Normal Momentum Equation

During the discretisation of the normal momentum equation in Section 3.6.2, an approximation of the pressure gradient normal to an edge $i$ is required. There are two choices for this approximation

1. Discretise the pressure term as a surface integral using the same discretisation scheme as for the convective terms

$$T_{p_i} = -\sum_{e(i)} \left\{ \bar{l}_e p_e^{n+1} \right\} \tag{A.1}$$

2. Use the divergence theorem (see [Lan87, p. 345] for example) to transform the pressure term to a volume integral. This results in

$$\bar{T}_{p_i} = -\int_{\Omega} \nabla p \cdot \boldsymbol{n}_i d\Omega \tag{A.2}$$

where $\bar{T}_{p_i}$ represents the exact (undiscretised) pressure term. During the discussion of the discretisation of volume integrals at the beginning of this section, it was pointed out that the common volume integral discretisation method of assuming the variable being integrated is constant over the volume and equal to its centroidal value could not be directly applied in the scheme being discussed here, since the centroidal value is not available. The solution given for this issue was to use piecewise linear interpolation to interpolate the centroidal value, and it was explained that this was necessary to obtain a formally second-order-accurate solver. In this case, however, this is not necessary because the volume integral of a *derivative* is being discretised. If the exact pressure field is linear then the normal pressure gradient field will be constant, and so second-order accuracy is

retained even when replacing the value at the centroid of the control volume with the value at the edge centre. The discretised pressure term is therefore

$$T_{p_i} = -\Omega_i \nabla p\big|_i \cdot \boldsymbol{n}_i \tag{A.3}$$

As discussed in Section 2.2, the properties of the grids being used allow this term to take the particularly simple form

$$T_{p_i} = \frac{\Omega_i \left(p_\ell - p_r\right)}{\left\|\boldsymbol{x}_\ell - \boldsymbol{x}_r\right\|_2} \tag{A.4}$$

where $\boldsymbol{x}$ denotes the circumcentre location. As explained in Section 3.2, $p_r$ refers to the pressure at the circumcentre of the cell adjacent to edge $i$ into which the unit normal $\boldsymbol{n}_i$ points, and $p_\ell$ refers to the pressure at the circumcentre of the other adjacent cell.

Due to its particularly simple form and the fact that numerical experiments showed good suppression of odd-even decoupling, the second option for discretisation of the pressure term was chosen.

### A.2.3 The Tangential Momentum Correction

The discretisation of the pressure correction equation requires an expression for the correction of the momentum due to the pressure correction. In Section 3.6.3, the normal momentum correction is derived directly from the normal momentum equation. To complete the momentum correction, an expression for the tangential momentum correction is also required. This cannot be obtained in the same way as the normal momentum correction, however, since the scheme does not enforce conservation of tangential momentum through the application of a tangential momentum equation. Two alternative methods were considered.

1. Neglect the tangential momentum correction, and assume momentum correction results from the normal momentum correction only. The resulting momentum correction is
$$\delta \boldsymbol{m}_i^{n+1} = -\left[\Delta t_i \nabla \delta p\big|_i^{n+1} \cdot \boldsymbol{n}_i\right] \boldsymbol{n}_i \tag{A.5}$$

2. Assume the tangential momentum correction takes the same form as the normal momentum correction. The resulting momentum correction is
$$\delta \boldsymbol{m}_i^{n+1} = -\Delta t_i \nabla \delta p\big|_i^{n+1} \tag{A.6}$$

During the derivation of the pressure correction equation from the energy equation, it will be found that the momentum correction is required both at the edge centres and at the cell circumcentres. Since normals and tangents are only defined along the edges, only the second momentum correction formula can be directly applied. For this reason, (3.33) will be chosen as the momentum correction.

### A.2.4 Edge Centre Piecewise Linear Normal Momentum Reconstruction

In Section 3.9.3, the reconstruction of the convected normal momentum $\boldsymbol{m}_e \cdot \boldsymbol{n}_i$ during the solution of the normal momentum equation for edge $i$ is discussed. There are two sensible choices for performing this reconstruction.

1. Reconstruct the full momentum vector, as is done for piecewise constant momentum reconstruction. That is, if $\bar{\boldsymbol{m}}_e$ is the reconstructed momentum at the centre of edge $e$, then the convected normal momentum is approximated by

$$\boldsymbol{m}_e \cdot \boldsymbol{n}_i \approx \bar{\boldsymbol{m}}_e \cdot \boldsymbol{n}_i \tag{A.7}$$

2. Reconstruct only the tangential component of momentum, and combine with the actual normal momentum. That is, the convected normal momentum is approximated by

$$\boldsymbol{m}_e \cdot \boldsymbol{n}_i \approx (\boldsymbol{m}_e \cdot \boldsymbol{n}_e)(\boldsymbol{n}_e \cdot \boldsymbol{n}_i) + (\bar{\boldsymbol{m}}_e \cdot \boldsymbol{t}_e)(\boldsymbol{t}_e \cdot \boldsymbol{n}_i) \tag{A.8}$$

Although both options seem reasonable, testing found that the first option leads to a signficantly less stable scheme than the second. This is perhaps because the stencil is too far upwind, with little information being sourced from the immediate neighbourhood of the edge itself. The second option is therefore the only option of practical use.

### A.2.5 Selection of the Control Volume for Approximating the Momentum Divergence during the Edge Centre Piecewise Linear Normal Momentum Reconstruction

In Section 3.9.3, the selection of the control volume used in the approximation of the momentum divergence for the piecewise linear reconstruction of the edge centre normal momentum is discussed. There are four sensible choices for this control volume

1. The control volume is defined to be the union of the interior cells adjacent to edge $e$.

2. The control volume is defined to be the cell that has edge $e$ as a side and the base node as a vertex.

3. The control volume is defined to be the union of the interior cells adjacent to edge $i$.

4. The control volume is defined to be the cell that is bounded by both edge $i$ and edge $e$.

These four options are depicted in Figure A.1. Testing of each option found that the second option leads to poor stability and an unreasonably strong dependence on grid quality. This is likely due to the fact that the definition of the control volume used to approximate the momentum divergence for the upwind state can change from one iteration to the next, particularly where the local flow direction is closely aligned with the target edge $e$. The other three options lead to good schemes, with testing finding only modest differences in stability and accuracy between them. However, further development of the solver, particularly the addition of viscous terms, lead to the final option being selected.

### A.2.6 Selection of the Control Volume for Approximating the Momentum Divergence during the Edge Centre Piecewise Linear Tangential Momentum Reconstruction

In Section 3.9.3, the selection of the control volume for approximating the momentum divergence during the during the piecewise linear reconstruction of the edge centre tangential momentums is considered. There are two sensible choices

1. The control volume is defined to be the union of the interior cells adjacent to the target edge.

2. The control volume is defined to be the cell upwind of the target edge.

These options are shown in Figure A.2. Testing found that this choice had little effect on the resulting scheme. This is perhaps to be expected, since this reconstruction is only of consequence for the secondary flow variables. It does have some minor effects, especially on the boundary conditions. Ultimately the second option was selected, although there appears to be little reason to prefer one choice over the other.

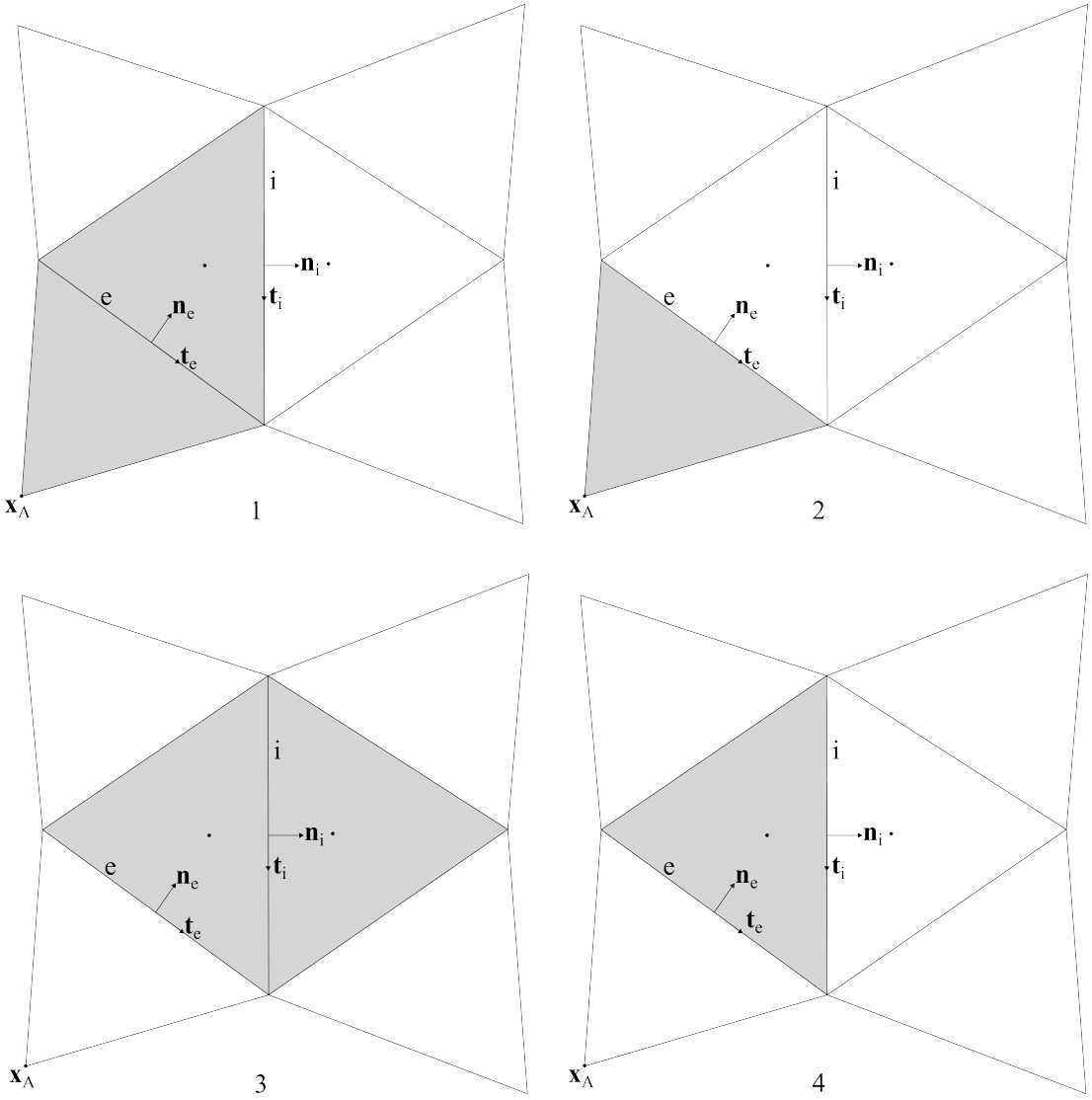



Figure A.1: The four possible control volume definitions (represented by the shaded areas) for approximating the momentum divergence when performing piecewise linear reconstruction of the left state convected normal momentum $(\boldsymbol{m}_e \cdot \boldsymbol{n}_i)_L$ for edge $e$ in the normal momentum equation for edge $i$.

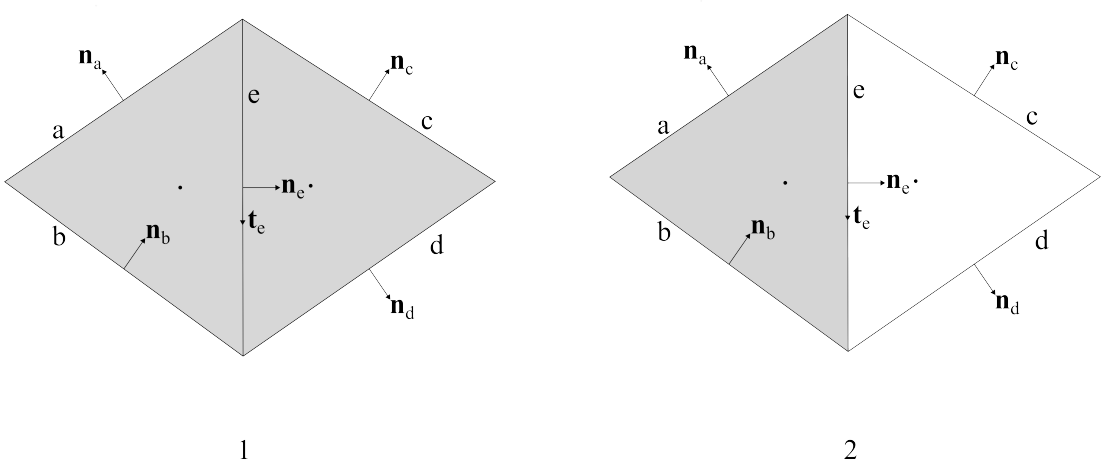



Figure A.2: The two possible control volume definitions (represented by the shaded areas) for use when performing explicit piecewise linear reconstruction of $(\boldsymbol{m} \cdot \boldsymbol{t})_e$.

## A.3 Chapter 4 - Further Development of the Inviscid Solver

### A.3.1 Gradient Limiting in Momentum Field Reconstructions

In Section 4.4.2, the application of gradient limiting to the piecewise linear momentum field reconstructon is considered. Since the full momentum vector is not stored at any location under the scheme discussed in this thesis, the momentum reconstruction cannot simply be written as a constant term plus a gradient correction term. Hence, the limiters cannot be used in the same way as for the scalar field reconstructions. There are two reasonable methods for handling this difficulty

1. Insert the limiter directly into the piecewise linear momentum field reconstruction. The reconstruction polynomial (3.66) becomes

$$\boldsymbol{m}(\boldsymbol{x}) \approx \bar{\boldsymbol{m}}\left(\Delta\boldsymbol{x}, \bar{d}, \Psi_i\right) \coloneqq \left\{ \begin{array}{c} a_x \\ a_y \end{array} \right\} + \Psi_i \left\{ \begin{array}{c} b\Delta x + c_x \Delta y \\ b_y \Delta x - b\Delta y \end{array} \right\} + \Psi_i \frac{\bar{d}}{2} \Delta\boldsymbol{x} \qquad \text{(A.9)}$$

   This method is similar to that used to limit the scalar recontruction, and does not require any additional reconstruction coefficients to be computed or stored, but it does have an important disadvantage. In the fully limited limit ($\Psi = 0$) a first-order reconstruction is obtained, but the stencil used to perform the second-order reconstruction is retained. As a result, instead of the two-point edge centre first-order momentum reconstruction stencils discussed in Section 3.9.1, the

reconstruction is performed using stencils with at least five points, and possibly many more (near boundaries). It is to be expected, therefore, that this method of limiting the momentum reconstruction will introduce far more numerical diffusion into the solution than strictly necessary, which will tend to smear discontinuities and boundary layers (the latter is not relevant to the development of the inviscid solver, but is important in the next chapter when the development of the viscous solver is considered).

2. Replace the second-order reconstruction by a blend of the first- and second-order reconstructions [VSW06]. This involves computing the piecewise constant momentum reconstruction $(\bar{\boldsymbol{m}}_e \cdot \boldsymbol{n}_i)^{(1)}$ from (3.57), computing the piecewise linear momentum reconstruction $(\bar{\boldsymbol{m}}_e \cdot \boldsymbol{n}_i)^{(2)}$ from (3.66), and defining the limited second-order convected momentum reconstruction to be

$$(\bar{\boldsymbol{m}}_e \cdot \boldsymbol{n}_i) = (1 - \Psi_e)\,(\bar{\boldsymbol{m}}_e \cdot \boldsymbol{n}_i)^{(1)} + \Psi_e\,(\bar{\boldsymbol{m}}_e \cdot \boldsymbol{n}_i)^{(2)} \tag{A.10}$$

where $\Psi_e$ is the pressure limiter at the base node being used for the reconstruction of $(\bar{\boldsymbol{m}}_e \cdot \boldsymbol{n}_i)^{(2)}$. This method of limiting the momentum reconstruction does require computing and storing additional reconstruction coefficients (for computing the first-order reconstruction), but it does have the advantage of minimising the size of the stencils used for computing the first-order reconstructions, so it can be expected to result in sharper discontinuities due to the reduced additional numerical dissipation. For this reason, it was the method used for all the examples presented in this thesis.

## A.4 Chapter 5 - Development of the Viscous Solver

### A.4.1 Non-Linearity of the Viscous Stress Correction

In Section 5.3.2 the discretisation of the viscous stress term in the pressure correction equation is considered. This requires an approximation of the correction to the viscous stress term due to the pressure correction, and it is found that this correction is quadratic in the pressure correction. There are two main methods for handling this

1. Use a Newton-like iterative method. This would involve solving the pressure correction equation iteratively, using the pressure corrections computed during each iteration to linearise this term during the next iteration. This results in an approximation of the form $\delta p^{n+1}\delta p^{n+1} \approx \delta p^{(m)}\delta p^{(m+1)}$, where $m$ is the inner

iteration counter and $\delta p^{(0)}$ is set to zero. This method could also be applied without change to the other linearised terms in the pressure correction equation.

2. If the time step is not too large then the pressure corrections can be assumed small, $\delta p \ll 1$. The non-linear terms are then simply neglected. In common with the linearisation of the inviscid terms this simplification should not affect the accuracy of the solution, since $\delta p \to 0$ as the steady state is reached. However, numerical testing is necessary to determine its effect on convergence to the steady state.

The first option could potentially lead to a more stable solver, allowing larger time steps to be used, but would significantly increase the computational cost of each iteration. On the other hand, it may be necessary if the second option is found to excessively harm convergence. Fortunately, numerical testing showed this not to be the case. The second choice is therefore adopted.

### A.4.2 Selection of the Stencil Base Nodes for the Reconstruction of the Normal Projection of the Edge Centre Viscous Stress Tensors

In Section 5.4.2, the reconstruction of the normal projection of the viscous stress tensor at the edge centres is shown to require the selection of two base nodes. Fortunately, there are only two sensible options to consider. These are now described, with reference to Figure A.3.

1. Use the two nodes opposite the target edge, i.e. set $n_\alpha = n_2$ and $n_\beta = n_3$. This has two serious drawbacks. The first is that it is not possible at boundary edges. The stencil must therefore be changed to either use the interior opposite node on its own, or to use the end nodes of the target boundary edge. The second drawback is that the resulting reconstruction is likely to be sensitive to grid geometry, since the centre of the target edge will not be at the mean position of its opposite nodes in general. Thus, grid stretching and growth could degrade the quality of the reconstruction.

2. Use the two end nodes of the target edge, i.e. set $n_\alpha = n_0$ and $n_\beta = n_1$. This stencil can be applied to all edges, both interior and boundary, and no extra dependency on the quality of the local grid geometry is introduced, since the centre of the target edge is always exactly at the mean position of its end nodes regardless of grid distortion.

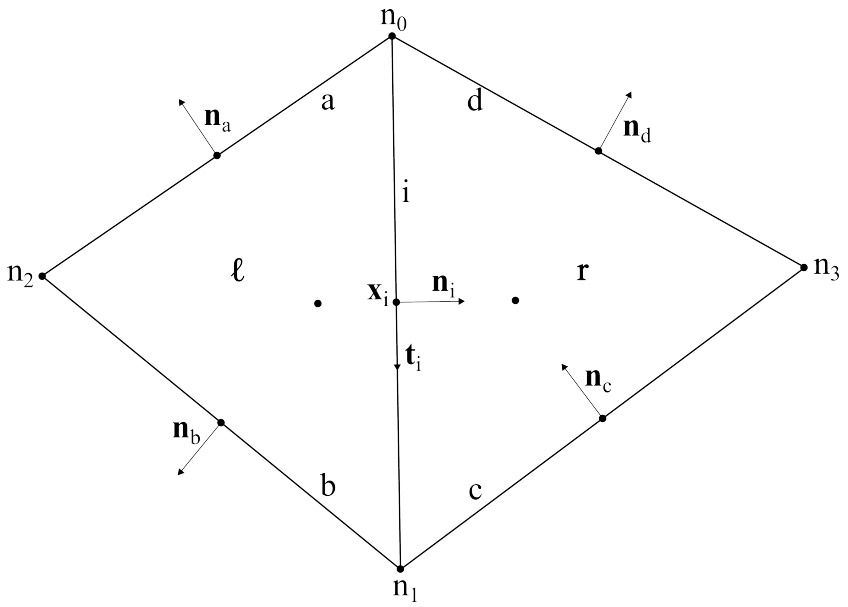



Figure A.3: Example of a grid layout in the neighbourhood of an arbitrary edge $i$. The viscous stress tensor is to be reconstructed at the centre of $i$ using the $\tau$ reconstructions at any two of the four nodes $\{n_0, n_1, n_2, n_3\}$.

The latter choice is clearly the better option.

# Appendix B

# Some Aspects of the Code Implementation

## B.1 Introduction

In this appendix, a brief overview and discussion of some aspects of the implementation of the Mach uniform solver is provided. This is not intended to be exhaustive, but instead serves to highlight and explain some of the key design choices.

Since the solver discussed in this thesis is relatively new and untested, the primary emphasis of the code design was flexibility. This ensured that as the project progressed, changes could easily be made to the structure of the solver. For example, different scalar and vector reconstruction schemes could be combined in a variety of ways at run-time, without the need to recompile the code. In pursuit of maximum flexibility, the object-oriented programming paradigm was adopted. In this model, data and operations pertaining to a single logical item are encapsulated in objects called *classes*. For example, in the present work a class representing a chimera component grid is defined. This class includes data such as the nodes, edges, and cells belonging to the component grid, as well as functions such as searching for a cell belonging to the component grid that contains an arbitrary point in the flow domain.

The solver was implemented in C++ 20 and parallelised using OpenMP. Code presented in this chapter is in C++-based pseudocode.

A well-known rule of thumb in the implementation of code for high performance computing is that structures of arrays should be preferred over arrays of structures (see [HW11], for example). This arrangement generally offers superior performance due to improved cache utilisation and more effective vectorisation. The improvement

in performance can be substantial, but comes at the cost of reduced flexibility and (sometimes) increased code complexity. For this reason, this rule was largely ignored in the implementation of the code for the current work.

## B.2 Notation

Several notational conventions are first established for use in the diagrams included in this chapter. These are explained with reference to Figure B.1.

- Classes and structures are enclosed in dotted boxes. The name of the class or structure is denoted by bold type. For example, **CEdge** is a class.
- Data members are denoted by a name in regular type. If relevant, the type of the data member is denoted by a solid arrow pointing from the data member to its type.
- Member functions are denoted by a name in italics, followed by brackets. For example, *MoveComponentGrid()* is a member function.
- Pointers are denoted by an asterisk after the name of the data member. A dotted arrow points from a pointer to its type. For example, "Component grid *" is a pointer to an object of type **Component Grid**.
- Arrays are denoted by square brackets after the name of the data member. If the size of the array is known at compile-time, the size is inserted between the brackets. For example, "End points * [2]" is an array of two pointers to objects of type **Node**. Two-dimensional arrays are notated as arrays of arrays, following the C++ convention. For example, Child leaves*[2][2] is a $2 \times 2$ array of pointers to quadtree leaves.
- Run-time polymorphism is denoted by a hash. For example, **Component Grid Movement #** is an abstract base class with several (concrete) derived classes. At run-time, a data member of type **Component Grid Movement #** is instantiated as the appropriate derived class according to a user-supplied grid movement specification. Due to the deleterious effect run-time polymorphism can have on performance as the problem size is scaled up, its use was generally limited to the serial part of the code.

## B.3 Solver

The code is built around the solver class, which is implemented as a singleton (see [Gam+95]), meaning that only a single solver object can exist and that it can be accessed globally. Whilst the global access would be unnecessary for a code implementing the finished scheme, it greatly increased the flexibility of the code during the design process.

The solver object contains all the data and objects required for the implementation of the scheme. This includes

- Arrays storing the current and previous values of the primary and secondary flow variables.
- Arrays storing the current edge-centre and cell-centre physical time steps and pseudo time steps.
- Solver parameters such as the selected convergence thresholds and reference values
- The grid object, discussed in Section B.4.
- The fluid model object, discussed in Section B.5.
- The sparse solver objects, discussed in Section B.6.
- Objects for performing scalar, vector, viscous stress, and convective flux reconstructions. The first of these is discussed in Section B.8.
- Objects for performing numerical volume integration.
- Objects implementing the boundary conditions.
- An object used to compute the values of the aerodynamic coefficients for solid objects within the flow domain, for use in applying the farfield vortex correction discussed in Section 4.5.2, for monitoring progress towards convergence, and for output.
- Solution export objects and functions.

In addition the solver object contains the member functions implementing the basic functionality of the solver. This includes

- Functions for starting, pausing, resuming, and terminating the solution process, and updating solver parameters whilst the solver is running. This part of the solver is implemented in a separate UI thread, so that these functions can be called whilst the solver is still running. The UI thread sends messages to the solver thread, which checks for messages before starting a new iteration.
- Setup functions such as those for importing the grid and the solver parameters, performing non-dimensionalisation, imposing the initial conditions and constructing the boundary condition objects.
- Functions for assembling each of the governing equations.
- A main solver loop, which iteratively assembles then solves the governing equations, updates the secondary flow variables and boundary conditions, and checks progress towards convergence.
- Functions for exporting solution data.
- Unit test functions.

## B.4 Grid

The basic structure of the grid objects and the relationship between them is shown in Figure B.1. All of the depicted classes can have multiple instances, with the exception of **Grid**.

The structure of the grid objects allows the algorithms implementing Chimera functionality discussed in Section 6.7.2, Section 6.7.3, and Section 6.7.4 to be implemented directly in the grid class itself. In addition, the functions responsible for identifying the most suitable reconstruction base nodes and cells discussed in Section 6.7.5 are also naturally implemented in the grid class.

The dynamic quadtree class is essentially a wrapper around a static quadtree class. At solver start-up, the quadtree covering each component grid is set-up in a local coordinate system coinciding with the initial global coordinate system. All nodes in the component grid are then added to the quadtree. Each time a component grid movement object applies a rigid transformation to the owning component grid, the applied transformation is composed with the previous overall transformation and stored. Searching the quadtree for a point given the current global coordinate is then a simple matter of applying the inverse of the stored transformation to the location of

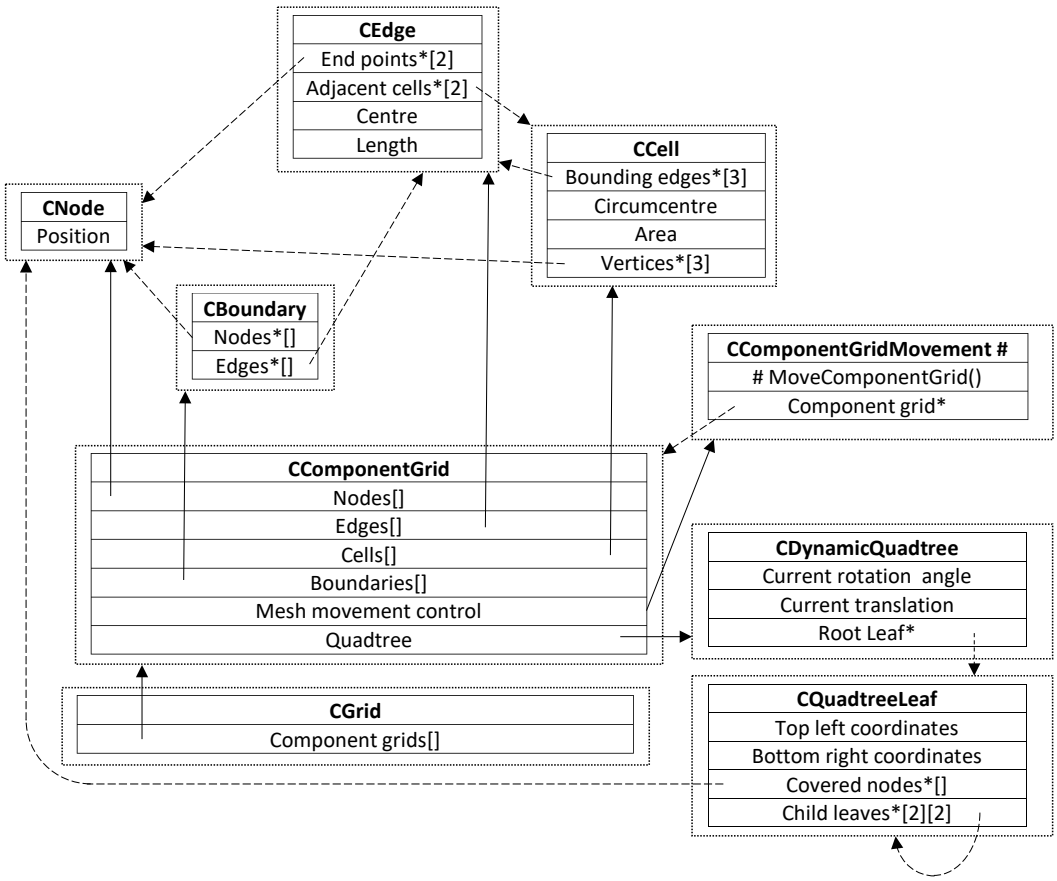



Figure B.1: The basic relationships between the grid objects.

the point, then searching the static quadtree. The static quadtree itself is implemented in the standard recursive manner [Sam82]. Each leaf has pointers to its four child leaves (if they exist) but not to its parent leaf. This allows efficient nearest-neighbour searching, as required for finding appropriate interpolation source nodes and cells.

## B.5 Fluid Model

This is a particularly simple area of the code due to the assumptions imposed on the fluid model during the development of the code. Specifically, the fluid model consists only of a set of model constants together with a pair of functions for calculating the dynamic viscosity and thermal conductivity at a point given the enthalpy. Since the dynamic viscosity and thermal conductivity only need to be updated once per iteration, this can be done efficiently even using run-time polymorpism. A skeleton of the implementation with one example concrete derived class is shown in Listing 1.

```
class CFluid {
        virtual void UpdateMuAndK() = 0;

        double dGamma;
};

class CAir : public CFluid {
        CAir() { dGamma = 1.401}

        void UpdateMuAndK() override;
};
```

Listing 1: The basic implementation of the fluid model class.

The virtual function `void UpdateMuAndK()` is called once per iteration, after solving the equation of state to update the cell centre enthalpies, to update the cell centre and edge centre dynamic viscosities and thermal conductivities. If necessary it can access the enthalpies (and any other flow variables) to calculate these, according to the specific fluid model the particular derived class is implementing.

## B.6 Sparse Solver

During each iteration three sparse systems must be assembled and solved, corresponding to the continuity, normal momentum, and pressure correction equations. These sparse systems differ significantly from each other. Even more importantly, when using certain configurations of the solver (with fully upwind convective flux reconstruction for example) the sparsity structure of the systems can change from one iteration to the next. In addition there is a great multitude of methods available for the solution and preconditioning of these systems, and no general rules exist for the selection of the best methods for a particular problem [Y03]. These concerns were addressed by encapsulating the sparse solver functionality in a polymorphic class, and providing a separate instance for each of the three governing equations. In this way, simple and rapid switching between the combinations of solver and preconditioner was possible without the need for time-consuming recompilation. The basic interface of the sparse solver class is shown in Listing 2.

```
class CSparseSolver {
        CSparseSolver(int iNumberOfRows);

        //Functions for performing setup
        void AddElementPostion(int iRow, int iColumn);
        virtual void FinishSetup() = 0;

        //Functions for assembling and solving the system
        void AddElement(int iRow, int iColumn, double dValue);
        void AddKnown(int iRow, double dValue);
        virtual void Solve() = 0;
        void Reset();
};
```

Listing 2: The basic implementation of the sparse solver class.

The function `AddElementPostion()` is called during sparse solver setup to specify the position of an element that could possibly be non-zero. Once all such elements have been identified, `FinishSetup()` is called. During each time step, the sparse systems are assembled and solved in turn. This is done by calling `AddElement(double dValue)` to add `dValue` to the currently set value of the specified element and `AddKnown(double dValue)`

to add `dValue` to the currently set value of the specified element of the vector of knowns. `Solve()` is then called. It is a pure virtual function since the details of the solution process depend on the particular combination of preconditioner and solver selected. It packs the non-zero elements into the arrays representing the sparse system matrix (as described below), then solves the complete system according to the details of the actual concrete class instantiated. Finally, `Reset()` is called to clear the value of all elements of the elements of the system matrix and the vector of knowns.

Since most of the elements in the system matrices are zero, there is no need to store the value of every element. Instead, *compressed sparse row* (CSR) format is employed. In this scheme, a sparse matrix is represented by a set of three arrays. The first two, `double Values[]` and `int Columns[]`, have size equal to the number of non-zero elements and contain the value of the element and the index of the column in which it lies respectively. These must be ordered row-by-row. The third array, `int RowIndices[]`, has size one larger than the number of rows in the system and contains the indices of the elements of `Values[]` that are the first in their row (the final element being set equal to one plus the index of the final element of `Values[]`). Since the sparsity structure of the system matrices cannot be assumed to remain the same from one iteration to the next, the arrays `Values[]` and `Columns[]` do not necessarily have the same length, and the elements of `RowIndices[]` are not fixed.

The simplest method of preparing the vectors representing the sparse matrix is to allocate a space for every element that could possibly be non-zero. The elements of `Columns[]` and `RowIndices[]` are then fixed throughout the iterations. During each iteration, the elements of `Values[]` are initially set to zero, and are filled-in as necessary. Since each row can have tens of elements, a searching structure such as a hash map (with column index as its key) is used to accelerate the process of finding the element of `Values[]` corresponding to a particular column index. The arrays are then passed to the function for solving sparse systems, without removing the elements that happen to zero on that iteration. This method works, but the solution process will be inefficient as the system matrix is treated as being less sparse than is actually the case.

Ultimately it was found that solving the sparse systems took the great majority of the execution time, and therefore it was justified to implement a more complex (and hence slower) assembly method that does remove zero elements from the system matrix. To do this, the potential number of non-zero elements was first counted and used to create sufficiently large arrays. The hash map structures were removed. Instead, a binary tree was created for each row. At the beginning of each iteration, the tree is

emptied. On a call to `AddElement(int iRow, int iColumn, double dValue)`, the magnitude of `dValue` was first checked against some threshold. If it is sufficiently large the tree is searched for an existing value in the appropriate column. If one is found, `dValue` is added to the attached value. If not, a new entry is added to the tree. When `Solve()` is called, the trees are traversed and `Values[]` and `Columns[]` are filled-in. Efficient algorithms exist for all three of the required operations on the binary trees (searching, inserting, and in-order traversal), and if the size of the trees is recorded during the building process the fill-in can be done in parallel using a parallel scan (see [Rei07, p. 49] for more details).

## B.7 Implicit Expressions

At several places in the code, the processing of implicit expressions is required. These expressions consist of weighted sums of cell-centre or edge-centre values, where the actual values are not known. The weighting coefficients could be scalars, vectors, or tensors. In some circumstances, an explicit term could also be included. A container was implemented to represent these expressions. It is essentially implemented as as templated wrapper around a `std::vector`, with the basic implementation shown in Listing 3.

```
template<typename ExpressionType>
class CImplicitExpression {
        CImplicitExpression() : _ExplicitTerm{}{}

        void AddImplicitTerm(int Index, ExpressionType Coefficient){
                _ImplicitExpression.emplace_back(Index, Coefficient);
        }

        ExpressionType Evaluate(double *Values)const
        {
                ExpressionType Result = _ExplicitTerm;

                for(auto Term : _ImplicitExpression)
                        Result += Term.second * Values[Term.first];

                return Result;
        }

        std::vector<std::pair<int, ExpressionType>> _ImplicitExpression;
        ExpressionType _ExplicitTerm;
};
```

Listing 3: The basic implementation of the implicit expression container.

The implementation is quite general, requiring only that the type of the weighting coefficients be default-constructible and have addition-assignment and scalar multiplication operators defined.

## B.8 Scalar Reconstruction Objects

The Mach-uniform scheme requires node-based and cell(-circumcentre)-based scalar gradient reconstructions, and edge(-centre)-based scalar reconstructions. The edge-based scalar reconstructions were required to use either a node-based or cell-based scalar gradient reconstruction. At the beginning of the code development the choice of which of these two options would actually be used in the finished scheme, and the possibility of the eventual use of third-order reconstructions requiring node-based

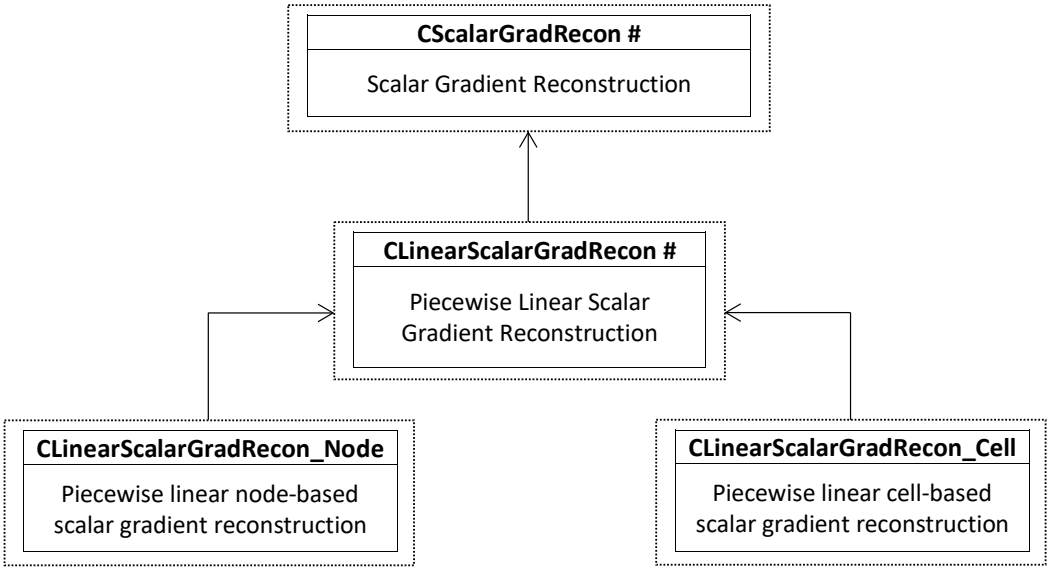



Figure B.2: The inheritance relationship between the objects implementing scalar gradient reconstruction. The arrows point from child to parent.

and cell-based scalar Hessian reconstruction were open. In addition, both explicit and implicit versions of the same reconstructions were required. The challenge was therefore to keep the objects representing these reconstructions as general as possible, whilst retaining reasonable efficiency.

The node-based and cell-based scalar gradient reconstruction objects were organised according to the class dependence diagram shown in Figure B.2. The basic implementation of these objects is shown in Listing 4, where `C2DVector` is the class representing a two-dimensional vector.

```
class CScalarGradRecon{
        virtual void Setup() = 0;
        virtual void Reconstruct(CImplicitExpression<C2DVector>& Recon) = 0;
        virtual void GetDiff_Linear(C2DVector vctStart, C2DVector vctEnd,
                CImplicitExpression<C2DVector>& Recon) = 0;

        std::vector<int> _StencilCellIDs;
}

class CLinearScalarGradRecon : public CScalarGradRecon{
        virtual void Setup() = 0;
        void Reconstruct(CImplicitExpression<C2DVector>& Recon);
        void GetDiff_Linear(C2DVector vctStart, C2DVector vctEnd,
                CImplicitExpression<C2DVector>& Recon)

        std::vector<C2DVector> _ReconCoeffs_Linear;
};

class CLinearScalarGradRecon_Node : public CLinearScalarGradRecon{
        CLinearScalarGradRecon_Node(CNode* TargetNode);
        void Setup();
        CNode* _TargetNode;
};

class CLinearScalarGradRecon_Cell : public CLinearScalarGradRecon{
        CLinearScalarGradRecon_Cell(CCell* TargetCell);
        void Setup();
        CNode* _TargetCell;
};
```

Listing 4: The basic implementation of the scalar gradient reconstruction objects.

With this setup, edge-based scalar reconstruction has a particularly simple form, as shown in Listing 5.

```
class CEdgeScalarRecon{
        void Setup(CScalarGradRecon* ScalarGradRecon_Left,
                CScalarGradRecon* ScalarGradRecon_Right);

        CImplicitExpression<double> _ScalarReconstructions_Linear[2];
        CEdge* _TargetEdge;
}
```

Listing 5: The basic implementation of the edge-centre scalar reconstruction objects.

The scalar gradient reconstruction objects to be used when the edge centre normal momentum is non-negative and when it is negative are passed to `Setup()` as abstract base class pointers. `Setup()` determines the linear part of the reconstructions, `_ScalarReconstructions_Linear[]`, by making virtual calls to `ScalarGradRecon_Left->GetDiff_Linear(...)` and `ScalarGradRecon_Right->GetDiff_Linear(...)`. Note that the linear part of the expression has to be kept separate to allow the limiter functions to be applied. The constant part of the expressions are trivially added by using the adjacent cell pointers stored in `_TargetEdge`.

The above structure was designed with the knowledge that third-order reconstruction could be added to the code at a later stage. Doing so could easily be done for testing purposes, though in a way that would not yield particularly efficient code, by simply adding another pair of expressions to `CEdgeScalarRecon` to represent the quadratic part of the expressions. This would again need to be kept as a separate expression, rather than combining with the linear parts, to allow application of the limiter functions.

The momentum reconstruction objects are implemented in a similar way, although the code is simpler due to the lack of cell-based vector gradient reconstruction objects.