A Parallel Adaptive-Mesh Method for Predicting Flows Through Vertical-Axis Wind Turbines

by

Samuel Heng Hsin Wong

A thesis submitted in conformity with the requirements
for the degree of Masters of Applied Science
Graduate Department of Aerospace Engineering
University of Toronto

Abstract

A Parallel Adaptive-Mesh Method for Predicting Flows Through Vertical-Axis Wind Turbines

Samuel Heng Hsin Wong

Masters of Applied Science

Graduate Department of Aerospace Engineering

University of Toronto

2011

Significant progress has been made towards developing an effective solution method for predicting low-speed flows through vertical-axis wind turbines. A Godunov-type finite-volume scheme has been developed for the solution of the Euler equations in two-dimensions on multi-block meshes. The proposed algorithm features a parallel block-based adaptive mesh refinement scheme and a mesh adjustment procedure to enable straightforward meshing of irregular solid boundaries. A low-Mach-Number preconditioner is used in conjunction with a dual time-stepping scheme to reduce the computational costs of simulating low-speed unsteady flows. A second-order backwards differencing time-marching scheme is used for the outer physical-time discretization, and an explicit optimally-smoothing multi-stage time-stepping scheme with multigrid acceleration is used for the inner pseudo-time loop. Results are presented for various low-speed flows that demonstrate the suitability of the algorithms for wind turbine flows. Additional theory and discussion are also presented for extension of the schemes to the full Navier-Stokes equations.

Acknowledgements

Looking back, the pursuit of this masters degree has been an extremely fruitful one. (Although I have not always seen it that way!) This research experience has been like a tough mountain climb. The trail, complete with twists and turns, has not been easy, but the view at the top is more than worth it. I am very grateful for the people who have helped me along the way:

Professor Clinton Groth, for his guidance, support, and commitment in advising me throughout the course of this project;

Edward Tsang, whose passion and insights have given me a broadened perspective on engineering in the world outside of academia;

My colleagues in the CFD lab, especially Jai Sachdev, who helped me tremendously with the embedded boundaries algorithm and left behind a solid foundation of code for me to build on;

Family and friends, for unwavering support and faith in me, even when I doubted myself. Their love and prayers brought me through the ups and downs of life as a graduate student.

Financial support from UTIAS, MITACS, and Zephyr Alternative Power Inc. is gratefully acknowledged.

Soli Deo Gloria

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

The wind has been harnessed to do all manner of useful work for centuries, but has become an important, viable, and attractive means of generating electricity only recently. Growing concerns about fossil fuels, pollution, and climate change have drawn attention to wind power as a clean and more environmentally-friendly alternative to coal, oil, and gas. Advances in materials and a better understanding of aerodynamics have enabled the construction of large installations capable of generating hundreds of megawatts of renewable power.

The vast majority of installed wind power are large-scale "wind farm" installations each averaging about 60–90 wind turbines. These wind farms are usually sited in remote areas such as farmlands, deserts, and offshore (lakes and oceans). As of 2009, there is an estimated 152,000 MW of installed wind power worldwide with 30,300 MW installed in 2009 alone [5]. We have certainly come a long way since the windmills of cloth and timber.

1.1.1 Wind Turbine Types

The modern wind turbine can be classified into one of two general categories, differentiated by the orientation of their rotors: vertical-axis wind turbines (VAWT) and horizontal-axis wind turbines (HAWT). Of the two types, the latter are by far the most common. The typical horizontal-axis wind turbine generally employ three blades ranging from 50m to 100m in diameter, generating up to 4 MW of power. The nacelle houses the mechanical system for yaw and blade pitch control, the gearbox, and electrical generator. Vertical-axis wind turbines

are further subdivided based on whether the power extracted from the wind is harnessed via aerodynamic drag or by aerodynamic lift. The Savonius turbine is a drag-type vertical-axis wind turbine, while the Darrieus and H-rotor turbines are lift-type vertical-axis wind turbines. Whether lift- or drag-based, vertical-axis wind turbines claim one advantage above their horizontal cousins: wind is accepted from any direction, whereas HAWTs need to be turned to face the wind. Figure 1.1 shows schematics of various wind turbine designs.

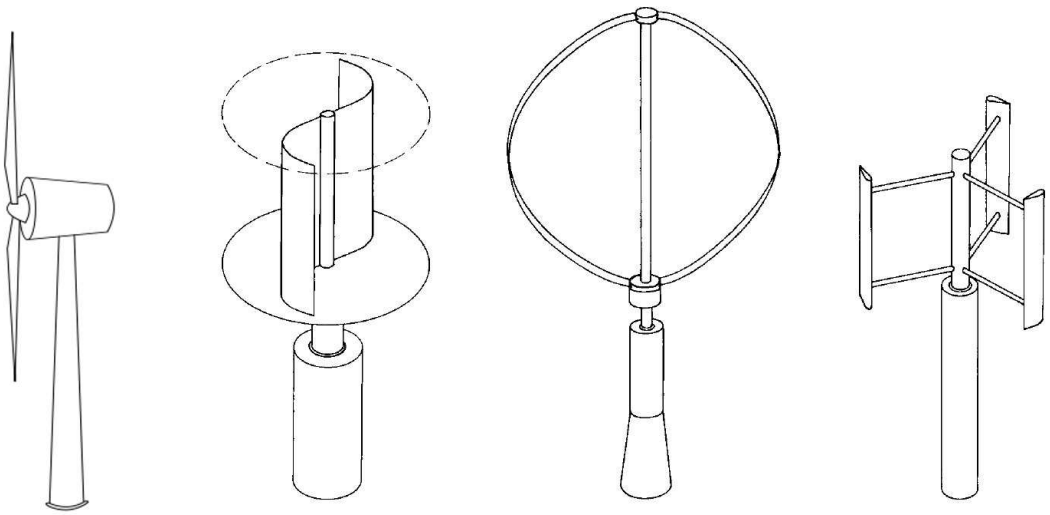


Figure 1.1: Schematics of various wind turbine designs [2]. From left to right: horizontal-axis, Savonius, Darrieus, H-rotor.

1.1.2 Efficiency of Wind Turbines

From the most basic perspective, a wind turbine is simply an energy conversion device. A wind turbine extracts the kinetic energy of air and converts it into mechanical energy, which is then converted into electrical energy via a generator. Thus, the total efficiency of a wind turbine is dependent on the product of two energy conversions: kinetic to mechanical, and mechanical to electrical. In this thesis, the focus is solely on the former as it directly relates to the aerodynamics of the wind turbine. Unless otherwise stated, all further discussion on efficiency will refer strictly to aerodynamic efficiency.

The efficiency of any wind turbine is defined as the amount of energy extracted by the turbine as a proportion of the power available in the wind. That is,

$$c_p = \frac{P_e}{P_o}, \tag{1.1}$$

where c_p is the coefficient of power (efficiency), P_e is the power extracted from the wind, and P_o is the power available in the wind. The power extracted is given by $P_e = \tau \cdot \omega$, where $\tau = \mathbf{r} \times \mathbf{F}$ is the torque developed on the shaft of the wind turbine and ω is the rotational speed. The power available in a mass of moving air can be calculated by considering the kinetic energy of an air mass, m, moving at a velocity, v_∞, so that

$$E = \frac{1}{2} m v_\infty^2. \tag{1.2}$$

For a turbine with a cross-sectional area, A, as seen from the incoming air, the mass flow rate is given by

$$\dot{m} = \rho v_\infty A. \tag{1.3}$$

Differentiating Equation (1.2) with respect to time and substituting Equation (1.3) into the resulting equation yields an energy flow rate, or power:

$$P_o = \frac{1}{2} \rho v_\infty^3 A \tag{1.4}$$

It can be shown that the theoretical maximum efficiency of any wind turbine is $c_p = 0.593$. This is commonly referred to as the Betz limit [6].

The wind turbine performance is typically plotted against tip-speed ratio (TSR), a non-dimensional quantity defined as

$$\lambda = \frac{\text{tipspeed}}{\text{windspeed}} = \frac{r\omega}{V_\infty} \tag{1.5}$$

where r is the radius of the turbine, and V_∞ is the freestream velocity. An overview of efficiencies for various wind turbines types is shown in Figure 1.2. In general, lift-based turbines operate most efficiently at large values of TSR; they spin much faster than the wind. However, these turbines have a low starting torque and often require mechanical assistance for start-up. For this reason, HAWTs are placed in locations with strong and consistent windspeeds. In contrast, the drag-based Savonius turbine operates most efficiently in the low-TSR regime, and hence rotate much slower than their lift-based counterparts. The Savonius turbine benefits from high starting torque and is self-starting, making it ideal for locations with lower windspeeds.

1.2 Motivation

There is a natural and obvious motivation to improve current wind turbine designs. Efficiency and durability are often the technical challenges that face the engineer. However, there is also a need to develop new means of harnessing wind energy that not only overcomes the technical challenges but is also acceptable from a socio-political perspective.

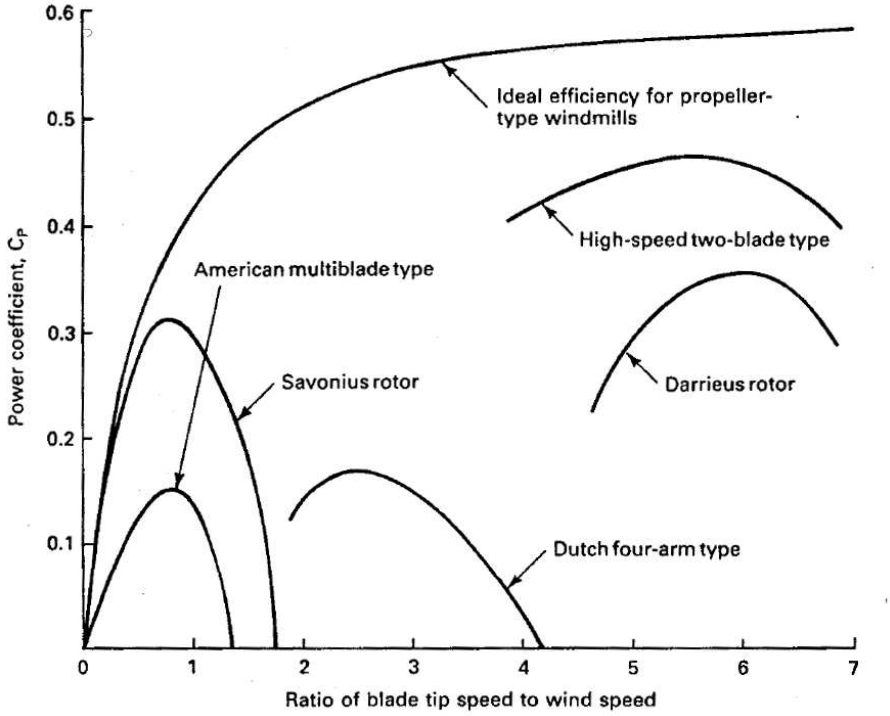


Figure 1.2: Power coefficients of various wind turbine designs [3].

The concerns most often raised against wind power are its effect on animal and plant life (especially birds and insects), noise emission, electromagnetic interference, visual impact, and changes in land use and land value [2]. As a result, large-scale wind farms are often mired in controversy involving a large of number of interested parties. Not surprisingly wind farm installations take several years to obtain public approval.

These issues, however, are almost exclusively applicable to large-scale installations. Independent, small-scale wind turbines have the benefit of being low-impact, hence more politically and socially palatable, and do not require a lengthy approval process. Moreover, small-scale turbines can be deployed where large multi-storey turbines cannot, such as in the urban landscape. Like solar panels on residential roofing, small-scale wind power has the potential to reclaim urban areas for electricity generation, supplying electricity where it is consumed. Recent legislation in Ontario now enables homeowners to generate and sell electricity back to the electric company. Coupled with smart grid technologies, small-scale wind generation in the urban environment is quickly gaining traction.

1.3 Numerical Predictions of Wind Turbine Flows

Due to their popularity, the majority of research and development efforts have been focused on the horizontal-axis wind turbine. However, vertical-axis wind turbines are beginning to draw more attention. Increased interest, coupled with advances in computational power, has seen extensive use of computational fluid dynamics (CFD) simulations to improve the aerodynamic design of wind turbines [7, 8]. Some recent achievements are now summarized for both horizontal-axis and vertical-axis wind turbines.

The Reynolds-averaged Navier-Stokes equations is a common approach for solving turbulent flows for both horizontal-axis and vertical-axis wind turbines. Mandas *et al.* [9], and Guerri and coworkers [10] have computed wind turbine flows using Menter's SST k-ω turbulence model. Sankar and coworkers [11], Kasmi and Masson [12], and Howell and co-workers [13] all used different variants of the k-ϵ turbulence model. A hybrid approach was used by Xu and Sankar [14] where the Navier-Stokes equations were solved in regions near the rotating turbine blades and the inviscid potential flow equations were solved in regions far away from the blades.

When simulating a rotating wind turbine, the computational mesh must allow for the motion of solid surfaces through the domain. Sezer-Uzol and Long [15] performed inviscid simulations using unstructured grids that rotated with the turbine blades.

1.4 Objective and Overview

The objective of this thesis is to develop a computational tool for predicting two-dimensional inviscid flows through cross-sections of rotating vertical-axis wind turbines. This objective will aid in the overall goal of designing and optimizing vertical-axis wind turbines for use in an urban environment.

A Godunov-type finite-volume scheme has been developed for the solution of the two-dimensional form of the Euler equations on multi-block quadrilateral meshes. The proposed solution method features a parallel block-based adaptive mesh refinement scheme amenable to solutions on large multi-processor parallel clusters, combined with a mesh adjustment procedure to enable straightforward meshing of irregular solid boundaries [16]. This procedure readily allows for moving and stationary embedded boundaries that are not aligned with the mesh. The AUSM$^+$-up flux function is used to evaluate the inviscid fluxes [17]. A low-Mach-number preconditioner has been developed to reduce the numerical stiffness of the governing equations associated with simulating low-speed flows and this is used in conjunction with a dual time-stepping scheme to

reduce computational costs and enable large time-steps when simulating unsteady flows [18]. A second-order backwards differencing time-marching scheme is used for the outer physical-time discretization and, within each physical time step, an explicit optimally-smoothing multi-stage time-stepping scheme with multigrid acceleration is used to reduce the inner pseudo-time residual [19]. Additional theory and discussion are also presented for extension of the proposed schemes to the Navier-Stokes equations.

This dissertation is organized as follows. Chapter 2 describes the set of partial differential equations that govern the behaviour of a compressible thermally-perfect gas. Chapter 3 describes the parallel, block-based finite volume scheme used to solve the governing equations. Several test problems and a full simulation of a vertical-axis wind turbine are presented in Chapter 4. A summary and concluding remarks are presented in Chapter 5. Supporting mathematical derivations are listed in the Appendices.

Chapter 2

Governing Equations

2.1 Conservation Equations

The Navier-Stokes equations are used to describe a single-phase, Newtonian fluid in the continuum regime. These equations can be derived by applying the principles of mass, momentum, and energy conservation to a control volume. Written in differential form using vector notation, the compressible Navier-Stokes equations are

$$\frac{\partial}{\partial t}(\rho) + \vec{\nabla} \cdot (\rho \vec{u}) = 0 \tag{2.1}$$

$$\frac{\partial}{\partial t}(\rho \vec{u}) + \vec{\nabla}(\rho \vec{u}\vec{u} + \vec{\vec{I}}p) = \vec{\nabla} \cdot \vec{\vec{\tau}} \tag{2.2}$$

$$\frac{\partial}{\partial t}(\rho E) + \vec{\nabla} \cdot \left[\rho \vec{u}\left(E + \frac{p}{\rho}\right)\right] = -\vec{\nabla} \cdot \vec{q} + \vec{\nabla} \cdot (\vec{\vec{\tau}} \cdot \vec{u}) \tag{2.3}$$

where ρ is the fluid density, $\vec{u}$ is the fluid velocity vector, p is the fluid pressure, $\vec{\vec{I}}$ is the identity tensor, E is the total energy per unit mass, $\vec{q}$ is the heat flux vector, and $\vec{\vec{\tau_{ij}}}$ is the fluid stress tensor. Equation (2.1) represents the conservation of mass, Equation (2.2) represents the conservation of momentum, and Equation (2.3) represents the conservation of energy. The total energy per unit mass can be considered as a sum of the total internal energy and the kinetic energy so that

$$E = e + \frac{1}{2}\vec{u} \cdot \vec{u} \tag{2.4}$$

with

$$e = \frac{p}{(\gamma - 1)\rho}, \tag{2.5}$$

where e is the total internal energy and γ is the ratio of specific heats taken to be equal to 1.4. In this work, the ideal gas equation of state is used, and is given by

$$p = \rho RT \tag{2.6}$$

where R is the universal gas constant and T is the gas temperature.

2.2 Inviscid Euler Equations

When viscosity and heat flux are neglected, Equations (2.2) and (2.3) can be simplified to obtain the inviscid Euler equations

$$\frac{\partial}{\partial t}(\rho) + \vec{\nabla} \cdot (\rho \vec{u}) = 0 \tag{2.7}$$

$$\frac{\partial}{\partial t}(\rho \vec{u}) + \vec{\nabla}(\rho \vec{u}\vec{u} + \vec{\vec{I}}p) = 0 \tag{2.8}$$

$$\frac{\partial}{\partial t}(\rho E) + \vec{\nabla} \cdot \left[\rho \vec{u}\left(E + \frac{p}{\rho}\right)\right] = 0. \tag{2.9}$$

While the above equations do not completely represent the fluid behaviour as applied to wind turbine flows, it is a good first approximation and they provide a useful starting point for numerical studies. Solutions to the Euler equations provide useful information on pressure distributions, key aerodynamic forces and moments, and can predict general fluid velocities outside of boundary layers. Obviously, flows with large separation regions cannot be accurately treated with an inviscid representation.

2.2.1 Eigensystem Analysis of the Euler Equations

The two-dimensional form of the Euler equations can be summarized in vector form as

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F_x}}{\partial x} + \frac{\partial \mathbf{F_y}}{\partial y} = 0 \tag{2.10}$$

where

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{bmatrix}, \quad \mathbf{F_x} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho u\left(E + \frac{p}{\rho}\right) \end{bmatrix}, \quad \mathbf{F_y} = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho v\left(E + \frac{p}{\rho}\right) \end{bmatrix}. \tag{2.11}$$

and u and v are the components of the velocity in the x- and y-direction, respectively. The above equations can also be re-written via the chain rule as

$$\frac{\partial \mathbf{U}}{\partial t} + \mathbf{A}\frac{\partial \mathbf{U}}{\partial x} + \mathbf{B}\frac{\partial \mathbf{U}}{\partial y} = 0 \tag{2.12}$$

where

$$\mathbf{A} = \frac{\partial \mathbf{F_x}}{\partial \mathbf{U}}, \quad \mathbf{B} = \frac{\partial \mathbf{F_y}}{\partial \mathbf{U}} \tag{2.13}$$

An eigenstructure analysis of the above system of equations can be performed by considering the eigenvalues. In the x-direction, the eigenvalues is obtained by solving for the roots of the characteristic equation $det(\mathbf{A} - \lambda \mathbf{I}) = 0$, which yields

$$\lambda_1 = u - a, \ \lambda_{2,3} = u, \ \lambda_4 = u + a \tag{2.14}$$

where a is the speed of sound given by $\sqrt{\frac{\gamma p}{\rho}}$. Similarly, in the y-direction, the solution of the characteristic equation $det(\mathbf{B} - \lambda \mathbf{I}) = 0$ are

$$\lambda_1 = v - a, \ \lambda_{2,3} = v, \ \lambda_4 = v + a \tag{2.15}$$

The eigenvalues of the system represent the speed of propagation of fundamental solution modes associated with the equations. The first and fourth eigenvalues represent the speed of the acoustic waves while the second and third eigenvalues represent the speed of the convective waves, which includes a contact or entropy wave and a shear wave.

2.3 Favre-Averaged Navier-Stokes Equations

It is generally accepted that the full Navier-Stokes equations, (2.1), (2.2), and (2.3), are sufficient to describe the full scale of fluid motion from the smallest eddies (on the order of the Kolmogorov scale) to the largest eddies. However, it is not practical to solve these equations directly due to the immense computational resources required. Instead, a statistical or time-averaging approach is taken and a set of modified equations are solved for the mean flow quantities. Solutions of the time-averaged Navier-Stokes equations are widely used in today's engineering studies of aerodynamic flows [4].

An instantaneous flow quantity, $\phi(\vec{x}, t)$, can be decomposed into a mean component, $\overline{\phi}(\vec{x}, t)$, and a fluctuating part, $\phi'(\vec{x}, t)$, so that

$$\phi(\vec{x}, t) = \overline{\phi}(\vec{x}, t) + \phi'(\vec{x}, t) \tag{2.16}$$

where the time-averaged quantity is defined as

$$\overline{\phi}(\vec{x}, t) = \frac{1}{T} \int_{t-\frac{1}{2}}^{t+\frac{1}{2}} \phi(\vec{x}, t')dt' \tag{2.17}$$

T is taken to be a sufficiently longer period of time compared to the time scale of the turbulence fluctuations but smaller than the time scale of the mean flow variations. See Figure 2.1.

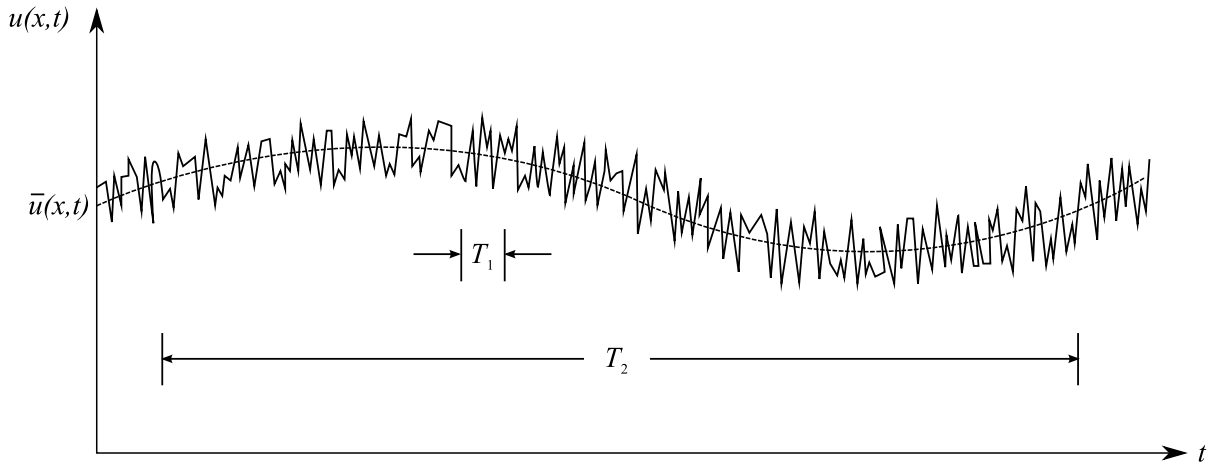


Figure 2.1: Time average of turbulence.

For compressible flows, a Favre-averaging procedure is introduced to avoid density correlations appearing in the averaged equations. Density-weighted quantities are decomposed into a Favre-averaged quantity, $\overline{\rho}\tilde{\phi}$, and a fluctuating quantity, $\rho\phi''$, so that

$$\rho\phi = \overline{\rho}\tilde{\phi} + \rho\phi'' \tag{2.18}$$

where the Favre-averaged quantity is defined as

$$\overline{\rho}\tilde{\phi}(\vec{x},t) = \frac{1}{T}\int_{t-\frac{1}{2}}^{t+\frac{1}{2}} \rho(\vec{x},t')\phi(\vec{x},t')dt' \tag{2.19}$$

Applying the Favre-averaging procedure to the Navier-Stokes equations results in

$$\frac{\partial}{\partial t}(\overline{\rho}) + \vec{\nabla}\cdot(\overline{\rho}\tilde{\vec{u}}) = 0 \tag{2.20}$$

$$\frac{\partial}{\partial t}(\overline{\rho}\tilde{\vec{u}}) + \vec{\nabla}\cdot\left(\overline{\rho}\tilde{\vec{u}}\tilde{\vec{u}} + \vec{\vec{I}}\overline{p}\right) = \vec{\nabla}\cdot\left(\vec{\vec{\tau}} - \overline{\rho\vec{u}''\vec{u}''}\right) \tag{2.21}$$

$$\frac{\partial}{\partial t}(\overline{\rho}\tilde{E}) + \vec{\nabla}\cdot\left[\overline{\rho}\tilde{\vec{u}}\left(\tilde{E} + \frac{\overline{p}}{\overline{\rho}}\right)\right] = \vec{\nabla}\cdot\left[\tilde{\vec{u}}\cdot\left(\vec{\vec{\tau}} - \overline{\rho\vec{u}''\vec{u}''}\right) - \overline{\vec{q}} - \overline{\rho\vec{u}''h''} + \overline{\vec{\vec{\tau}}\cdot\vec{u}''} - \overline{\vec{u}''\frac{1}{2}\rho\vec{u}''\cdot\vec{u}''}\right] \tag{2.22}$$

where the Favre-averaged total energy is

$$\overline{\rho}\tilde{E} = \frac{\overline{p}}{\gamma-1} + \frac{1}{2}\overline{\rho}\tilde{\vec{u}}\cdot\tilde{\vec{u}} + \frac{1}{2}\overline{\rho\vec{u}''\cdot\vec{u}''} \tag{2.23}$$

As a result of the averaging procedure, additional terms appear in the equations. These turbulent correlations correspond to physical processes or properties of the turbulence and its affect on the mean flow, and must be modelled to obtain accurate mean-flow solutions. The correlation $-\overline{\rho\vec{u}''\vec{u}''}$ is the Reynolds-stress tensor, $\vec{\vec{\lambda}}$, and $\overline{\rho\vec{u}''h''}$ corresponds to the turbulent

transport of heat, $\overline{\vec{\tau}\cdot\vec{u}''}$ corresponds to the molecular diffusion of turbulent kinetic energy, and $\overline{\vec{u}''\frac{1}{2}\rho\vec{u}''\cdot\vec{u}''}$ corresponds to the turbulent transport of turbulent kinetic energy. The correlation between $\vec{u}''$ and itself that appears in Equation (2.23) is the so-called turbulent kinetic energy and is defined as

$$\overline{\rho}k = \frac{1}{2}\overline{\rho\vec{u}''\cdot\vec{u}''}. \tag{2.24}$$

An equation governing the transport of the turbulent kinetic energy, k, can be derived by multiplying the momentum equation by $\vec{u}''$ and time averaging to obtain

$$\frac{\partial}{\partial t}(\overline{\rho}k)+\vec{\nabla}\cdot(\overline{\rho}\overline{\vec{u}}k) = -\overline{\rho\vec{u}''\vec{u}''}:\vec{\nabla}\overline{\vec{u}}-\overline{\vec{\tau}:\vec{\nabla}\vec{u}''}+\vec{\nabla}\cdot\left(\overline{\vec{u}''\cdot\vec{\tau}}-\overline{\vec{u}''\frac{1}{2}\rho\vec{u}''\cdot\vec{u}''}-\overline{p'\vec{u}''}\right)-\overline{\vec{u}''}\cdot\vec{\nabla}\overline{p}+\overline{p'\vec{\nabla}\cdot\vec{u}''} \tag{2.25}$$

where the term $-\overline{\rho\vec{u}''\vec{u}''}:\vec{\nabla}\overline{\vec{u}}$ is an inner product of two dyadic quantities and corresponds to the production of turbulent kinetic energy due to energy transfer from mean to turbulent motion. The term $\overline{\vec{\tau}:\vec{\nabla}\vec{u}''}$ corresponds to the dissipation of turbulent kinetic energy, $\overline{\vec{u}''\cdot\vec{\tau}}$ corresponds to the molecular diffusion of turbulent kinetic energy, $\overline{\vec{u}''\frac{1}{2}\rho\vec{u}''\cdot\vec{u}''}$ corresponds to the turbulent transport for turbulent kinetic energy, and $\overline{p'\vec{u}''}$ corresponds to the pressure diffusion of turbulent energy.

DNS research has shown that the pressure diffusion term is very small for both mixing layers and boundary layers. Hence, the pressure diffusion correlation is generally neglected [4]. The last two terms in the transport equation for k correspond to the pressure work and pressure dilatation, respectively. Both these terms vanish in the limit of incompressible flow; when density fluctuations are zero, the time average of $\vec{u}''$ is zero and the velocity is divergence-free. This can be seen from the continuity equation which becomes $\vec{\nabla}\cdot\vec{u}=0$ under the incompressible assumption. Since this work deals with low-speed flows, these two terms will be neglected. The rest of the correlations require modelling to achieve closure.

2.3.1 Turbulence Modelling

In spite of the fact that a number of correlations can be neglected as described above, there are a number of terms in the FANS equations that require modelling. In this work, Wilcox's k-ω turbulence model is considered to achieve closure [4].

Boussinesq Approximation

In the two-equation k-ω turbulent model of Wilcox [4], the Boussinesq approximation is used to relate the Reynolds stress to the turbulent eddy-viscosity, μ_T, such that

$$\vec{\vec{\lambda}} = -\overline{\rho\vec{u}''\vec{u}''} = 2\mu_T\left(\vec{\nabla}\vec{u} - \frac{1}{3}\vec{\nabla}\cdot\vec{u}\vec{\vec{I}}\right) - \frac{2}{3}\overline{\rho}k\vec{\vec{I}}. \tag{2.26}$$

where μ_T is modelled in the k-ω turbulence model as $\mu_T = \frac{\overline{\rho}k}{\omega}$, and ω is the specific dissipation rate.

Dissipation of Turbulent Kinetic Energy

The dissipation rate is assumed to be proportional to turbulent kinetic energy such that

$$\overline{\vec{\vec{\tau}} : \vec{\nabla}\vec{u}''} = \overline{\rho}\tilde{\epsilon} = \beta^*\overline{\rho}k\omega \tag{2.27}$$

where β^* is a closure coefficient.

Turbulent Heat Flux

The turbulent heat-flux vector, $\overline{\rho\vec{u}''h''}$, found in the energy equation is assumed to be proportional to the mean temperature gradient such that

$$\overline{\rho\vec{u}''h''} = -\kappa_T\vec{\nabla}\tilde{T} \tag{2.28}$$

where κ_T is the turbulent thermal conductivity taken to have the form $\kappa_T = \mu_T c_p/\mathrm{Pr}_T$, and Pr_T is the turbulent Prandtl number, which is assumed to have a constant value of 0.9.

Molecular Diffusion and Turbulent Transport of Turbulent Kinetic Energy

The molecular diffusion and turbulent transport terms are commonly modelled as

$$\overline{\vec{u}''\cdot\vec{\vec{\tau}}} - \overline{\vec{u}''\frac{1}{2}\rho\vec{u}''\cdot\vec{u}''} = \left(\mu + \frac{\mu_T}{\sigma^*}\right)\vec{\nabla}k \tag{2.29}$$

where σ^* is another closure constant of the model.

Transport of Specific Dissipation Rate

Wilcox postulated a transport equation for the specific dissipation rate, ω, given by

$$\frac{\partial}{\partial t}(\overline{\rho}\omega) + \vec{\nabla}\cdot\left(\overline{\rho}\tilde{\vec{u}}\omega\right) = \alpha\frac{\omega}{k}\vec{\vec{\lambda}} : \vec{\nabla}\tilde{\vec{u}} + \vec{\nabla}\cdot\left[(\mu + \sigma\mu_T)\vec{\nabla}\omega\right] - \beta\overline{\rho}\omega^2 \tag{2.30}$$

where σ and β are further closure coefficients.

Combining all the assumptions and modelled correlations results in the following set of coupled partial differential equations that provide a description of the time evolution of the mean-flow quantities, $\overline{\rho}$, $\tilde{\vec{u}}$, $\overline{p}$, k, and ω:

$$\frac{\partial}{\partial t}(\overline{\rho}) + \vec{\nabla}\cdot(\overline{\rho}\tilde{\vec{u}}) = 0 \tag{2.31}$$

$$\frac{\partial}{\partial t}(\overline{\rho}\tilde{\vec{u}}) + \vec{\nabla}\cdot\left[\overline{\rho}\tilde{\vec{u}}\tilde{\vec{u}} + \left(\overline{p} + \frac{2}{3}\overline{\rho}k\right)\overline{\overline{I}}\right] = \vec{\nabla}\cdot\overline{\overline{\tau}} \tag{2.32}$$

$$\frac{\partial}{\partial t}(\overline{\rho}\tilde{E}) + \vec{\nabla}\cdot\left[\overline{\rho}\tilde{\vec{u}}\left(\tilde{E} + \frac{\overline{p}}{\overline{\rho}} + \frac{2}{3}k\right)\right] = \vec{\nabla}\cdot\left[\tilde{\vec{u}}\cdot\overline{\overline{\tau}} - \overline{\vec{q}} - \overline{\rho\vec{u}''h''} + (\mu + \sigma^*\mu_T)\vec{\nabla}k\right] \tag{2.33}$$

$$\frac{\partial}{\partial t}(\overline{\rho}k) + \vec{\nabla}\cdot(\overline{\rho}\overline{\vec{u}}k) = \overline{\vec{\lambda}} : \vec{\nabla}\tilde{\vec{u}} + \vec{\nabla}\cdot\left[(\mu + \sigma^*\mu_T)\vec{\nabla}k\right] - \beta^*\overline{\rho}k\omega \tag{2.34}$$

$$\frac{\partial}{\partial t}(\overline{\rho}\omega) + \vec{\nabla}\cdot(\overline{\rho}\tilde{\vec{u}}\omega) = \alpha\frac{\omega}{k}\overline{\vec{\lambda}} : \vec{\nabla}\tilde{\vec{u}} + \vec{\nabla}\cdot\left[(\mu + \sigma\mu_T)\vec{\nabla}\omega\right] - \beta\overline{\rho}\omega^2 \tag{2.35}$$

Note that the stress tensor, $\overline{\overline{\tau}}$, now includes contributions from the laminar and Reynolds stress tensors such that

$$\overline{\overline{\tau}} = 2(\mu + \mu_T)(\vec{\nabla}\tilde{\vec{u}} - \frac{1}{3}\vec{\nabla}\cdot\tilde{\vec{u}}\overline{\overline{I}}). \tag{2.36}$$

2.3.2 Turbulence Closure Coefficients

The following closure coefficients for the k-ω turbulence model as given by Wilcox [4] and are as follows:

$$\alpha = \frac{13}{25}, \quad \beta = \beta_o f_\beta, \quad \beta_o = \frac{9}{125}, \quad \beta^* = \beta_o^* f_{\beta^*}, \quad \beta_o^* = \frac{9}{100}, \quad \sigma = \sigma^* = \frac{1}{2} \tag{2.37}$$

$$f_{\beta^*} = \begin{cases} 1, & \chi_k \le 0 \\ \frac{1+680\chi_k^2}{1+80\chi_k^2}, & \chi_k > 0 \end{cases} \quad \text{where} \quad \chi_k = \frac{1}{\omega^3}\vec{\nabla}k\cdot\vec{\nabla}\omega \tag{2.38}$$

$$f_\beta = \frac{1+70\chi_\omega}{1+80\chi_\omega} \quad \text{where} \quad \chi_\omega = \left|\frac{(\overline{\overline{\Omega}}\otimes\overline{\overline{\Omega}}) : \overline{\overline{S}}}{(\beta_o^*\omega)^3}\right| \tag{2.39}$$

where $\overline{\overline{\Omega}}$ and $\overline{\overline{S}}$ are the mean rotation and mean strain rate tensors, respectively.

2.3.3 Near-Wall Boundary Conditions

Specific turbulent flow regimes can be identified in the near-wall region through the use of the non-dimensional wall distance, y^+, and non-dimensional velocity, u^+, defined as

$$y^+ = \frac{u_\tau y}{\nu}, \quad u^+ = \frac{u}{u_\tau} \tag{2.40}$$

where y is the dimensional wall distance, $u_\tau = \sqrt{\frac{\tau_w}{\rho}}$ is the friction velocity and $\tau_w = \mu \left.\frac{\partial u}{\partial y}\right|_{y=0}$ is the wall stress. Traditionally, u is taken to be the magnitude of velocity component tangential to the wall. However, in some cases, it can be convenient to take u to be the magnitude of the velocity vector. This is done in practical engineering applications so that y^+ will not be zero when the flow is predominately perpendicular to the wall surface, such as near corners and in recirculation regions. Figure 2.2 illustrates the various near-wall turbulent flow regimes and law-of-the-wall turbulence.

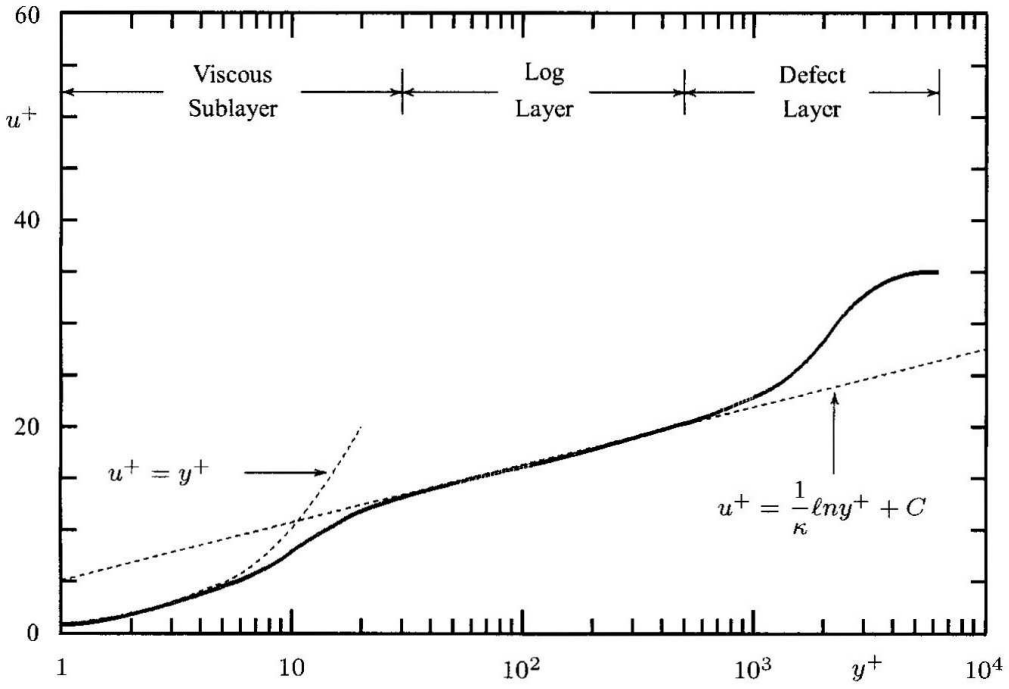


Figure 2.2: Boundary layer profile showing near-wall regimes of turbulence, law-of-the-wall [4].

The region in which $y^+ \leq 30$ is known as the viscous sublayer. In this regime, the flow is dominated by viscous effects and

$$u^+ = y^+. \tag{2.41}$$

The region where $30 \leq y^+ \leq 500-2000$ is known as the log layer. In the log layer, the velocity profile follows the law of the wall,

$$u^+ = \frac{1}{\kappa}\ln y^+ + C \tag{2.42}$$

where $C \approx 5$ and $\kappa \approx 0.41$. Beyond the log layer, lies the defect layer. When solving the above equation, an iterative scheme can be used to calculate the values of y^+ and u_τ simultaneously.

Appropriate boundary conditions for k and ω must be applied for grid points within the sublayer and log layer. One approach is the use of wall functions, which attempt to mimic the law of

the wall [4]. In this approach, the equations for k and ω become

$$k = \frac{u_\tau^2}{\sqrt{\beta_o^*}}, \quad \omega = \frac{u_\tau}{\sqrt{\beta_o^*}\kappa y}. \tag{2.43}$$

The application of wall functions is sensitive to mesh resolution and significant numerical errors can result. To circumvent these difficulties, points lying within the viscous sublayer can be integrated directly so that

$$\omega = \frac{6\nu}{\beta y^2}, \quad y^+ < 2.5 \tag{2.44}$$

in this regime. However, direct integration requires sufficient grid resolution within $y^+ < 2.5$; typically 7 to 10 grid points above the surface is required. Checking for sufficient mesh resolution is straightforward for body-fitted structured grids, but is more difficult for regions where the surface is not aligned with the mesh.

An alternative approach that does not rely on mesh resolution is the slightly-rough-surface boundary condition, where

$$\omega = \frac{40000\nu}{k_s^2} \tag{2.45}$$

where k_s is the surface-roughness height [4]. A hydraulically smooth surface is ensured via the scaled surface-roughness height, $k_s^+ = \frac{u_\tau k_s}{\nu}$, such that $k_s^+ < 5$.

Chapter 3

Finite-Volume Scheme

Coupled systems of partial differential equations, such as the ones presented in the previous chapter, can be solved using a variety numerical algorithms. In the field of computational fluid dynamics, finite-difference and finite-volume methods are the predominant algorithms of choice. In the present work, the finite-volume method is used to solve the governing equations.

The Euler equations, (2.7), (2.8) and (2.9), are solved herein using a Godunov-type high-resolution upwind finite-volume method [20]. In this method, the domain of interest is discretized into finite volumes or computational cells. In this work, a two-dimensional scheme utilizing quadrilateral cells has been developed. Solution quantities are explicitly conserved within each cell, and the cells can be dynamically adjusted to accommodate moving embedded boundaries within the domain [16, 21]. This chapter summarizes the basics of the finite volume method.

3.1 Integral Form of the Euler Equations

The Euler equations in integral form are given by

$$\frac{d}{dt}\int_{A(t)} \mathbf{U} dA + \oint_{\partial A(t)} [\mathbf{F}(\mathbf{U}) - \vec{w}\mathbf{U}] \cdot \hat{n} dl = 0 \tag{3.1}$$

where $\mathbf{U}$ is the vector of conserved quantities with

$$\mathbf{U} = \left[\ \rho, \ \rho u, \ \rho v, \ \rho E \ \right]^T , \tag{3.2}$$

and $\mathbf{F}(\mathbf{U}) = [\mathbf{F_x}, \mathbf{F_y}]$ is the inviscid flux dyad, whose components in the x- and y-direction are given by

$$\mathbf{F_x} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho u \left(E + \frac{p}{\rho} \right) \end{bmatrix}, \quad \mathbf{F_y} = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho v \left(E + \frac{p}{\rho} \right) \end{bmatrix}. \tag{3.3}$$

The variables $A(t)$ and $\partial A(t)$ are the area and perimeter of the finite volume, respectively; $\hat{n}$ is the outward unit normal of the cell face, dl is the differential length along the perimeter path. Since the geometry of the computational cell may be changing with time, the term $\vec{w}\mathbf{U}$ is present to account for fluxes that result from a cell interface moving at velocity $\vec{w}$.

3.2 Semi-Discrete Form

In the present finite-volume scheme, the computational domain is discretized into structured quadrilateral cells. An averaged value for the conserved solution vector is computed for each cell, where the cell-averaged solution in cell (i,j), $\overline{\mathbf{U}}_{ij}$ of the structured grid is defined as

$$\overline{\mathbf{U}}_{ij} = \frac{1}{A} \int_{A(t)} \mathbf{U} dA \tag{3.4}$$

Equation (3.4) can be applied to the unsteady term in Equation (3.1) to yield

$$\frac{d}{dt} \int_{A(t)} \mathbf{U} dA = \frac{d}{dt} (\overline{\mathbf{U}} A) = A \frac{d\overline{\mathbf{U}}}{dt} + \overline{\mathbf{U}} \frac{dA}{dt} \tag{3.5}$$

The last term, $\frac{dA}{dt}$, refers to the rate of change of the area of the quadrilateral cell. It can be approximated using a geometric conservation law such that

$$\frac{dA}{dt} = \sum_k \vec{w}_k \cdot \hat{n}_k \Delta l_{ij_k} \tag{3.6}$$

The closed integrals of Equation (3.1) can also be simplified by considering quadrilateral cells with k faces. The equivalent flux can be obtained by summing the flux contributions through each of the k faces. Hence,

$$\oint_{\partial A(t)} [\mathbf{F}(\mathbf{U}) - \vec{w}\mathbf{U}] \cdot \hat{n} dl = \sum_k (\mathbf{F}_k - \vec{w}_k \mathbf{U}_k) \cdot \hat{n_k} \Delta l_{ij_k} \tag{3.7}$$

Substituting these expressions into Equation (3.1) and rearranging results in the following semi-discrete form of the Euler equations

$$\frac{d\overline{\mathbf{U}}_{ij}}{dt} = -\frac{1}{A_{ij}} \sum_k (\mathbf{F}_k - \vec{w}_k \mathbf{U}_k) \cdot \hat{n}_{ij_k} \Delta l_{ij_k} - \frac{\overline{\mathbf{U}}_{ij}}{A_{ij}} \left(\sum_k \vec{w}_k \cdot \hat{n}_k \Delta l_{ij_k} \right) \tag{3.8}$$

Equation (3.8) is a coupled non-linear system of first-order ordinary differential equations that can be solved using standard time-marching schemes and/or iterative solution methods.

3.3 Low-Mach-Number Preconditioning

The operation of wind turbines falls into the low-speed regime, with Mach numbers ranging only from 0.005 (light breeze) to 0.05 (gale force winds). At these speeds, the flow is considered to be largely incompressible. Solving incompressible flows using a density-based (compressible) flow solution algorithm can present several numerical challenges.

In explicit time-marching schemes, the maximum allowable time-step size is inversely proportional to the largest wave speed or eigenvalue of the system of governing equations. For inviscid flows, the largest eigenvalue is related the acoustic wave speed, $u + a$. Since u is small for low-speed flows, there is a large disparity between the acoustic wavespeed and the convective wavespeed, e.g. $u + a \gg u$. As the Mach number approaches zero, the condition number, defined as the ratio of the largest and smallest eigenvalues, approaches infinity (see Figure 3.1). Therefore, the time-step size is restricted to be significantly smaller than the time-scale of the convective flow, which can result in extremely slow convergence rates, excessive numerical dissipation, and high computational costs per solution [22].

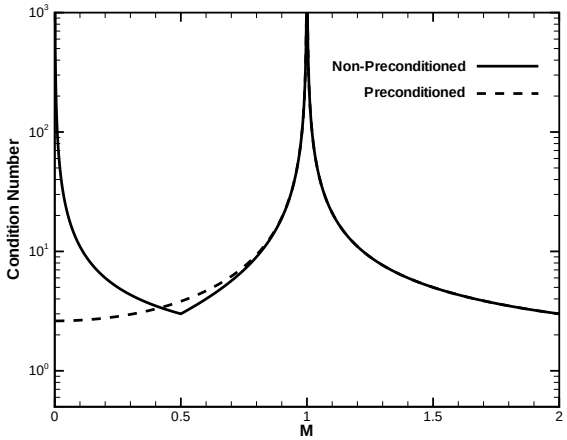


Figure 3.1: Condition number for non-preconditioned and preconditioned inviscid form of the conservation equations.

While it is possible to avoid this problem by solving the incompressible Navier-Stokes equations instead, it is highly desirable to have a single unified solution method that is applicable to a wide range of flow conditions. To this end, a local preconditioner matrix is introduced in the

semi-discrete form of the governing equations so that the large disparity between the eigenvalues is reduced in the low-Mach-number limit. Preconditioning techniques have been successfully used by many researchers to enable simulations of low-speed flows using compressible solvers [22, 23, 24, 25, 26].

The principle of local preconditioning involves pre-multiplying the time derivative within each cell (i,j) by a preconditioner matrix $\mathbf{\Gamma}$. The preconditioner is introduced in such a way so as not to affect the steady-state solution as $\frac{d\overline{\mathbf{U}}}{dt}$ approaches zero. Applying the preconditioner matrix $\mathbf{\Gamma}$ to the semi-discrete form of the Euler equations of interest here yields

$$\mathbf{\Gamma_{ij}}\frac{d\overline{\mathbf{U}}_{ij}}{dt} = -\frac{1}{A_{ij}}\sum_k (\mathbf{F}_k - \vec{w}_k\mathbf{U}_k)\cdot\hat{n}_{ij_k}\Delta l_{ij_k} - \frac{\overline{\mathbf{U}}_{ij}}{A_{ij}}\left(\sum_k \vec{w}_k\cdot\hat{n}_k\Delta l_{ij_k}\right). \tag{3.9}$$

Note that the presence of the preconditioner in this form results in a system of equations that is no longer time accurate.

In this work, we use the local preconditioner developed by Weiss and Smith [18]. The preconditioning matrix, in terms of the conserved variables in this case is given by

$$\mathbf{\Gamma} = \frac{1}{a^2M_r^2}\begin{bmatrix} -V\alpha + a^2M_r^2 & u\alpha & v\alpha & -\alpha \\ -Vu\alpha & u^2\alpha + a^2M_r^2 & uv\alpha & -u\alpha \\ -Vv\alpha & uv\alpha & v^2\alpha + a^2M_r^2 & -v\alpha \\ -V\delta & u\delta & v\delta & -\delta + a^2M_r^2 \end{bmatrix}, \tag{3.10}$$

where

$$\begin{aligned} V &= \frac{u^2+v^2}{2} \\ \alpha &= (M_r^2-1)(\gamma-1) \\ \delta &= (M_r^2-1)\left[\left(\frac{u^2+v^2}{2}\right)(\gamma-1) + a^2\right] \end{aligned}$$

and M_r is the reference Mach number defined for inviscid flows as

$$M_r = \min\left(\max\left(\frac{u}{a}, M_{\mathrm{r_{min}}}\right), 1\right). \tag{3.11}$$

The (i,j) notation has been omitted for clarity and the above matrix is understood to be applied to each cell in the computational domain.

The eigenvalues of the preconditioned system $\lambda(\Gamma^{-1}\frac{\partial\mathbf{F}}{\partial\mathbf{U}})$ can be determined and are

$$\lambda_1 = u' - a', \quad \lambda_{2,3} = u, \quad \lambda_4 = u' + a'. \tag{3.12}$$

The preconditioned velocity, u', and preconditioned sound speed, a', are given by

$$u' = u(1-\alpha) \tag{3.13}$$

$$a' = \sqrt{\alpha^2u^2 + M_r^2a^2} \tag{3.14}$$

with

$$\alpha = \frac{1}{2}\left(1 - M_r^2\right). \tag{3.15}$$

This preconditioner is readily extended to viscous flows by considering the characteristic form of the Navier-Stokes equations. See Appendix B for the eigenstructure analysis of the non-preconditioned system, and Appendix C for the eigenstructure analysis of the resulting preconditioned system.

3.4 Temporal Discretization

3.4.1 Explicit Time-Marching Scheme

The right-hand side of the semi-discrete form of the equations is termed the residual, $\mathbf{R}$, such that the semi-discrete form can be re-cast as

$$\mathbf{\Gamma}\frac{d\overline{\mathbf{U}}}{dt} = -\mathbf{R}(\mathbf{U}) \tag{3.16}$$

For steady state problems, the residual is driven to zero. In this thesis, the explicit optimally-smoothing multi-stage time-marching schemes by van Leer *et al.* are used to integrate Equation (3.16) forwarde in time and arrive at the steady, time-invariant solution [1, 27]. These schemes are not time-accurate and have been designed to provide optimal damping of the high frequency content of the solution, making them ideal for use with multigrid. The M-stage schemes advance the solution from time level n to $n+1$ through the following operations:

$$\overline{\mathbf{U}}_0 = \overline{\mathbf{U}}_n \tag{3.17}$$

$$\overline{\mathbf{U}}_k = \overline{\mathbf{U}}_0 - \alpha_m \Delta t \mathbf{\Gamma}^{-1}\mathbf{R}(\overline{\mathbf{U}}_{k-1}) \quad \text{for} \quad k = 1, \ldots, M \tag{3.18}$$

$$\overline{\mathbf{U}}_{n+1} = \overline{\mathbf{U}}_M \tag{3.19}$$

where α_m are the multi-stage coefficients given in Table 3.1.

To accelerate the convergence, a Full-Approximate Storage (FAS) multigrid method is used in which the residual is reduced over a set of successively coarser grids. Coarser grids are generated by removing every other node for the desired number of multigrid levels (Figure 3.2). The solution is successively restricted onto the coarser grids and the restricted solution is smoothed using the time-marching method described previously. After the desired number of smooths have been performed, the coarse grid solution is prolonged back onto the original (fine) mesh. A detailed explanation of the restriction and prolongation process can be found in

M	2	3	4	5	6
α_1	0.4242	0.1918	0.1084	0.0695	0.0482
α_2	1	0.4929	0.2602	0.1602	0.1085
α_3		1	0.5052	0.2898	0.1885
α_4			1	0.5060	0.3050
α_5				1	0.5063
α_6					1

Table 3.1: Multi-stage coefficients for second order optimal smoothing [1].

Li [28] and Lomax *et al* [8]. A number of multigrid parameters can be selected to improve the convergence of the algorithm: the number of pre- and post-smooths, the number of smooths on the finest and coarsest levels, and the multigrid cycle type (V- or W-cycle).

3.4.2 Dual Time-Stepping Scheme for Time-Accurate Problems

A dual time-stepping (DTS) algorithm is used to enable larger time steps in an effort to further reduce computational cost. It is also used to restore time accuracy when preconditioning is employed. A number of researchers have applied this technique with great success. Among them, Jameson pioneered the approach by applying it to the Euler equations with multigrid acceleration for unsteady flows past airfoils and wings [19], Weiss and Smith used dual time-stepping within an explicit, multistage algorithm with preconditioning for low-speed flows [18], and Daily and Pletcher used dual time-stepping with multigrid acceleration with the preconditioner of Daily and Chen [29] to solve the unsteady Navier-Stokes equations in two-dimensions [30].

In the dual time-stepping approach, a pseudo-time derivative, $\frac{d\overline{\mathbf{U}}}{d\tau}$, is introduced to the original semi-discrete formulation in Equation (3.16) so that the coupled ordinary differential equations become

$$\mathbf{\Gamma}\frac{d\overline{\mathbf{U}}}{d\tau} + \frac{d\overline{\mathbf{U}}}{dt} + \mathbf{R}(\mathbf{U}) = 0. \tag{3.20}$$

Defining the modified residual as

$$\mathbf{R}^*(\mathbf{U}) = \frac{d\overline{\mathbf{U}}}{dt} + \mathbf{R}(\mathbf{U}), \tag{3.21}$$

results in the psuedo-steady problem of the form

$$\mathbf{\Gamma}\frac{d\overline{\mathbf{U}}}{d\tau} + \mathbf{R}^*(\mathbf{U}) = 0. \tag{3.22}$$

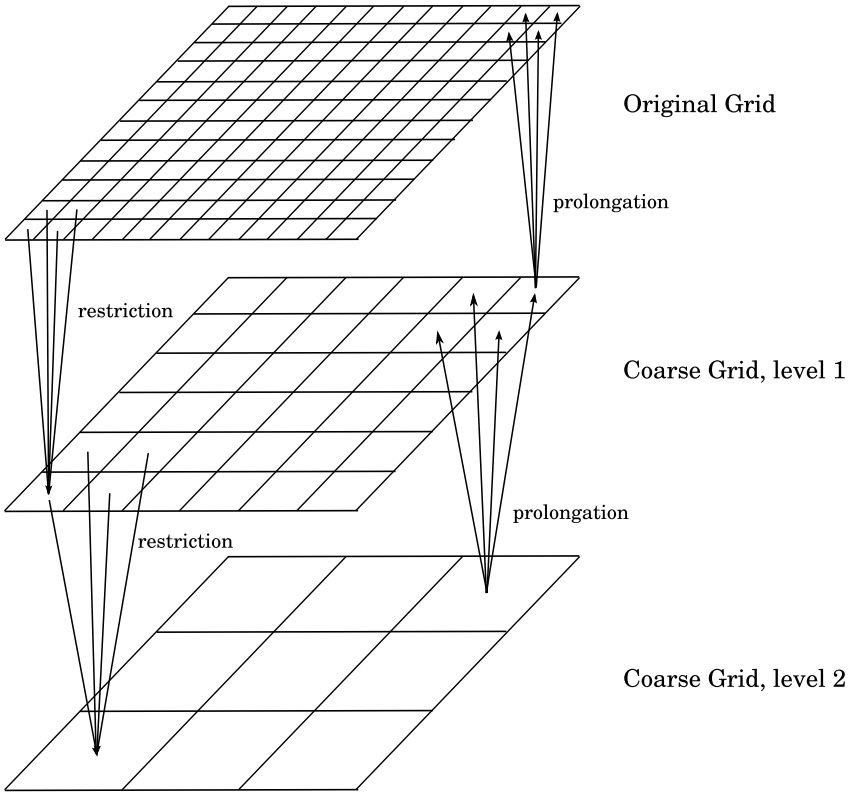


Figure 3.2: Multigrid algorithm with successively coarser grids.

The modified residual is driven to zero using the convergence techniques discussed in the previous section. When the psuedo-time derivative is equal to zero, the original formulation is recovered.

In this work, the physical-time derivative, $\frac{d\overline{\mathbf{U}}}{dt}$, is discretized using a second-order backwards difference formula (BDF) such that

$$\frac{d\overline{\mathbf{U}}}{dt} = \frac{3\overline{\mathbf{U}}_{n+1} - 4\overline{\mathbf{U}}_n + \overline{\mathbf{U}}_{n-1}}{2\Delta t} \tag{3.23}$$

This scheme is unconditionally stable, permitting arbitrarily large CFL numbers to be chosen. The dual time-stepping approach has a significant advantage over explicit time-marching schemes, which are restricted to CFL numbers of less than 1.

3.4.3 CFL Stability Criterion

The pseudo-time step, $\Delta\tau$, is limited by the Courant-Friedrich-Lewy (CFL) stability criterion imposed by the explicit optimally-smoothing multi-stage scheme.

$$\Delta\tau = \beta_{\text{pseu}} \min\left(\frac{A}{(u' + a')\Delta l_k}\right) \tag{3.24}$$

where β_{pseu} is the pseudo-time CFL number, A is the cell area, Δl_k is the length of the k^{th} cell face, w_k is the interface speed, u' and a' are the preconditioned convective and sound speeds, respectively.

The physical-time step, Δt, for the algorithm can now be selected to be arbitrarily large as the BDF scheme is unconditionally stable. However, a constraint is imposed if dynamically moving embedded boundaries are present. Since the motion of the boundaries is treated explicitly, the physical-time CFL number cannot exceed 1. Time step restrictions may also be required for reasons of solution accuracy and will be shown and discussed in Chapter 4 to follow. Moreover, restrictions on Δx must also be imposed to ensure accuracy of the unsteady solutions.

3.5 Inviscid Flux Evaluation

In the Godunov method, the inviscid flux vector is determined numerically by solving a Riemann problem composed of the piecewise-constant solution states to the left and right of the cell interface [20]. That is,

$$\mathcal{F}_{i+\frac{1}{2}} = \mathbf{F}(\mathcal{R}(\mathbf{U_L}, \mathbf{U_R})) \tag{3.25}$$

where $\mathcal{F}$ is the numerical flux evaluated at the cell interface $x_{i+\frac{1}{2}}$, $\mathcal{R}$ is the solution to the Riemann problem at the cell interface, $\mathbf{U_L}$ is the state evaluated at x_i and $\mathbf{U_R}$ is the state evaluated at x_{i+1}.

In Godunov's original method, the Riemann problem is solved exactly for the evaluation of the numerical flux. This approach can be accomplished through an iterative method as proposed by Gottlieb and Groth [31] for the Euler equations. Approximate Riemann solvers have also been developed by Roe [32], Harten, Lax, and van Leer [33], Einfeldt [34], and Linde [35] and are also suitable. The latter benefit from being more readily applicable to other hyperbolic systems of equations. In this work, the AUSM$^+$-up flux function of Liou [36] is used exclusively in the evaluation of the numerical flux at the cell boundaries. A discussion of this flux function now follows.

3.5.1 AUSM$^+$-up Numerical Flux Function

The Advection Upstream Splitting Method (AUSM) family of flux functions is one approach to solving the fluxes at the cell interface. Here, the flux vector is split into a convective term and a pressure term. Upwinding is accomplished through the use of an interface Mach number that is a function of the left and right states. The seminal work by Liou [36] on the original AUSM scheme has prompted a number of extensions and variations that have been applied to a wide range of flows [37, 38, 39] Among the several extensions is the AUSM$^+$-up scheme developed by Liou [17] which has been shown to be effective over a wide range of flow speeds, especially in the low-Mach-number limit. It is used in this work and has been extended for use in conjection with the FANS equations. The specifics of the algorithm are presented here for the one-dimensional case; the extension to multiple dimensions is straightforward using direction splitting.

The inviscid flux vector as presented in Section 3.1 is explicitly split into a convective flux $\mathbf{F_c}$ and a pressure flux $\mathbf{P}$ such that

$$\mathbf{F_x} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho u \left(E + \frac{p}{\rho} \right) \end{bmatrix} = \mathbf{F_c} + \mathbf{P}. \tag{3.26}$$

The convective flux is further decomposed into a common scalar flux mass flux $\dot{m}$ and a vector quantity $\vec{\psi}$ that is convected by $\dot{m}$. That is, $\mathbf{F_c} = \dot{m}\vec{\psi}$, where

$$\dot{m} = \rho u, \quad \vec{\psi} = \begin{bmatrix} 1 \\ u \\ v \\ h \end{bmatrix}, \quad \mathbf{P} = \begin{bmatrix} 0 \\ p + \frac{2}{3}\rho k \\ 0 \\ 0 \end{bmatrix}. \tag{3.27}$$

The numerical flux at the cell interface is formulated similarly,

$$\mathcal{F}_{1/2} = \dot{m}_{1/2}\vec{\psi}_{L/R} + \mathbf{P}_{1/2} \tag{3.28}$$

where $\vec{\psi}_{L/R}$ is determined in a simple upwind fashion as

$$\vec{\psi}_{L/R} = \begin{cases} \vec{\psi}_L & \text{if } \dot{m}_{1/2} > 0 \\ \vec{\psi}_R & \text{otherwise} \end{cases} \tag{3.29}$$

To complete the flux calculation, the value of $\dot{m}_{1/2}$ and $\mathbf{P}_{1/2}$ must be computed.

The mass flux at the interface is taken to have the form

$$\dot{m}_{1/2} = a_{1/2} M_{1/2} \begin{cases} \rho_L & \text{if } M_{1/2} > 0 \\ \rho_R & \text{otherwise} \end{cases} \tag{3.30}$$

where $M_{1/2}$ is the interface Mach number and $a_{1/2}$ is a simple average of the left and right sound speeds. To determine $M_{1/2}$, the left and right Mach numbers are defined as

$$M_L = \frac{u_L}{a_{1/2}}, \quad M_R = \frac{u_R}{a_{1/2}}. \tag{3.31}$$

Next, a mean local Mach number, $\bar{M}$, and reference Mach number, M_o, are calculated as

$$\bar{M}^2 = \frac{u_L^2 + u_R^2}{2a_{1/2}^2} \tag{3.32}$$

$$M_o^2 = min\left(1, max\left(\bar{M}^2, M_{r_{min}}^2\right)\right) \in [0,1] \tag{3.33}$$

A scaling function, f_a, is calculated to properly scale the numerical dissipation with the flow speed. This function has the form

$$f_a(M_o) = M_o(2 - M_o) \in [0,1] \tag{3.34}$$

Finally, the interface Mach number, $M_{1/2}$, is calculated as

$$M_{1/2} = \mathscr{M}_{(4)}^{+}(M_L) + \mathscr{M}_{(4)}^{-}(M_R) - \frac{K_p}{f_a} max(1 - \sigma\bar{M}^2, 0)\frac{p_R - p_L}{\rho_{1/2}a_{1/2}^2} \tag{3.35}$$

where $\rho_{1/2} = \frac{1}{2}(\rho_L + \rho_R)$, $0 \le K_p \le 1$ and $\sigma \le 1$. The split Mach numbers $\mathscr{M}_{(m)}^{\pm}$ are polynomial functions of degree m, and are given as

$$\mathscr{M}_{(1)}^{\pm} = \frac{1}{2}(M \pm |M|) \tag{3.36}$$

$$\mathscr{M}_{(2)}^{\pm} = \pm\frac{1}{4}(M \pm 1)^2 \tag{3.37}$$

$$\mathscr{M}_{(4)}^{\pm} = \begin{cases} \mathscr{M}_{(1)}^{\pm} & \text{if } |M| \ge 1 \\ \mathscr{M}_{(2)}^{\pm}(1 \mp 16\beta\mathscr{M}_{(2)}^{\mp}) & \text{otherwise} \end{cases} \tag{3.38}$$

with $\beta = 1/8$.

The pressure flux is computed as

$$p_{1/2} = \mathscr{P}_{(5)}^{+}(M_L)p_L + \mathscr{P}_{(5)}^{-}(M_R)p_R - K_u\mathscr{P}_{(5)}^{+}\mathscr{P}_{(5)}^{-}(\rho_L + \rho_R)(f_a a_{1/2})(u_R - u_L) \tag{3.39}$$

where $0 \le K_u \le 1$ and

$$\mathscr{P}_{(5)}^{\pm} = \begin{cases} \frac{1}{M}\mathscr{M}_{(1)}^{\pm} & \text{if } |M| \ge 1 \\ \mathscr{M}_{(2)}^{\pm}\left[(\pm 2 - M) \mp 16\alpha M\mathscr{M}_{(2)}^{\mp}\right] & \text{otherwise} \end{cases} \tag{3.40}$$

with

$$\alpha = \frac{3}{16}\left(-4 + 5f_a^2\right) \in \left[-\frac{3}{4}, \frac{3}{16}\right] \tag{3.41}$$

The AUSM$^+$-up scheme can be extended to the Favre-averaged form of the Navier-Stokes equations in a rather straightforward manner. See Appendix A for a detailed description of this extended flux function, which has been developed as part of this research.

3.6 Higher-Order Spatial Accuracy

Using the left and right cell-averaged solution values from cells adjacent to the interface directly in the flux evaluations results in first-order spatial accuracy. Higher-order accuracy can be obtained by extending the piece-wise constant representation to a piece-wise linear representation. This is accomplished by computing the gradients, $\frac{\partial \overline{\mathbf{U}}}{\partial x}$ and $\frac{\partial \overline{\mathbf{U}}}{\partial y}$, at the cell centers using a least-squares approach. Second-order accuracy is achieved by using reconstructed left and right states at the interface in the flux evaluations. The reconstructed left and right states at the cell interface, $i + \frac{1}{2}$, is given by

$$\overline{\mathbf{U}}_{i+\frac{1}{2},L} = \overline{\mathbf{U}}_i + \phi_i \vec{\nabla}\mathbf{U}\Big|_i \cdot (\mathbf{x}_{i+\frac{1}{2}} - \mathbf{x}_i) \tag{3.42}$$

$$\overline{\mathbf{U}}_{i+\frac{1}{2},R} = \overline{\mathbf{U}}_{i+1} - \phi_{i+1} \vec{\nabla}\mathbf{U}\Big|_{i+1} \cdot (\mathbf{x}_{i+1} - \mathbf{x}_{i+\frac{1}{2}}) \tag{3.43}$$

where $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$ are the position vectors of the cell-centers at cell i and $i+1$, respectively, and $\mathbf{x}_{i+\frac{1}{2}}$ is the position vector at the cell interface. $\vec{\nabla}\mathbf{U}\Big|_i$ is the reconstructed gradient evaluated at cell i and ϕ_i is the slope limiter at cell i. By limiting the gradients near discontinuities, the scheme is reduced to first-order and solution monotonicity is preserved. The limiter developed by Venkatakrishnan is used in this work [40].

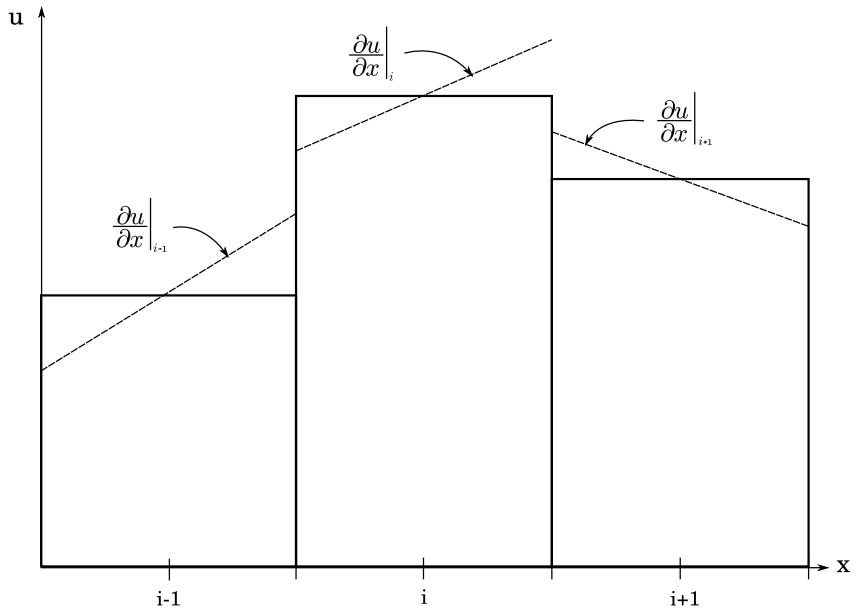


Figure 3.3: Linear reconstruction applied to cell-averaged quantities.

3.7 Mesh Adjustment Scheme for Embedded Boundaries

The irregular geometry of the vertical-axis wind turbine requires special meshing considerations since the embedded boundaries are not aligned with the background structured mesh. Various approaches to this problem have been proposed in the literature, including the cut-cell [41, 42, 43] and the immersed boundary methods [44, 45].

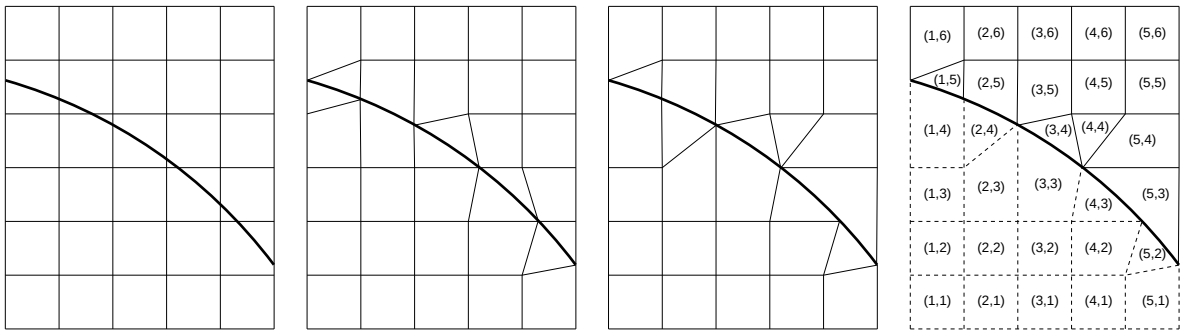
In this work, the mesh adjustment scheme developed by Sachdev *et al.* [21] is used. This method has the advantage of maintaining the (i, j) data structure of the structured mesh. The resulting mesh also allows for straightforward application of boundary conditions at the embedded boundary. A thorough description of the adjustment algorithm is presented in Sachdev *et al.* [21] and is summarized here for completeness.

To illustrate the algorithm, an embedded boundary and initial mesh is given in Figure 3.4(a). The first step of the algorithm involves tagging computational cells via a ray-tracing procedure. Cells are tagged as either internal to the interface, external to the interface, or identified as "unknown" if the cell is intersected by the interface. Unknown cells that are candidates for adjustment.

The primary adjustment involves relocating the mesh nodes that are closest to the interface point the intersection point between the spline defining the embedded boundary and the mesh lines. The resulting adjusted grid is shown in Figure 3.4(b). A secondary adjustment is required to account for computational cells that are bisected diagonally by the embedded boundary. This is accomplished by choosing the closest not-aligned node to the embedded boundary and relocating it to that boundary point, as shown in Figure 3.4(c).

The final step of the algorithm involves re-tagging the computational cells as either internal or external to the interface. Internal cells are deemed "inactive" and are not used in the computation. In the figure, "inactive" cells are denoted by dashed lines while "active" cells are denoted by solid lines. The resulting adjusted mesh retains the (i, j) data structure of the original mesh, as shown in Figure 3.4(d).

Mesh adjustment for moving embedded boundaries is accomplished by reverting the grid to the original unadjusted form, moving the interface to the new location, and then readjusting the mesh again. This is performed at each time step. A limitation of the adjustment scheme is encountered when the embedded boundary is a thin closed surface. Sufficient mesh resolution must be used to ensure at least several computational cells are within the boundary after adjustment. Otherwise, an adjustment that results in a zero width surface may result.



(a) (b) (c) (d)

Figure 3.4: Mesh adjustment algorithm: (a) initial mesh, (b) after primary adjustment, (c) after secondary adjustment, (d) final adjusted mesh. The embedded boundary is represented by the thick line. Dash lines represent inactive cells and solid lines represent active cells.

3.8 Block-Based Adaptive Mesh Refinement

A block-based adaptive mesh refinement (AMR) technique is used to cluster computational cells in areas of interest and is highly effective in treating problems with disparate length scales. AMR has been used with great success by a number of researchers [46, 41, 47] and has recently been applied on meshes with dynamic embedded boundaries [16, 48]. The block-based AMR method adopted here is fully compatible with the proposed solution algorithm and embedded mesh treatments.

In the present technique, cells are grouped into blocks of N_i by N_j, where N_i and N_j are even. Physics-based or interface-based refinement criteria are computed to determine whether a block should be refined or coarsened. When flagged for refinement, a "parent" block is subdivided into four "child" blocks, each having the same number of cells as its parent, effectively doubling the cell resolution. When blocks are flagged for coarsening, the process is reversed: four "child" blocks are coarsened into a single "parent" block. The connectivity between solution blocks is stored in a quadtree data structure (see Figure 3.5). The mesh refinement is constrained such that the difference in resolution between adjacent blocks is not greater than a factor of two and blocks are not coarsened beyond the resolution of the initial mesh. Standard multigrid-type restriction and prolongation operators are used to evaluate the solution on all blocks created by the coarsening and division processes, respectively.

The AMR technique greatly reduces the memory footprint and computational cost when used with thin embedded boundaries as the mesh resolution required to accurately adjust the mesh can be significantly finer than is necessary to solve the flow. Figure 3.6 illustrates the refinement

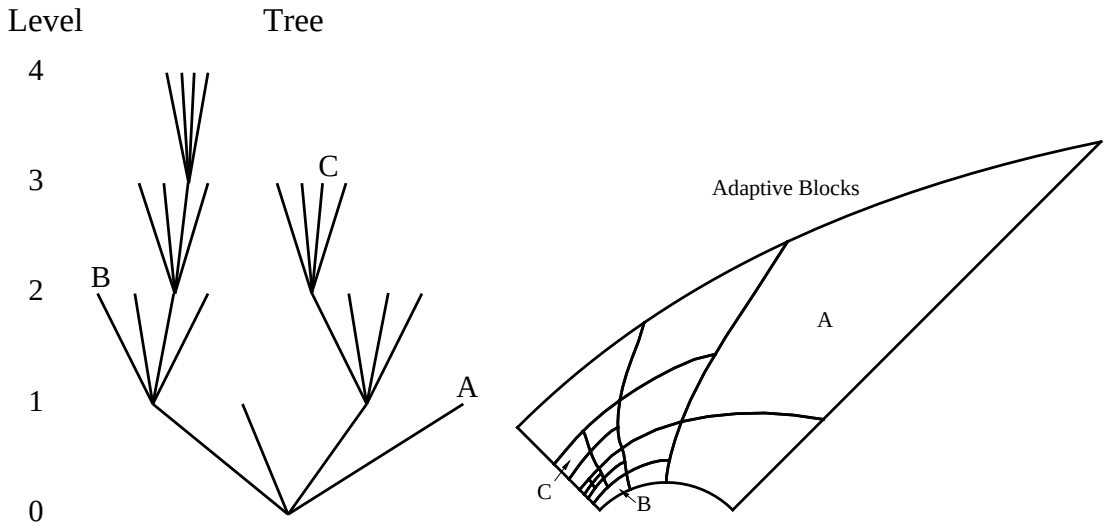


Figure 3.5: Adaptive mesh refinement quadtree data structure and associated solution blocks for a quadrilateral body fitted mesh.

algorithm applied on a section of a vertical-axis wind turbine blade of the type of interest herein.

3.9 Parallel Implementation

The multi-block AMR technique described in the previous section lends itself naturally to domain decomposition and parallelization by distributing solution blocks across multiple processors [49, 21]. Perfect load-balancing is achieved by distributing the same number of blocks to each processor. Solution information is shared between blocks via two layers of overlapping "ghost cells" associated with each block. The ghost cells store solution information for the current time level from neighbouring blocks so that each block can be computed on independently. Inter-processor communication is necessary to exchange solution information between blocks residing on different processors. In this work, the parallel AMR finite volume scheme is implemented using the C++ programming language and Message Passing Interface (MPI) library for inter-processor communication.

The algorithm described has been successfully carried out on the SciNet Consortium General Purpose Cluster (GPC) at the University of Toronto. The GPC consists of 3,780 IBM i-dataplex nodes connected by a non-blocking 4x-DDR Infiniband network. Each node contains two 2.53 GHz quad-core Intel Xeon 5,500 (Nehalem) processors with 16 GB of RAM, for a total of 30,240 cores.

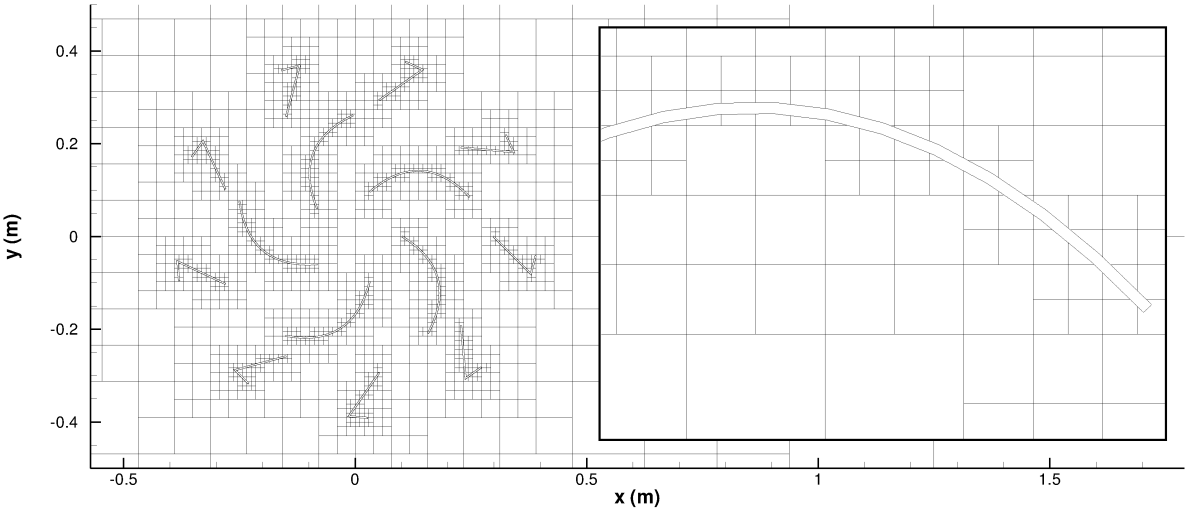


Figure 3.6: A multi-block mesh generated by the AMR algorithm for the Zephyr vertical-axis wind turbine. Solution blocks clustered around the embedded boundaries. The inset is a close-up of the one of the blades.

3.10 Extension to the Navier-Stokes Equations

The finite volume scheme described in the preceding sections can also be applied to the Favre-Averaged Navier-Stokes (FANS) equations. The integral form of the FANS equations is given by

$$\frac{d}{dt}\int_{A(t)} \mathbf{U} dA + \oint_{\partial A(t)} [\mathbf{F}(\mathbf{U}) - \vec{w}\mathbf{U}] \cdot \hat{n} dl = \oint_{\partial A(t)} \mathbf{G}(\mathbf{U}, \vec{\nabla}\mathbf{U}) \cdot \hat{n} dl + \int_{A(t)} \mathbf{S}(\mathbf{U}) dA \qquad (3.44)$$

where $\mathbf{U}$ is again the vector of conserved quantities,

$$\mathbf{U} = \left[\ \rho, \ \rho u, \ \rho v, \ \rho E, \ \rho k, \ \rho \omega \ \right]^T \ , \qquad (3.45)$$

$\mathbf{F}(\mathbf{U}) = [\mathbf{F_x}, \mathbf{F_y}]$ is the inviscid flux dyad with components

$$\mathbf{F_x} = \begin{bmatrix} \rho u \\ \rho u^2 + p + \frac{2}{3}\rho k \\ \rho uv \\ \rho u \left(E + \frac{p}{\rho} + \frac{2}{3}k \right) \\ \rho ku \\ \rho \omega u \end{bmatrix}, \quad \mathbf{F_y} = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p + \frac{2}{3}\rho k \\ \rho v \left(E + \frac{p}{\rho} + \frac{2}{3}k \right) \\ \rho kv \\ \rho \omega v \end{bmatrix}, \qquad (3.46)$$

G is the viscous flux dyad with components

$$\mathbf{G_x} = \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ u\tau_{xx} + v\tau_{xy} - q_x + (\mu + \sigma_k\mu_T)\frac{\partial k}{\partial x} \\ (\mu + \sigma_k\mu_T)\frac{\partial k}{\partial x} \\ (\mu + \sigma_\omega\mu_T)\frac{\partial \omega}{\partial x} \end{bmatrix}, \quad \mathbf{G_y} = \begin{bmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ u\tau_{xy} + v\tau_{yy} - q_y + (\mu + \sigma_k\mu_T)\frac{\partial k}{\partial y} \\ (\mu + \sigma_k\mu_T)\frac{\partial k}{\partial y} \\ (\mu + \sigma_\omega\mu_T)\frac{\partial \omega}{\partial y} \end{bmatrix} \tag{3.47}$$

and **S** is the vector of source terms associated with the turbulence modelling,

$$\mathbf{S} = \begin{bmatrix} 0,\ 0,\ 0,\ 0,\ \vec{\vec{\lambda}} : \vec{\nabla} \cdot \tilde{\vec{u}} - \beta^* \rho k \omega,\ \alpha\frac{\omega}{k}\vec{\vec{\lambda}} : \vec{\nabla} \cdot \tilde{\vec{u}} - \beta\rho\omega^2 \end{bmatrix}^T . \tag{3.48}$$

The two additional conserved variables ρk and $\rho\omega$ come from the Favre-averaging and turbulence modelling procedure. The result is a 6-component system of equations that must be solved at each cell (i, j).

Following the procedure outlined in Sections 3.2 and 3.3, the preconditioned semi-discrete form of the equations is

$$\Gamma_{ij}\frac{d\overline{\mathbf{U}}_{ij}}{dt} = -\frac{1}{A_{ij}}\sum_k (\mathbf{F}_k - \vec{w}_k\mathbf{U}_k - \mathbf{G}_k) \cdot \hat{n}_{ij_k}\Delta l_{ij_k} + \overline{\mathbf{S}}(\overline{\mathbf{U}}) - \frac{\overline{\mathbf{U}}_{ij}}{A_{ij}}\left(\sum_k \vec{w}_k \cdot \hat{n}_k \Delta l_{ij_k}\right) \tag{3.49}$$

where $\mathbf{\Gamma}$ is a 6×6 local preconditioner matrix. The requirement for $\mathbf{\Gamma}$ is that it must account for the coupling between the equations when scaling the convective wave speeds. In the case where k and ω are zero, $\mathbf{\Gamma}$ should revert to the preconditioner matrix for the Euler equations with ones along the diagonal corresponding to k and ω, and zeros elsewhere. Derivation of an appropriate preconditioning matrix for the FANS equations and the resulting preconditioned eigenstructure is given in Appendices B and C.

Chapter 4

Results

4.1 Overview

Various inviscid flow problems were studied as part of this research to explore the usability and efficiency of the proposed algorithm. In particular, the effect of using low-Mach-number preconditioning with dual time-stepping for aerodynamic analysis of vertical-axis wind turbines was examined with an emphasis on performance and solution accuracy. The results of these studies are presented and discussed. After demonstrating the effectiveness of this combined approach for low-speed flows with embedded boundaries, the proposed algorithm is applied to the prediction of low-speed inviscid flow for an impulsively started, translating ellipse. Inviscid flow over a stationary turbine blade of the type used in the vertical-axis wind turbine under investigation is also presented and discussed.

Finally, results are presented for the intended application: flow through a vertical-axis wind turbine. Here, the turbine configuration consists of several embedded boundaries: a set of inner blades form the rotor, and a set of outer vanes form the stator. Solutions are shown for a stationary vertical-axis wind turbine in low-speed flow. Analysis and discussion are presented for the simulation of a dynamically moving turbine in which the rotor blades are rotating relative to the stationary stator vanes.

4.2 Low-Speed Inviscid Channel Flow with a Bump

A low-speed inviscid channel flow has been used to validate the application of the Weiss-Smith low-Mach-number preconditioner and mesh adjustment scheme. The test case consists of a

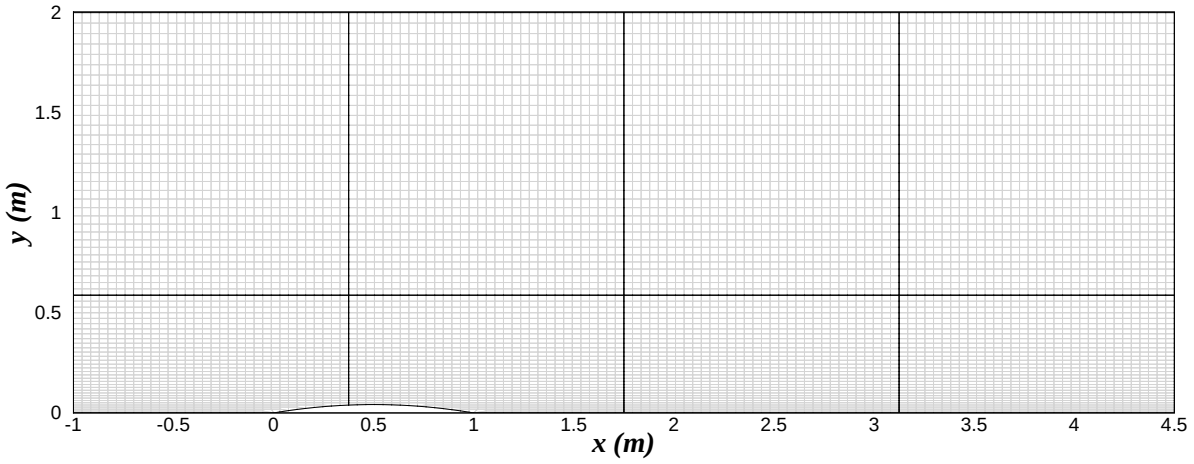


Figure 4.1: 128 × 64 computational mesh for the inviscid bump channel flow problem.

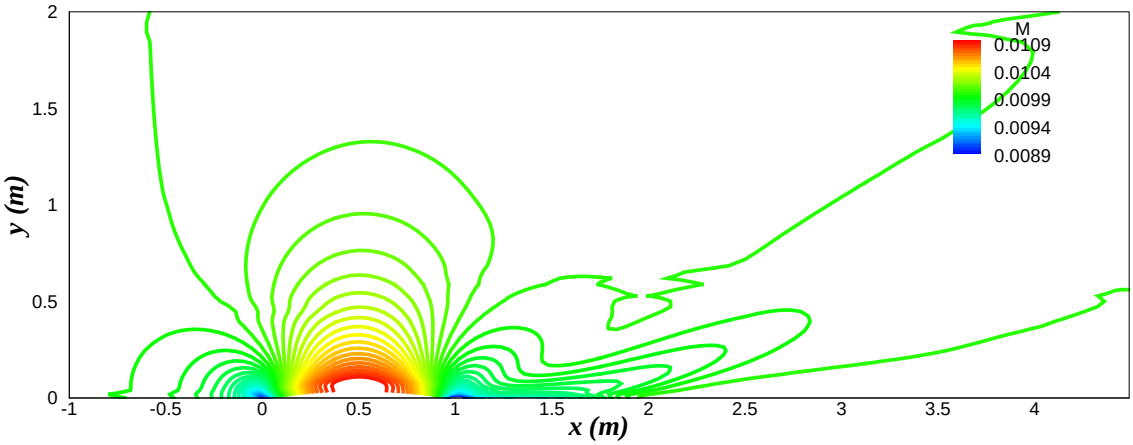
planar inviscid flow in a 4.5 m × 2.0 m channel with a non-smooth bump as described by Lynn [50]. The bump is constructed using a circular interface of radius 2.997190476 m; this results in a bump within the channel that is 1 m in length along the lower boundary. Mesh nodes near the interface are adjusted according to the embedded mesh algorithm described in Section 3.7 and cells within the interface are tagged as inactive. The mesh resolution is 128 cells × 64 cells, distributed equally over 8 blocks. Mesh stretching is used to cluster cells in the domain near the bump. The computational mesh is shown in Figure 4.1.

The boundaries conditions applied at the inlet are Dirichlet-type fixed free stream flow conditions. At the outlet, Neumann-type boundary conditions are applied. The top and bottom channel walls are reflective boundaries. The initial condition for the problem is a uniform flow from left to right at a Mach number of 0.01, which is well within the low-Mach-number regime.

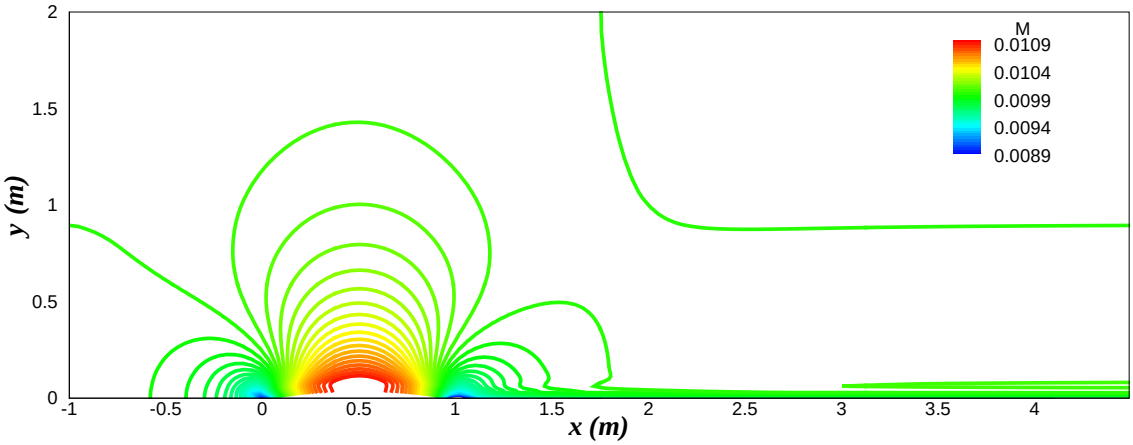
4.2.1 Steady Flow

Steady-state predicted solutions for the low-speed channel flow have been obtained by solving the Euler equations with and without the use of low Mach number preconditioning, as shown in Figures 4.2(a) and 4.2(b), respectively. In both solutions, the AUSM$^+$-up flux function is used with the Venkatakrishnan slope limiter. The solution is driven to steady state using an explicit 5-stage optimally-smoothing time-marching method with scalar local time-stepping. The reference Mach number is set at 0.01 and the CFL number is 0.20 for both cases.

Comparing the results of Figure 4.2(a) and 4.2(b), the improvement in accuracy due to precon-



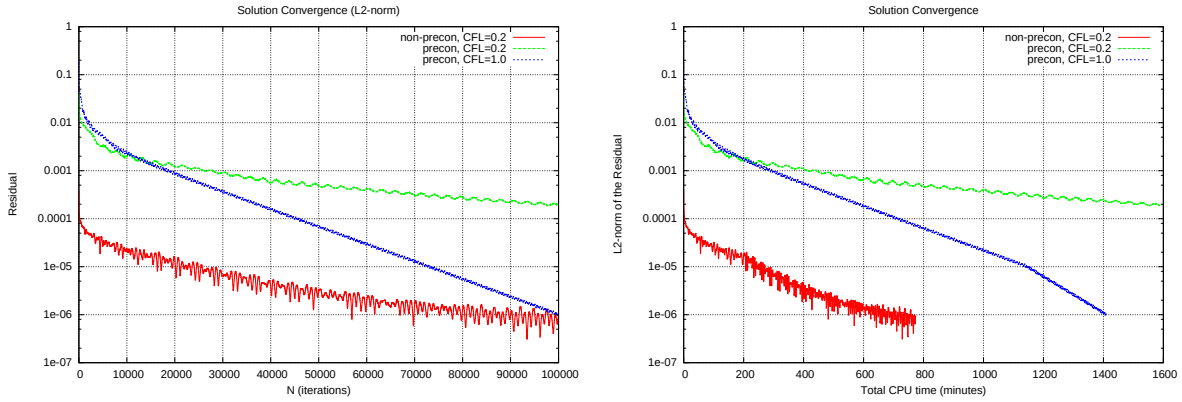
(a) Without preconditioning.



(b) With preconditioning.

Figure 4.2: Predicted Mach contours for the inviscid bump channel flow problem.

ditioning can be clearly seen. The non-preconditioned solution shows distorted Mach contours while the preconditioned solution shows the expected symmetric Mach contours across the bump. The convergence histories for the solutions as a function of iteration count and CPU time are given in Figures 4.3(a) and 4.3(b). When comparing the convergence histories of the preconditioned and non-preconditioned results at a CFL number of 0.2, it is clear that using the preconditioner requires greater computational effort per time step. For the same number of iterations, the preconditioned solution requires nearly twice as much CPU time. Hence, there is a clear trade-off between convergence rate and computational cost. For the unpreconditioned scheme to obtain the same level of accuracy as the preconditioned solution, a smaller time step would be required to resolve the convective waves. This can be achieved by increasing the mesh resolution or decreasing the CFL number.



(a) Convergence history by number of iterations

(b) Convergence history by CPU time

Figure 4.3: Convergence histories for the inviscid bump channel flow problem.

The additional computational cost in using the preconditioner can be offset by choosing a larger CFL number (time step). Without preconditioning, the maximum stable CFL number for this problem was 0.2. Values of the CFL number larger than this value resulted in unphysical solutions. With the preconditioning applied, however, larger CFL numbers were permitted. For a CFL of 1.0, the residual is reduced by nearly 5 orders of magnitude over 100,000 iterations. As seen in Figure 4.3(a), this rate of convergence is approximately twice that of the non-preconditioned case. Hence, the larger stability region afforded by the preconditioner effectively compensates for the increase in computational cost per iteration and justifies the use of the low-Mach-number preconditioner for low-speed flows.

4.2.2 Unsteady Flow

The bump flow problem presented previously has also been studied as an unsteady problem in order to assess the capabilities of the proposed solution method for unsteady flows. To obtain time-accurate solutions, the dual time-stepping algorithm is applied. In the following studies, we explore how CFL number and multigrid parameters affect the computational cost and the time accuracy of the simulation. Where applicable, comparisons to explicit time-marching are made to determine the effectiveness of the dual time-stepping method.

Multigrid Parameters and Number of Multigrid Cycles

In the dual time-stepping algorithm, the pseudo-time residual is reduced using an explicit optimally-smoothing time-marching scheme with multigrid acceleration. A number of parame-

ters can be set to control the multigrid performance, including the number of multigrid levels, the number of stages in the smoothing scheme, the number of iterations on the finest level, the number of pre-restriction and post-prolongation iterations, and the number of iterations on the coarsest level. The effect of choosing various multigrid parameters is evaluated by measuring the decrease in the residual per CPU-minute over 25 pseudo-time iterations.

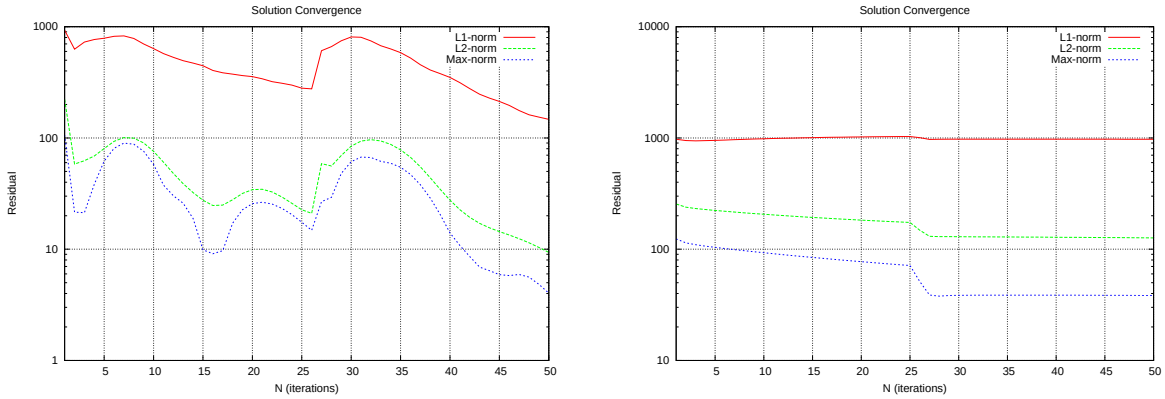
Two test cases were investigated here: one at a physical-time CFL number of 100 and one at a physical-time CFL number of 0.75. The psuedo-time CFL number was 0.75 for both studies. It was found that for a CFL number of 100, the best results were obtained when using 4 levels of multigrid with a 4-stage optimally-smoothing scheme in which the multigrid parameters were: 1 iteration on the finest level, 1 pre-restriction iteration, 0 post-prolongation iterations, and 5 iterations on the coarsest level (1-1-0-5). For a CFL number of 0.75, the best results were obtained when using 4 levels of multigrid with a 4-stage optimally-smoothing scheme in which the multigrid parameters were 1-0-1-7.

The number of inner multigrid cycles performed per physical time-step can also be adjusted. For large physical-time CFL numbers, the residual is readily decreased by a few cycles of multigrid. Figure 4.4(a) shows the residual history for the unsteady bump channel flow at a physical-time CFL number of 100. In this case, 25 iterations of multigrid are performed within each physical time-step. For small physical-time CFL numbers, the rate of decrease in the residual is not as significant. Figure 4.4(b) shows the residual history for the same case with a physical-time CFL number of 0.75.

The slower rate of convergence in the inner iterations for small physical-time CFL numbers is expected due to the fact that for small time steps, the solution content does not change significantly in time. Hence, the physical-time derivative, $\frac{d\overline{\mathbf{U}}}{dt}$, is small and the modified residual, $\mathbf{R}^*(\mathbf{U})$, differs from the previous time level only by a small amount. Therefore, because the residual is already well converged when the physical-time CFL number is small, fewer multigrid iterations are necessary.

Effect of CFL Number on Solution Accuracy

The physical-time CFL number determines the size of the time step that is taken for each iteration of the outer loop of the dual time-stepping method. In conjunction with an unconditionally stable scheme in the outer loop, a large CFL number can be chosen, enabling large time steps to be taken. Large time steps are advantageous because they allow the solver to "skip over" time steps that are much smaller than the time scales of interest for the flow. This reduces the



(a) CFL number of 100

(b) CFL number of 0.75

Figure 4.4: Convergence history for the unsteady bump flow problem at two CFL numbers. The iteration count shown is the number of multigrid cycles performed (pseudo-time iterations).

total computational cost of time-accurate simulations. However, care should be exercised in selecting the time step so as not to become too large such that the errors associated with the time-marching scheme result in inaccurate predictions.

In the bump flow problem, three points in the flow are tracked as a function of time. The points of interest are chosen to be the front, top, and rear of the bump where the flow is expected to change over time before reaching steady state. The time evolution of the Mach number at these points is plotted for various values of CFL number in Figures 4.5, 4.6, and 4.7. In all four cases, 50 inner pseudo-time iterations are performed with 4 levels of multigrid, using a 4-stage optimally-smoothing time-marching scheme. The multigrid parameters were 1-1-0-5. It can be seen that for small CFL numbers, high-frequency oscillations in the flow leading up to steady state are captured while for large CFL numbers, only the general time-evolution characteristics are captured.

Effect of CFL Number on Computational Cost

The dual time-stepping algorithm is advantageous when it is possible to take large time steps. However, when finer temporal resolution is required, or when an explicit moving boundary is present, the CFL number cannot be large and the advantage of the method is reduced. The following study compares the computational costs of performing DTS with that of explicit time-marching.

The computational cost of performing a single iteration of explicit time-marching is less than

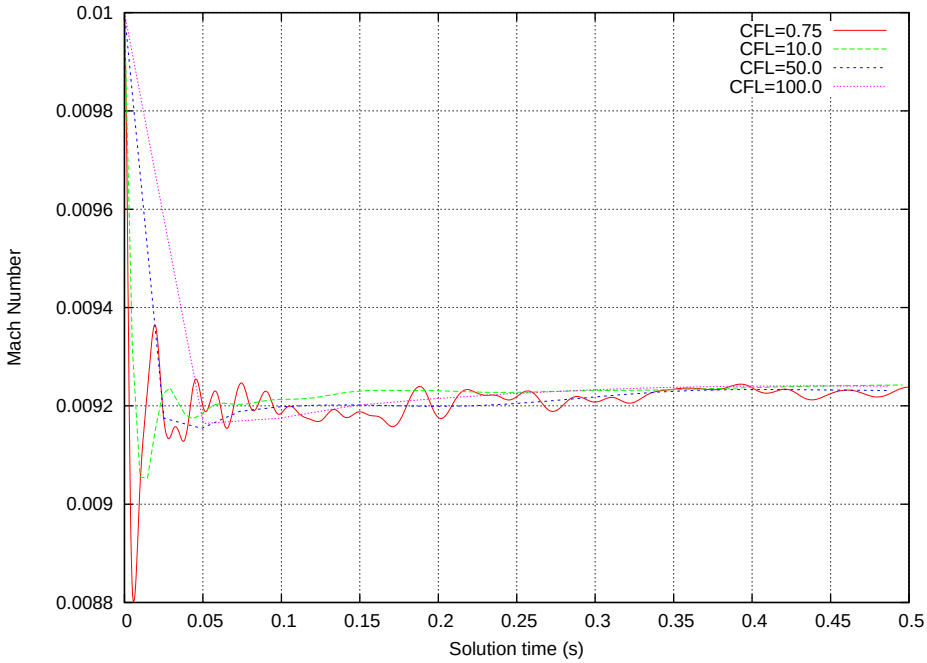


Figure 4.5: Transient solution at the front of the bump for various CFL numbers.

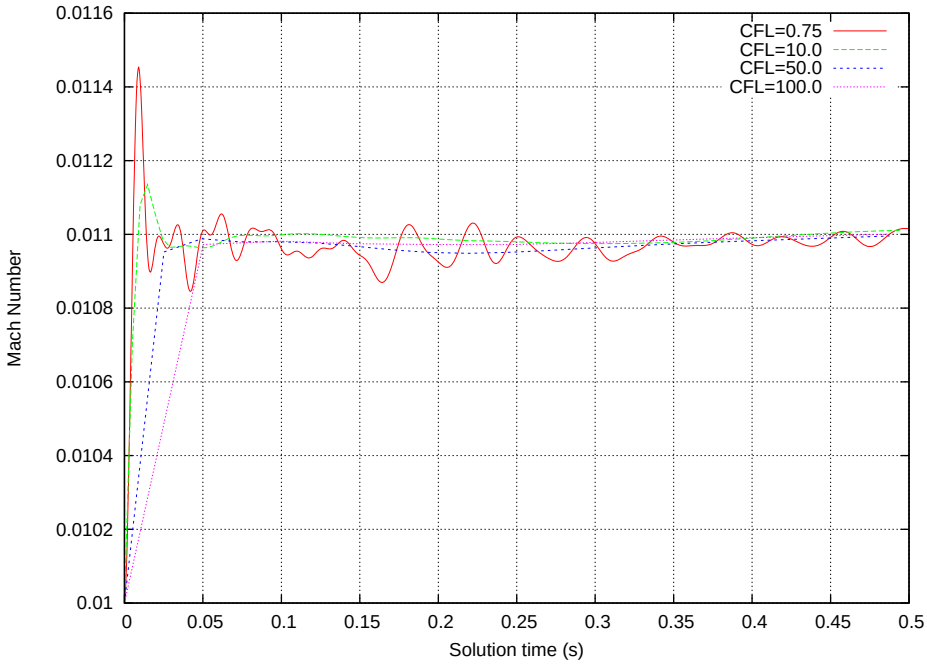


Figure 4.6: Transient solution at the top of the bump for various CFL numbers.

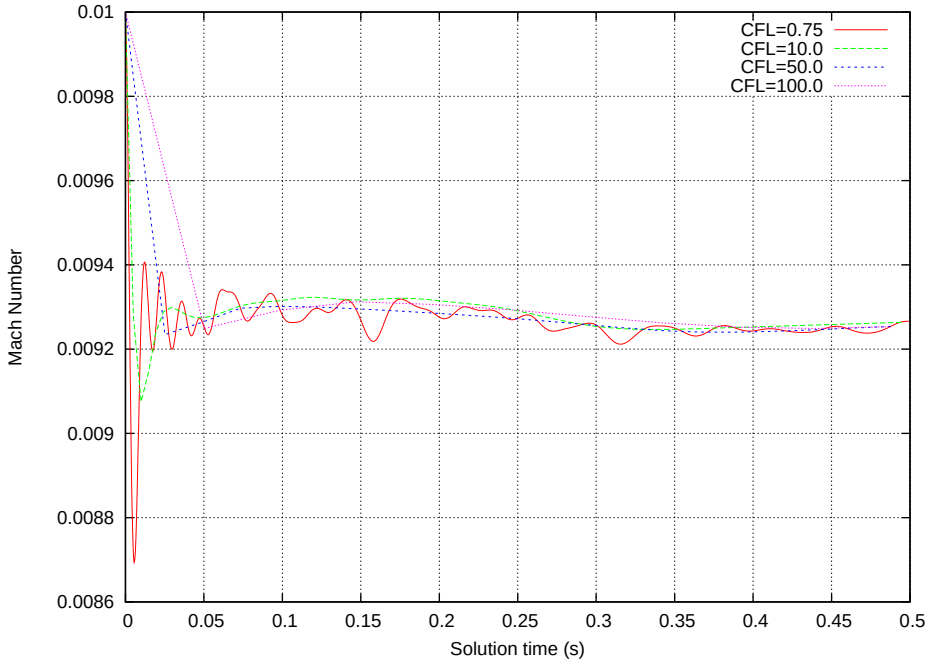


Figure 4.7: Transient solution at the rear of the bump for various CFL numbers.

the cost of performing a single physical-time iteration with dual time-stepping. In explicit time-marching, the preconditioning matrix does not need to be computed and multigrid iterations are not performed. However, without preconditioning, the time step is based on the non-preconditioned acoustic wave speed, $u + a$. Here, $u + a$ is much larger than the preconditioned wave speed, which is on the order of the convective wave speed, u. Hence, preconditioning enables larger time steps; in comparison, without preconditioning, more iterations are needed as the time steps are much smaller. This study answers the question of whether it is better to do many inexpensive iterations (explicit time-marching) or relatively few expensive iterations (dual time-stepping).

This trade-off is explored using the same time-accurate bump flow problem as described in Section 4.2. The computational costs of simulating this flow are compared for various explicit schemes and dual time-stepping for several CFL numbers. Table 4.1 lists the maximum allowable CFL number limited by stability and the corresponding computation time for the Runge-Kutta family of explicit time-marching schemes. The results show a reduced stability range for RK4, which is unexpected behaviour as this scheme has a larger stability region in the complex plane than RK1 and RK2. Table 4.2 lists the computation time for various values of physical-time CFL number obtained with dual time-stepping. In the results shown, four levels of multigrid with a explicit 4-stage optimally-smoothing time-marching scheme for the inner

Time-marching Method	CFL number	CPU time (CPU-mins)
RK1	0.10	605.59
RK1	0.15	635.45
RK2	0.10	891.09
RK2	0.15	587.64
RK4	unstable for CFL $\geq$ 0.01	–

Table 4.1: Computational costs for various explicit time-marching schemes applied on the time-accurate bump flow problem. The CFL numbers shown are based on the maximum allowable time step limited by stability.

Physical-time CFL number	Δt (ms)	CPU time (CPU-mins)
0.50	0.25118	199.69
0.75	0.376771	133.53
10.0	5.02362	10.108
50.0	25.1181	1.9877
100.0	50.2362	1.0325

Table 4.2: Computational cost for dual time-stepping using various physical-time CFL numbers.

loop. The CFL number for the inner loop is 0.75 for all test cases. For each physical time-step, twenty-five multigrid cycles of 1-1-0-5 are performed. In all cases, the bump flow problem is simulated up to a time of 500 ms using a single 8-core node on the University of Toronto's General Purpose Cluster supercomputer at the SciNet HPC Consortium. Each core is an Intel 2.5-GHz Nehalem processor.

It is clear from the results depicted in the two tables that performing the additional computational work per iteration in dual time-stepping is worth the cost of overcoming the severe time-step restriction found in the explicit time-marching schemes. With a modest CFL number of 0.75, using the dual-time stepping method results in computational savings of nearly 350% when compared with the least-expensive explicit time-marching scheme. With a larger CFL number of 100, the computational cost of the two methods differ by two orders of magnitude.

It is important to note that these results are for the inviscid system of equations only. For viscous flows, the time-step varies as Δx^2 instead of Δx, where Δx is the cell spacing. This will significantly increase the computational cost for explicit time-marching methods. The advantage of using dual time-stepping will be even greater when solving the full Navier-Stokes

equations.

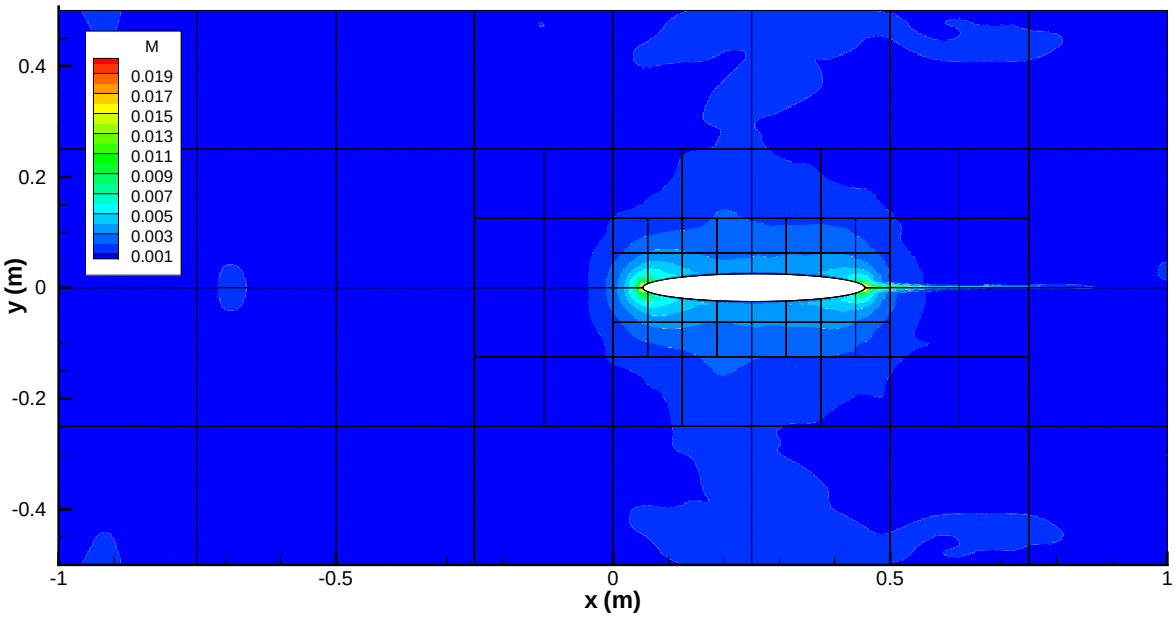
4.3 Translating Ellipse

To further test the proposed solution method, the dual time-stepping algorithm is applied to a moving embedded boundary. An ellipse is impulsively started in quiescent air and moves linearly in the negative x-direction at a speed of 10 m/s. This corresponds to a Mach number of 0.03 at the embedded boundary. Reflection boundary conditions are applied all on four domain boundaries. As in the previous case, pseudo-time iterations were achieved using 4 levels of multigrid with a 4-stage optimally-smoothing explicit scheme at a CFL number of 0.75. The multigrid parameters were 1-1-0-5, with 10 pseudo-time iterations performed for each physical time step. Since motion of the ellipse is treated explicitly, the physical-time CFL number is restricted to less than 1. For the results shown here, temporal discretization in physical time was achieved using second-order backwards differencing at a physical-time CFL number of 0.75. After initial refinement of blocks intersected by the interface, the mesh consisted of 74 blocks and 303,104 cells over 3 levels of refinement. Periodic adaptive mesh refinements were performed every 10 time steps to track and refine blocks intersected by the moving interface.

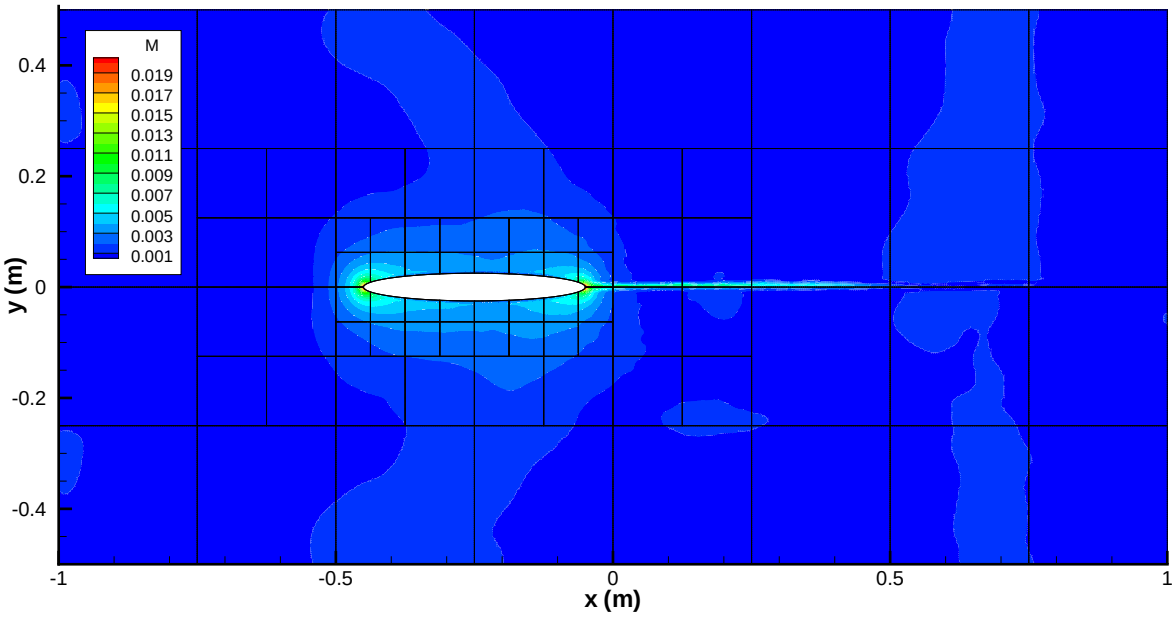
The computed Mach contours are shown in Figure 4.8 at two different times during the simulation. As expected, the motion of the interface induced localized regions of fluid movement above and below the ellipse. The periodic AMR algorithm successfully refined blocks intersected by the interface and ahead of the path of motion of the interface. When the ellipse is translated a sufficient distance away and no longer intersect the refined blocks, the AMR algorithm coarsens the blocks to maintain a high refinement efficiency.

4.4 Flow Over a Turbine Blade

A stationary turbine blade of the type that may be found in use on typical vertical-axis wind turbines is now simulated using the dual time-stepping algorithm. The blade is a circular arc with a material thickness of 0.125 inches. The mesh adjustment scheme is used to locally adjust nodes to lie directly on the interface. For the adjustment scheme to work properly, a sufficient number of cells must be internal to the interface to provide boundary conditions at the interface. To achieve this, the mesh blocks intersected by the interface are progressively refined until the desired resolution is reached. The ability to cluster cells near the embedded boundary shows that block-based mesh refinement is an effective complement to the mesh adjustment scheme.



(a)



(b)

Figure 4.8: An ellipse translating from right to left at 10/ms at (a) 50 ms and (b) at 100 ms. Mach contours and mesh block boundaries are shown.

Figure 4.9 shows a single blade in a uniform flow of Mach 0.01. A physical-time CFL number of 100.0 and a pseudo-time CFL number of 0.75 is used. Four levels of multigrid were conducted with a 1-1-0-5 smoothing arrangement. Fifty multigrid cycles were done per physical-time step. Blocks intersected by the interface were successively refined to produce 325 blocks across 7 refinement levels.

The computed solution reveals an attached flow over the turbine blade with an unsteady recirculation region below the blade. Minor vortex shedding is also observed at the trailing edge of the blade. The solution convergence plot presented in Figure 4.10 indicate a sharp initial decrease in the residual which is to be expected as the flow initially develops around the embedded boundary. Once the major flow features are established, the convergence rate is reduced, indicating that a pseudo-steady state has been achieved.

4.5 Vertical-Axis Wind Turbine Flow

Predictions for a complete vertical-axis wind turbine are now presented to demonstrate the ability of the solver to simulate low-speed flows with multiple embedded boundaries. The Zephyr Vertical-Axis Wind Turbine consists of stationary stator vanes arranged around a set of rotating rotor blades of the type studied in Section 4.4. Designed to increase the overall efficiency of the turbine, the stator vanes are shaped and positioned such that air approaching on the reciprocating side of the turbine will still contribute to a positive torque. Figure 4.11 shows a schematic of the cross-section of the vertical-axis wind turbine investigated herein. The presence of multiple stationary and dynamic embedded boundaries makes this flow case an ideal candidate for the mesh adjustment scheme. The flow regime in which this turbine will be operating is between 6 mph and 30 mph (about 3-13 m/s), which is in the low-speed, incompressible regime. The dual time-stepping algorithm with preconditioning will enable computationally feasible solutions.

4.5.1 Stationary Turbine

A stationary wind turbine with five rotor blades and nine stator vanes is placed in freestream Mach number M=0.0131 flow. The freestream flow velocity corresponds to about 10 mph, which is representative of typical wind speeds targeted by this turbine design. The computational mesh used for this simulation consisted of 16 root blocks with each block containing 8 cells by 8 cells. Blocks intersected by an interface were successively refined up to 10 levels of refinement in order to increase mesh resolution near the blades and vanes, resulting in 10,261 blocks

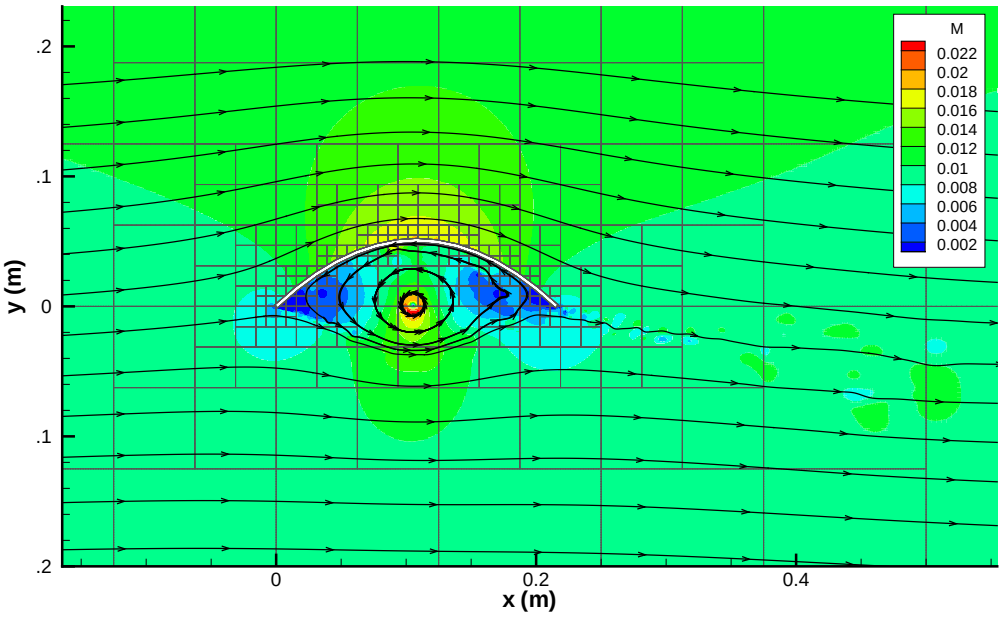


Figure 4.9: A single turbine blade in Mach 0.01 flow at 333.559 ms (325 blocks, 1,331,200 cells, $\eta = 0.995$). Mach contours, block boundaries, and streamlines are shown.

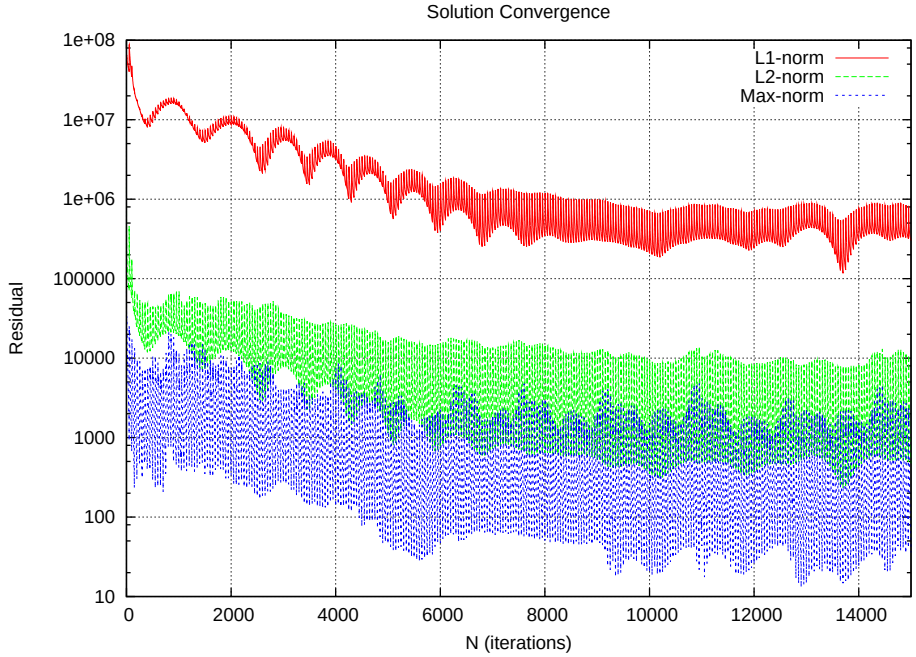
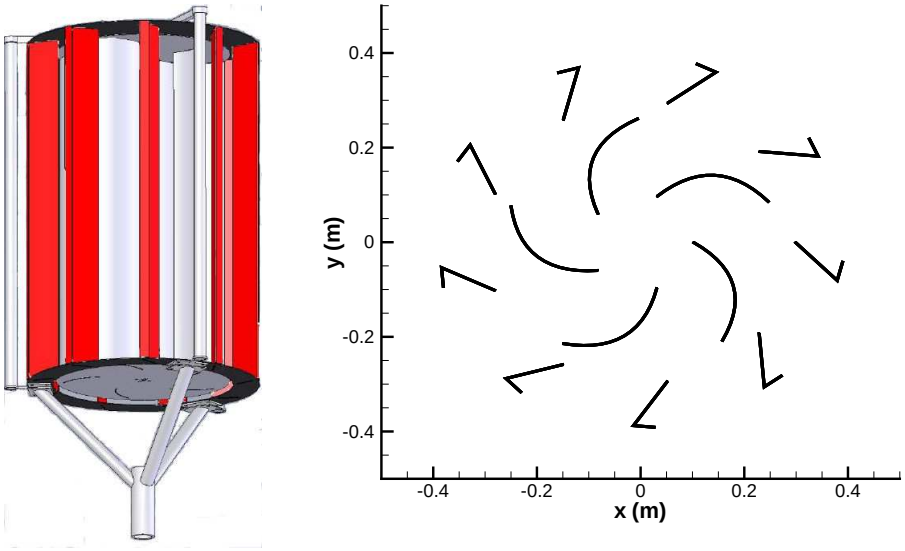


Figure 4.10: Convergence rate for flow over a single turbine blade.



(a) (b)

Figure 4.11: A 3D perspective model (a) and schematic of the cross-section (b), of the Zephyr vertical-axis wind turbine.

and 656,704 mesh points. Due to the high mesh resolution requirements imposed by the thin interfaces, multigrid was not used for the pseudo-time iterations. Instead, fifty iterations were performed on the original mesh and coarse mesh levels were not generated in order to reduce computational cost. A 4-stage optimally-smoothing multi-stage explicit scheme was used for the inner iterations at a CFL number of 0.75. The physical-time CFL number was 1,500, which corresponds to about 5 ms of solution time per physical time-step. The results were obtained with about 8 hours (wall-time) on 96 processors, corresponding to 768 CPU-hours of computation.

The computed solution is shown in Figure 4.12. Complex flow interactions within the turbine are revealed, with recirculation regions and eddies forming behind the blades and vanes. A large unsteady, strongly detached wake is generated behind the turbine assembly. A close-up of the flow around a stator vane and rotor blade is presented in Figure 4.13. Two recirculation regions can be seen forming in the regions behind the leading and trailing edges of the stator vane. An unsteady stream of vortices can be seen stemming from the leading edge of the rotor blade as well. The interaction of flow structures created by the embedded boundaries creates strongly detached flows and unsteady flow phenomena.

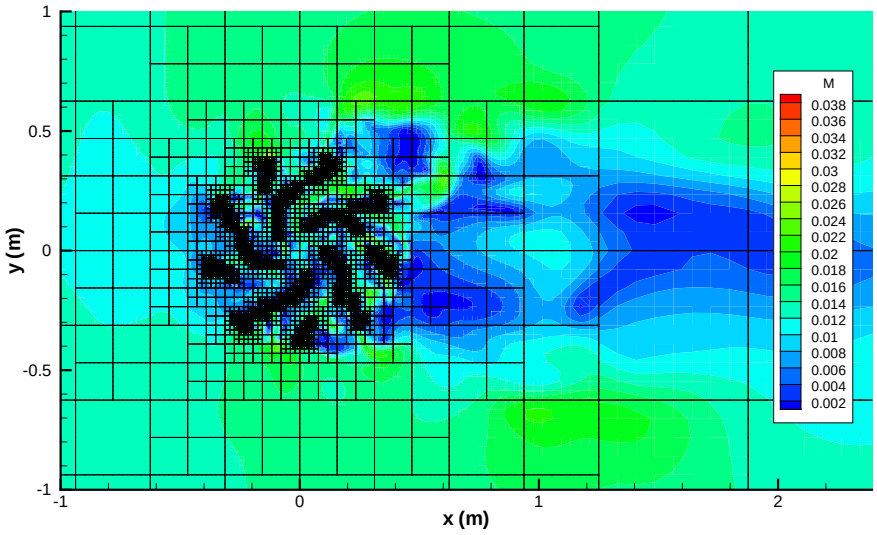


Figure 4.12: A stationary vertical-axis wind turbine in Mach 0.0131 flow, corresponding to a windspeed of 10 mph, at t=1 ms. The mesh consists of 10,261 blocks and 656,704 cells across 10 levels of refinement. Mach contours and block boundaries are shown.

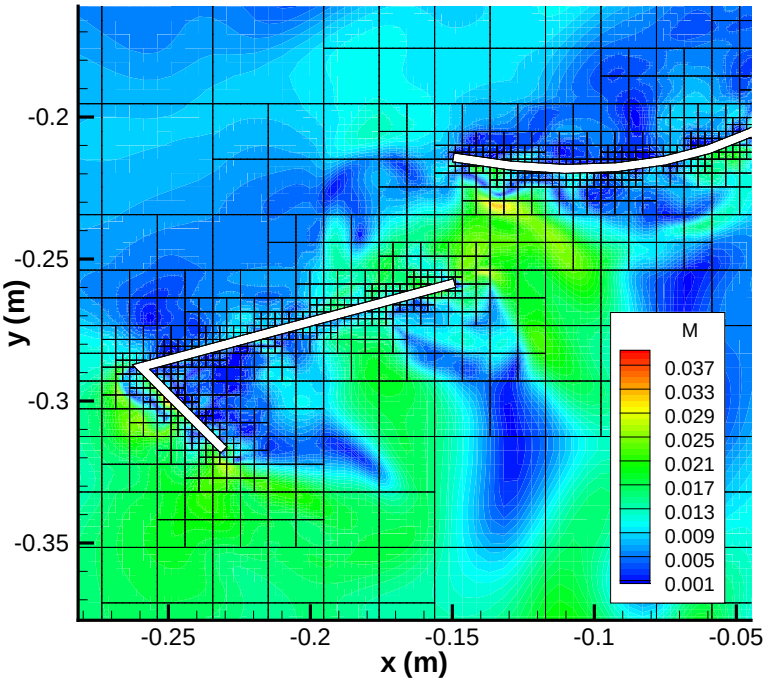


Figure 4.13: A close-up of the flow around a stator vane and rotor blade.

4.5.2 Rotating Turbine

A rotating wind turbine simulation is much more computational expensive than the stationary case. Since the motion of the embedded boundaries is treated explicitly – that is, the interface cannot move more than one cell length per time-step – the physical-time CFL number is constrained to be less than one. This significantly increases the number of iterations required.

To illustrate the increase in computational cost, a rough estimate can be made by extrapolating from the computational cost of the stationary turbine simulation presented in the previous section. When the stationary turbine simulation is run with a physical-time CFL of 0.75, which is 2,000 times smaller than a CFL of 1,500, it is expected that the size of the time-step will be proportionately reduced and the amount of computation time will be proportionately increased. Hence, the computational cost for the rotating case will be 2,000 greater, on the order of 1.5 million CPU-hours. However, this is not a reasonable estimate since fewer pseudo-time iterations are necessary when the size of the physical time-step is small. It may be more reasonable to use 5 iterations per physical time-step, resulting in a 10-fold decrease in computational cost. This results in a cost of 0.15 million CPU-hours to simulate 1 millisecond of solution time. This estimate is extremely conservative: the computational cost of performing the mesh adjustment at each physical time-step and the cost of performing periodic AMR has been neglected since these operations were not done in the stationary case.

For a wind turbine with a radius of 0.25 m, rotating at a TSR of 0.5 in a freestream Mach number flow of M= 0.0131 flow, the angular frequency can be calculated using Eq. (1.5). Therefore,

$$\omega = \frac{V}{\lambda r} = \frac{4.5 \text{ m/s}}{(0.5)(0.25 \text{ m})} = 36 \text{ rad/s}. \tag{4.1}$$

This corresponds to a period of about 175 ms. If a simulation of a full rotation of the turbine is desired, it would require more than 26 million CPU-hours. Clearly, this is computational unfeasible!

A factor that contributed to the prohibitive computational cost is the mesh resolution required to resolve the thin turbine blades and vanes. The mesh adjustment algorithm is unable to perform the adjustment when the interface is on the order of the cell size Δx or smaller. It was found that a minimum of 8 cells must be internal to the interface to ensure proper adjustment. The removal of this limitation will enable coarser meshes, and thus reduce computational cost and memory required. Another contributing factor to the computational cost is the treatment of moving embedded boundaries. The explicit treatment constrains the physical-time CFL number to less than one, effectively negating the benefit of using the dual time-stepping algorithm.

Chapter 5

Conclusions

5.1 Summary

Significant progress has been made towards a computational framework for predicting turbulent flow in vertical-axis wind turbines. A finite volume scheme has been proposed and devloped for solving the inviscid Euler equations on a two-dimensional body-fitted, multi-block, quadrilateral mesh. A mesh adjustment scheme is used to locally adjust nodes onto stationary and moving embedded boundaries. This has enabled complex geometries to be meshed in a straightforward manner. In addition, a parallel block-based adaptive mesh refinement (AMR) algorithm has been used to accurately resolve the wide range of length scales present in the flow while minimizing computational cost. The block-based approach lends itself naturally to domain decomposition, allowing efficient and scalable distribution of computational work on large clusters of distributed-memory multi-processor machines. The use of both AMR and the mesh adjustment scheme has been shown to be effective in resolving disparate fluid length scales and structural length scales, particularly the thin-walled structures of VAWTs studied in this thesis.

Low-speed flows are typical of the operating regime of wind turbines. To simulate such flows, low-Mach-number preconditioning has been applied to remove numerical stiffness and maintain solution accuracy. The preconditioner of Weiss and Smith has been applied for the Euler equations and has been shown to enable larger time steps. A dual time-stepping approach has also been used to enable even larger time steps in simulations of unsteady flows.

The proposed computational framework has been applied to a variety of test problems. An inviscid flow over a bump validated the flux algorithms and demonstrated the effectiveness of the low-Mach-number preconditioner in maintaining accuracy and reducing computational cost.

An impulsively started, translating ellipse was used to validate unsteady low-speed flow using dual time-stepping with moving embedded boundaries. Flow over a single turbine blade was also computed and the proposed algorithm was shown to be effective for the geometry of the target problem.

In addition, parameter studies were conducted on multigrid smoothing and number of multigrid cycles to determine the effect on solution accuracy and residual reduction. The effect of physical-time CFL number in the dual time-stepping scheme on computational cost and solution accuracy was also explored.

Finally, the proposed solution method was applied to the target problem: predicting flows through vertical-axis wind turbines. The mesh adjustment scheme, in conjunction with AMR, was shown to be particularly effective in resolving the thin-walled vanes and blades of the wind turbine. An unsteady simulation of a stationary five-bladed wind turbine with nine stator vanes was presented. It was shown that the current solution method becomes prohibitively expensive when applied to rotating wind turbines with dynamic embedded boundaries due to the explicit treatment of the interface motion and the significant mesh resolution requirements of this particular problem.

5.2 Future Research

The course of this project has identified several areas for future research. These recommendations are summarized below.

5.2.1 Improved Modelling

The accuracy of the simulations presented in this thesis can be improved by including the viscous terms and solving the Reynolds-Averaged Navier-Stokes equations. For inviscid flow, the only contributor to the torque experienced by the turbine is the pressure acting normal to the blades. The inclusion of the wall shear stress acting tangentially to the blades will result in a more accurate model of the fluid, which will result in a more accurate prediction of the power output.

Another area for improvement is in the area of turbulence modelling. It is well-known that the k-ω turbulence model and other turbulence models based on the Boussinesq approximation suffer from reduced accuracy for flows with high swirl, flows over curved surfaces, and flows with secondary motions [4]. Thus, greater accuracy can be obtained by choosing an alternative

approach to solving the Navier-Stokes equations. Large-eddy simulation (LES), an approach in which large energy-carrying eddies are resolved directly and small-scale eddies are modelled, holds significant promise in accurately predicting turbulent flows [51]. LES is the subject of much research, and is gaining traction among CFD practitioners.

5.2.2 Reduced Computational Cost

In the present dual time-stepping scheme, the pseudo-steady problem is solved using an explicit time-marching scheme with multigrid for convergence acceleration. This can be improved by considering parallel implicit methods, which have been shown to be effective for steady problems. The Newton-Krylov schemes proposed by Northrup and Groth is amenable to the finite-volume scheme presented in this work [52]. It may also be possible to overcome the prohibitive computational costs of simulating a rotating turbine by considering other meshing techniques that are not limited by the thickness of the turbine blades, such as unstructured and hybrid meshes. These techniques would reduce the mesh resolution requirement, and hence reduce the computational memory and time needed to compute a solution. It may also be possible to avoid adjusting the mesh altogether by considering a rotating frame of reference.

5.2.3 Extension to Three-Dimensional Flows

A natural extension to the work presented in this thesis is the implementation of the finite volume scheme in three dimensions. This would be beneficial on several fronts. Turbulence is a three-dimensional phenomena and as such, should be modelled in three dimensions. A three-dimensional framework would also enable the exploration of more complex wind turbine geometries. Recent variations of the vertical-axis wind turbine cannot be sufficiently modelled using two-dimensional approaches (Figure 5.1). A prerequisite to developing such a framework is the extension of the mesh adjustment scheme to three dimensions. The low-Mach-number preconditioner used in this study can be extended to three dimensions in a straightforward manner.

5.2.4 Computational Optimization

This thesis work contributes towards the overall development of a suite of computational tools that can be used to search for a more efficient wind turbine design. Design variables include the number of rotating blades, blade angle, length, and curvature, and as well as the number of stationary vanes, vane angle, length, and geometry of the patented “winglet”. This is not

Figure 5.1: A helical Savonius vertical-axis wind turbine.

an exhaustive list; there are many other possibilities including the use of airfoils. Naturally, the design space for this product is large, making computational optimization techniques an attractive tool. Optimization algorithms have been applied with great success on individual airfoils as well as complete aircraft over a range of operating conditions [53, 54]. It is worth exploring whether or not these techniques can be applied to the design of vertical-axis wind turbines.

References

[1] B. van Leer, C. H. Tai, and K. G. Powell. Design of optimally-smoothing multi-stage schemes for the Euler equations. Paper 89-1933-CP, AIAA, June 1989.

[2] E. Hau. *Wind turbines: fundamentals, technologies, application, and economics*. New York: Springer, 2nd edition, 2006.

[3] G.L. Johnson. *Wind Energy Systems*. 2nd edition, Oct 2006. URL: `http://eece.ksu.edu/~gjohnson/`.

[4] D. C. Wilcox. *Turbulence Modeling for CFD*. DCW Industries, La Cañada, California, 2nd edition, 1998.

[5] World wind energy report 2009. Technical report, World Wind Energy Association, Feb 2008.

[6] A. Betz. *Wind-Energie und ihre Ausnutzung durch Windmhlen*. Vandenhoek und Rupprecht, 1926.

[7] C. Hirsch. *Numerical Computation of Internal and External Flows, Volume 2, Computational Methods for Inviscid and Viscous Flows, Chapter 16*. John Wiley & Sons, Toronto, 1990.

[8] H. Lomax, T.H. Pulliam, and D.W. Zingg. *Fundamentals of Computational Fluid Dynamics*. New York: Springer, 2003.

[9] N. Mandas, F. Cambuli, and C.E. Carcangiu. Numerical prediction of horizontal axis wind turbine flow. In *Proceedings of the European Wind Energy Conference & Exhibition, Athens, Greece, February 27–March 2, 2006*. The European Wind Energy Association, 2006.

[10] O. Guerri1, A. Sakout, and K. Bouhadef. Simulations of the fluid flow around a rotating vertical axis wind turbine. *Wind Engineering*, 31(3):149–163, 2007.

[11] T. Sankar and Tiryakioğlu M. Design and power characterization of a novel vertical axis wind energy conversion system (vawecs). *Wind Engineering*, 32(6):559–572, 2008.

[12] A.E. Kasmi and C. Masson. An extended k-ϵ model for turbulent flow through horizontal-axis wind turbines. *Journal of Wind Engineering and Industrial Aerodynamics*, 96(1):103–122, 2008.

[13] R. Howell, N. Qin, J. Edwards, and N. Durrani. Wind tunnel and numerical study of a small vertical axis wind turbine. *Renewable Energy*, 35:412–422, Feb 2010.

[14] G. Xu and L.N. Sankar. Computational study of horizontal axis wind turbines. *Journal of Solar Energy Engineering*, 122:35–39, 2000.

[15] N. Sezer-Uzol and L.N. Long. 3-d time-accurate cfd simulations of wind turbine rotor flow fields. (2006-0394), Jan 2006.

[16] J. S. Sachdev. *Parallel Solution-Adaptive Method For Predicting Solid Propellant Rocket Motor Core Flows*. PhD thesis, University of Toronto Institute for Aerospace Studies, 2007.

[17] M.-S. Liou. A sequel to AUSM, part ii: AUSM$^+$-up for all speeds. *Journal of Computational Physics*, 214:137–170, 2006.

[18] J. M. Weiss and W. A. Smith. Preconditioning applied to variable and constant density flows. *AIAA Journal*, 33(11):2050–2057, 1995.

[19] A. Jameson. Time dependent calculations using multigrid, with applications to unsteady flows past airfoils and wings. Paper 1991–1596, AIAA, June 1991.

[20] S. K. Godunov. Finite-difference method for numerical computations of discontinuous solutions of the equations of fluid dynamics. *Matematicheskii Sbornik*, 47:271–306, 1959. Translated by I. Bohachevsky.

[21] J.S. Sachdev and C.P.T. Groth. A mesh adjustment scheme for embedded boundaries. *Communications in Computational Physics*, 2(6):1095–1124, 2007.

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

[23] Y.-H. Choi and C. L. Merkle. Time derivative preconditioning in viscous flows. Paper 91-1652, AIAA, June 1991.

[24] Y.-H. Choi and C. L. Merkle. The application of preconditioning in viscous flows. *Journal of Computational Physics*, 105:207–223, 1993.

[25] J. Housman, C. Kiris, and M. Hafez. Preconditioned methods for simulations of low speed compressible flows. *Computers & Fluids*, 38(7):1411–1423, 2008.

[26] I. Mary, P. Sagaut, and M. Deville. An algorithm for unsteady viscous flows at all speeds. *International Journal for Numerical Methods in Fluids*, 32:371–401, 2000.

[27] J. F. Lynn and B. van Leer. Multi-stage schemes for the Euler and Navier-Stokes equations with optimal smoothing. Paper 93-3355-CP, AIAA, July 1993.

[28] E. Li. A parallel multigrid method for predicting compressible flow in turbomachinery, masters thesis, university of toronto, 2004.

[29] L.D. Dailey and K.-H. Chen. On solving the compressible navier-stokes equations for unsteady flows at very low mach numbers. (93-3368), 1993.

[30] L.D. Dailey and R.H. Pletcher. Evaluation of multigrid acceleration for preconditioned time-accurate navier-stokes algorithms. *Computers & Fluids*, 25(8):791–811, 1996.

[31] J. J. Gottlieb and C. P. T. Groth. Assessment of Riemann solvers for unsteady one-dimensional inviscid flows of perfect gases. *Journal of Computational Physics*, 78:437–458, 1988.

[32] P. L. Roe. Characteristic-based schemes for the Euler equations. *Annual Review of Fluid Mechanics*, 18:337–365, 1986.

[33] A. Harten, P. D. Lax, and B. van Leer. On upstream differencing and Godunov-type schemes for hyperbolic conservation laws. *SIAM Review*, 25(1):35–61, 1983.

[34] B. Einfeldt. On Godunov-type methods for gas dynamics. *SIAM Journal on Numerical Analysis*, 25:294–318, 1988.

[35] T. Linde. A practical, general-purpose, two-state HLL Riemann solver for hyperbolic conservation laws. *International Journal for Numerical Methods in Fluids*, 40:391–402, 2002.

[36] M.-S. Liou and C. J. Steffen. A new flux splitting scheme. *Journal of Computational Physics*, 107:23–39, 1993.

[37] M.-S. Liou. A sequel to AUSM: AUSM$^+$. *Journal of Computational Physics*, 129:364–382, 1996.

[38] Y. Wada and M.-S. Liou. An accurate and robust flux splitting scheme for shock and contact discontinuities. *SIAM Journal on Scientific Computing*, 18(3):633–657, 1997.

[39] K.H. Kim, J.H. Lee, and O.H. Rho. An improvement of ausm schemes by introducing the pressure-based weight functions. *Computers & Fluids*, 27(3):311–346, 1998.

[40] V. Venkatakrishnan. On the accuracy of limiters and convergence to steady state solutions. Paper 93-0880, AIAA, January 1993.

[41] D. De Zeeuw and K. G. Powell. An adaptively refined Cartesian mesh solver for the Euler equations. *Journal of Computational Physics*, 104:56–68, 1993.

[42] M. J. Aftomis, M. J. Berger, and J. E. Melton. Robust and efficient Cartesian mesh generation for component-base geometry. *AIAA Journal*, 36(6):952–960, 1998.

[43] M. J. Aftomis, M. J. Berger, and G. Adomavicius. A parallel multilevel method for adaptively refined cartesian grids with embedded boundaries. Paper 2000-0808, AIAA, January 2000.

[44] M. C. Lai and C. S. Peskin. An immersed boundary method with formal second-order accuracy and reduced numerical viscosity. *Journal of Computational Physics*, 160:705–719, 2000.

[45] C. S. Peskin. The immersed boundary method. *Acta Numerica*, pages 1–39, 2002.

[46] M. J. Berger. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Computational Physics*, 53:484–512, 1984.

[47] C. P. T. Groth and S. A. Northrup. Parallel implicit adaptive mesh refinement scheme for body-fitted multi-block mesh. Paper 2005-5333, AIAA, June 2005. 17th AIAA Computational Fluid Dynamics Conference, 6-9 June 2005, Toronto, Canada.

[48] J. S. Sachdev, C. P. T. Groth, and J. J. Gottlieb. Parallel AMR scheme for turbulent multi-phase rocket motor core flows. Paper 2005-5334, AIAA, June 2005. 17th AIAA Computational Fluid Dynamics Conference, 6-9 June 2005, Toronto, Canada.

[49] X. Gao and C. P. T. Groth. A parallel adaptive mesh refinement algorithm for predicting turbulent non-premixed combusting flows. *International Journal of Computational Fluid Dynamics*, 20(5):349–357, 2006.

[50] J.F. Lynn. *Multigrid Solution of the Euler Equations with Local Preconditioning*. PhD thesis, University of Michigan, 1995.

[51] U. Piomelli. Large-eddy simulation: achievements and challenges. *Progress in Aerospace Sciences*, 35:335–362, 1999.

[52] S. Northrup and C.P.T. Groth. Parallel implicit AMR scheme for unsteady reactive flows. In *Proceedings of the 18th Annual Conference of the CFD Society of Canada, London, Ontario, Canada, May 18–19, 2010*, 2010.

[53] J. R. R. A. Martins, J. J. Alonso, and J. J. Reuther. High-fidelity aerostructural design optimization of a supersonic business jet. *Journal of Aircraft*, 41(3):523–530, 2004.

[54] M. Nemec and D. W. Zingg. Multipoint and multi-objective aerodynamic shape optimization. *AIAA Journal*, 42(6), 2004.

Appendix A

AUSM$^+$-up Scheme

The AUSM$^+$-up scheme is used to evaluate the inviscid flux at the cell interface. In this work, the scheme has been extended for the Favre-Averaged Navier-Stokes (FANS) equations. The scheme is able to deal with flows at all speed regimes, with special attention to accuracy and robustness at low Mach numbers. The specifics of the algorithm extensions for the FANS equations are presented in one dimension; the extension to multiple dimensions is straightforward using direction splitting. Refer to the paper by Liou [17] for a full derivation and discussion of the original AUSM$^+$-up scheme.

For the FANS formulation considered herein, the inviscid flux vector as presented in Chapter 3 can be explicitly split into a convective flux $\mathbf{F_c}$ and a pressure flux $\mathbf{P}$ such that

$$\mathbf{F_x} = \begin{bmatrix} \rho u \\ \rho u^2 + p + \frac{2}{3}\rho k \\ \rho uv \\ \rho u\left(E + \frac{p}{\rho} + \frac{2}{3}k\right) \\ \rho ku \\ \rho \omega u \end{bmatrix} = \mathbf{F_c} + \mathbf{P}. \tag{A.1}$$

The convective flux is further decomposed into a common scalar flux mass flux $\dot{m}$ and a vector

quantity $\vec{\psi}$ that is convected by $\dot{m}$. That is, $\mathbf{F_c} = \dot{m}\vec{\psi}$, where

$$\dot{m} = \rho u, \quad \vec{\psi} = \begin{bmatrix} 1 \\ uk \\ v \\ h + \frac{2}{3}k \\ k \\ \omega \end{bmatrix}, \quad \mathbf{P} = \begin{bmatrix} 0 \\ p + \frac{2}{3}\rho k \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{A.2}$$

The numerical flux at the cell interface is then formulated as

$$\mathcal{F}_{1/2} = \dot{m}_{1/2}\vec{\psi}_{L/R} + \mathbf{P}_{1/2} \tag{A.3}$$

where $\vec{\psi}_{L/R}$ is determined in a simple upwind fashion,

$$\vec{\psi}_{L/R} = \begin{cases} \vec{\psi}_L & \text{if } \dot{m}_{1/2} > 0 \\ \vec{\psi}_R & \text{otherwise} \end{cases} \tag{A.4}$$

To complete the flux calculation, the value of $\dot{m}_{1/2}$ and $\mathbf{P}_{1/2}$ must be determined.

A.1 Mass Flux

The mass flux at the interface is then taken again to have the form

$$\dot{m}_{1/2} = a_{1/2} M_{1/2} \begin{cases} \rho_L & \text{if } M_{1/2} > 0 \\ \rho_R & \text{otherwise} \end{cases} \tag{A.5}$$

where $M_{1/2}$ is the interface Mach number and $a_{1/2}$ is a simple average of the left and right sound speeds. To determine $M_{1/2}$, the left and right Mach numbers are defined as

$$M_L = \frac{u_L}{a_{1/2}}, \quad M_R = \frac{u_R}{a_{1/2}}. \tag{A.6}$$

Next, a mean local Mach number, $\bar{M}$, and a reference Mach number, M_o, is calculated

$$\bar{M}^2 = \frac{u_L^2 + u_R^2}{2a_{1/2}^2} \tag{A.7}$$

$$M_o^2 = min\left(1, max\left(\bar{M}^2, M_{r_{min}}^2\right)\right) \in [0, 1] \tag{A.8}$$

A scaling function, f_a, is calculated to properly scale the numerical dissipation with the flow speed. It is taken to have the form

$$f_a(M_o) = M_o(2 - M_o) \in [0, 1] \tag{A.9}$$

Finally, the interface Mach number, $M_{1/2}$, is calculated as

$$M_{1/2} = \mathscr{M}^{+}_{(4)}(M_L) + \mathscr{M}^{-}_{(4)}(M_R) - \frac{K_p}{f_a} max(1 - \sigma \bar{M}^2, 0) \frac{p_R - p_L}{\rho_{1/2} a^2_{1/2}} \tag{A.10}$$

where $\rho_{1/2} = \frac{1}{2}(\rho_L + \rho_R)$, $0 \leq K_p \leq 1$ and $\sigma \leq 1$. The split Mach numbers $\mathscr{M}^{\pm}_{(m)}$ are polynomial functions of degree m, and are given as

$$\mathscr{M}^{\pm}_{(1)} = \frac{1}{2}(M \pm |M|) \tag{A.11}$$

$$\mathscr{M}^{\pm}_{(2)} = \pm \frac{1}{4}(M \pm 1)^2 \tag{A.12}$$

$$\mathscr{M}^{\pm}_{(4)} = \begin{cases} \mathscr{M}^{\pm}_{(1)} & \text{if } |M| \geq 1 \\ \mathscr{M}^{\pm}_{(2)}(1 \mp 16\beta \mathscr{M}^{\mp}_{(2)}) & \text{otherwise} \end{cases} \tag{A.13}$$

with $\beta = 1/8$.

A.2 Pressure Flux

The pressure flux is computed as

$$p_{1/2} = \mathscr{P}^{+}_{(5)}(M_L) p_L + \mathscr{P}^{-}_{(5)}(M_R) p_R - K_u \mathscr{P}^{+}_{(5)} \mathscr{P}^{-}_{(5)} (\rho_L + \rho_R)(f_a a_{1/2})(u_R - u_L) \tag{A.14}$$

where $0 \leq K_u \leq 1$ and

$$\mathscr{P}^{\pm}_{(5)} = \begin{cases} \frac{1}{M} \mathscr{M}^{\pm}_{(1)} & \text{if } |M| \geq 1 \\ \mathscr{M}^{\pm}_{(2)} \left[(\pm 2 - M) \mp 16\alpha M \mathscr{M}^{\mp}_{(2)} \right] & \text{otherwise} \end{cases} \tag{A.15}$$

with

$$\alpha = \frac{3}{16}\left(-4 + 5 f_a^2\right) \in \left[-\frac{3}{4}, \frac{3}{16} \right] \tag{A.16}$$

Appendix B

Eigenstructure Analysis for the Favre-Averaged Navier-Stokes Equations

The conserved solution vector, $\mathbf{U}$, of the Favre-Averaged Navier-Stokes (FANS) equations as summarized in Chapter 2 is given by

$$\mathbf{U} = [\rho \quad \rho u \quad \rho v \quad \rho E \quad \rho k \quad \rho\omega]^T \tag{B.1}$$

The corresponding primitive solution vector, $\mathbf{W}$, is given by

$$\mathbf{W} = [\rho \quad u \quad v \quad p \quad k \quad \omega]^T \tag{B.2}$$

B.1 Jacobian Matrix of Inviscid Flux

The Jacobian of the inviscid flux, $\frac{\partial \mathbf{F}}{\partial \mathbf{U}}$, is

$$\frac{\partial \mathbf{F}}{\partial \mathbf{U}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ -u^2 + \frac{u^2+v^2}{2}(\gamma-1) & -u\,(\gamma-3) & -v\,(\gamma-1) & \gamma-1 & \frac{5}{3}-\gamma & 0 \\ -uv & v & u & 0 & 0 & 0 \\ A_{41} & A_{42} & -uv\,(\gamma-1) & u\gamma & -u\left(\gamma-\frac{5}{3}\right) & 0 \\ -uk & k & 0 & 0 & u & 0 \\ -u\omega & \omega & 0 & 0 & 0 & u \end{bmatrix} \tag{B.3}$$

where

$$A_{41} = u\left[\left(\frac{u^2+v^2}{2}\right)(\gamma-2) - \frac{5}{3}k - \frac{a^2}{\gamma-1}\right] \tag{B.4}$$

$$A_{42} = \frac{u^2+v^2}{2} - u^2(\gamma-1) + \frac{5}{3}k + \frac{a^2}{\gamma-1} \tag{B.5}$$

B.2 Coefficient Matrices

The coefficient matrix, $\mathbf{A}$, is then given by

$$\mathbf{A} = \begin{bmatrix} u & \rho & 0 & 0 & 0 & 0 \\ \frac{2}{3}\frac{k}{\rho} & u & 0 & \frac{1}{\rho} & \frac{2}{3} & 0 \\ 0 & 0 & u & 0 & 0 & 0 \\ 0 & \gamma p + \frac{2}{3}\rho k(\gamma-1) & 0 & u & 0 & 0 \\ 0 & 0 & 0 & 0 & u & 0 \\ 0 & 0 & 0 & 0 & 0 & u \end{bmatrix} \tag{B.6}$$

B.3 Eigenvalues

The eigenvalues for the Jacobian matrix, $\frac{\partial \mathbf{F}}{\partial \mathbf{U}}$, are

$$\lambda_1 = u - c, \quad \lambda_2 = u, \quad \lambda_3 = u, \quad \lambda_4 = u, \quad \lambda_5 = u, \quad \lambda_6 = u + c \tag{B.7}$$

where c is the turbulence-modified speed of sound given by

$$c = \sqrt{a^2 + \frac{2}{3}\gamma k} \tag{B.8}$$

B.4 Eigenvectors

The right primitive eigenvector matrix, R_p, can then be shown to have the form

$$\mathbf{R_p} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ -\frac{c}{\rho} & 0 & 0 & 0 & 0 & \frac{c}{\rho} \\ 0 & 0 & 0 & 0 & 1 & 0 \\ c^2 - \frac{2}{3}k & 0 & -\frac{2}{3}k & -\frac{2}{3}\rho & 0 & c^2 - \frac{2}{3}k \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{B.9}$$

and the left primitive eigenvector matrix, L_p, is given by

$$\mathbf{L_p} = \begin{bmatrix} \frac{k}{3c^2} & -\frac{\rho}{2c} & 0 & \frac{1}{2c^2} & \frac{\rho}{3c^2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1-\frac{2}{3}\frac{k}{c^2} & 0 & 0 & -\frac{1}{c^2} & -\frac{2}{3}\frac{\rho}{c^2} & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ \frac{k}{3c^2} & \frac{\rho}{2c} & 0 & \frac{1}{2c^2} & \frac{\rho}{3c^2} & 0 \end{bmatrix} \tag{B.10}$$

Finally, the right conservative eigenvector matrix, R_c, is given by

$$\mathbf{R_c} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ u-c & 0 & 0 & 0 & u & u+c \\ v & 0 & 0 & 1 & 0 & v \\ h-cu+\frac{2k}{3} & 0 & \frac{3\gamma-5}{3(\gamma-1)} & v & \frac{u^2-v^2}{2} & h+cu+\frac{2k}{3} \\ k & 0 & 1 & 0 & 0 & k \\ \omega & 1 & 0 & 0 & 0 & \omega \end{bmatrix} \tag{B.11}$$

where

$$h = \frac{\gamma p}{(\gamma-1)\rho} + \frac{u^2+v^2}{2} + k \tag{B.12}$$

Appendix C

Low-Mach Number Preconditioning and Eigenstructure for the Favre-Averaged Navier-Stokes Equations

The difficulty with the prediction of low-speed flows using the compressible form of the Navier-Stokes equations lies in the disparity between the acoustic and convective wave speeds. These wave speeds are associated with the eigenvalues of the inviscid portion of the equations. An appropriate preconditioner matrix modifies the eigenstructure in such a way to remove the disparity between the eigenvalues at low Mach numbers and can be used to omprove the numerical predictions.

To develop a preconditioner for the Favre-Averaged Navier-Stokes (FANS) equations, it is useful to first examine the preconditioner matrix for the Euler equations. The Weiss-Smith preconditioner is examined in the context of the Euler equations in Section C.1. This preconditioner is then extended to the FANS equations given in Section C.2.

C.1 Preconditioner for the Euler Equations

Consider the preconditioned Euler equations

$$\mathbf{\Gamma}\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y} = 0 \tag{C.1}$$

where

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho u(E + \frac{p}{\rho}) \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho v(E + \frac{p}{\rho}) \end{bmatrix}. \tag{C.2}$$

The low Mach number preconditioner developed by Weiss and Smith in terms of the conserved variables is given by

$$\mathbf{\Gamma} = \frac{1}{a^2 M_r^2} \begin{bmatrix} -V\alpha + a^2 M_r^2 & u\alpha & v\alpha & -\alpha \\ -Vu\alpha & u^2\alpha + a^2 M_r^2 & uv\alpha & -u\alpha \\ -Vv\alpha & uv\alpha & v^2\alpha + a^2 M_r^2 & -v\alpha \\ -V\delta & u\delta & v\delta & -\delta + a^2 M_r^2 \end{bmatrix}, \tag{C.3}$$

where

$$\begin{aligned} V &= \frac{u^2 + v^2}{2} \\ \alpha &= (M_r^2 - 1)(\gamma - 1) \\ \delta &= (M_r^2 - 1)\left[\left(\frac{u^2 + v^2}{2}\right)(\gamma - 1) + a^2\right] \end{aligned}$$

and M_r is the reference Mach Number defined for inviscid flows as

$$M_r = \min\left(\max\left(\frac{u}{a}, M_{\mathrm{r_{min}}}\right), 1\right) \tag{C.4}$$

with the additional constraint for viscous flows,

$$M_r = \max\left(M_r, \frac{\mu}{a\rho\Delta x}\right). \tag{C.5}$$

This preconditioner can be re-written in terms of the characteristic variables,

$$\mathrm{d}\mathbf{W_c} = \begin{bmatrix} -\frac{\rho}{2a}\mathrm{d}u + \frac{1}{2a^2}\mathrm{d}p \\ \mathrm{d}\rho - \frac{1}{a^2}\mathrm{d}p \\ \mathrm{d}v \\ \frac{\rho}{2a}\mathrm{d}u + \frac{1}{2a^2}\mathrm{d}p \end{bmatrix}, \tag{C.6}$$

by the following transformation

$$\mathbf{\Gamma_c} = \frac{\partial \mathbf{W_c}}{\partial \mathbf{U}} \mathbf{\Gamma} \frac{\partial \mathbf{U}}{\partial \mathbf{W_c}} \tag{C.7}$$

$$= \begin{bmatrix} \frac{M_r^2+1}{2M_r^2} & 0 & 0 & -\frac{M_r^2-1}{2M_r^2} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\frac{M_r^2-1}{2M_r^2} & 0 & 0 & \frac{M_r^2+1}{2M_r^2} \end{bmatrix}. \tag{C.8}$$

This results in a preconditioned system in terms of the characteristic variables

$$\mathbf{\Gamma_c}\frac{\partial \mathbf{W_c}}{\partial t} + \mathbf{\Lambda_x}\frac{\partial \mathbf{W_c}}{\partial x} + \mathbf{\Lambda_y}\frac{\partial \mathbf{W_c}}{\partial y} = 0 \tag{C.9}$$

where

$$\mathbf{\Lambda_x} = \begin{bmatrix} u-a & 0 & 0 & 0 \\ 0 & u & 0 & 0 \\ 0 & 0 & u & 0 \\ 0 & 0 & 0 & u+a \end{bmatrix}, \quad \mathbf{\Lambda_y} = \begin{bmatrix} v-a & 0 & 0 & 0 \\ 0 & v & 0 & 0 \\ 0 & 0 & v & 0 \\ 0 & 0 & 0 & v+a \end{bmatrix} \tag{C.10}$$

From inspection, it is clear that the preconditioner modifies the eigenvalues, $u \pm a$ and $v \pm a$, associated with the acoustic waves and does not modify the eigenvalues associated with the convective waves.

C.2 Extension to the Favre-Averaged Navier-Stokes Equations

We now consider the Favre-Averaged Navier-Stokes equations with the k-ω turbulence model. The viscous fluxes and source terms are not considered in this analysis since the eigenvalues pertain only to the inviscid fluxes.

The governing equations in terms of the characteristic variables,

$$\mathrm{d}\mathbf{W_c} = \begin{bmatrix} \frac{k}{3c^2}\mathrm{d}\rho - \frac{\rho}{2c}\mathrm{d}u + \frac{1}{2c^2}\mathrm{d}p + \frac{\rho}{3c^2}\mathrm{d}k \\ \mathrm{d}\omega \\ -\frac{(-3c^2+2k)}{3c^2}\mathrm{d}\rho - \frac{1}{c^2}\mathrm{d}p - \frac{2\rho}{3c^2}\mathrm{d}k \\ \mathrm{d}k \\ \mathrm{d}v \\ \frac{k}{3c^2}\mathrm{d}\rho + \frac{\rho}{2c}\mathrm{d}u + \frac{1}{2c^2}\mathrm{d}p + \frac{\rho}{3c^2}\mathrm{d}k \end{bmatrix}, \tag{C.11}$$

are given by

$$\frac{\partial \mathbf{W_c}}{\partial t} + \mathbf{\Lambda_x}\frac{\partial \mathbf{W_c}}{\partial x} + \mathbf{\Lambda_y}\frac{\partial \mathbf{W_c}}{\partial y} = 0 \tag{C.12}$$

where

$$\mathbf{\Lambda_x} = \begin{bmatrix} u-c & 0 & 0 & 0 & 0 & 0 \\ 0 & u & 0 & 0 & 0 & 0 \\ 0 & 0 & u & 0 & 0 & 0 \\ 0 & 0 & 0 & u & 0 & 0 \\ 0 & 0 & 0 & 0 & u & 0 \\ 0 & 0 & 0 & 0 & 0 & u+c \end{bmatrix}, \quad \mathbf{\Lambda_y} = \begin{bmatrix} v-c & 0 & 0 & 0 & 0 & 0 \\ 0 & v & 0 & 0 & 0 & 0 \\ 0 & 0 & v & 0 & 0 & 0 \\ 0 & 0 & 0 & v & 0 & 0 \\ 0 & 0 & 0 & 0 & v & 0 \\ 0 & 0 & 0 & 0 & 0 & v+c \end{bmatrix}$$

and $c = \sqrt{a^2 + \frac{2}{3}\gamma k}$, the turbulence-modified speed of sound.

From the previous analysis, an appropriate preconditioner that modifies the eigenvalues associated with the acoustic waves, $u \pm c$ and $v \pm c$, is given by

$$\mathbf{\Gamma_c} = \begin{bmatrix} \frac{M_r^2+1}{2M_r^2} & 0 & 0 & 0 & 0 & -\frac{M_r^2-1}{2M_r^2} \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ -\frac{M_r^2-1}{2M_r^2} & 0 & 0 & 0 & 0 & \frac{M_r^2+1}{2M_r^2} \end{bmatrix} \tag{C.13}$$

where M_r is re-defined for inviscid flows as

$$M_r = \min\left(\max\left(\frac{u}{c}, M_{\mathrm{r_{min}}}\right), 1\right) \tag{C.14}$$

with the additional constraint for viscous flows that

$$M_r = \max\left(M_r, \frac{\mu}{c\rho\Delta x}\right). \tag{C.15}$$

Written in conservative variables via the transformation

$$\mathbf{\Gamma} = \frac{\partial \mathbf{U}}{\partial \mathbf{W_c}} \mathbf{\Gamma_c} \frac{\partial \mathbf{W_c}}{\partial \mathbf{U}} \tag{C.16}$$

the preconditioner is then given by

$$\mathbf{\Gamma} = \frac{1}{M_r^2 c^2} \begin{bmatrix} -V\alpha + M_r^2 c^2 & u\alpha & v\alpha & -\alpha & \beta & 0 \\ -Vu\alpha & u^2\alpha + M_r^2 c^2 & uv\alpha & -u\alpha & u\beta & 0 \\ -Vv\alpha & uv\alpha & v^2\alpha + M_r^2 c^2 & -v\alpha & v\beta & 0 \\ -V\delta & u\delta & v\delta & -\delta + M_r^2 c^2 & \frac{3\gamma-5}{3(\gamma-1)}\delta & 0 \\ -Vk\alpha & uk\alpha & vk\alpha & -k\alpha & k\beta + M_r^2 c^2 & 0 \\ -V\omega\alpha & u\omega\alpha & v\omega\alpha & -\omega\alpha & \omega\beta & M_r^2 c^2 \end{bmatrix} \tag{C.17}$$

where

$$\begin{aligned} V &= \frac{u^2+v^2}{2} \\ \alpha &= (M_r^2 - 1)(\gamma - 1) \\ \beta &= (M_r^2 - 1)\left(\gamma - \frac{5}{3}\right) \\ \delta &= (M_r^2 - 1)\left[\left(\frac{u^2+v^2}{2} + k\right)(\gamma - 1) - \frac{2}{3}k + c^2\right]. \end{aligned}$$

C.3 Preconditioned Eigenvalues

The preconditioned eigenvalues are

$$\lambda_1 = u' - c', \quad \lambda_2 = u, \quad \lambda_3 = u, \quad \lambda_4 = u, \quad \lambda_5 = u, \quad \lambda_6 = u' + c'. \tag{C.18}$$

The preconditioned velocity, u', and preconditioned turbulence-modified sound speed, c', are given by

$$u' = u(1-\alpha) \tag{C.19}$$

$$c' = \sqrt{\alpha^2 u^2 + M_r^2 c^2} \tag{C.20}$$

with

$$\alpha = \frac{1}{2}\left(1 - M_r^2\right). \tag{C.21}$$

C.4 Preconditioned Eigenvectors

The preconditioned right primitive eigenvector matrix R'_p is given by

$$\mathbf{R'_p} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ \frac{2c^2}{\rho[u(M_r^2-1)-2c']} & 0 & 0 & 0 & 0 & \frac{2c^2}{\rho[u(M_r^2-1)+2c']} \\ 0 & 0 & 0 & 0 & 1 & 0 \\ c^2 - \frac{2}{3}k & 0 & -\frac{2}{3}k & -\frac{2}{3}\rho & 0 & c^2 - \frac{2}{3}k \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{C.22}$$

The preconditioned left primitive eigenvector matrix L'_p is given by

$$\mathbf{L'_p} = \begin{bmatrix} -\frac{k[u(M_r^2-1)-2c']}{6c'c^2} & -\frac{M_r^2\rho}{2c'} & 0 & -\frac{u(M_r^2-1)-2c'}{4c'c^2} & -\frac{\rho[u(M_r^2-1)-2c']}{6c'c^2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 - \frac{2k}{3c^2} & 0 & 0 & -\frac{1}{c^2} & -\frac{2\rho}{3c^2} & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ \frac{k[u(M_r^2-1)+2c']}{6c'c^2} & \frac{M_r^2\rho}{2c'} & 0 & \frac{u(M_r^2-1)+2c'}{4c'c^2} & \frac{\rho[u(M_r^2-1)+2c']}{6c'c^2} & 0 \end{bmatrix} \tag{C.23}$$

The preconditioned right conservative eigenvector matrix R'_c is given by

$$\mathbf{R'_c} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ \frac{u(1+M_r^2)-2c'}{2M_r^2} & 0 & 0 & 0 & u & \frac{2(u^2-c^2)}{u(1+M_r^2)-2c'} \\ v & 0 & 0 & 1 & 0 & v \\ R'_{c_{41}} & 0 & \frac{3\gamma-5}{3(\gamma-1)} & v & \frac{u^2-v^2}{2} & R'_{c_{46}} \\ k & 0 & 1 & 0 & 0 & k \\ \omega & 1 & 0 & 0 & 0 & \omega \end{bmatrix} \tag{C.24}$$

where

$$R'_{c_{41}} = h + \frac{c'u}{M_r^2} + \frac{2k}{3} + \frac{u^2(1-M_r^2)}{2M_r^2} \tag{C.25}$$

$$R'_{c_{46}} = h - \frac{c'u}{M_r^2} + \frac{2k}{3} + \frac{u^2(1-M_r^2)}{2M_r^2} \tag{C.26}$$

C.5 Modifications for a Moving Cell Interface

The Jacobian $\frac{\partial F}{\partial U}$ has been modified to account for flux due to a moving cell interface. Hence, the preconditioned eigenvalues and preconditioned eigenvectors need to be modified as well.

For a cell interface moving at a velocity $\vec{w} = w_x\hat{i} + w_y\hat{j}$, the preconditioned eigenvalues are

$$\lambda_1 = u''-c'', \quad \lambda_2 = u-w_x, \quad \lambda_3 = u-w_x, \quad \lambda_4 = u-w_x, \quad \lambda_5 = u-w_x, \quad \lambda_6 = u''+c'' \tag{C.27}$$

where the preconditioned velocity and preconditioned turbulence-modified sound speed are modified to account for a moving cell interface

$$u'' = (u-w_x)(1-\alpha) \tag{C.28}$$

$$c'' = \sqrt{\alpha^2(u-w_x)^2 + M_r^2c^2}. \tag{C.29}$$

The preconditioned right primitive eigenvector matrix $\mathbf{R'_p}$ is given by

$$\mathbf{R'_p} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ \frac{(u-w_x)(1-M_r^2)-2c''}{2M_r^2\rho} & 0 & 0 & 0 & 0 & \frac{(u-w_x)(1-M_r^2)+2c''}{2M_r^2\rho} \\ 0 & 0 & 0 & 0 & 1 & 0 \\ c^2-\frac{2}{3}k & 0 & -\frac{2}{3}k & -\frac{2}{3}\rho & 0 & c^2-\frac{2}{3}k \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{C.30}$$

The preconditioned left primitive eigenvector matrix $\mathbf{L}'_{\mathbf{p}}$ is given by

$$\mathbf{L}'_{\mathbf{p}} = \begin{bmatrix} -\frac{k[(u-w_x)(M_r^2-1)-2c'']}{6c''c^2} & -\frac{M_r^2\rho}{2c''} & 0 & -\frac{(u-w_x)(M_r^2-1)-2c''}{4c''c^2} & -\frac{\rho[(u-w_x)(M_r^2-1)-2c'']}{6c''c^2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1-\frac{2k}{3c'} & 0 & 0 & -\frac{1}{c^2} & -\frac{2\rho}{3c^2} & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ \frac{k[(u-w_x)(M_r^2-1)+2c'']}{6c''c^2} & \frac{M_r^2\rho}{2c''} & 0 & \frac{(u-w_x)(M_r^2-1)+2c''}{4c''c^2} & \frac{\rho[(u-w_x)(M_r^2-1)+2c'']}{6c''c^2} & 0 \end{bmatrix} \tag{C.31}$$

The preconditioned right conservative eigenvector matrix $\mathbf{R}'_{\mathbf{c}}$ is given by

$$\mathbf{R}'_{\mathbf{c}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ R'_{c_{21}} & 0 & 0 & 0 & u & R'_{c_{26}} \\ v & 0 & 0 & 1 & 0 & v \\ R'_{c_{41}} & 0 & \frac{3\gamma-5}{3(\gamma-1)} & v & \frac{u^2-v^2}{2} & R'_{c_{46}} \\ k & 0 & 1 & 0 & 0 & k \\ \omega & 1 & 0 & 0 & 0 & \omega \end{bmatrix} \tag{C.32}$$

where

$$R'_{c_{21}} = \frac{2\left[u^2-c^2+uw_x(M_r^2-1)\right]}{M_r^2(u+w_x)+u-w_x+2c''} \tag{C.33}$$

$$R'_{c_{26}} = \frac{2\left[u^2-c^2+uw_x(M_r^2-1)\right]}{M_r^2(u+w_x)+u-w_x-2c''} \tag{C.34}$$

$$R'_{c_{41}} = \frac{u^2}{2M_r^2}-\frac{uw_x}{2M_r^2}+\frac{uw_x}{2}+\frac{v^2}{2}-\frac{c''u}{M_r^2}+\frac{c^2}{\gamma-1}+\frac{k\gamma}{\gamma-1}-\frac{5k}{3(\gamma-1)} \tag{C.35}$$

$$R'_{c_{46}} = \frac{u^2}{2M_r^2}-\frac{uw_x}{2M_r^2}+\frac{uw_x}{2}+\frac{v^2}{2}+\frac{c''u}{M_r^2}+\frac{c^2}{\gamma-1}+\frac{k\gamma}{\gamma-1}-\frac{5k}{3(\gamma-1)} \tag{C.36}$$