Development of methods to improve the accuracy and efficiency of unsteady incompressible flow simulation

Matthew Vincent Fischels
Iowa State University

Development of methods to improve the accuracy and efficiency of unsteady incompressible flow simulation

by

Matthew Vincent Fischels

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Aerospace Engineering and Wind Energy Science, Engineering, and Policy

Iowa State University

Ames, Iowa

2018

DEDICATION

To Kelsey, for her constant support and patience.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENTS

This work is supported by the National Science Foundation through its Integrated Graduate Education and Research Traineeship program, award 1069283, and under grant numbers 0947929, 1101284, and 1440446. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

I would like to thank Dr. Ganesh Rajagopalan for believing in me and supporting me throughout my research. You gave me the freedom to research the questions I was interested in, but also provided a depth of knowledge and advice crucial for crossing difficult roadblocks and finishing my degree. I would also like to thank Dr. James McCalley for allowing me the opportunity to be part of the NSF IGERT WESEP program, and teaching me so many things I never expected to learn.

Also, thank you to my graduate student colleagues (Dan Garrick, Avinaash Murali, Kshitiz Subedi, and Jiangli Yin) for the time you spent patiently teaching and listening to me. Thank you to my POS committee for their time spent listening to my research presentations and giving positive feedback that challenged my thinking.

I also want to thank my wife Kelsey, who never doubted me even when I doubted myself. Your support and patience through this difficult but rewarding time has kept me going. Lastly, thank you to my mother, father, and sister, who have done more for me than I can ever repay.

ABSTRACT

The runtime of unsteady incompressible flows were reduced through different techniques in the formulation and solution of the governing equations. The implicit Runge-Kutta based IRK-SIMPLER algorithm was developed and compared to the Crank-Nicholson based SIMPLER and the explicit RK-SIMPLER algorithms. The IRK-SIMPLER algorithm was tested for steady and unsteady problems, both on structured Cartesian and unstructured vertex-centered median-dual grids, and proved to be an accurate and robust algorithm that also required less runtime. Further, a second order unstructured Flux Corrected Method improved the accuracy of flux calculation with minimal/negligible increase in runtime. The Flux Corrected Method provided a small stencil scheme that required little additional runtime compared to the commonly used Power Law scheme. Wind turbine cases were simulated with momentum source modeling. Both steady rotor models and three variations of the unsteady momentum source model were tested. The power predicted by the simulations matched experiments well, and for the cases presented, the simulations required 447 times less runtime than traditional methods.

CHAPTER 1. INTRODUCTION

1.1 Background

Numerical simulation of complex and large scale engineering fluid flow problems requires substantial resources, including time and computing power. One example is the simulation of flow through a wind farm, which provides a challenge due to the differences in the time and length scales present. Two possible types of remedies to this problem arise: (1) improving the underlying algorithms and methods used to discretize and solve the equations, and (2) using the advances in computing power, such as parallel processing and graphics processing unit acceleration, to solve the equations faster. There continues to be abundant research into using greater computing power, but the area of algorithmic improvements provides an opportunity to develop new methods that increase the efficiency and accuracy of flow simulation, no matter what computing power is available.

The present research focuses on solving unsteady incompressible flows and the particular methods required to solve these problems. This work investigates three different areas in an attempt to reduce runtime: algorithms to solve the set of coupled equations, time integration methods for unsteady flows, and interpolation schemes to compute fluxes at interfaces. Several test cases examine the accuracy and efficiency of the new methods, including steady and unsteady flows and the flow through wind turbines. The rest of this document is laid out as follows. The remainder of Chapter 1 covers the background of previous methods used to solve incompressible flows. Chapter 2 discusses the development of new algorithms, and time integration methods, required for unsteady simulation, using several test cases to evaluate the different algorithms. Chapter 3 covers the formulation of the equations on unstructured vertex-centered grids. Chapter 4 investigates several flux interpolation methods for both structured and unstructured grids. Chapter 5 discusses the details of turbulence modeling with application to unstructured grids. Chapter 6 covers the application of these new methods to the simulation of flow around wind turbines. Chapter 7 covers conclusions and recommendations for future work.

1.2 Unsteady Incompressible Navier-Stokes Equations

The unsteady incompressible Navier-Stokes equations are written as

$$\nabla \cdot (\rho \vec{V}) = 0 \tag{1.1}$$

$$\frac{\partial(\rho \vec{V})}{\partial t} + \nabla \cdot (\rho \vec{V}\vec{V}) = -\nabla p + \nabla \cdot (\mu \nabla \vec{V}) + \vec{S} \,, \tag{1.2}$$

where ρ, $\vec{V}$, t, p, μ, and $\vec{S}$ represent fluid density, velocity, time, pressure, dynamic viscosity, and source term, respectively. A major difficulty in solving these equations is the lack of an equation to solve for pressure, as pressure does not appear in the continuity equation (Eq. 1.1) and only appears as a source in the momentum equation (Eq. 1.2). If the above equations are discretized in space using the finite volume method, the continuity equation becomes a fully discrete, algebraic equation of the form

$$\sum_{f=1}^{nfaces} (\rho \vec{V}) \cdot \vec{A} = 0 \,, \tag{1.3}$$

where $\vec{A}$ represents the area vector pointing out of the control volume, and the summation is over all control volume faces of a cell. Discretizing the momentum equation in space results in a semi-discrete equation of the form

$$\frac{d\vec{V}}{dt} = \vec{F}(t, \vec{V}, p) = \frac{\vec{R}}{\rho \Delta \forall} \,. \tag{1.4}$$

where $\vec{R}$ contains all integrated terms in the momentum equation *excluding* time derivative (i.e. convection, diffusion, pressure gradient, and source terms).

The semi-discrete momentum equation (Eq. 1.4) and discrete continuity equation (Eq. 1.3) form differential algebraic equations (DAEs) (defined as type 2 in Harier [1]) and have the following form.

$$\frac{d\vec{V}}{dt} = f(\vec{V}, p) \tag{1.5}$$

$$0 = g(\vec{V}) \tag{1.6}$$

The spatially discretized momentum terms are contained in f, and the discrete continuity terms are contained in g. DAEs differ from ordinary differential equations (ODEs) as they cannot be simply updated in time simultaneously because there is an equation that does not contain a time derivative term. In the case of the incompressible Navier-Stokes equations, this form does not lead to a straightforward solution of pressure.

Some of the many algorithms developed to solve the incompressible Navier-Stokes equations will be discussed in Section 1.2.1. Sections 1.2.2 and 1.2.3 discuss two other important aspects of solving these equations, time integration methods and flux interpolation methods. After the equations are discretized, integrated, and an algorithm is developed, the methods by which the equations are solved can make a significant difference on the solution's accuracy, as well as how quickly it can be achieved. Some of the different numerical methods are discussed in Section 1.2.4.

1.2.1 Algorithms

In solving unsteady incompressible flow problems, traditional iterative algorithms become numerically inefficient as problems become increasingly complex. Pressure is not a variable in the continuity equation and does not have an explicit equation. Three approaches are often used to remedy this problem: (a) artificial compressibility methods [2], (b) vorticity-streamfunction approach [3], and (b) pressure-based methods.

Pressure-based methods derive an equation for pressure to enforce continuity. Projection or fractional step methods [4] are explicit, pressure-based methods which calculate an intermediate velocity, solve a pressure equation to enforce continuity, and correct velocity by using an operator splitting. The Semi-Implicit Method for Pressure Linked Equations (SIMPLE) and SIMPLE Revised (SIMPLER) family of algorithms [5] are implicit pressure-based methods which derive an equation for pressure (or pressure correction) by substituting the discretized momentum equations into the discretized continuity equation. The SIMPLER algorithm is based on an exact pressure equation to find pressure and an approximate pressure correction equation that adjusts velocity to satisfy continuity. The SIMPLE and SIMPLER family of algorithms requires relaxation and sub-iterations at each time step to converge.

In an attempt to improve the efficiency of pressure-based methods and increase the convergence rate, many variants of the pressure-based algorithms have been developed including Runge-Kutta SIMPLER (RK-SIMPLER) [6], Pressure-Implicit with Splitting of Operators (PISO) [7], Coupled and Linked Equations Algorithm Revised (CLEAR) [8], and Inner Doubly-Iterative Efficient Algorithm for Linked equations (IDEAL) [9]. Among these algorithms, CLEAR and IDEAL are implicit algorithms, which use relaxation and sub-iterations to converge at each time step; while PISO (an implicit algorithm) and RK-SIMPLER (an explicit algorithm) update the solution in time without requiring sub-iterations within a time step.

RK methods are commonly used in solving Euler equations [10],[11] and the compressible Navier-Stokes equations [12],[13],[14] because the spatially discretized conservation equations form ordinary differential equations that are integrated in time with RK methods. However, for incompressible flow, there is no explicit equation for pressure, and the spatially discretized continuity and momentum equations form differential algebraic equations with an index of 2 as defined in Hairer [1]. Solving the incompressible Navier-Stokes equations with RK methods is more intricate than the compressible Navier-Stokes equations. Several incompressible algorithms that use RK methods are developed in [15],[16],[17],[18],[19],[20],[21], and [22]. Sanderse [22] developed an algorithm that uses explicit RK methods and a projection method to integrate in time while achieving a higher temporal order of accuracy. Sanderse shows that to achieve the temporal order of accuracy of the RK method, continuity must be accounted for at each RK stage. The primary interest in RK based methods is the accuracy of the algorithm, not on the the efficiency of the algorithms, which is often not presented or discussed.

The RK-SIMPLER algorithm is a pressure-based method developed to solve the incompressible Navier-Stokes equations using explicit RK methods. The RK-SIMPLER algorithm forms an equation for pressure by combining the discretized continuity and momentum equations in exactly the same way as the original SIMPLER algorithm. The solution of this equation is used as an explicit source term in the momentum equations. This results in momentum equations becoming ordinary differential equations, which are integrated in time with explicit RK stages. However, the RK-SIMPLER algorithm often requires small time steps due to the explicit nature of the algorithm [6].

To relax the time step restrictions, implicit RK stages are used in place of the explicit stage equations. As a part of this research, an implicit RK algorithm is developed called IRK-SIMPLER. For both the RK-SIMPLER and IRK-SIMPLER algorithms, two variants are investigated. The first variant derives and solves the pressure equation once per time step, like the RK-SIMPLER algorithm in Rajagopalan [6]. The second variant uses the conclusion of Sanderse [22] that pressure should be solved at each stage to improve accuracy. These four different algorithms are developed (in Secs. 2.3.3 and 2.3.4) and tested to determine the algorithm that results in an accurate solution with the lowest runtime.

1.2.2 Time Integration Methods

Integration of a semi-discrete equation in time is completed with many different methods. One of the important qualities of a method is the order of accuracy. For any given method, as the time step size (Δt) decreases the error (e) in the solution also decreases. The order of accuracy of a method determines how much the error decreases with time step size. If the error is written as $e = A\ \Delta t^n$, then the order of accuracy is n. This is often found graphically by plotting the error versus time step size on a log-log scale, where the order of accuracy is the slope of the line. Methods with a higher order of accuracy tend to achieve accurate solutions using larger time step sizes, but are not guaranteed because the magnitude A can differ between methods.

The three factors that determine overall effectiveness and efficiency of a time integration method are order of accuracy, stability (maximum allowable time step size), and the computational work required to complete the integration. An ideal time integration method would have a high order of accuracy, allow for large time step sizes, and require little computational time. Unfortunately, this ideal method does not exist. Implicit methods allow for larger time step sizes but require more computational effort. Explicit methods require low time step sizes but have lower computational cost. For both implicit and explicit methods, a higher order of accuracy method requires more computational effort to complete one time step. By using a more accurate method, the same level of accuracy can be achieved by taking fewer time steps and requiring less time overall.

One type of time integration is the single step method. Three common methods in this family are Euler Explicit, Crank-Nicolson, and Euler Implicit (or Fully Implicit). These three methods integrate to the next time step using only the current time level and the final time level information, with Euler Explicit and Euler Implicit being first order accurate and Crank-Nicolson being second order accurate. The advantage of an explicit method like Euler Explicit is the need to only know information at the current time level. However the stability of these methods are poor and require small time step sizes. Implicit methods, like Crank-Nicolson and Euler Implicit, have better stability and are able to take large time steps, but they require information from the next time step, which is unknown. This requires a more expensive solution method, particularly for a large system of coupled equations. The overall efficiency of an explicit or implicit method depends on the problem and method of choice.

Another type of time integration method is the family of multi-step methods, including Adams-Bashforth methods [23] and Backwards Differentiation Formulas (BDF) [24]. These multi-step methods are implicit methods that integrate from one time step to the next using information from the current time step and information from at least one previous time step. The single step method is considered the simplest form of the multi-step method, using only one step. To achieve higher order, more time steps are included, which requires more information to be stored. Another issue with multi-step methods is start-up, when information from past time steps is not known. Despite these issues, multi-step methods are commonly used, and provide techniques for higher order accuracy.

A family of integration methods called Runge-Kutta (RK) methods involve stages between time steps. To integrate from one time step to the next, only the information at the current time step is required. Explicit and implicit Runge-Kutta methods exist and many different variations have been developed to achieve a certain level of stability and accuracy. To achieve higher order accuracy, Runge-Kutta methods require more stages, which increases the amount of computations for both explicit and implicit methods.

Explicit Runge-Kutta methods only need information from past stages to solve each stage, leading to simple explicit equations. To increase the order of accuracy, many more stages need to be added.

Implicit Runge-Kutta methods need information from the current stage and future stages to solve each stage, leading to a system of equations that must be solved simultaneously. For a large system of coupled ODEs, solving a system of simultaneous equations is costly. Different types of implicit RK methods are developed to be both efficient and accurate. The first type of implicit RK method is the Fully Implicit Runge-Kutta (FIRK) method. This method requires all stages to be solved simultaneously, with each stage dependent on all other stages. For a large system (i.e. a grid with many grid points to be solved), the system of equations to be solved becomes very large as the number of stages increases. FIRK methods achieve high order accuracy with few stages, with an order of accuracy of $n = 2 * S - 1$, where n is order of accuracy and S is the number of stages. This means 1st order accuracy is the best possible for a one stage method, 3rd order accuracy is the best possible for a two stage method, and 5th order accuracy is possible with only three stages.

Other forms of implicit RK methods exist, including methods such as Diagonally Implicit RK (DIRK), Singly-Diagonal RK (SDIRK), Explicit first stage Diagonal RK (EDIRK), and Explicit first stage Singly Diagonal RK (ESDIRK). These four methods all have the property that each stage only requires information from previous stages and the current stage. Therefore, each stage is solved sequentially with a smaller system of simultaneous equations at each stage.

Section 2.2 continues the discussion of different time integration methods and their application to the solution of the incompressible Navier-Stokes equations.

1.2.3 Flux Interpolation Schemes

For finite volume methods, the flux at the interface between two control volumes are calculated from flow variables typically stored at the center of the control volumes. For viscous fluid flow, both the inviscid (convective) and viscous (diffusive) fluxes are calculated. Many different methods calculate or interpolate flux to the control volume face. Following the goal of the present research, the most desirable flux methods yield an accurate solution with the lowest runtime. Often methods are developed to yield a high order of accuracy and achieve the most accurate solution on the coarsest grid. It is possible that a higher order accuracy method requires more work and takes more runtime to achieve a certain level of accuracy than a lower order of accuracy method. To determine the best method in the present research, a certain level of accuracy is specified (for example an error tolerance of 1%) and the method which reaches that accuracy with the lowest runtime is determined to be the best method.

The calculation of the flux at an interface is placed into two areas: methods based primarily on geometry, and methods based primarily on physics. The methods based primarily on geometry use a certain stencil, including a number of points where data is known, and fit a polynomial or some other function to match that data. Methods based primarily on physics use known conservation equations to develop a method to calculate fluxes based on a local approximation to the conservation equation.

Within the geometric methods, the inviscid and viscous fluxes are often calculated using different methods because the physics and numerical behavior of each is different. Viscous fluxes are relatively benign and stable due to the physics of diffusion, and central difference methods are often used. For inviscid fluxes, stability becomes an issue, and central difference methods are unstable. Methods for inviscid fluxes often use upwinding, common examples include first order upwinding [5], second order upwinding [25], and QUICK schemes [26]. These methods take the velocity vector into account and fit a polynomial through one upwind point, two upwind points, and two upwind and one downwind point, respectively. Other methods include the essentially non-oscillatory

(ENO) [27] and weighted ENO (WENO) [28] schemes, which provide high-order interpolation in both smooth regions and regions with discontinuities.

The physics based schemes take the full momentum conservation equations into account for flux calculation. The simplest of these methods is the Exponential scheme [5], which simplifies the momentum equations to a one-dimensional, steady convection-diffusion equation, and uses the exact solution which is an exponential function. Two variations of this method are the Hybrid and Power Law schemes [5], which avoid the expensive calculation of exponential values with polynomial fits to the exponential function. These methods are stable for calculating both the inviscid and viscous fluxes, though the schemes are too diffusive on coarse grids. An improvement to these physics based methods is made in the Flux Corrected Method (FCM) in which the full momentum conservation equation is used to calculate the flux on the interface instead of the one-dimensional steady convection-diffusion equation [29].

For structured grids, all schemes discussed are relatively simple to develop; however, on unstructured grids some methods become much more difficult. For example, the SOU and QUICK schemes require a line through two upwind points, but on unstructured grids, where grid points do not fall on grid lines, fitting this line becomes difficult. On unstructured grids, some of the WENO schemes, as well as the physics based methods, follow the same procedure as the structured grid, only requiring information from both sides of a control volume interface.

For finite element methods, the equations are discretized using the discontinuous Galerkin method, and the accuracy is improved using higher order elements. The elements used in these methods are functions that vary throughout the domain and have variables and their derivatives continuous or discontinuous between elements. In contrast, most finite volume methods assume values are constant throughout a control volume.

There are finite volume methods developed that use finite element shape functions to allow values to vary throughout the control volume [30],[31]. The physics-based Power Law scheme is based on this principle.

1.2.4 Numerical Methods to Solve Linear Equation

Once the Navier-Stokes equations are discretized in space and a time integration method is chosen, the equations become linear systems of equations to be solved using any number of techniques. Directly inverting the system of equations becomes prohibitively costly for even coarse grids. Because the systems of equations are diagonally dominant and sparse, relaxation methods are used to iteratively solve until residuals of the equations reach a certain tolerance. Common methods include Jacobi's method, Gauss-Seidel, and successive over-relaxation.

Other methods use factorization to reduce the linear system to diagonal matrices, such as ILU factorization, LU factorization, and approximate factorization. The factorization allows the system of equations to be solved with fewer computations. The LU and approximate factorizations are used within the IRK-SIMPLER algorithm and are discussed in detail in Appendix B.

Other methods use Newton-Krylov sub-space to approach a zero residual more efficiently. These include Bi-Conjugate Gradient (BiCG), BiCG Stabilized (BiCGSTAB), and Generalized Minimum RESidual (GMRES) methods. These methods involve more computations each iteration, but reduce the residual much faster.

Another approach is multigrid, which uses relaxation methods, like Gauss-Seidel, on several grids with different grid spacing. Multigrid methods quickly reduce error in the equation which have a wavelength proportional to the grid spacing, but longer wavelength errors take many iterations to be removed. Multigrid methods take advantage of this by using a set of coarse and refined grids to reduce all wavelengths of error quickly. Multigrid methods are developed for the IRK-SIMPLER algorithm in Appendix C, but the reduction in runtime was not significant, and other lines of research were investigated in more detail.

Although different numerical methods used to solve linear systems of equations are an important factor in determining the time required to solve the incompressible Navier-Stokes equations, it is not the primary focus of this work. Some methods (LU and approximate factorization and multigrid) are investigated and results are given in the present research, but greater reduction in runtime is found by improving the algorithms and flux methods.

1.3 Turbulence Modeling

For many problems, including wind turbine simulation, flow is turbulent, with eddies and fluctuations at many scales in time and space. The largest eddies have the most energy and are produced from mean flow shear. The smallest eddies have the least energy and are dissipated through viscosity. The transfer from the mean shearing of flow to large eddies and then to small eddies is called the energy cascade. This cascade is a mechanism by which the high energy, large eddies are broken down into smaller and smaller eddies, which are eventually dissipated by viscosity.

The region in which large eddies are broken down into smaller eddies is called the inertial subrange. Eddies in the inertial sub-range all follow the same rate of dissipation which is their size to the 2/3 power, known from Kolmogoroff's law. This law dictates how eddies decrease in energy as they become smaller.

Because turbulence occurs over such a large range of scales, the simulation of turbulent flows have different modeling techniques for different problems. The Navier-Stokes equations include all the physics that govern fluid flow; however, fully resolving the smallest scales of turbulence in space and time requires very high resolution. Simulations that directly use Navier-Stokes with highly refined grids (or using spectral methods) are called Direct Numerical Simulation (DNS). This so called Direct Numerical Simulation is used to gain knowledge in some of the fundamentals of turbulence, but DNS is too costly for realistic engineering applications.

To avoid fully resolving all scales of turbulence, different methods to model (but not actually resolve) the turbulence are developed. One common method to model turbulence is the Reynolds Averaged Navier-Stokes equations (RANS). RANS equations average the Navier-Stokes equations in time, which introduces a new unknown called the Reynolds stress. Many different methods are developed to model the Reynolds stress, including the one-equation Spalart-Allmaras (SA) [32] and Baldwin-Lomax [33] models, two-equation $k-\epsilon$ [34] and $k-\omega$ [35] models, and six-equation models, which solves all six terms of the Reynolds stress using the Reynolds stress transport (RST) equation (see Durbin [36]).

The use of RANS modeling is common in engineering applications for a relatively low cost. However, for some applications large scale eddies have a large impact on flow, and RANS modeling does not give sufficient results. Large Eddy Simulation (LES) is a higher fidelity model that uses refined grids to capture the largest scale of turbulent eddies in the energetic scale, while the smaller scales are modeled with a sub-grid model. LES filters the Navier-Stokes equations, rather than averaging like RANS, and provides a more accurate representation of turbulence with higher grid requirements and runtime. Despite the increased runtime, LES is becoming more prevalent in engineering applications, such as wind turbine simulation, where large scale turbulence is important to accurately capture. A mixture of LES and RANS modeling is possible with Detached Eddy Simulation (DES) [37], which uses flow information to switch between LES and RANS at different points in the domain.

Using the RANS equations and $k-\epsilon$ model is popular for engineering applications, and is used to simulate turbulent flows in present research. One issue with $k-\epsilon$, and similar RANS models, is the near wall boundary conditions. Conditions on the k and ϵ terms can be specified, but require refined grids near the wall for accuracy. Alternatively, wall functions can be used to represent the physics of the turbulent flow near the wall, without the need for highly refined grids. Wall models use the universal law of the wall to set values of k and ϵ near the wall. Wall functions work best when grids are not too close or far from the wall, making grid generation difficult for turbulent problems when the turbulent boundary layer is not known a priori.

Chapter 5 develops $k-\epsilon$ models for unstructured grids and includes the FCM scheme into the turbulence calculation.

1.4 Wind Turbine Modeling

Numerical simulation of wind farms is a complex and costly undertaking, largely due to the significant difference in length scales from the thickness of the boundary layer on turbine blades to the overall size of a wind farm. The physical geometry of the blades can be modeled in the computational domain as moving bodies. Resolving the flow around the moving blades requires a highly refined grid near the blades to accurately capture the boundary layer profile. The blade grid must move as the turbine rotates, and there must be some way to connect the blade grid to the rest of the domain that is not rotating. This can be done with overset grids [38] or with a grid reconnected to the outer domain each time step [39]. A highly refined grid required to fully resolve the blade flow is costly and requires significant computing power and time.

The simulation of wind farms is often simplified using rotor momentum source modeling [40] or similar actuator disk and actuator line methods. These methods use known airfoil data for each blade section to assemble a model for forces acting on the blade. Airfoil data is commonly tabulated for different angles of attach, Reynolds numbers, Mach numbers, and other parameters. By assuming that blade forces have little impact from flow along the blade span, known two-dimensional data is used to find the force on a blade section, and the force over the blade is integrated to find thrust, torque, and other forces. These forces are applied to the flow by adding them to the momentum equation sources at the location of each blade section. The momentum source method has been used for vertical axis wind turbines [40], helicopters [41],[42],[43], and horizontal axis wind turbines [44]. This method greatly reduces the grid points required to achieve an accurate solution.

Using the momentum source method, as rotor blades pass through grid cells, the blade forces must be transferred to the flow. Several methods are developed to accomplish this. One method is to simply add the force of each blade section into the grid cell being intersected by that blade section [40]. Another method is to use a Gaussian distribution to spread the rotor source onto the domain in a smooth manner [45].

One issue with momentum source modeling is the assumption that flow over the blades is two-dimensional, however, this is not a good assumption near the root and tip of the blades and requires corrections to improve results. One option is to use Prandtl's tip correction. Rajagopalan [40] developed a tip correction that uses the local pressure gradient and modifies the blade section angle of attack such that the lift at the tip goes to zero. Jha [46] developed a method that uses an elliptic distribution along the span of the blade, which reduces the force at the tip to zero. Another method uses a non-uniform Gaussian distribution that varies in width along the blade [47].

The methods previously mentioned all introduce the rotor source for a given time step with the blade fixed at one location. However, given that the rotor blades are moving within the time step, present research examines the effect of using a new time accurate, unsteady rotor source location to improve the unsteady solution and reduce the runtime required. This new technique uses the time integration method to determine the blade location within the time step.

The momentum source method is used in the present research to yield accurate values of torque and power on horizontal axis wind turbines. Chapter 6 covers the detail of modeling the turbine blades and results.

1.5 Current Work

Current research examines different methods where the accurate solution of unsteady incompressible flows is achieved faster, with a focus on improving the algorithms and methods to solve the equations. A family of pressure-based algorithms using Runge-Kutta is developed, validated, and tested for efficiency and accuracy. Different flux schemes are examined to determine which methods yield accurate results economically. A new flux scheme for unstructured grids is also presented and works efficiently with a higher order of accuracy. The methods are tested on steady and unsteady cases, as well as cases with turbulence and wind turbine modeling.

CHAPTER 2. ALGORITHMS AND TIME INTEGRATION METHODS

2.1 Governing Equations

To demonstrate the methods required to solve these equations, a two-dimensional structured Cartesian grid is used. The two-dimensional unsteady incompressible flow equations in Cartesian coordinates are written as follows.

$$\frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y} = 0 \tag{2.1}$$

$$\frac{\partial(\rho u)}{\partial t} + \frac{\partial(\rho u u)}{\partial x} + \frac{\partial(\rho v u)}{\partial y} = -\frac{\partial p}{\partial x} + \frac{\partial}{\partial x}\left(\mu\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(\mu\frac{\partial u}{\partial y}\right) + S_u \tag{2.2}$$

$$\frac{\partial(\rho v)}{\partial t} + \frac{\partial(\rho u v)}{\partial x} + \frac{\partial(\rho v v)}{\partial y} = -\frac{\partial p}{\partial y} + \frac{\partial}{\partial x}\left(\mu\frac{\partial v}{\partial x}\right) + \frac{\partial}{\partial y}\left(\mu\frac{\partial v}{\partial y}\right) + S_v \ , \tag{2.3}$$

where (x, y) are the Cartesian coordinates, (u, v) are the x and y velocity components, and (S_u, S_v) are the x and y source term components. The control volume used to integrate these equations is seen in Fig. 2.1.

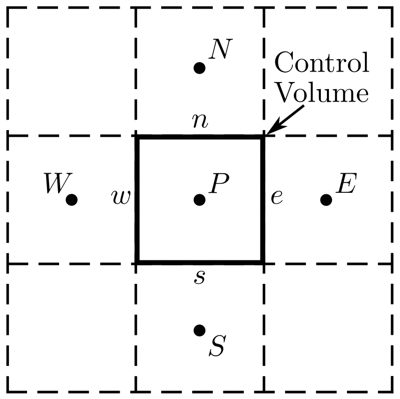


Figure 2.1: Control volume for two-dimensional Cartesian system.

To avoid pressure oscillations, as discussed in Patankar [5], pressure is located at the cell center (P), while velocity components are stored at the cell faces (u at faces e and w, v at faces n and s). Using this arrangement (sometimes referred to as a staggered grid), the continuity equation (Eq.

2.1) is integrated over the control volume and leads to the following discrete equation.

$$F_e - F_w + F_n - F_s = 0 \ , \tag{2.4}$$

where

$$F_e = (\rho u)_e \Delta y \qquad\qquad F_w = (\rho u)_w \Delta y \tag{2.5}$$

$$F_n = (\rho v)_n \Delta x \qquad\qquad F_s = (\rho v)_s \Delta x \ . \tag{2.6}$$

Flux across control volume faces (F) is calculated using the face centered velocity components.

The momentum equations (Eqs. 2.2 and 2.3) are rewritten as

$$\frac{\partial(\rho u)}{\partial t} + \frac{\partial J_{ux}}{\partial x} + \frac{\partial J_{uy}}{\partial y} = -\frac{\partial p}{\partial x} + S_u \tag{2.7}$$

$$\frac{\partial(\rho v)}{\partial t} + \frac{\partial J_{vx}}{\partial x} + \frac{\partial J_{vy}}{\partial y} = -\frac{\partial p}{\partial y} + S_v \ , \tag{2.8}$$

where J is the total (convective plus diffusive) flux in the x and y directions for the u and v momentum equations.

$$J_{ux} = \rho u u - \mu \frac{\partial u}{\partial x} \qquad\qquad J_{uy} = \rho v u - \mu \frac{\partial u}{\partial y} \tag{2.9}$$

$$J_{vx} = \rho u v - \mu \frac{\partial v}{\partial x} \qquad\qquad J_{vy} = \rho v v - \mu \frac{\partial v}{\partial y} \tag{2.10}$$

The integration of the momentum equations over the control volume leads to the semi-discrete equations

$$\frac{d(\rho u)_P}{dt} \Delta\forall + (J_{u-e} - J_{u-w}) + (J_{u-n} - J_{u-s}) = -(p_e - p_w)\,\Delta y + S_u \Delta\forall \tag{2.11}$$

$$\frac{d(\rho v)_P}{dt} \Delta\forall + (J_{v-e} - J_{v-w}) + (J_{v-n} - J_{v-s}) = -(p_n - p_s)\,\Delta x + S_v \Delta\forall \ , \tag{2.12}$$

where $\Delta\forall = \Delta x \Delta y$ is the volume of the control volume of interest, and the flux values ($J_{u,v-e,w,n,s}$) are integrated over the faces and contain the face area. Here the pressure derivative is calculated using a central difference, $(\frac{dp}{dx})_P = (p_e - p_w)/\Delta x$ and $(\frac{dp}{dy})_P = (p_n - p_s)/\Delta y$.

The method used to evaluate the fluxes J at the control volume faces becomes an important factor for the stability and accuracy of the system of equations. Different methods to evaluate this flux are examined in Sec. 4. For now the Power Law scheme of Patankar [5] is used.

$$J_{\phi-e} = a_E(\phi_P - \phi_E) + F_e\phi_P \qquad J_{\phi-w} = a_W(\phi_W - \phi_P) + F_w\phi_P \tag{2.13}$$

$$J_{\phi-n} = a_N(\phi_P - \phi_N) + F_n\phi_P \qquad J_{\phi-s} = a_S(\phi_S - \phi_P) + F_s\phi_P \tag{2.14}$$

Here ϕ represents either u or v, depending on the momentum equation component of interest. The coefficients a are found by

$$a_E = D_e A(|P_e|) + [\![-F_e, 0]\!] \qquad a_W = D_w A(|P_w|) + [\![F_w, 0]\!] \tag{2.15}$$

$$a_N = D_n A(|P_n|) + [\![-F_n, 0]\!] \qquad a_S = D_s A(|P_s|) + [\![F_s 0]\!] \ , \tag{2.16}$$

where F is the convective flux defined in Eqs. 2.5 and 2.6. The operator $[\![a, b]\!]$ returns the maximum value of a and b. The D values are diffusive coefficients defined by

$$D_e = \frac{\mu_e \Delta y}{\delta x_e} \qquad D_w = \frac{\mu_w \Delta y}{\delta x_w} \tag{2.17}$$

$$D_n = \frac{\mu_n \Delta x}{\delta y_n} \qquad D_s = \frac{\mu_s \Delta x}{\delta y_s} \ , \tag{2.18}$$

where δx and δy represent the distance between two cell centers (i.e. $\delta x_e = x_E - x_P$ and $\delta y_s = y_P - y_S$). The Peclet number $P = F/D$ is a non-dimensional number that represents the local grid Reynolds number. The function $A(|P|)$ takes different values depending on the scheme used. For Power Law,

$$A(|P|) = [\![0, (1 - 0.1|P|)^5]\!] \ . \tag{2.19}$$

Substituting Eqs. 2.13 and 2.14 into Eqs. 2.11 and 2.12, using the appropriate values of u and v in place of ϕ, leads to the following representations of the momentum equations.

$$\frac{d(\rho u)_P}{dt}\Delta\forall + (a_{u-E} + a_{u-W} + a_{u-N} + a_{u-S})\, u_P - a_{u-E}\, u_E - a_{u-W}\, u_W - a_{u-N}\, u_N - a_{u-S}\, u_S$$
$$+ (F_e - F_w + F_n - F_s)u_P = -(p_e - p_w)\,\Delta y + S_u\Delta\forall \tag{2.20}$$

$$\frac{d(\rho v)_P}{dt}\Delta\forall + (a_{v-E} + a_{v-W} + a_{v-N} + a_{v-S})\, v_P - a_{v-E}\, v_E - a_{v-W}\, v_W - a_{v-N}\, v_N - a_{v-S}\, v_S$$
$$+ (F_e - F_w + F_n - F_s)v_P = -(p_n - p_s)\,\Delta x + S_v\Delta\forall \tag{2.21}$$

By recognizing that $(F_e - F_w + F_n - F_s) = 0$ from continuity (Eq. 2.4) and defining

$$a_{u-P} = a_{u-E} + a_{u-W} + a_{u-N} + a_{u-S} \qquad b_u = (p_w - p_e)\Delta y + S_u \Delta\forall \tag{2.22}$$

$$a_{v-P} = a_{v-E} + a_{v-W} + a_{v-N} + a_{v-S} \qquad b_v = (p_s - p_n)\Delta x + S_v \Delta\forall \ , \tag{2.23}$$

the momentum equations are cast into the final semi-discrete form

$$\begin{aligned}\frac{du_P}{dt}\rho\Delta\forall &= a_{u-E}\,u_E + a_{u-W}\,u_W + a_{u-N}\,u_N + a_{u-S}\,u_S + b_u - a_{u-P}\,u_P \\ &= \sum a_{u-nb}\,u_{nb} + b_u - a_{u-P}\,u_P \qquad (2.24)\\ \frac{dv_P}{dt}\rho\Delta\forall &= a_{v-E}\,v_E + a_{v-W}\,v_W + a_{v-N}\,v_N + a_{v-S}\,v_S + b_v - a_{v-P}\,v_P \\ &= \sum a_{v-nb}\,v_{nb} + b_v - a_{v-P}\,v_P \ , \qquad (2.25)\end{aligned}$$

where the summation is over all the neighboring nodes (E, W, N, S) that share a face with the main control volume P. Equations 2.4, 2.24, and 2.25 represent the semi-discrete equations that must be satisfied to obtain a solution for unsteady incompressible flows. Before going into the different algorithms used to solve these equations, methods used to integrate the momentum equations in time are examined.

2.2 Time Integration Methods

When solving the unsteady incompressible Navier-Stokes equations, an important aspect is the integration in time, or the method by which the solution is marched forward in time. These methods are classified as explicit or implicit. Explicit methods only require information that is already known to advance while implicit methods rely on information that is not yet known, resulting in a system of equations that require more computation time to solve. The advantage of implicit methods is the ability to be stable and yield accurate results for large time step sizes. Explicit methods, on the other hand, result in poor stability, and often require small time step sizes to converge to an accurate solution.

The semi-discrete equations (discretized in space) have the following form.

$$\frac{d\phi}{dt} = F(\phi, t) \tag{2.26}$$

There are many different methods to integrate this type of equation in time. Some of the different methods are discussed in this section.

2.2.1 Single Step Methods

Single step methods integrate from one time level to the next using only one step. Three methods that fall into this category are the Euler Explicit (or forward Euler), Crank-Nicolson, and the Euler Implicit (or backward Euler or Fully Implicit) methods.

The Euler Explicit method integrates Eq. 2.26 from time t^n to time $t^{n+1} = t^n + \Delta t$ by

$$\phi^{n+1} = \phi^n + \Delta t F(\phi^n, t^n) \ , \tag{2.27}$$

where the superscript of ϕ represents the time level (i.e. $\phi^n = \phi(t^n)$). The Euler Explicit method is very simple to evaluate, given values of t^n and u^n and the function F, the update only requires the evaluation of the function F.

The Euler Implicit method integrates Eq. 2.26 by

$$\phi^{n+1} = \phi^n + \Delta t F(\phi^{n+1}, t^{n+1}) \ . \tag{2.28}$$

The only difference between Euler Explicit and Euler Implicit is the values input into the function F. For Euler Implicit, the value of ϕ^{n+1} is unknown and present on both the left hand side and right hand side of the equation. Solving this implicit equation for a scalar problem is still relatively simple: substitute ϕ^{n+1} into the function F, linearize the function F if it is non-linear in ϕ, and solve for ϕ^{n+1}. For a coupled set of functions, solving the implicit equation requires the solution of a linear system of equations, which becomes more costly as the system of equations becomes larger.

The final single step method of interest is the Crank-Nicolson method, which is equivalent to the trapezoidal rule, and is the average of Euler Explicit and Euler Implicit.

$$\phi^{n+1} = \phi^n + \frac{1}{2}\left[\Delta t F(\phi^n, t^n) \Delta t F(\phi^{n+1}, t^{n+1})\right] \tag{2.29}$$

The Crank-Nicolson method requires a similar solution process as the Euler Implicit, including the solution of a linear system of equations for a coupled set of functions. Both the Euler Explicit and Euler Implicit methods are first order accurate in time, while Crank-Nicolson is second order accurate in time. All three of these methods are written together as

$$\phi^{n+1} = \phi^n + \alpha_t \left[\Delta t F(\phi^n, t^n)\right] + (1 - \alpha_t) \left[\Delta t F(\phi^{n+1}, t^{n+1})\right] \ , \tag{2.30}$$

where α_t equals 1.0 for Euler Implicit, 0.5 for Crank-Nicolson, and 0.0 for Euler Explicit.

2.2.2 Runge-Kutta Methods

Runge-Kutta (RK) methods use several steps or stages between time level n and $n + 1$ to integrate Eq. 2.26. RK methods involve a weighted average of the function $F(u, t)$ evaluated at the different stages. The generic form for any RK integration of Eq. 2.26 from time level n to time level $n + 1$ has the form

$$\phi^{n+1} = \phi^n + \Delta t \sum_{s=1}^{S} \beta_s F\big(\phi_s, t^n + \gamma_s \Delta t\big) \ , \tag{2.31}$$

where S is the number of RK stages and ϕ_s are the stages values. The β_s coefficient is the weight of the function F for a given stage s. The coefficient γ_s represents a fraction of the time step, where the time for stage s is $t_s = t^n + \gamma_s \Delta t$. To evaluate Eq. 2.31, the values of ϕ_s at each stage $(1 \le s \le S)$ are required. The equation for each stage has the form

$$\phi_s = \phi^n + \Delta t \sum_{l=1}^{S} \alpha_{s,l} F\big(\phi_l, t^n + \gamma_l \Delta t\big) \qquad \text{for } 1 \le s \le S \ . \tag{2.32}$$

For each stage s there are a set of weights $\alpha_{s,l}$, where the index l is a free index. There are a set of conditions on β and γ.

$$\sum_{s=1}^{S} \beta_s = 1 \tag{2.33}$$

$$\gamma_s = \sum_{l=1}^{S} \alpha_{s,l} \qquad \text{for } 1 \le s \le S \tag{2.34}$$

The first condition (Eq. 2.33) requires the sum of the weights to equal 1. The second condition (Eq. 2.34) requires that at each stage, the sum of the stage weights equals the time fraction for that stage, γ_s. The values of the coefficients α, β, and γ determine the specific RK method and are often visualized in a Butcher tableau, as shown in Table 2.1. Many RK methods are developed with coefficients set to satisfy accuracy and stability conditions [24].

An important point to observe is the function F can include source terms that are functions of time. The way that source terms are evaluated has a large impact on the accuracy of the integration. Appendix A shows source terms should be evaluated at each stage with the given stage time (t_s) to yield the most accurate results.

Table 2.1: Form of the Butcher tableau.

γ_1	$\alpha_{1,1}$	$\alpha_{1,2}$	$\cdots$	$\alpha_{1,l}$	$\cdots$	$\alpha_{1,S}$
γ_2	$\alpha_{2,1}$	$\alpha_{2,2}$	$\cdots$	$\alpha_{2,l}$	$\cdots$	$\alpha_{2,S}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
γ_s	$\alpha_{s,1}$	$\alpha_{s,2}$	$\cdots$	$\alpha_{s,l}$	$\cdots$	$\alpha_{s,S}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
γ_S	$\alpha_{S,1}$	$\alpha_{S,2}$	$\cdots$	$\alpha_{S,l}$	$\cdots$	$\alpha_{S,S}$
	β_1	β_2	$\cdots$	β_l	$\cdots$	β_S

2.2.2.1 Fully Implicit Runge-Kutta

Looking again at Eq. 2.32, one notices that to solve any stage value (ϕ_s), all other stage values must also be known and substituted into the function for F. This is the most general form of RK, without assuming or forcing any coefficient to be zero, and is called Fully Implicit Runge-Kutta (FIRK). FIRK methods have every stage coupled together, so all stages must be solved simultaneously. For a coupled set of F functions with each ϕ in the set related to all other ϕ values in the set, FIRK integration requires a system of equations be solved with a dimension $(N * S)^2$, where N is the number of ϕ values in the set (for our case of incompressible flow N is the number of grid points in the domain). The advantages of FIRK methods are the stability and high order accuracy with only a few stages. FIRK methods based on Gauss quadrature are able to achieve an order of accuracy of $p = 2 * S$ [24]. Despite the advantages of FIRK methods, the solution of the large system often requires so much work that the method is unreasonable (although with recent advancements in numerical methods, research shows that FIRK methods can be competitive [48]).

To reduce the complexity of solving FIRK methods, several variations of RK are developed that reduce the number of non-zero coefficients in the Butcher Tableau. A RK method is implicit if it has at least one non-zero $\alpha_{s,l}$ on or above the diagonal. Put another way: for any $1 \leq s \leq S$, if $\alpha_{s,l} \neq 0$ for any $l \geq s$, then the RK method is implicit. If this is not true, ($\alpha_{s,l} = 0$ for all $l \geq s$), then the RK method is explicit.

2.2.2.2 Explicit Runge-Kutta

A generic explicit Runge-Kutta (ERK) method has the stage values of the form

$$\phi_s = \phi^n + \Delta t \sum_{l=1}^{s-1} \alpha_{s,l} F\big(\phi_l, t^n + \gamma_l \Delta t\big) \qquad \text{for } 1 \leq s \leq S \ . \tag{2.35}$$

The stage value for the first stage simplifies to $\phi_1 = \phi^n$, and Eq. 2.31 is used to update to the next time level. The coefficients $\alpha_{l,s}$ are only non-zero for $l < s$. Note the only difference between Eq. 2.35 and Eq. 2.32 is the index on the summation. ERK methods only rely on previous stages (previously known info), but implicit Runge-Kutta (IRK) methods rely on at least one stage value not yet known (info from past, present, and future).

Integrating with ERK methods results in each stage becoming an algebraic equation that requires little computational effort, evaluation of the function F with known values. The disadvantages of ERK methods are poor stability (requiring low time step sizes) and the high number of stages required to achieve higher order of accuracy. It is possible to develop an ERK method with an order of accuracy equal to the number of stages ($p = S$) up to fourth order ($p = 4$), but to get an ERK method with $p \geq 5$, then $S > p$ [24].

2.2.2.3 Low-Storage Explicit Runge-Kutta

A variation on the ERK method is the low-storage ERK (ERK-LS) method, in which each stage value is calculated with only the last stage value [49]. For ERK-LS methods, the coefficients simplify to $\alpha_{s,l} = 0$ for all $l \neq (s-1)$. The simplified equation to find each stage value is as follows

$$\phi_s = \phi^n + \alpha_{s,s-1} \Delta t F\big(\phi_{s-1}, t^n + \gamma_{s-1} \Delta t\big) \qquad \text{for } 1 \leq s \leq S \ , \tag{2.36}$$

and the update to time level $n+1$ is

$$\phi^{n+1} = \phi^n + \beta_S \Delta t F\big(\phi_S, t^n + \gamma_S \Delta t\big) \ . \tag{2.37}$$

The coefficients $\alpha_{l,s}$ are only non-zero for $l = (s-1)$. The ERK-LS methods only require storage of ϕ^n and ϕ_{s-1} to find ϕ_s, reducing the memory required compared to generic ERK methods. The simplified equations also reduce the number of computations required to update each value. The commonly used RK4 method falls into this category.

2.2.2.4 Diagonally Implicit Runge-Kutta

The FIRK methods are known for high accuracy and stability, but are costly for large systems. ERK methods require less computation to update each stage value, but suffer from poor stability and require a large number of stages to achieve a high order of accuracy. A variation of IRK methods, known as Diagonally Implicit Runge-Kutta (DIRK), maintains some of the advantages of FIRK methods (namely high stability), while allowing each stage to be solved sequentially with a reduced computational cost. The stage equations for DIRK are

$$\phi_s = \phi^n + \Delta t \sum_{l=1}^{s} \alpha_{s,l} F\big(\phi_l, t^n + \gamma_l \Delta t\big) \qquad \text{for } 1 \leq s \leq S \ . \tag{2.38}$$

Equation 2.31 is used to update to the next time level. The coefficients $\alpha_{l,s}$ are non-zero for $l \leq s$ and zero for $l > s$. The difference in the stage equation is on the summation, only going from $l = 1$ to $l = s$. This means that to solve each stage s, the stages values for all past stages $1 \leq l \leq s-1$ and the current stage s must be known, but future stages are not required.

Using DIRK methods to solve a set of size N requires a linear system of equations of size N^2 be solved at each stage ($1 \leq s \leq S$), or a total of S linear systems of size N^2. Solving several smaller systems of equations is often more efficient than solving one large system of equations, allowing DIRK methods to be more efficient than FIRK methods at integrating over a time step.

The diagonal coefficient, $\alpha_{s,s}$, in DIRK methods is sometimes taken to be a constant ($\alpha_{s,s} = k$ for all $1 \leq s \leq S$) [24],[50]. Some researchers call these methods Singly Diagonal Implicit Runge-Kutta (SDIRK), while others simply call them DIRK methods. One advantage of the singly diagonal term is that if the function F is either linear in ϕ or is linearized once for all stages, then the coefficient matrix formed to solve the linear system of equations does not change from stage to stage. This allows for factorization to improve the efficiency of solving the system of equations. In present research these are called SDIRK methods, and they have stage values

$$\phi_s = \phi^n + \Delta t k F\big(\phi_s, t^n + \gamma_s \Delta t\big) + \Delta t \sum_{l=1}^{s-1} \alpha_{s,l} F\big(\phi_l, t^n + \gamma_l \Delta t\big) \qquad \text{for } 1 \leq s \leq S\ . \tag{2.39}$$

The order of accuracy of DIRK methods is not as good as the FIRK methods, which can achieve $p = 2 * s$, but DIRK methods can achieve $p = s + 1$ for order of accuracy up to $p = 4$ with $s = 3$ [50].

2.2.2.5 Explicit first stage, Diagonally Implicit Runge-Kutta

Another variation of IRK similar to DIRK methods are Explicit first stage, Diagonally Implicit Runge-Kutta (EDIRK) methods [51].These methods are diagonally implicit methods with the first stage explicit, requiring the solution of $S - 1$ implicit stages and reducing the computational cost. The stage values for EDIRK methods are

$$\phi_1 = \phi^n \tag{2.40}$$

$$\phi_s = \phi^n + \Delta t \sum_{l=1}^{s} \alpha_{s,l} F\big(\phi_l, t^n + \gamma_l \Delta t\big) \qquad \text{for } 2 \leq s \leq S\ . \tag{2.41}$$

Similar to the SDIRK methods, there are Explicit first stage, Singly Diagonal Implicit Runge-Kutta (ESDIRK) methods in which EDIRK methods have a singly diagonal term $\alpha_{s,s} = k$. Like the SDIRK methods, this allows for the possibility of factorizing the coefficient matrix and reducing the computational cost of solving all stages. The stage values for ESDIRK methods are

$$\phi_1 = \phi^n \tag{2.42}$$

$$\phi_s = \phi^n + \Delta t k F\big(\phi_s, t^n + \gamma_s \Delta t\big) + \Delta t \sum_{l=1}^{s-1} \alpha_{s,l} F\big(\phi_l, t^n + \gamma_l \Delta t\big) \qquad \text{for } 2 \leq s \leq S\ . \tag{2.43}$$

2.2.2.6 Stiffly Accurate Runge-Kutta Methods

A Runge-Kutta method is stiffly accurate if the last stage value is equal to the value at time level $(n+1)$, or $\alpha_{S,l} = \beta_l$ for all $1 \leq l \leq S$ [50]. These methods are particularly useful for DAE systems (such as the incompressible Navier-Stokes equations). RK methods that are not stiffly accurate tend to suffer order reduction (in the case of incompressible flow, order reduction in pressure) [51].When the stiffly accurate condition is paired with either ERK or EDIRK/ESDIRK, then the first-same-as-last (FSAL) property is met [24].The FSAL property acts as if the RK method is one stage less than it actually is, and is often more efficient than methods without the FSAL property [24].

The differences in RK methods are be seen qualitatively in the representations of the Butcher tableau for these RK methods in Fig. 2.2.

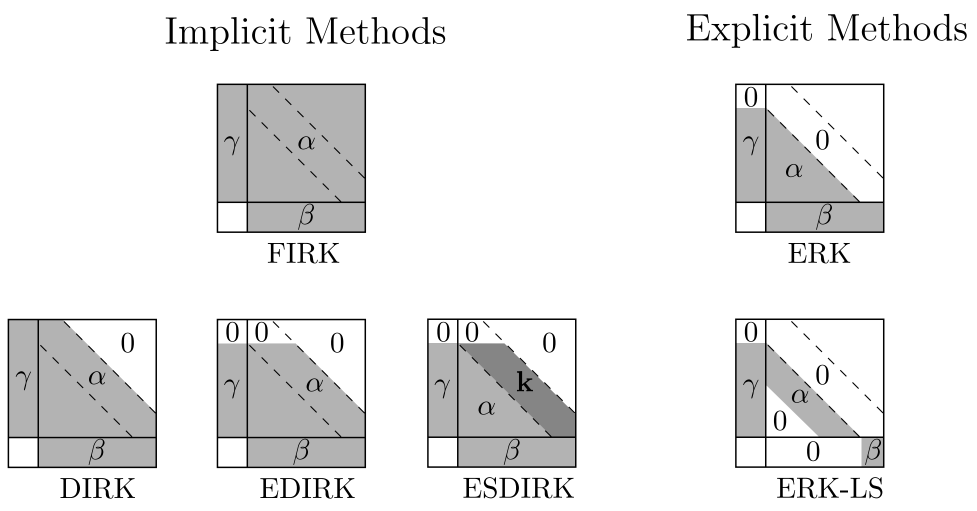


Figure 2.2: Qualitative representation of Butcher tableaus for Runge-Kutta methods.

2.3 Algorithms

With methods to integrate the momentum and continuity equations available, the process of solving the coupled equations is now discussed. Recalling the set of equations for a two-dimensional

Cartesian grid have the following form.

$$0 = (\rho u \Delta y)_e - (\rho u \Delta y)_w + (\rho v \Delta x)_n - (\rho v \Delta x)_s \tag{2.44}$$

$$\frac{du_P}{dt} \rho \Delta \forall = \sum a_{u-nb}\, u_{nb} + b_u - a_{u-P}\, u_P \tag{2.45}$$

$$\frac{dv_P}{dt} \rho \Delta \forall = \sum a_{v-nb}\, v_{nb} + b_v - a_{v-P}\, v_P \tag{2.46}$$

Equations in this form are called differential algebraic equations (DAEs) of type 2 [1] because there is an algebraic equation as well as differential equations coupled together. DAEs of type 2 are often written as

$$\frac{d\vec{V}}{dt} = f(\vec{V}, p) \tag{2.47}$$

$$0 = g(\vec{V}) \ , \tag{2.48}$$

where the differential equations are the momentum equations with pressure on the right-hand-side, and the algebraic equation is continuity.

An important driving force for fluid flow is pressure; however, it is only present in the source terms of the momentum equations (b_u and b_v). There are many algorithms discussed in Section 1.2.1 that have been developed to solve these equations. The present research focuses on pressure-based methods, and in particular, starts by examining the SIMPLE family of methods.

2.3.1 SIMPLE

The SIMPLE (Semi-IMplicit Pressure Linked Equations) algorithm of Patankar [5] was developed by creating a pressure correction equation that allowed both pressure and velocity to be corrected to satisfy continuity. The pressure correction equation starts with the fully discrete momentum equation integrated in time with an implicit method. Eqs. 2.45 and 2.46 are integrated from time t^n to t^{n+1} using Crank-Nicolson ($\alpha_t = 0.5$) or Fully Implicit ($\alpha_t = 1.0$) for all terms except pressure, which, by choice, is always integrated with the Fully Implicit scheme.

$$\begin{aligned} u_P^{n+1} &= u_P^n + \alpha_t \frac{\Delta t}{\rho \Delta \forall}\left(\sum a_{u-nb}\, u_{nb} + S_u \Delta \forall - a_{u-P}\, u_P\right)^{n+1} \\ &+(1-\alpha_t)\frac{\Delta t}{\rho \Delta \forall}\left(\sum a_{u-nb}\, u_{nb} + S_u \Delta \forall - a_{u-P}\, u_P\right)^{n} + \frac{\Delta t}{\rho \Delta \forall}\left(p_w - p_e\right)^{n+1} \Delta y \end{aligned} \tag{2.49}$$

$$\begin{aligned} v_P^{n+1} &= v_P^n + \alpha_t \frac{\Delta t}{\rho \Delta \forall}\left(\sum a_{v-nb}\, v_{nb} + S_v \Delta \forall - a_{v-P}\, v_P\right)^{n+1} \\ &+(1-\alpha_t)\frac{\Delta t}{\rho \Delta \forall}\left(\sum a_{v-nb}\, v_{nb} + S_v \Delta \forall - a_{v-P}\, v_P\right)^{n} + \frac{\Delta t}{\rho \Delta \forall}\left(p_s - p_n\right)^{n+1} \Delta x \end{aligned} \tag{2.50}$$

Rearranging leads to

$$a'_{u-P}\, u_P^{n+1} = \sum a'_{u-nb}\, u_{nb}^{n+1} + b'_u + \left(p_w - p_e\right)^{n+1} \Delta y \tag{2.51}$$

$$a'_{v-P}\, v_P^{n+1} = \sum a'_{v-nb}\, v_{nb}^{n+1} + b'_v + \left(p_s - p_n\right)^{n+1} \Delta x \ , \tag{2.52}$$

with

$$a'_{u-P} = \alpha_t\, a_{u-P}^{n+1} + \frac{\rho \Delta \forall}{\Delta t} \tag{2.53}$$

$$a'_{u-nb} = \alpha_t\, a_{u-nb}^{n+1} \tag{2.54}$$

$$b'_u = (1-\alpha_t)\left(\sum a_{u-nb}\, u_{nb} + S_u \Delta \forall - a_{u-P}\, u_P\right)^{n} + \frac{\rho \Delta \forall}{\Delta t} u_P^n \tag{2.55}$$

$$a'_{v-P} = \alpha_t\, a_{v-P}^{n+1} + \frac{\rho \Delta \forall}{\Delta t} \tag{2.56}$$

$$a'_{v-nb} = \alpha_t\, a_{v-nb}^{n+1} \tag{2.57}$$

$$b'_v = (1-\alpha_t)\left(\sum a_{v-nb}\, v_{nb} + S_v \Delta \forall - a_{v-P}\, v_P\right)^{n} + \frac{\rho \Delta \forall}{\Delta t} v_P^n \ . \tag{2.58}$$

Equations 2.51 and 2.52 are the fully discrete momentum equations that are solved to find the updated velocity components at time t^{n+1}. If the pressure field is known, this equation is solved using any number of linear equation solvers (i.e. Gauss-Seidel). However, the pressure is not typically known when solving fluid flow, so a guessed value for pressure must be used to solve the momentum equations.

2.3.1.1 Pressure Correction Equation

The pressure correction equation was derived to have a mechanism to correct pressure after updating the velocity field. Recall for a staggered grid, the pressure is located at the cell center (P), and velocity components are at the cell faces (e, w, n, s). For convenience of notation, momentum equations are written with the P taken as the location of the velocity component. If we instead write the momentum equations centered at face e for u and face n for v, we get

$$a_{u-e}\, u_e = \sum a_{u-nb}\, u_{nb} + b_u + (p_P - p_E)\, \Delta y \tag{2.59}$$

$$a_{v-n}\, v_n = \sum a_{v-nb}\, v_{nb} + b_v + (p_P - p_N)\, \Delta x \ , \tag{2.60}$$

noting the change in pressure term subscripts. For ease of notation, the primes have been removed from the coefficients and source term and the superscript $n+1$ is dropped. All values are assumed to be from the $n+1$ time level unless otherwise noted.

Following Patankar's procedure for pressure correction, if a guessed pressure field, p^*, is used to solve the momentum equations, an imperfect velocity field, u^* and v^*, are found.

$$a_{u-e}\, u_e^* = \sum a_{u-nb}\, u_{nb}^* + b_u + (p_P^* - p_E^*)\, \Delta y \tag{2.61}$$

$$a_{v-n}\, v_n^* = \sum a_{v-nb}\, v_{nb}^* + b_v + (p_P^* - p_N^*)\, \Delta x \tag{2.62}$$

To improve the pressure field, a pressure correction, p', is defined such that the corrected pressure is defined by

$$p = p^* + p' \ . \tag{2.63}$$

Correcting the pressure results in a corrected velocity field, denoted as

$$u = u^* + u' \qquad\qquad v = v^* + v' \ . \tag{2.64}$$

Subtracting Eq. 2.61 from Eq. 2.59, as well as Subtracting Eq. 2.62 from Eq. 2.60, yields the following.

$$a_{u-e}\, u_e' = \sum a_{u-nb}\, u_{nb}' + \left(p_P' - p_E'\right) \Delta y \tag{2.65}$$

$$a_{v-n}\, v_n' = \sum a_{v-nb}\, v_{nb}' + \left(p_P' - p_N'\right) \Delta x \tag{2.66}$$

If the approximate values are correct, then the values of p', u', and v' are zero. Patankar drops the terms $\sum a_{u-nb}\, u'_{nb}$ and $\sum a_{v-nb}\, v'_{nb}$, leading to the following equations.

$$a_{u-e}\, u'_e = \left(p'_P - p'_E\right) \Delta y \tag{2.67}$$

$$a_{v-n}\, v'_n = \left(p'_P - p'_N\right) \Delta x \tag{2.68}$$

Rearranging leads to

$$u'_e = d_u \left(p'_P - p'_E\right) \qquad\qquad v'_n = d_v \left(p'_P - p'_N\right) \ , \tag{2.69}$$

where

$$d_u = \frac{\Delta y}{a'_{u-e}} \qquad\qquad d_v = \frac{\Delta x}{a'_{v-e}} \ . \tag{2.70}$$

Substituting Eq. 2.69 into Eq. 2.64 leads to

$$u_e = u^*_e + d_u \left(p'_P - p'_E\right) \qquad\qquad v_n = v^*_n + d_v \left(p'_P - p'_N\right) \ . \tag{2.71}$$

Equation 2.71 is the equation used to correct velocity once the pressure correction is known. To find the pressure correction, Eq. 2.71 is substituted into the continuity equation (Eq. 2.44) at all control volume faces.

$$a_{p-P}\, p'_P = a_{p-E}\, p'_E + a_{p-W}\, p'_W + a_{p-N}\, p'_N + a_{p-S}\, p'_S + bp' \ , \tag{2.72}$$

where

$$a_{p-E} = \left(\rho\, d_u\right)_e \Delta y \qquad\qquad a_{p-W} = \left(\rho\, d_u\right)_w \Delta y \tag{2.73}$$

$$a_{p-N} = \left(\rho\, d_v\right)_n \Delta x \qquad\qquad a_{p-S} = \left(\rho\, d_v\right)_s \Delta x \tag{2.74}$$

$$a_{p-P} = a_{p-E} + a_{p-W} + a_{p-N} + a_{p-S} \tag{2.75}$$

$$b'_p = (\rho u^* \Delta y)_e - (\rho u^* \Delta y)_w + (\rho v^* \Delta x)_n - (\rho v^* \Delta x)_s \ . \tag{2.76}$$

2.3.1.2 Relaxation

The SIMPLE algorithm converges to steady state solutions using sub-iterations alongside relaxation of the momentum equation, and pressure correction is used to correct both the pressure and velocity. For momentum equations, relaxation is included by modifying the coefficients in Eqs. 2.53-2.58, with a relaxation parameter α_u and α_v for the x and y momentum equations respectively.

$$a'_{u-P} = \frac{a'_{u-P}}{\alpha_u} \qquad b'_u = b'_u + (1-\alpha_u)\frac{a'_{u-P}}{\alpha_u}u^*_P \tag{2.77}$$

$$a'_{v-P} = \frac{a'_{v-P}}{\alpha_v} \qquad b'_v = b'_v + (1-\alpha_v)\frac{a'_{v-P}}{\alpha_v}v^*_P\ , \tag{2.78}$$

where u^*_P and v^*_P are the last sub-iteration values. For the pressure correction equation, relaxation is added by limiting the correction to pressure.

$$p = p^* + \alpha_p p' \tag{2.79}$$

Patankar quotes values of $\alpha_u = \alpha_v = 0.5$ and $\alpha_p = 0.8$ work for many problems, but some cases require lower values. These relaxation parameters heavily influence the convergence rate and runtime of the algorithm.

For unsteady problems, the solution is converged using this relaxation and sub-iterations each time step.

2.3.1.3 Solution Procedure

The solution procedure for the SIMPLE algorithm is as follows:

1. Start with an initial velocity field and a guess for an initial pressure field.
2. Calculate the momentum equation coefficients (Eqs. 2.53-2.58) using the most up to date information (including relaxation).
3. Solve the momentum equations (Eqs. 2.61 and 2.62) for an updated velocity field.
4. Calculate the pressure correction coefficients (Eqs. 2.73-2.76) using the updated velocity field.
5. Solve the pressure correction equation (Eq. 2.72).
6. Correct pressure using Eq. 2.63 (including relaxation).
7. Correct velocity field using Eq. 2.64.

8. Return to step 2, treating the updated pressure field as the new guessed value and iterate until convergence.

9. Advance in time ($t = t + \Delta t$) and go to step 2, using the converged pressure field as the guess for the next time step.

This procedure is shown visually in Fig. 2.3.

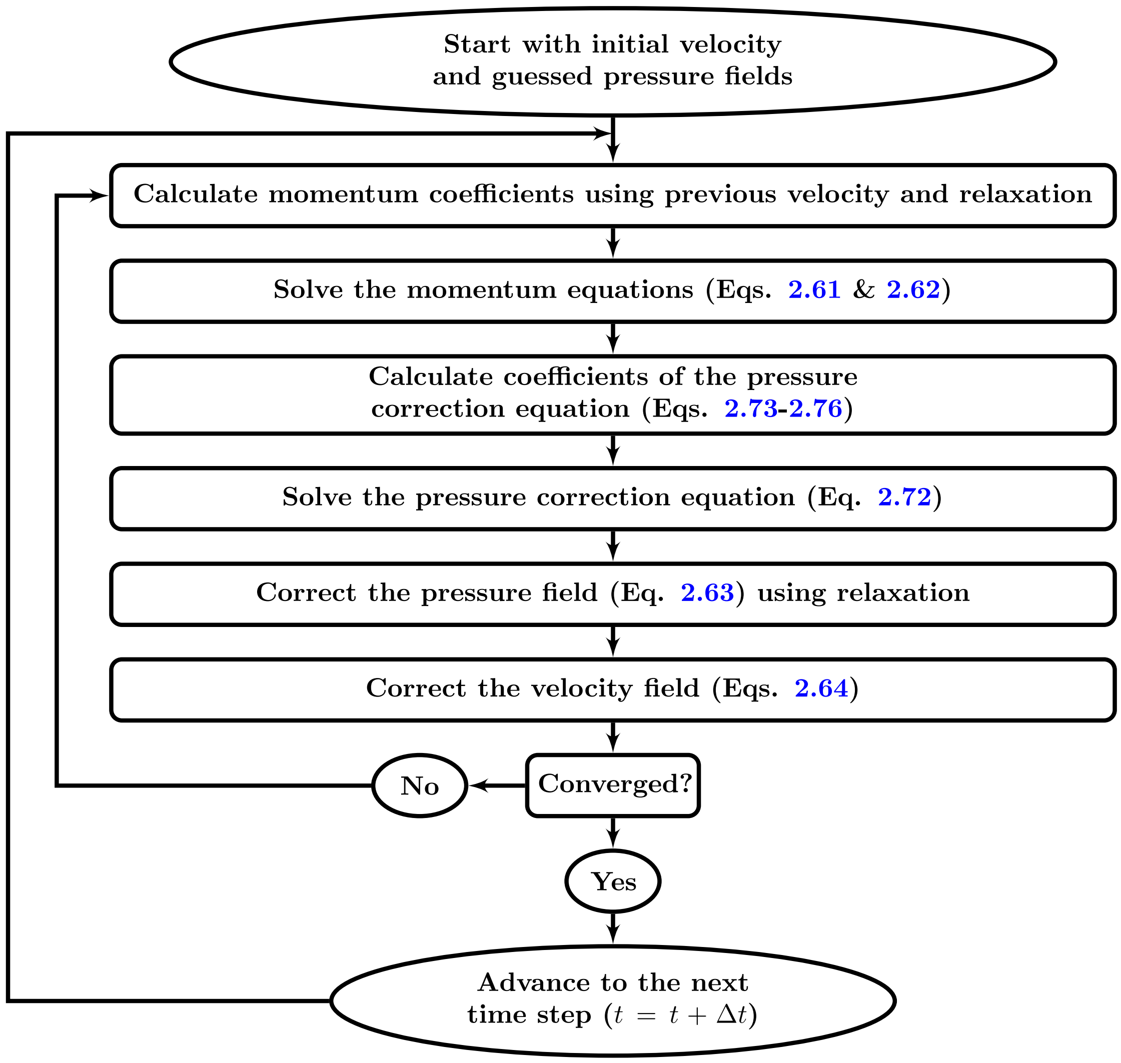

Start with initial velocity and guessed pressure fields
Calculate momentum coefficients using previous velocity and relaxation
Solve the momentum equations (Eqs. 2.61 & 2.62)
Calculate coefficients of the pressure correction equation (Eqs. 2.73-2.76)
Solve the pressure correction equation (Eq. 2.72)
Correct the pressure field (Eq. 2.63) using relaxation
Correct the velocity field (Eqs. 2.64)
No
Converged?
Yes
Advance to the next time step ($t = t + \Delta t$)

Figure 2.3: Diagram of the SIMPLE algorithm.

2.3.2 SIMPLER

The SIMPLE algorithm has preformed well for many problems, but the SIMPLE Revised (SIMPLER) algorithm is developed to improve the convergence rate and reduce runtime. The main motivation leading to the development of SIMPLER is the approximation made in deriving the pressure correction equation, which leads to the requirement of relaxation in the pressure correction.

2.3.2.1 Pressure Equation

The SIMPLER algorithm starts by deriving an equation to solve pressure from a given velocity field. Writing the momentum equations (Eqs. 2.59 and 2.60) slightly differently

$$u_e^{n+1} = \hat{u} + d_u \left(p_P - p_E\right)^{n+1} \tag{2.80}$$

$$v_n^{n+1} = \hat{v} + d_v \left(p_P - p_N\right)^{n+1} \ , \tag{2.81}$$

where the pseudo-velocities ($\hat{u}$ and $\hat{v}$) are defined by

$$\hat{u} = \frac{\sum a'_{u-nb}\, u_{nb}^{n+1} + b'_u}{a'_{u-e}} \qquad \hat{v} = \frac{\sum a'_{v-nb}\, v_{nb}^{n+1} + b'_v}{a'_{v-e}} \ . \tag{2.82}$$

Substituting Eqs. 2.80 and 2.81 into the continuity equation (Eq. 2.44) at all control volume faces leads to the pressure equation.

$$a_{p-P}\, p_P = a_{p-E}\, p_E + a_{p-W}\, p_W + a_{p-N}\, p_N + a_{p-S}\, p_S + bp \ , \tag{2.83}$$

where

$$a_{p-E} = \left(\rho\, d_u\right)_e \Delta y \qquad a_{p-W} = \left(\rho\, d_u\right)_w \Delta y \tag{2.84}$$

$$a_{p-N} = \left(\rho\, d_v\right)_n \Delta x \qquad a_{p-S} = \left(\rho\, d_v\right)_s \Delta x \tag{2.85}$$

$$a_{p-P} = a_{p-E} + a_{p-W} + a_{p-N} + a_{p-S} \tag{2.86}$$

$$b_p = (\rho\hat{u}\Delta y)_e - (\rho\hat{u}\Delta y)_w + (\rho\hat{v}\Delta x)_n - (\rho\hat{v}\Delta x)_s \ . \tag{2.87}$$

The coefficients (a values) for the pressure and pressure correction equations are identical, but the source term (b) for the pressure equation uses pseudo-velocities where the pressure correction equation uses approximate velocities values.

2.3.2.2 Solution Procedure

The solution procedure for the SIMPLER algorithm is as follows:

1. Start with an initial velocity field.
2. Calculate the momentum equation coefficients (Eqs. 2.53-2.58) using previous velocity (including relaxation).
3. Calculate the pressure coefficients (Eqs. 2.84-2.87).
4. Solve the pressure equation (Eq. 2.83).
5. Solve the momentum equations (Eqs. 2.61 and 2.62) using the calculated pressure field.
6. Calculate the pressure correction source (Eq. 2.76) using the updated velocity field.
7. Solve the pressure correction equation (Eq. 2.72).
8. Correct velocity field using Eq. 2.64, but do not correct the pressure field.
9. Return to step 2 and iterate until convergence.
10. Advance in time $t = t + \Delta t$ and go to step 2.

This procedure is shown visually in Fig. 2.4.

The SIMPLER algorithm requires more computations because the additional pressure equation is solved, but the convergence rate is improved and solutions are usually found with less runtime. Two important factors are the removal of a pressure correction that requires relaxation and the fact that a guessed pressure field is not required.

For a more thorough discussion of the SIMPLE and SIMPLER algorithms refer to Patankar[5].

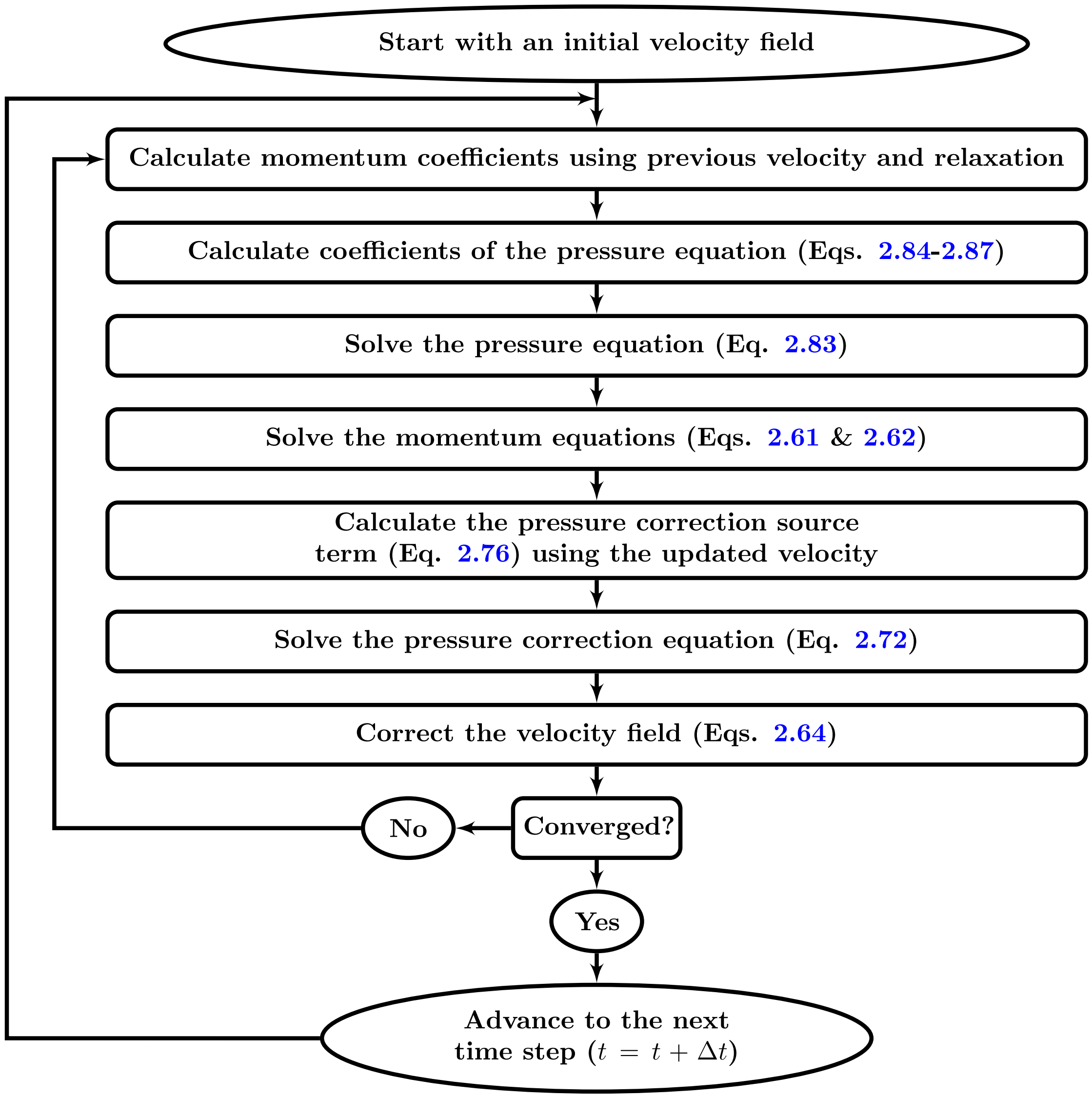

Start with an initial velocity field
Calculate momentum coefficients using previous velocity and relaxation
Calculate coefficients of the pressure equation (Eqs. 2.84-2.87)
Solve the pressure equation (Eq. 2.83)
Solve the momentum equations (Eqs. 2.61 & 2.62)
Calculate the pressure correction source term (Eq. 2.76) using the updated velocity
Solve the pressure correction equation (Eq. 2.72)
Correct the velocity field (Eqs. 2.64)
Converged?
No
Yes
Advance to the next time step ($t = t + \Delta t$)

Figure 2.4: Diagram of the SIMPLER algorithm.

2.3.3 RK-SIMPLER

The RK-SIMPLER algorithm for the Cartesian system is presented by Purohit [52] and is further discussed by Rajagopalan and Lestari [6]. The three main motivating factors for developing RK-SIMPLER are to use explicit methods to integrate the momentum equations, remove the need for relaxation and sub-iterations, and remove the approximate pressure correction equation. The original algorithm developed by Purohit is called Variation A, and a modified algorithm developed in the present work is called Variation B.

2.3.3.1 Variation A

The RK-SIMPLER algorithm Variation A, denoted as RK-SIMPLER(A), uses the pressure equation and the semi-discrete form of the momentum equations to update the solution field each time step. The initial velocity field is assumed to be known, and the pressure equation is solved to extract the pressure field. With the pressure now known, the momentum equations form first order linear ODEs in time that are solved with pressure as a source term.

The semi-discrete form of the momentum equations in Eqs. 2.45 and 2.46 are both of the form

$$\frac{d\phi}{dt} = F(\phi, t) \ , \tag{2.88}$$

and are integrated with Runge-Kutta methods. For the x and y momentum equations, the ODEs are

$$\frac{du_P}{dt} = F_u(u, t) = \frac{R_u}{\rho \Delta \forall} \tag{2.89}$$

$$\frac{dv_P}{dt} = F_v(v, t) = \frac{R_v}{\rho \Delta \forall} \ . \tag{2.90}$$

R_u and R_v are defined as

$$R_u = \sum a_{u-nb}\, u_{nb} + b_u - a_{u-P}\, u_P \tag{2.91}$$

$$R_v = \sum a_{v-nb}\, v_{nb} + b_v - a_{v-P}\, v_P \ , \tag{2.92}$$

where the coefficients and sources are the unmodified values defined in Eqs. 2.16, 2.15, 2.22, and 2.23.

To form a pressure equation, momentum equations are integrated in time with Crank-Nicolson or Fully Implicit discretization, as shown in Eqs. 2.51-2.58. Purohit originally used Fully Implicit, while Lestari [53] used a Crank-Nicolson integration for testing the RK-SIMPLER algorithm on unstructured grids. Crank-Nicolson is second order accurate and Lestari found it works well with RK-SIMPLER, so it is used in the present work. The pressure equation requires evaluating $\hat{u}$ and $\hat{v}$ in Eq. 2.82, however, the values of u_{nb}^{n+1} and v_{nb}^{n+1} are not known at the beginning of the time step. To remedy this, the known values of u_{nb}^{n} and v_{nb}^{n} are used instead.

$$\hat{u} = \frac{\sum a'_{u-nb} u_{nb}^{n} + b'_{u}}{a'_{u-P}} \qquad \hat{v} = \frac{\sum a'_{v-nb} v_{nb}^{n} + b'_{v}}{a'_{v-P}} \tag{2.93}$$

The momentum coefficients are also known at time level n, and are used in the above formulation of the pseudo-velocities instead of coefficients from time level $n+1$. By making these simplifications, the pressure and momentum equations become decoupled. The pressure equation is solved first from the known velocity field, then the momentum equations are updated using the known velocity field and pressure. These simplifications are present in the simulations of Purohit [52], Lestari [53], and Rajagopalan [6], and result in time accurate solutions.

To remove these simplifications, iterations are used, similar to SIMPLE and SIMPLER, to update the coefficients and pseudo-velocities after the momentum equations are updated. Section 2.4.2.1 shows these iterations reduce the error for large time step sizes but increase the runtime, making the algorithm less efficient. These iterations are only used in Section 2.4.2.1 and are not recommended.

The momentum equations (Eqs. 2.89 and 2.90) are integrated in time using explicit Runge-Kutta methods. Purohit uses a Low-Storage ERK method with four stages and second order accuracy from Jameson [11]. The Butcher tableau for this method (denoted ERK-LS4) is as follows.

Table 2.2: ERK-LS4.

0				
1/4	1/4			
1/3	0	1/3		
1/2	0	0	1/2	
	0	0	0	1

Referring back to the form of the momentum equations in Eqs. 2.89 and 2.90, the update procedure for velocity components using ERK-LS4 is

$$(u_P)_1 = u_P^n \qquad (v_P)_1 = v_P^n \tag{2.94}$$

$$(u_P)_2 = u_P^n + \frac{1}{4}\frac{\Delta t}{\rho \Delta \forall}(R_u)_1 \qquad (v_P)_2 = v_P^n + \frac{1}{4}\frac{\Delta t}{\rho \Delta \forall}(R_v)_1 \tag{2.95}$$

$$(u_P)_3 = u_P^n + \frac{1}{3}\frac{\Delta t}{\rho \Delta \forall}(R_u)_2 \qquad (v_P)_3 = v_P^n + \frac{1}{3}\frac{\Delta t}{\rho \Delta \forall}(R_v)_2 \tag{2.96}$$

$$(u_P)_4 = u_P^n + \frac{1}{2}\frac{\Delta t}{\rho \Delta \forall}(R_u)_3 \qquad (v_P)_4 = v_P^n + \frac{1}{2}\frac{\Delta t}{\rho \Delta \forall}(R_v)_3 \tag{2.97}$$

$$u_P^{n+1} = u_P^n + \frac{\Delta t}{\rho \Delta \forall}(R_u)_4 \qquad v_P^{n+1} = v_P^n + \frac{\Delta t}{\rho \Delta \forall}(R_v)_4 \ , \tag{2.98}$$

where $(R_u)_i$ represents the function R_u evaluated with the stage values of u_i. For this velocity update procedure the pressure and momentum coefficients are constant, and the only values changing are the velocities.

2.3.3.2 RK-SIMPLER(A) Solution Procedure

The RK-SIMPLER(A) algorithm solves the decoupled pressure equation and momentum equations sequentially, and does not require sub-iterations or relaxation. The RK-SIMPLER(A) solution procedure is as follows.

1. Start with an initial velocity field.
2. Calculate the momentum equation coefficients (Eqs. 2.53-2.58) using previous velocity field.
3. Calculate the pressure coefficients (Eqs. 2.84-2.87) using the pseudo-velocities defined in Eq. 2.93.

4. Solve the pressure equation (Eq. 2.83).
5. Update the velocity field using explicit Runge-Kutta methods with the calculated pressure field as a source (Eqs. 2.94-2.98).
6. Advance in time $t = t + \Delta t$ and go to step 2.

No correction equations are used in the RK-SIMPLER(A) algorithm. This solution procedure works with the assumption that coefficients of the momentum equations and pressure source are constant over a time step. These assumptions reduce the computation, only requiring coefficients and pressure to be calculated once, without the need for iterations within a time step.

Note that the momentum coefficients used to solve the pressure equation are modified using either Crank-Nicolson or Fully Implicit, while the momentum coefficients used to update the momentum equations with ERK-LS4 are the unmodified coefficients.

The RK-SIMPLER(A) algorithm is shown visually in Fig. 2.5.

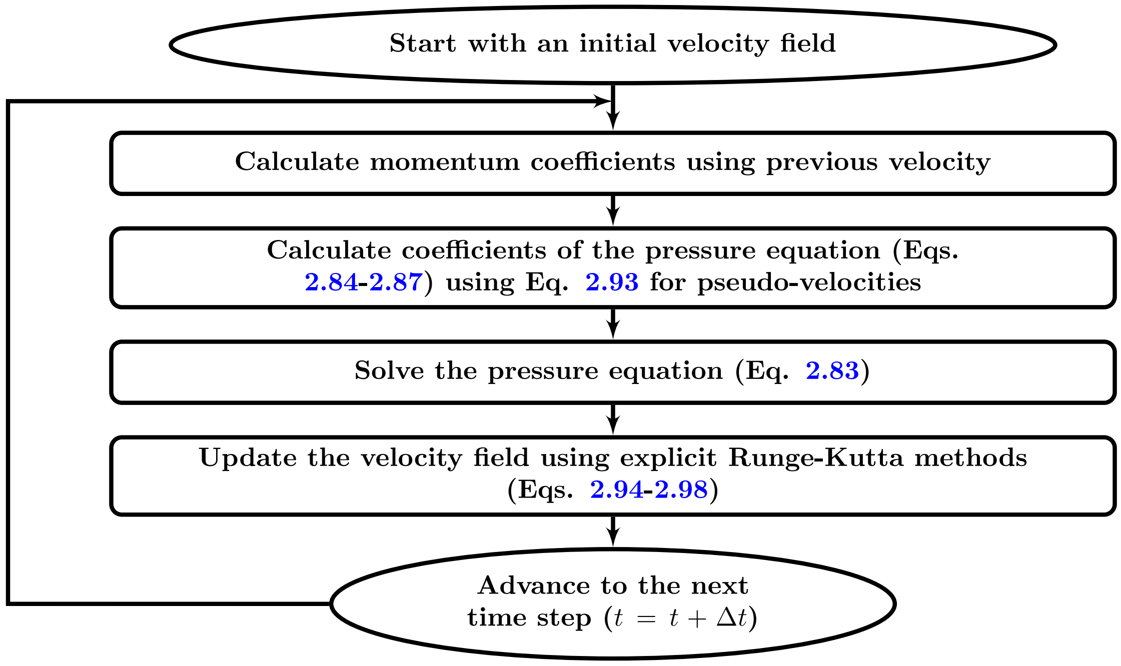

Start with an initial velocity field
Calculate momentum coefficients using previous velocity
Calculate coefficients of the pressure equation (Eqs. 2.84-2.87) using Eq. 2.93 for pseudo-velocities
Solve the pressure equation (Eq. 2.83)
Update the velocity field using explicit Runge-Kutta methods (Eqs. 2.94-2.98)
Advance to the next time step ($t = t + \Delta t$)

Figure 2.5: Diagram of the RK-SIMPLER(A) algorithm.

2.3.3.3 Variation B

The original RK-SIMPLER(A) algorithm keeps a constant pressure field for each RK stage, and therefore, does not account for continuity at each stage. As discussed in Sanderse [22], the order of accuracy may be limited. In an attempt to improve the RK-SIMPLER algorithm, variation B accounts for continuity at each stage. To achieve this, a pressure equation is formulated each Runge-Kutta stage from the stage equations, and not from a Fully Implicit or Crank-Nicolson formulation as was used in variation A. The momentum equations are integrated with explicit RK methods, with the following stage equation for u.

$$(u_P)_s = (u_P)^n + \Delta t \sum_{l=1}^{s-1} \alpha_{s,l} F_u\big(u_l, t^n + \gamma_l \Delta t\big) \qquad \text{for } 2 \le s \le S \tag{2.99}$$

Starting with $s = 2$ and recalling that in explicit RK methods $u_1 = u^n$, Eq. 2.99 is expanded to

$$(u_P)_2 = (u_P)^n + \alpha_{2,1} \frac{\Delta t}{\rho \Delta \forall} \Big(\sum a_{u-nb} u_{nb} - a_{u-P} u_P + b_u\Big)^n - \alpha_{2,1} \frac{\Delta t}{\rho \Delta \forall} \Delta y (p_e - p_w)_1 \ , \tag{2.100}$$

where coefficients are unmodified as defined in Eqs. 2.16, 2.15, and 2.22. The source term is defined by $b_u = S_u \Delta \forall$, which does not include pressure.

If, at the beginning of the time step, the velocity is known but the pressure is unknown (as was assumed in RK-SIMPLER), $(u_P)_2$ and p_1 are the only unknowns in Eq. 2.100. The stage equation is simplified to the following.

$$(u_P)_2 = \hat{u}_2 + (d_u)_2 (p_w - p_e)_1 \ , \tag{2.101}$$

with

$$\hat{u}_2 = (u_P)^n + \alpha_{2,1} \frac{\Delta t}{\rho \Delta \forall} \Big(\sum a_{u-nb} u_{nb} - a_{u-P} u_P + b_u\Big)^n \tag{2.102}$$

$$(d_u)_2 = \alpha_{2,1} \frac{\Delta t}{\rho \Delta \forall} \Delta y \ . \tag{2.103}$$

A similar equation is found for $(v_P)_2$.

$$(v_P)_2 = \hat{v}_2 + (d_v)_2 (p_s - p_n)_1 \ , \tag{2.104}$$

with

$$\hat{v}_2 = (v_P)^n + \alpha_{2,1}\frac{\Delta t}{\rho\Delta\forall}\left(\sum a_{v-nb}v_{nb} - a_{v-P}v_P + b_v\right)^n \tag{2.105}$$

$$(d_v)_2 = \alpha_{2,1}\frac{\Delta t}{\rho\Delta\forall}\Delta x \ , \tag{2.106}$$

where the source term defined by $b_v = S_v\Delta\forall$ does not contain pressure, and coefficients are unmodified as in Eqs. 2.16, 2.15, and 2.23.

The fully discrete momentum equations (Eqs. 2.101 and 2.104) are substituted into the discrete continuity equation (Eq. 2.44) at control volume faces to form an equation for pressure at stage 1.

$$\begin{aligned}(a_{p-P})_1\,(p_P)_1 =& (a_{p-E})_1\,(p_E)_1 + (a_{p-W})_1\,(p_W)_1 \\ &+ (a_{p-N})_1\,(p_N)_1 + (a_{p-S})_1\,(p_S)_1 + (bp)_1 \ ,\end{aligned} \tag{2.107}$$

where

$$(a_{p-E})_1 = \left[\rho\,(d_u)_1\right]_e \Delta y \qquad (a_{p-W})_1 = \left[\rho\,(d_u)_1\right]_w \Delta y \tag{2.108}$$

$$(a_{p-N})_1 = \left[\rho\,(d_v)_1\right]_n \Delta x \qquad (a_{p-S})_1 = \left[\rho\,(d_v)_1\right]_s \Delta x \tag{2.109}$$

$$(a_{p-P})_1 = (a(_{p-E})_1 + (a_{p-W})_1 + (a_{p-N})_1 + (a_{p-S})_1 \tag{2.110}$$

$$(b_p)_1 = \left[\rho(\hat{u})_1\right]_e\Delta y - \left[\rho(\hat{u})_1\right]_w\Delta y + \left[\rho(\hat{v})_1\right]_n\Delta x - \left[\rho(\hat{v})_1\right]_s\Delta x \ . \tag{2.111}$$

Once pressure is known, the velocity components are updated to stage 2 by Eqs. 2.101 and 2.104. The velocity and pressure are decoupled at each stage, with pressure solved first and momentum equations explicitly updated second. The pressure equation extracts a pressure that drives the velocity field towards satisfying continuity when the velocity components are updated. This process is carried out for all stages in the RK method. The process shown above for stage 2 is generalized for any RK stage below.

$$(u_P)_s = \hat{u}_s + (d_u)_s(p_w - p_e)_{s-1} \qquad \text{for } 2 \le s \le S \ , \tag{2.112}$$

with

$$\begin{aligned}\hat{u}_s =&(u_P)^n + \alpha_{s,s-1}\frac{\Delta t}{\rho \Delta \forall}\left(\sum a_{u-nb}u_{nb} - a_{u-P}u_P + b_u\right)_{s-1}\\ &+ \Delta t \sum_{l=1}^{s-2} \alpha_{s,l} F_u(u_l, t^n + \gamma_l \Delta t) \end{aligned} \qquad (2.113)$$

$$(d_u)_s = \alpha_{s,s-1}\frac{\Delta t}{\rho \Delta \forall}\Delta y \ . \qquad (2.114)$$

For the y velocity component,

$$(v_P)_s = \hat{v}_s + (d_v)_s (p_s - p_n)_{s-1} \qquad \text{for } 2 \leq s \leq S \ , \qquad (2.115)$$

with

$$\begin{aligned}\hat{v}_s =&(v_P)^n + \alpha_{s,s-1}\frac{\Delta t}{\rho \Delta \forall}\left(\sum a_{v-nb}v_{nb} - a_{v-P}v_P + b_v\right)_{s-1}\\ &+ \Delta t \sum_{l=1}^{s-2} \alpha_{s,l} F_v(v_l, t^n + \gamma_l \Delta t) \end{aligned} \qquad (2.116)$$

$$(d_v)_s = \alpha_{s,s-1}\frac{\Delta t}{\rho \Delta \forall}\Delta x \ . \qquad (2.117)$$

In both Eq. 2.113 and 2.116, the source terms for the $s-1$ stages do not contain pressure, but the functions F_u and F_v for stages $1 \leq l \leq s-2$ contain pressure from previous stages. The momentum coefficients are calculated at each stage using the velocity for that given stage.

The fully discrete momentum equations at each stage are substituted into the discrete continuity equation (Eq. 2.44) at all control volume faces to form an equation for pressure at each stage.

$$\begin{aligned}(a_{p-P})_{s-1}(p_P)_{s-1} =&(a_{p-E})_{s-1}(p_E)_{s-1} + (a_{p-W})_{s-1}(p_W)_{s-1}\\ &+(a_{p-N})_{s-1}(p_N)_{s-1} + (a_{p-S})_{s-1}(p_S)_{s-1} + (bp)_{s-1} \ , \end{aligned} \qquad (2.118)$$

where

$$(a_{p-E})_{s-1} = \left[\rho (d_u)_s\right]_e \Delta y \qquad (a_{p-W})_{s-1} = \left[\rho (d_u)_s\right]_w \Delta y \qquad (2.119)$$

$$(a_{p-N})_{s-1} = \left[\rho (d_v)_s\right]_n \Delta x \qquad (a_{p-S})_{s-1} = \left[\rho (d_v)_s\right]_s \Delta x \qquad (2.120)$$

$$(a_{p-P})_{s-1} = (a(_{p-E})_{s-1} + (a_{p-W})_{s-1} + (a_{p-N})_{s-1} + (a_{p-S})_{s-1} \tag{2.121}$$

$$(b_p)_{s-1} = \left[\rho(\hat{u})_s\right]_e \Delta y - \left[\rho(\hat{u})_s\right]_w \Delta y + \left[\rho(\hat{v})_s\right]_n \Delta x - \left[\rho(\hat{v})_s\right]_s \Delta x \; . \tag{2.122}$$

The final $n+1$ value for velocity is found by noting p_S is not yet known, and forming the following equations.

$$(u_P)^{n+1} = \hat{u}^{n+1} + (d_u)^{n+1}(p_w - p_e)_S \; , \tag{2.123}$$

with

$$\begin{aligned}\hat{u}^{n+1} =& (u_P)^n + \beta_S \frac{\Delta t}{\rho \Delta \forall} \left(\sum a_{u-nb} u_{nb} - a_{u-P} u_P + b_u\right)_S \\ &+ \Delta t \sum_{s=1}^{S-1} \beta_s F_u(u_s, t^n + \gamma_s \Delta t) \qquad (2.124)\\ (d_u)^{n+1} =& \beta_S \frac{\Delta t}{\rho \Delta \forall} \Delta y \; . \qquad (2.125)\end{aligned}$$

For y velocity,

$$(v_P)^{n+1} = \hat{v}^{n+1} + (d_v)^{n+1}(p_s - p_n)_S \; , \tag{2.126}$$

with

$$\begin{aligned}\hat{v}^{n+1} =& (v_P)^n + \beta_S \frac{\Delta t}{\rho \Delta \forall} \left(\sum a_{v-nb} v_{nb} - a_{v-P} v_P + b_v\right)_S \\ &+ \Delta t \sum_{s=1}^{S-1} \beta_s F_v(v_s, t^n + \gamma_s \Delta t) \qquad (2.127)\\ (d_v)^{n+1} =& \beta_S \frac{\Delta t}{\rho \Delta \forall} \Delta x \; . \qquad (2.128)\end{aligned}$$

The pressure equation for the final stage is

$$\begin{aligned}(a_{p-P})_S \, (p_P)_S =& (a_{p-E})_S \, (p_E)_S + (a_{p-W})_S \, (p_W)_S \\ &+ (a_{p-N})_S \, (p_N)_S + (a_{p-S})_S \, (p_S)_S + (bp)_S \; , \qquad (2.129)\end{aligned}$$

where

$$(a_{p-E})_S = \left[\rho \, (d_u)^{n+1}\right]_e \Delta y \qquad (a_{p-W})_S = \left[\rho \, (d_u)^{n+1}\right]_w \Delta y \tag{2.130}$$

$$(a_{p-N})_S = \left[\rho \, (d_v)^{n+1}\right]_n \Delta x \qquad (a_{p-S})_S = \left[\rho \, (d_v)^{n+1}\right]_s \Delta x \tag{2.131}$$

$$(a_{p-P})_S = (a(_{p-E})_S + (a_{p-W})_S + (a_{p-N})_S + (a_{p-S})_S \tag{2.132}$$

$$(b_p)_S = \left[\rho(\hat{u})^{n+1}\right]_e \Delta y - \left[\rho(\hat{u})^{n+1}\right]_w \Delta y + \left[\rho(\hat{v})^{n+1}\right]_n \Delta x - \left[\rho(\hat{v})^{n+1}\right]_s \Delta x \ . \tag{2.133}$$

Three different ERK methods are used in the RK-SIMPLER(B) algorithm. ERK1 is a one stage, first order accurate method equivalent to the Euler explicit integration. ERK2 is a two stage, second order accurate method, commonly known as the midpoint method. ERK3 is a three stage, third order accurate method of Wray [22]. The Butcher tableaus for these three ERK methods are given in Tables 2.3-2.5.

Table 2.3: ERK1.

0	0
	1

Table 2.4: ERK2.

0	0	
1/2	1/2	0
	0	1

Table 2.5: ERK3.

0	0		
8/15	8/15	0	
2/3	1/4	5/12	0
	1/4	0	3/4

2.3.3.4 RK-SIMPLER(B) Solution Procedure

Solving pressure and velocity at each stage and the final update to the $n+1$ time level is the same as described for the second stage. First, pressure is solved from the pressure equation, then velocities are updated explicitly. The procedure for the RK-SIMPLER(B) is as follows.

1. Start with an initial velocity field.

2. Start a new time step with $s = 2$. Set $u_1 = u^n$ and $v_1 = v^n$.

3. Calculate the momentum equation coefficients (Eqs. 2.16, 2.15, 2.22, and 2.23) using the previous stage velocity.

4. Calculate the pressure coefficients (Eqs. 2.119-2.122) using the values defined in Eqs. 2.113-2.117.

5. Solve the pressure equation for p_{s-1} (Eq. 2.118).

6. Update velocity components from Eqs. 2.112 and 2.115.

7. If $s = S$ go to step 8, otherwise set $s = s + 1$ and go to step 3.

8. Calculate the momentum equation coefficients (Eqs. 2.16, 2.15, 2.22, and 2.23) using the final stage velocity.

9. Calculate the pressure coefficients (Eqs. 2.130-2.133) using the values defined in Eqs. 2.124-2.128.

10. Solve the pressure equation for p_S (Eq. 2.129).

11. Update velocity components to $n + 1$ using Eqs. 2.123 and 2.126.

12. Advance in time $t = t + \Delta t$ and go to step 2.

This procedure is shown visually in Fig. 2.6. With variation B the pressure and velocity are decoupled, and equations to satisfy both momentum and continuity are developed each stage. RK-SIMPLER(B) requires more calculations per time step than RK-SIMPLER(A) because the momentum coefficients are updated and the pressure equation is solved at each stage. The pressure is not updated to $n + 1$ time level until the first stage of the next time step.

2.3.4 IRK-SIMPLER Algorithm

The RK-SIMPLER(A) algorithm efficiently and accurately simulates unsteady flow problems, but time step restrictions are severe in many cases [6]. IRK methods improve the time step restrictions. The IRK-SIMPLER algorithm uses IRK methods to integrate momentum equations and solves a pressure equation to satisfy continuity. Two variations are discussed, one that solves the pressure equation once per time step and one that solves the pressure equation each stage.

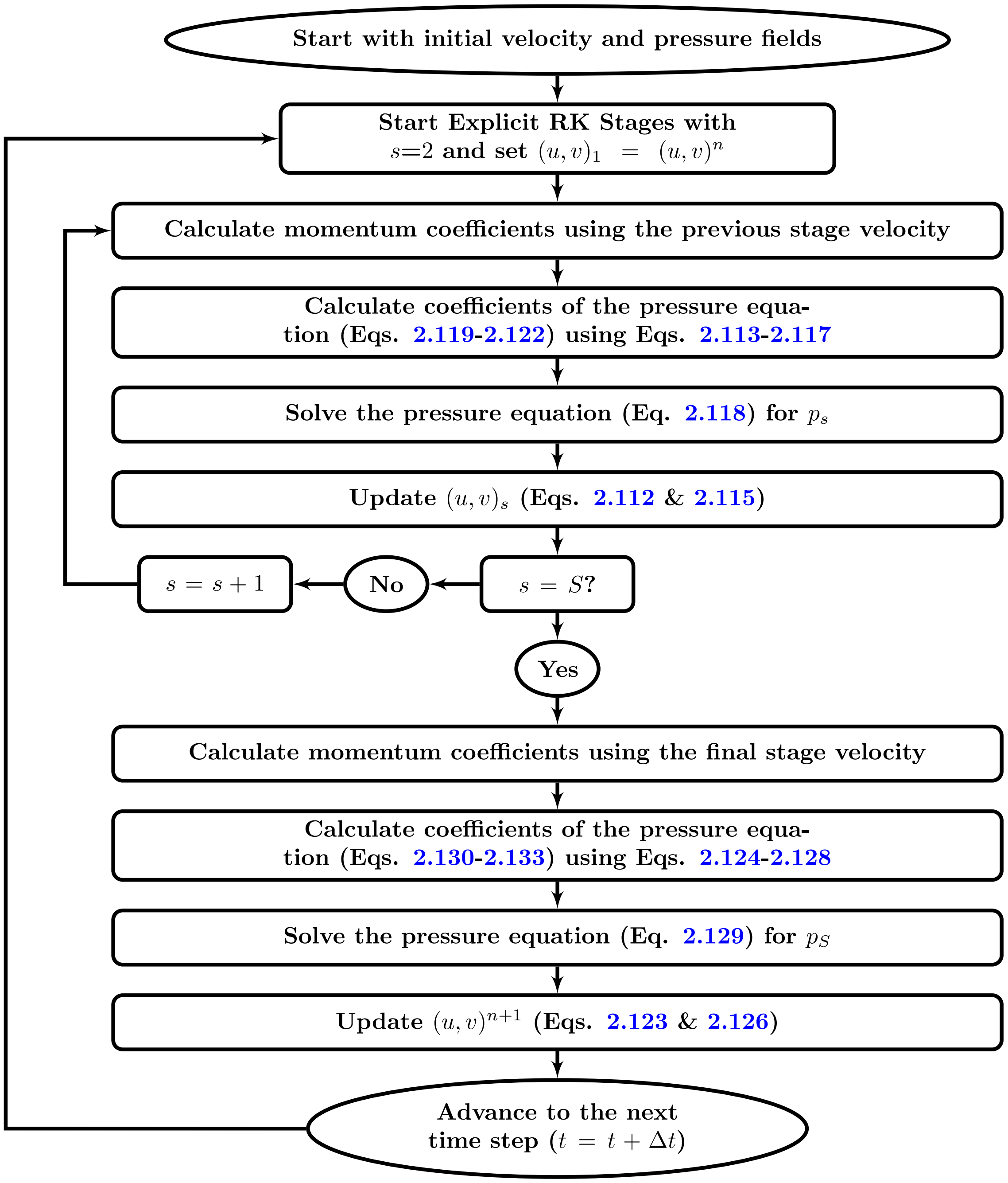

Start with initial velocity and pressure fields
Start Explicit RK Stages with s=2 and set $(u, v)_1 = (u, v)^n$
Calculate momentum coefficients using the previous stage velocity
Calculate coefficients of the pressure equation (Eqs. 2.119-2.122) using Eqs. 2.113-2.117
Solve the pressure equation (Eq. 2.118) for p_s
Update $(u, v)_s$ (Eqs. 2.112 & 2.115)
$s = S$?
No
$s = s + 1$
Yes
Calculate momentum coefficients using the final stage velocity
Calculate coefficients of the pressure equation (Eqs. 2.130-2.133) using Eqs. 2.124-2.128
Solve the pressure equation (Eq. 2.129) for p_S
Update $(u, v)^{n+1}$ (Eqs. 2.123 & 2.126)
Advance to the next time step ($t = t + \Delta t$)

Figure 2.6: Diagram of the RK-SIMPLER(B) algorithm.

2.3.4.1 Variation A

The IRK-SIMPLER(A) algorithm follows the same procedure as the RK-SIMPLER(A) algorithm, but instead of integrating the momentum equations with an explicit method, IRK-SIMPLER(A) integrates the momentum equations with implicit RK methods, specifically ESDIRK methods. The ESDIRK methods are chosen over FIRK or DIRK methods because of the lower cost of computation.

Similar to the explicit methods, ESDIRK methods require $u_1 = u^n$. Eq. 2.89 expands to the following for ESDIRK methods.

$$(u_P)_s = u^n + k\Delta t F_u(u_s, t^n + \gamma_s \Delta t) + \Delta t \sum_{l=1}^{s-1} \alpha_{s,l} F_u(u_l, t^n + \gamma_l \Delta t) \qquad 2 \le s \le S \ , \tag{2.134}$$

where $k = \alpha_{s,s}$ which is constant for all stages. Rearranging and using the definition of F_u

$$\begin{aligned}(u_P)_s = u^n &+ \frac{k\Delta t}{\rho \Delta \forall}\Big[\sum (a_{u-nb})^n\,(u_{nb})_s + (b_u)^n - (a_{u-P})^n\,(u_P)_s\Big] \\ &+ \Delta t \sum_{l=1}^{s-1} \alpha_{s,l} F_u(u_l, t^n + \gamma_l \Delta t) \qquad 2 \le s \le S \ ,\end{aligned} \tag{2.135}$$

where coefficients and source are unmodified as defined in Eqs. 2.16, 2.15, 2.22, and 2.23. The coefficients and source are evaluated once for all stages at time level n. This equation is set in a form suitable for solving with a linear solver.

$$a'_{u-P}(u_P)_s = \sum (a_{u-nb})^n (u_{nb})_s + b'_u \ , \tag{2.136}$$

where

$$a'_{u-P} = (a_{u-P})^n + \frac{\rho \Delta \forall}{k\Delta t} \tag{2.137}$$

$$b'_u = (b_u)^n + \frac{\rho \Delta \forall}{k \Delta t}(u_P)^n + \frac{1}{k}\sum_{l=1}^{s-1} \alpha_{s,l} \left(R_u\right)_l \ . \tag{2.138}$$

Similar to RK-SIMPLER(A), once the pressure equation is solved this equation updates the velocity field through each stage without recalculating momentum coefficients or pressure. Equation 2.136 is solved once for ESDIRK2 and twice for ESDIRK3. Because the coefficients do not change from stage to stage, LU factorization is used to factorize the coefficient matrix for the first stage. That factorized coefficient matrix is then used for all other stages with only a new set of source terms. See Appendix B for how this LU factorization, along side an approximate factorization, reduces the computation time to solve these equations. The IRK-SIMPLER(A) algorithm is tested with and without these factorizations.

Because the ESDIRK methods used are also stiffly accurate, the final update to time level $n+1$ is trivial, $u^{n+1} = u_S$.

Two different ESDIRK methods are used to test the IRK-SIMPLER(A) algorithm. ESDIRK2 is a two stage, second order accurate method that is equivalent to Crank-Nicolson time integration. ESDIRK3 is a three stage, third order accurate method. Both of these methods are stiffly accurate, implying $u^{n+1} = u_S$. The Butcher Tableaus for these two methods are given in Tables 2.6 and 2.7.

Table 2.6: ESDIRK2.

0	0	
1	1/2	1/2
	1/2	1/2

Table 2.7: ESDIRK3, $\delta = \sqrt{3}/3$.

0	0		
$1-\delta$	$\frac{1}{2}(1-\delta)$	$\frac{1}{2}(1-\delta)$	
1	$1-\frac{1}{2}(\frac{1}{1-\delta}-\delta)$	$\frac{1}{2}(\frac{1}{1-\delta}-1)$	$\frac{1}{2}(1-\delta)$
	$1-\frac{1}{2}(\frac{1}{1-\delta}-\delta)$	$\frac{1}{2}(\frac{1}{1-\delta}-1)$	$\frac{1}{2}(1-\delta)$

2.3.4.2 IRK-SIMPLER(A) Solution Procedure

The IRK-SIMPLER(A) algorithm follows the RK-SIMPLER(A) procedure closely by solving the pressure equation first using a Crank-Nicolson or Fully Implicit time integration, then updating the velocity using an ESDIRK method.

1. Start with an initial velocity field.
2. Calculate the momentum equation coefficients (Eqs. 2.53-2.58) using previous velocity field.
3. Calculate the pressure coefficients (Eqs. 2.84-2.87) using the pseudo-velocities defined in Eq. 2.93.
4. Solve the pressure equation (Eq. 2.83).

5. Update the velocity field by solving the implicit equations for each stage (Eq. 2.136) with the calculated pressure field as a source.

6. Final update to $n+1$ time level ($u^{n+1} = u_S$ and $v^{n+1} = v_S$) for stiffly accurate IRK methods.

7. Advance in time $t = t + \Delta t$ and go to step 2.

Figure 2.7 shows this procedure visually.

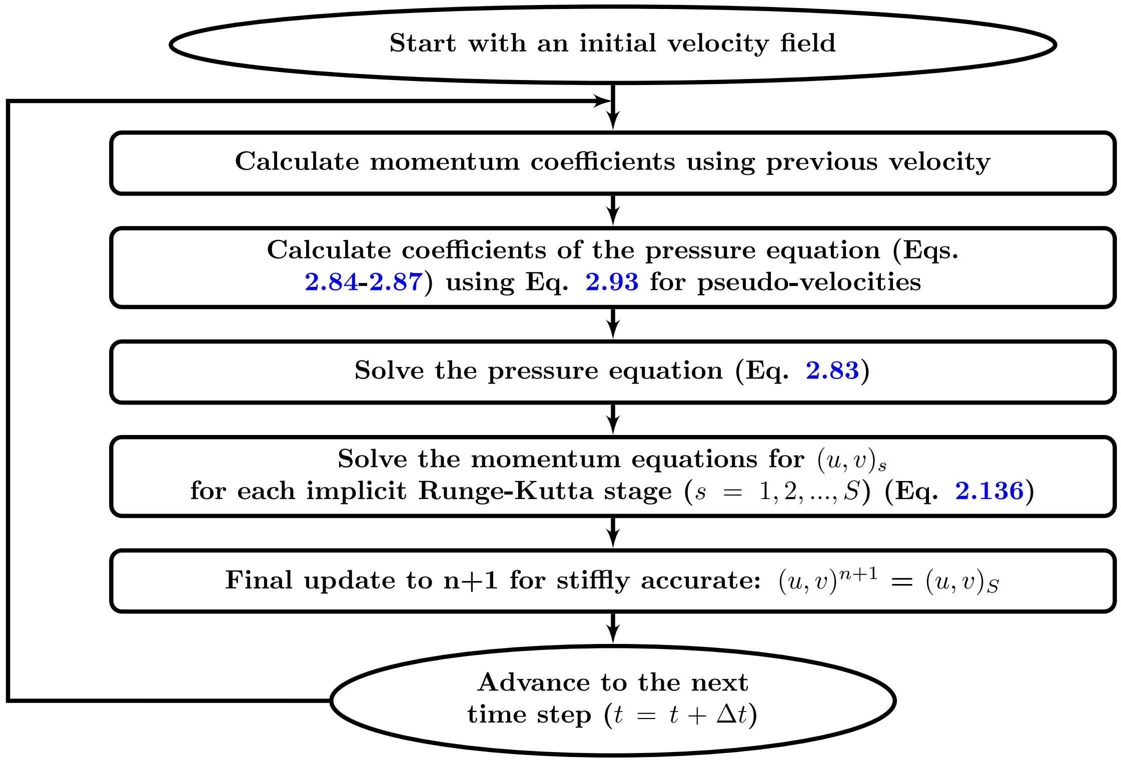

Start with an initial velocity field
Calculate momentum coefficients using previous velocity
Calculate coefficients of the pressure equation (Eqs. 2.84-2.87) using Eq. 2.93 for pseudo-velocities
Solve the pressure equation (Eq. 2.83)
Solve the momentum equations for $(u,v)_s$ for each implicit Runge-Kutta stage ($s = 1,2,...,S$) (Eq. 2.136)
Final update to n+1 for stiffly accurate: $(u,v)^{n+1} = (u,v)_S$
Advance to the next time step ($t = t + \Delta t$)

Figure 2.7: Diagram of the IRK-SIMPLER(A) algorithm.

2.3.4.3 Variation B

IRK-SIMPLER(B) uses DIRK methods and, like RK-SIMPLER(B), a pressure equation is developed each stage to better satisfy all conservation equations. As with RK-SIMPLER(B), the pressure equation is formed each stage from the RK stage equation. Unlike RK-SIMPLER(B), the pressure and velocity are coupled and solved simultaneously to satisfy all equations. Recall the stage equations for DIRK methods.

$$(u_P)_s = (u_P)^n + \Delta t \sum_{l=1}^{s} \alpha_{s,l} F_u\big(u_l, t^n + \gamma_l \Delta t\big) \qquad \text{for } 1 \le s \le S \tag{2.139}$$

Rearranging and using the definition of F_u

$$\begin{aligned}(u_P)_s = (u_P)^n + a_{s,s}\frac{\Delta t}{\rho \Delta \forall}\Big(\sum a_{u-nb}\, u_{nb} - a_{u-P}\, u_P + b_u - \Delta y(p_e - p_w)\Big)_s \\ + \Delta t \sum_{l=1}^{s-1} \alpha_{s,l} F_u\big(u_l, t^n + \gamma_l \Delta t\big)\ .\end{aligned} \tag{2.140}$$

Defining the term

$$R_s(u) = \rho \Delta \forall \sum_{l=1}^{s-1} \alpha_{s,l} F_u\big(u_l, t^n + \gamma_l \Delta t\big) \tag{2.141}$$

leads to the fully discrete x momentum equation for each stage.

$$a'_{u-P}\,(u_P)_s = \sum (a_{u-nb})_s\,(u_{nb})_s + b'_u + \Delta y(p_w - p_e)_s\ , \tag{2.142}$$

where

$$a'_{u-P} = (a_{u-P})_s + \frac{\rho \Delta \forall}{\alpha_{s,s} \Delta t} \tag{2.143}$$

$$b'_u = (b_u)_s + \frac{\rho \Delta \forall}{\alpha_{s,s} \Delta t}(u_P)^n + \frac{R_s(u)}{\alpha_{s,s}}\ . \tag{2.144}$$

Similarly for the y momentum equation

$$a'_{v-P}\,(v_P)_s = \sum (a_{v-nb})_s\,(v_{nb})_s + b'_v + \Delta x(p_s - p_n)_s\ , \tag{2.145}$$

where

$$a'_{v-P} = (a_{v-P})_s + \frac{\rho \Delta \forall}{\alpha_{s,s} \Delta t} \tag{2.146}$$

$$b'_v = (b_v)_s + \frac{\rho \Delta \forall}{\alpha_{s,s} \Delta t}(v_P)^n + \frac{R_s(v)}{\alpha_{s,s}}\ . \tag{2.147}$$

The momentum equations are written as

$$(u_P)_s = \hat{u}_s + (d_u)_s(p_w - p_e)_s \tag{2.148}$$

$$(v_P)_s = \hat{v}_s + (d_v)_s(p_s - p_n)_s\ , \tag{2.149}$$

with

$$\hat{u}_s = \frac{\sum(a_{u-nb}\, u_{nb})_s + b'_u}{a'_{u-P}} \qquad (d_u)_s = \frac{\Delta y}{a'_{u-P}} \tag{2.150}$$

$$\hat{v}_s = \frac{\sum(a_{v-nb}\, v_{nb})_s + b'_v}{a'_{v-P}} \qquad (d_v)_s = \frac{\Delta x}{a'_{v-P}} \; . \tag{2.151}$$

To derive a pressure equation, the momentum equations (Eqs. 2.148 and 2.149) are substituted into the discrete continuity equation (Eq. 2.44), and an equation for pressure is found.

$$\begin{aligned}(a_{p-P})_s\,(p_P)_s =& (a_{p-E})_s\,(p_E)_s + (a_{p-W})_s\,(p_W)_s \\ &+ (a_{p-N})_s\,(p_N)_s + (a_{p-S})_s\,(p_S)_s + (bp)_s \; ,\end{aligned} \tag{2.152}$$

where

$$(a_{p-E})_s = \left[\rho\,(d_u)_s\right]_e \Delta y \qquad (a_{p-W})_s = \left[\rho\,(d_u)_s\right]_w \Delta y \tag{2.153}$$

$$(a_{p-N})_s = \left[\rho\,(d_v)_s\right]_n \Delta x \qquad (a_{p-S})_s = \left[\rho\,(d_v)_s\right]_s \Delta x \tag{2.154}$$

$$(a_{p-P})_s = (a(_{p-E})_s + (a_{p-W})_s + (a_{p-N})_s + (a_{p-S})_s \tag{2.155}$$

$$(b_p)_s = \left[\rho(\hat{u})_s\right]_e \Delta y - \left[\rho(\hat{u})_s\right]_w \Delta y + \left[\rho(\hat{v})_s\right]_n \Delta x - \left[\rho(\hat{v})_s\right]_s \Delta x \; . \tag{2.156}$$

Unlike RK-SIMPLER(B), the pressure equation source term contains $\hat{u}_s$ and $\hat{v}_s$ which are unknown, being functions of the velocity components at the s^{th} stage. IRK-SIMPLER(B) has a coupled pressure and velocity at each stage (i.e., the stage equation, Eq. 2.142, has pressure and velocity as unknowns on the right hand side of the equation, so it cannot be substituted into continuity and solved for pressure immediately).

The procedure for coupling the pressure and velocity is as follows. First, the momentum equations (Eqs. 2.142 and 2.145) are solved implicitly with the momentum coefficients and pressure term lagged by one stage (i.e., assume $p_s = p_{s-1}$ and $a_s = a_{s-1}$). With approximations for the updated velocity components, the momentum coefficients are updated to the s^{th} stage (i.e., calculate a_s). An iterative loop is then preformed to couple pressure and velocity. This corrective inner loop follows closely to the inner loop of the IDEAL algorithm [9]. In the loop, first pressure coefficients are calculated from $\hat{u}_s$, $\hat{v}_s$, $(d_u)_s$, and $(d_v)_s$, and the pressure equation is solved. With an updated pressure, the momentum equations are updated explicitly with Eqs. 2.148 and 2.149. The values of $\hat{u}_s$ and $\hat{v}_s$ are then updated, and pressure is solved again. This loop of calculating $\hat{u}$ and $\hat{v}$, solving pressure, and updating momentum explicitly is an efficient method to couple pressure and velocity without multiple implicit momentum equation solutions, relaxation, or outer iterations. In practice, only a few iterations are needed to effectively couple the pressure and velocity (five iterations are used for all simulations in this paper).

If the initial pressure field is not known, as was assumed for all other methods, then a guess pressure field is calculated before the first time step from a pressure equation. This pressure equation is formed in the same way as RK-SIMPLER(A) with Crank-Nicolson or Fully Implicit discretization. The choice of integration method for this initial guess pressure has little effect on the solution as it is only used in the first time step to find an initial guess of the pressure field. Once the first time step has been completed, an accurate pressure field is obtained from the coupling procedure of IRK-SIMPLER(B). Using uniform pressure for a guessed pressure field the first time step instead of calculating a pressure field also yields accurate results.

The momentum coefficients are updated after the momentum equations are implicitly solved and contain some error. Optional iterations may be done, such that the momentum coefficients are recalculated after some number of inner coupling iterations. It is shown in Section 2.4.2.1 that only one iteration is sufficient to reach an accurate solution and additional iterations yield a small reduction of error with added computations. The final update to the $n+1$ time level forms an explicit equation similar to the explicit RK method. By using stiffly accurate DIRK methods which require $u_S = u^{n+1}$, once the final stage is completed the updated velocity and pressure are known. Stiffly accurate DIRK methods are used in the IRK-SIMPLER(B) algorithm because they require fewer computations and have improved stability when solving stiff equations.

Three stiffly accurate DIRK methods are used to test the IRK-SIMPLER(B) algorithm. DIRK1 is a one stage, first order accurate method equivalent to Euler implicit (Fully Implicit) integration. DIRK2 is a two stage, second order accurate method, and DIRK3 is a three stage, third order accurate method (both are developed in Alexander [50]). The Butcher Tableaus for these three DIRK methods are given in Tables 2.8-2.10.

Table 2.8: DIRK1.

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}$$

Table 2.9: DIRK2, $\alpha = 1 - \sqrt{2}/2$.

$$\begin{array}{c|cc} \alpha & \alpha & \\ 1 & 1-\alpha & \alpha \\ \hline & 1-\alpha & \alpha \end{array}$$

Table 2.10: DIRK3.

$$\begin{array}{c|ccc} \alpha & \alpha & & \\ \tau_2 & \tau_2 - \alpha & \alpha & \\ 1 & b_1 & b_2 & \alpha \\ \hline & b_1 & b_2 & \alpha \end{array}$$

α is the root of $x^3 - 3x^2 + \frac{3}{2}x - \frac{1}{6} = 0$ lying in $(\frac{1}{6}, \frac{1}{2})$

$\tau_2 = \quad (1+\alpha)/2$

$b_1 = -(6\alpha^2 - 16\alpha + 1)/4$

$b_2 = \quad (6\alpha^2 - 20\alpha + 5)/4$

2.3.4.4 IRK-SIMPLER(B) Solution Procedure

The procedure for IRK-SIMPLER(B) is as follows.

1. Start with initial velocity and pressure fields.

2. Start implicit Runge-Kutta stages with $s = 1$.

3. Calculate the momentum equation coefficients (Eqs. 2.143, 2.144, 2.146, and 2.147) using previous velocity field.

4. Solve momentum equations implicitly with pressure and coefficients lagged (Eqs. 2.142 and 2.145).

5. Update momentum coefficients with updated velocities (Eqs. 2.143, 2.144, 2.146, and 2.147).

6. Calculate the pressure coefficients from d_u and d_v (Eqs. 2.153-2.155).

7. Couple the pressure and velocity with inner iterations via:

   (a) Update $\hat{u}$ and $\hat{v}$ and calculate pressure source b_p (Eq. 2.156).

(b) Solve the pressure equation (Eq. 2.152).

(c) Update velocity components explicitly from Eqs. 2.148 and 2.149.

8. If updating momentum coefficients again go to step 5, otherwise continue to step 9.

9. If $s < S$ set $s = s + 1$ and go to step 3, otherwise continue to step 10.

10. Set $u^{n+1} = u_S$, $v^{n+1} = v_S$, and $p^{n+1} = p_S$ for stiffly accurate final updates

11. Advance in time $t = t + \Delta t$ and go to step 2.

The IRK-SIMPLER(B) procedure is also shown in Figure 2.8.

2.3.5 Algorithm Flowcharts

Figure 2.9 shows an outline of all algorithms of interest for one time step to compare the amount of work required to complete one time step. Shaded boxes in Fig. 2.9 represent implicit, iterative solution processes (i.e., matrix equation solutions), which require more computations than the explicit white boxes. RK-SIMPLER(A) requires only one implicit equation to be solved, while RK-SIMPLER(B) requires one implicit equation to be solved *for each* Runge-Kutta stage. For IRK-SIMPLER(A) in two-dimensions, three implicit equations are solved (one for each velocity component and one for pressure), but IRK-SIMPLER(B) requires the solution of the momentum equations implicitly for each Runge-Kutta stage, and the pressure equation to be solved multiple times within each Runge-Kutta stage. In practice, the pressure equation in the inner loop of IRK-SIMPLER(B) does not require a fully converged solution each iteration, but is solved alongside the explicit update of momentum equations for a more efficient solution procedure (i.e. each inner loop a few iterations of the pressure solver is used followed by the explicit momentum update). The SIMPLER algorithm requires the solution of four implicit equations *each sub-iteration* within a time step.

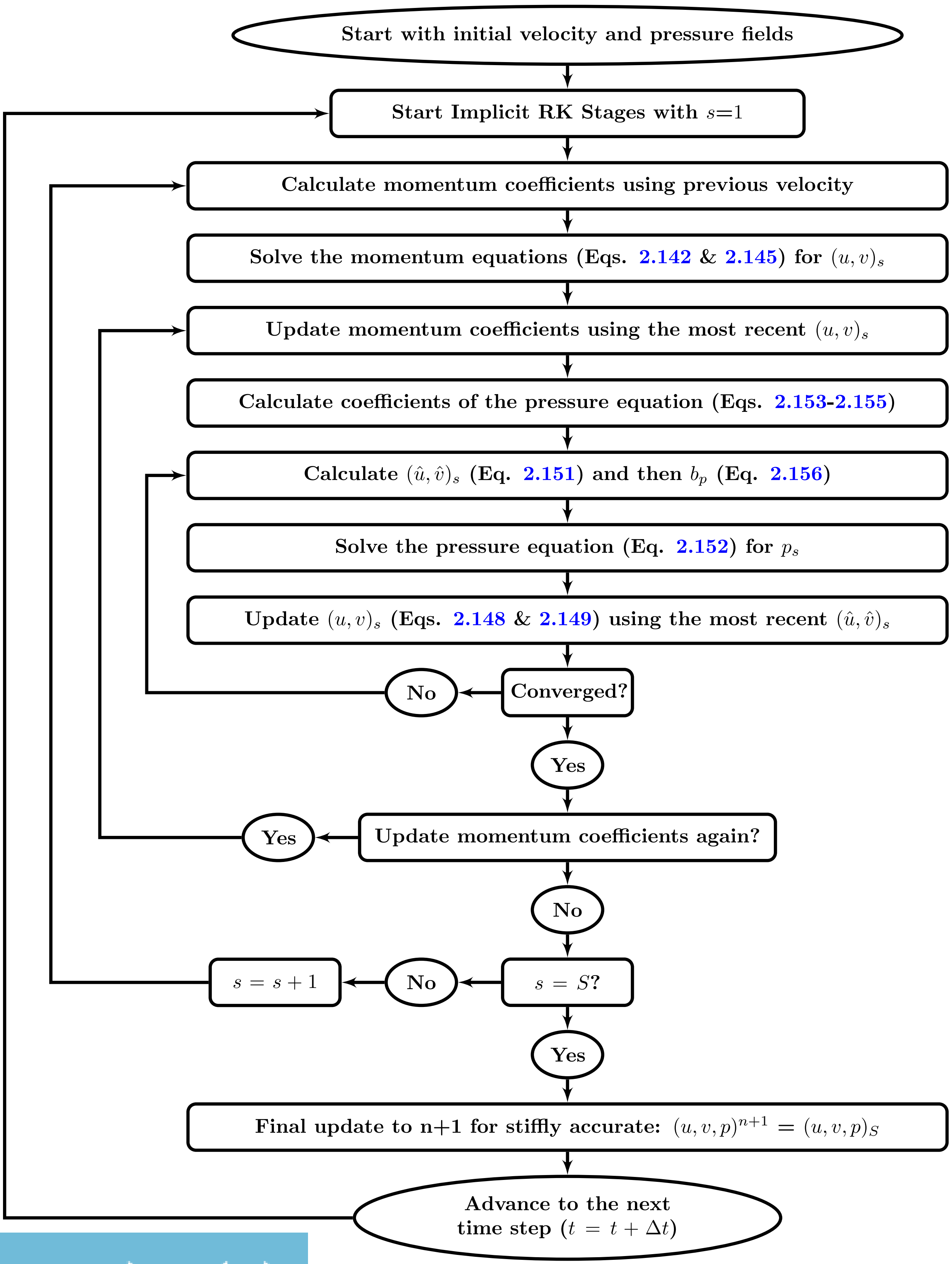

Start with initial velocity and pressure fields
Start Implicit RK Stages with s=1
Calculate momentum coefficients using previous velocity
Solve the momentum equations (Eqs. 2.142 & 2.145) for $(u,v)_s$
Update momentum coefficients using the most recent $(u,v)_s$
Calculate coefficients of the pressure equation (Eqs. 2.153-2.155)
Calculate $(\hat{u},\hat{v})_s$ (Eq. 2.151) and then b_p (Eq. 2.156)
Solve the pressure equation (Eq. 2.152) for p_s
Update $(u,v)_s$ (Eqs. 2.148 & 2.149) using the most recent $(\hat{u},\hat{v})_s$
Converged?
No
Yes
Update momentum coefficients again?
Yes
No
$s = S$?
No
$s = s + 1$
Yes
Final update to n+1 for stiffly accurate: $(u,v,p)^{n+1} = (u,v,p)_S$
Advance to the next time step ($t = t + \Delta t$)

Figure 2.8: Diagram of the IRK-SIMPLER(B) algorithm.

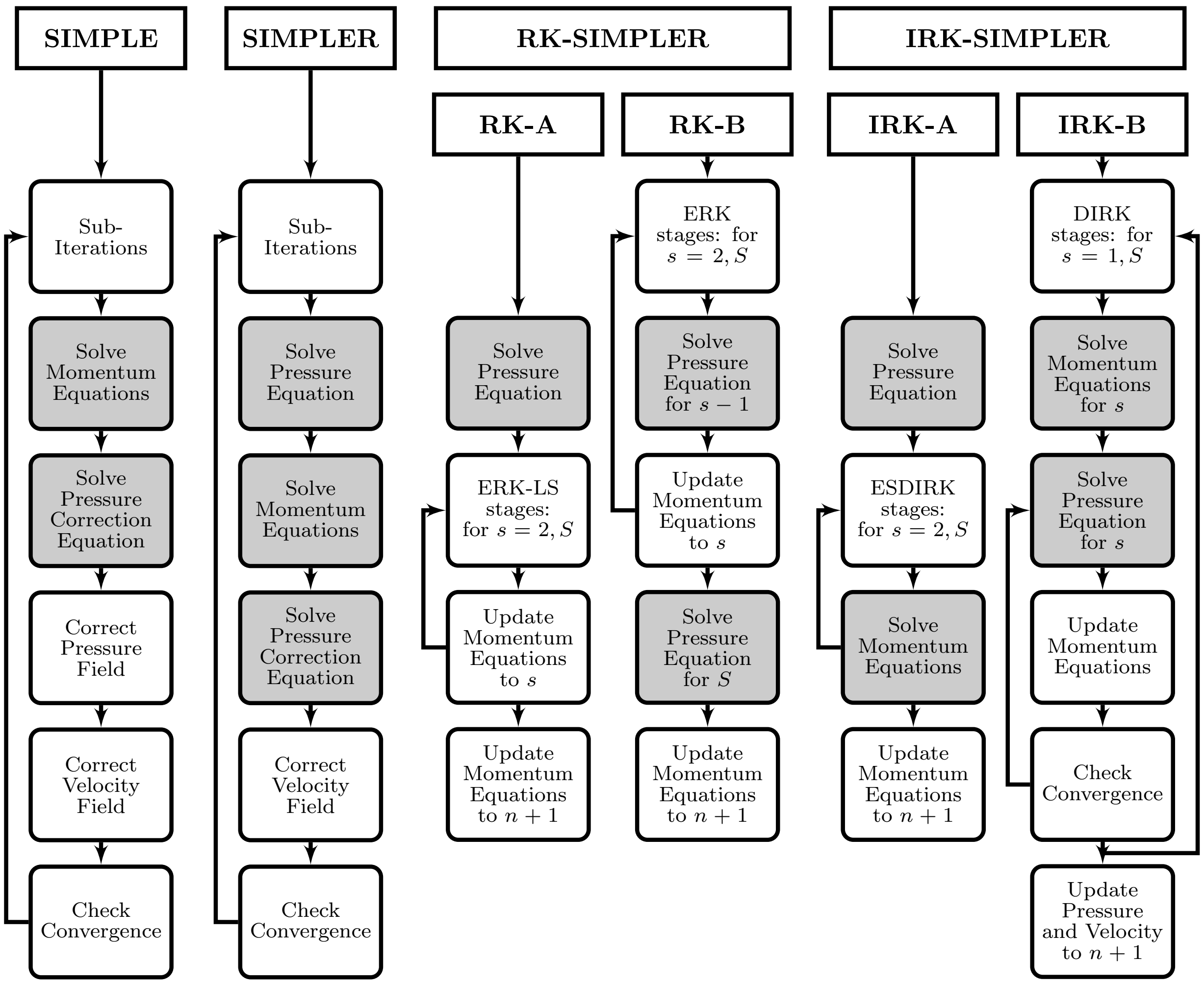

SIMPLE
SIMPLER
RK-SIMPLER
IRK-SIMPLER
RK-A
RK-B
IRK-A
IRK-B
Sub-Iterations
Solve Momentum Equations
Solve Pressure Correction Equation
Correct Pressure Field
Correct Velocity Field
Check Convergence
Sub-Iterations
Solve Pressure Equation
Solve Momentum Equations
Solve Pressure Correction Equation
Correct Velocity Field
Check Convergence
Solve Pressure Equation
ERK-LS stages: for $s = 2, S$
Update Momentum Equations to s
Update Momentum Equations to $n + 1$
ERK stages: for $s = 2, S$
Solve Pressure Equation for $s - 1$
Update Momentum Equations to s
Solve Pressure Equation for S
Update Momentum Equations to $n + 1$
Solve Pressure Equation
ESDIRK stages: for $s = 2, S$
Solve Momentum Equations
Update Momentum Equations to $n + 1$
DIRK stages: for $s = 1, S$
Solve Momentum Equations for s
Solve Pressure Equation for s
Update Momentum Equations
Check Convergence
Update Pressure and Velocity to $n + 1$

Figure 2.9: Algorithm procedure for one time step (shaded boxes are implicit solutions).

For comparison, if the SIMPLER algorithm has a fixed 20 sub-iterations, each time step requires the solution of 80 implicit equations each time step. RK-SIMPLER(A) requires the solution of only one implicit equation, and IRK-SIMPLER(A) requires the solution of three implicit equations each time step. For an S stage ERK method, RK-SIMPLER(B) requires S implicit equations to be solved, and IRK-SIMPLER(B) requires $3S$ implicit equations to be solved each time step. Ranking the amount of work required for one time step of each algorithm from least to greatest for a typical number of RK stages: 1) RK-SIMPLER(A), 2) IRK-SIMPLER(A), 3) RK-SIMPLER(B), 4) IRK-SIMPLER(B), 5) SIMPLE, and 6) SIMPLER.

RK-SIMPLER(A) and IRK-SIMPLER(A) are simple, efficient methods that require no sub-iterations and the solution of the pressure and momentum equations only once per time step. However, the A variants rely on the assumption that pressure and momentum coefficients are constant for a time step. RK-SIMPLER(B) requires pressure to be solved for each RK stage, but no iterations are needed within the stages. IRK-SIMPLER(B) requires iterations within each RK stage and the pressure equation is solved each stage alongside the momentum equations. All RK-SIMPLER and IRK-SIMPLER algorithms require no relaxation.

SIMPLE and SIMPLER require sub-iterations (typically 10 to 50) for which all steps are repeated. RK-SIMPLER(A) and IRK-SIMPLER(A) require no sub-iterations, only iteration through the RK stages (either ERK-LS or ESDIRK), where only the momentum equations are solved/updated. RK-SIMPLER(B) also requires no sub-iterations, but the iteration through the ERK stages involve both the solution of the pressure equation and the update of the momentum equations. IRK-SIMPLER(B) requires iteration through DIRK stages to solve the momentum equations once implicitly and then inner iteration to both solve the pressure equation and update the momentum equations. The iterations in IRK-SIMPLER(B) include the inner loop to couple pressure and velocity as well as the loop to update the momentum equation coefficients (see Fig. 2.8). IRK-SIMPLER(B) iterations are fewer in number than SIMPLE and SIMPLER (typically 1-2 momentum coefficient updates and 5-10 pressure-velocity loops, 5-20 total iterations) with much less work each iteration compared to either SIMPLE or SIMPLER.

The total number of implicit equations each algorithm requires to solve each time step is shown in Table 2.11. IRK-SIMPLER(B) requires the implicit pressure equation to be solved many times; however, in the inner loop the pressure equation is not solved completely. In the inner loop, a lower residual tolerance or fewer fixed iterations are used rather than a more rigorous solution found in other algorithms. Using fewer iterations on the pressure equation and more inner iterations, the pressure and velocity are solved more closely together.

Table 2.11: Number of implicit equations each time step for each algorithm[1].

Algorithm	# Implicit Eqs. for Two-Dimensions	# Implicit Eqs. for Three-Dimensions
SIMPLE	$3N_{sub}$	$4N_{sub}$
SIMPLER	$4N_{sub}$	$5N_{sub}$
RK-SIMPLER(A)	1	1
RK-SIMPLER(B)	S	S
IRK-SIMPLER(A)	$1 + 2(S - 1)$	$1 + 3(S - 1)$
IRK-SIMPLER(B)	$(2 + N_{up}N_{p-v})S$	$(3 + N_{up}N_{p-v})S$

Table 2.12 shows the naming conventions used to denote different algorithms and time integration methods tested. Both Approximate Factorization (AF) and LU factorization are used together in this study for IRK-SIMPLER(A) and is denoted by the acronym AF.

Table 2.12: Nomenclature to define algorithms and time integration methods.

Algorithm	Time Integration Method	Shorthand Name
SIMPLER	Fully Implicit	S-FI
	Crank-Nicolson	S-CN
RK-SIMPLER(A)	ERK-LS4	RK-A
RK-SIMPLER(B)	ERK1	RK-B1
	ERK2	RK-B2
	ERK3	RK-B3
IRK-SIMPLER(A)	ESDIRK2	IRK-A1
	ESDIRK3	IRK-A2
	ESDIRK2 w/ AF	IRK-A1-AF
	ESDIRK3 w/ AF	IRK-A2-AF
IRK-SIMPLER(B)	DIRK1	IRK-B1
	DIRK2	IRK-B2
	DIRK3	IRK-B3

[1] N_{sub}=number of sub-iterations, S=number of RK stages, N_{up}=number of momentum coefficient updates for IRK-SIMPLER(B), N_{p-v}=number of pressure-velocity coupling iterations for IRK-SIMPLER(B).

2.4 Algorithm Validation and Efficiency

2.4.1 Lid-Driven Cavity

The lid-driven cavity is a simple test case for incompressible flow and is used to analyze the algorithms of interest. The lid-driven cavity case is set up as a two-dimensional box with a moving lid on the top (Fig. 2.10). The lid moves at a constant speed U_{lid}, and the dimensions of the box are L by L. All sides of the box are no-slip walls. The Reynolds number for this case is $Re = \rho U_{lid} L/\mu$. The values used to non-dimensionalized this case are L for length, U_{lid} for velocity, and L/U_{lid} for time. Non-dimensional values are denoted with primes. For example, $u' = u/U_{lid}$ and $\Delta t' = \Delta t * U_{lid}/L$.

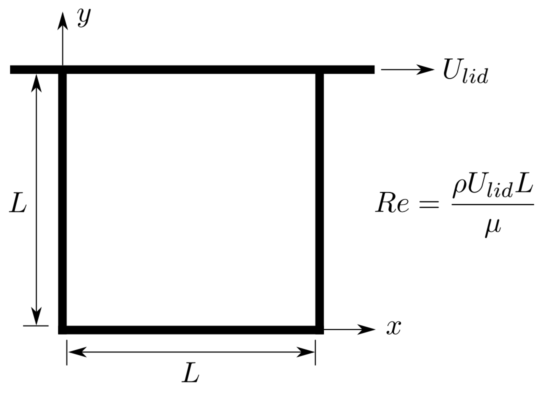


Figure 2.10: Schematic of the lid-driven cavity problem.

Simulations are run on a 42 by 42 uniform grid ($\Delta x' = \Delta y' = 0.025$) with steady state results from two Reynolds number cases (100 and 1,000) presented. The C-N SIMPLER, RK-SIMPLER, and IRK-SIMPLER algorithms are compared in Fig. 2.11 by plotting u-velocity profiles against y at the vertical center-line ($x' = 0.5$), and v-velocity profiles against x at the horizontal center-line ($y' = 0.5$).

The results for all algorithms fall on top of each other because the same spatial discretization is used for all algorithms, and the simulations are all run until a steady state is reached. The results for all methods are not shown, but all profiles match with u and v L2 error less than 1×10^{-10}. These results verify all Runge-Kutta algorithms accurately simulate steady flows.

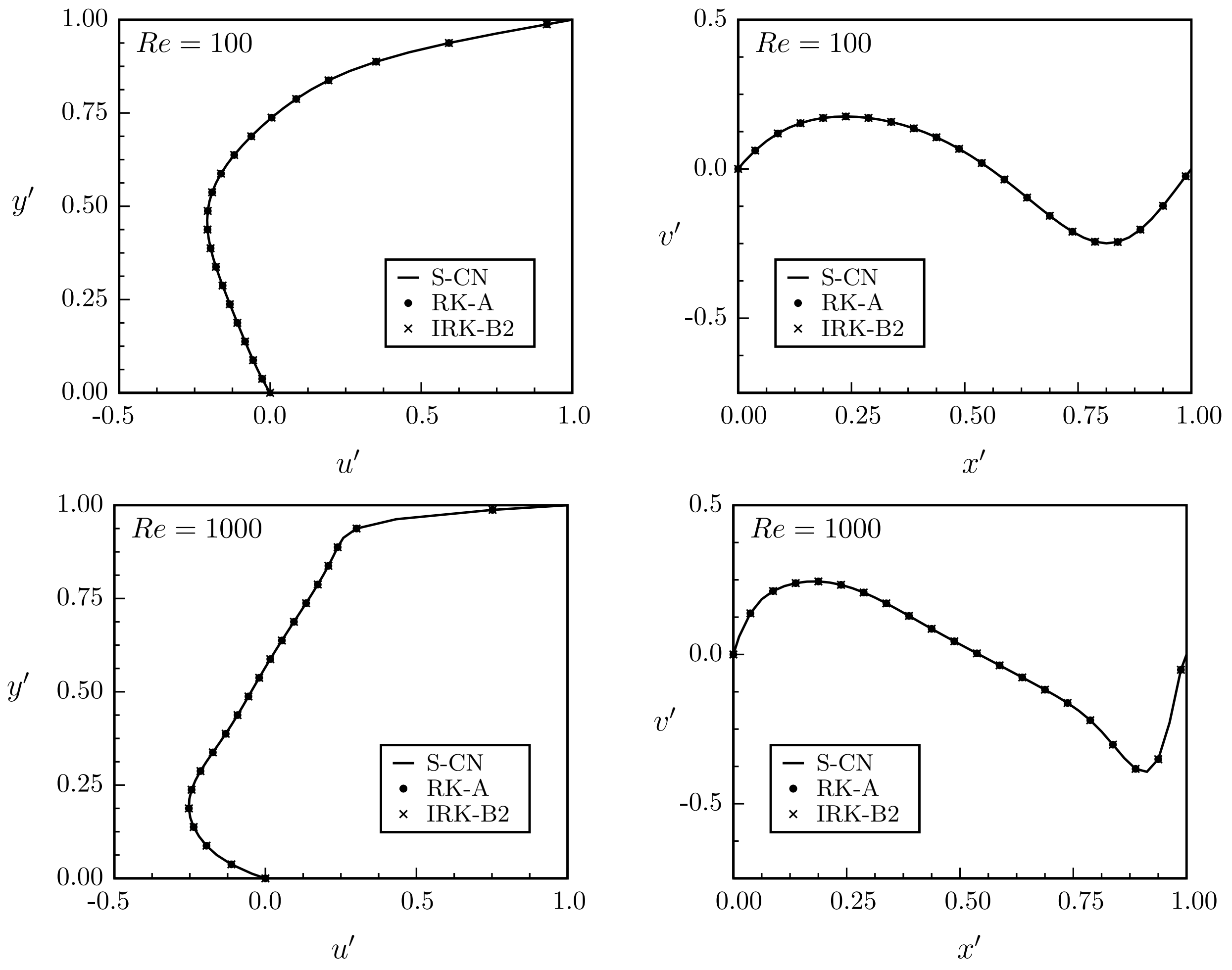


Figure 2.11: Driven cavity center-line velocity profiles.

Results are compared to simulation results of Wirogo [29] in Figs. 2.12. Results at Reynolds number of 100 match well on this relatively coarse grid, while results at Reynolds number of 1,000 do not match as closely. A more refined grid is required to better match the results of Wirogo [29]. Chapter 4 examines different flux schemes, and shows how different methods converge to match Wirogo's data.

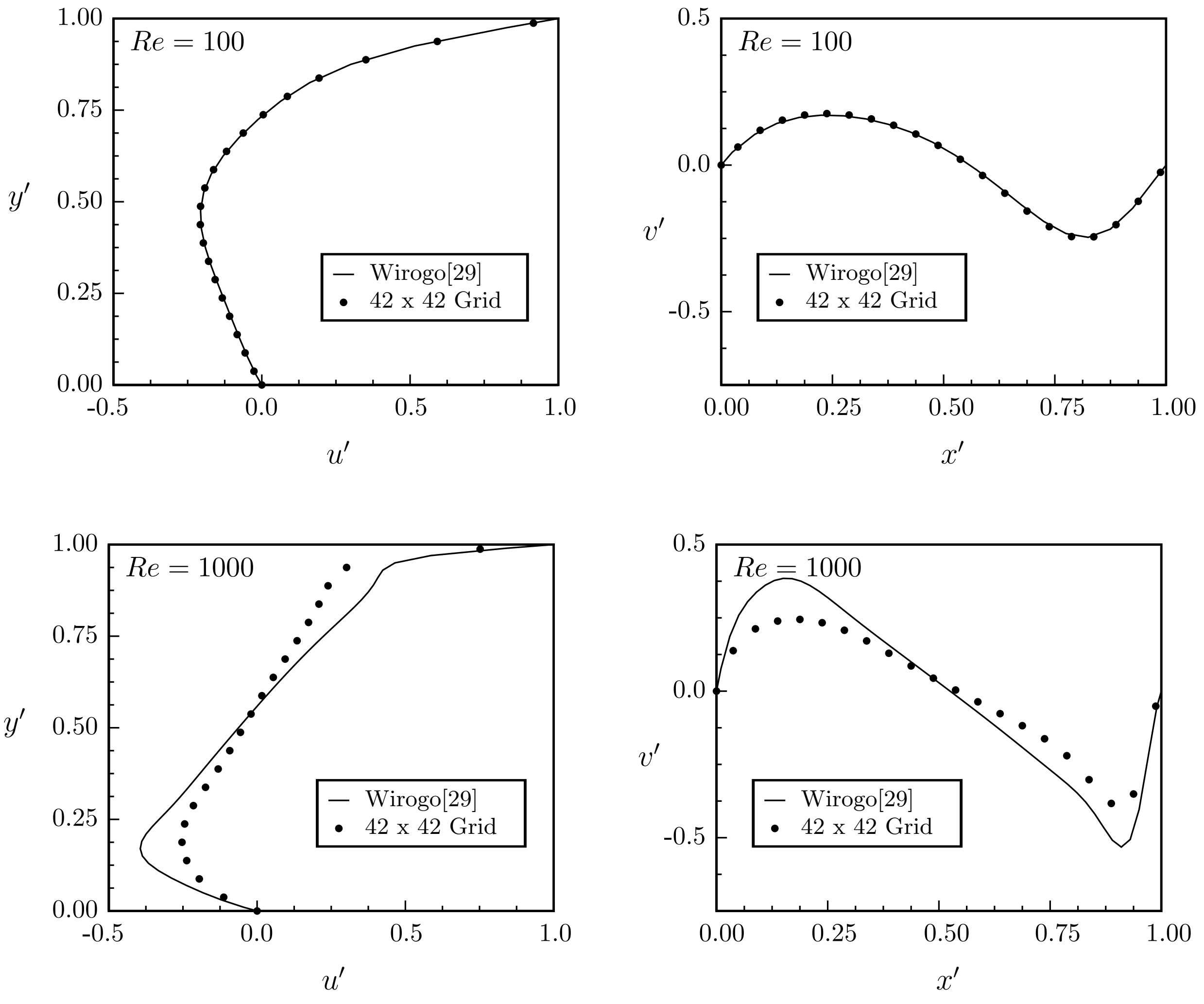


Figure 2.12: Driven cavity velocity profiles compared to Wirogo's results.

The RK-SIMPLER and IRK-SIMPLER algorithms have time step restrictions based on stability constraints. The SIMPLER (as well as the untested SIMPLE) algorithms simulate steady cases without a time step size, using suitable relaxation and sub-iterations. In practice, the maximum allowable time step size for RK- and IRK-SIMPLER is found by incrementally increasing the time step until the algorithm diverges. For driven cavity simulations, the maximum allowable time steps are given in Tables 2.13 and 2.14. The implicit methods allow for larger time step sizes than the explicit method, with the IRK-SIMPLER(B2) method allowing for the largest time step size. The time step size does not have any impact on the steady state solution.

Table 2.13: RK-SIMPLER(A) and IRK-SIMPLER(A) maximum allowable time step size.

	Maximum Allowable Time Step Size (non-dimensional)	
Algorithm	$Re = 100$	$Re = 1,000$
RK-A	0.020	0.054
IRK-A1	0.037	0.140
IRK-A2	0.037	0.140
IRK-A1-AF	0.037	0.140
IRK-A2-AF	0.037	0.140

Table 2.14: RK-SIMPLER(B) and IRK-SIMPLER(B) maximum allowable time step size.

		Maximum Allowable Time Step Size (non-dimensional)	
Algorithm	N_p[2]	$Re = 100$	$Re = 1,000$
RK-B1	5	0.015	0.050
	200	0.015	0.050
RK-B2	5	0.015	0.050
	200	0.015	0.051
RK-B3	5	0.019	0.064
	200	0.019	0.064
IRK-B1	5	0.069	0.490
	200	0.200	0.640
IRK-B2	5	0.110	0.470
	200	0.700	0.480
IRK-B3	5	0.063	0.200
	200	0.470	0.200

For RK-SIMPLER(B) and IRK-SIMPLER(B), the impact of the number of iterations used to solve the pressure equation is of interest. To show the effect of iterations on the pressure equation, Fig. 2.13 shows the residual of the pressure equation with number of iterations. The method used

[2] N_p is the number of pressure equation iterations each stage.

to solve the pressure equation is a line-by-line, Tri-Diagonal Matrix Algorithm (TDMA) method, in which grid lines are solved by TDMA one by one with symmetric Gauss-Seidel marching of grid lines in the y direction, then the x direction. The x and y momentum residuals are shown with number of iterations in Fig. 2.14. The momentum equation's residual drops to near zero in just a few iterations, but the pressure equation converges much more slowly, with a residual drop of three orders of magnitude after about 200 iterations.

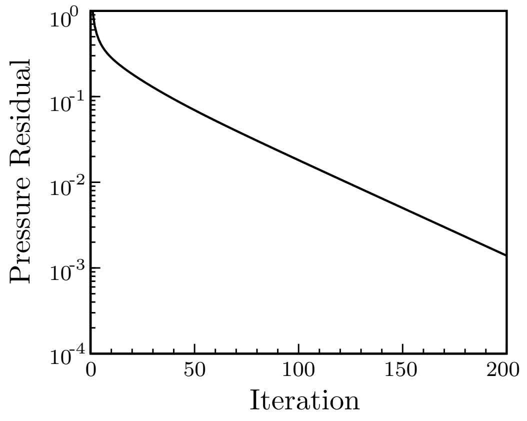


Figure 2.13: Pressure equation residual.

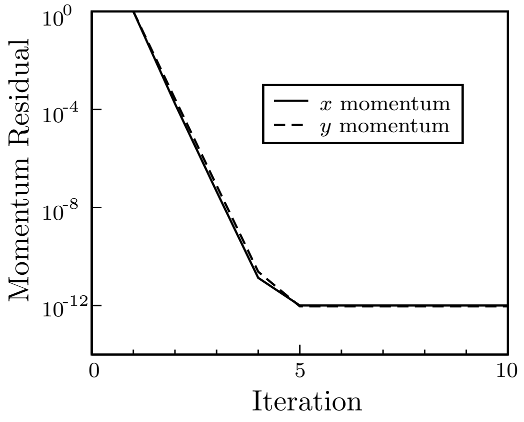


Figure 2.14: Momentum equation residual.

The pressure equation is traditionally solved with five iterations for SIMPLER and RK-SIMPLER(A). For RK-SIMPLER(B), five and 200 iterations are tested for an economical solution of pressure and a more accurately converged solution of pressure, respectively. For IRK-SIMPLER(B), the number of inner iterations used to couple velocity and pressure is fixed at five, and the pressure equation is iterated one and 40 times (resulting in five and 200 pressure iterations per stage respectively). For RK-SIMPLER(B), the pressure equation iterations have little to no effect on the maximum allowable time step size. For IRK-SIMPLER(B), the number of pressure equation iterations significantly impact allowable time step size at Reynolds number of 100, but less of an impact at Reynolds number of 1,000. All IRK-SIMPLER(B) methods have larger allowable time steps than the IRK-SIMPLER(A) methods. Also, the different IRK-SIMPLER(B) methods have large differences in the time step size possible, with IRK-B2 allowing for the largest time step size at Reynolds number of 100, and IRK-B1 allowing for the largest time step size at Reynolds number of 1,000.

The RK algorithms are developed as methods for simulating unsteady flows efficiently. To solve the steady driven cavity, simulations carry on until steady state is reached. The SIMPLER algorithm simulates steady flows by iterating until convergence without time steps needed.

Note that the IRK-SIMPLER(B) results shown here for the driven cavity are all using one momentum coefficient update each stage. The impact of using more than one momentum coefficient update will be examined in later sections.

2.4.2 Temporal Order of Accuracy

The driven cavity problem with a Reynolds number of 100 is used to test the temporal order of accuracy of the algorithms. For each algorithm, the cavity is run to a non-dimensional time of 0.1 over a range of time step sizes, all using the same 42 by 42 uniform grid. To calculate the error of the simulations, an exact solution is needed. To approximate that exact solution, the driven cavity is simulated with SIMPLER at a very fine time step size (1×10^{-7}), using Crank-Nicolson for time integration and 50 sub-iterations every time step. All simulations are compared to the "exact" solution of SIMPLER, and the L2 norm of u error over the domain is computed. Figures 2.15-2.18 show the error versus time step size for all methods tested. The slope of the error gives the order of accuracy for the method. Reference slope lines are shown as solid lines in the figures, and the x-axis is inverted to show decreasing time step size along the positive x-direction.

For SIMPLER, both Fully Implicit (FI) and Crank-Nicolson (C-N) are tested to validate the process used in the analysis. FI is known to be first order accurate, and C-N is second order accurate. Figure 2.15 verifies this procedure, showing FI to be first order and C-N to be second order.

Figure 2.16 shows the error for RK-SIMPLER(A) and IRK-SIMPLER(A) methods. The explicit (RK-A) and implicit (IRK-A1) methods fall on top of each other and are first order accurate. Only IRK-A1 is shown, but all IRK-A methods fall on this same line. For time step sizes lower than 1×10^{-4}, the slope is constant and first order, but for larger time step sizes, the error increases rapidly. This increase in error for large time steps may be an issue of stability of the algorithm.

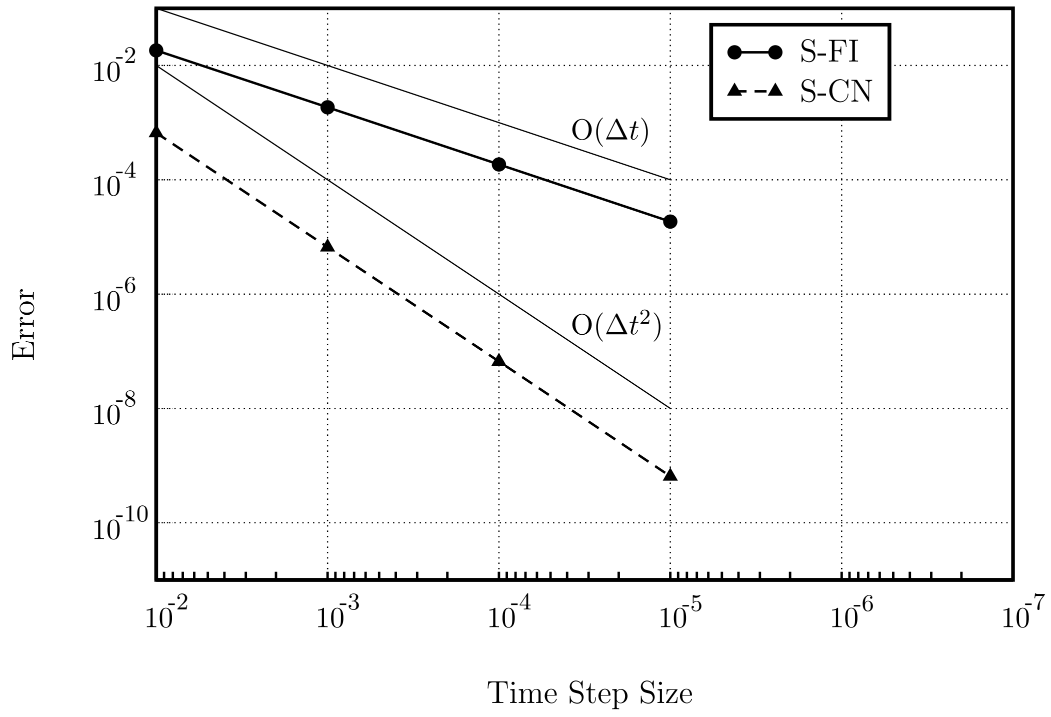


Figure 2.15: Temporal order of accuracy for SIMPLER.

The RK-A and IRK-A methods, while only first order, produce less error than the first order S-FI method for time step sizes lower than 2×10^{-3}.

Figure 2.17 shows the error for RK-SIMPLER(B) methods, with the number of pressure equation iterations per RK stage in parentheses. The RK-B1 method is first order accurate, and the RK-B2 method is second order accurate. Using 200 pressure equation iterations for RK-B1 and RK-B2 has no impact on the order of accuracy and is omitted from Fig. 2.17. The number of pressure equation iterations does have an impact on the RK-B3 order of accuracy; when five iterations are used second order accuracy is achieved, but when 200 iterations are used RK-B3 becomes third order accurate. Therefore, to achieve an order of accuracy higher than second order, the pressure equation must be solved to a lower residual to accurately satisfy continuity during the RK stages. To achieve first or second order accuracy, a few pressure equations iterations are sufficient.

Figure 2.18 shows the error for IRK-SIMPLER(B), with the number of pressure equation iterations per stage in parentheses. These methods behave similar to RK-SIMPLER(B), with IRK-B1

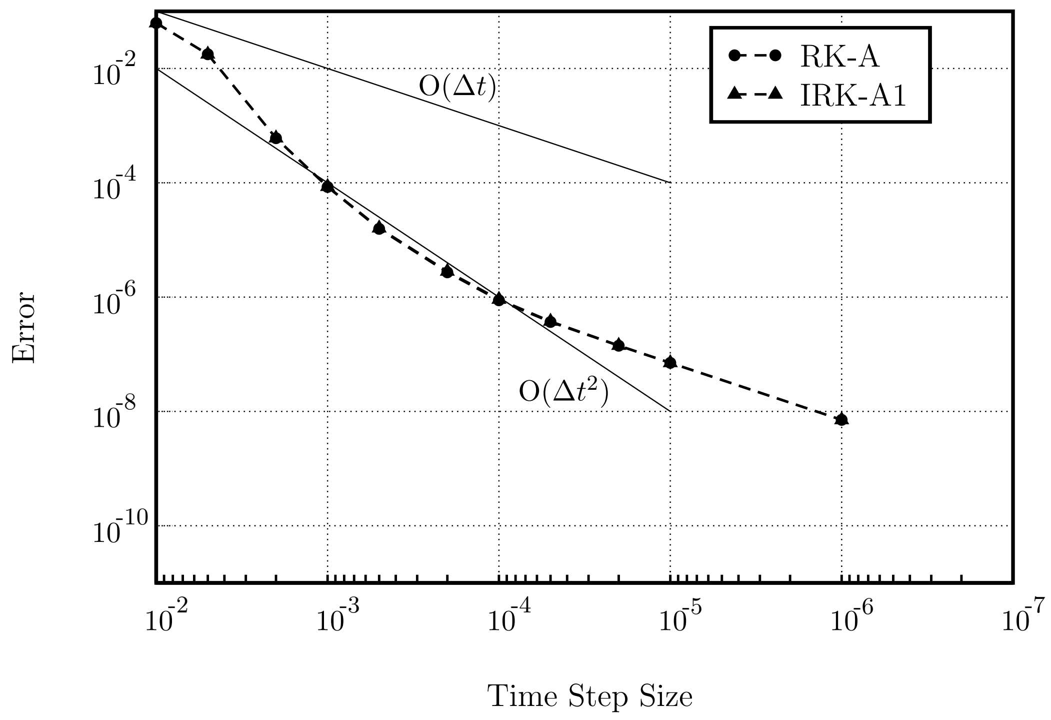


Figure 2.16: Temporal order of accuracy for RK-A and IRK-A.

being first order accurate and IRK-B2 being second order accurate. Again, using 200 pressure equation iterations for IRK-B1 and IRK-B2 has no impact on the order of accuracy, and is omitted from Fig. 2.18. The number of pressure equation iterations impacts IRK-B3, as was found for RK-B3. With five iterations IRK-B3 is second order accurate, but with 200 iterations IRK-B3 becomes third order accurate. As seen in Figs. 2.17 and 2.18, for RK-B3 and IRK-B3 with 200 pressure equation iterations, the slope of the error becomes less at the smallest time steps, due to numerical limits (i.e. the computations have become as accurate as possible given double precision floating numbers).

Table 2.15 presents the order of accuracy for all methods tested. Although not shown, the order of accuracy of the v-velocity component follows the u-velocity component. This is because the RK methods used for the x and y momentum equations are the same, so the accuracy of both u and v are the same.

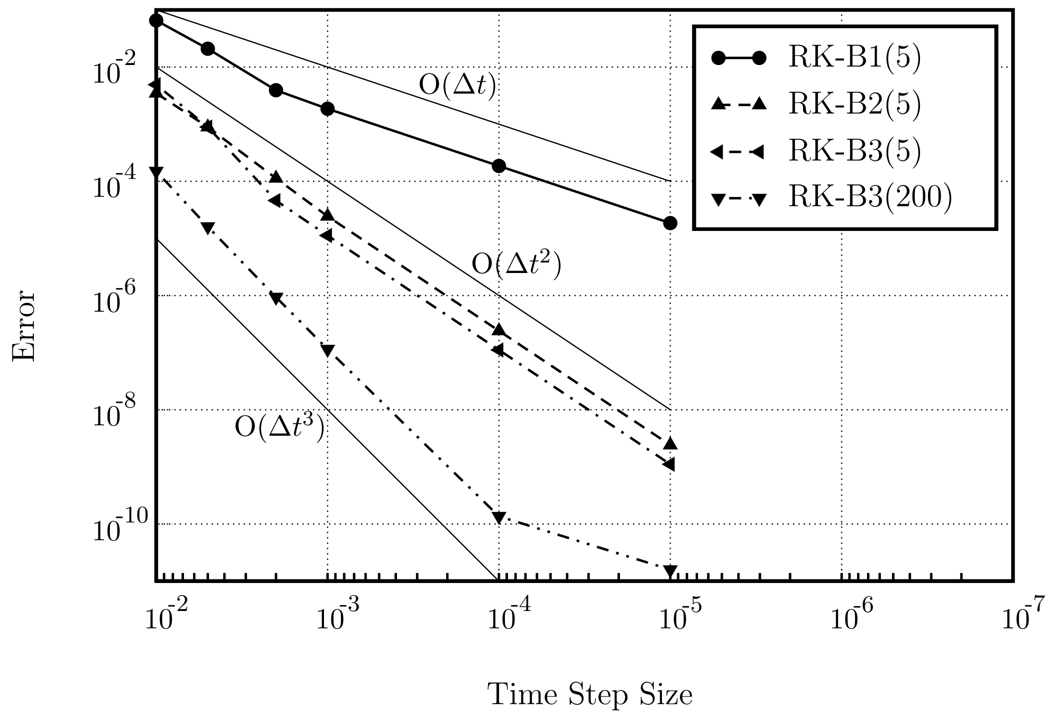


Figure 2.17: Temporal order of accuracy for RK-B.

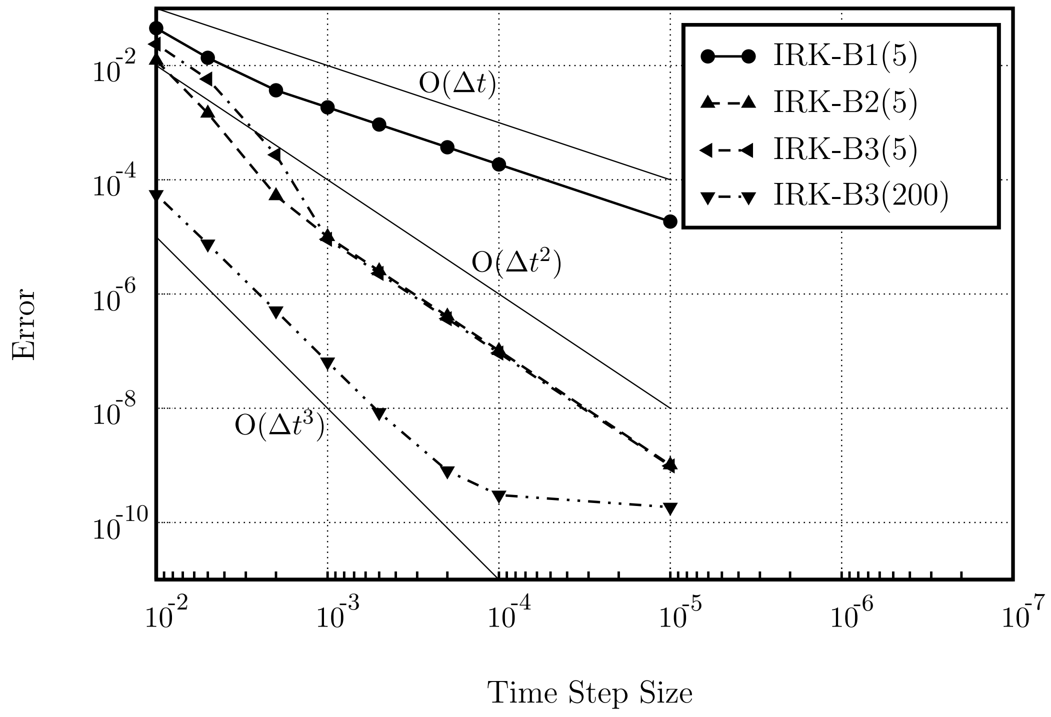


Figure 2.18: Temporal order of accuracy for IRK-B.

Table 2.15: Temporal Order of Accuracy

Algorithm	Order of Accuracy
S-FI	O(Δt)
S-CN	O(Δt^2)
RK-A	O(Δt)
RK-B1	O(Δt)
RK-B2	O(Δt^2)
RK-B3(5)	O(Δt^2)
RK-B3(200)	O(Δt^3)
IRK-A	O(Δt)
IRK-B1	O(Δt)
IRK-B2	O(Δt^2)
IRK-B3(5)	O(Δt^2)
IRK-B3(200)	O(Δt^3)

2.4.2.1 Additional Iterations

RK-SIMPLER(A), IRK-SIMPLER(A), and IRK-SIMPLER(B) each have approximations in their formulation that can be reduced through iterations. The RK-SIMPLER(A) and IRK-SIMPLER(A) algorithms use sub-iterations, similar to SIMPLER, to solve the pressure equation and update the RK form of the momentum equations multiple times in each time step. These iterations attempt to improve the Crank-Nicolson form of the pressure equation by including implicit terms instead of terms from the last time step. The error versus time step size for the RK-SIMPLER(A) and IRK-SIMPLER(A) algorithms with iterations are shown in Figs. 2.19 and 2.20. The iterations do not improve the order of accuracy of these methods, but the error at large time step sizes decreases with iterations, making the first order slope present for higher time steps. Interestingly, the iterations do not affect the error for small time steps, effectively only adding runtime to the simulation. Therefore, additional iterations are not suggested.

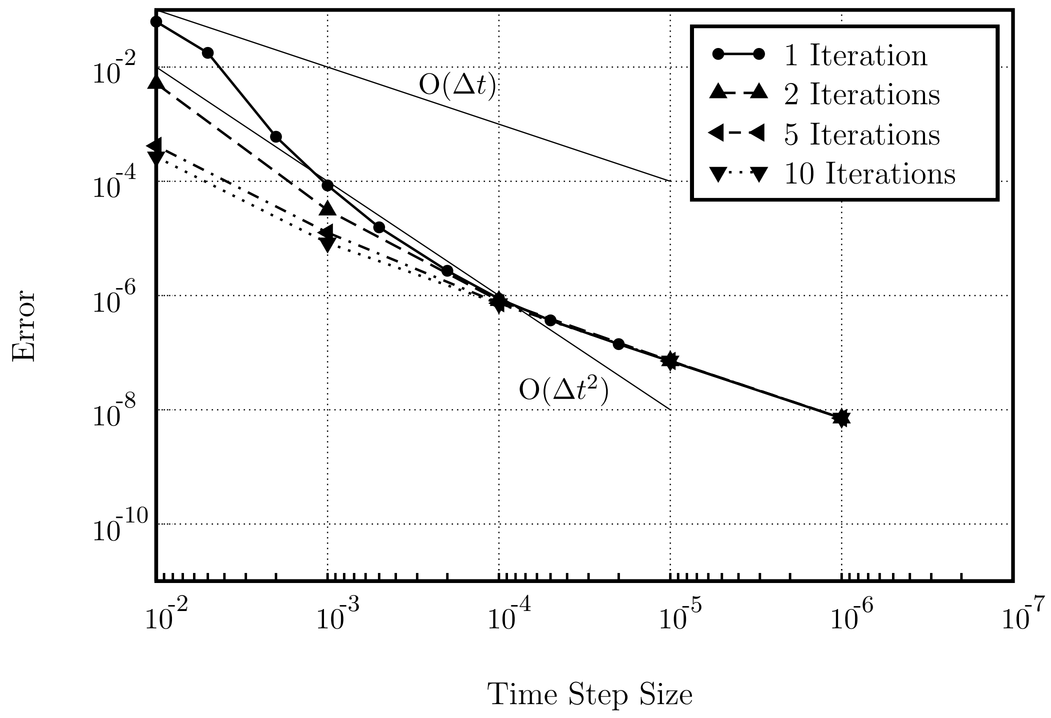


Figure 2.19: Temporal order of accuracy of RK-SIMPLER(A) with iterations.

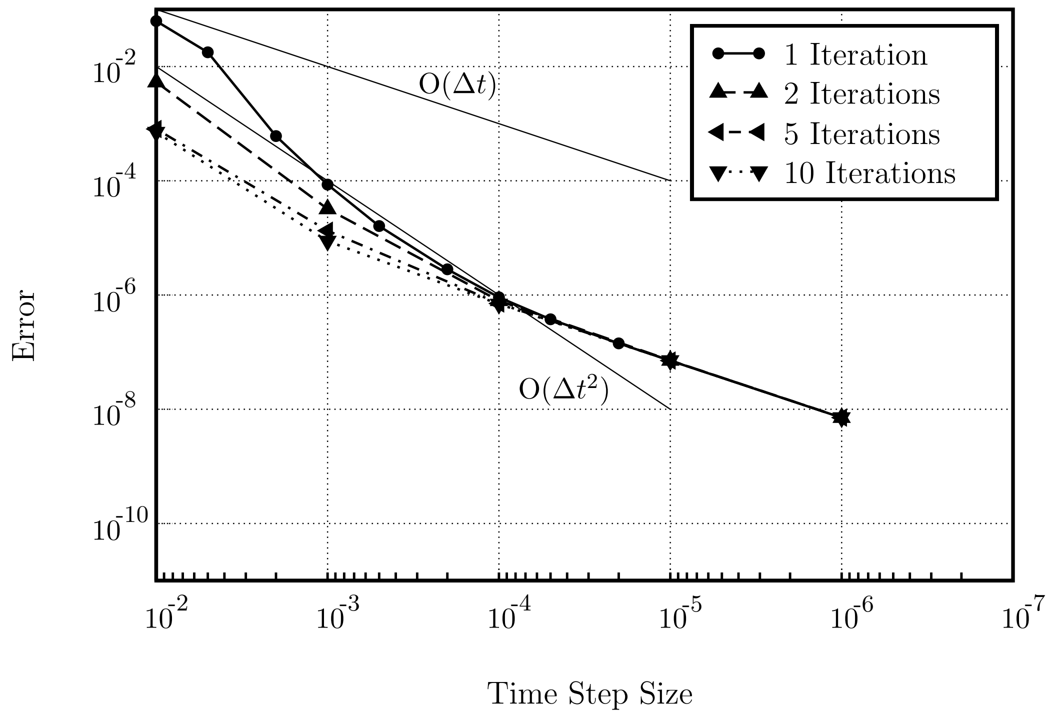


Figure 2.20: Temporal order of accuracy of IRK-SIMPLER(A) with iterations.

The IRK-SIMPLER(B) algorithm is iterated each stage to update the implicit momentum coefficients. These iterations occur after the implicit momentum update and do not require additional implicit momentum equation solutions. The temporal order of accuracy of IRK-B2 with iterations is shown in Fig. 2.21. The order of accuracy does not change with iterations; only the magnitude of error decreases for all time steps. The other IRK-SIMPLER(B) variations follow a similar pattern. The extra iterations in IRK-SIMPLER(B) to update the momentum coefficients reduce the error for all time step sizes and are examined in the following section to examine see if efficiency can be improved.

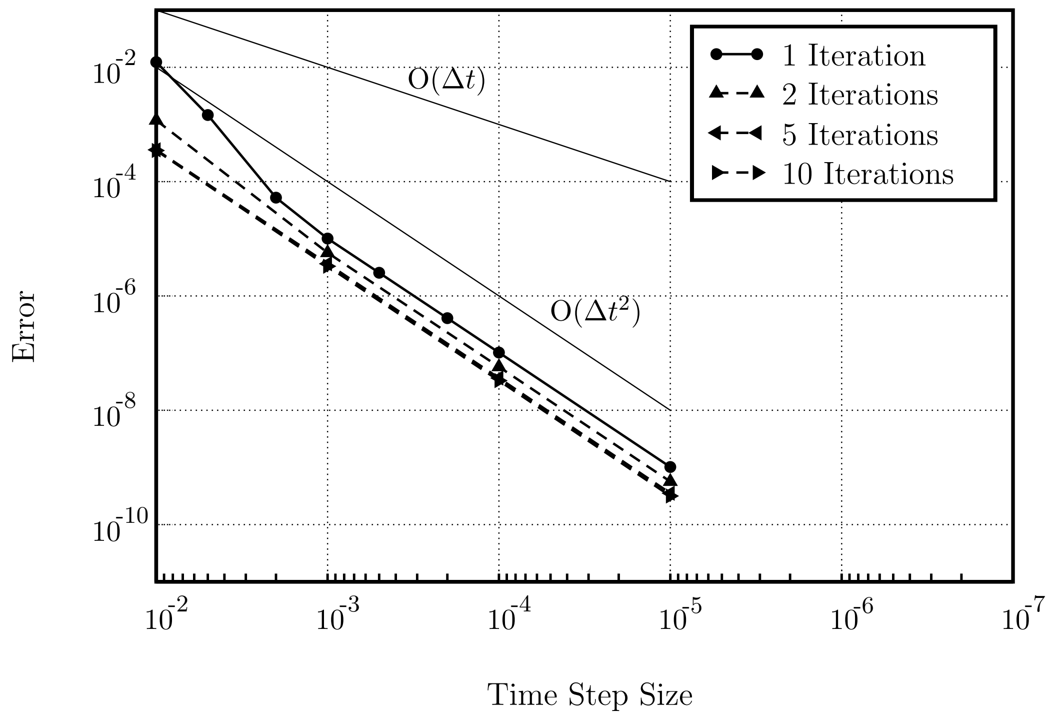


Figure 2.21: Temporal order of accuracy of IRK-B2 with iterations.

2.4.3 Unsteady Simulation

2.4.3.1 Thin Flat Plate Normal to the Flow

To test the algorithms on an unsteady problem, laminar flow over a thin, flat plate normal to the flow direction is simulated. A schematic of the flat plate problem is shown in Fig. 2.22. For moderate to high Reynolds numbers, vortices shed in a cyclical pattern from the top and bottom of the flat plate. For the present simulations a Reynolds number of 17,800 is used (following Lestari [53]). The grid is 172 by 172 with 20 grid cells along the height (L) and 10 grid cells along the width (w) of the plate, which follows Purohit [52], and results in flow characteristics that agree with previous simulation results [53]. The flat plate surfaces are no-slip walls, the left boundary is uniform inflow, the right boundary is velocity outlet corrected for mass conservation, and the top and bottom boundaries are inviscid walls. Simulations are started impulsively with initial conditions of freestream velocity and uniform pressure field, and are run until $t/(L/U_i) = 400$. The traditional SIMPLER algorithm, with Crank-Nicolson time integration, is the baseline case, with the number of sub-iterations within each time step fixed at 20. For the sake of runtime, the pressure equation is solved with five iterations for all first and second order accurate methods, while the IRK-B3 method is tested with five and 200 total pressure equation iterations at each stage.

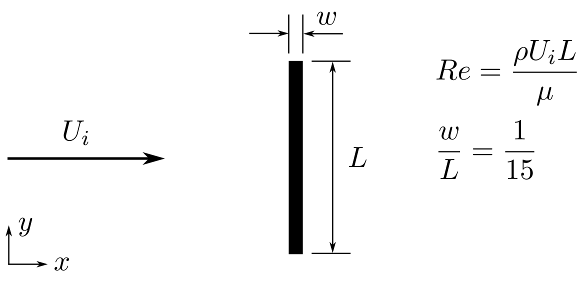


Figure 2.22: Schematic of the thin flat plate normal to the flow problem.

The coefficients of drag (C_d) and lift (C_l) are plotted versus time in Fig.2.23. After some time, the flat plate starts to shed vortices (seen as a steep rise in drag and oscillation in lift). Eventually, lift and drag oscillate with a constant amplitude and frequency. Once the simulation reaches this constant oscillatory pattern, the flow reaches unsteady convergence.

The time at which the flow reaches unsteady convergence varies with the algorithm used and time step size. Figure 2.24 shows the coefficient of drag versus time for several different methods and time step sizes. The time at which shedding starts changes between the different methods, and

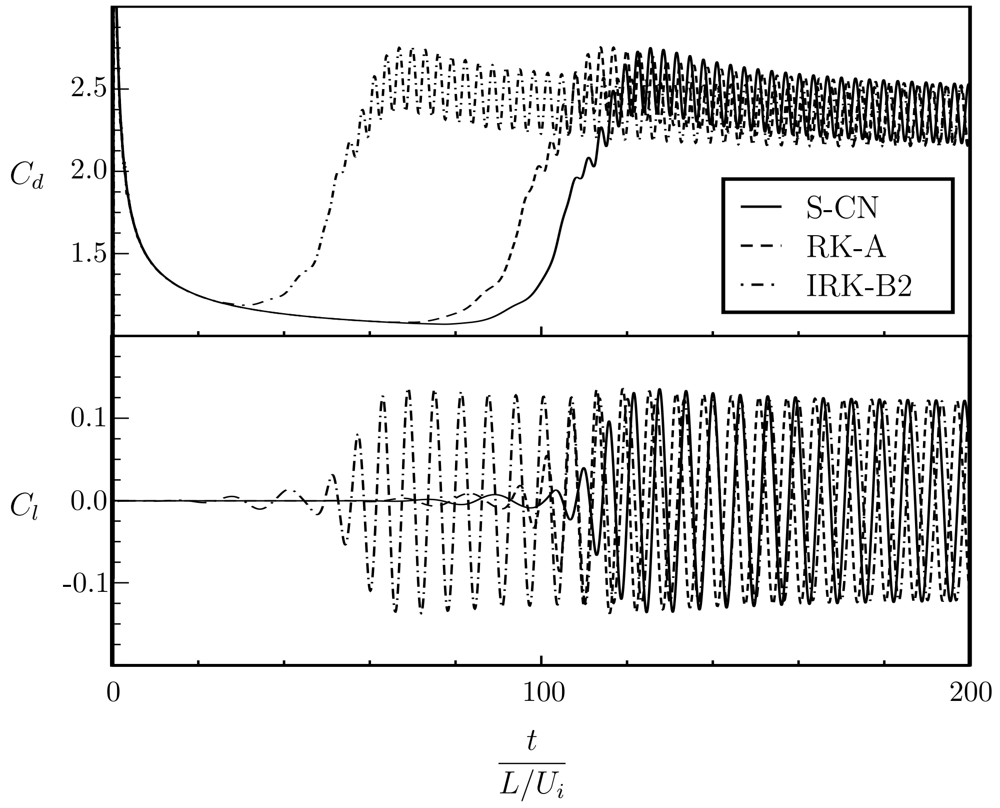


Figure 2.23: Time history of the coefficient of drag and lift on the flat plate.

even for the same method with different time step sizes. With increasing numbers of sub-iterations and decreasing time step sizes, shedding occurs later in all figures.

Because different algorithms begin shedding at different times, the lift and drag coefficients do not oscillate at the same simulation time. To compare the results for the different algorithms, the data is shifted so time equals zero for a peak of lift or drag coefficient. After time shifting the data, the different algorithms fall on top of each other (Fig. 2.25).

A measure of unsteady vortex shedding is the Strouhal number ($Sr = Lf/U_i$), where f is the frequency of shedding. When a vortex sheds from the top of the flat plate, there is a peak in drag and lift. When a vortex sheds from the bottom of the flat plate, there is a peak in drag and a trough in lift. Therefore, the frequency of lift is half of drag, and the frequency of shedding is calculated by finding the period between alternating peaks of drag or by finding the period between consecutive peaks of lift.

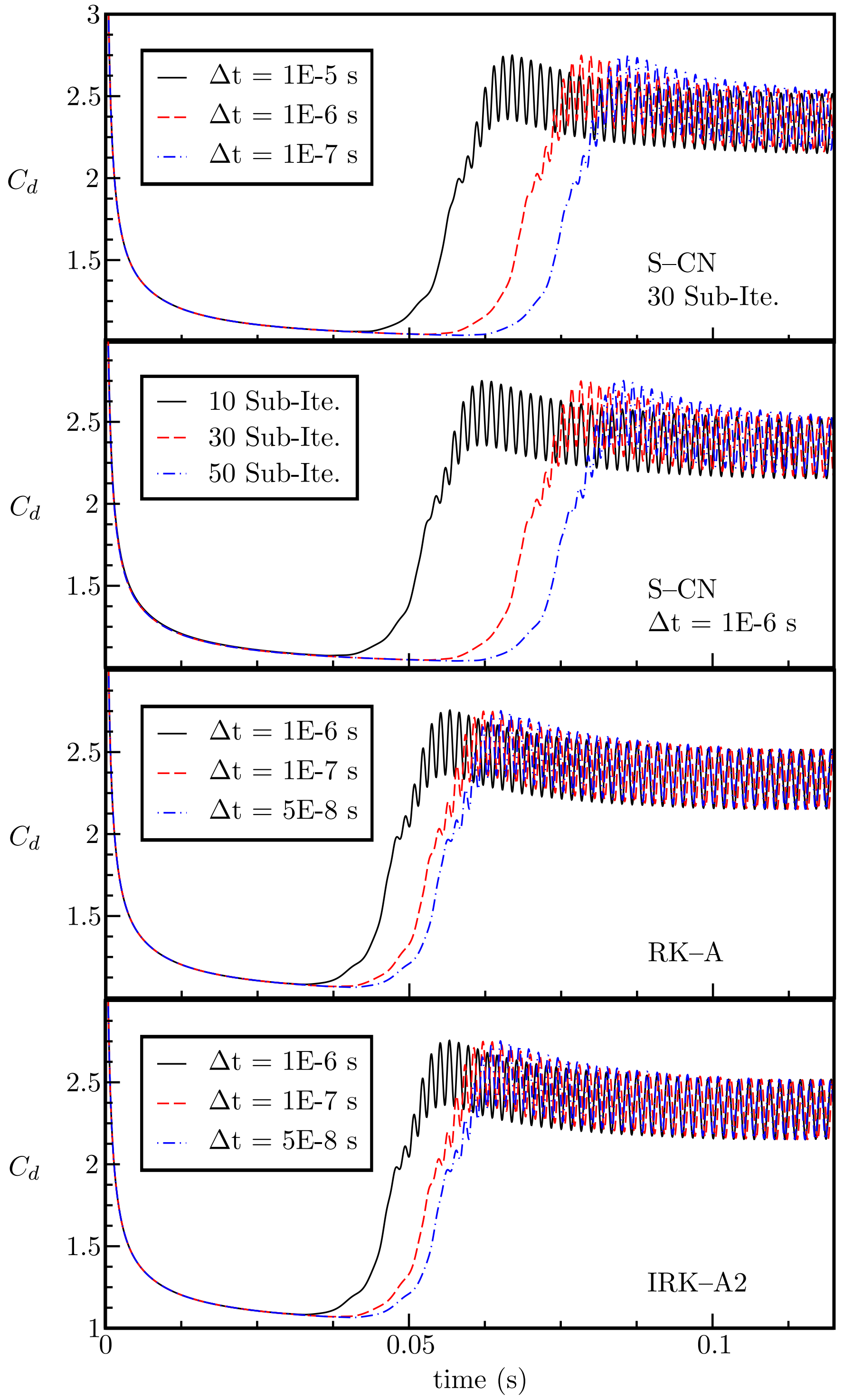


Figure 2.24: Drag history of flat plate with different methods.

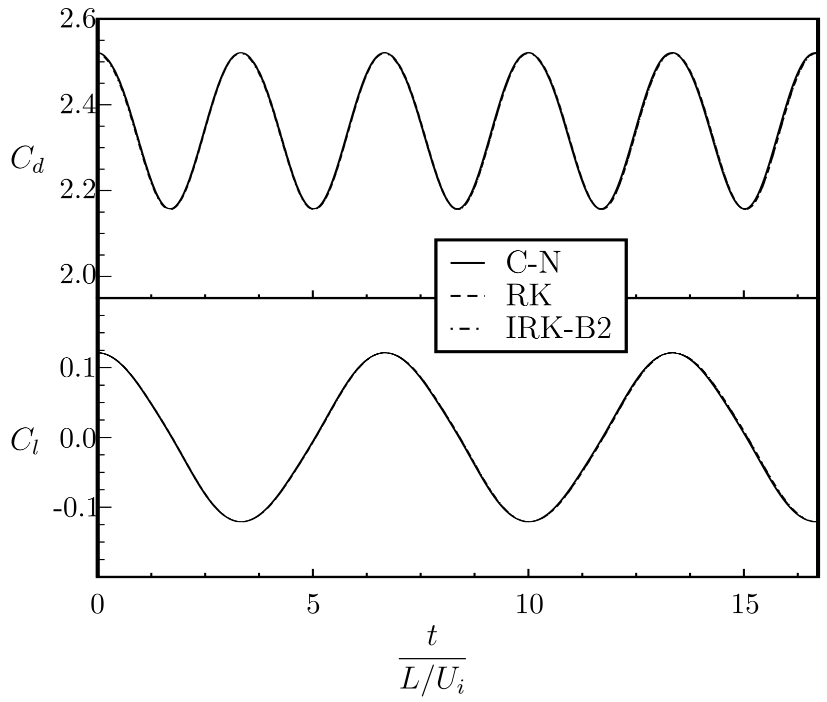


Figure 2.25: Coefficient of drag and lift in the region of cyclic shedding of vortices.

Figure 2.26 shows the average coefficient of drag for different algorithms over a range of time step sizes, with IRK-B methods using $N_{up} = 1$. As time step size decreases, all methods approach an average coefficient of drag around 2.3385. C-N and IRK-B1 both converge for large time step sizes, but are not accurate. RK and IRK-A converge only for small time step sizes, but are accurate for all allowable time step sizes. IRK-B2 and IRK-B3 converge for relatively large time step sizes and are accurate for all time step sizes allowed.

Table 2.16 shows the maximum allowable time step (Δt_{max}), as well as the average coefficient of drag, $\overline{C_d}$, and Strouhal number using the maximum allowable time step size. Table 2.16 also shows the maximum time step for which the average coefficient of drag is 2.3385±0.002, or within 0.1% of the converged value (defined as the accurate time step size Δt_{Acc}). The runtime and speedup using the accurate time step sizes are also listed in the table. IRK-SIMPLER variation B is tested with different numbers of momentum coefficient updates ($N_{up} = 1, 2, 3$). Using $N_{up} > 1$ allows for larger Δt_{max} and Δt_{Acc} than using $N_{up} = 1$ for all methods except IRK-B1. For all methods, the Strouhal number is 0.15 when the average drag coefficient falls within the desired accuracy.

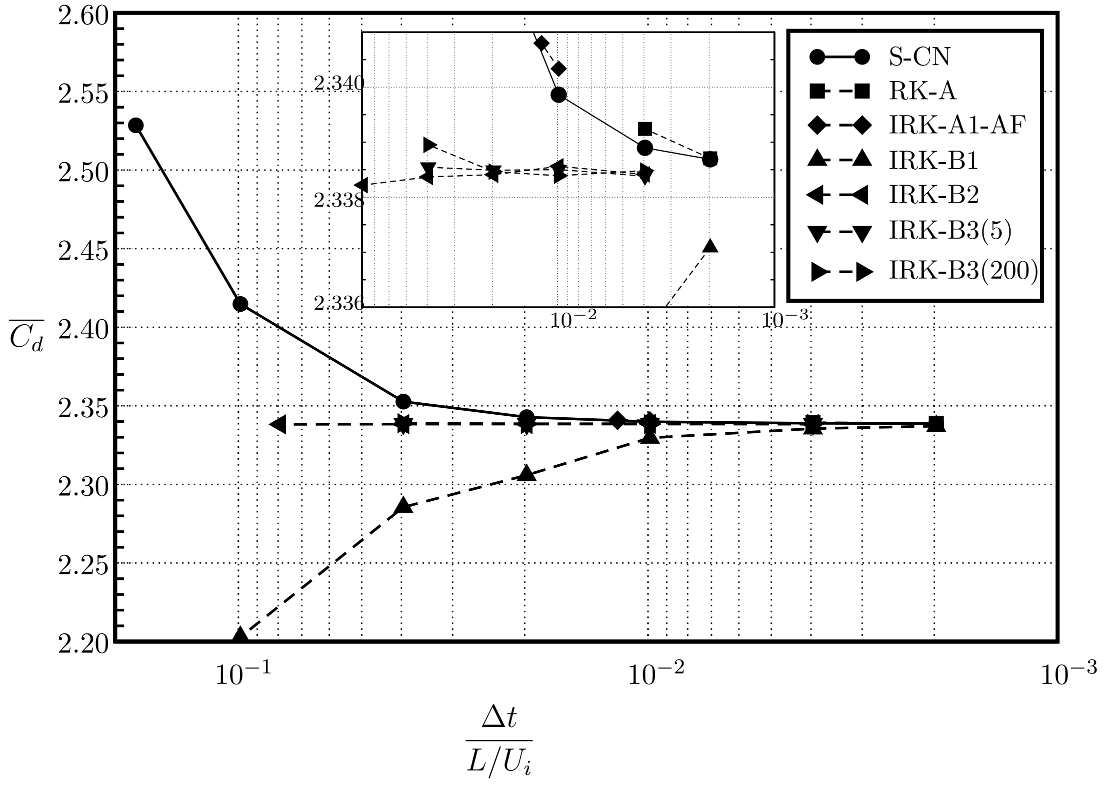


Figure 2.26: Average coefficient of drag for different time steps sizes[3].

Previous computational [53],[54],[55] and experimental [55] results on the flat plate indicate the average coefficient of drag and Strouhal numbers are in good agreement with current results.

2.4.4 Algorithm Conclusions and Recommendations

Both the RK-SIMPLER and IRK-SIMPLER algorithms show improvement over the traditional SIMPLER algorithm with C-N time integration. The explicit RK-SIMPLER algorithm requires relatively few computations per time step and, despite the low time step required to converge, result in accurate solutions in less runtime than SIMPLER. Variation A with ERK-LS4 is only first order accurate in time, but, compared to the other RK-SIMPLER methods, results in the lowest runtime to achieve a time accurate solution of the flat plate problem. The RK-SIMPLER(B) algorithm achieves the same order of accuracy as the ERK method chosen, but the additional computations

[3]IRK-B results using $N_{up} = 1$.

Table 2.16: Flat plate results.

	N_{up}	$\frac{\Delta t_{max}}{L/U_i}$	$\overline{C_d}$ at Δt_{max}	Sr at Δt_{max}	$\frac{\Delta t_{Acc}}{L/U_i}$	CPU Time at Δt_{Acc} (min.)	Speedup[4] at Δt_{Acc}
S-CN	—	0.178	2.528	0.149	0.010	317.41	1.0
RK-A	—	0.004	2.339	0.150	0.004	20.54	15.5
RK-B1	—	0.002	2.340	0.150	0.002	40.21	7.9
RK-B2	—	0.002	2.338	0.150	0.002	57.03	5.6
RK-B3	—	0.002	2.333	0.150	0.002	87.93	3.6
IRK-A1	—	0.012	2.341	0.149	0.010	11.55	27.5
IRK-A2	—	0.012	2.341	0.149	0.010	18.23	17.4
IRK-A1-AF	—	0.012	2.341	0.149	0.010	8.38	37.9
IRK-A2-AF	—	0.012	2.341	0.149	0.010	10.17	31.2
IRK-B1	1	0.099	2.203	0.146	0.002	68.42	4.6
	2	0.198	1.991	0.137	0.002	103.67	3.1
	3	0.593	1.607	0.107	0.002	129.66	2.4
IRK-B2	1	0.079	2.338	0.150	0.079	3.65	87.0
	2	0.198	2.343	0.150	0.119	3.15	100.8
	3	0.198	2.344	0.150	0.119	5.06	62.7
IRK-B3(5)	1	0.040	2.339	0.150	0.040	11.59	27.4
	2	0.138	2.335	0.150	0.119	4.53	70.1
	3	0.198	2.334	0.150	0.138	6.51	48.8
IRK-B3(200)	1	0.040	2.339	0.150	0.040	125.98	2.5
	2	0.198	2.337	0.150	0.198	54.20	5.9
	3	0.593	2.330	0.149	0.198	73.20	4.3

required make the algorithm less efficient than RK-SIMPLER(A). Variation A is the most efficient method and is recommended. It is used for all future RK-SIMPLER simulations unless otherwise noted.

The implicit IRK-SIMPLER algorithm allows for larger time steps than RK-SIMPLER and, although more computations are required, achieves time accurate results in less runtime. IRK-SIMPLER(A) is only first order accurate in time, but requires less runtime than RK-SIMPLER(A) or RK-SIMPLER(B). IRK-SIMPLER(B) achieves the same order of accuracy as the DIRK method

[4]Speedup is relative to C-N.

used. IRK-B2 results in the lowest runtime of all the methods for the flat plate case and is recommended. It is used for all future IRK-SIMPLER simulations unless otherwise noted.

To achieve third order accuracy with RK-SIMPLER(B) and IRK-SIMPLER(B), more iterations are required. More iterations lowers the efficiency of the algorithm, and results in longer runtime compared to the second order accurate methods.

CHAPTER 3. UNSTRUCTURED GRID FORMULATION

For many problems, creating a structured grid is challenging and time consuming, particularly for problems with complex geometries. To alleviate this challenge, unstructured grids are used to fill the domain using arbitrary shaped grid cells. The IRK-SIMPLER algorithm has been tested and validated on the simple case of structured Cartesian grids, but the behavior of IRK-SIMPLER on unstructured grids is of interest. To test IRK-SIMPLER, two-dimensional, triangular vertex-centered grids and three-dimensional, tetrahedral vertex-centered grids are used. This chapter covers the formulation of IRK-SIMPLER on these grids and results comparing IRK-SIMPLER to SIMPLER and RK-SIMPLER.

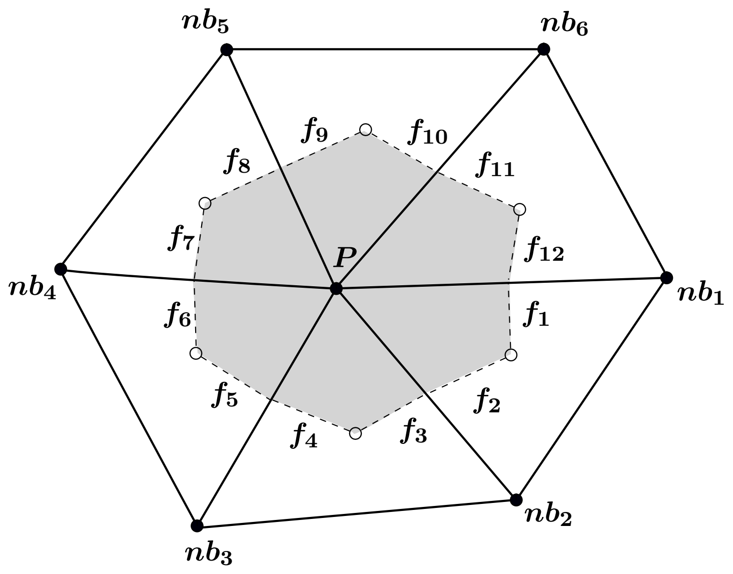
3.1 Two-Dimensional Triangular Grid Formulation

Rewriting the governing equations for incompressible fluid flow as

$$\frac{\partial(\rho\vec{V})}{\partial t} + \nabla\cdot(\rho\vec{V}\vec{V}) = -\nabla p + \nabla\cdot(\mu\nabla\vec{V}) + \vec{S} \tag{3.1}$$

$$\nabla\cdot(\rho\vec{V}) = 0 \; . \tag{3.2}$$

In two dimensions, a triangular, vertex-centered, median-dual control volume formulation is used following the work of Maresca [56] and Lestari [53] (see Fig. 3.1). Each triangle is an element characterized by three nodes, three edges, and three interior faces. Figure 3.2 shows a triangular element with the nodes labeled n_i, the edge midpoints labeled e_i, the element center labeled c, and the interior face midpoints labeled m_i. Each node has a control volume made up of quadrilateral areas from all the triangular elements it connects to. The faces of the control volumes are the interior faces of the elements (lines connecting edge midpoints to element centroids) as well as the element edges in the case of boundary elements (Fig. 3.3). For this formulation, density and viscosity are constant within each triangular element and pressure varies linearly.



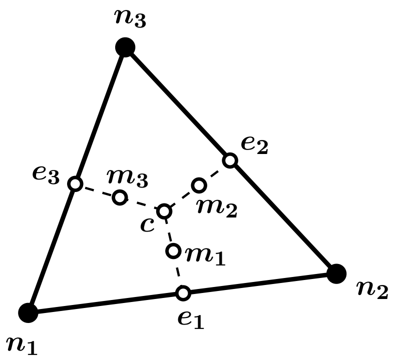


Figure 3.2: Triangular element.

Figure 3.1: Triangular median-dual control volume.

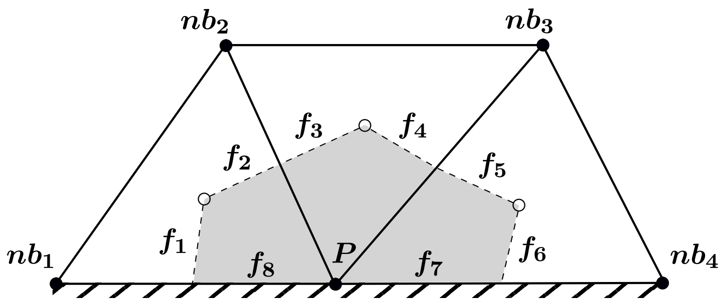


Figure 3.3: Control volume of a boundary node with boundary faces at triangle edges.

3.1.1 Interpolation of Velocity

The interpolation method of Prakash [31] interpolates the velocities over each triangular element to avoid spurious pressure oscillation when developing the pressure equation (a common problem when using collocated grids). The velocity interpolation uses fully discrete momentum equations to interpolate the velocity components to the control volume faces. Starting with the x momentum equation in the form

$$u_n = \hat{u}_n - (d_u)_n \left(\frac{\partial p}{\partial x}\right)_n , \tag{3.3}$$

where the index n represents the node of interest. The details of how to get all the terms in this equation is discussed later for each algorithm, but if we assume this form, an artificial velocity term $(\tilde{u})$ is defined by

$$\tilde{u}_{e-n} = \hat{u}_n - (d_u)_n \left(\frac{\partial p}{\partial x}\right)_e \; , \tag{3.4}$$

where the index e represents the element of interest, and the subscript on the artificial velocity $e-n$ represents the value at node n for the specific element e. The artificial velocity at a given node is different when looking at each element because the element pressure gradient is used instead of the nodal gradient. The linear interpolation of the artificial velocity across the triangular element is used any time a velocity value is needed on the interior of a triangle. This interpolation is particularly important when deriving the pressure equation, where checkerboarding of pressure can occur if the same interpolation is used for both pressure and velocity. To find the artificial velocity in the interior of each element, a linear interpolation is used (using the notation from Fig. 3.2).

$$\tilde{u}_{e_1} = \frac{1}{2}\left(\tilde{u}_{n_1} + \tilde{u}_{n_2}\right) \qquad \tilde{u}_{e_2} = \frac{1}{2}\left(\tilde{u}_{n_2} + \tilde{u}_{n_3}\right) \qquad \tilde{u}_{e_3} = \frac{1}{2}\left(\tilde{u}_{n_3} + \tilde{u}_{n_1}\right) \tag{3.5}$$

$$\tilde{u}_{c} = \frac{1}{3}\left(\tilde{u}_{n_1} + \tilde{u}_{n_2} + \tilde{u}_{n_3}\right) \tag{3.6}$$

$$\tilde{u}_{m_1} = \frac{1}{2}\left(\tilde{u}_{e_1} + \tilde{u}_{c}\right) \qquad \tilde{u}_{m_2} = \frac{1}{2}\left(\tilde{u}_{e_2} + \tilde{u}_{c}\right) \qquad \tilde{u}_{m_3} = \frac{1}{2}\left(\tilde{u}_{e_3} + \tilde{u}_{c}\right) \tag{3.7}$$

The y direction artificial velocity is interpolated in the same manner with

$$\tilde{v}_{e-n} = \hat{v}_n - (d_v)_n \left(\frac{\partial p}{\partial y}\right)_e \; . \tag{3.8}$$

This method of interpolation, called equal order interpolation by Prakash [31], is similar to Rhie-Chow interpolation [57] and Modified Momentum Interpolation Method [58]. These methods all attempting to avoid pressure checkerboarding that occurs when collocated pressure and velocity are both interpolated using the same method. When using the interpolation stencil described above, at the first time step (when the domain is initialized) setting $\hat{u} = u_{init}$, $\hat{v} = v_{init}$, and $d_u = d_v = 0$ works well. The quantities $\hat{u}$, $\hat{v}$, d_u and d_v are updated every time step, each sub-iteration in SIMPLER, and each time momentum coefficients are updated in IRK-SIMPLER(B).

3.1.2 Triangular Element Momentum Flux Computation

The momentum fluxes are calculated using an interpolation shape function developed by Baliga [30], which is similar to the Power Law scheme that Patankar [5] developed for structured grids. The unstructured scheme of Baliga solves the two-dimensional,steady convection-diffusion equation for a general variable ϕ with no source, given by

$$\frac{\partial \rho u_e \phi}{\partial x} + \frac{\partial \rho v_e \phi}{\partial y} - \frac{\partial}{\partial x}\left[\mu \frac{\partial \phi}{\partial x}\right] - \frac{\partial}{\partial y}\left[\mu \frac{\partial \phi}{\partial y}\right] = 0 \ , \tag{3.9}$$

where u_e and v_e are the element centroid velocity components for x and y directions, respectively.

$$u_e = \frac{1}{3}(u_{n_1} + u_{n_2} + u_{n_3}) \qquad\qquad v_e = \frac{1}{3}(u_{n_1} + u_{n_2} + u_{n_3}) \tag{3.10}$$

A rotational transformation aligns a new coordinate direction, X, to the element velocity direction, $U_e = \sqrt{u_e^2 + v_e^2}$ (see Fig. 3.4).

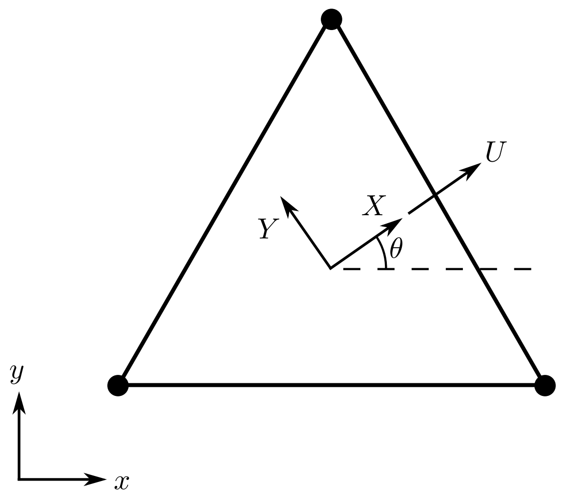


Figure 3.4: Triangular element with the element velocity vector shown.

The local coordinate system is found by translating the origin to the element centroid and rotating the X direction to match the element average velocity vector direction. The rotation matrix coefficients for angle θ are given by

$$\cos\theta = \frac{u_e}{U_e} \qquad\qquad \sin\theta = \frac{v_e}{U_e} \ . \tag{3.11}$$

In the event $U_e = 0$, the rotation angle is set to zero (i.e. $\cos\theta = 1$ and $\sin\theta = 0$). The transformation from (x, y) to the element local coordinate system is given by

$$X = (x - x_e)\cos\theta + (y - y_e)\sin\theta \tag{3.12}$$

$$Y = -(x - x_e)\sin\theta + (y - y_e)\cos\theta \ , \tag{3.13}$$

and the transformation for velocity is

$$U = u\cos\theta + v\sin\theta \tag{3.14}$$

$$V = -u\sin\theta + v\cos\theta \ . \tag{3.15}$$

The convection-diffusion equation is written for the new coordinate system as

$$\frac{\partial \rho U_e \phi}{\partial X} - \frac{\partial}{\partial X}\left[\mu \frac{\partial \phi}{\partial X}\right] - \frac{\partial}{\partial Y}\left[\mu \frac{\partial \phi}{\partial Y}\right] = 0 \ . \tag{3.16}$$

To interpolate across the triangle, a shape function is assumed that varies linearly in the Y direction and varies by some function $\xi(X)$ in the X direction.

$$\phi = A\xi(X) + BY + C \ , \tag{3.17}$$

where the values of A, B, and C are constant for each element and are given in summation form as

$$A = L_i\phi_i \qquad B = M_i\phi_i \qquad C = N_i\phi_i \ , \tag{3.18}$$

with $i = 1, 2, 3$ for the three triangle vertices. The terms L, M, and N are given by

$$\begin{aligned} L_1 &= (Y_2 - Y_3)/\Delta & L_2 &= -(Y_1 - Y_3)/\Delta & L_3 &= (Y_1 - Y_2)/\Delta \\ M_1 &= -(\xi_2 - \xi_3)/\Delta & M_2 &= (\xi_1 - \xi_3)/\Delta & M_3 &= -(\xi_1 - \xi_2)/\Delta \\ N_1 &= (\xi_2 Y_3 - \xi_3 Y_2)/\Delta & N_2 &= -(\xi_1 Y_3 - \xi_3 Y_1)/\Delta & N_3 &= (\xi_1 Y_2 - \xi_2 Y_1)/\Delta \ , \end{aligned} \tag{3.19}$$

with

$$\Delta = \xi_1(Y_2 - Y_3) - \xi_2(Y_1 - Y_3) + \xi_3(Y_1 - Y_2) \ . \tag{3.20}$$

The exact solution of $\xi(X)$ is the exponential function [30]

$$\xi(X) = \frac{\Delta X}{P_e}\left[\exp\left(\frac{P_e(X - X_{max})}{\Delta X}\right) - 1\right] \ , \tag{3.21}$$

with

$$\Delta X = X_{max} - X_{min} \qquad P_e = \frac{\rho U_e \Delta X}{\mu} \tag{3.22}$$

$$X_{max} = \max(X_{n_1}.X_{n_2}, X_{n_3}) \qquad X_{min} = \min(X_{n_1}.X_{n_2}, X_{n_3}) \ . \tag{3.23}$$

Because exponential functions are computationally costly to execute, a power law representation of the function $\xi(X)$ is used [30].

$$\xi(X) = \frac{X - X_{max}}{P_e + [\![0, (1 - 0.1|P_e|)^5]\!]} \tag{3.24}$$

This scheme of Baliga is called the Power Law scheme for unstructured triangular grids, and has also been successfully used by Maresca [56] and Lestari [53].

The interpolation of ϕ is rewritten in summation form as

$$\phi = [L_i \xi(X) + M_i Y + N_i]\, \phi_i \ , \tag{3.25}$$

with derivatives (using the exponential function 3.21)

$$\frac{\partial \phi}{\partial X} = \left[\frac{\rho U_e}{\mu}\xi + 1\right] L_i \phi_i \tag{3.26}$$

$$\frac{\partial \phi}{\partial Y} = M_i \phi_i \ . \tag{3.27}$$

In this summation form, repeated indices represent a summation over the three nodes (i.e. $L_i\phi_i = L_1\phi_1 + L_2\phi_2 + L_3\phi_3$).

The total convective and diffusive momentum equation flux is written as

$$\begin{aligned} J_{\phi-X} &= \rho U \phi - \mu_e \frac{\partial \phi}{\partial X} \\ &= f_i \phi_i \end{aligned} \tag{3.28}$$

$$\begin{aligned} J_{\phi-Y} &= \rho V \phi - \mu_e \frac{\partial \phi}{\partial Y} \\ &= g_i \phi_i \ , \end{aligned} \tag{3.29}$$

with

$$f_i = \rho \left[(U - U_e) L_i \xi(X) + U(M_i Y + N_i)\right] - \mu L_i \tag{3.30}$$

$$g_i = \rho V \left[L_i \xi(X) + M_i Y + N_i\right] - \mu M_i \ . \tag{3.31}$$

The integrated flux across the interior face is given by

$$\int_{f_i} \vec{J_\phi} \cdot d\vec{A} = \int_{f_i} \left(J_{\phi-X} dA_X + J_{\phi-Y} dA_Y \right) , \tag{3.32}$$

where A_X and A_Y are the areas in the X and Y direction for face f_i. The area vector is defined to points in the direction from node 1 to 2 for face 1, from node 2 to 3 for face 2, and from node 3 to 1 for face 3.

$$(\Delta A_X)_{f_1} = Y_c - Y_{e_1} \qquad (\Delta A_Y)_{f_1} = X_{e_1} - X_c \tag{3.33}$$

$$(\Delta A_X)_{f_2} = Y_c - Y_{e_2} \qquad (\Delta A_Y)_{f_2} = X_{e_2} - X_c \tag{3.34}$$

$$(\Delta A_X)_{f_3} = Y_c - Y_{e_3} \qquad (\Delta A_Y)_{f_3} = X_{e_3} - X_c \tag{3.35}$$

Simpson's quadrature rule (which is equivalent to the Gauss-Lobatto quadrature rule for three points) is used to integrate over each face f_i.

$$\begin{aligned} \int_{f_i} \vec{J_\phi} \cdot d\vec{A} =& \Big[J_{\phi-X}(e_i) + 4J_{\phi-X}(m_i) + J_{\phi-X}(c) \Big] \frac{(\Delta A_X)_{f_i}}{6} \\ &+ \Big[J_{\phi-Y}(e_i) + 4J_{\phi-Y}(m_i) + J_{\phi-Y}(c) \Big] \frac{(\Delta A_Y)_{f_i}}{6} \qquad (3.36) \\ =& \Big([f_1(e_i) + 4f_1(m_i) + f_1(c)] \frac{(\Delta A_X)_{f_i}}{6} + [g_1(e_i) + 4g_1(m_i) + g_1(c)] \frac{(\Delta A_Y)_{f_i}}{6} \Big) \phi_1 \\ &+ \Big([f_2(e_i) + 4f_2(m_i) + f_2(c)] \frac{(\Delta A_X)_{f_i}}{6} + [g_2(e_i) + 4g_2(m_i) + g_2(c)] \frac{(\Delta A_Y)_{f_i}}{6} \Big) \phi_2 \\ &+ \Big([f_3(e_i) + 4f_3(m_i) + f_3(c)] \frac{(\Delta A_X)_{f_i}}{6} + [g_3(e_i) + 4g_3(m_i) + g_3(c)] \frac{(\Delta A_Y)_{f_i}}{6} \Big) \phi_3 \qquad (3.37) \\ =& F_{i,j} \phi_j \; , \qquad (3.38) \end{aligned}$$

with summation over the index $j = 1, 2, 3$. The integrated flux coefficient is defined by

$$F_{i,j} = [f_j(e_i) + 4f_j(m_i) + f_j(c)] \frac{(\Delta A_X(_{f_i}}{6} + [g_j(e_i) + 4g_j(m_i) + g_j(c)] \frac{(\Delta A_Y)_{f_i}}{6} \; . \tag{3.39}$$

Equation 3.38 gives the total flux through control volume face f_i as a function of coefficients $F_{i,j}$ and the node values ϕ_j. For each triangle there are three interior control volume faces that use the same shape function. The fluxes for all interior control volume faces are found by looping through all triangles, calculating the flux through all three control volume faces, and adding the flux to the momentum coefficients if the face area points out of the control volume or subtracting the flux if the face area points into the control volume. The procedure for adding fluxes into the momentum coefficients is as follows.

```
For $n = 1, nele$
    $(a_P)_1 = (a_P)_1 + F_{1,1} - F_{3,1}$
    $(a_{nb-2})_1 = (a_{nb-2})_1 - F_{1,2} + F_{3,2}$
    $(a_{nb-3})_1 = (a_{nb-3})_1 - F_{1,3} + F_{3,3}$
    $(a_P)_2 = (a_P)_2 + F_{2,2} - F_{1,2}$
    $(a_{nb-1})_2 = (a_{nb-1})_2 - F_{2,1} + F_{1,1}$
    $(a_{nb-3})_2 = (a_{nb-3})_2 - F_{2,3} + F_{1,3}$
    $(a_P)_3 = (a_P)_3 + F_{3,3} - F_{2,3}$
    $(a_{nb-1})_3 = (a_{nb-1})_3 - F_{3,1} + F_{2,1}$
    $(a_{nb-2})_3 = (a_{nb-2})_3 - F_{3,2} + F_{2,2}$
```

Here $(a_P)_i$ is the central coefficient for node i, and $(a_{nb-j})_i$ is the neighbor coefficient for the node i equation with the local node j as the neighbor node. The form of the momentum equations is

$$(a_P)_i\phi_i = \sum_{j=1}^{nnb_i}(a_{nb-j})_i\phi_{nb-j} + b_i \ . \tag{3.40}$$

The overall procedure for finding the total convective-diffusive flux across the interior control volume faces is as follows.

1. Using the element centroid velocity, find the rotation angle coefficients (Eq. 3.11).
2. Calculate the artificial velocities defined in Eqs. 3.4 and 3.8, and then interpolate those velocities to the seven integration points e_i, m_i, and c in Fig. 3.2.
3. Transform to local coordinate system, and find U, V, X, and Y at the integration points.

4. Calculate $\xi(X)$ at the integration points.

5. Calculate f_i and g_i at the integration points using Eqs. 3.107 and 3.107.

6. Calculate the integrated flux coefficients $F_{i,j}$ at all three interior faces, $i = 1, 2, 3$, (for all three nodes $j = 1, 2, 3$) using Eq. 3.39.

7. Add the flux coefficients into the momentum coefficients using the procedure given above (the coefficients are the same for x and y momentum equations and do not need to be calculated twice).

3.1.2.1 Boundary Fluxes and Boundary Conditions for the Momentum Equation

The boundary fluxes are computed at the triangle edges, not on the interior faces (see Fig. 3.3). If the mass flux leaving the boundary edge j is $\dot{m}_j$, then the momentum equation for a boundary node is as follows.

$$\left[(a_P)_i + \sum_{j=1}^{nbndedg_i} \dot{m}_j\right] \phi_i = \sum_{n=1}^{nnb_i} (a_{nb-n})_i \phi_{nb-n} + b_i - J_{bnd} \;, \tag{3.41}$$

where $nbndedg_i$ is the number of boundary edges for the node i control volume, and J_{bnd} is the total diffusive momentum flux leaving the domain through all the boundary edges of node i. If the value of ϕ_i is set using a Dirichlet boundary condition, then Eq. 3.41 is used to calculate the total diffusive flux J_{bnd}. If ϕ_i is unknown, and the diffusive and convective fluxes on the boundary are known, then Eq. 3.41 is used to calculate the value of ϕ_i.

3.1.3 Triangular Element Shape Function for Pressure

The linear interpolation of pressure is accomplished with a shape function of the form

$$p(x, y) = \overline{A}x + \overline{B}y + \overline{C} \;. \tag{3.42}$$

The terms $(\overline{A}, \overline{B}, \overline{C})$ are coefficients for each element given in summation notation for $i = 1, 2, 3$ as

$$\overline{A} = \overline{L}_i p_i \qquad \overline{B} = \overline{M}_i p_i \qquad \overline{C} = \overline{N}_i p_i \;. \tag{3.43}$$

The $(\overline{L}, \overline{M}, \overline{N})$ coefficients are functions only of the element geometry and are as follows.

$$\begin{aligned}
\overline{L}_1 &= (y_2 - y_3)/\overline{\Delta} & \overline{L}_2 &= -(y_1 - y_3)/\overline{\Delta} & \overline{L}_3 &= (y_1 - y_2)/\overline{\Delta} \\
\overline{M}_1 &= -(x_2 - x_3)/\overline{\Delta} & \overline{M}_2 &= (x_1 - x_3)/\overline{\Delta} & \overline{M}_3 &= -(x_1 - x_2)/\overline{\Delta} \\
\overline{N}_1 &= (x_2 y_3 - x_3 y_2)/\overline{\Delta} & \overline{N}_2 &= -(x_1 y_3 - x_3 y_1)/\overline{\Delta} & \overline{N}_3 &= (x_1 y_2 - x_2 y_1)/\overline{\Delta} \ ,
\end{aligned} \tag{3.44}$$

with

$$\overline{\Delta} = x_1(y_2 - y_3) - x_2(y_1 - y_3) + x_3(y_1 - y_2) \ . \tag{3.45}$$

The shape function coefficients for pressure have the same form as the momentum flux shape function, but with $\xi = x$ and $Y = y$. The momentum flux shape functions are dependent on the local flow and need to be calculated each time the flow changes. The pressure shape functions are only functions of the grid geometry and are calculated once, assuming the grid does not change.

The shape function for each element is written as

$$p(x, y) = \left(\overline{L}_i x + \overline{M}_i y + \overline{N}_i\right) p_i \ . \tag{3.46}$$

In this form it becomes clear that the gradient of pressure is constant across each element, with components given by

$$\frac{\partial p}{\partial x} = \overline{L}_i p_i \qquad\qquad \frac{\partial p}{\partial y} = \overline{M}_i p_i \ . \tag{3.47}$$

The above formulation gives the element pressure gradient. To find the pressure gradient at each node, the pressure gradient is integrated over the node's control volume and divided by the control volume's total volume. The integration used is as follows.

$$\int_{CV} \frac{\partial p}{\partial x} d\forall = \sum_{e=1}^{nnb-ele} \frac{\Delta\forall_e}{3} \left(\frac{\partial p}{\partial x}\right)_e \tag{3.48}$$

The pressure gradient component is computed from

$$\begin{aligned}
\left(\frac{\partial p}{\partial x}\right)_n &= \frac{1}{\Delta\forall_n} \sum_{e=1}^{nnb-ele} \frac{\Delta\forall_e}{3} \left(\frac{\partial p}{\partial x}\right)_e \\
&= \frac{1}{\Delta\forall_n} \sum_{e=1}^{nnb-ele} \frac{\Delta\forall_e}{3} \left(\overline{L}_1 p_1 + \overline{L}_2 p_2 + \overline{L}_3 p_3\right)_e \ ,
\end{aligned} \tag{3.49}$$

with

$$\Delta\forall_n = \sum_{e=1}^{nnb-ele} \frac{\Delta\forall_e}{3} \ , \tag{3.50}$$

where $nnb - ele$ is the number of neighbor elements that contain node n, or all the elements that make up the control volume for node n. The gradient in the y direction follows similarly.

$$\left(\frac{\partial p}{\partial y}\right)_n = \frac{1}{\Delta\forall_n} \sum_{e=1}^{nnb-ele} \frac{\Delta\forall_e}{3} \left(\overline{M}_1 p_1 + \overline{M}_2 p_2 + \overline{M}_3 p_3\right)_e \tag{3.51}$$

3.1.4 Integration of Pressure and Other Source Terms

Looking back at the momentum equation (Eq. 3.1), the convective and diffusive terms are known, but the unsteady, pressure, and source terms are yet to be integrated. The pressure and source terms are constant over each triangular element, and the x and y momentum equations are given in summation form as

$$-\frac{\partial p}{\partial x} + S_u = -\overline{L}_i p_i + (S_u)_e \tag{3.52}$$

$$-\frac{\partial p}{\partial y} + S_v = -\overline{M}_i p_i + (S_v)_e \ , \tag{3.53}$$

where $(S_u)_e$ and $(S_v)_e$ are the source terms evaluated for the triangular element e, and i is the index for summation over the three nodes on element e. The integration of the source terms results in

$$\int_e (-\frac{\partial p}{\partial x} + S_u) d\forall = \left[-\overline{L}_i p_i + (S_u)_e\right] \Delta\forall_e \tag{3.54}$$

$$\int_e (-\frac{\partial p}{\partial y} + S_v) d\forall = \left[-\overline{M}_i p_i + (S_v)_e\right] \Delta\forall_e \ . \tag{3.55}$$

Each control volume is made up of one third of neighboring element volumes. Therefore, each control volume will have one third of the element source term added.

$$
\begin{aligned}
(b_u)_1 &= (b_u)_1 + \frac{1}{3}\left[-\overline{L}_i p_i + (S_u)_e\right]\Delta\forall_e \\
(b_u)_2 &= (b_u)_2 + \frac{1}{3}\left[-\overline{L}_i p_i + (S_u)_e\right]\Delta\forall_e \\
(b_u)_3 &= (b_u)_3 + \frac{1}{3}\left[-\overline{L}_i p_i + (S_u)_e\right]\Delta\forall_e
\end{aligned}
$$

$$
\begin{aligned}
(b_v)_1 &= (b_v)_1 + \frac{1}{3}\left[-\overline{M}_i p_i + (S_v)_e\right]\Delta\forall_e \\
(b_v)_2 &= (b_v)_2 + \frac{1}{3}\left[-\overline{M}_i p_i + (S_v)_e\right]\Delta\forall_e \\
(b_v)_3 &= (b_v)_3 + \frac{1}{3}\left[-\overline{M}_i p_i + (S_v)_e\right]\Delta\forall_e
\end{aligned}
$$

3.1.5 Discretized Equations

After discretizing the momentum equation (Eq. 3.1) over triangular elements and integrating over the control volume, the equations have the following form.

$$\rho\Delta\forall\frac{du}{dt} = \sum_{nb=1}^{nnb} a_{u-nb}u_{nb} - a_{u-P}u_P + b_u - \Delta\forall\frac{\partial p}{\partial x} \tag{3.56}$$

$$\rho\Delta\forall\frac{dv}{dt} = \sum_{nb=1}^{nnb} a_{v-nb}v_{nb} - a_{v-P}u_P + b_v - \Delta\forall\frac{\partial p}{\partial y} \tag{3.57}$$

Here the source terms b_u and b_v do not contain pressure gradient terms. The momentum equations are integrated in time with any number of methods discussed in Chapter 2, and rearranged into the following form.

$$a'_{u-P}u_P = \sum_{nb=1}^{nnb} a'_{u-nb}u_{nb} + b'_u - d'_u\Delta\forall\frac{\partial p}{\partial x} \tag{3.58}$$

$$a'_{v-P}v_P = \sum_{nb=1}^{nnb} a'_{v-nb}v_{nb} + b'_v - d'_v\Delta\forall\frac{\partial p}{\partial x}\ , \tag{3.59}$$

where the coefficients contain unsteady factors and potentially relaxation factors. The coefficients for each specific algorithm are determine in Chapter 2. The form of the momentum equations required to derive the pressure equation is as follows.

$$u_P = \hat{u} - d_u \frac{\partial p}{\partial x} \qquad\qquad v_P = \hat{v} - d_v \frac{\partial p}{\partial y} \ , \tag{3.60}$$

where

$$\hat{u} = \frac{\sum_{nb=1}^{nnb} a'_{u-nb} u_{nb} + b'_u}{a'_{u-P}} \qquad\qquad d_u = \frac{d'_u \Delta \forall}{a'_{u-P}} \tag{3.61}$$

$$\hat{v} = \frac{\sum_{nb=1}^{nnb} a'_{v-nb} v_{nb} + b'_v}{a'_{v-P}} \qquad\qquad d_v = \frac{d'_v \Delta \forall}{a'_{v-P}} \ . \tag{3.62}$$

The continuity equation (Eq. 3.2) is integrated over the two-dimensional control volume surrounding the primary grid point (P) to yield the following algebraic equation.

$$\sum_{f=1}^{nfaces} (\rho \vec{V}) \cdot \vec{A} = 0 \tag{3.63}$$

The summation is over all control volume faces surrounding the point P.

3.1.6 Generic Pressure Equation

When developing the pressure equation, the velocity interpolation discussed in Sec. 3.1.1 is again used to avoid spurious pressure oscillations. The form of the momentum equations that are interpolated and substituted into the mass conservation equations are

$$\tilde{u} = \hat{u} - d_u \left[\frac{\partial p}{\partial x}\right]_e \qquad\qquad \tilde{v} = \hat{v} - d_v \left[\frac{\partial p}{\partial y}\right]_e \ , \tag{3.64}$$

rather than Eqs. 2.80 and 2.81 used in the staggered structured grid of Sec. 2.3. Equation 3.64 includes pseudo-velocity and pressure coefficient terms (d_u and d_v) evaluated at nodes, but the pressure gradient is evaluated for each triangular element and is a function of the nodal pressure values.

The time integration and relaxation required varies between the SIMPLER, RK-SIMPLER, and IRK-SIMPLER algorithms (shown in Sec. 2.3); however, once the equation is in the form of Eq. 3.64, the algorithm steps are the same.

Equation 3.64 is used to linearly interpolate the velocity across the triangular element. The integrated mass flux through control volume faces are computed using the velocity at the midpoint of the face (m_i in Fig. 3.2). The discrete continuity equation becomes

$$\sum_{f=1}^{nfaces} \rho\Big[\tilde{u}_m(\Delta A_x)_f + \tilde{v}_m(\Delta A_y)_f\Big] = 0 \;, \tag{3.65}$$

where $\tilde{u}_m$ and $\tilde{v}_m$ are the artificial velocity components interpolated to the midpoint of the face and $(\Delta A_x)_f$ and $(\Delta A_y)_f$ are the area components for the face f. This equation is used to find the mass residual for a given velocity and pressure field. Recalling that interpolation of velocity uses a linear interpolation of $\hat{u}$, $\hat{v}$, d_u, and d_v across the triangle while the pressure gradient terms are constant, the continuity equation becomes

$$\sum_{f=1}^{nfaces} \rho\Big[\left(\hat{u}_m - (d_u)_m\overline{L}_i p_i\right)(\Delta A_x)_f + \left(\hat{v}_m - (d_v)_m\overline{M}_i p_i\right)(\Delta A_y)_f\Big] = 0 \;, \tag{3.66}$$

where the pressure gradient for each element is given in summation notation. Rearranging leads to

$$\begin{aligned}\sum_{f=1}^{nfaces} \Big[&\rho\left(\hat{u}_m(\Delta A_x)_f + \hat{v}_m(\Delta A_y)_f\right) \\ &-\rho\left[(d_u)_m\overline{L}_1(\Delta A_x)_f + (d_v)_m\overline{M}_1(\Delta A_y)_f\right]p_1 \\ &-\rho\left[(d_u)_m\overline{L}_2(\Delta A_x)_f + (d_v)_m\overline{M}_2(\Delta A_y)_f\right]p_2 \\ &-\rho\left[(d_u)_m\overline{L}_3(\Delta A_x)_f + (d_v)_m\overline{M}_3(\Delta A_y)_f\right]p_3\Big] = 0 \;,\end{aligned} \tag{3.67}$$

or

$$\sum_{f=1}^{nfaces} \left[\hat{G}_f + G_{i,f}\; p_i\right] = 0 \;. \tag{3.68}$$

with summation over $i = 1, 2, 3$. The coefficients are defined by

$$\begin{aligned}\hat{G}_f &= \rho\left[\hat{u}_m(\Delta A_x)_f + \hat{v}_m(\Delta A_y)_f\right] \\ G_{i,f} &= -\rho\left[(d_u)_m\overline{L}_i(\Delta A_x)_f + (d_v)_m\overline{M}_i(\Delta A_y)_f\right] \;.\end{aligned} \tag{3.69}$$

Again using the convention that area vectors point from node 1 to 2, 2 to 3, and 3 to 1, the pressure coefficients and sources are set by looping through all elements as follows.

$$
\begin{aligned}
&\text{For } n = 1,\ nele\\
(a_{p-P})_1 &= (a_P)_1 + G_{1,1} - G_{3,1}\\
(a_{p-nb-2})_1 &= (a_{nb-2})_1 - G_{1,2} + G_{3,2}\\
(a_{p-nb-3})_1 &= (a_{nb-3})_1 - G_{1,3} + G_{3,3}\\
(b_p)_1 &= (b_p)_1 - \hat{G}_1 + \hat{G}_3\\
(a_{p-P})_2 &= (a_P)_2 + G_{2,2} - G_{1,2}\\
(a_{p-nb-1})_2 &= (a_{nb-1})_2 - G_{2,1} + G_{1,1}\\
(a_{p-nb-3})_2 &= (a_{nb-3})_2 - G_{2,3} + G_{1,3}\\
(b_p)_2 &= (b_p)_2 - \hat{G}_2 + \hat{G}_1\\
(a_{p-P})_3 &= (a_P)_3 + G_{3,3} - G_{2,3}\\
(a_{p-nb-1})_3 &= (a_{nb-1})_3 - G_{3,1} + G_{2,1}\\
(a_{p-nb-2})_3 &= (a_{nb-2})_3 - G_{3,2} + G_{2,2}\\
(b_p)_3 &= (b_p)_3 - \hat{G}_3 + \hat{G}_2\ ,
\end{aligned}
$$

where the subscripts on the pressure coefficients and sources (a_p and b_p) are for the local node number, the subscripts for $G_{i,f}$ represent the local node i and the local face f, and the subscript on $\hat{G}$ represents the local face number.

The pressure equation is derived from mass conservation, and boundary conditions are included in the pressure source term by adding any mass flux leaving the domain through the boundary edges.

$$(b_p)_i = (b_p)_i + \sum_{j=1}^{nbndedg_i} \dot{m}_j \tag{3.70}$$

With the discretization, integration, and coefficients defined for the momentum and pressure equations, the algorithms follow the same procedure given in Sec. 2.3.

3.2 Three-Dimensional Tetrahedral Grid Formulation

In three dimensions, a tetrahedral, vertex-centered, median-dual control volume formulation is used following Guntupalli [59]. Each tetrahedral element contains four nodes, four faces, six edges, and six interior faces. Figures 3.5 and 3.6 show a tetrahedral element with nodes labeled n_i, faces centers labeled f_i, and edge centers labeled e_i. Each interior control volume face is comprised of two adjacent triangular faces, with each triangular face defined by one edge centroid, one face centroid, and the element centroid (c). Figure 3.7 shows three of these triangular faces. The two triangular faces that share an edge centroid define the control volume face between the two nodes that make up the shared edge. These two triangular faces are equal in both magnitude and direction. The points that define the six interior control volume faces for a tetrahedral element are given in Table 3.1. The area of the faces are found by taking the cross product of the vector from the lower left point to the upper left point crossed with the vector from the lower left point to the lower right point. For example, $\vec{A_1} = (\vec{r}_c - \vec{r}_{f_1}) \times (\vec{r}_{e_1} - \vec{r}_{f_1})$ where $\vec{r}_p$ is the vector from the origin to the point p.

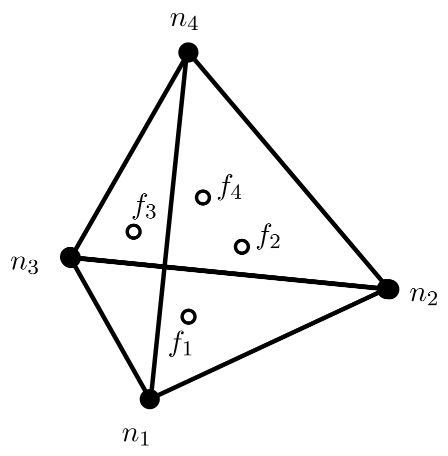


Figure 3.5: Tetrahedral element faces.

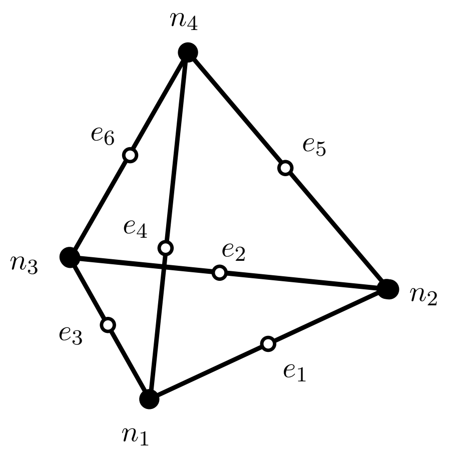


Figure 3.6: Tetrahedral element edges.

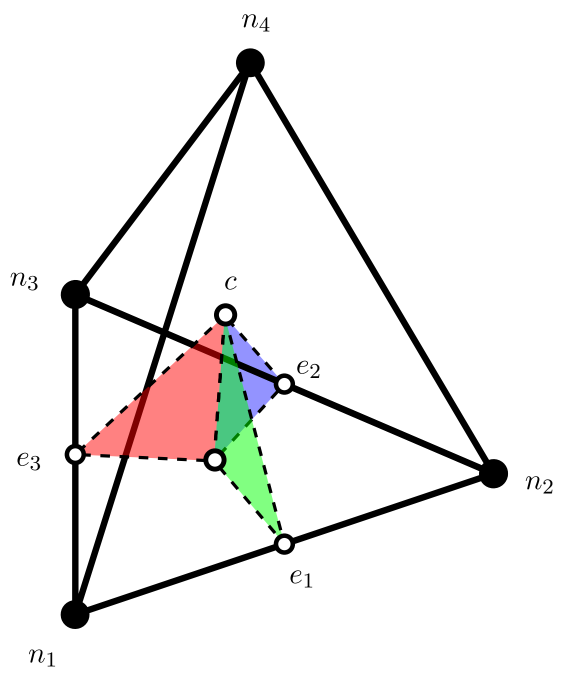


Figure 3.7: Three triangles that are part of tetrahedral interior control volume faces.

Table 3.1: Tetrahedral interior control volume faces[1].

Interior Face #	Area Vector Pointing Out Of	Area Vector Pointing Into	Lower Left Point	Lower Right Point	Upper Right Point	Upper Left Point
1	n_1	n_2	f_1	e_1	f_2	c
2	n_2	n_3	f_1	e_2	f_4	c
3	n_3	n_1	f_1	e_3	f_3	c
4	n_1	n_4	f_2	e_4	f_3	c
5	n_2	n_4	f_4	e_5	f_2	c
6	n_3	n_4	f_3	e_6	f_4	c

As in the two-dimensional case of triangular elements, the density and viscosity are constant within each triangular element and pressure varies linearly.

[1]Area vectors point into the page for the given orientation.

3.2.1 Interpolation of Velocity

The velocity interpolation described in Sec. 3.1.1 is used with

$$\tilde{u}_{e-n} = \hat{u}_n - (d_u)_n \left(\frac{\partial p}{\partial x}\right)_e \tag{3.71}$$

$$\tilde{v}_{e-n} = \hat{v}_n - (d_v)_n \left(\frac{\partial p}{\partial y}\right)_e \tag{3.72}$$

$$\tilde{w}_{e-n} = \hat{w}_n - (d_w)_n \left(\frac{\partial p}{\partial z}\right)_e , \tag{3.73}$$

where the values of $\hat{u}$, $\hat{v}$, $\hat{w}$, d_u, d_v, and d_w are determined by the fully discrete momentum equations at each node. The artificial velocities are interpolated linearly (only $\tilde{u}$ is shown but all linear interpolations follow the same pattern).

$$\tilde{u}_{e_1} = \frac{1}{2}\left(\tilde{u}_{n_1} + \tilde{u}_{n_2}\right) \qquad \tilde{u}_{e_2} = \frac{1}{2}\left(\tilde{u}_{n_2} + \tilde{u}_{n_3}\right) \qquad \tilde{u}_{e_3} = \frac{1}{2}\left(\tilde{u}_{n_3} + \tilde{u}_{n_1}\right) \tag{3.74}$$

$$\tilde{u}_{e_4} = \frac{1}{2}\left(\tilde{u}_{n_1} + \tilde{u}_{n_4}\right) \qquad \tilde{u}_{e_5} = \frac{1}{2}\left(\tilde{u}_{n_2} + \tilde{u}_{n_4}\right) \qquad \tilde{u}_{e_6} = \frac{1}{2}\left(\tilde{u}_{n_3} + \tilde{u}_{n_4}\right) \tag{3.75}$$

$$\tilde{u}_c = \frac{1}{4}\left(\tilde{u}_{n_1} + \tilde{u}_{n_2} + \tilde{u}_{n_3} + \tilde{u}_{n_4}\right) \tag{3.76}$$

$$\tilde{u}_{f_1} = \frac{1}{3}\left(\tilde{u}_{n_1} + \tilde{u}_{n_2} + \tilde{u}_{n_3}\right) \qquad \tilde{u}_{f_2} = \frac{1}{3}\left(\tilde{u}_{n_1} + \tilde{u}_{n_2} + \tilde{u}_{n_4}\right) \tag{3.77}$$

$$\tilde{u}_{f_3} = \frac{1}{3}\left(\tilde{u}_{n_1} + \tilde{u}_{n_3} + \tilde{u}_{n_4}\right) \qquad \tilde{u}_{f_4} = \frac{1}{3}\left(\tilde{u}_{n_2} + \tilde{u}_{n_3} + \tilde{u}_{n_4}\right) \tag{3.78}$$

3.2.2 Tetrahedral Element Momentum Flux Computation

The momentum fluxes for tetrahedral elements are calculated similarly to the triangular elements, with a Power Law scheme from Baliga [30]. The scheme starts with the three-dimensional, steady convection-diffusion equation for a general variables ϕ with no source, given by

$$\frac{\partial \rho u_e \phi}{\partial x} + \frac{\partial \rho v_e \phi}{\partial y} + \frac{\partial \rho w_e \phi}{\partial z} - \frac{\partial}{\partial x}\left[\mu \frac{\partial \phi}{\partial x}\right] - \frac{\partial}{\partial y}\left[\mu \frac{\partial \phi}{\partial y}\right] - \frac{\partial}{\partial z}\left[\mu \frac{\partial \phi}{\partial z}\right] = 0 , \tag{3.79}$$

where u_e, v_e, and w_e are the element centroid velocity components for x, y, and z directions respectively.

$$u_e = \frac{1}{4}(u_{n_1} + u_{n_2} + u_{n_3} + u_{n_4}) \tag{3.80}$$

$$v_e = \frac{1}{4}(v_{n_1} + v_{n_2} + v_{n_3} + v_{n_4}) \tag{3.81}$$

$$w_e = \frac{1}{4}(w_{n_1} + w_{n_2} + w_{n_3} + w_{n_4}) \tag{3.82}$$

A rotational transformation then aligns a new coordinate direction, X, to the element velocity direction, $U_e = \sqrt{u_e^2 + v_e^2 + w_e^2}$ (see Fig. 3.8).

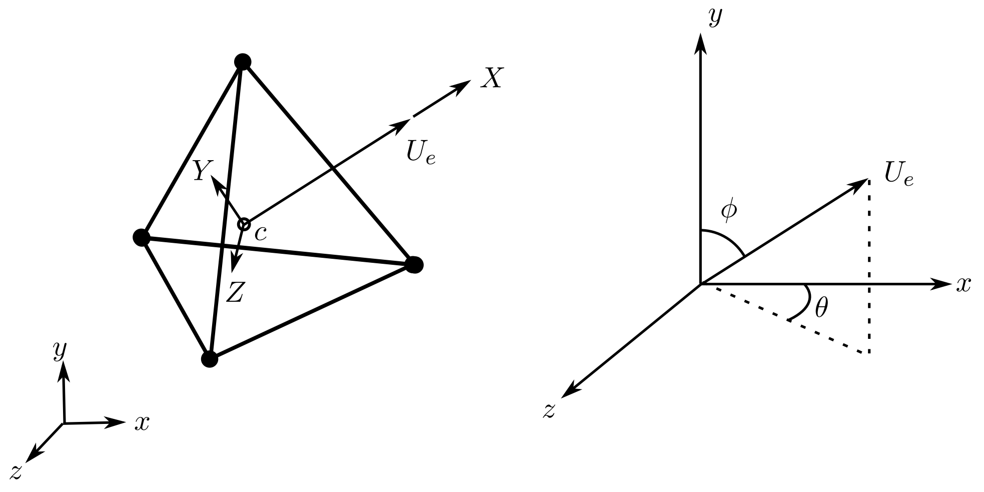


Figure 3.8: Tetrahedral element rotation to local coordinates.

The local coordinate system is found by translating the origin to the element centroid and rotating the new X direction to point in the direction of the element average velocity vector. The rotation matrix coefficients for angles ϕ and θ are given by

$$\cos\phi = \frac{v_e}{U_e} \qquad \sin\phi = \frac{\sqrt{u_e^2 + w_e^2}}{U_e} \tag{3.83}$$

$$\cos\theta = \frac{u_e}{U_e \sin\phi} \qquad \sin\theta = \frac{w_e}{U_e \sin\phi}\ . \tag{3.84}$$

In the event $U_e = 0$, the rotation angles are set such that $\sin\phi = \cos\theta = 1$ and $\cos\phi = \sin\theta = 0$. The transformation from (x, y, z) to the element local coordinate system is

$$X = \Big[(x - x_e)\cos\theta + (z - z_e)\sin\theta\Big]\sin\phi + (y - y_e)\cos\phi \tag{3.85}$$

$$Y = -\Big[(x - x_e)\cos\theta + (z - z_e)\sin\theta\Big]\cos\phi + (y - y_e)\sin\phi \tag{3.86}$$

$$Z = -(x - x_e)\sin\theta + (z - z_e)\cos\theta\ , \tag{3.87}$$

and the transformation for velocity is

$$U = \Big[u\cos\theta + w\sin\theta\Big]\sin\phi + v\cos\phi \tag{3.88}$$

$$V = -\Big[u\cos\theta + w\sin\theta\Big]\cos\phi + v\sin\phi \tag{3.89}$$

$$W = -u\sin\theta + w\cos\theta\ . \tag{3.90}$$

The convection diffusion equation is written for the new coordinate system.

$$\frac{\partial\rho U_e\phi}{\partial X} - \frac{\partial}{\partial X}\left[\mu\frac{\partial\phi}{\partial X}\right] - \frac{\partial}{\partial Y}\left[\mu\frac{\partial\phi}{\partial Y}\right] - \frac{\partial}{\partial Z}\left[\mu\frac{\partial\phi}{\partial Z}\right] = 0 \tag{3.91}$$

To interpolate across the tetrahedron, a shape function is assumed that varies linearly in the Y and Z directions and by some function $\xi(X)$ in the X direction.

$$\phi = A\xi(X) + BY + CZ + D \tag{3.92}$$

where the values of A, B, C, and D are constant for each element and are given in summation form as

$$A = L_i\phi_i \qquad B = M_i\phi_i \qquad C = N_i\phi_i \qquad D = O_i\phi_i\ , \tag{3.93}$$

with $i = 1, 2, 3, 4$ for each tetrahedron vertex. The terms L, M, N, and O are given by

$$
\begin{aligned}
L_1 &= \big[Y_2(Z_3-Z_4) - Y_3(Z_2-Z_4) + Y_4(Z_2-Z_3)\big]/\Delta \\
L_2 &= -\big[Y_1(Z_3-Z_4) - Y_3(Z_1-Z_4) + Y_4(Z_1-Z_3)\big]/\Delta \\
L_3 &= \big[Y_1(Z_2-Z_4) - Y_2(Z_1-Z_4) + Y_4(Z_1-Z_2)\big]/\Delta \\
L_4 &= -\big[Y_1(Z_2-Z_3) - Y_2(Z_1-Z_3) + Y_3(Z_1-Z_2)\big]/\Delta
\end{aligned}
\tag{3.94}
$$

$$
\begin{aligned}
M_1 &= -\big[\xi_2(Z_3-Z_4) - \xi_3(Z_2-Z_4) + \xi_4(Z_2-Z_3)\big]/\Delta \\
M_2 &= \big[\xi_1(Z_3-Z_4) - \xi_3(Z_1-Z_4) + \xi_4(Z_1-Z_3)\big]/\Delta \\
M_3 &= -\big[\xi_1(Z_2-Z_4) - \xi_2(Z_1-Z_4) + \xi_4(Z_1-Z_2)\big]/\Delta \\
M_4 &= \big[\xi_1(Z_2-Z_3) - \xi_2(Z_1-Z_3) + \xi_3(Z_1-Z_2)\big]/\Delta
\end{aligned}
\tag{3.95}
$$

$$
\begin{aligned}
N_1 &= \big[\xi_2(Y_3-Y_4) - \xi_3(Y_2-Y_4) + \xi_4(Y_2-Y_3)\big]/\Delta \\
N_2 &= -\big[\xi_1(Y_3-Y_4) - \xi_3(Y_1-Y_4) + \xi_4(Y_1-Y_3)\big]/\Delta \\
N_3 &= \big[\xi_1(Y_2-Y_4) - \xi_2(Y_1-Y_4) + \xi_4(Y_1-Y_2)\big]/\Delta \\
N_4 &= -\big[\xi_1(Y_2-Y_3) - \xi_2(Y_1-Y_3) + \xi_3(Y_1-Y_2)\big]/\Delta
\end{aligned}
\tag{3.96}
$$

$$
\begin{aligned}
O_1 &= -\big[\xi_2(Y_3Z_4-Y_4Z_3) - \xi_3(Y_2Z_4-Y_4Z_2) + \xi_4(Y_2Z_3-Y_3Z_2)\big]/\Delta \\
O_2 &= \big[\xi_1(Y_3Z_4-Y_4Z_3) - \xi_3(Y_1Z_4-Y_4Z_1) + \xi_4(Y_1Z_3-Y_3Z_1)\big]/\Delta \\
O_3 &= -\big[\xi_1(Y_2Z_4-Y_4Z_2) - \xi_2(Y_1Z_4-Y_4Z_1) + \xi_4(Y_1Z_2-Y_2Z_1)\big]/\Delta \\
O_4 &= \big[\xi_1(Y_2Z_3-Y_3Z_2) - \xi_2(Y_1Z_3-Y_3Z_1) + \xi_3(Y_1Z_2-Y_2Z_1)\big]/\Delta \ ,
\end{aligned}
\tag{3.97}
$$

with

$$
\begin{aligned}
\Delta =&(\xi_1-\xi_2)(Y_3Z_4-Y_4Z_3) - (\xi_1-\xi_3)(Y_2Z_4-Y_4Z_2) + (\xi_1-\xi_4)(Y_2Z_3-Y_3Z_2) \\
&+(\xi_2-\xi_3)(Y_1Z_4-Y_4Z_1) - (\xi_2-\xi_4)(Y_1Z_3-Y_3Z_1) + (\xi_3-\xi_4)(Y_1Z_2-Y_2Z_1)\ .
\end{aligned}
\tag{3.98}
$$

The function $\xi(X)$ has the same solution as in the triangular elements (Eq. 3.21 for exponential and Eq. 3.24 for Power Law), with the minimum and maximum X now defined by

$$X_{min} = \min(X_{n_1}.X_{n_2}, X_{n_3}, X_{n_4}) \qquad X_{max} = \max(X_{n_1}.X_{n_2}, X_{n_3}, X_{n_4}) \ . \tag{3.99}$$

The interpolation of ϕ is rewritten as

$$\phi = [L_i\xi(X) + M_iY + N_iZ + O_i]\,\phi_i \ , \tag{3.100}$$

with derivatives

$$\frac{\partial\phi}{\partial X} = \left[\frac{\rho U_e}{\mu}\xi + 1\right] L_i\phi_i \tag{3.101}$$

$$\frac{\partial\phi}{\partial Y} = M_i\phi_i \tag{3.102}$$

$$\frac{\partial\phi}{\partial Z} = N_i\phi_i \ . \tag{3.103}$$

The total momentum equation convective and diffusive flux is written as

$$J_{\phi-X} = \rho U\phi - \mu_e\frac{\partial\phi}{\partial X} \qquad J_{\phi-Y} = \rho V\phi - \mu_e\frac{\partial\phi}{\partial Y} \qquad J_{\phi-Z} = \rho W\phi - \mu_e\frac{\partial\phi}{\partial Z} \tag{3.104}$$

$$= F_i\phi_i \qquad = G_i\phi_i \qquad = H_i\phi_i \ , \tag{3.105}$$

with

$$F_i = \rho\left[(U - U_e)L_i\xi(X) + U(M_iY + N_iZ + O_i)\right] - \mu L_i \tag{3.106}$$

$$G_i = \rho V\left[L_i\xi(X) + M_iY + N_iZ + O_i\right] - \mu M_i \tag{3.107}$$

$$H_i = \rho W\left[L_i\xi(X) + M_iY + N_iZ + O_i\right] - \mu N_i \ . \tag{3.108}$$

The integrated flux across the interior face is given by

$$\int_{CVf} \vec{J_\phi}\cdot d\vec{A} = \int_{CVf} (J_{\phi-X}dA_X + J_{\phi-Y}dA_Y + J_{\phi-Z}dA_Z) \ , \tag{3.109}$$

where A_X, A_Y, and A_Z are the areas in the X, Y, and Z directions for the interior control volume face CVf. Each triangular face is integrated using a quadratic quadrature based on equal weighting of the three triangle edge midpoints. For an interior control volume face containing the tetrahedral face centroid f and the tetrahedral edge centroid e, the three midpoints used for integration are between the face center and the edge center $(\overline{e-f})$, between the edge center and the element centroid $(\overline{c-e})$, and between the face center and the element centroid $(\overline{c-f})$. The flux at the midpoints are found by using the X, Y, Z locations of the midpoints, which for X are

$$X(\overline{e-f}) = \frac{1}{2}\Big[X(e) + X(f)\Big] \tag{3.110}$$

$$X(\overline{c-e}) = \frac{1}{2}\Big[X(c) + X(e)\Big] \tag{3.111}$$

$$X(\overline{c-f}) = \frac{1}{2}\Big[X(c) + X(f)\Big] \; . \tag{3.112}$$

The integration of flux for one triangular area of the control volume face $CVf(e, f_a)$ containing the face f_c and edge e is

$$\begin{aligned}\int_{CVf(e,f_a)} \vec{J}_\phi \cdot d\vec{A} &= \int_{CVf(e,f_a)} (J_{\phi-X} dA_X + J_{\phi-Y} dA_Y + J_{\phi-Z} dA_Z) \\ &= \bigg(\Big[F_i(\overline{c-e}) + F_i(\overline{c-f_a}) + F_i(\overline{e-f_a})\Big]\frac{\Delta A_X}{3} \\ &\quad + \Big[G_i(\overline{c-e}) + G_i(\overline{c-f_a}) + G_i(\overline{e-f_a})\Big]\frac{\Delta A_Y}{3} \\ &\quad + \Big[H_i(\overline{c-e}) + H_i(\overline{c-f_a}) + H_i(\overline{e-f_a})\Big]\frac{\Delta A_Z}{3}\bigg)\phi_i \; ,\end{aligned} \tag{3.113}$$

where $\Delta\vec{A}$ is the the area vector for only one of the triangular faces, but is equal to the other triangle making up each interior face.

The integration of the flux across both triangles making up each control volume face is found by adding the flux for first triangle face, $CVf(e, f_a)$, to the second triangle, $CVf(e, f_b)$, which gives the total flux for the control volume face CVf.

$$\int_{CVf(e,f_a)} \vec{J}_\phi \cdot d\vec{A} + \int_{CVf(e,f_b)} \vec{J}_\phi \cdot d\vec{A} + = \Big(\Big[F_i(\overline{c-e}) + F_i(\overline{c-f_a}) + F_i(\overline{e-f_a})\Big]\frac{\Delta A_X}{3}$$
$$+\Big[G_i(\overline{c-e}) + G_i(\overline{c-f_a}) + G_i(\overline{e-f_a})\Big]\frac{\Delta A_Y}{3}$$
$$+\Big[H_i(\overline{c-e}) + H_i(\overline{c-f_a}) + H_i(\overline{e-f_a})\Big]\frac{\Delta A_Z}{3}$$
$$+\Big[F_i(\overline{c-e}) + F_i(\overline{c-f_b}) + F_i(\overline{e-f_b})\Big]\frac{\Delta A_X}{3}$$
$$+\Big[G_i(\overline{c-e}) + G_i(\overline{c-f_b}) + G_i(\overline{e-f_b})\Big]\frac{\Delta A_Y}{3}$$
$$+\Big[H_i(\overline{c-e}) + H_i(\overline{c-f_b}) + H_i(\overline{e-f_b})\Big]\frac{\Delta A_Z}{3})\phi_i \ , \tag{3.114}$$

or

$$\int_{CVf} \vec{J}_\phi \cdot d\vec{A} = \Big(\Big[2F_i(\overline{c-e}) + F_i(\overline{c-f_a}) + F_i(\overline{e-f_a}) + F_i(\overline{c-f_b}) + F_i(\overline{e-f_b})\Big]\frac{\Delta A_X}{3}$$
$$+\Big[2G_i(\overline{c-e}) + G_i(\overline{c-f_a}) + G_i(\overline{e-f_a}) + G_i(\overline{c-f_b}) + G_i(\overline{e-f_b})\Big]\frac{\Delta A_Y}{3}$$
$$+\Big[2H_i(\overline{c-e}) + H_i(\overline{c-f_a}) + H_i(\overline{e-f_a}) + H_i(\overline{c-f_b}) + H_i(\overline{e-f_b})\Big]\frac{\Delta A_Z}{3}\Big)\phi_i \tag{3.115}$$

$$= E_{CFf,i}\phi_i \ . \tag{3.116}$$

For the interior control volume face shown in Fig. 3.7, going from node 1 to 2, the edge center is $e = e_1$ and the two face centers are $f_a = f_1$ and $f_b = f_2$. Table 3.1 shows all six interior faces and the edge and face centers that are used to compute the integrated flux.

Similar to the triangular grid, the fluxes for each interior face is calculated by looping through all tetrahedral elements, finding the flux through all six interior faces, and either adding the flux to the momentum coefficients if the area vector points out of the control volume or subtracting the flux if the face area points into the control volume. Using the convention in Table 3.1 (with CFf being the index in the first column), the procedure for adding fluxes to the momentum coefficients is as follows.

$$\begin{aligned}
&\text{For } n = 1,\, nele \\
(a_P)_1 &= (a_P)_1 + E_{1,1} - E_{3,1} + E_{4,1} \\
(a_{nb-2})_1 &= (a_{nb-2})_1 - E_{1,2} + E_{3,2} - E_{4,2} \\
(a_{nb-3})_1 &= (a_{nb-3})_1 - E_{1,3} + E_{3,3} - E_{4,3} \\
(a_{nb-4})_1 &= (a_{nb-4})_1 - E_{1,4} + E_{3,4} - E_{4,4} \\
(a_P)_2 &= (a_P)_2 - E_{1,2} + E_{2,2} + E_{5,2} \\
(a_{nb-1})_2 &= (a_{nb-1})_2 + E_{1,1} - E_{2,1} - E_{5,1} \\
(a_{nb-3})_2 &= (a_{nb-3})_2 + E_{1,3} - E_{2,3} - E_{5,3} \\
(a_{nb-4})_2 &= (a_{nb-4})_2 + E_{1,4} - E_{2,4} - E_{5,4} \\
(a_P)_3 &= (a_P)_3 - E_{2,3} + E_{3,3} + E_{6,3} \\
(a_{nb-1})_3 &= (a_{nb-1})_3 + E_{2,1} - E_{3,1} - E_{6,1} \\
(a_{nb-2})_3 &= (a_{nb-2})_3 + E_{2,2} - E_{3,2} - E_{6,2} \\
(a_{nb-4})_3 &= (a_{nb-4})_3 + E_{2,4} - E_{3,4} - E_{6,4} \\
(a_P)_4 &= (a_P)_4 - E_{4,4} - E_{5,4} - E_{6,4} \\
(a_{nb-1})_4 &= (a_{nb-1})_4 + E_{4,1} + E_{5,1} + E_{6,1} \\
(a_{nb-2})_4 &= (a_{nb-2})_4 + E_{4,2} + E_{5,2} + E_{6,2} \\
(a_{nb-3})_4 &= (a_{nb-3})_4 + E_{4,3} + E_{5,3} + E_{6,3} \ ,
\end{aligned}$$

where $(a_P)_i$ is the central coefficient for node i, and $(a_{nb-j})_i$ is the neighbor coefficient for the node i equation with the local node j as the neighbor node. The form of the momentum equations is as follows.

$$(a_P)_i\phi_i = \sum_{j=1}^{nnb_i}(a_{nb-j})_i\phi_{nb-j} + b_i \tag{3.117}$$

The procedure for finding fluxes follows the triangular procedure found in Sec. 3.1.2, and the boundary conditions for momentum equations follows the triangular grid discussed in Sec. 3.1.2.1.

3.2.3 Tetrahedral Element Shape Function for Pressure

The linear interpolation of pressure is accomplished with a shape function of the form

$$p(x, y, z) = \overline{A}x + \overline{B}y + \overline{C}z + \overline{D} \ . \tag{3.118}$$

The terms $(\overline{A}, \overline{B}, \overline{C})$ are coefficients for each element given in summation notation for $i = 1, 2, 3, 4$ as

$$\overline{A} = \overline{L}_i p_i \qquad \overline{B} = \overline{M}_i p_i \qquad \overline{C} = \overline{N}_i p_i \qquad \overline{D} = \overline{O}_i p_i \ . \tag{3.119}$$

The $(\overline{L}, \overline{M}, \overline{N})$ coefficients are functions only of the element geometry and are as follows.

$$\begin{aligned}
\overline{L}_1 &= \big[Y_2(Z_3 - Z_4) - Y_3(Z_2 - Z_4) + Y_4(Z_2 - Z_3)\big]/\overline{\Delta} \\
\overline{L}_2 &= -\big[Y_1(Z_3 - Z_4) - Y_3(Z_1 - Z_4) + Y_4(Z_1 - Z_3)\big]/\overline{\Delta} \\
\overline{L}_3 &= \big[Y_1(Z_2 - Z_4) - Y_2(Z_1 - Z_4) + Y_4(Z_1 - Z_2)\big]/\overline{\Delta} \\
\overline{L}_4 &= -\big[Y_1(Z_2 - Z_3) - Y_2(Z_1 - Z_3) + Y_3(Z_1 - Z_2)\big]/\overline{\Delta}
\end{aligned} \tag{3.120}$$

$$\begin{aligned}
\overline{M}_1 &= -\big[X_2(Z_3 - Z_4) - X_3(Z_2 - Z_4) + X_4(Z_2 - Z_3)\big]/\overline{\Delta} \\
\overline{M}_2 &= \big[X_1(Z_3 - Z_4) - X_3(Z_1 - Z_4) + X_4(Z_1 - Z_3)\big]/\overline{\Delta} \\
\overline{M}_3 &= -\big[X_1(Z_2 - Z_4) - X_2(Z_1 - Z_4) + X_4(Z_1 - Z_2)\big]/\overline{\Delta} \\
\overline{M}_4 &= \big[X_1(Z_2 - Z_3) - X_2(Z_1 - Z_3) + X_3(Z_1 - Z_2)\big]/\overline{\Delta}
\end{aligned} \tag{3.121}$$

$$\begin{aligned}
\overline{N}_1 &= \big[X_2(Y_3 - Y_4) - X_3(Y_2 - Y_4) + X_4(Y_2 - Y_3)\big]/\overline{\Delta} \\
\overline{N}_2 &= -\big[X_1(Y_3 - Y_4) - X_3(Y_1 - Y_4) + X_4(Y_1 - Y_3)\big]/\overline{\Delta} \\
\overline{N}_3 &= \big[X_1(Y_2 - Y_4) - X_2(Y_1 - Y_4) + X_4(Y_1 - Y_2)\big]/\overline{\Delta} \\
\overline{N}_4 &= -\big[X_1(Y_2 - Y_3) - X_2(Y_1 - Y_3) + X_3(Y_1 - Y_2)\big]/\overline{\Delta}
\end{aligned} \tag{3.122}$$

$$\begin{aligned}
\overline{O}_1 &= -\big[X_2(Y_3Z_4 - Y_4Z_3) - X_3(Y_2Z_4 - Y_4Z_2) + X_4(Y_2Z_3 - Y_3Z_2)\big]/\overline{\Delta} \\
\overline{O}_2 &= \big[X_1(Y_3Z_4 - Y_4Z_3) - X_3(Y_1Z_4 - Y_4Z_1) + X_4(Y_1Z_3 - Y_3Z_1)\big]/\overline{\Delta} \\
\overline{O}_3 &= -\big[X_1(Y_2Z_4 - Y_4Z_2) - X_2(Y_1Z_4 - Y_4Z_1) + X_4(Y_1Z_2 - Y_2Z_1)\big]/\overline{\Delta} \\
\overline{O}_4 &= \big[X_1(Y_2Z_3 - Y_3Z_2) - X_2(Y_1Z_3 - Y_3Z_1) + X_3(Y_1Z_2 - Y_2Z_1)\big]/\overline{\Delta} \ ,
\end{aligned} \tag{3.123}$$

with

$$\overline{\Delta} = (X_1 - X_2)(Y_3 Z_4 - Y_4 Z_3) - (X_1 - X_3)(Y_2 Z_4 - Y_4 Z_2) + (X_1 - X_4)(Y_2 Z_3 - Y_3 Z_2) \\ + (X_2 - X_3)(Y_1 Z_4 - Y_4 Z_1) - (X_2 - X_4)(Y_1 Z_3 - Y_3 Z_1) + (X_3 - X_4)(Y_1 Z_2 - Y_2 Z_1) \ . \qquad (3.124)$$

The shape function coefficients for pressure have the same form as the momentum flux shape function with $\xi = x$, $Y = y$, and $Z = z$. The momentum flux shape functions are dependent on the local flow and need to be calculated each time the flow changes. The pressure shape functions are only functions of the grid geometry and are calculated once, assuming the grid does not change.

The shape function for each element is rewritten as

$$p(x, y, z) = \left(\overline{L}_i x + \overline{M}_i y + \overline{N}_i z + \overline{O}_i\right) p_i \ . \qquad (3.125)$$

The pressure gradient components are given by

$$\frac{\partial p}{\partial x} = \overline{L}_i p_i \qquad \frac{\partial p}{\partial y} = \overline{M}_i p_i \qquad \frac{\partial p}{\partial z} = \overline{N}_i p_i \ . \qquad (3.126)$$

The above formulation gives the element pressure gradient. To find the pressure gradient at each node, the pressure gradient is integrated over the node's control volume, and then divided by the control volume's total volume. The integration used is as follows.

$$\int_{CV} \frac{\partial p}{\partial x} d\forall = \sum_{e=1}^{nnb-ele} \frac{\Delta\forall_e}{4} \left(\frac{\partial p}{\partial x}\right)_e \qquad (3.127)$$

The pressure gradient component is computed from

$$\left(\frac{\partial p}{\partial x}\right)_n = \frac{1}{\Delta\forall_n} \sum_{e=1}^{nnb-ele} \frac{\Delta\forall_e}{4} \left(\frac{\partial p}{\partial x}\right)_e \qquad (3.128)$$

$$= \frac{1}{\Delta\forall_n} \sum_{e=1}^{nnb-ele} \frac{\Delta\forall_e}{4} \left(\overline{L}_1 p_1 + \overline{L}_2 p_2 + \overline{L}_3 p_3 + \overline{L}_4 p_4\right)_e \ , \qquad (3.129)$$

with

$$\Delta\forall_n = \sum_{e=1}^{nnb-ele} \frac{\Delta\forall_e}{4} \ , \qquad (3.130)$$

where $nnb-ele$ is the number of neighbor elements that contain node n, or all the elements that make up the control volume for node n. The gradient in the y and z directions follows similarly.

$$\left(\frac{\partial p}{\partial y}\right)_n = \frac{1}{\Delta\forall_n}\sum_{e=1}^{nnb-ele}\frac{\Delta\forall_e}{4}\left(\overline{M}_1 p_1 + \overline{M}_2 p_2 + \overline{M}_3 p_3 + \overline{M}_4 p_4\right)_e \tag{3.131}$$

$$\left(\frac{\partial p}{\partial z}\right)_n = \frac{1}{\Delta\forall_n}\sum_{e=1}^{nnb-ele}\frac{\Delta\forall_e}{4}\left(\overline{N}_1 p_1 + \overline{N}_2 p_2 + \overline{N}_3 p_3 + \overline{N}_4 p_4\right)_e \tag{3.132}$$

3.2.4 Integration of Pressure and Other Source Terms

The pressure and source terms are constant over each triangular element, and for the x, y, and z momentum equations are

$$-\frac{\partial p}{\partial x} + S_u = -\overline{L}_i p_i + (S_u)_e \tag{3.133}$$

$$-\frac{\partial p}{\partial y} + S_v = -\overline{M}_i p_i + (S_v)_e \tag{3.134}$$

$$-\frac{\partial p}{\partial z} + S_w = -\overline{N}_i p_i + (S_w)_e \ , \tag{3.135}$$

where $(S_u)_e$, $(S_v)_e$, and $(S_w)_e$ are the source terms evaluated for the triangular element e. The integration of the source terms result in

$$\int_e (-\frac{\partial p}{\partial x} + S_u)d\forall = \left[-\overline{L}_i p_i + (S_u)_e\right]\Delta\forall_e \tag{3.136}$$

$$\int_e (-\frac{\partial p}{\partial y} + S_v)d\forall = \left[-\overline{M}_i p_i + (S_v)_e\right]\Delta\forall_e \tag{3.137}$$

$$\int_e (-\frac{\partial p}{\partial z} + S_w)d\forall = \left[-\overline{N}_i p_i + (S_w)_e\right]\Delta\forall_e \ . \tag{3.138}$$

Each control volume is made up of one forth of neighboring element volumes. Therefore, each control volume will have one forth of the element source term added.

$$(b_u)_1 = (b_u)_1 + \frac{1}{4}\left[-\overline{L}_i p_i + (S_u)_e\right]\Delta\forall_e$$

$$(b_u)_2 = (b_u)_2 + \frac{1}{4}\left[-\overline{L}_i p_i + (S_u)_e\right]\Delta\forall_e$$

$$(b_u)_3 = (b_u)_3 + \frac{1}{4}\left[-\overline{L}_i p_i + (S_u)_e\right]\Delta\forall_e$$

$$(b_u)_4 = (b_u)_4 + \frac{1}{4}\left[-\overline{L}_i p_i + (S_u)_e\right]\Delta\forall_e$$

$$(b_v)_1 = (b_v)_1 + \frac{1}{4}\left[-\overline{M}_i p_i + (S_v)_e\right]\Delta\forall_e$$
$$(b_v)_2 = (b_v)_2 + \frac{1}{4}\left[-\overline{M}_i p_i + (S_v)_e\right]\Delta\forall_e$$
$$(b_v)_3 = (b_v)_3 + \frac{1}{4}\left[-\overline{M}_i p_i + (S_v)_e\right]\Delta\forall_e$$
$$(b_v)_4 = (b_v)_4 + \frac{1}{4}\left[-\overline{M}_i p_i + (S_v)_e\right]\Delta\forall_e$$

$$(b_w)_1 = (b_w)_1 + \frac{1}{4}\left[-\overline{N}_i p_i + (S_w)_e\right]\Delta\forall_e$$
$$(b_w)_2 = (b_w)_2 + \frac{1}{4}\left[-\overline{N}_i p_i + (S_w)_e\right]\Delta\forall_e$$
$$(b_w)_3 = (b_w)_3 + \frac{1}{4}\left[-\overline{N}_i p_i + (S_w)_e\right]\Delta\forall_e$$
$$(b_w)_4 = (b_w)_4 + \frac{1}{4}\left[-\overline{N}_i p_i + (S_w)_e\right]\Delta\forall_e$$

3.2.5 Generic Pressure Equation

The velocity interpolation discussed in Sec. 3.1.1 is used to avoid spurious pressure oscillations. The form of the momentum equations that are interpolated and substituted into the mass conservation equations are

$$\tilde{u} = \hat{u} - d_u\left[\frac{\partial p}{\partial x}\right]_e \tag{3.139}$$

$$\tilde{v} = \hat{v} - d_v\left[\frac{\partial p}{\partial y}\right]_e \tag{3.140}$$

$$\tilde{w} = \hat{w} - d_w\left[\frac{\partial p}{\partial z}\right]_e \quad . \tag{3.141}$$

The continuity equation is integrated over the control volume and has the form

$$\sum_{CVf=1}^{nfaces}\int_{CVf}\rho\left[\tilde{u}dA_x + \tilde{v}dA_y + \tilde{w}dA_z\right] = 0 \ , \tag{3.142}$$

where CVf is the interior control volume face and $nfaces$ are the total number of faces for each control volume. The same quadratic quadrature is used to integrate the continuity equation (see

Sec. 3.2.2 and integration in Eqs. 3.113-3.116). Integration of the continuity equation leads to

$$\begin{aligned}
&\int_{CVf} \rho\Big[\tilde{u}dA_x + \tilde{v}dA_y + \tilde{w}dA_z\Big] \\
&= \rho\Big(\Big[2\tilde{u}(\overline{c-e}) + \tilde{u}(\overline{c-f_a}) + \tilde{u}(\overline{e-f_a}) + \tilde{u}(\overline{c-f_b}) + \tilde{u}(\overline{e-f_b})\Big]\frac{\Delta A_X}{3} \\
&\quad + \Big[2\tilde{v}(\overline{c-e}) + \tilde{v}(\overline{c-f_a}) + \tilde{v}(\overline{e-f_a}) + \tilde{v}(\overline{c-f_b}) + \tilde{v}(\overline{e-f_b})\Big]\frac{\Delta A_Y}{3} \\
&\quad + \Big[2\tilde{w}(\overline{c-e}) + \tilde{w}(\overline{c-f_a}) + \tilde{w}(\overline{e-f_a}) + \tilde{w}(\overline{c-f_b}) + \tilde{w}(\overline{e-f_b})\Big]\frac{\Delta A_Z}{3}\Big) \\
&= \rho\Big(\Big[2\hat{u}(\overline{c-e}) + \hat{u}(\overline{c-f_a}) + \hat{u}(\overline{e-f_a}) + \hat{u}(\overline{c-f_b}) + \hat{u}(\overline{e-f_b})\Big]\frac{\Delta A_X}{3} \\
&\quad + \Big[2\hat{v}(\overline{c-e}) + \hat{v}(\overline{c-f_a}) + \hat{v}(\overline{e-f_a}) + \hat{v}(\overline{c-f_b}) + \hat{v}(\overline{e-f_b})\Big]\frac{\Delta A_Y}{3} \\
&\quad + \Big[2\hat{w}(\overline{c-e}) + \hat{w}(\overline{c-f_a}) + \hat{w}(\overline{e-f_a}) + \hat{w}(\overline{c-f_b}) + \hat{w}(\overline{e-f_b})\Big]\frac{\Delta A_Z}{3}\Big) \\
&\; -\rho\Big(\Big[2d_u(\overline{c-e}) + d_u(\overline{c-f_a}) + d_u(\overline{e-f_a}) + d_u(\overline{c-f_b}) + d_u(\overline{e-f_b})\Big]\frac{\Delta A_X}{3}\overline{L}_i \\
&\quad + \Big[2d_v(\overline{c-e}) + d_v(\overline{c-f_a}) + d_v(\overline{e-f_a}) + d_v(\overline{c-f_b}) + d_v(\overline{e-f_b})\Big]\frac{\Delta A_Y}{3}\overline{M}_i \\
&\quad + \Big[2d_w(\overline{c-e}) + d_w(\overline{c-f_a}) + d_w(\overline{e-f_a}) + d_w(\overline{c-f_b}) + d_w(\overline{e-f_b})\Big]\frac{\Delta A_Z}{3}\overline{N}_i\Big)p_i \qquad (3.143) \\
&\quad = \hat{D}_{CVf} + D_{CVf,i}p_i\ . \qquad (3.144)
\end{aligned}$$

Using the convention in Table 3.1, the pressure coefficients and sources are computed by looping through all elements as follows

For $n = 1$, *nele*

$$(a_{p-P})_1 = (a_{p-P})_1 + D_{1,1} - D_{3,1} + D_{4,1}$$

$$(a_{p-nb-2})_1 = (a_{p-nb-2})_1 - D_{1,2} + D_{3,2} - D_{4,2}$$

$$(a_{p-nb-3})_1 = (a_{p-nb-3})_1 - D_{1,3} + D_{3,3} - D_{4,3}$$

$$(a_{p-nb-4})_1 = (a_{p-nb-4})_1 - D_{1,4} + D_{3,4} - D_{4,4}$$

$$(b_{p-p})_1 = (b_{p-p})_1 - \hat{D}_1 + \hat{D}_3 - \hat{D}_4$$

$$
\begin{aligned}
(a_{p-P})_2 &= (a_{p-P})_2 - D_{1,2} + D_{2,2} + D_{5,2} \\
(a_{p-nb-1})_2 &= (a_{p-nb-1})_2 + D_{1,1} - D_{2,1} - D_{5,1} \\
(a_{p-nb-3})_2 &= (a_{p-nb-3})_2 + D_{1,3} - D_{2,3} - D_{5,3} \\
(a_{p-nb-4})_2 &= (a_{p-nb-4})_2 + D_{1,4} - D_{2,4} - D_{5,4} \\
(b_{p-p})_2 &= (b_{p-p})_2 + \hat{D}_1 - \hat{D}_2 - \hat{D}_5
\end{aligned}
$$

$$
\begin{aligned}
(a_{p-P})_3 &= (a_{p-P})_3 - D_{2,3} + D_{3,3} + D_{6,3} \\
(a_{p-nb-1})_3 &= (a_{p-nb-1})_3 + D_{2,1} - D_{3,1} - D_{6,1} \\
(a_{p-nb-2})_3 &= (a_{p-nb-2})_3 + D_{2,2} - D_{3,2} - D_{6,2} \\
(a_{p-nb-4})_3 &= (a_{p-nb-4})_3 + D_{2,4} - D_{3,4} - D_{6,4} \\
(b_{p-p})_3 &= (b_{p-p})_3 + \hat{D}_2 - \hat{D}_3 - \hat{D}_6
\end{aligned}
$$

$$
\begin{aligned}
(a_{p-P})_4 &= (a_{p-P})_4 - D_{4,4} - D_{5,4} - D_{6,4} \\
(a_{p-nb-1})_4 &= (a_{p-nb-1})_4 + D_{4,1} + D_{5,1} + D_{6,1} \\
(a_{p-nb-2})_4 &= (a_{p-nb-2})_4 + D_{4,2} + D_{5,2} + D_{6,2} \\
(a_{p-nb-3})_4 &= (a_{p-nb-3})_4 + D_{4,3} + D_{5,3} + D_{6,3} \\
(b_{p-p})_3 &= (b_{p-p})_3 + \hat{D}_4 + \hat{D}_5 + \hat{D}_6 \ ,
\end{aligned}
$$

where the subscripts on pressure coefficients and source (a_p and b_p) are the local node number, the subscripts for $D_{f,i}$ represent the local node i and the local face f, and the subscript on $\hat{D}$ represents the local face number.

The boundary condition for pressure is the same as for the triangular grid (see Sec. 3.1.6). With the discretization, integration, and coefficients defined for the momentum and pressure equations, the algorithms follow the same procedure as given in Sec. 2.3.

3.3 Results

3.3.1 Lid-Driven Cavity

To validate the unstructured grid formulation, the driven cavity problem described in Sec. 2.4.1 is simulated at a Reynolds number of 100 on a triangulated uniform Cartesian grid. Two grids are tested with 20 by 20 and 40 by 40 grid spacing. The steady results for the three algorithms tested (C-N SIMPLER, RK-SIMPLER, and IRK-SIMPLER) all match, and are compared to Wirogo's results for the same case on a structured Cartesian grid that used the second order Flux Corrected Method [29]. The results on both grids match the results from Wirogo well, verifying the unstructured grid accurately simulates steady flow.

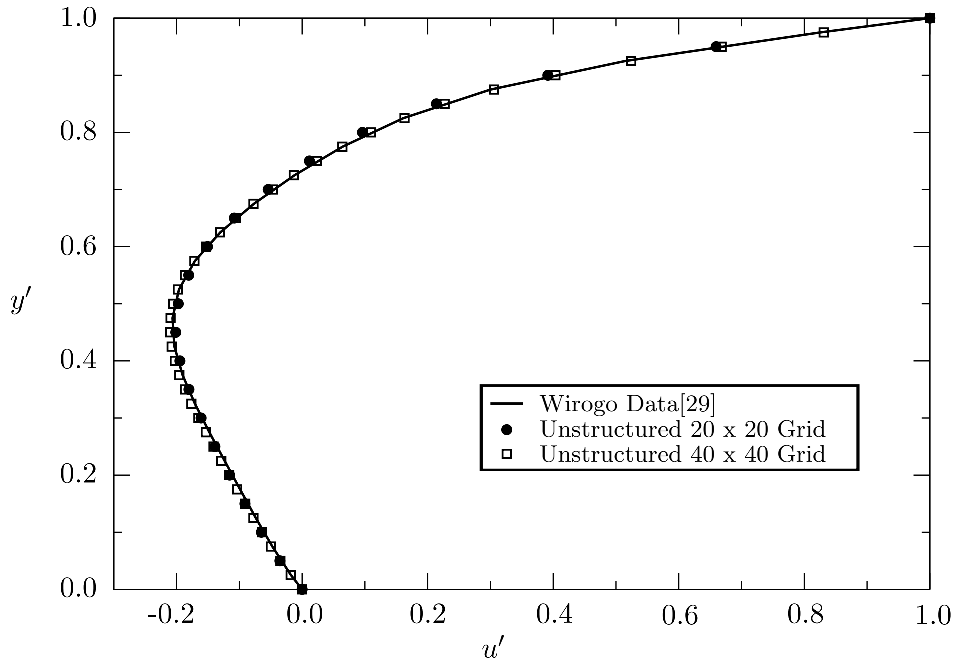


Figure 3.9: Unstructured driven cavity centerline velocity (Re=100).

3.3.2 Thin Flat Plate Normal to the Flow

The next test verifies unsteady flows are accurately simulated with the triangular grid and compares the runtime of the algorithms. The test case is the unsteady flow over a thin, flat plate described in Sec. 2.4.3.1. The flat plate surfaces are no-slip walls, the left boundary is uniform inflow, the right boundary is a velocity outlet corrected for mass conservation, and the top and bottom boundaries are set to free-stream conditions. The domain is rectangular with a spacing of $20L$ between the plate and the inlet, as well as the plate and the top and bottom boundaries, and a spacing of $40L$ between the plate and the outlet. The grid on the plate surface has a spacing of $L/75$ along the top and bottom and $L/60$ on the left and right. The grid has 16,504 nodes and 32,702 elements, mostly clustered near the body and in the wake region. Simulations start impulsively and are run until a non-dimensional time of $t' = t/(L/U_i) = 400$. The traditional SIMPLER algorithm with Crank-Nicolson time integration is the baseline case with the number of sub-iterations within each time step fixed at 30.

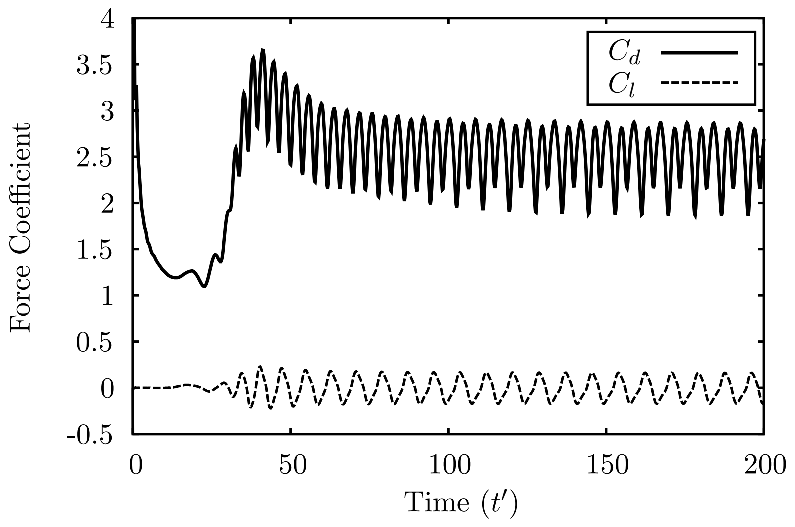


Figure 3.10: Flat plate coefficient of drag and lift.

The coefficient of drag (C_d) and lift (C_l) are plotted versus time in Fig. 3.10 for IRK-SIMPLER, with $\Delta t' = 2.0E-1$. Table 3.2 shows the average coefficient of drag, Strouhal number, and runtime for various time step sizes with Crank-Nicolson based SIMPLER (C-N), RK-SIMPLER (RK), and

IRK-SIMPLER (IRK). All simulations use the same unstructured triangular grid. The runtime (RT_{cvg}) is the wall time it takes for the simulation to reach the unsteady converged region when oscillations become uniform. C-N SIMPLER and IRK-SIMPLER converge for time steps up to $\Delta t' = 2.0E-1$, while RK-SIMPLER only converges for time steps up to $\Delta t' = 4.0E-3$.

Table 3.2: Flat plate simulation results at $Re = 17,800$.

$\Delta t'$	$\overline{C_d}$			Sr			RT_{cvg}[2]		
	IRK	C-N	RK	IRK	C-N	RK	IRK	C-N	RK
$2.0E-1$	2.489	2.596		0.1184	0.1187		16.2	70.9	
$1.6E-1$	2.496	2.569		0.1185	0.1186		20.8	85.9	
$1.2E-1$	2.497	2.544		0.1186	0.1185		26.3	111	
$8.0E-2$	2.503	2.527		0.1187	0.1185		40.8	176	
$4.0E-2$	2.509	2.519		0.1187	0.1185		66.0	185	
$2.0E-2$	2.514	2.519		0.1187	0.1185		92.5	367	
$1.6E-2$	2.515	2.520		0.1187	0.1185		104	456	
$1.2E-2$	2.517	2.521		0.1187	0.1185		148	612	
$8.0E-3$	2.519	2.522		0.1186	0.1185		206	958	
$4.0E-3$			2.518			0.1184			43.1
$2.0E-3$			2.521			0.1185			88.6
$1.6E-3$			2.522			0.1185			104
$1.2E-3$			2.523			0.1185			137
$8.0E-4$			2.524			0.1185			207
$4.0E-4$			2.527			0.1185			426

The average coefficient of drag and Strouhal number are plotted against time step size in Fig. 3.11. The average coefficient of drag and Strouhal numbers continue to change as the time step size decreases and do not converge to one value. The reason for this is explained by looking at the formulation used to derive the pressure equation. Multiple researchers have shown grids that store pressure and velocity at the same location (often called collocated grids) can lead to spurious pressure oscillations for small time step sizes and relaxation factors [60],[61],[62],[63]. This is likely the reason why the coefficient of drag and Strouhal number do not converge as expected with

[2] RT_{cvg} in minutes.

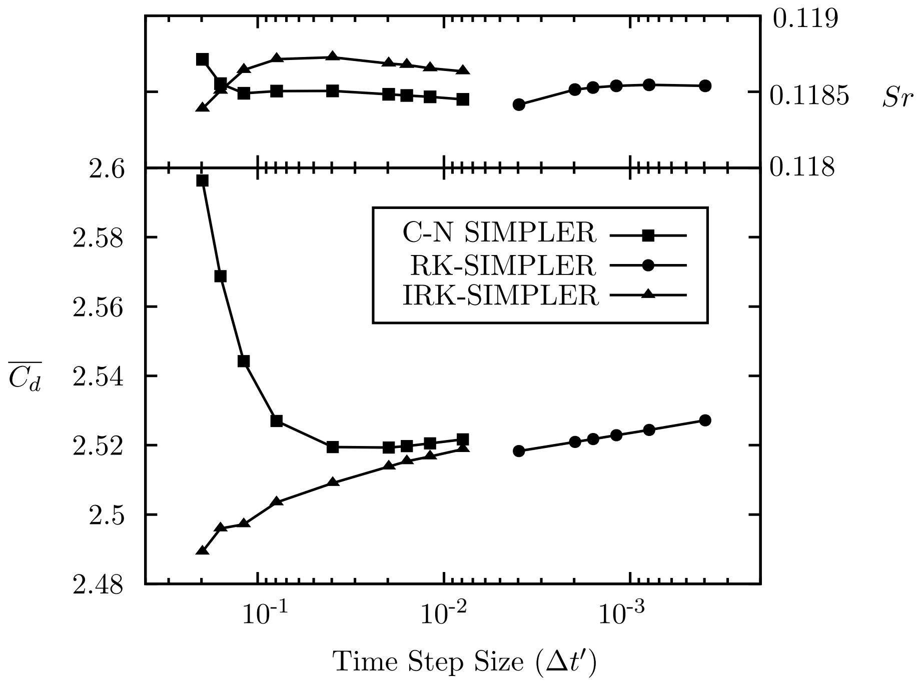


Figure 3.11: Flat plate Strouhal number and coefficient of drag and lift.

decreasing time step size. Pressure contours in Figs. 3.12 and 3.13 show that for successively smaller time step sizes, the pressure field goes from smooth to oscillatory (this is present in all algorithms tested). Researchers present methods by which this effect is reduced or eliminated [60],[61],[62],[63], but examining these different methods is not in the scope of present research.

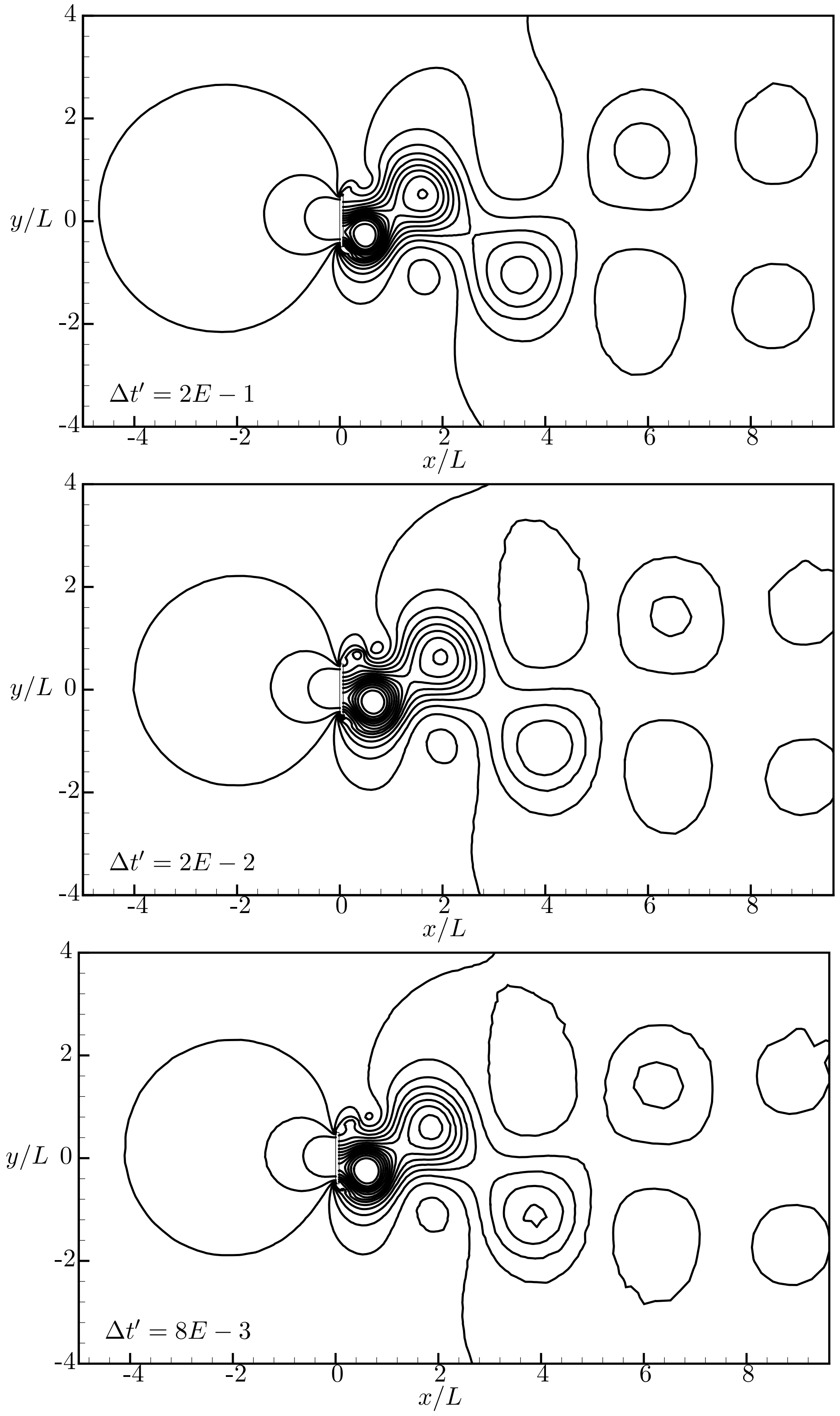


Figure 3.12: IRK-SIMPLER pressure contours for the flat plate (levels at $\Delta p/(0.5\rho U_i^2) = 0.284$).

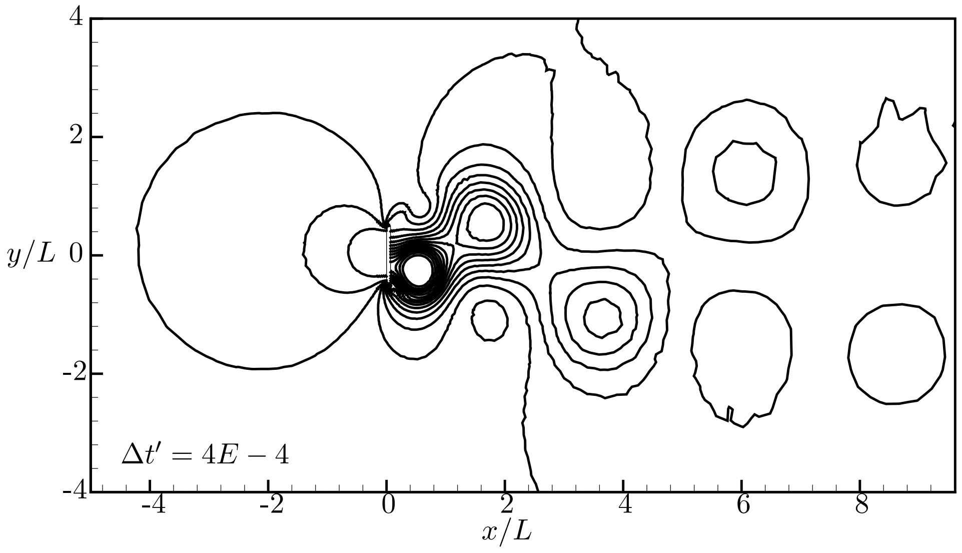
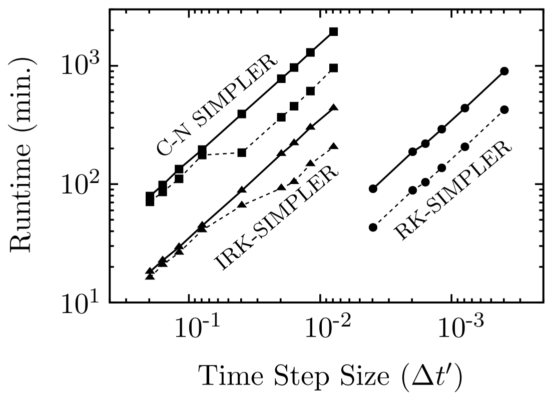


Figure 3.13: RK-SIMPLER pressure contours for the flat plate (levels at $\Delta p/(0.5\rho U_i^2) = 0.284$).

Figure 3.14 shows the runtime for the time steps tested. Figure 3.15 shows mass and momentum residuals averaged for all time steps after unsteady convergence is met. The residuals show C-N SIMPLER has lower mass residual than IRK-SIMPLER for the same time step size, but the momentum residuals for C-N SIMPLER do not drop as rapidly with decreasing time step size as IRK-SIMPLER. For time step sizes below about $\Delta t' = 6E-2$ IRK-SIMPLER has lower momentum residual than C-N SIMPLER.



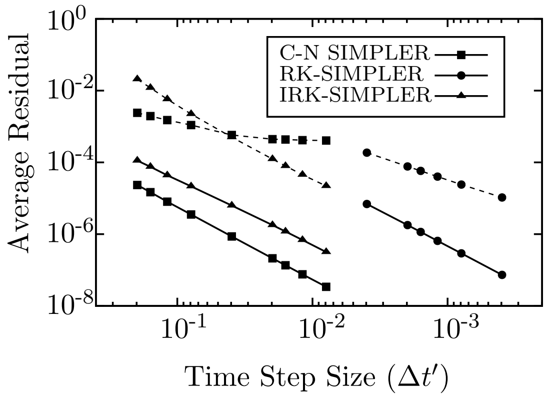


Figure 3.14: Flat plate runtime[a].

[a]Solid lines are total runtime, and dashed lines are convergence runtime.

Figure 3.15: Flat plate average residual[a].

[a]Solid lines are mass residual, and dashed lines are momentum residual.

Despite issues with the spurious pressure fields, the IRK-SIMPLER algorithm results match the C-N SIMPLER and RK-SIMPLER algorithms as well as previous simulation results [53],[54] (which report a wide range of results for average coefficient of drag of 2.34-3.25 and Strouhal number of 0.12-0.15). If it is assumed the spurious pressure field causes an increase in drag as the time step size decreases, then Fig. 3.11 suggests IRK-SIMPLER results in more accurate coefficient of drag than C-N SIMPLER for larger time steps. Certainly for the largest time step size allowed of $\Delta t' = 2.0E-1$ IRK-SIMPLER's result of 2.489 is more accurate than C-N SIMPLER's result of 2.596 on this grid.

Table 3.3 shows the average runtime each algorithm requires to complete one time step. IRK-SIMPLER is able to take the same time steps as C-N SIMPLER but only takes about 24% of the time to complete each step. RK-SIMPLER only takes about 2.4% as much time as C-N SIMPLER and about 10% as much time as IRK-SIMPLER, but RK-SIMPLER is required to take much smaller time steps.

Table 3.3: Average runtime per time step for the flat plate.

Algorithm	Seconds/Step
C-N SIMPLER	2.213
IRK-SIMPLER	0.534
RK-SIMPLER	0.054

3.3.3 Comparison of Structured and Unstructured Runtime

Both structured and unstructured grids are tested with C-N SIMPLER, RK-SIMPLER, and IRK-SIMPLER. IRK-SIMPLER results in lower runtime, with greater speedup in the structured grid. Table 3.4 shows the seconds per time step required on the flat plate case relative to C-N SIMPLER for RK-SIMPLER and IRK-SIMPLER on both structured and unstructured grids. RK-SIMPLER has a similar value for both structured and unstructured grids (around 2.5% of C-N SIMPLER). IRK-SIMPLER has a value of about 9% for the structured grid, but that more than doubles to 24% for the unstructured grid.

Table 3.4: Normalized runtime for the flat plate case[3].

Algorithm	Normalized Runtime/Step	
	Structured Grid (0.476)	Unstructured Grid (2.213)
RK-SIMPLER	0.026	0.024
IRK-SIMPLER	0.092	0.241

Table 3.5 shows the percent of runtime each algorithm spends solving linear equations for the flat plate problem. The percent of runtime for all algorithms on unstructured grids is much lower than structured grids, by about half for C-N SIMPLER and IRK-SIMPLER and by almost a quarter for RK-SIMPLER. This means relatively more time is spent calculating fluxes and coefficients for the unstructured grid.

Table 3.5: Percent of runtime spent solving linear equations for the flat plate case.

Algorithm	Percent Time in Linear Solvers	
	Structured Grid	Unstructured Grid
C-N SIMPLER	84.6%	41.8%
RK-SIMPLER	58.5%	15.1%
IRK-SIMPLER	68.7%	38.5%

The RK-SIMPLER algorithm only has one implicit equation (pressure equation) and also only computes coefficients once per time step. IRK-SIMPLER has more implicit equations to solve, and also recalculates the coefficients multiple times each time step. Because the calculation of the coefficients requires much more effort for the unstructured grids, the IRK-SIMPLER algorithm does not reduce the time compared to SIMPLER as much on the unstructured grid.

3.4 Conclusions

IRK-SIMPLER accurately and efficiently simulates flows on unstructured grids. For the flat plate case, IRK-SIMPLER is four times faster than C-N SIMPLER each time step with similar stability and more accurate solutions at higher time step sizes.

[3]C-N SIMPLER actual seconds/step given in parentheses.

Compared to the structured simulations in Chapter 2, the reduction in runtime for IRK-SIMPLER is not as much. Part of the reason for this is the increased time required for both calculation of momentum coefficients (with a shape function calculated for each element every time the coefficients are computed) as well as the linear solvers (Gauss-Seidel is primarily used and has much slower convergence than the line-by-line TDMA method used for the structured simulations). One future area to look for improvement in the IRK-SIMPLER algorithm is linear solvers with faster convergence rates, such as BiCGSTAB or GMRES.

An issue with pressure oscillations was seen when small time steps and large grid cells are used. The velocity interpolation method of Prakash [31] does not result in pressure oscillations for steady problems and when large time steps are taken, but as the time step size decreases pressure oscillations start to occur. Other researchers find similar problems for other incompressible methods on collocated grids [61],[62],[63].

CHAPTER 4. MOMENTUM FLUX INTERPOLATION

For traditional finite volume methods, values of two neighboring control volumes are used to compute the flux at an interface. The central difference method, where values are linearly interpolated from each grid point to the interface, is a commonly used discretization, however these methods have stability issues for inviscid fluxes.

To avoid central difference stability issues, two common approaches are used. The first uses a central difference scheme for the viscous fluxes, and a more stable scheme for the inviscid fluxes, such as upwinding (ex. first order upwind, second order upwind (SOU) [25], quadratic upstream interpolation for convective kinematics (QUICK) [26]), essentially non-oscillatory (ENO) [27], or weighted ENO (WENO) [28] schemes. The second approach is to use a more stable physics-based method to calculate the total viscous and inviscid fluxes with one scheme. The Exponential, Hybrid, and Power Law schemes [5] do this by finding the local solution to the steady, one-dimensional convection-diffusion equation. These schemes provide a stable way to calculate fluxes, but are overly diffusive.

An improved physics-based scheme, called the flux correction method (FCM), is developed by Wirogo [29] for structured grids. FCM uses the local solution to the convection-diffusion equation with sources (including unsteady, pressure, non-directional flux, and external sources) to find the flux at an interface. The FCM scheme improves accuracy over the Power Law scheme while maintaining good stability [29].

On structured grids, methods like SOU, QUICK, and higher order WENO schemes improve the accuracy of the flux computation using a larger stencil along the grid lines and including more than two neighboring nodes. For unstructured grids, increasing the stencil does not follow as naturally, and improving the accuracy of flux computation becomes complex.

Advances in finite element methods show the ability to achieve a high order of accuracy by increasing the polynomial order and number of nodes used in the shape function within each cell [64],[65]. Using a vertex-centered unstructured grid instead of cell-centered, the idea of fitting a polynomial over each cell is used. When using this vertex-centered method (Fig. 3.1) the control volume faces are in the middle of the cells, and the flow variables are known along these faces by fitting a shape function through the cell vertices.

A Power Law scheme was developed for triangular grids, similar to the structured Power Law scheme [30]. Like the structured Power Law scheme, the unstructured vertex-centered Power Law scheme has better stability than the linear shape function, but it also is too diffusive. The present research develops a flux correction method (FCM) for unstructured grids to improve the accuracy of the Power Law scheme, without significantly increasing the computational effort required.

Methods for both structured and unstructured grids are investigated in this Chapter.

4.1 Methods for Structured Grids

4.1.1 Methods of Interest

Five methods for structured grids are examined, starting with the central difference method, two upwind methods (first order and QUICK), and lastly two physics based methods (Power Law and the Flux Corrected Method). Figure 4.1 shows the nomenclature used for the structured grid when finding the flux on face e between grid points P and E, assuming a Cartesian grid (although non-Cartesian structured grids follow similarly). The following definitions are used in the different schemes

$$(\delta x)_w = x_P - x_W \qquad (\delta x)_e = x_E - x_P \qquad (\delta x)_{ee} = x_{EE} - x_E \tag{4.1}$$

$$(f_+)_e = \frac{x_E - x_e}{x_E - x_P} \qquad (f_-)_e = \frac{x_e - x_P}{x_E - x_P} \tag{4.2}$$

$$F_e = (\rho u)_e \qquad D_e = \frac{\mu_e}{(\delta x)_e} \ . \tag{4.3}$$

For uniformly spaced grids $(f_-)_e = (f_+)_e = 0.5$.

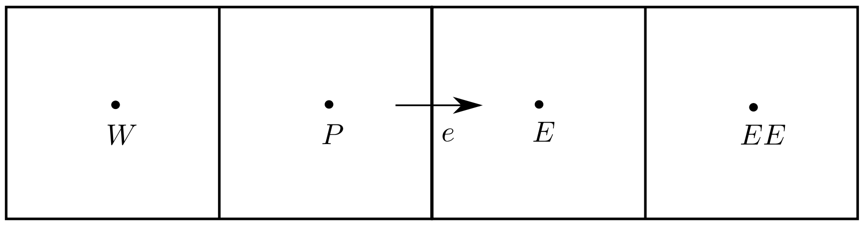


Figure 4.1: Grid nomenclature for structured schemes when finding flux at interface e.

The x direction momentum flux being interpolated has the form

$$J = \rho u \phi - \mu \frac{d\phi}{dx} \ , \tag{4.4}$$

where J is the momentum flux (inviscid and viscous).

4.1.1.1 Central Difference

The central difference method is a common and simple method based on linear interpolation between two points. Using the central difference to interpolate the total flux J results in

$$J_e = (\rho u)_e \phi_e - \mu_e \left(\frac{d\phi}{dx}\right)_e \tag{4.5}$$

$$= F_e \Big[(f_-)_e \phi_E + (f_+)_e \phi_P \Big] - D_e(\phi_E - \phi_P) \ . \tag{4.6}$$

4.1.1.2 Upwind

To avoid instability in inviscid flux interpolation, upwinding schemes are used for the inviscid fluxes, while the viscous fluxes are computed with the central difference scheme. The first order upwind scheme (or simply upwind scheme) takes the local flow direction into account, to decide whether to use the left or right point to determine the inviscid flux.

$$J_e = F_e \phi_e - D_e(\phi_E - \phi_P) \ , \tag{4.7}$$

with

$$\phi_e = \begin{cases} \phi_P, & \text{if } u_e > 0 \\ \phi_E, & \text{if } u_e < 0 \ . \end{cases} \tag{4.8}$$

This equation takes the nearest upwind grid point value of ϕ to calculate the interface flux.

The upwind scheme is only first order accurate but is a more stable scheme, especially for high Reynolds number flows. Another scheme not discussed here is a second order upwinding (SOU) scheme, that uses a linear extrapolation with two grid points upwind of the interface. The SOU scheme improves the accuracy to second order [66].

4.1.1.3 QUICK

Another upwinding scheme is the QUICK (Quadratic Upstream Interpolation for Convective Kinematics) scheme from Leonard [26]. The QUICK scheme uses three points to interpolate the inviscid flux, two grid points upwind and one grid point downwind. The implementation of the QUICK scheme (as well as SOU) using Lagrange interpolation is discussed in Rajagopalan and Yu [66], and is used in current testing. The QUICK scheme inviscid flux is calculated as

$$\phi_e = \begin{cases} \phi_P + \Big[\gamma_{1e}(\phi_W - \phi_P) + \gamma_{2e}(\phi_E - \phi_P)\Big], & \text{if } u_e > 0 \\ \phi_E + \Big[\delta_{1e}(\phi_P - \phi_E) + \delta_{2e}(\phi_{EE} - \phi_E)\Big], & \text{if } u_e < 0 \, , \end{cases} \tag{4.9}$$

with

$$\gamma_{1e} = \left[\frac{x_e - x_E}{x_W - x_E}\right]\left[\frac{x_e - x_P}{x_W - x_P}\right] \qquad \gamma_{2e} = \left[\frac{x_e - x_P}{x_E - x_P}\right]\left[\frac{x_e - x_W}{x_E - x_W}\right] \tag{4.10}$$

$$\delta_{1e} = \left[\frac{x_e - x_E}{x_P - x_E}\right]\left[\frac{x_e - x_{EE}}{x_P - x_{EE}}\right] \qquad \delta_{2e} = \left[\frac{x_e - x_P}{x_{EE} - x_P}\right]\left[\frac{x_e - x_E}{x_{EE} - x_E}\right] \, . \tag{4.11}$$

In this form the QUICK scheme can be thought of as the first order upwind scheme with correction terms to improve accuracy (terms in [ ] in Eq. 4.9). The number of grid points used in the interpolation can continue to be increased (at least until a boundary is met); however, the more points included, the more computations are required. Although the QUICK scheme uses a third order interpolation, for the finite volume formulation, only second order accuracy is found in the accuracy analysis.

4.1.1.4 Power Law

Already discussed in Sec. 2 is the Power Law scheme of Patankar [5]. Unlike the upwinding schemes discussed, the Power Law scheme is a method to interpolate the total momentum flux, including both inviscid and viscous fluxes. The Power Law scheme is derived by solving the steady one-dimension convection-diffusion equation without any sources.

$$\frac{\partial \rho u_e \phi}{\partial x} - \frac{\partial}{\partial x}\left[\Gamma \frac{\partial \phi}{\partial x}\right] = 0 \tag{4.12}$$

The exact solution to this equation is an exponential function, which is expensive to compute. The Hybrid and Power Law schemes are two different curve fits to the exponential function commonly used. All three of these schemes result in a flux of the form

$$J_e = \left[D_e A(|P_e|) + [\![-F_e, 0]\!]\right](\phi_P - \phi_E) + F_e \phi_P \ , \tag{4.13}$$

where $P_e = F_e/D_e$ is the grid Peclet number. The function $A(|P_e|)$ is defined for the Exponential, Hybrid, and Power Law schemes, as well as the upwind and central difference schemes in the following relationships.

$$A(|P_e|) = \frac{|P_e|}{[\exp(|P_e|) - 1]} \qquad \text{Exponential Scheme} \tag{4.14}$$

$$A(|P_e|) = [\![0, 1 - 0.5|P|]\!] \qquad \text{Hybrid Scheme} \tag{4.15}$$

$$A(|P_e|) = [\![0, (1 - 0.1|P|)^5]\!] \qquad \text{Power Law Scheme} \tag{4.16}$$

$$A(|P_e|) = 1 \qquad \text{Upwind Scheme} \tag{4.17}$$

$$A(|P_e|) = 1 - 0.5|P_e| \qquad \text{Central Difference Scheme} \tag{4.18}$$

This form of the central difference scheme is for the interface e located midway between E and P, or $(f_-)_e = (f_+)_e = 0.5$. If this is not the case, the function for central difference becomes $A(|P_e|) = 1 - (f_-)_e P_e - [\![-P_e, 0]\!]$. The QUICK scheme is equivalent to the upwind scheme with an additional term required for the correction terms (terms in [ ] in Eq. 4.9).

4.1.1.5 Flux Corrected Method

Wirogo [29] developed a scheme correcting the Power Law scheme by including unsteady, pressure gradient, and two-dimensional flow terms into the physics based interpolation. The starting point for the Flux Corrected Method (FCM) is the steady one-dimension convection-diffusion equation with a source.

$$\frac{\partial \rho u_e \phi}{\partial x} - \frac{\partial}{\partial x}\left[\Gamma \frac{\partial \phi}{\partial x}\right] = S_{\phi-x} \ , \tag{4.19}$$

where the source term ($S_{\phi-x}$) includes any additional terms in the momentum equations (namely unsteady, pressure gradient, two- or three-dimensional, and other external source terms).

The flux using the FCM term is defined by

$$J_e = \Big[D_e A(|P_e|) + [\![-F_e, 0]\!]\Big](\phi_P - \phi_E) + F_e\phi_P + (J_e)_S \ , \tag{4.20}$$

where the first part of the flux is equal to the Power Law scheme (or other schemes depending on the choice of $A(P_e)$), and the last term $(J_e)_S$ is the flux coming from the FCM source term. The additional flux term is defined by

$$(J)_S = \Big[(\delta x)_e Q(P_e) + (x - x_P)\Big] S_{\phi-x} \ . \tag{4.21}$$

This additional flux is dependent on the location of interest, and for the interface e the value of $x = x_e$ leads to

$$(J_e)_S = (\delta x)_e \Big[Q(P_e) + (f_-)_e\Big] S_{\phi-x} \ . \tag{4.22}$$

The function $Q(P_e)$ is defined by

$$Q(P_e) = \frac{P_e - \exp(P_e) + 1}{P_e \exp(P_e) - P_e} = \frac{1}{\exp(P_e) - 1} - \frac{1}{P_e} \ , \tag{4.23}$$

with the limit $Q(0) = -0.5$. This function can be recast in the form

$$Q(P_e) = \frac{A(|P_e|) - 1}{P_e} + \frac{[\![-F_e, 0]\!]}{F_e} \ . \tag{4.24}$$

Matching 4.23 to 4.24 requires the exponential form of $A(|P_e|)$ in Eq. 4.14, but the power law fit in Eq. 4.16 also reduces the computational cost. Figure 4.2 shows the value of $Q(P_e)$ over a range of P_e, using both the exponential and power law forms. The Q function has a minimum of -1.0, as $P_e \to -\infty$, and a maximum of 0.0, as $P_e \to \infty$. Using the power law fit shows no difference in Fig. 4.2, and upon closer examination small differences appear in the range $-4 < P_e < 4$, with a maximum difference of approximately 0.009 around $P_e = \pm 1.0$.

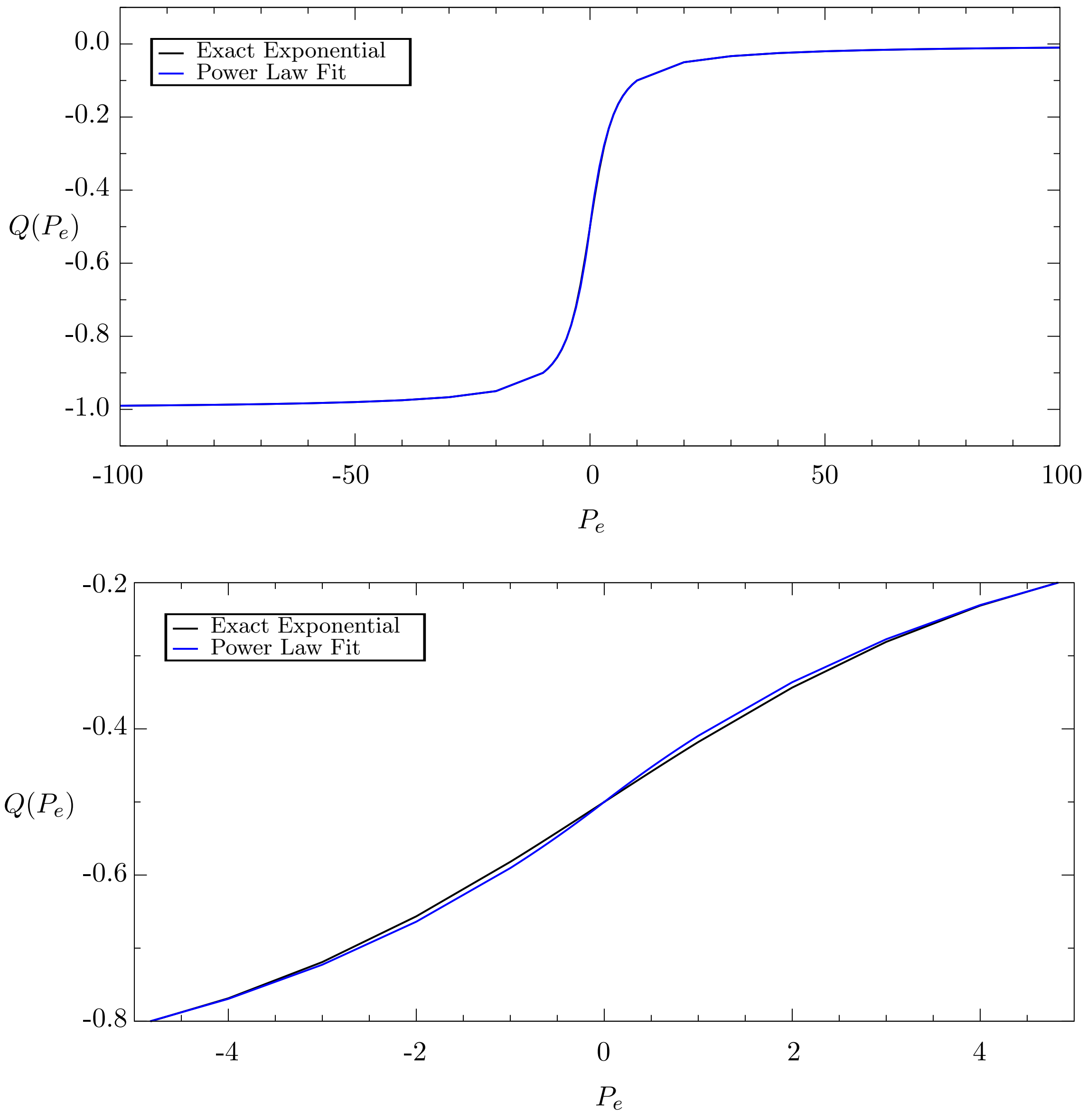


Figure 4.2: $Q(P_e)$ versus P_e with the exact exponential and the power law fit.

For the two-dimensional momentum equations, the FCM source term are

$$S_{u-x} = \left[-\frac{\partial \rho u}{\partial t}\right] + \left[-\frac{\partial \rho v_e u}{\partial y} + \frac{\partial}{\partial y}\left(\mu \frac{\partial u}{\partial y}\right)\right] + \left[-\frac{\partial p}{\partial x}\right] + \left[S_u\right] \tag{4.25}$$

$$S_{u-y} = \left[-\frac{\partial \rho u}{\partial t}\right] + \left[-\frac{\partial \rho u_e u}{\partial x} + \frac{\partial}{\partial x}\left(\mu \frac{\partial u}{\partial x}\right)\right] + \left[-\frac{\partial p}{\partial x}\right] + \left[S_u\right] \tag{4.26}$$

$$S_{v-x} = \left[-\frac{\partial \rho v}{\partial t}\right] + \left[-\frac{\partial \rho v_e v}{\partial y} + \frac{\partial}{\partial y}\left(\mu \frac{\partial v}{\partial y}\right)\right] + \left[-\frac{\partial p}{\partial y}\right] + \left[S_v\right] \tag{4.27}$$

$$S_{v-y} = \left[-\frac{\partial \rho v}{\partial t}\right] + \left[-\frac{\partial \rho u_e v}{\partial x} + \frac{\partial}{\partial x}\left(\mu \frac{\partial v}{\partial x}\right)\right] + \left[-\frac{\partial p}{\partial y}\right] + \left[S_v\right] . \tag{4.28}$$

The first term in Eqs. 4.25-4.28 is the unsteady term. The second term in Eqs. 4.25 and 4.28 is the directional flux term, and the second term in Eqs. 4.26 and 4.27 is the non-directional flux term (named for whether the flux direction matches the velocity direction). The third and fourth terms are pressure gradient and external source terms. Wirogo finds including all terms except the directional flux terms results in the most accurate scheme and is used for present research. The final source terms used are

$$S_{u-x} = \left[-\frac{\partial \rho u}{\partial t}\right] + \left[-\frac{\partial p}{\partial x}\right] + \left[S_u\right] \tag{4.29}$$

$$S_{u-y} = \left[-\frac{\partial \rho u}{\partial t}\right] + \left[-\frac{\partial \rho u_e u}{\partial x} + \frac{\partial}{\partial x}\left(\mu \frac{\partial u}{\partial x}\right)\right] + \left[-\frac{\partial p}{\partial x}\right] + \left[S_u\right] \tag{4.30}$$

$$S_{v-x} = \left[-\frac{\partial \rho v}{\partial t}\right] + \left[-\frac{\partial \rho v_e v}{\partial y} + \frac{\partial}{\partial y}\left(\mu \frac{\partial v}{\partial y}\right)\right] + \left[-\frac{\partial p}{\partial y}\right] + \left[S_v\right] \tag{4.31}$$

$$S_{v-y} = \left[-\frac{\partial \rho v}{\partial t}\right] + \left[-\frac{\partial p}{\partial y}\right] + \left[S_v\right] . \tag{4.32}$$

4.1.2 Results

4.1.2.1 Lid-Driven Cavity

The lid-driven cavity case (2.4.1) is used to compare the five different schemes discussed. A Reynolds number of 1,000 is used and the grid spacing of the uniform Cartesian structured grid is varied from 16 by 16 to 48 by 48. The $u' = u/U_{lid}$ error along the centerline $x' = x/L = 0.5$ is computed by comparing the current results to the results found on a highly refined grid of 1024 by 1024 using the central difference scheme. Figure 4.3 shows the error for the different schemes on the four grids tested. The slope of these lines show the spatial order of accuracy (dashed lines for reference slopes of first and second order are also shown).

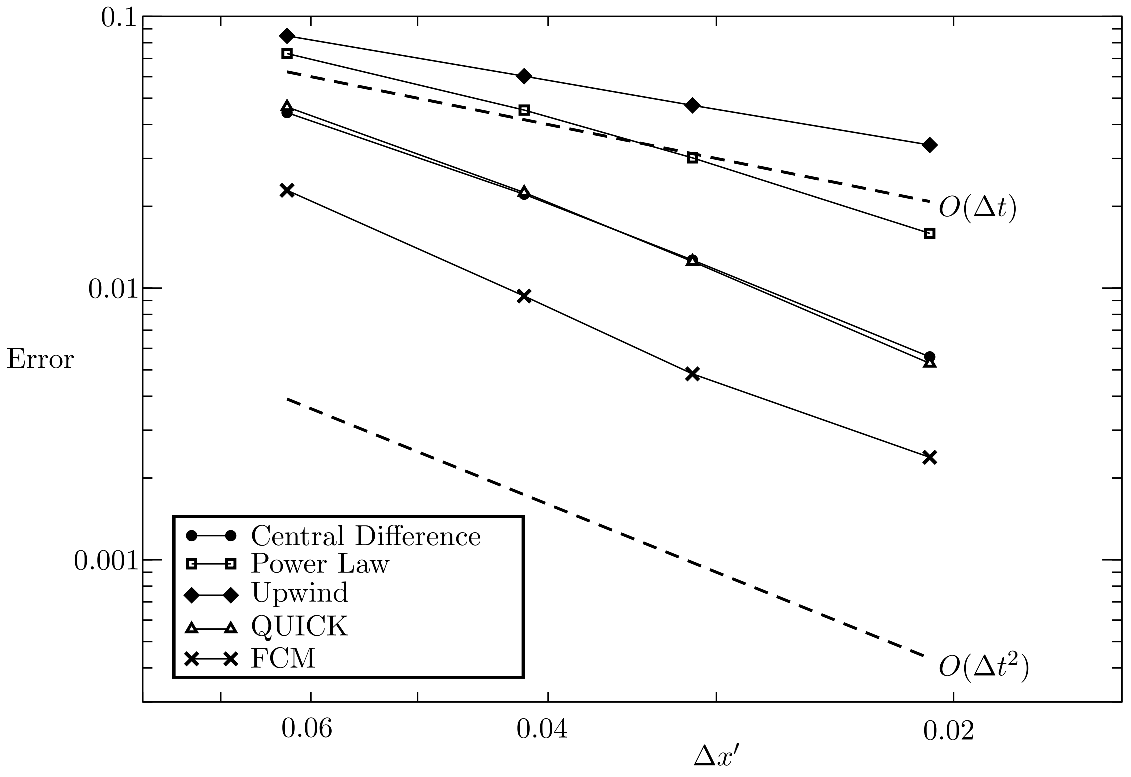


Figure 4.3: Spatial order of accuracy for the structured grid schemes.

The highest error is found in the upwind scheme which is first order accurate. The next highest error comes from the Power Law scheme, which is better than first order but not second order accurate. The central difference and QUICK schemes fall close to each other with second order accuracy. The lowest error comes from the FCM scheme, which is also second order accurate.

4.1.2.2 Three-Dimensional Square Cylinder

The next case is the flow over a square cylinder of infinite length. The three-dimensional square cylinder is shown in Fig. 4.4, with a cross-section that is L by L. The domain is $6L$ in the spanwise direction (z), with symmetric boundary conditions to represent an infinite square cylinder with a uniform inflow. Up to a Reynolds number of 150, the square cylinder sheds vortices with only-two dimensional flow and no variation in the z direction ($w = 0$ and $\frac{\partial u,v,p}{\partial z} = 0$), but above a Reynolds number of 150, secondary vortices form along the length of the cylinder with y vorticity [67]. For the present test case, a Reynolds number of 175 is used. The grid is 162 by 82 by 26, with 16 grid cells along the cylinder in the x and y directions and 24 grid cells along the cylinder in the z direction, and gives results that agree with previous simulation results by Saha [67]. The body surfaces are no-slip walls, the left boundary is uniform inflow, the right boundary is a velocity outflow corrected for mass conservation, and all other boundaries are inviscid walls. Simulations are started impulsively, with uniform freestream velocity and pressure, and are run until $t' = t/(L/U_i) = 400$.

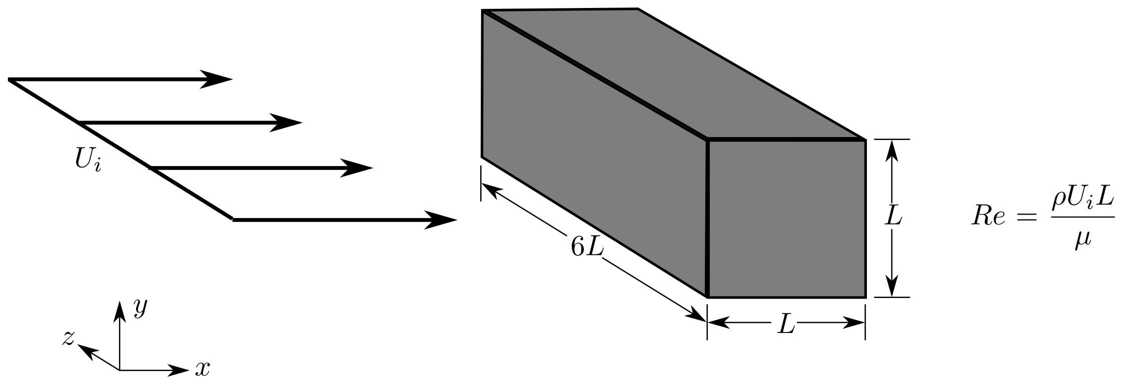
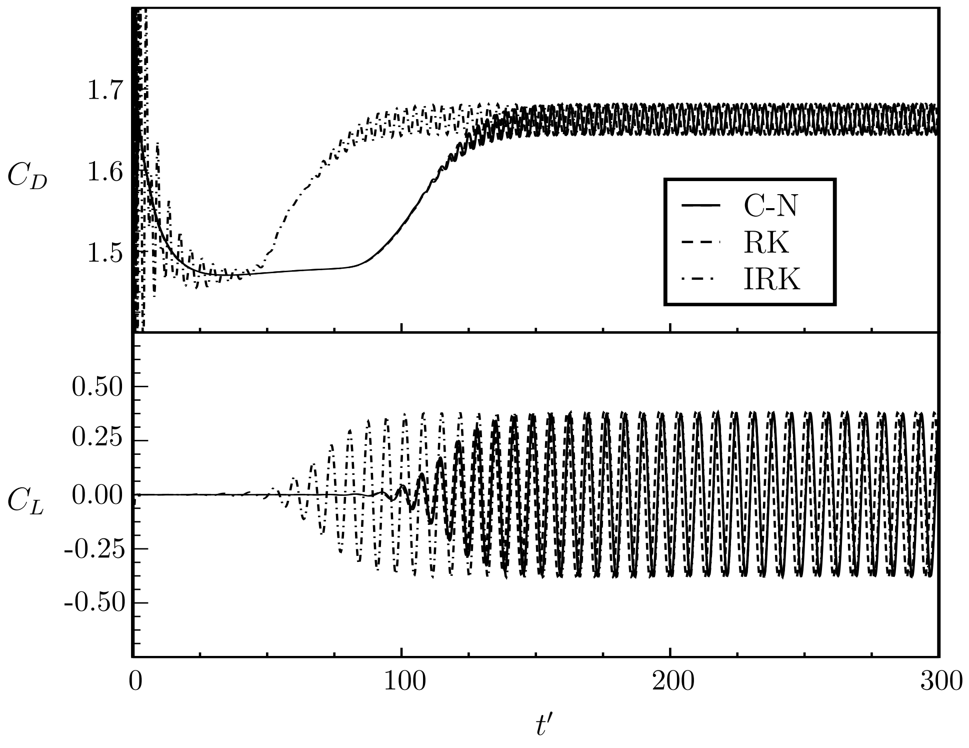


Figure 4.4: Schematic of the three-dimensional square cylinder problem.

The formulation for three-dimensional flow in Cartesian structured grids follows closely to the two-dimensional formulation found in Sec. 2.3. The C-N SIMPLER algorithm is the baseline, with the number of sub-iterations within each time step fixed at 20. Figure 4.5 shows the drag and lift coefficient on the square cylinder for the three algorithms tested using both the Power Law and QUICK schemes. The Power Law scheme does not show secondary vortices on this grid. The QUICK scheme results show the regular oscillations of lift and drag coefficient as seen in the Power Law scheme, but also non-regular oscillations that occur with lower frequency. These lower frequency oscillations are a result of secondary vortices (with y vorticity), as observed by Saha [67].



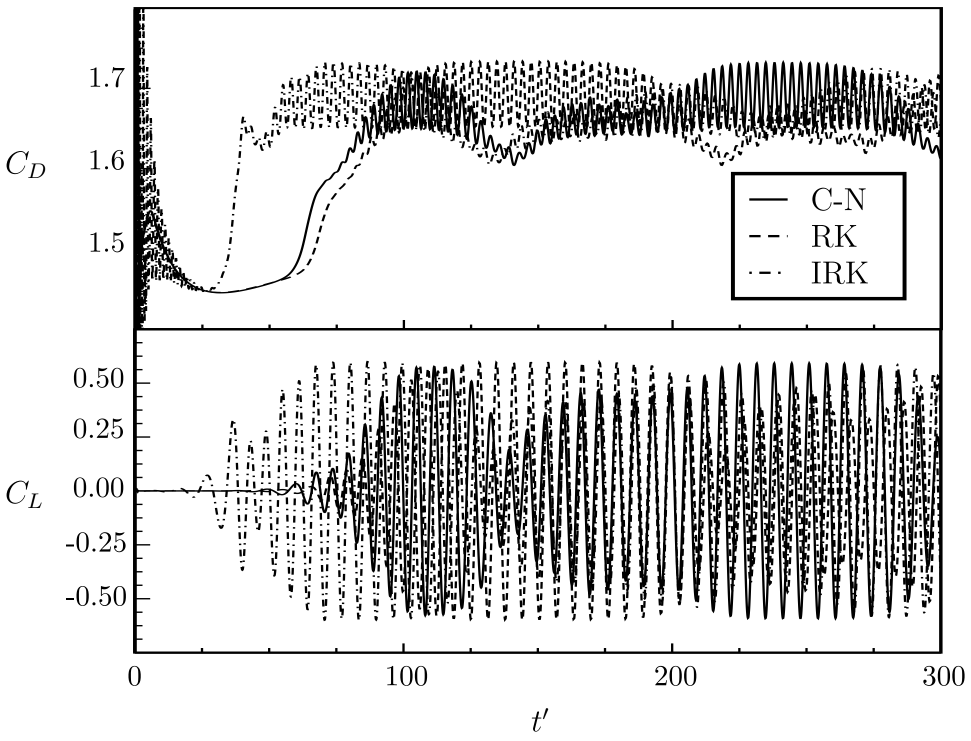


Figure 4.5: Drag and lift coefficient for the three-dimensional square cylinder with the Power Law (top) and QUICK (bottom) schemes.

Figure 4.6 shows the z-vorticity for both the Power Law and QUICK schemes at the midplane $z = 3L$. Near the body, where the grid is more refined, results are similar; but as the vortices are convected downstream, the Power Law scheme results in vortices that dissipate, while the QUICK scheme results in vortices that travel downstream and slowly dissipate.

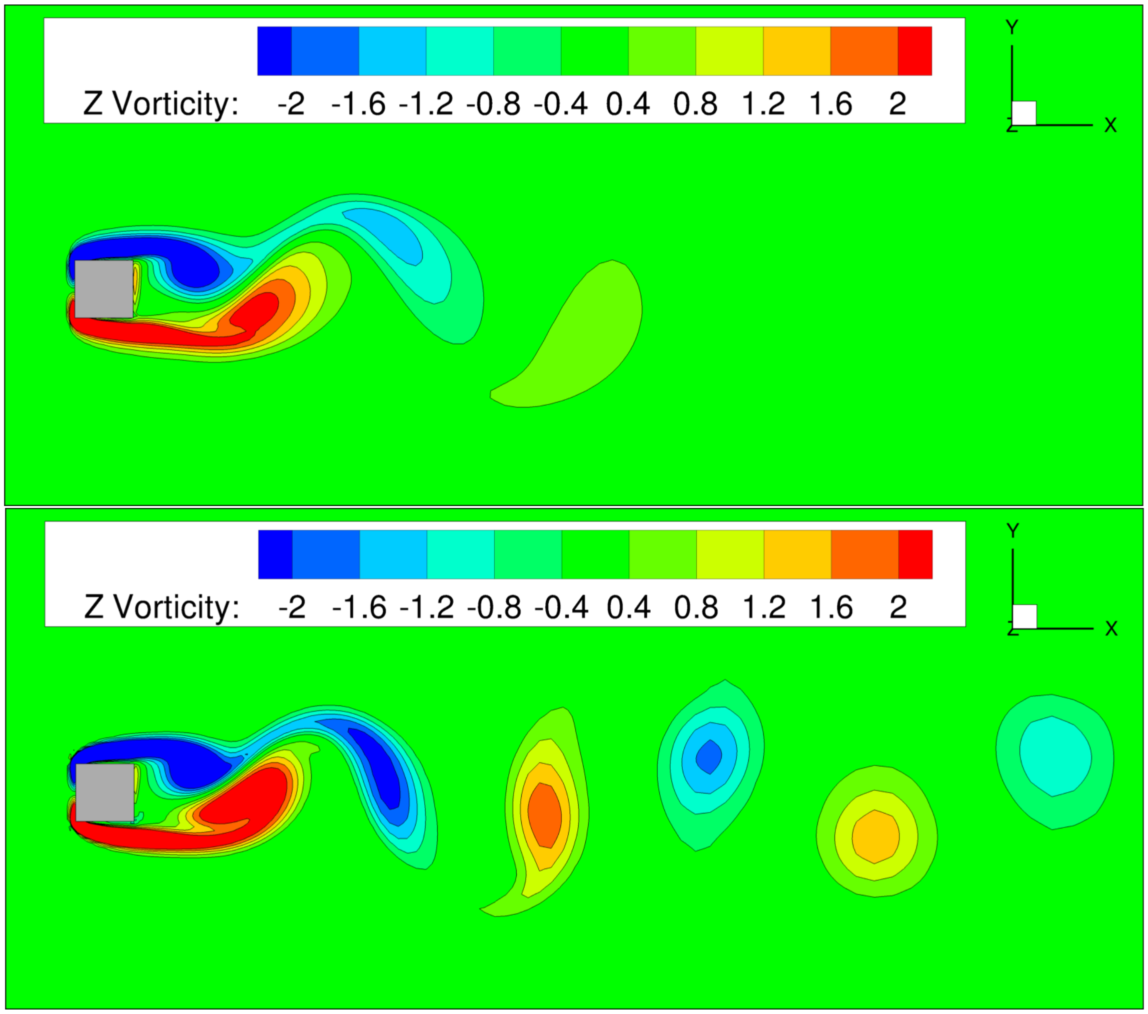


Figure 4.6: Three-dimensional square cylinder z-vorticity contours at the z midplane with the Power Law (top) and QUICK (bottom) schemes.

Figure 4.7 shows the y-vorticity for both the Power Law and QUICK schemes at the midplane through the center of the cylinder, $y = 0$. The Power Law results show no y-vorticity being produced, while QUICK results in a pattern of y-vorticity that closely matches the results obtained by Saha [67]. This shedding of y-vorticity causes lower frequency variation in the lift and drag coefficients.

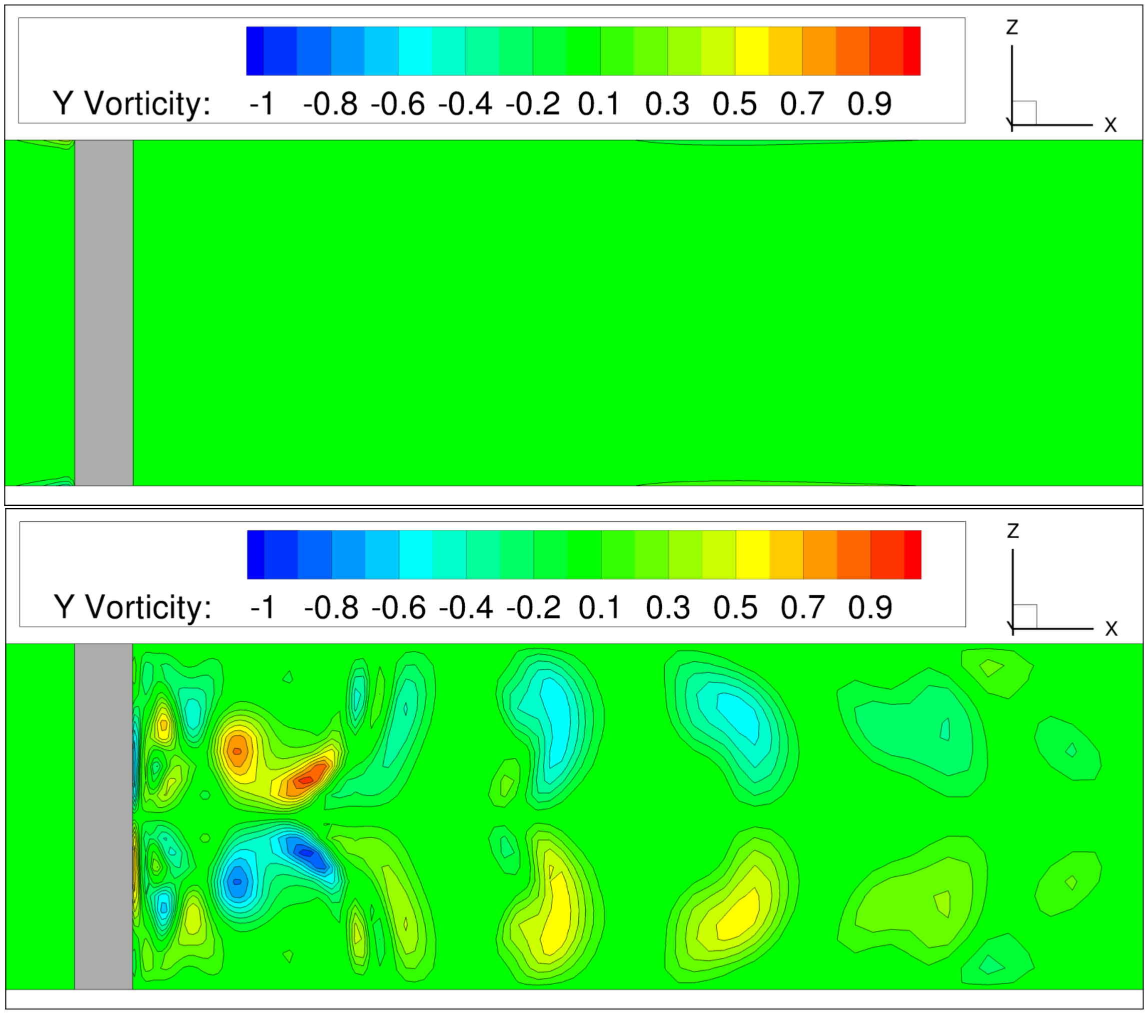


Figure 4.7: Three-dimensional square cylinder y-vorticity contours at the y midplane with the Power Law (top) and QUICK (bottom) schemes.

Table 4.1 shows time step restriction, runtime, average drag coefficients, and Strouhal number results for both the Power Law and QUICK schemes. The average coefficient of drag and Strouhal number approach 1.660 and 0.14, respectively, as time step size decreases for the Power Law simulations. For temporal accuracy of the Power Law simulations, an average drag coefficient of 1.660±0.002 is an acceptable range. The QUICK simulations result in an average drag coefficient and Strouhal number of 1.666 and 0.15, respectively, as time step size decreases. For temporal accuracy of QUICK simulations, an average coefficient of drag of 1.666±0.002 is used.

Table 4.1: Three-dimensional square cylinder results.

	$\frac{\Delta t_{max}}{L/U_i}$	$\overline{C_D}$ at Δt_{max}	Sr at Δt_{max}	$\frac{\Delta t_{Acc}}{L/U_i}$	CPU Time at Δt_{Acc} (min.)	Speedup[1] at Δt_{Acc}	Flux Scheme
C-N	1.000	1.796	0.112	0.100	1304.75	1.0	Power Law
RK	0.060	1.674	0.144	0.002	1181.05	1.1	
IRK	0.700	1.660	0.143	0.700	21.11	61.8	
C-N	0.300	1.670	0.126	0.100	1277.44	1.0	QUICK
RK	0.040	1.662	0.144	0.002	1187.55	1.1	
IRK	0.600	1.666	0.148	0.600	28.69	44.5	

The IRK-SIMPLER algorithm is able to take the largest time step size and give accurate results. IRK-SIMPLER also has the lowest runtime, with a speedup of 61.9 for the Power Law simulations and 44.5 for the QUICK simulations. For this case, RK-SIMPLER has about the same runtime as SIMPLER and results in little speedup (1.1 times).

4.1.3 Conclusions

Five different flux interpolation schemes are tested for the structured Cartesian grid. The FCM scheme has the lowest error for the lid-driven cavity case, and is second order accurate in space. The central difference and QUICK schemes are also second order accurate but with a larger error magnitude. The Power Law scheme is better than first order but did not result in second order accuracy. The upwind scheme is only first order accurate.

The Power Law and QUICK schemes are tested on the three-dimensional square cylinder case. On the same grid, Power Law is not able to capture the three-dimensional vortices found by other researchers, while the QUICK scheme accurately captures the full three-dimensionality of the flow.

[1]Speedup is relative to C-N SIMPLER.

4.2 Methods for Vertex-Centered Unstructured Grids

4.2.1 Methods of Interest

For both two- and three-dimensions, three methods are discussed: the Power Law scheme for unstructured vertex-centered grids, the central difference or linear element scheme, and a new FCM developed for unstructured grids.

4.2.1.1 Unstructured Power Law Scheme

Baliga [30] developed a scheme using the steady, two-dimensional convection-diffusion equation without any sources as a basis for a modified linear triangular element shape function. This scheme is similar to Patankar's Power Law scheme, and is called the unstructured Power Law scheme (or simply Power Law). This scheme is developed in Sec. 3.1.2, which should be referred to for details omitted in this section. The Power Law scheme is derived by starting with the steady, two-dimensional convection-diffusion equation with no sources

$$\frac{\partial \rho u_e \phi}{\partial x} + \frac{\partial \rho v_e \phi}{\partial y} - \frac{\partial}{\partial x}\left[\mu \frac{\partial \phi}{\partial x}\right] - \frac{\partial}{\partial y}\left[\mu \frac{\partial \phi}{\partial y}\right] = 0 \ . \tag{4.33}$$

After transforming to a coordinate system based on the element velocity direction, the form of the shape function is

$$\phi = A\xi(X) + BY + C \tag{4.34}$$

with coefficients A, B, and C defined in Eq. 3.18. The exact solution for $\xi(X)$ is an exponential function, and a Power Law fit found by Baliga is

$$\xi(X) = \frac{X - X_{max}}{P_e + [\![0, (1 - 0.1|P_e|)^5]\!]} \ . \tag{4.35}$$

The total momentum equation convective and diffusive flux is written in summation notation as

$$J_{\phi-X} = f_i \phi_i \qquad\qquad J_{\phi-Y} = g_i \phi_i \ , \tag{4.36}$$

with functions f_i and g_i defined in Eqs. 3.106 and 3.107.

4.2.1.2 Unstructured Central Difference Scheme - Linear Element

The shape function for the central difference scheme is represented in the same form as the Power Law scheme, with the simple modification of setting $\xi(X) = X$. This can also be accomplished without the coordinate transformation by using the linear interpolation in the form of Eq. 3.42 and the coefficients defined in 3.43 (the central difference scheme only depends on geometry).

For the present research, setting $\xi(X) = X$ is used, as it is an easy modification to the existing Power Law based code. If the second method of not translating the coordinate system is used instead, significant modification of the code is required. If these modifications are made, the simulations would likely be more efficient.

4.2.1.3 Unstructured Flux Corrected Method

The Power Law scheme yields good results; however, just as the structured Power Law scheme is improved by including more terms into the interpolation, the unstructured scheme can be improved as well. A new unstructured scheme based on the unsteady, two-dimensional convection-diffusion equation with sources is developed following the concepts originally developed by Wirogo [29]. This new scheme is called the unstructured Flux Corrected Method (FCM). Starting with the two-dimensional steady convection-diffusion equation with a transformed coordinate system aligned to X, a source term is added to the right hand side of the equation (this source contains unsteady, pressure, and other source terms).

$$\frac{\partial \rho U \phi}{\partial X} - \frac{\partial}{\partial X}\left[\mu \frac{\partial \phi}{\partial X}\right] - \frac{\partial}{\partial Y}\left[\mu \frac{\partial \phi}{\partial Y}\right] = S_\phi \tag{4.37}$$

The shape function is then assumed to have a homogeneous part (ϕ_h) and a particular part (ϕ_p).

$$\begin{aligned} \phi_h &= A'\xi(X) + B'Y + C' \\ \phi_p &= \frac{S_\phi X}{\rho U_e}\ , \end{aligned} \tag{4.38}$$

with a total solution of

$$\phi = A'\xi(X) + B'Y + C' + \frac{S_\phi X}{\rho U_e} \ . \tag{4.39}$$

Substituting this shape function into the differential equation yields the modified coefficients

$$A' = L_i\phi_i - (L_iX_i)\frac{S_\phi}{\rho U_e} \tag{4.40}$$

$$B' = M_i\phi_i - (M_iX_i)\frac{S_\phi}{\rho U_e} \tag{4.41}$$

$$C' = N_i\phi_i - (N_iX_i)\frac{S_\phi}{\rho U_e} \ , \tag{4.42}$$

again using summation notation with $i = 1, 2, 3$ for the three nodes in each triangular element. The coefficients L, M, and N follow the same equations as the Power Law scheme. The shape function then becomes only a function of the nodal values and the additional FCM source term.

$$\phi = \left[L_i\xi(X) + M_iY + N_i\right]\phi_i + \left[X - (L_i\xi(X) + M_iY + N_i)X_i\right]\frac{S_\phi}{\rho U_e} \tag{4.43}$$

The flux in the X and Y directions are found by

$$J_X = f_i\phi_i + J_{X-S} \tag{4.44}$$

$$J_Y = g_i\phi_i + J_{Y-S} \ , \tag{4.45}$$

with the same functions for f_i and g_i as Power Law (Eqs. 3.106 and 3.107).

The flux that comes from the FCM source term is defined by

$$J_{X-S} = \left[\rho UX - f_iX_i - \mu\right]\frac{S_\phi}{\rho U_e} \tag{4.46}$$

$$J_{Y-S} = \left[\rho VX - g_iX_i\right]\frac{S_\phi}{\rho U_e} \ . \tag{4.47}$$

Setting the source term to zero recovers the original Power Law scheme. For the x and y momentum equations, the FCM source terms are

$$S_u = -\rho\frac{\partial u_e}{\partial t} - \left(\frac{\partial p}{\partial x}\right)_e + (S_{u-mom})_e \tag{4.48}$$

$$S_v = -\rho\frac{\partial v_e}{\partial t} - \left(\frac{\partial p}{\partial y}\right)_e + (S_{v-mom})_e \ , \tag{4.49}$$

where the subscript e represents terms for a given element and S_{u-mom} and S_{v-mom} are the sources directly from the momentum equations. It is assumed that the FCM source terms are constant over each element.

Unlike the structured FCM scheme, there are no directional or non-directional fluxes in the FCM source because the unstructured FCM scheme is a shape function to interpolate flux over both the x and y directions.

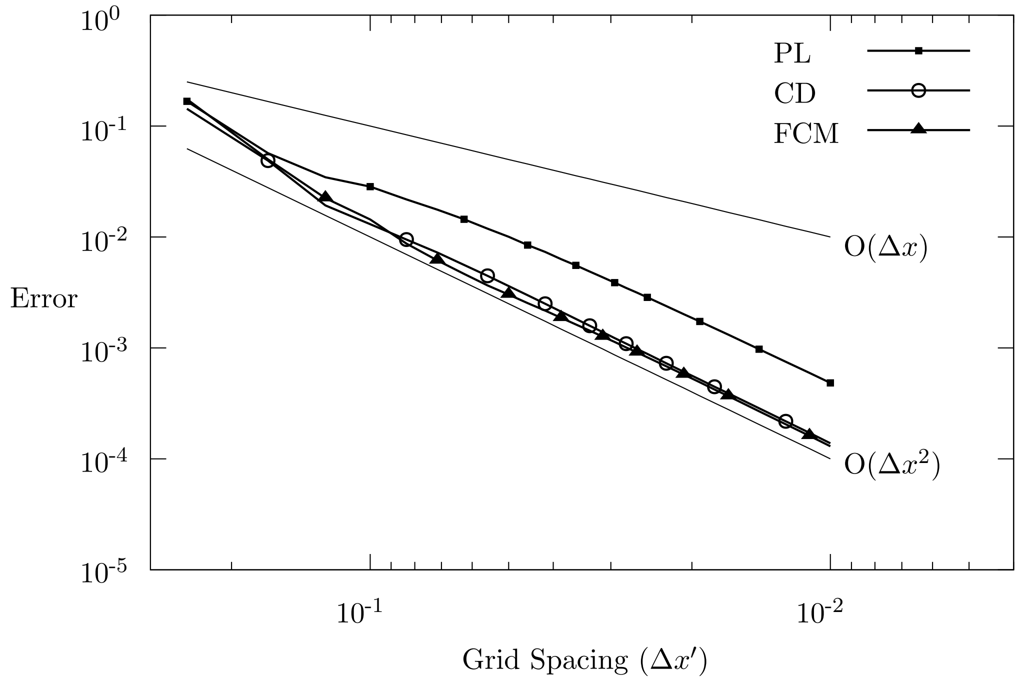
4.2.2 Results

4.2.2.1 Order of Accuracy

The lid-driven cavity case (Section 2.4.1) is used again to test the spatial order of accuracy for the three unstructured schemes. The grids used are triangulated from uniform Cartesian grids with equal grid spacing in x and y of $\Delta x' = \Delta x/L = \Delta y/L$. The grid spacing is varied, and the error is computed by comparing the velocity profile at the vertical centerline ($x/L = 0.5$) to a solution obtained by a structured solver with a central difference scheme on a 1024 by 1024 uniform Cartesian grid. The Power Law and FCM schemes are tested, as well as the central difference (CD) scheme. The results for a Reynolds number of 100 are presented in Fig. 4.8, with all three schemes resulting in second order accuracy. The central difference and FCM schemes have almost identical errors and Power Law has the largest error.

Figure 4.8 also shows the results for a Reynolds number of 400. Again, all schemes are second order accurate, although Power Law does not asymptote to second order until a grid spacing of about $\Delta x' = 1E-2$. Central difference has the lowest error, however it does not converge for grid spacing larger than $\Delta x' = 6E-2$. FCM on the other hand, converges for grid spacing up to $\Delta x' = 2.5E-1$.

The FCM error is about three times less than the Power Law error for Reynolds number of 100 at the smallest grid spacing, while at a Reynolds number of 400, the FCM error is about ten times less than Power Law.



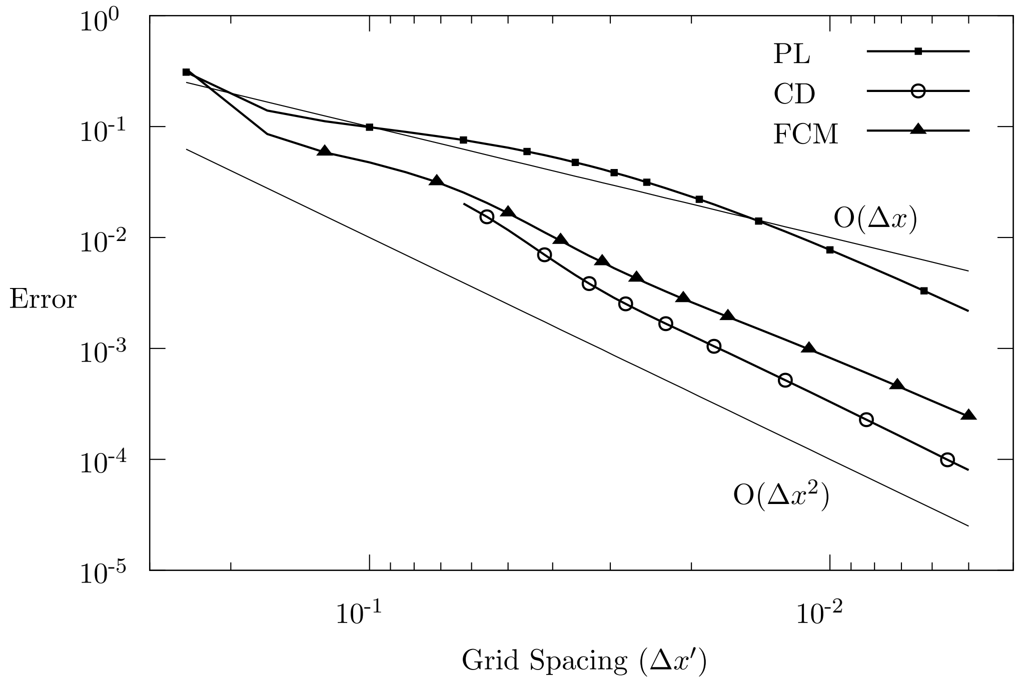


Figure 4.8: Spatial order of accuracy for the unstructured grid schemes (top: $Re = 100$, bottom: $Re = 400$).

4.2.2.2 Flow over a circular cylinder

To test FCM on a more complex case, the flow over a circular cylinder is considered. A schematic of the case is shown in Fig. 4.9. Like the flat plate, at certain Reynolds numbers (approximately above $Re = 49$ [68]) the flow becomes unsteady, with laminar vortices shed from the cylinder. A Reynolds number of 300 is used to simulate the vortex shedding in this region, although it should be noted that researchers have found the flow to be three-dimensional at Reynolds number of 300 [69],[68]. A Reynolds number of 300 is still used as a test case because of the amount of two-dimensional computational results available to compare.

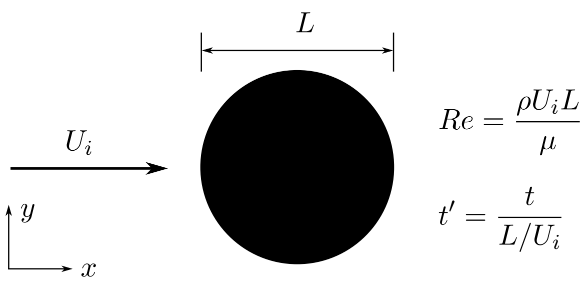


Figure 4.9: Schematic of the flow over a circular cylinder problem.

Simulations are started impulsively and run until a non-dimensional time of $t' = t/(L/U_i) = 200$. The domain is rectangular with a distance of $20L$ between the cylinder and the inlet, top, and bottom boundaries. The outlet is $50L$ away from the cylinder. The cylinder surface is represented by a grid made up of straight lines with nodes spaced either a half or a quarter degree apart, depending on the grid (see Table 4.4). The IRK-SIMPLER algorithm, as well as the C-N SIMPLER and RK-SIMPLER algorithms, are tested using FCM and PL schemes on the coarsest grid (grid 1). Table 4.2 shows the average coefficient of drag, Strouhal number, and converged runtime (RT_{cvg}) for PL and FCM.

Figure 4.10 shows the coefficient of lift and drag on the cylinder over time. After some time, vortices shed from the cylinder and an unsteady convergence is eventually reached when the oscillations become constant. The flat plate coefficient of drag varies by about 40% compared to $\overline{C_d}$ and the coefficient of lift oscillates approximately between ±0.2, while the circular cylinder coefficient of drag varies by only about 8% compared to $\overline{C_d}$ and the coefficient of lift oscillates approximately between ±0.7.

Table 4.2: Cylinder results on grid 1 at $Re = 300$[2].

	$\Delta t'$	$\overline{C_d}$			Sr			RT_{cvg}		
		IRK	C-N	RK	IRK	C-N	RK	IRK	C-N	RK
PL	$1E-2$	1.278	1.275		0.1924	0.1921		98.2	327	
	$8E-3$	1.278	1.276		0.1924	0.1921		127	423	
	$6E-3$	1.278	1.276		0.1924	0.1921		158	525	
	$8E-5$			1.277			0.1923			803
	$6E-5$			1.277			0.1923			1120
	$4E-5$			1.277			0.1923			1690
FCM	$1E-2$	1.374	1.370		0.2062	0.2058		94.2	381	
	$8E-3$	1.374	1.371		0.2061	0.2058		134	434	
	$6E-3$	1.373	1.372		0.2061	0.2058		174	472	
	$8E-5$			1.350			0.2006			875
	$6E-5$			1.350			0.2006			1170
	$4E-5$			1.350			0.2006			1750

IRK-SIMPLER and C-N SIMPLER converge for time step sizes up to $\Delta t' = 1E-2$, while RK-SIMPLER is limited to $\Delta t' = 8E-5$. For PL, all results are similar, with $\overline{C_d} = 1.28$ and $Sr = 0.192$ for all algorithms and time steps tested. For FCM, the results for IRK-SIMPLER and C-N SIMPLER, are similar with $\overline{C_d} = 1.37$ and $Sr = 0.206$ for all time steps tested, but RK-SIMPLER has slightly different values of $\overline{C_d} = 1.35$ and $Sr = 0.201$ for all time steps tested. Table 4.3 compares simulation results from other researchers for the same problem (all with $Re = 300$), as well as the most refined result for FCM (shown below). On grid 1, FCM results match quite well, but PL results do not. For both PL and FCM, IRK-SIMPLER reaches an accurate, converged solution faster than RK-SIMPLER or C-N SIMPLER with only 30% of the runtime per time step compared to C-N SIMPLER (Table 4.5).

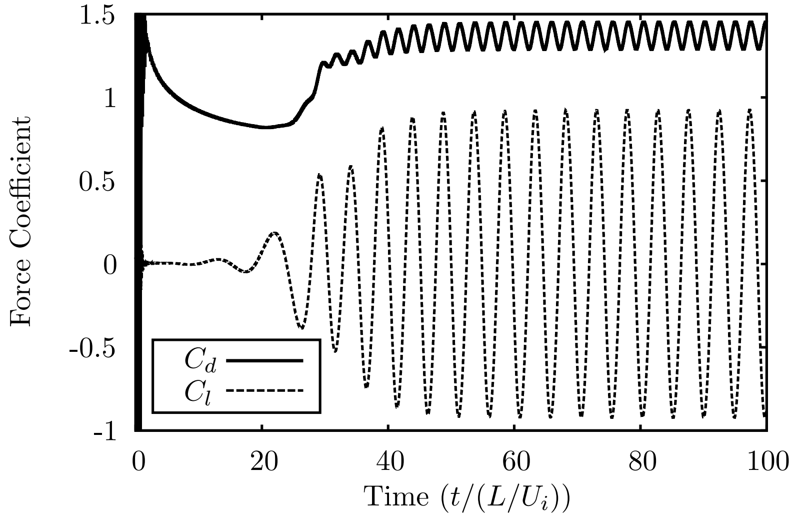


Figure 4.10: Circular cylinder coefficient of drag and lift.

Table 4.3: Simulation results for flow over a cylinder at Re=300.

Source	$\overline{C_d}$	Sr
Present	1.39	0.212
Ref [70]	1.37	0.207
Ref [68]	1.37	0.215
Ref [69]	1.38	0.213

Table 4.4: Cylinder grids tested.

Grid	# Nodes	# Elements	Body Node Spacing
1	76,497	150,137	1/2 Degree
2	106,738	210,447	1/4 Degree
3	164,336	325,639	1/4 Degree

To see how PL and FCM converge as the grid is refined, two more refined grids are tested (see Table 4.4 for grid information). Given that IRK-SIMPLER is the most accurate and efficient method, only it is tested on the refined grids. Table 4.6 shows the results for the three grids tested for PL and FCM.

As the grid is refined, FCM converges to $\overline{C_d} = 1.39$ and $Sr = 0.212$ with little difference between grid 2 and grid 3. PL approaches a similar value but only reaches $\overline{C_d} = 1.37$ and $Sr = 0.208$ for grid 3, with a relatively large difference between grid 2 and grid 3. Table 4.7 shows a summary of the IRK-SIMPLER results on the three grids, all using $\Delta t' = 6E-2$. These results show improved accuracy of FCM, with an additional cost per time step of less than 1% compared to PL. The solution for FCM on the coarsest grid 1 yields similar results to PL on the finest grid 3, but

[3] RT_{step} is runtime per step in seconds.

Table 4.5: Average runtime per time step on grid 1[3].

Algorithm	RT_{step}	
	PL	FCM
C-N SIMPLER	1.361	1.415
RK-SIMPLER	0.038	0.042
IRK-SIMPLER	0.409	0.411

FCM only takes 0.411 seconds each time step compared to PL at 1.030 seconds each time step. Comparing these solutions at similar accuracy levels, FCM achieves the results about 2.5 times faster per step.

Table 4.6: Cylinder results using IRK-SIMPLER at $Re = 300$.

Grid #	$\Delta t'$	$\overline{C_d}$		Sr		RT_{cvg}	
		PL	FCM	PL	FCM	PL	FCM
1	$1E-2$	1.278	1.374	0.1924	0.2062	98.2	94.2
	$8E-3$	1.278	1.374	0.1924	0.2061	127	134
	$6E-3$	1.278	1.373	0.1924	0.2061	158	174
2	$1E-2$	1.348	1.392	0.2042	0.2111	79.7	88.2
	$8E-3$	1.348	1.392	0.2042	0.2111	99.4	144
	$6E-3$	1.348	1.392	0.2043	0.2111	133	145
3	$1E-2$	1.369	1.395	0.2078	0.2120	137	135
	$8E-3$	1.369	1.395	0.2078	0.2120	171	157
	$6E-3$	1.369	1.394	0.2078	0.2120	231	223

Table 4.7: Summary of cylinder results on different grids for $Re = 300$.

Grid #	$\overline{C_d}$		Sr		RT_{step}		Time Difference
	PL	FCM	PL	FCM	PL	FCM	(FCM-PL)/PL
1	1.28	1.37	0.192	0.206	0.409	0.411	0.5%
2	1.35	1.39	0.204	0.211	0.652	0.678	0.4%
3	1.37	1.39	0.208	0.212	1.032	1.101	0.7%

Figures 4.11-4.16 show pressure contours for the cylinder using Power Law and FCM on all three grids. The spurious pressure oscillations are present is all cases and is worse further away

[3] RT_{cvg} in minutes.

from the cylinder where the grid is coarser. The spurious behavior improves for refined grids (2 and 3), as well as improves for FCM compared to Power Law. Vortices are present further downstream when using more refined grids and when using FCM compared to Power Law.

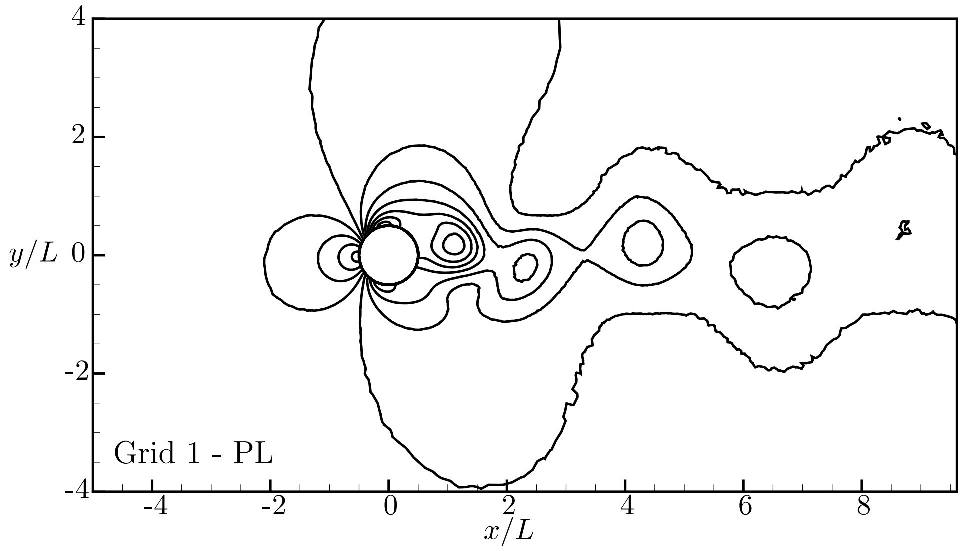


Figure 4.11: Pressure contours for grid 1 using Power Law (levels at $\Delta p/(0.5\rho U_i^2) = 0.25$).

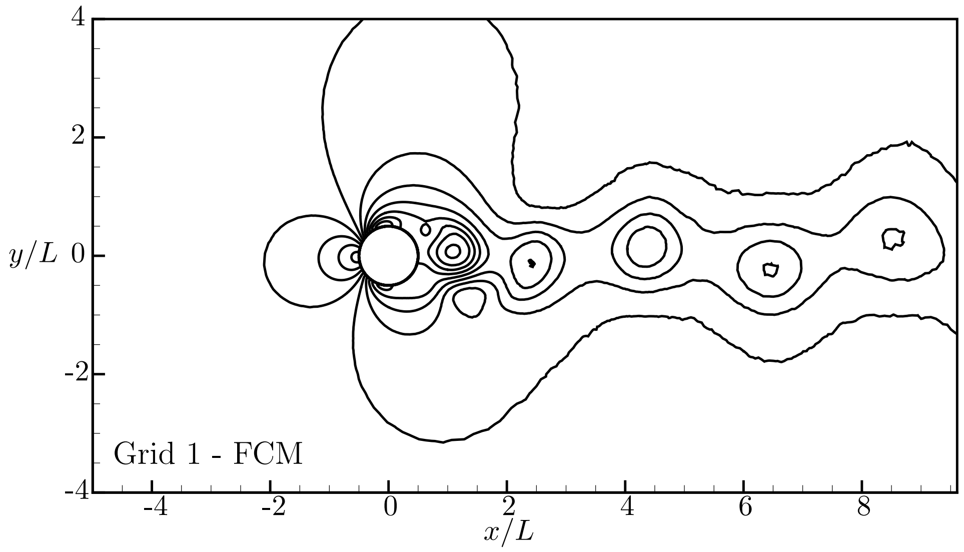


Figure 4.12: Pressure contours for grid 1 using FCM (levels at $\Delta p/(0.5\rho U_i^2) = 0.25$).

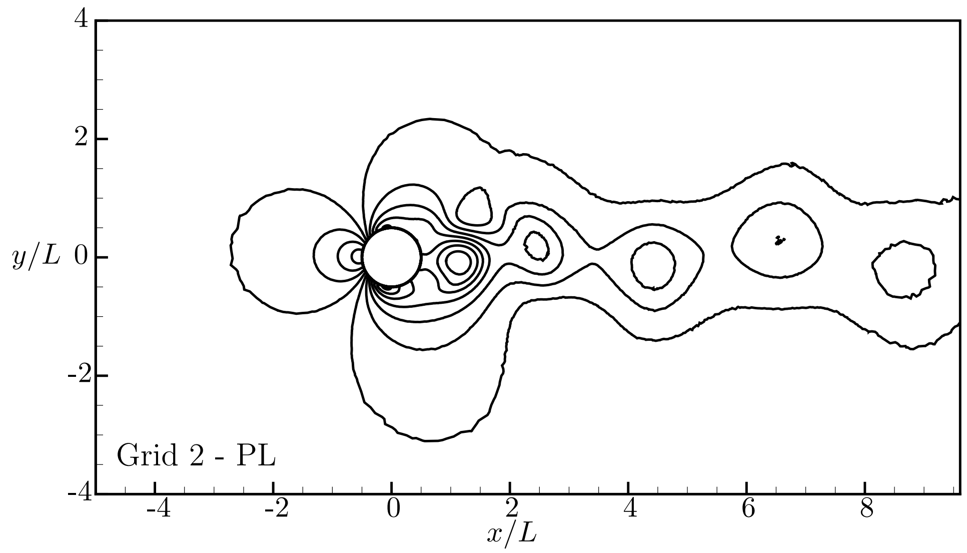


Figure 4.13: Pressure contours for grid 2 using Power Law (levels at $\Delta p/(0.5\rho U_i^2) = 0.25$).

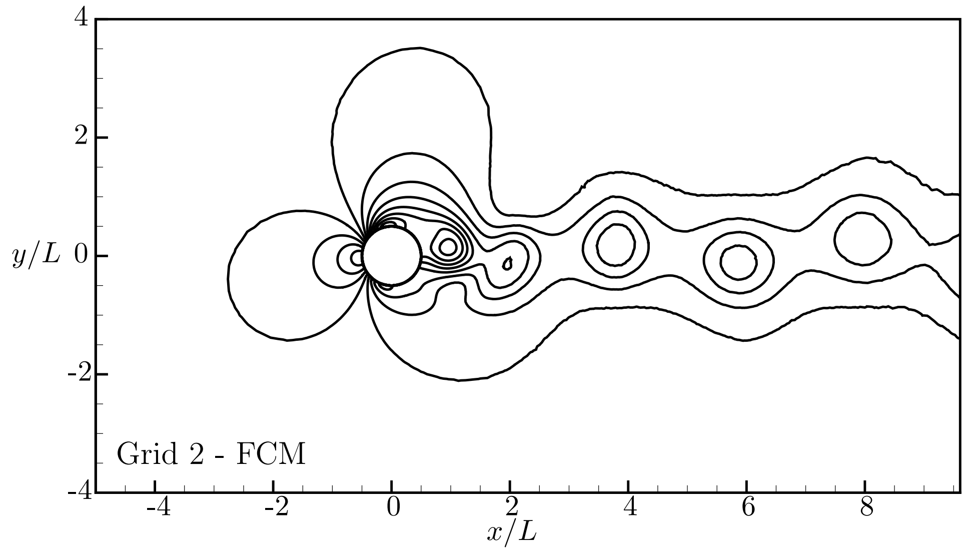


Figure 4.14: Pressure contours for grid 2 using FCM (levels at $\Delta p/(0.5\rho U_i^2) = 0.25$).

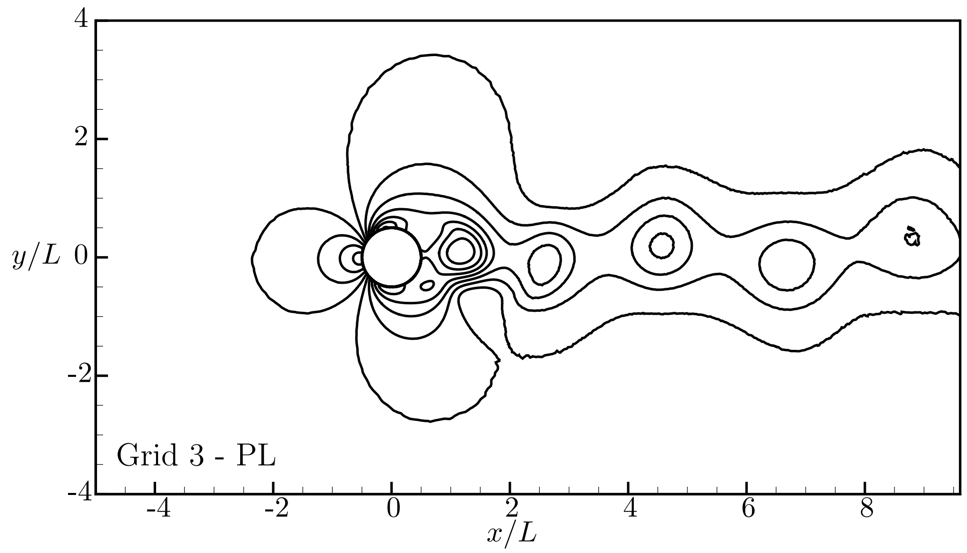


Figure 4.15: Pressure contours for grid 3 using Power Law (levels at $\Delta p/(0.5\rho U_i^2) = 0.25$).

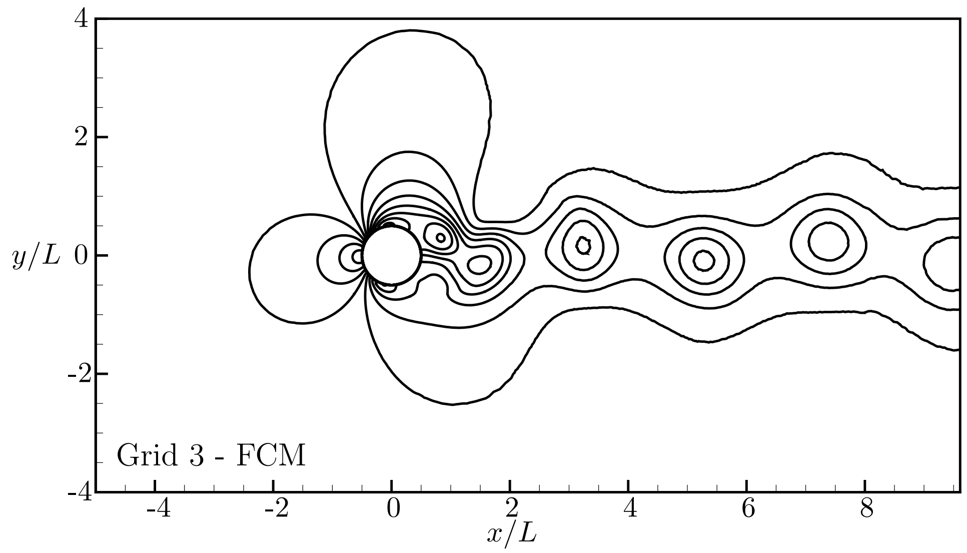


Figure 4.16: Pressure contours for grid 3 using FCM (levels at $\Delta p/(0.5\rho U_i^2) = 0.25$).

4.2.2.3 Flow over a sphere

The IRK-SIMPLER algorithm with the new unstructured FCM scheme is developed for three-dimensions using vertex-centered tetrahedral grids. The formulation of the FCM source terms is similar to the two-dimensional formulation. The laminar flow over a sphere at Reynolds numbers of 100 and 300 are simulated to compare the PL and FCM schemes. At Reynolds numbers below 200, flow is steady and axisymmetric, while at Reynolds numbers above 280, the flow sheds unsteady vortices [71]. Four grids are tested from coarse to fine, with the sphere surface grid created by mapping a uniform Cartesian cube grid to a sphere centered at the origin [72].

$$\begin{aligned} x_{sphere} &= x_{cube}\sqrt{1 - \frac{y_{cube}^2 + z_{cube}^2}{2} + \frac{y_{cube}^2 z_{cube}^2}{3}} \\ y_{sphere} &= y_{cube}\sqrt{1 - \frac{x_{cube}^2 + z_{cube}^2}{2} + \frac{x_{cube}^2 z_{cube}^2}{3}} \\ z_{sphere} &= z_{cube}\sqrt{1 - \frac{x_{cube}^2 + y_{cube}^2}{2} + \frac{x_{cube}^2 y_{cube}^2}{3}} \end{aligned} \tag{4.50}$$

This results in a surface grid on the sphere made up of planar triangles. The number of uniform cube grid spacing varies from 50 to 150 depending on the grid (see Table 4.10), resulting in 15002 to 135002 nodes on the sphere surface. For the sphere, the characteristic length L is the sphere diameter.

Table 4.8: Sphere results for $Re = 100$[4].

Grid #	C_D		RT_{cvg}	
	PL	FCM	PL	FCM
1	1.122	1.081	1.95	2.52
2	1.104	1.080	5.98	6.49
3	1.097	1.078	8.39	9.33
4	1.095	1.082	34.8	35.9

Table 4.8 shows the results for $Re = 100$ on the four grids tested, including coefficient of drag and converged runtime (RT_{cvg}, reached when the L2 norm of all residuals are less than $1E-6$). Comparison to other results in Table 4.9 shows that FCM has good agreement with previous results,

[4] RT_{cvg} in hours.

even on the coarsest grid, while PL results in a higher coefficient of drag than expected on all grids tested. The PL cases converge with less runtime on the same grid, but to achieve an accurate solution, PL requires a more refined grid and longer runtime to converge.

Table 4.9: Sphere simulation results.

	Re=100	Re=300	
Source	$\overline{C_D}$	$\overline{C_D}$	Sr
Present	1.082	0.654	0.132
Ref [71]	1.087	0.657	0.134
Ref [73]	—	0.655	0.136
Ref [74]	—	0.656	0.137
Ref [75]	1.085	—	—

Table 4.10: Sphere grids.

Grid #	# Nodes	# Elements	Body Grid
1	342,656	2,030,113	50^3
2	643,887	3,772,254	100^3
3	821,229	4,881,586	100^3
4	1,913,138	11,435,539	150^3

Table 4.11: Sphere results for $Re = 300$.

Grid #	$\Delta t'$	$\overline{C_D}$		Sr		RT_{cvg}	
		PL	FCM	PL	FCM	PL	FCM
1	$6E-1$	0.6946	0.6511	Steady	0.1195	0.850	0.772
	$4E-1$	0.6945	0.6508	Steady	0.1215	1.49	0.828
	$2E-1$	0.6941	0.6504	Steady	0.1229	2.59	1.69
	$1E-1$	0.6938	0.6500	Steady	0.1235	5.87	3.85
	$8E-2$	0.6937	0.6499	Steady	0.1236	6.28	4.76
2	$4E-1$	0.6780	diverge	Steady	—	1.28	—
	$2E-1$	0.6779	0.6519	Steady	0.1289	3.13	5.12
	$1E-1$	0.6778	0.6520	Steady	0.1303	6.37	10.4
	$8E-2$	0.6778	0.6519	Steady	0.1305	7.95	12.9
3	$2E-1$	0.6719	0.6504	0.1058	0.1308	16.0	8.72
	$1E-1$	0.6719	0.6504	0.1053	0.1316	31.1	18.1
	$8E-2$	0.6719	0.6504	0.1054	0.1317	39.5	22.0
4	$1E-1$	0.6672	0.6536	0.1259	0.1323	70.6	51.6
	$8E-2$	0.6673	0.6536	0.1261	0.1324	88.6	65.2

Table 4.11 shows the results for $Re = 300$ on the four grids tested, including average coefficient of drag, Strouhal number and converged runtime (RT_{cvg} in hours). On grids 1 and 2, PL is unable to capture the unsteady vortex shedding and results in a steady solution. For all grids, FCM

captures the vortex shedding well and has good agreement with other results in Table 4.9. For the finer grids, PL captures some of the unsteady vortex shedding, but both average drag coefficient and Strouhal number differ significantly from results in Table 4.9.

Table 4.12 shows the results on all grids for the lowest time step size tested ($\Delta t' = 8E - 2$) along with average runtime per time step for each grid. For all four grids, the additional time required each time step for FCM is less than 5%. This benefit, alongside the improved accuracy, allows FCM to converge to accurate solutions much faster than PL. For the grids tested, PL has yet to converge and requires further refinement and additional runtime to yield accurate results.

Table 4.12: $Re = 300$ Sphere results summary using $\Delta t' = 8E - 2$[5].

Grid #	$\overline{C_D}$		Sr		RT_{step}		Time Difference
	PL	FCM	PL	FCM	PL	FCM	(FCM-PL)/PL
1	0.694	0.650	Steady	0.124	0.509	0.533	4.7%
2	0.678	0.652	Steady	0.131	0.762	0.783	2.8%
3	0.672	0.650	0.105	0.132	1.262	1.322	4.8%
4	0.667	0.654	0.126	0.132	3.099	3.120	3.4%

Figures 4.17 and 4.18 show the coefficient of drag versus time for the grids tested using Power Law and FCM, respectively. Because the FCM lines are close together, a zoomed region is shown in Fig. 4.19. The PL solution on grid 3, although resulting in a nonzero Strouhal number, has very low magnitude oscillations that are not visible in Fig. 4.17. On grid 4, PL has more pronounced oscillations, but not as large as FCM.

[5] RT_{step} in minutes.

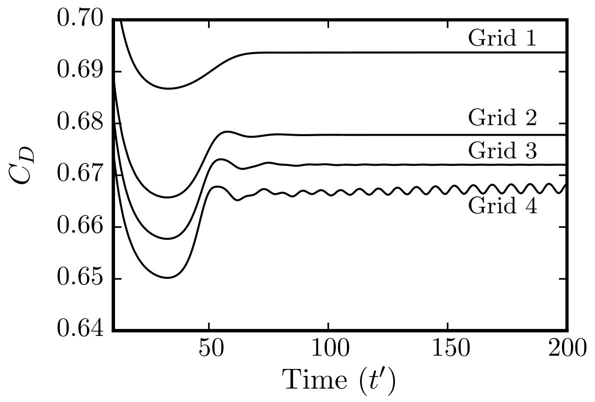


Figure 4.17: Sphere coefficient of drag using Power Law at $Re = 300$.

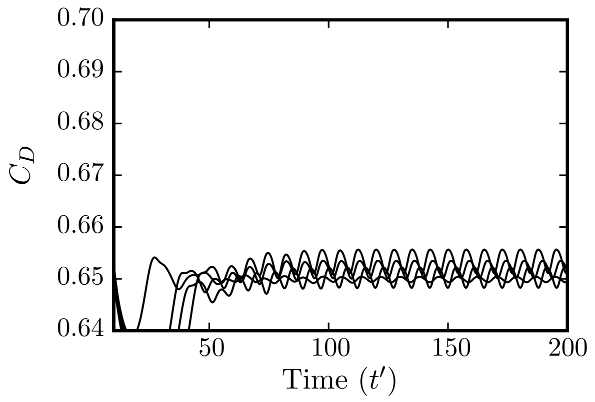


Figure 4.18: Sphere coefficient of drag using FCM at $Re = 300$.

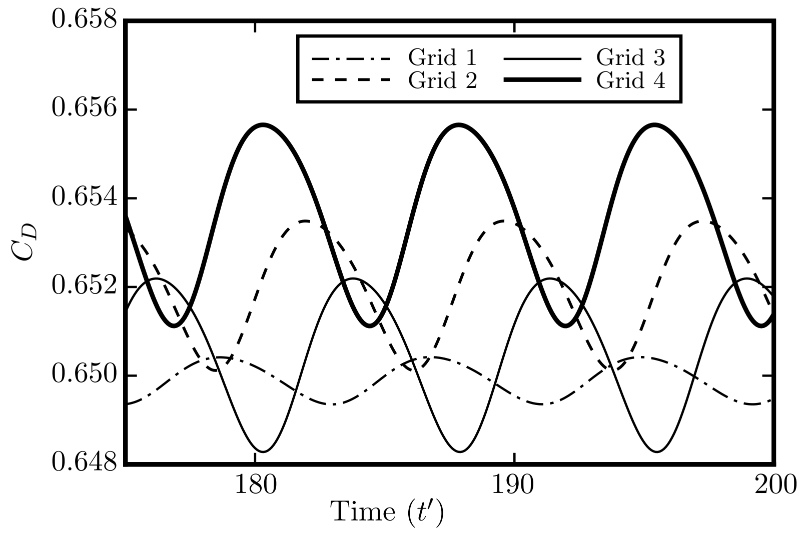


Figure 4.19: Sphere coefficient of drag using FCM at $Re = 300$, zoomed in.

4.2.3 Conclusions

The unstructured FCM scheme is developed and shown to be second order accurate with lower error than the Power Law scheme. FCM also shows some improvement in reducing the pressure oscillations seen in the unsteady cases. FCM requires less than 1% and 5% more runtime per time step than Power Law for two- and three-dimensional cases respectively. With the improved accuracy and modest increase in runtime over Power Law, FCM yields accurate results, with up to 19 times speedup over Power Law.

CHAPTER 5. TURBULENCE MODELING

Turbulence is an important factor in wind turbine and wind farm performance. Although the Navier-Stokes equations include all the physics that govern fluid flow, fully resolving the smallest scales of turbulence in space and time requires very high resolution Direct Numerical Simulation (DNS). To avoid fully resolving all the scales of turbulence, turbulence modeling is used.

For present research, RANS modeling is used to handle turbulence without increasing the computational cost dramatically. The model chosen is the $k - \epsilon$ model with both the standard and realizable versions.

5.1 Governing Equations

Index notation is commonly used for turbulence equations, and the following terms are defined.

$$u_1 = u \qquad u_2 = v \qquad u_3 = w \qquad x_1 = x \qquad x_2 = y \qquad x_3 = z \tag{5.1}$$

When using index notation, a repeated index represents a summation. For example, the divergence of a vector is represented as

$$\frac{\partial u_j}{\partial x_j} = \frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2} + \frac{\partial u_3}{\partial x_3} \ . \tag{5.2}$$

The Navier-Stokes equations written in index form are

$$\frac{\partial}{\partial t}(\rho u_i) + \frac{\partial}{\partial x_j}(\rho u_j u_i) = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left[\mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\right] + b_i \tag{5.3}$$

$$\frac{\partial u_j}{\partial x_j} = 0 \ , \tag{5.4}$$

where b_i is the momentum source.

The RANS equations are formed by assuming the flow has mean and turbulent fluctuating parts

$$u_i = U_i + u_i' \qquad\qquad p = P + p' \ , \tag{5.5}$$

where the turbulent fluctuations have a zero mean, resulting in the following time averaged quantities (where an upper bar represents time averaging).

$$\overline{U_i} = U_i \qquad \overline{u'_i} = 0 \qquad \overline{u_i} = U_i \tag{5.6}$$

$$\overline{P} = P \qquad \overline{p'} = 0 \qquad \overline{p} = P \tag{5.7}$$

Substituting the terms in Eq. 5.5 into Eqs. 5.3 and 5.4

$$\frac{\partial}{\partial t}\Big[\rho(U_i+u'_i)\Big] + \frac{\partial}{\partial x_j}\Big[\rho(U_j+u'_j)(U_i+u'_i)\Big] = -\frac{\partial}{\partial x_i}(P+p') + \frac{\partial}{\partial x_j}\Big[\mu\frac{\partial}{\partial x_j}(U_i+u'_i)\Big] + S_i \tag{5.8}$$

$$\frac{\partial}{\partial x_j}(U_j+u'_j) = 0\ . \tag{5.9}$$

Averaging Eqs. 5.8 and 5.9 over time results in the RANS equations

$$\frac{\partial}{\partial t}(\rho U_i) + \frac{\partial}{\partial x_j}(\rho U_j U_i) = -\frac{\partial P}{\partial x_i} + \frac{\partial}{\partial x_j}\left[\mu\left(\frac{\partial U_i}{\partial x_j}+\frac{\partial U_j}{\partial x_i}\right)\right] + \frac{\partial}{\partial x_j}(-\rho\overline{u'_j u'_i}) + b_i \tag{5.10}$$

$$\frac{\partial U_j}{\partial x_j} = 0\ , \tag{5.11}$$

where the only term that differs from Eqs. 5.3 and 5.4 is $-\rho\overline{u'_j u'_i}$, which is called the Reynolds stress. The Reynolds stress is a tensor, and some RANS models, known as Reynolds Stress Transport (RST) models, develop six equations to model each component of the symmetric Reynolds stress tensor. The one- and two-equation models develop a scalar model that relates the Reynolds stress to terms like mixing length, eddy viscosity, and turbulent kinetic energy.

The model investigated in the present research is the $k-\epsilon$ model, one of the most common methods used for engineering applications. In future equations, the fluctuating terms are denoted as u'_i, but the mean terms are denoted u_i.

5.1.1 $k-\epsilon$ Models

The $k-\epsilon$ models use a linear stress-strain relationship to find the Reynolds stress, and assume equilibrium between Reynolds stress and the mean rate of strain.

$$-\rho\overline{u'_j u'_i} = 2\mu_T S_{i,j} - \frac{2}{3}\rho k\delta_{i,j}\ , \tag{5.12}$$

where μ_T is the turbulent eddy viscosity, $S_{i,j}$ is the rate of strain remote, k is the turbulence kinetic energy, and $\delta_{i,j}$ is the Kronecker delta function.

$$\mu_T = C_\mu \rho \frac{k^2}{\epsilon} \qquad S_{i,j} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{5.13}$$

$$k = \frac{1}{2}\overline{u_i' u_i'} \qquad \delta_{i,j} = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases} \tag{5.14}$$

where ϵ is the turbulent dissipation and C_μ is a model constant (or a known variable in some models). At this point the RANS momentum equation is written with the new relationship for Reynolds stress.

$$\frac{\partial}{\partial t}(\rho u_i) + \frac{\partial}{\partial x_j}(\rho u_j u_i) = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left[(\mu + \mu_T)\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\right] - \frac{2}{3}\frac{\partial}{\partial x_i}(\rho k) + b_i \tag{5.15}$$

$$= -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left[2(\mu + \mu_T)S_{i,j}\right] - \frac{2}{3}\frac{\partial}{\partial x_i}(\rho k) + b_i \tag{5.16}$$

The $k - \epsilon$ model has two unknowns, k and ϵ, that must be solved with two equations (hence it is called a two-equation model). The two equations are both transport equations with unsteady, convection, diffusion, production, and dissipation terms.

5.1.1.1 Standard $k - \epsilon$

The standard $k-\epsilon$ model was first developed by Jones and Launder [76] with empirical constants given in Launder and Sharma [34]. The equations for k and ϵ are

$$\frac{\partial}{\partial t}(\rho k) + \frac{\partial}{\partial x_j}(\rho u_j k) = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_T}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] + P_k - \rho\epsilon \tag{5.17}$$

$$\frac{\partial}{\partial t}(\rho \epsilon) + \frac{\partial}{\partial x_j}(\rho u_j \epsilon) = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_T}{\sigma_\epsilon}\right)\frac{\partial \epsilon}{\partial x_j}\right] + C_{1\epsilon}\frac{\epsilon}{k}P_k - C_{2\epsilon}\rho\frac{\epsilon^2}{k} \ , \tag{5.18}$$

where P_k is the production of turbulent kinetic energy and σ_k, σ_ϵ, $C_{1\epsilon}$, and $C_{2\epsilon}$ are empirical constants set to match experimental data (values given in Table 5.1). The production term is found by

$$P_k = \mu_T S^2 \ , \tag{5.19}$$

where S is

$$S = \sqrt{\frac{1}{2} S_{i,j} S_{i,j}} \ . \tag{5.20}$$

This production term can be modified using the Kato-Launder modification [77], which can improve the accuracy in flows with high acceleration regions. The Kato-Launder modified production term is

$$P_k = \mu_T S \Omega \ , \tag{5.21}$$

where Ω is

$$\Omega = \sqrt{\frac{1}{2} \Omega_{i,j} \Omega_{i,j}} \tag{5.22}$$

and $\Omega_{i,j}$ is the rate of rotation

$$\Omega_{i,j} = \frac{1}{2} \left(\frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial x_i} \right) \ . \tag{5.23}$$

The standard production term (without the Kato-Launder modification) is used, unless otherwise noted.

Table 5.1: Model constants used for standard $k - \epsilon$.

Constant	Value
σ_k	1.0
σ_ϵ	1.3
$C_{1\epsilon}$	1.44
$C_{2\epsilon}$	1.92
C_μ	0.09

5.1.1.2 Realizable $k - \epsilon$

The realizable $k - \epsilon$ model [78] was developed to improve upon the standard $k - \epsilon$ model. There are two differences between the models. The realizable $k - \epsilon$ model treats C_μ as a variable and uses different production and dissipation terms in the ϵ equation. The k equation is identical to the standard model (Eq. 5.17), and the ϵ equation is

$$\frac{\partial}{\partial t}(\rho \epsilon) + \frac{\partial}{\partial x_j}(\rho u_j \epsilon) = \frac{\partial}{\partial x_j} \left[\left(\mu + \frac{\mu_T}{\sigma_\epsilon} \right) \frac{\partial \epsilon}{\partial x_j} \right] + \rho C_1 S \epsilon - \rho C_2 \frac{\epsilon^2}{k + \sqrt{\frac{\mu \epsilon}{\rho}}} \ . \tag{5.24}$$

The following definitions are used in the realizable $k-\epsilon$ model

$$C_1 = \max\left[0.43, \frac{\eta}{\eta+0.5}\right] \qquad \eta = S\frac{k}{\epsilon} \tag{5.25}$$

$$C_\mu = \frac{1}{A_o + A_s\frac{ku^*}{\epsilon}} \qquad u^* = \sqrt{S_{i,j}S_{i,j} + \Omega_{i,j}\Omega_{i,j}} \tag{5.26}$$

$$A_s = \sqrt{6}\cos\phi \qquad \phi = \frac{1}{3}\cos^{-1}\left[\sqrt{6}W\right] \tag{5.27}$$

$$W = \frac{S_{i,j}S_{j,k}S_{k,i}}{\tilde{S}^3} \qquad \tilde{S} = \sqrt{S_{i,j}S_{i,j}}\ . \tag{5.28}$$

The realizable $k-\epsilon$ model constants are given in Table 5.2.

Table 5.2: Model constants used for realizable $k-\epsilon$.

Constant	Value
σ_k	1.0
σ_ϵ	1.2
A_o	4.04
C_2	1.90

5.1.2 Boundary Conditions

At non-wall boundaries, boundary conditions are either given as known values of k and ϵ, or an known turbulence intensity, I, and turbulence length scale, l. If I and l are known, k and ϵ are set from

$$k = \frac{3}{2}(U_i I)^2 \qquad \epsilon = C_\mu \frac{k^{3/2}}{l}\ , \tag{5.29}$$

using $C_\mu = 0.09$ and U_i is the free-stream mean velocity. These equations are also used to initialize the domain when other values are not known.

On walls, $k = 0$ due to the no-slip condition, but a boundary condition on ϵ is not immediately apparent. Instead of finding a boundary condition for ϵ and requiring several grid points near the wall to capture the thin boundary layer, wall functions are used.

5.1.2.1 Wall Functions

For a fully developed turbulent boundary layer, three regions exist near the wall [36].

- Law of the Wall (Viscous Sub-Layer): $y_+ < 5$

  In this region molecular viscous stress is dominant and turbulent stress is small.

$$u_+ = y_+ \tag{5.30}$$

- Buffer Layer : $5 < y_+ < 40$

  This layer is between the viscous and inertial sub-layers and blends the velocity profile between the two.

- Log Layer (Inertial Sub-Layer): $40 < y_+ < 200$

$$u_+ = \frac{1}{\kappa}\ln(Ey_+) \tag{5.31}$$

Outside the log layer is the law of the wake region that is not universal and is problem dependent. The non-dimensional terms are defined as

$$y_+ = \frac{\rho u_* y}{\mu} \qquad\qquad u_+ = \frac{u}{u_*} \ . \tag{5.32}$$

The friction velocity, u_*, is defined by

$$u_* = \sqrt{\tau_w/\rho} \ , \tag{5.33}$$

where τ_w is the shear stress due to skin friction at the wall. The constants for the wall layers are

$$\kappa = 0.42 \qquad\qquad E = 9.81 \ . \tag{5.34}$$

Wall functions use the fact that for turbulent boundary layers, below about $y_+ = 200$, the flow is the same for all cases. Therefore, wall functions requires the first grid point to fall within the inertial sub-layer, $y_+ \leq 200$, and also, the wall functions should have the first grid point above the viscous sub-layer, $y_+ \geq 40$. Because this is not guaranteed at all points, the wall functions are used outside of these bounds, but may not result in accurate solutions.

The wall stress is calculated by combining Eqs. 5.33, 5.31, and 5.32 into

$$\tau_w = \rho u_*^2 = \frac{\rho\kappa u_p u_*}{\ln(Ey_+)} \ , \tag{5.35}$$

where u_p is the velocity at the first grid point above the wall in the inertial sub-layer. In practice, the wall stress is calculated using the above equation, and the effective viscosity next to the wall is calculated from

$$\mu_w = \tau_w \frac{y_p}{u_p} = \frac{\rho \kappa y_p u_*}{\ln(E y_+)} \ . \tag{5.36}$$

The equations for wall stress and effective wall viscosity are functions of u_*, which are not known. Launder and Spalding [79] developed the standard wall function by setting

$$u_* = C_\mu^{1/4} k_p^{1/2} \ . \tag{5.37}$$

The wall stress is calculated from the given flow variables at the point p above the wall.

$$\tau_w = \frac{\rho \kappa u_p C_\mu^{1/4} k_p^{1/2}}{\ln\left[E(y_+)_p\right]} \ , \tag{5.38}$$

where

$$(y_+)_p = \frac{\rho C_\mu^{1/4} k_p^{1/2} y_p}{\mu} \ , \tag{5.39}$$

with μ being the fluid viscosity. If the point y_p is lower than $(y_+)_e = 11.225$, the relationship for the viscous sub-layer is used. This results in a wall stress of

$$\tau_w = \mu \frac{u_p}{y_p} \ . \tag{5.40}$$

Whether the point is in the viscous or inertial sub-layer, the effective viscosity is calculated using

$$\mu_w = \tau_w \frac{y_p}{u_p} \ . \tag{5.41}$$

The production term, P_k and dissipation at the first point above the wall are set by assuming local equilibrium [80].

$$P_k = \frac{\tau_w^2}{\rho \kappa y_p C_\mu^{1/4} k_p^{1/2}} \tag{5.42}$$

$$\epsilon_p = \frac{C_\mu^{3/4} k^{3/2}}{\kappa y_p} \tag{5.43}$$

For rough walls with roughness height K_s, the effective viscosity in the inertial region is

$$\tau_w = \frac{\rho \kappa u_p C_\mu^{1/4} k_p^{1/2}}{\ln\left[E(y_+)_p\right] - \Delta B} \tag{5.44}$$

where

$$\Delta B = \ln\left[1 + 0.3(K_s)_+\right] \qquad\qquad (K_s)_+ = K_s \frac{\rho u_*}{\mu} \ . \tag{5.45}$$

5.1.3 Implementation Details for Unstructured Grids

The $k-\epsilon$ model has been implemented and tested on structured grids by Sayan [80] and Murali [44]. The present research is interested in testing the $k-\epsilon$ model on the unstructured vertex-based grids defined in Chapter 3. The k and ϵ terms are stored at the vertices of the elements, and the turbulent viscosity, μ_T, is stored at the element center and is assumed constant over each element. The convective and diffusive fluxes for k and ϵ equations are computed using the same Power Law and FCM methods as discussed in Chapters 3 and 4.

When using the FCM scheme, the source term for the k equation is

$$S_k = -\rho \frac{\partial k_e}{\partial t} + (P_k)_e - \rho \epsilon_e \ , \tag{5.46}$$

where k_e and ϵ_e are quantities averaged to the element center, and $(P_k)_e$ is the production term calculated at the element center. The FCM source terms for the ϵ equation are

$$S_\epsilon^{std} = -\rho \frac{\partial \epsilon_e}{\partial t} + C_{1\epsilon} \frac{\epsilon_e}{k_e}(P_k)_e - C_{2\epsilon}\rho \frac{\epsilon_e^2}{k_e} \tag{5.47}$$

$$S_\epsilon^{rlz} = -\rho \frac{\partial \epsilon_e}{\partial t} + \rho C_1 S \epsilon_e - \rho C_2 \frac{\epsilon_e^2}{k_e + \sqrt{\frac{\mu_e \epsilon_e}{\rho}}} \ , \tag{5.48}$$

for standard and realizable $k-\epsilon$ models respectively, and again all terms are evaluated at the element center.

Using the FCM scheme for the momentum equations involves additional source terms from the turbulence terms

$$S_u = -\rho \frac{\partial u_e}{\partial t} - \left(\frac{\partial p}{\partial x}\right)_e + (S_{u-mom})_e + \frac{\partial}{\partial x}\left[(\mu + \mu_T)\frac{\partial u}{\partial x} - \frac{2}{3}\rho k - p\right] + \frac{\partial}{\partial y}\left[(\mu + \mu_T)\frac{\partial v}{\partial x}\right] \tag{5.49}$$

$$S_v = -\rho \frac{\partial v_e}{\partial t} - \left(\frac{\partial p}{\partial y}\right)_e + (S_{v-mom})_e + \frac{\partial}{\partial x}\left[(\mu + \mu_T)\frac{\partial u}{\partial y}\right] + \frac{\partial}{\partial y}\left[(\mu + \mu_T)\frac{\partial v}{\partial y} - \frac{2}{3}\rho k - p\right] \ . \tag{5.50}$$

Figure 5.1 shows the layout of an unstructured vertex-centered grid at a wall. For elements with an edge on a wall, such as element e_1, the wall stress, $(\tau_w)_1$, is calculated using $(y_p)_1$, $(u_p)_1$

(which is parallel to the boundary edge), and k_p from node n_1. These elements use the same values to calculate effective viscosity, $\mu_{eff} = \mu + \mu_T$. For elements that contain wall nodes but do not have an edge on the wall, the effective viscosity is calculated by averaging the neighboring elements with wall edges as

$$\mu_{eff} = \frac{1}{\sum_{n=1}^{N} \frac{1}{N(\mu_{eff})_n}} \ , \tag{5.51}$$

where N is the total number of neighbor elements with wall edges. For element e_3 in Fig. 5.1,

$$(\mu_{eff})_3 = \frac{1}{\frac{1}{2(\mu_{eff})_1} + \frac{1}{2(\mu_{eff})_2}} \ . \tag{5.52}$$

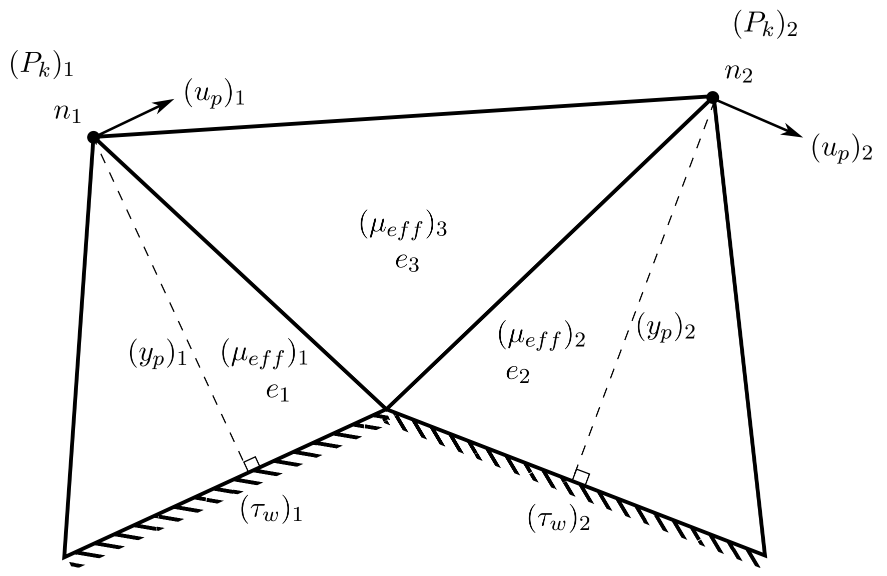


Figure 5.1: Boundary conditions and wall functions for triangle elements.

For nodes adjacent to the wall, like n_1 and n_2 in Fig. 5.1, the k production term, P_k, is computed using the wall stress at the nearest wall edge and the quantities at the node (including y_p to the nearest wall edge). The production term is integrated using the node control volume, not the element volume. Also at these nodes, the value of ϵ is set by Eq. 5.43.

The realizable $k - \epsilon$ model is stable and converges for the cases tested, but the standard $k - \epsilon$ model tends to diverge. To improve the stability, limits on the minimum and maximum of k, ϵ

and μ_T are set, and allows the standard model to converge for more cases. A downside to this is the limiting values are not known before starting the simulation. In practice, limits are found by running the realizable model to find the maximum values in the converged solution and limits were set slightly above the maximum values found. Minimum limits are set on a trial and error basis. Another stability improvement is a variable C_μ formulation developed by Rodi [81] as

$$C_\mu = \min\left[0.09, \frac{0.10738(0.64286 + 0.19607R)}{[1 + 0.357(R-1)]^2}\right] , \tag{5.53}$$

where R is the ratio of production to dissipation

$$R = \frac{P_k}{\rho\epsilon} . \tag{5.54}$$

This modification is used in all simulations where the standard $k-\epsilon$ model is used.

The source terms for k and ϵ are calculated for each element based on the centroid values and are integrated over the elemental volume. The source term for each element is split evenly among all nodes contained in the elements. The k and ϵ equations are integrated in time using either fully implicit or Crank-Nicolson schemes, with iterations and relaxation (typically 10 and 0.5-0.9) to couple the k and ϵ equations. The process to solve the $k-\epsilon$ equations is

1. Solve the k equation with relaxation
2. Solve the ϵ equation with relaxation
3. Repeat steps 1 and 2 until convergence (or n iterations)
4. Calculate μ_T

Calculation of of turbulent quantities are done once per time step, after the momentum and continuity equations are solved for the time step.

5.2 Results

5.2.1 Backwards Facing Step

The first test case of turbulent simulation on unstructured grids is the flow over a backwards facing step. Figure 5.2 shows the case setup, in which flow through a upper channel reaches a sudden expansion over a backwards facing step. The flow separates from the wall, and a recirculation region exists for some distance x_r. The ratio of the lower channel height (H_L) to the step height (H) is $R = H_L/H$. Driver [82] conducted an experiment with $R = 12$, while Yoder [83] presented simulation results using both $k - \epsilon$ and SST RANS models using $R = 9$ and compared to the experiments of Driver. Present simulations use $R = 9$ and compare to both Driver and Yoder data. The Reynolds number for this case is $Re = \rho U_i H/\mu = 36,000$, where U_i is the uniform inlet velocity.

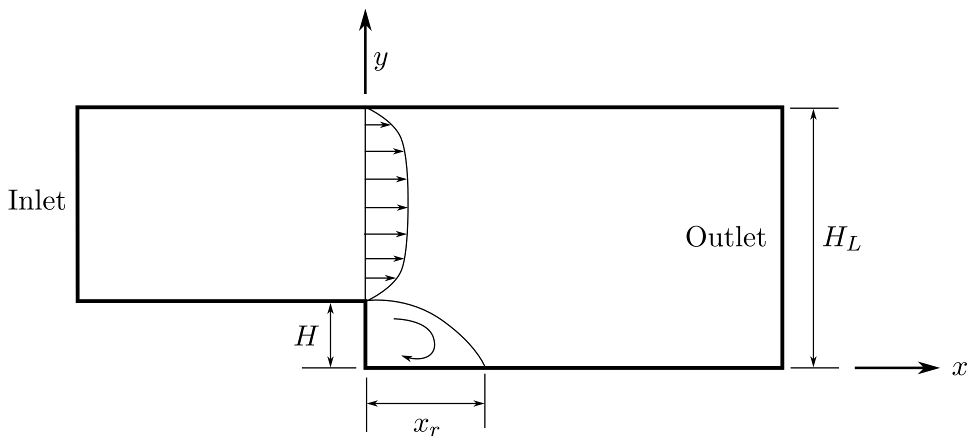


Figure 5.2: Backward facing step case.

Two cases are tested, both run with IRK-SIMPLER until a steady state is reached. The first case simulates flow through the upper channel (starting with uniform inflow at the inlet), along with the step and lower channel. The second case sets an inlet profile at $x = 0$ that matches the Driver data at the step, and only simulates the flow in the lower channel ($x > 0$).

For present testing, the upper channel has a height of $8H$ and a length of $100H$ to allow the channel flow to fully develop before reaching the step. The lower channel has a height of $9H$ and a length of $50H$.

The first set of grids are created by inputting the boundaries of the domain into Triangle grid generator [84], which generates a triangular grid using Delaunay triangulation. An example section of this type of grid is shown in Fig. 5.3. Using this type of grid results in a skin friction coefficient, C_f, distribution on the wall that is not smooth, as seen in Fig. 5.5. The next set of grids are created by including a wall layer by adding lines parallel to the wall surfaces offset by a small distance and again using the Triangle grid generator. An example section of this improved grid is shown in Fig. 5.4. The second type of grid, with a wall layer, results in a smooth skin friction coefficient on the wall (Fig. 5.5) and allows for some control of the height of the first grid point, which is important for wall functions.

The grid used for the results presented has a wall layer with an offset of $0.00866H$, which resulted in $9 < y_+ < 14$ for the standard $k-\epsilon$ model (SKE) and $4 < y_+ < 16$ for the realizable $k-\epsilon$ model (RKE).

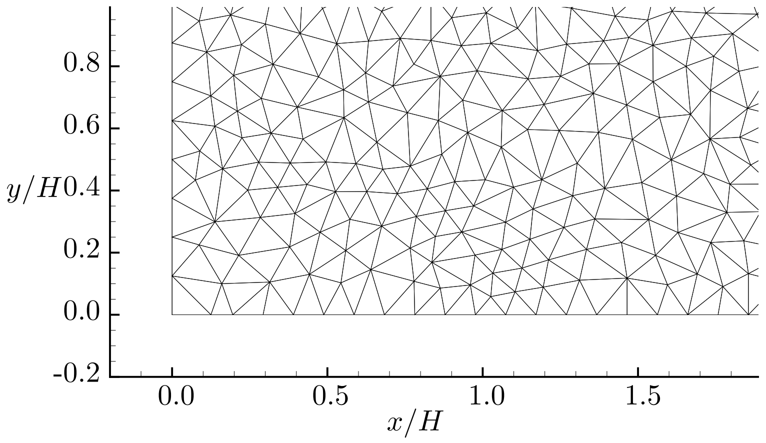


Figure 5.3: Section of grid for the backward facing near the step wall *without* a wall layer.

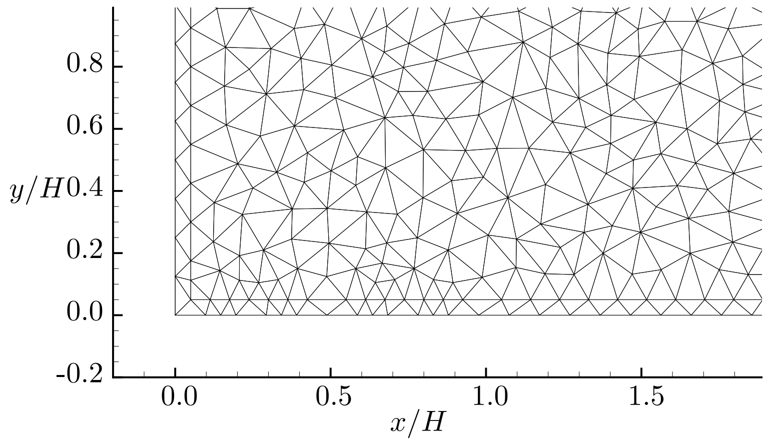
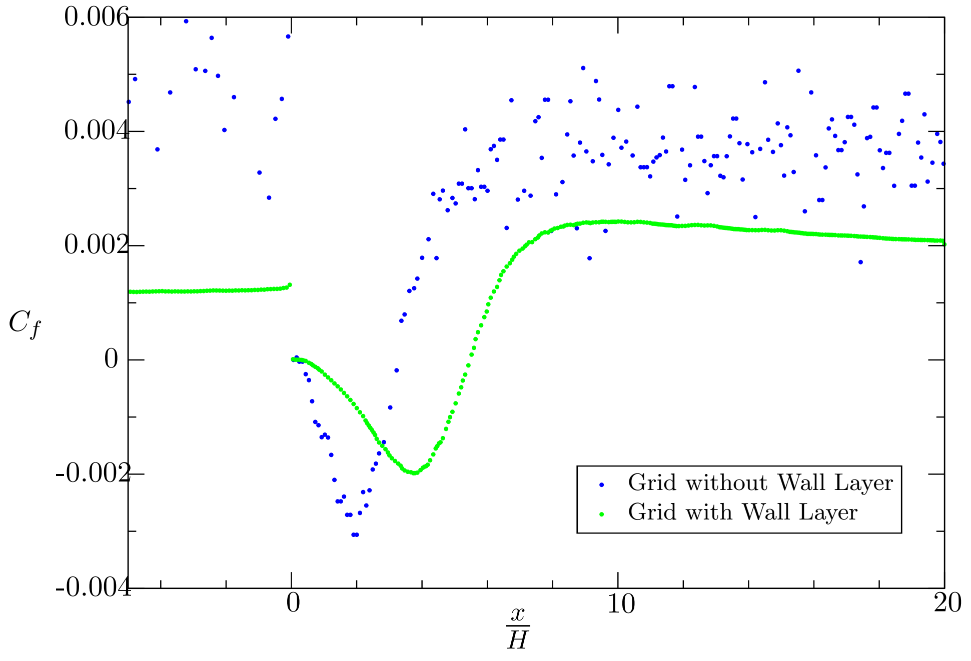


Figure 5.4: Section of grid for the backward facing near the step wall *with* a wall layer



.

Figure 5.5: Backward facing step skin friction coefficient along step wall.

5.2.1.1 Simulation Including Upper Channel

For these simulations, the flow is set to uniform flow on the inlet of the upper channel (with zero turbulence intensity) and the outlet of the lower channel an outflow corrected for mass conservation. All other boundaries are no-slip walls. The domain is initialized to uniform velocity, pressure, and turbulent quantities (free-stream values). The turbulence intensity and length scale are set to $I = 0.10$ and $l = 0.025$, to improve stability for the initial time steps.

Figures 5.6 and 5.7 show the profiles of u and k, respectively, at five locations ($x/H = 0, 4, 8, 12$, and 16) after the solution has reached steady state. The results are compared to the experimental results from Driver [82]. The RKE model matches experiments well for u in the recirculation region near the bottom of the lower wall, but not further away from the wall. The profile at the step ($x/H = 0$) does not match the experimental trend for RKE. The SKE u profiles are not as close in the recirculating region, but match the trends better above $y/H = 1$. For k profiles, the SKE model matches the first two locations better, but RKE matches the further downstream locations better.

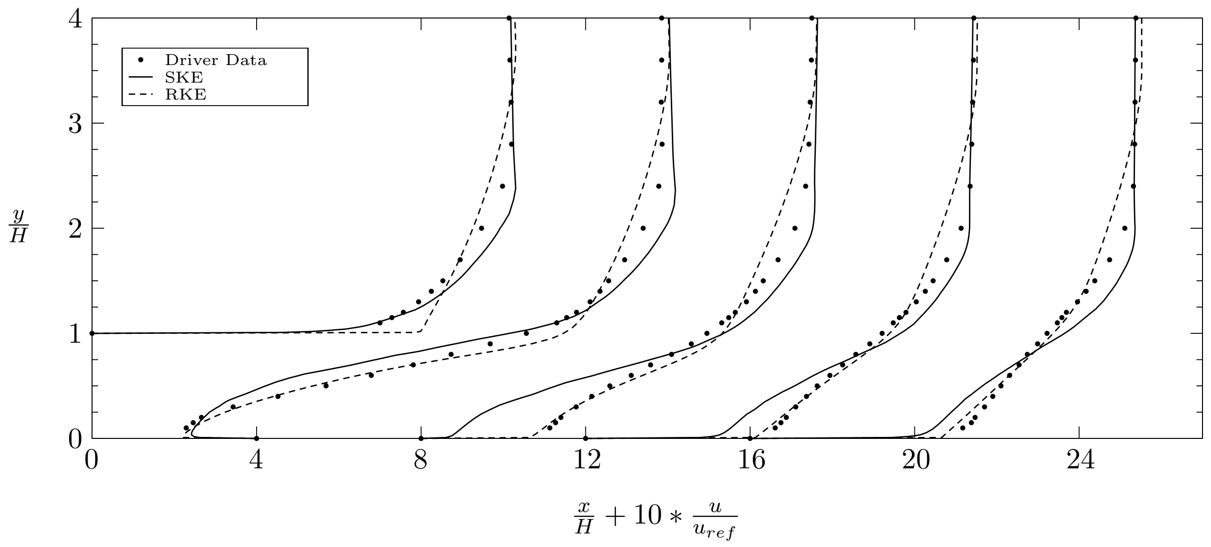


Figure 5.6: Backward facing step u velocity profiles at $x/H = 0, 4, 8, 12, 16$ with the upper channel.

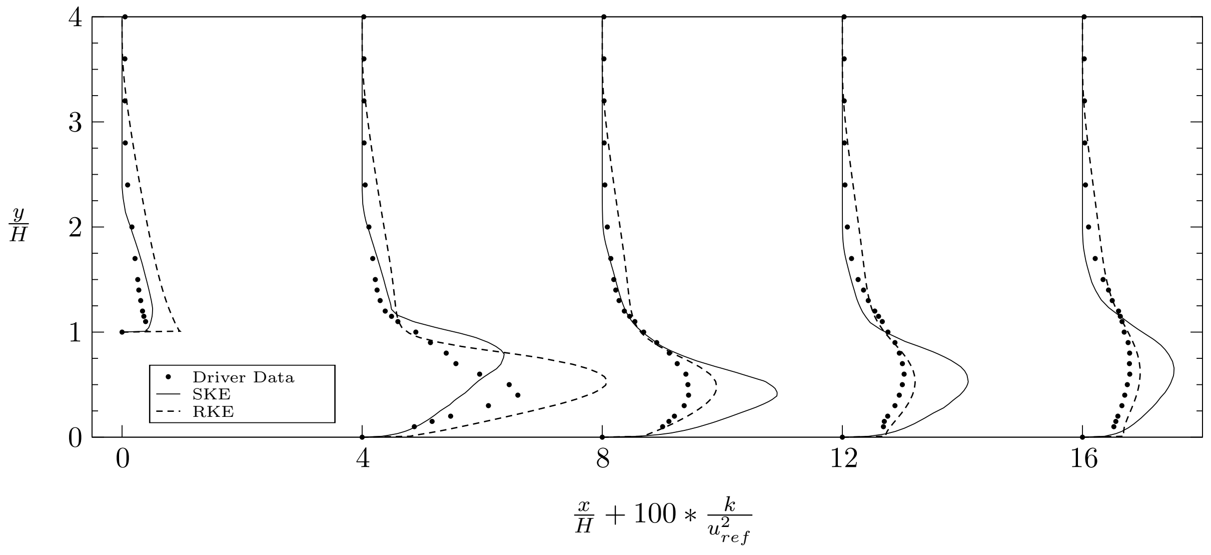


Figure 5.7: Backward facing step k profiles at $x/H = 0, 4, 8, 12, 16$ with the upper channel.

The skin friction coefficient is plotted along the bottom of the upper and lower channels in Fig. 5.8, along with the Driver results and four computation results found in Yoder [83]. The SKE model predicts the skin friction quite well over the entire range, but with the minimum value shifted to the right compared to other results. The RKE model falls close to the SST model for the recirculating range, but ends up with a friction coefficient much higher at the end of the lower channel. Both methods follow experimental results better than the $k - \epsilon$ models shown, which have a large peak after the recirculating region.

5.2.1.2 Simulation Setting Inflow at Step

Because flow at the step varies from the experiment, particularly for RKE, velocity and k profiles are set at the step ($x/H = 0$) and only the lower channel is simulated. Results for this case (SKE and RKE) are shown in Figs. 5.9–5.11. The velocity profiles for SKE are closer to the experiment than RKE, but they both match more closely above $y/H = 1$ than when the upper channel was simulated. k profiles also match better away from the wall than before, but both SKE and RKE over-predict k near the wall. For skin friction coefficient, RKE matches better near the step but

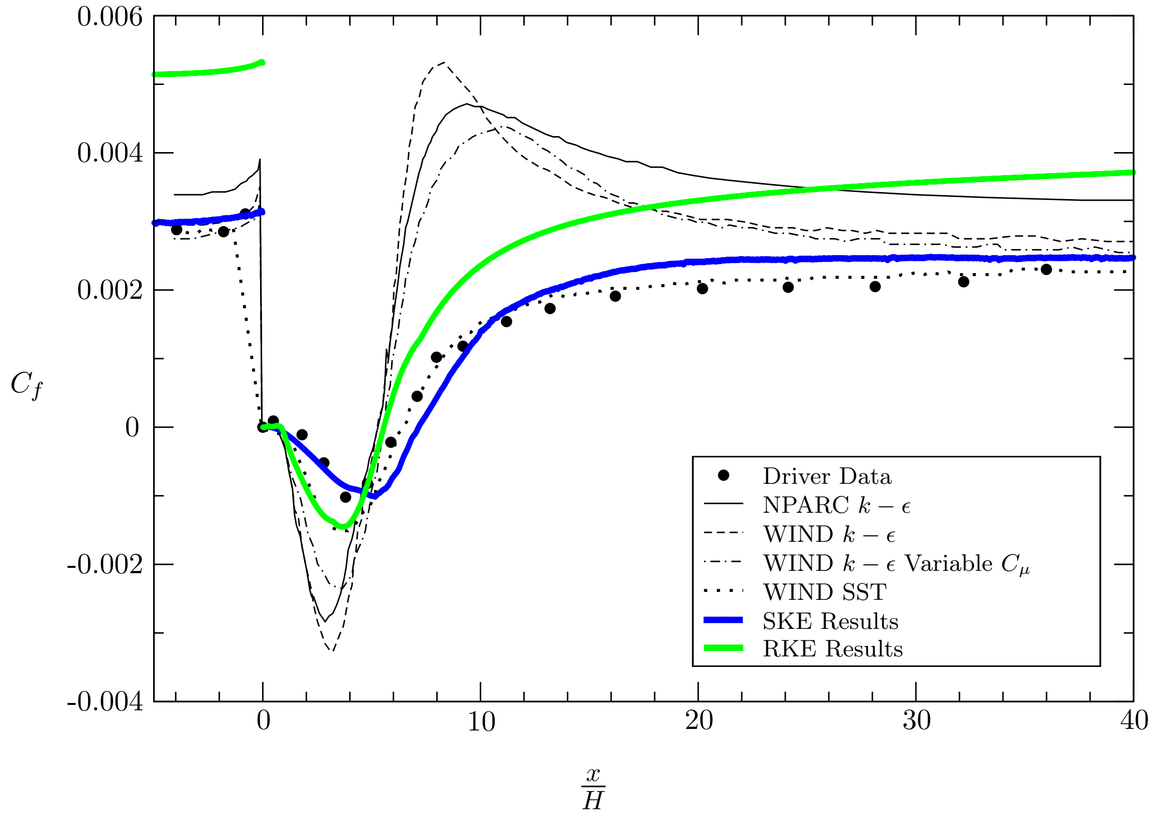


Figure 5.8: Backward facing step skin friction coefficient along step wall with the upper channel.

overshoots near the outlet (although less than with the upper channel), while SKE matches well downstream and not as well near the step.

Table 5.3 gives the recirculation length for the backward facing step, with data from Driver and Yoder.

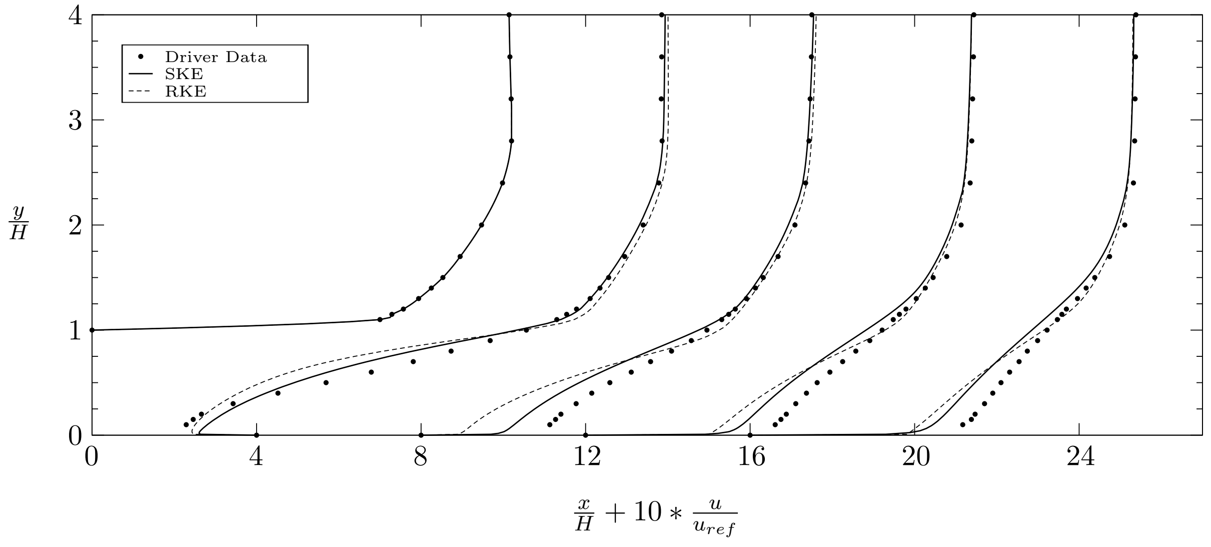


Figure 5.9: Backward facing step u velocity profiles at $x/H = 0, 4, 8, 12, 16$ with no upper channel.

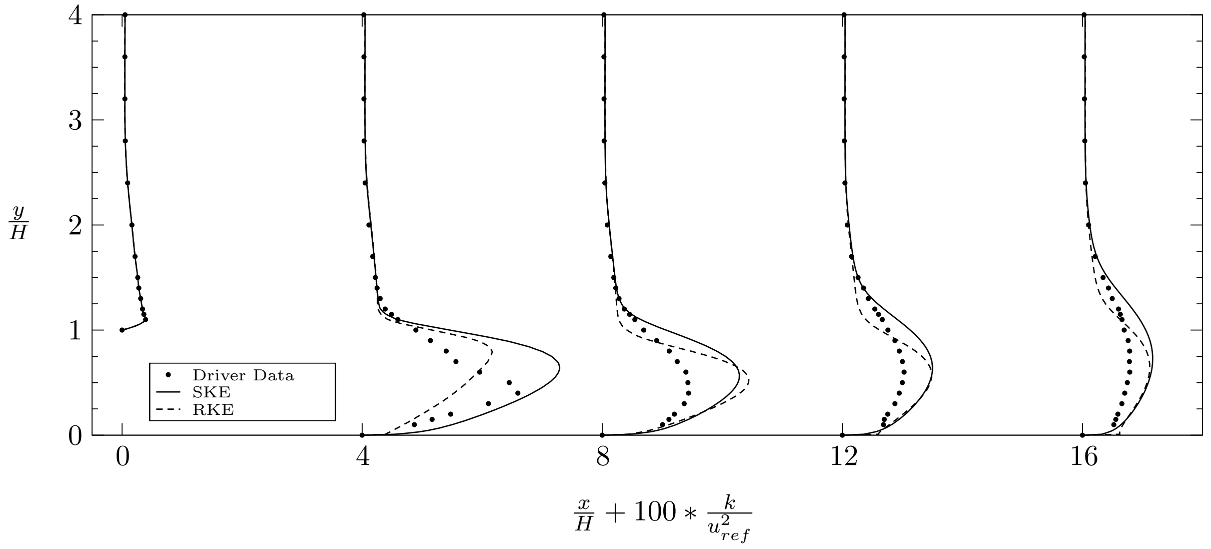


Figure 5.10: Backward facing step k profiles at $x/H = 0, 4, 8, 12, 16$ with no upper channel.

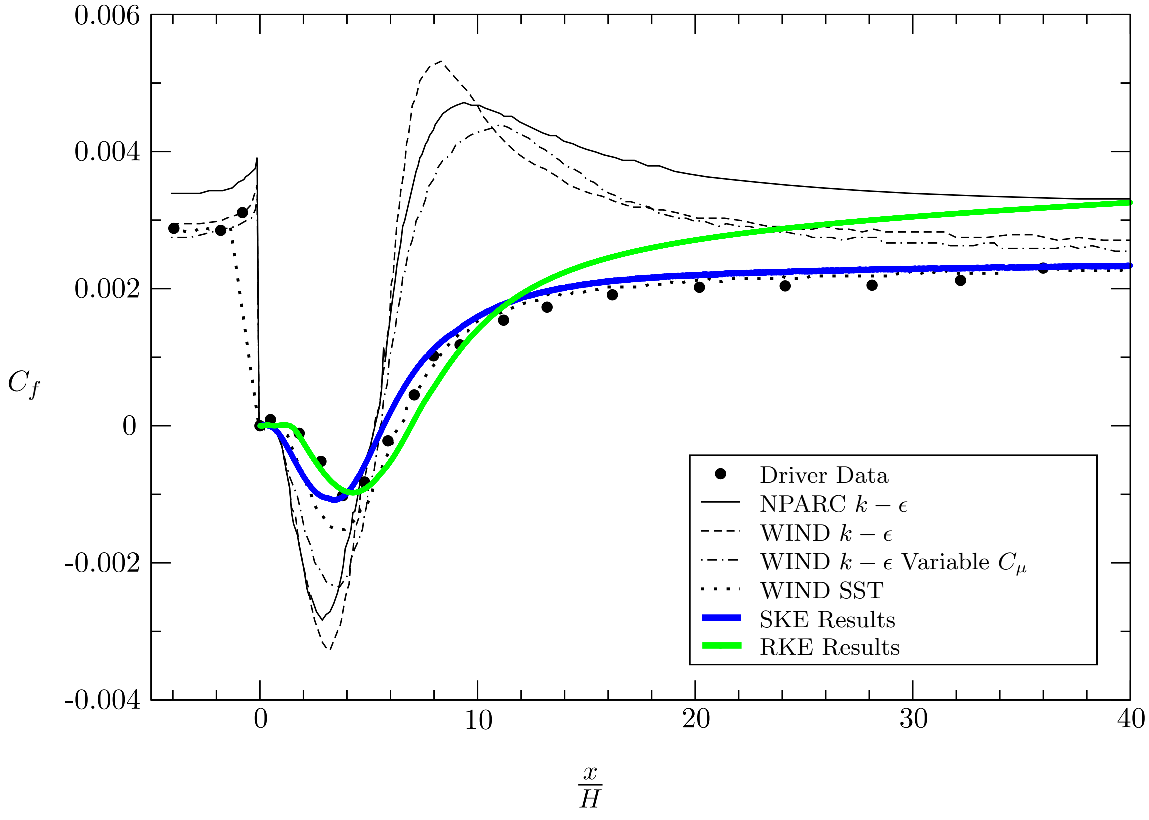


Figure 5.11: Backward facing step skin friction coefficient along step wall with no upper channel.

Table 5.3: Backwards facing step recirculation length (x_r/H.

Source	Model	x_r/H
Driver [82]	Experiment	6.26
Yoder [83]	NPARC $k-\epsilon$	5.31
Yoder [83]	WIND $k-\epsilon$	5.30
Yoder [83]	WIND $k-\epsilon$ Variable C_μ	5.55
Yoder [83]	WIND SST	6.43
Present Simulation	SKE	7.16
Present Simulation	RKE	5.56
Present Simulation	SKE w/ Inlet	5.68
Present Simulation	RKE w/ Inlet	6.95

5.2.2 Turbulent Vortex Shedding from a Square Cylinder

The next test case is the unsteady vortex shedding from behind a two-dimensional square cylinder. Although the RANS equations are time averaged, unsteady flows can still be simulated. The turbulence model captures lower energy turbulence, while the high energy associated with the mean flow vortex shedding is captured explicitly by the unsteady Navier-Stokes equations [36].

The square cylinder (Fig. 5.12) is centered at the origin, and the domain spans $-5L \leq x \leq 15$ and $-7 \leq y \leq 7$. The Reynolds number is $Re = \rho U_i L/\mu = 22,000$ [85]. The grid is refined around the cylinder and in the wake, with a near wall grid that results in $3 \leq y_+ \leq 35$.

Both SKE and RKE models are used. The RKE model uses the Crank-Nicolson method for time integration, but the SKE model was unstable for all tested time step sizes using Crank-Nicolson and fully implicit was used instead.

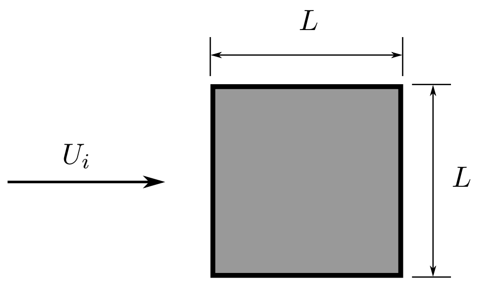


Figure 5.12: Square cylinder case.

The IRK-SIMPLER algorithm is used with a time step size of $\Delta t' = \Delta t/(U_i/L) = 0.05$ and simulations ran until $t' = 400$. Similar to the flat plate, after some time vortices shed from the square cylinder and eventually begin to shed at a constant frequency and magnitude. Figure 5.13 shows the coefficient of lift and drag on the cylinder for SKE and RKE. After $t' = 150$ the frequency and amplitude are constant for both methods. The SKE results have lower amplitude oscillations than the RKE results. The coefficient of drag for SKE appears constant, but there are oscillations with a small RMS. The average coefficient of drag, Strouhal number, recirculation length for averaged solutions (x_r/L), and RMS values of drag and lift coefficient ($(C_d)_{rms}$ and $(C_l)_{rms}$) are given in Table 5.4 and compared to other results found in Iaccarino [85].

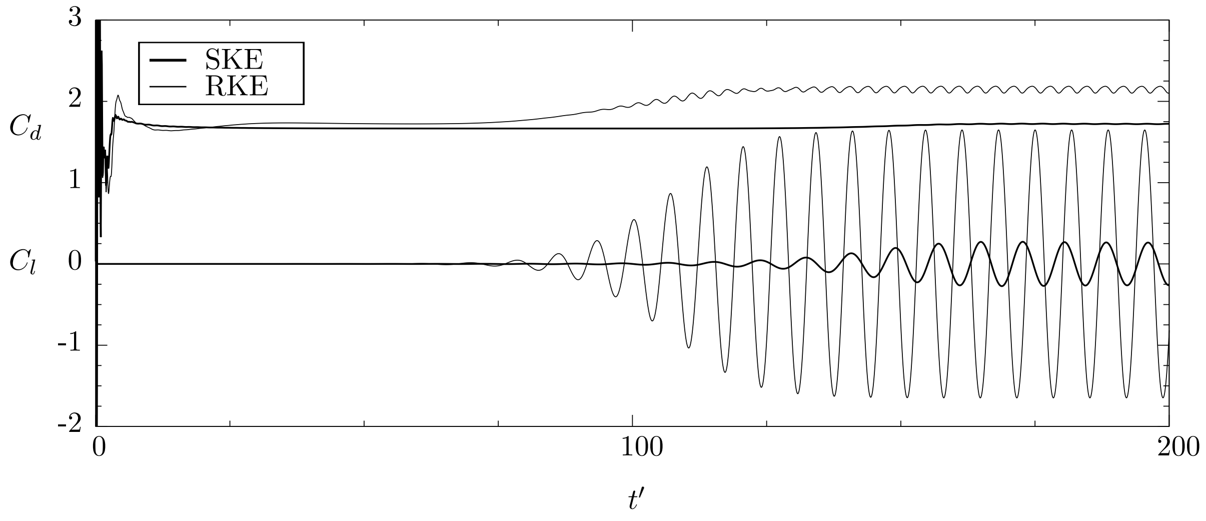


Figure 5.13: Square cylinder coefficient of lift and drag for SKE and RKE.

Table 5.4: Square cylinder results.

Source	Model	$\overline{C_d}$	Sr	x_r/L	$(C_d)_{rms}$	$(C_l)_{rms}$
Lyn [86]	Experiment	2.1	0.132	1.38	—	—
Lee [87]	Experiment	2.05	—	—	0.16-0.23	—
Vickery [88]	Experiment	2.05	—	—	0.1-0.2	0.68-1.32
Rodi [89]	LES	2.2	0.13	1.32	0.14	1.01
Rodi [89]	Two-Layer $k-\epsilon$	2.004	0.143	1.25	—	—
Iaccarino [85]	v^2-f [90]	2.22	0.141	1.45	0.056	1.83
Present Simulation	SKE	1.72	0.128	2.95	0.022	0.185
Present Simulation	RKE	2.15	0.147	1.43	0.029	1.174

The solutions are averaged in time after $t' = 200$ until the final time of $t' = 400$. Figures 5.14-5.17 show time averaged velocity profiles at five locations, $x/L = 0.875, 1.125, 1.875, 3.5$ and $y/L = 0$. The RKE results match the experimental results of Lyn and computational results of Iaccarino well, but the SKE results do not, particularly further down stream. The likely cause of the errors in SKE is because the Fully Implicit time integration method is used, which is only first order accurate.

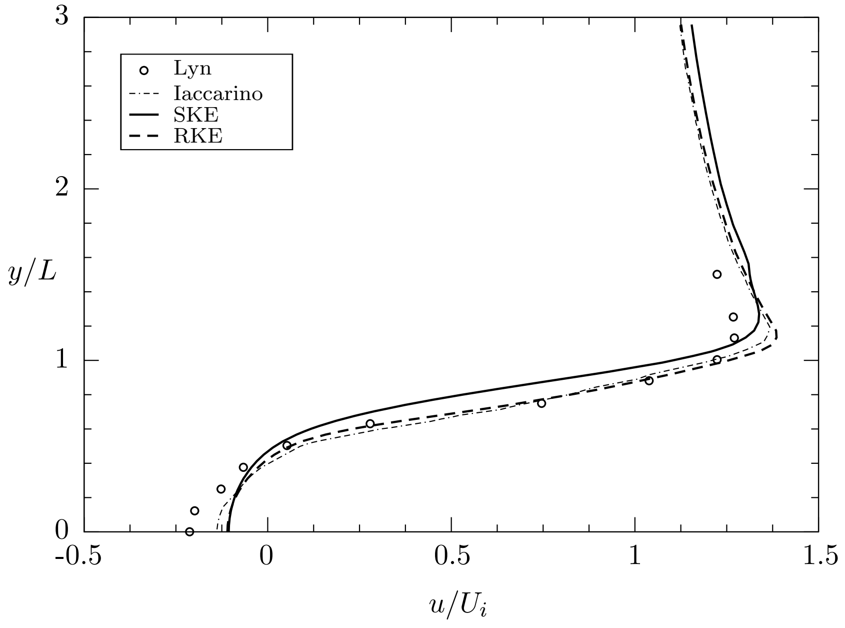


Figure 5.14: Square cylinder u/U_i profile at $x/L = 0.875$.

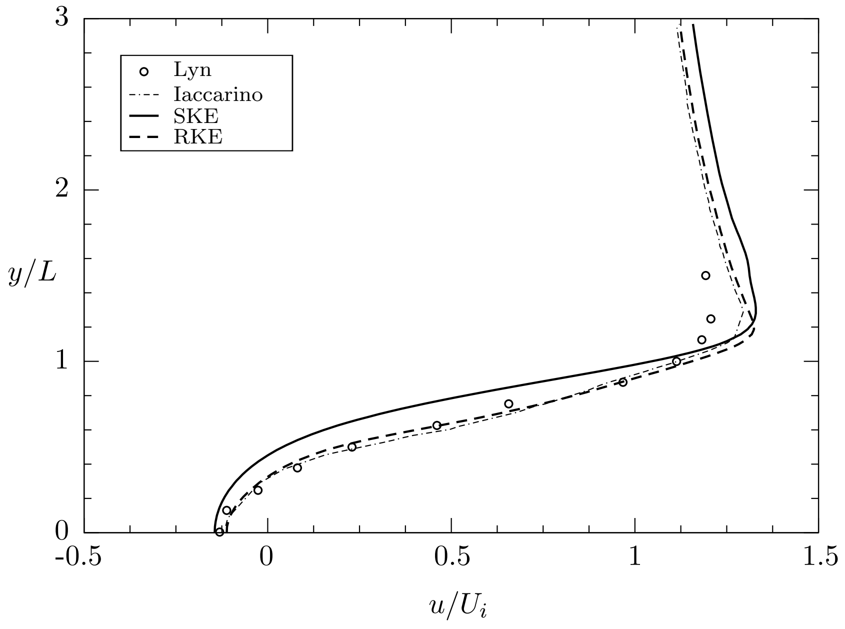


Figure 5.15: Square cylinder u/U_i profile at $x/L = 1.125$.

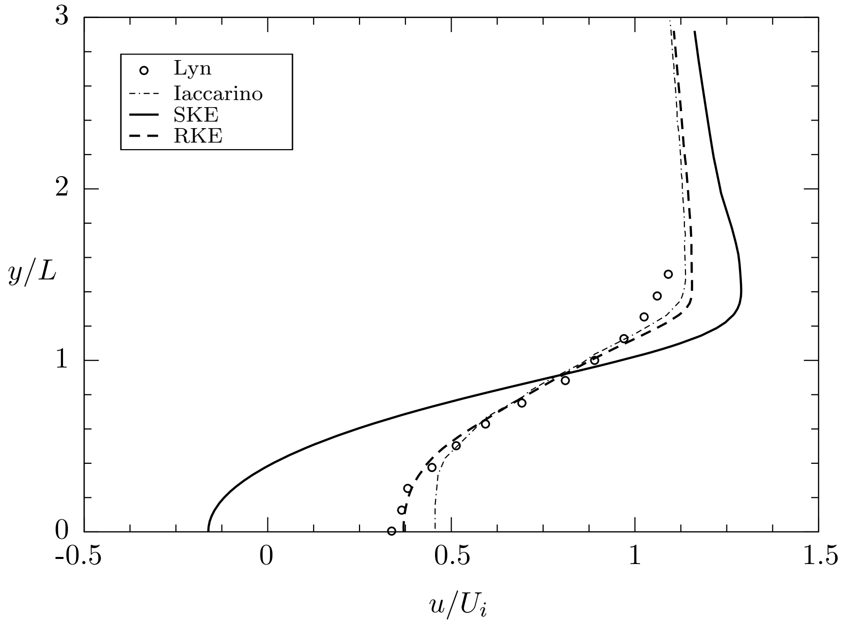


Figure 5.16: Square cylinder u/U_i profile at $x/L = 1.875$.

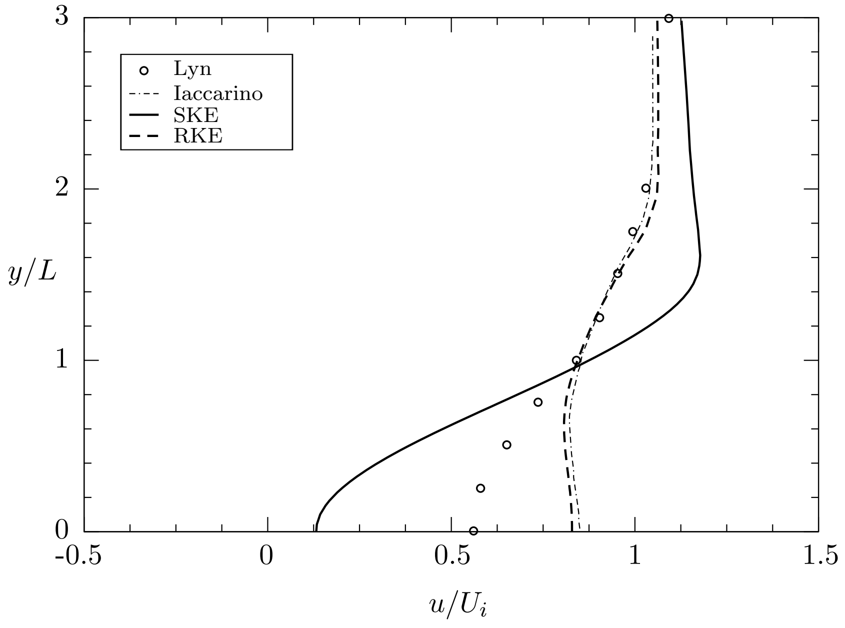


Figure 5.17: Square cylinder u/U_i profile at $x/L = 3.5$.

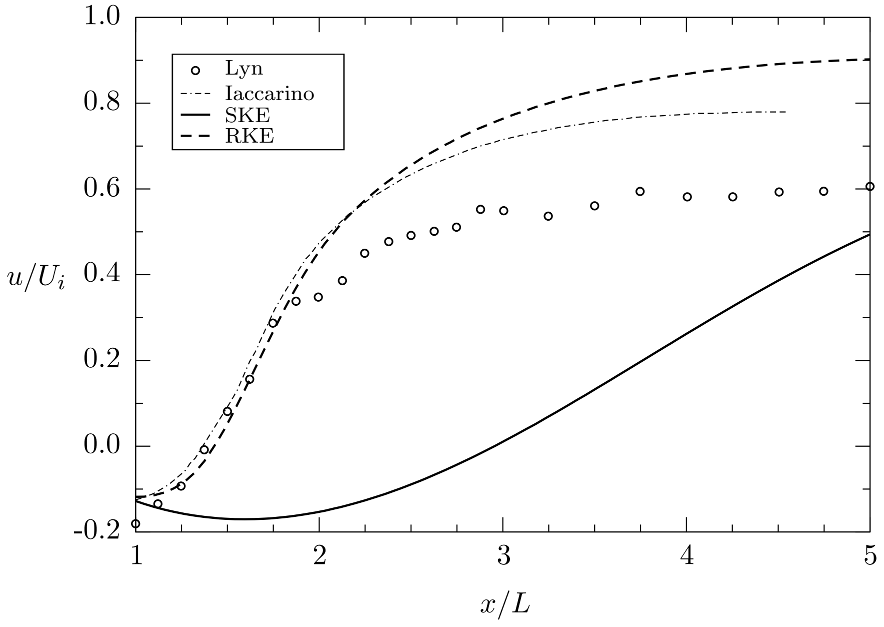


Figure 5.18: Square cylinder u/U_i profile at $y/L = 0$.

5.3 Conclusions

The triangular unstructured scheme with IRK-SIMPLER and the $k-\epsilon$ turbulence model accurately simulates both steady and unsteady turbulent flows. The standard $k-\epsilon$ model results match the experimental trends well for the steady backward facing step case, but due to stability issues, the Fully Implicit scheme is required for unsteady cases, which gives inaccurate results. The realizable $k-\epsilon$ model results match experiments well for both the steady and unsteady problems, and is able to use Crank-Nicolson to give accurate unsteady results.

For triangular grids, results are more smooth and accurate when a line of nodes is created parallel to the wall surface, like in a structured grid. For more complex geometries, another grid topology, such as prisms or hexahedral cells, can be created normal to the surface to provide a better method for boundary layer problems.

CHAPTER 6. ROTOR MODELING

The IRK-SIMPLER algorithm is tested, alongside SIMPLER and RK-SIMPLER, by simulating flows around wind turbines. Both structured and unstructured grid formulations are used to test IRK-SIMPLER. The structured method has the advantage of relative simplicity in calculating fluxes and coefficients, as well as efficient linear solvers (ex. line-by-line TDMA), but for large scale wind farms a structured grid might not be the most appropriate. Unstructured grids, on the other hand, are natural for wind farms as they can be clustered near the turbines and in the wake regions, while remaining coarse in regions where the flow is relatively uniform. The disadvantages of unstructured grids are the additional computations required to calculate and reconstruct fluxes and the difficulty in increasing the spatial accuracy. Because both methods have merits, they are both tested to see how RK based algorithms behave.

6.1 Rotor Source Formulation

The momentum source method (as developed in [40],[41],[42],[43]) uses ideas from the blade element momentum theory to discretize a blade and impart its forces onto the flow through the momentum equation sources. Because the blades are not modeled as bodies in the domain, coarser grids that do not require grid movement can be used.

6.1.1 Rotor Coordinate Systems

Starting from the Cartesian coordinate system that the flow is solved on, several coordinates systems are defined to aid in developing the momentum sources.

6.1.1.1 Rotor Cartesian Coordinates (ξ, η, ζ)

A Cartesian coordinate system is fixed to the rotor with the origin at the rotor center. The coordinate ξ is normal to the rotor disk and aligned with the axis of rotation (with a rotation speed Ω), while η and ζ are normal to the axis of rotation (Fig. 6.1). A positive rotation speed Ω is a clockwise rotation, and a negative rotation speed is counter-clockwise. The incoming velocity V_i is expected to be in the direction of $\hat{e}_\xi$ for typical cases, like a horizontal axis wind turbine (HAWT) with no yaw angle or a helicopter in hover.

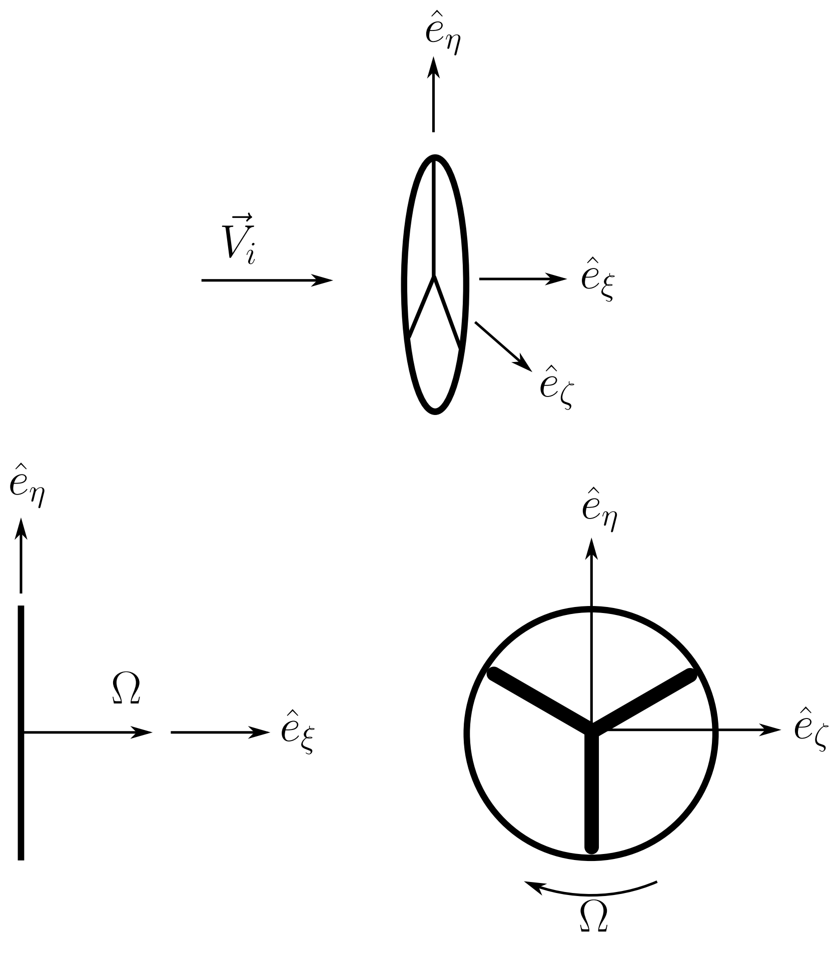


Figure 6.1: Rotor Cartesian coordinate system.

If the rotor is centered at the global Cartesian coordinates (x_R, y_R, z_R), the transfer from (x, y, z) to (ξ, η, ζ) is

$$\begin{bmatrix} \xi \\ \eta \\ \zeta \end{bmatrix} = \begin{bmatrix} \cos(B) & \sin(B)\sin(A) & -\sin(B)\cos(A) \\ 0 & \cos(A) & \sin(A) \\ \sin(B) & -\cos(B)\sin(A) & \cos(B)\cos(A) \end{bmatrix} \begin{bmatrix} x - x_R \\ y - y_R \\ z - z_R \end{bmatrix} \tag{6.1}$$

$$= M \begin{bmatrix} x - x_R \\ y - y_R \\ z - z_R \end{bmatrix} , \tag{6.2}$$

where the transformation first rotates by angle B about the y-axis, then by angle A about the x-axis. For a HAWT with free-stream velocity in the x-direction, the angle B is set the yaw angle if angle A is set to 90 degrees. For a helicopter rotor, angle B is set to 90 degrees and angle A controls the banking of the rotor (with an angle $A = 0$ leading to a rotor normal to the z direction). The transformation matrix is orthogonal, leading to the following inverse transformation.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = M^{-1} \begin{bmatrix} \xi \\ \eta \\ \zeta \end{bmatrix} + \begin{bmatrix} x_R \\ y_R \\ z_R \end{bmatrix} \tag{6.3}$$

The velocity is transformed similarly.

$$\begin{bmatrix} u_\xi \\ u_\eta \\ u_\zeta \end{bmatrix} = M \begin{bmatrix} u \\ v \\ w \end{bmatrix} \qquad \begin{bmatrix} u \\ v \\ w \end{bmatrix} = M^{-1} \begin{bmatrix} u_\xi \\ u_\eta \\ u_\zeta \end{bmatrix} \tag{6.4}$$

6.1.1.2 Rotor Cylindrical Coordinates (r, θ, z)

A cylindrical coordinate system is fixed to the rotor with the origin at the rotor center. Figure 6.2 shows the rotor with both Cartesian and cylindrical coordinates.

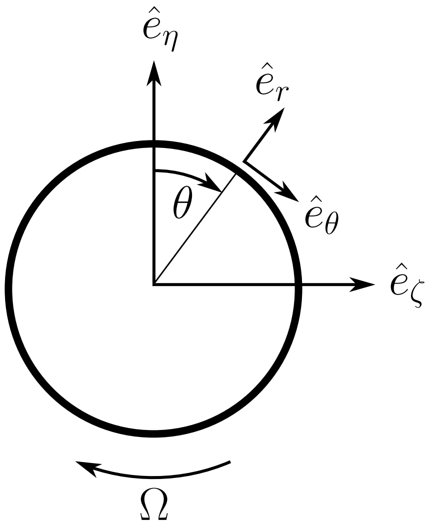


Figure 6.2: Rotor cylindrical and Cartesian coordinate system.

The transformations between the rotor Cartesian to cylindrical coordinate systems are

$$\begin{bmatrix} v_r \\ v_\theta \\ v_z \end{bmatrix} = \begin{bmatrix} 0 & \cos(\theta) & \sin(\theta) \\ 0 & -\sin(\theta) & \cos(\theta) \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_\xi \\ v_\eta \\ v_\zeta \end{bmatrix} \tag{6.5}$$

$$\begin{bmatrix} v_\xi \\ v_\eta \\ v_\zeta \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \end{bmatrix} \begin{bmatrix} v_r \\ v_\theta \\ v_z \end{bmatrix} . \tag{6.6}$$

6.1.1.3 Rotor Local Blade Coordinates (n, θ, s)

A local coordinate system is created on the blades that follows the deflection of the blade out of the $z = 0$ plane, where, at any given point, the deflection angle is δ (Fig. 6.3). For a rigid rotor with a constant cone angle, δ is a constant for all blade sections. The coordinate s follows the blade surface from the rotor center to the radius R, and the coordinate n is normal to the surface.

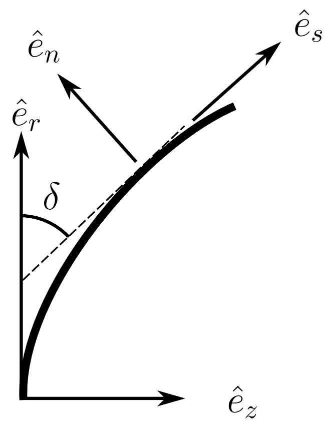


Figure 6.3: Rotor local and cylindrical coordinate system.

The transformations between the rotor cylindrical and local coordinate systems are

$$\begin{bmatrix} v_n \\ v_\theta \\ v_s \end{bmatrix} = \begin{bmatrix} \sin(\delta) & 0 & -\cos(\delta) \\ 0 & 1 & 0 \\ \cos(\delta) & 0 & \sin(\delta) \end{bmatrix} \begin{bmatrix} v_r \\ v_\theta \\ v_z \end{bmatrix} \tag{6.7}$$

$$\begin{bmatrix} v_r \\ v_\theta \\ v_z \end{bmatrix} = \begin{bmatrix} \sin(\delta) & 0 & \cos(\delta) \\ 0 & 1 & 0 \\ -\cos(\delta) & 0 & \sin(\delta) \end{bmatrix} \begin{bmatrix} v_n \\ v_\theta \\ v_s \end{bmatrix} \tag{6.8}$$

6.1.2 Discrete Rotor Blade Sections

The momentum source method for rotors starts by discretizing the blades into a number of radial sections with a control point at the center of each section (Fig. 6.4). Typically around 100 radial sections accurately discretize the blades.

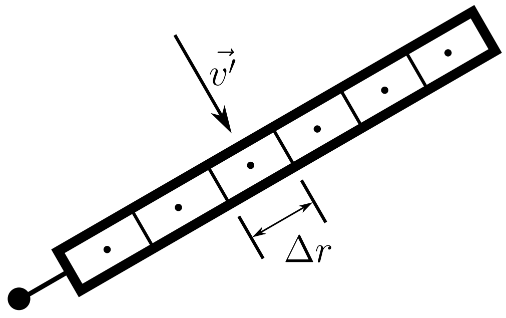


Figure 6.4: Rotor discretization.

6.1.2.1 HAWT Blade Section Relative Velocity

Up to this point, the formulation has been general to any rotor, but from here on the focus will be on HAWT rotors. As the rotor blades move through the domain and intersect grid cells, the relative velocity on the blades is found by first converting the flow velocity from (u, v, w) to (v_n, v_θ, v_s) using Eqs. 6.4, 6.5, and 6.7. Noting that $\Omega = \frac{d\theta}{dt}$, the blades are moving with velocity $(v_\theta)_b = r\Omega$ in the $\hat{e}_\theta$ direction and $(v_n)_b$ in the $\hat{e}_n$ direction. The $(v_n)_b$ velocity comes from any flapping terms (or unsteady out-plane deflection), which are not common for wind turbine blades. The relative velocity over the blades is found by

$$v'_n = v_n - (v_n)_b \tag{6.9}$$

$$v'_\theta = v_\theta - (v_\theta)_b = v_\theta - r\Omega \tag{6.10}$$

$$v'_s = v_s \ , \tag{6.11}$$

where (v_n, v_θ, v_s) is the velocity of the flow at the blade section and (v'_n, v'_θ, v'_s) is the relative velocity on the blade section. This velocity is used to calculate the airfoil angle of attack and coefficients. Because the airfoil sections are two-dimensional, only the n and θ velocity components are considered when calculating the airfoil forces.

6.1.2.2 Airfoil Section Angles and Forces

Figure 6.5 shows the $n-\theta$ plane, on which each airfoil section is located. The direction of the airfoil depends on whether the rotor is spinning clockwise or counter-clockwise (Figs. 6.6 and 6.7). In the clockwise Fig. 6.6, $v'_n < 0$ and $v'_\theta < 0$, and in the counter-clockwise Fig. 6.7, $v'_n < 0$ and $v'_\theta > 0$. These values are typical, but not true in all cases. The angle ϕ is the airfoil section pitch angle and is determined by the blade twist distribution and the blade pitch angle. The relative velocity magnitude is $v' = \sqrt{(v'_n)^2 + (v'_\theta)^2}$.

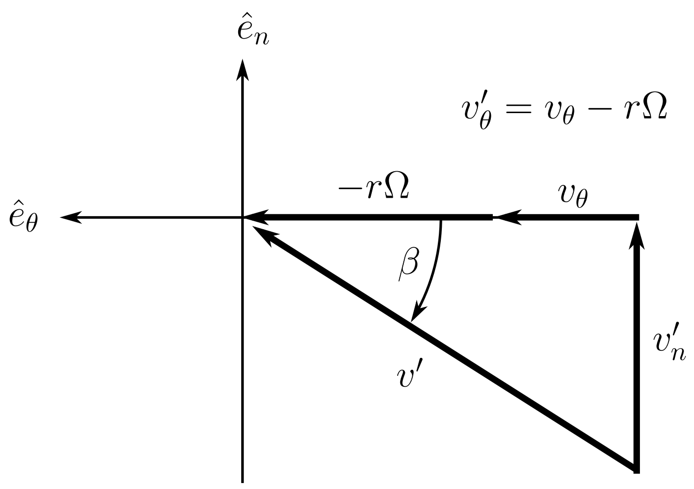


Figure 6.5: Airfoil section $n-\theta$ plane.

Given the relative velocity components v'_n and v'_θ, the flow angle β is

$$\beta = \tan^{-1}\left(\frac{v'_n}{v'_\theta}\right) \ . \tag{6.12}$$

Angle β' is calculated for clockwise and counter-clockwise as

<u>Clockwise</u>	<u>Counter-Clockwise</u>
$\beta' = 180^o + \beta$	$\beta' = 360^o - \beta$. (6.13)

The airfoil angle of attack is then given by

$$\alpha = \beta' - \phi \ . \tag{6.14}$$

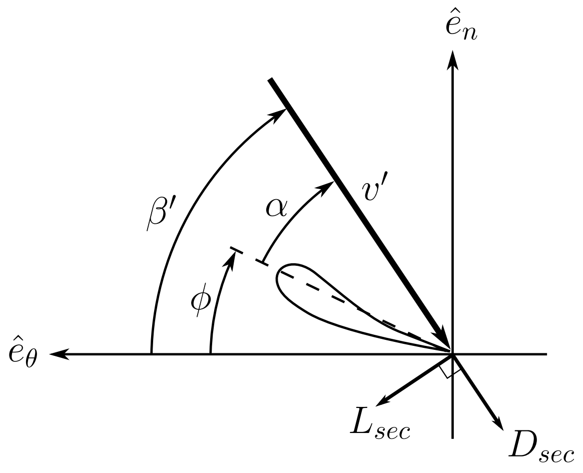


Figure 6.6: Airfoil orientation for clockwise rotation.

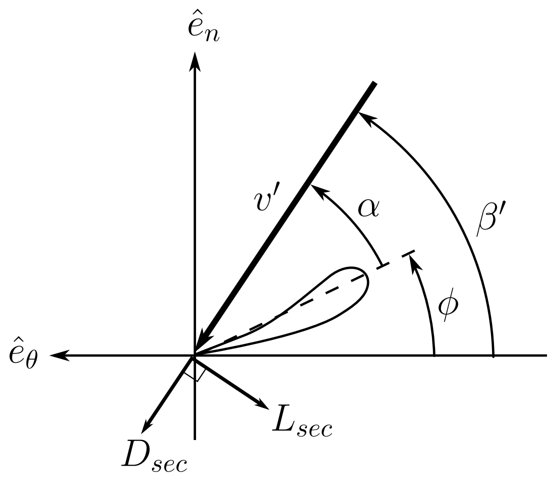


Figure 6.7: Airfoil orientation for counter-clockwise rotation.

The airfoil coefficient of lift and drag are found by look up values from a table. The coefficients are functions of angle of attack and can also be dependent on Mach number, Reynolds number, or any other parameters causes different airfoil characteristics.

$$C_l = C_l(\alpha, M, Re, ...) \qquad C_d = C_d(\alpha, M, Re, ...) \tag{6.15}$$

From the two-dimensional coefficients, section lift and drag are calculated by

$$L_{sec} = \left[\frac{1}{2}\rho(v')^2 c\right] C_l \qquad D_{sec} = \left[\frac{1}{2}\rho(v')^2 c\right] C_d \ , \tag{6.16}$$

where c is the chord length of the airfoil at the section of interest.

The section lift and drag are transformed to forces in the n and θ directions by

<u>Clockwise</u> <u>Counter-Clockwise</u>

$$(F_n)_{sec} = -\cos(\beta')L_{sec} - \sin(\beta')D_{sec} \qquad (F_n)_{sec} = -\cos(\beta')L_{sec} - \sin(\beta')D_{sec} \tag{6.17}$$

$$(F_\theta)_{sec} = \sin(\beta')L_{sec} - \cos(\beta')D_{sec} \qquad (F_\theta)_{sec} = -\sin(\beta')L_{sec} + \cos(\beta')D_{sec} \ . \tag{6.18}$$

The forces are transformed back into the (r, θ, z) system by

$$(F_r)_{sec} = \sin(\delta)(F_n)_{sec} \tag{6.19}$$

$$(F_\theta)_{sec} = (F_\theta)_{sec} \tag{6.20}$$

$$(F_z)_{sec} = -\cos(\delta)(F_n)_{sec} \ . \tag{6.21}$$

From here, two tasks are required. The first is to integrate forces over the entire blade to find the rotor performance (thrust, torque, and power). The thrust and torque are calculated by integrating the forces over the span of all the blades (N_b is the total number of blades).

$$T = \sum_{n=1}^{N_b} \int_{r=r_{hub}}^{R} (F_z)_{sec} \, dr \tag{6.22}$$

$$Q = \sum_{n=1}^{N_b} \int_{r=r_{hub}}^{R} r(F_\theta)_{sec} \, dr \ , \tag{6.23}$$

where T is thrust, Q is torque, and r_{hub} is the radius where the hub and rotor meet. The rotor power is computed as

$$P = Q\Omega \ . \tag{6.24}$$

The second task is to add forces into the momentum equation sources. The section forces are integrated over the section to yield

$$F_r = \Delta r(F_n)_{sec} \tag{6.25}$$

$$F_\theta = \Delta r(F_\theta)_{sec} \tag{6.26}$$

$$F_z = \Delta r(F_n)_{sec} \ . \tag{6.27}$$

These forces are transformed first into (ξ, η, ζ) coordinates by Eq. 6.6, then to (x, y, z) using the M^{-1} matrix in Eq. 6.1, giving force on each radial section center at $\vec{F} = (F_x, F_y, F_z)$.

This is the force acting on the blade, therefore, the force acting on the fluid must be $-\vec{F}$. So, as each radial section center intersects a cell, the force is transferred to the momentum equations for that cell by

$$b_u = b_u - F_x \tag{6.28}$$

$$b_v = b_v - F_y \tag{6.29}$$

$$b_w = b_w - F_z \ . \tag{6.30}$$

6.1.3 Steady Rotor Model

The steady rotor model looks at all cells intersected by the blades over the full 360 degree rotation and weights the momentum source by the fraction of rotation that occurs in each cell.

$$W = \frac{N_b \Delta\theta_{cell}}{2\pi} \tag{6.31}$$

where W is the weighting applied to the blade section force before it is added into the momentum source, N_b is the number of rotor blades, and $\Delta\theta_{cell}$ is the angle the rotor blades sweep between entering and exiting a grid cell in radians. The steady rotor model averages the rotor forces that occur over the full rotation for each time step. The benefit of the steady rotor model is the ability to yield good results on relatively coarse grids with large time step sizes (or using a steady algorithm). The disadvantage of the steady model is the unsteady effects, like the tip and root vortex, cannot be captured. Because of this an unsteady rotor model is often desired.

For the steady rotor model, all cells intersected by the rotor are computed in the preprocessing step, before time steps start. The procedure for detecting intersected cells is as follows.

- Loop through all radial sections of a rotor.
  - Find the first cell intersected at $\theta = 0$.
  - Loop through θ.
    - Increase θ until it reaches the face of the currently intersected cell.
    - Store information about the cell being intersected including $\overline{\theta}_{cut}$, $\Delta\theta_{cell}$, r_{cut}, and $N_{cut-total}$.

    * Once $\theta = 0$ again (i.e. all $\Delta\theta_{cut}$ add up to 2π). Exit and go to next radial section.

Note that $\overline{\theta}_{cut}$ is the value of θ midway between the two face intersection points, r_{cut} is the radius for the given intersection, and $N_{Cut-Total}$ is the total number of intersections made by the rotor.

Every time the rotor source needs to be computed the following procedure is used.

- Loop through all intersections ($N_{Cut-Total}$).
  - Find the flow velocity at the current r_{cut} and $\overline{\theta}_{cut}$.
  - Find the integrated force on that section.
  - Include the force in the momentum equations as
    $b_u = b_u - WF_x = b_u - \frac{N_b \Delta\theta_{cell}}{2\pi} F_x$
    $b_v = b_v - WF_y = b_v - \frac{N_b \Delta\theta_{cell}}{2\pi} F_y$
    $b_w = b_w - WF_z = b_w - \frac{N_b \Delta\theta_{cell}}{2\pi} F_z$.

It should be noted that the rotor source is computed at each stage of the IRK-SIMPLER algorithm and included in the unsteady source terms with another weighting term based on the DIRK method coefficients.

6.1.4 Unsteady Rotor

The unsteady rotor model considers the blade locations to be time accurate, with each blade moving by and angle $\Delta\theta_{blade} = \Omega\Delta t$ each time step (where Ω is the rotor rotation speed and $\Delta t = t^{n+1} - t^n$ is the time step size). As each blade section moves through $\Delta\theta_{blade}$, more than one grid cell can be intersected. Three different methods are developed to handle the unsteady rotor intersections and source calculation.

6.1.4.1 Method 1

The first method is similar to the steady rotor model, and the rotor source is calculated and added to all cells intersected within the time step and weighted by

$$W = \frac{\Delta\theta_{cell}}{\Omega\Delta t} = \frac{\Delta\theta_{cell}}{\Delta\theta_{blade}} . \tag{6.32}$$

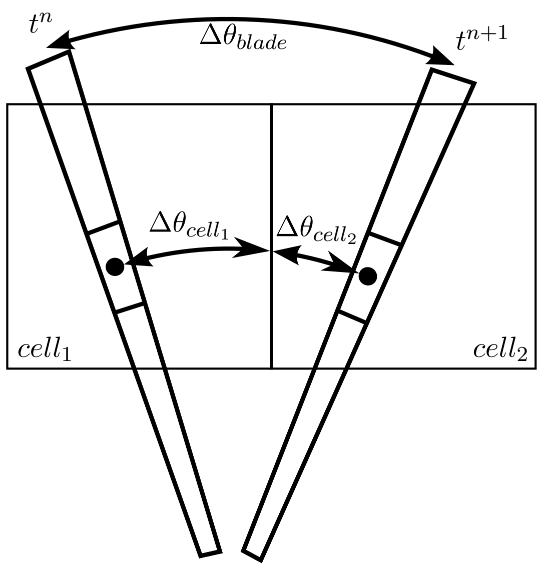


Figure 6.8: Unsteady rotor rotation and intersection of grid cells.

For example, if there is a small time step size then only one cell might be intersected, in which case the weighting is equal to one. On the other hand, for larger time step sizes and/or small grid cells, many grid cells might be intersected by a blade section and the rotor source is weighted and added to each. Figure 6.8 shows an example in which two cells are intersected. This method averages the source out over the entire time step.

The procedure for detecting the intersected cells at each time step for method 1 is as follows.

- Loop through all rotor blades.
  - Loop through all radial sections of a rotor.
    - Find the first cell intersected at $\theta = \theta^n$.
    - Loop through θ.
      - Increase θ until it reaches the face of the currently intersected cell .
      - If θ for the next face is past θ^{n+1}, use θ^{n+1} for the out face value.
      - Store information about the cell being intersected including $\overline{\theta}_{cut}$, $\Delta\theta_{cell}$, r_{cut}, and $N_{cut-total}$.

    · Once $\theta = \theta^{n+1}$ (i.e. all $\Delta\theta_{cut}$ add up to $\Delta\theta_{blade}$, exit and go to next radial section.

- Once all radial sections are complete, go to the next blade.

This procedure is similar to the steady rotor, except the intersections for each individual blade is found and the blades are swept through $\Delta\theta_{blade}$, not 2π.

The procedure for adding the rotor force to the momentum equations is similar to the steady procedure, except the weighting term is now from Eq. 6.32.

6.1.4.2 Method 2

The second method keeps the rotor blade at one location in time t^{n+1} (at the end of the time step) and adds the rotor source into all the cells intersected. For this method, each blade section only intersects one cell each time step, and all the source goes to that cell. This is similar to the Fully Implicit or Euler implicit integration method.

6.1.4.3 Method 3

The last method uses a time integration method that integrates the momentum equations in time to find both the location and weight for the rotor source. For SIMPLER and RK-SIMPLER, Crank-Nicolson (i.e. trapezoid or midpoint method) is used with half of the source at time t^n and half at time t^{n+1}. Using Crank-Nicolson, the blade intersections are found at time t^n (with one intersection for each blade section), and the velocities at time t^n are used to find the rotor source with a weighting of half. The other half of the source is found by calculating the rotor intersections at time t^{n+1} and using the velocities at t^{n+1} to find the rotor source. Using IRK-SIMPLER, both two and three stage methods are used, and each stage the intersections are found and the rotor source is computed with the stage velocities and the weights directly from the DIRK method. This is different from methods 1 and 2, which both find the intersected cells once and use the velocities at t^n and t^{n+1} for SIMPLER and RK-SIMPLER, or use the stage velocities for IRK-SIMPLER to find the rotor source. Method 3 uses both time accurate blade locations and time accurate velocity values to compute the rotor source. This method is motivated by the results in Appendix A, which show the importance of using the stage time to evaluate unsteady sources.

Figure 6.9 shows the pattern for each unsteady source method for an exaggerated time step with $\Delta\theta_{blade} = 30^o$. In Fig. 6.9, method 3 shows the source for IRK-SIMPLER with the two stage DIRK method.

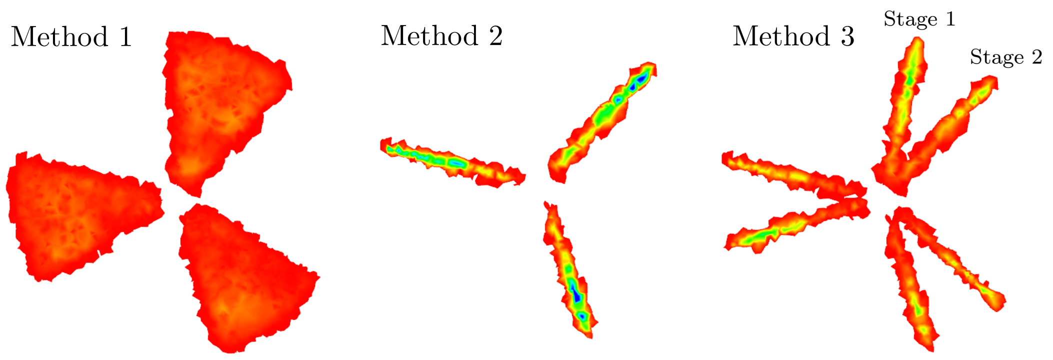


Figure 6.9: Rotor source contours for three unsteady rotor source methods[1].

For both the steady and unsteady rotor sources, when adding the rotor source to a grid cell in the staggered Cartesian structured grid, all the source is added to the cell which has a cell centered velocity component. For the vertex-centered tetrahedral unstructured grid, the rotor source is calculated as it intersects a tetrahedral cell, and the rotor source is added to that cell. To add the rotor source to each control volume, the two methods were tested had little difference. The first method adds a quarter of the tetrahedral element rotor source to all four nodes that make up the element. The second method adds the source to each node with a fraction based on the inverse of the square of the distance to each node, so that the closer the intersection is to a node the larger the source is for that node. The difference between the two methods is small, and the seconds method is used for the results given in present work.

The momentum source method uses known two-dimensional airfoil data (from experiments or computations) to look up the forces on discretized rotor blade sections. That airfoil data (namely coefficient of lift and drag) are then integrated into forces for each blade section and added into the

[1]Red is low magnitude and blue is high magnitude rotor source. Method 3 is shown with DIRK2.

momentum equations as a source term. The momentum source model is applied to vertical axis wind turbines [40], helicopters [41],[42],[43], and horizontal axis wind turbines [44].

The momentum source model uses the following steps to impart the wind turbine blade forces onto the flow:

1. Discretize the rotor blade into many sections along the radius of the blade.
2. Find the cells intersected by the blade sections at any point in time.
3. For each intersected cell, find the relative velocity on the blade section.
4. Using known, two-dimensional airfoil data, look up the airfoil lift and drag coefficients, which are a function of velocity magnitude, angle of attach, Reynolds number, Mach number, and other effects.
5. Add the three-dimensional force for each blade section into the intersected cells.
6. Integrate the force over all blade sections to find the total forces and moments on the rotor.

6.2 Results

To test different algorithms, a test case of the NREL Combined Experiment Rotor [91] is used (a 10 meter diameter, stall regulated, constant pitch horizontal axis wind turbine). The rotor is simulated as an isolated turbine with no tower, nacelle, ground, or atmospheric boundary layer. The domain boundary is spaced $7.5D$ away from the rotor in all directions except in the downwind direction which is $15D$ away from the rotor. Specifications for the turbine are given in Table 6.1.

6.2.1 Steady Simulations

The turbine is simulated over a range of wind speeds from 7 m/s to 23 m/s, with and without the tip correction model. The rotor power is computed, and generator power is found from the relationship given in [92]

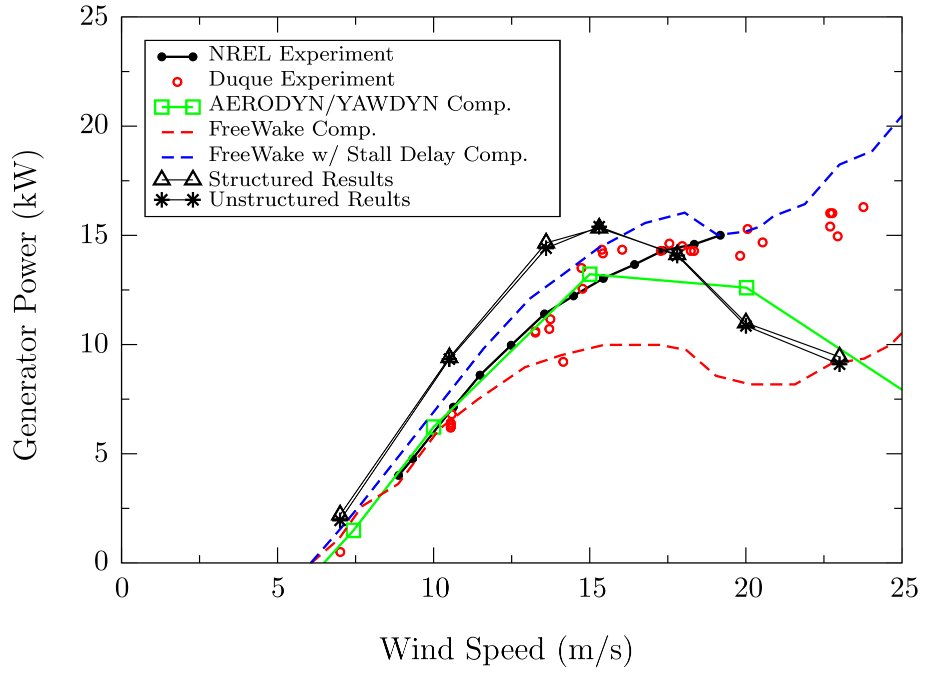
$$P_{gen} = 0.9036 P_{rotor} - 0.847 \;, \tag{6.33}$$

Table 6.1: NREL combined experiment rotor specifications.

Diameter	10 m
Number of Blades	3
Rotation Speed	72 RPM
Chord (constant)	0.45 m
Hub Radius (r/R)	0.14
Blade Pitch	12^o
Cone Angle	3.5^o
Twist (constant)	0^o
Airfoil (constant)	s809

where both generator and rotor power are measured in kW. Figure 6.10 shows the generator power versus wind speed with and without tip correction, respectively. The results for both the structured and unstructured grids match well. The data is compared to the power curve given by NREL in Butterfield [91], experimental results from Duque [92], and three computational results presented in Duque. The computational results include a Blade Element Method called YAWDYN/AERODYN [93], and a vortex-lattice method known as CAMRAD II [94] using a free wake model, with and without a stall delay model developed by Du and Selig [95].

The present results for the structured and unstructured grids follow the trends in the external results with an over-prediction in power before stall (around 15 m/s), and under-prediction of power after stall that has similarities to other numerical results.



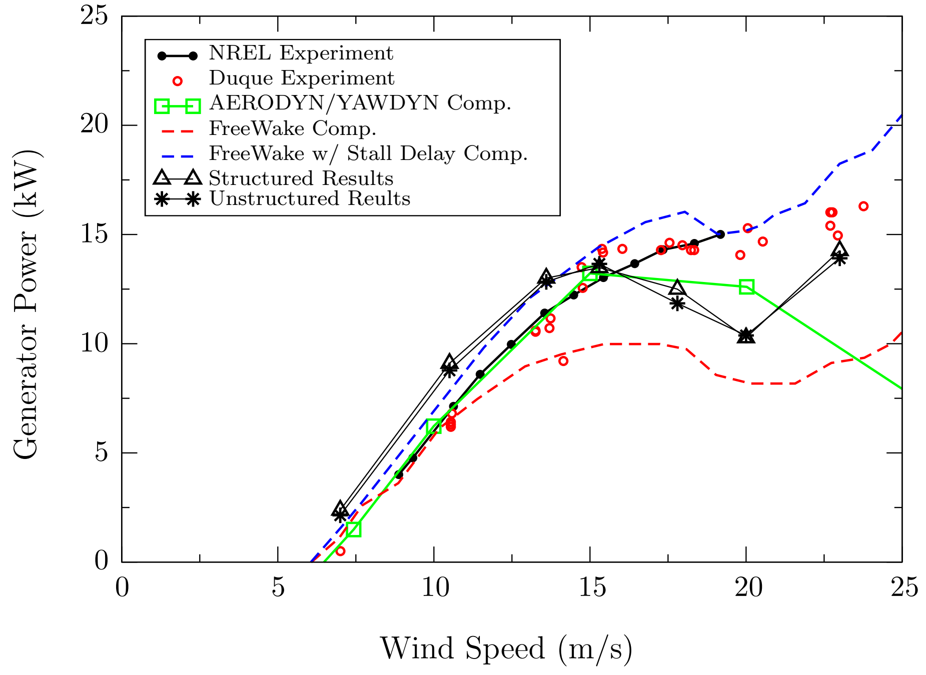


Figure 6.10: Steady power production with (bottom) and without (top) tip correction.

The SIMPLER, RK-SIMPLER, and IRK-SIMPLER algorithms all result in the same steady power prediction. Table 6.2 shows the runtime for the structured grid results, and Table 6.3 shows the runtime for the unstructured grid results. For the structured grid simulations, the SIMPLER algorithm was run in steady mode and did not require a time step size. In the unstructured grid simulations, the SIMPLER algorithm was also tested in steady mode, but was quicker to converge in unsteady mode a with time step size of $\Delta t = 0.10$ seconds. The runtime for the minimum and maximum wind speeds tested are both given in Table 6.3 to demonstrate the low sensitivity to wind speed of IRK-SIMPLER compared to the other algorithms. The SIMPLER and RK-SIMPLER algorithms required lower time step sizes when the wind speed is increased from 7 m/s to 23 m/s, but IRK-SIMPLER is able to take the same time step size and converge for the entire range of wind speeds.

Table 6.2: Structured steady rotor runtime.

Algorithm	Time Step (s)	Runtime	Speedup
SIMPLER	N.A.	3.38	1.0
RK-SIMPLER	0.03	1.04	3.3
IRK-SIMPLER(2)	0.20	1.48	2.3

Table 6.3: Unstructured steady rotor runtime.

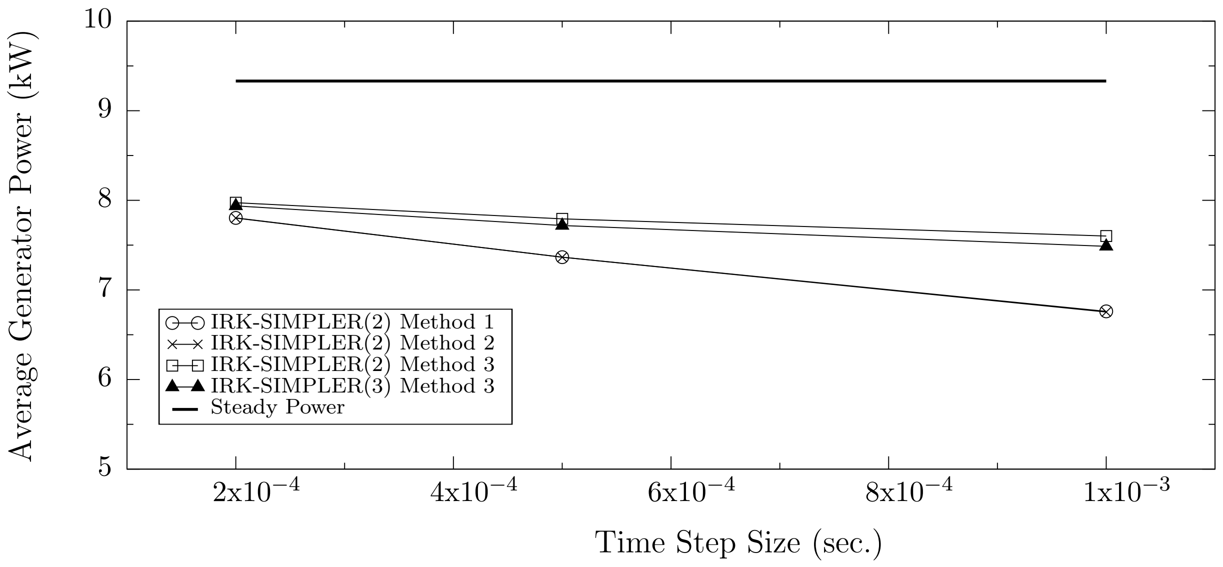
Algorithm	Time Step (s)	Runtime	Speedup
	Wind Speed = 7 m/s		
SIMPLER	0.10	1.01	1.0
RK-SIMPLER	0.005	0.57	1.8
IRK-SIMPLER(2)	0.10	0.22	4.6
	Wind Speed = 23 m/s		
SIMPLER	0.08	0.38	1.0
RK-SIMPLER	0.002	0.43	0.9
IRK-SIMPLER(2)	0.10	0.11	3.5

6.2.2 Unsteady Simulations

The unsteady rotor model is also used to simulate the NREL rotor with same problem specifications. The three different unsteady rotor methods are first tested on the same grid as the steady rotor, with a wind speed of 10.5 m/s and no tip correction. The time step size varies and the simulations run for 14 seconds of simulation time. The average generator power is computed starting at four seconds of simulation time.

Figure 6.11 shows the average generator power for these simulations on the unstructured grid for the IRK-SIMPLER and RK-SIMPLER algorithms, respectively, for the three unsteady rotor methods. For the IRK-SIMPLER algorithm with the 2 stage DIRK method, IRK-SIMPLER(2), methods 1 and 2 give similar results, and approach a value of 8 kW as the time step size decreases. For method 3, a value of 8 kW is also approached, but does not change as much with time step size and is closer to 8 kW at the highest time step tested. For method 3, where new source is found each stage, the three stage IRK-SIMPLER(3) is shown and results in similar but slightly lower power. For RK-SIMPLER, all three methods are similar and farther from the value of 8 kW than all IRK-SIMPLER methods at each time step. Comparing results, the IRK-SIMPLER(2) algorithm with unsteady rotor method 1 is the most accurate, followed by IRK-SIMPLER(3) method 3, IRK-SIMPLER(2) methods 1 and 2, while the least accurate method is RK-SIMPLER using any method.

The SIMPLER algorithm with Crank-Nicolson time integration and unsteady rotor method 2 is the baseline algorithm. Tables 6.4 and 6.5 show the time step required to achieve an accurate, unsteady generator power (set as $7.5kw < P_{gen} < 8.5kW$) for structured and unstructured grid results respectively (where IRK-SIMPLER uses unsteady rotor method 3 and RK-SIMPLER uses method 2). On the structured grid, IRK-SIMPLER(3) has the advantage of being able to take the highest time step size of 0.01 seconds and give accurate results, while the same method on the unstructured grid does not converge for the time step size. The highest speedup for the structured grid is IRK-SIMPLER(3), and for the unstructured grid is IRK-SIMPLER(2).



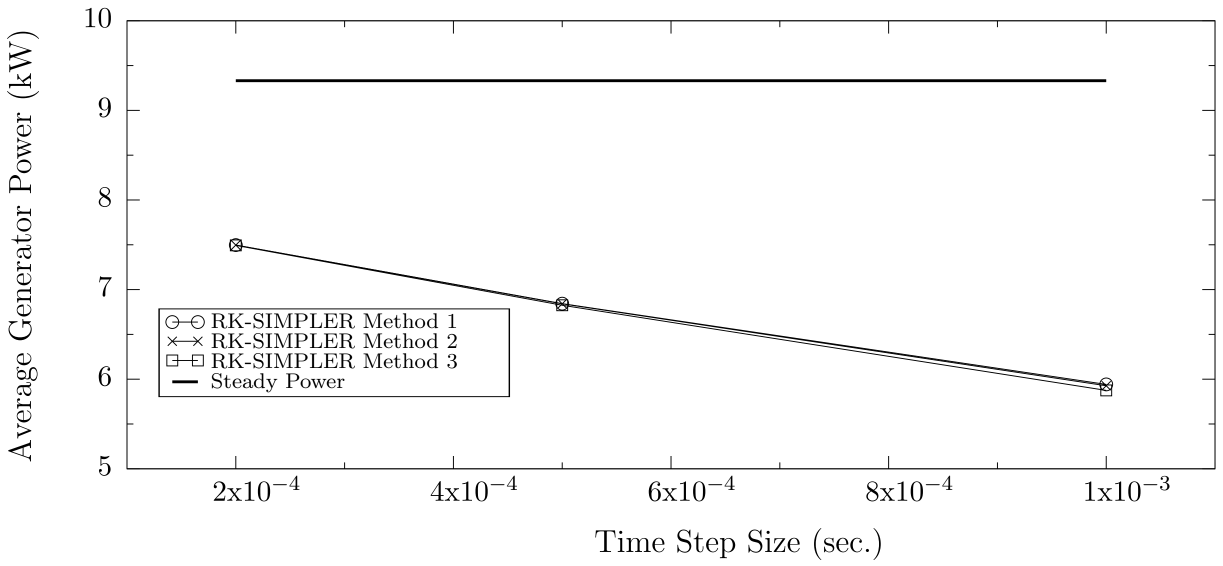


Figure 6.11: Average unsteady rotor power production for V=10.5 m/s (number of stages in parenthesis).

Table 6.4: Structured unsteady rotor runtime.

Algorithm	Time Step (s)	Runtime	Speedup
SIMPLER	0.0001	550.0	1.0
RK-SIMPLER	0.0001	16.6	33.1
IRK-SIMPLER(2)	0.001	6.75	81.5
IRK-SIMPLER(3)	0.01	1.23	447.2

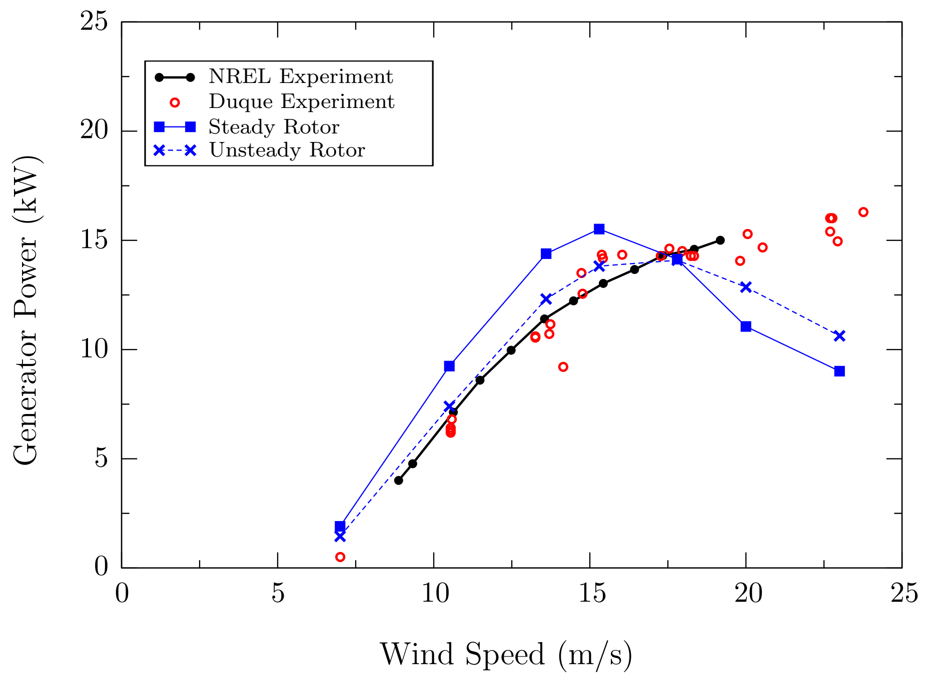
Table 6.5: Unstructured unsteady rotor runtime.

Algorithm	Time Step (s)	Runtime	Speedup
SIMPLER	0.0001	926.0	1.0
RK-SIMPLER	0.0001	64.3	14.4
IRK-SIMPLER(2)	0.001	19.9	46.5
IRK-SIMPLER(3)	0.001	29.9	31.0

Using the unstructured IRK-SIMPLER algorithm with two stage DIRK, the average generator power versus wind speed is computed with and without tip correction, and is shown in Fig. 6.12. The unsteady power falls lower than the steady power in the region before stall (about 15 m/s) and closer to the experimental power. In the region after stall, the unsteady power drops off less dramatically than the steady power.

6.2.3 Tip Vortex Capturing

The wake caused by a wind turbine exists far downstream and influences other turbines. Turbine forces and power can be accurately predicted on relatively coarse grids, with grid refinement just around the rotor and in the near wake, but to accurately capture the wake, more refined grids in the far wake region are required. For a large array of turbines, the grid becomes large and increases the runtime. Flux schemes that accurately capture the wake and tip vortex of a turbine on coarser grids allows for faster simulation. In Chapter 4 different flux schemes were investigated.



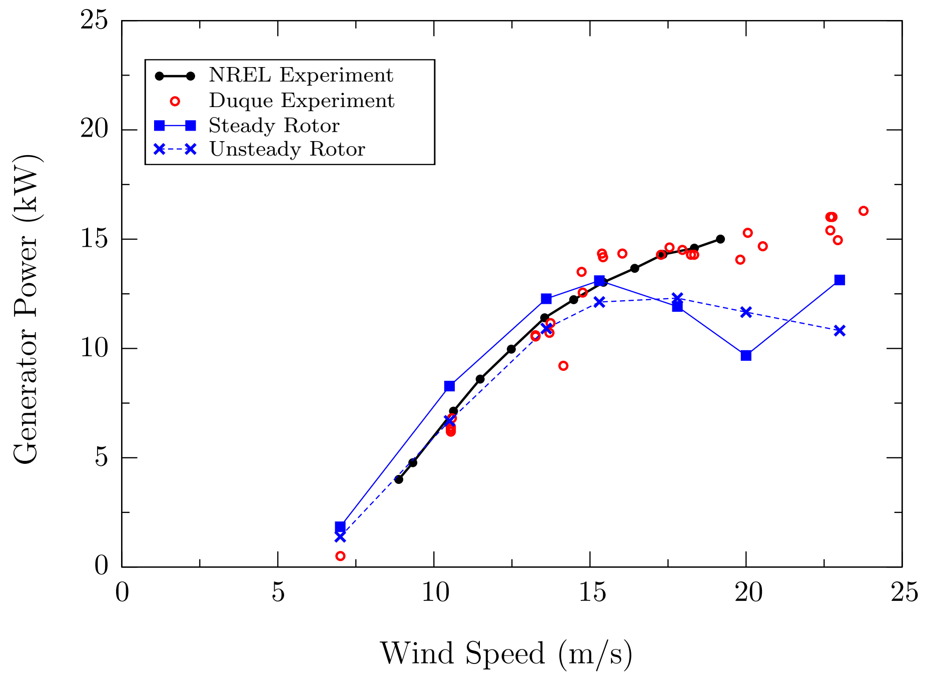
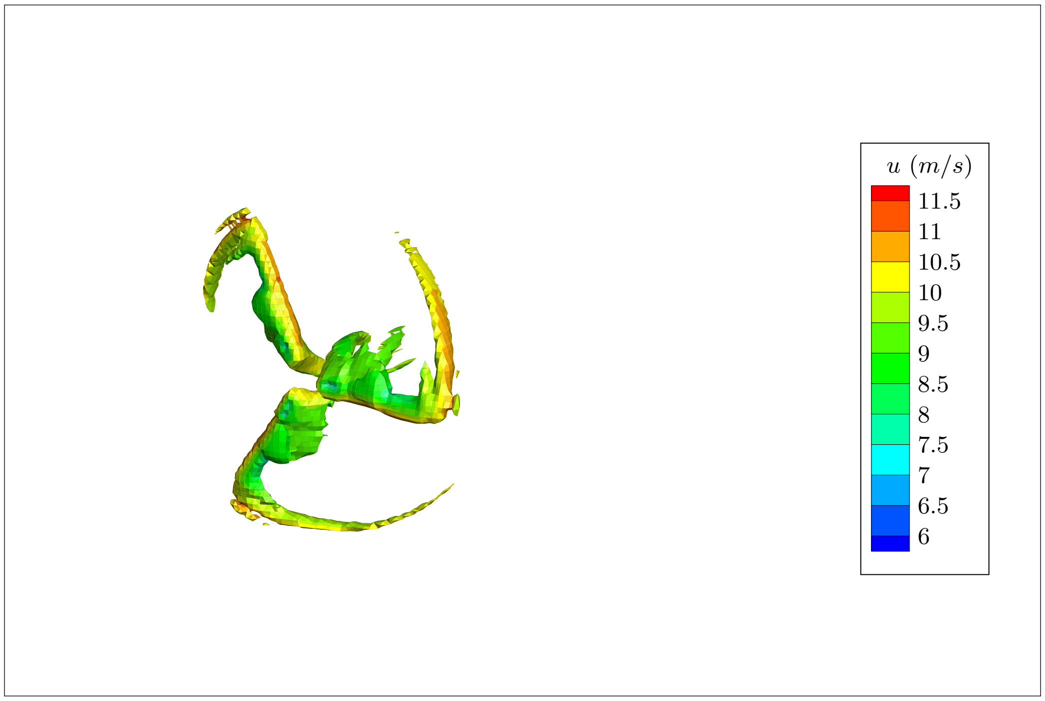


Figure 6.12: Unsteady power production with (bottom) and without (top) tip correction.

For the present test case of the unsteady NREL rotor, the structured grid simulations use the Power Law and QUICK schemes, and the unstructured grid simulations use the Power Law and FCM schemes. For a wind speed of 10.5 m/s, Fig. 6.13 shows the tip vortex in the wake on the same structured grid for the Power Law and QUICK schemes. The plots show iso-surfaces of vorticity magnitude and the colors are u velocity contours. The Power Law scheme solution shows the tip vortex dissipating almost immediately, while the QUICK scheme shows the tip vortex convecting downstream some distance before dissipating. The QUICK scheme is more qualitatively representative of the real tip vortex, which can exist far downstream of the turbine (see examples in Churchfield [47]).

The same case is simulated on the unstructured grid with Power Law and FCM schemes and plotted in Fig. 6.14. Both schemes show similar results which more closely match the structured Power Law results. Unfortunately, the improvement from Power Law to QUICK is not also seen from Power Law to FCM. Perhaps the advantage of QUICK is the larger stencil that may help to more accurately resolve the tip vortex further downstream. These type of large stencil methods are not easily possible in unstructured grid, but what is possible and can improve the results is adaptive grid refinement which automatically refines the grid based on flow gradients and properties.



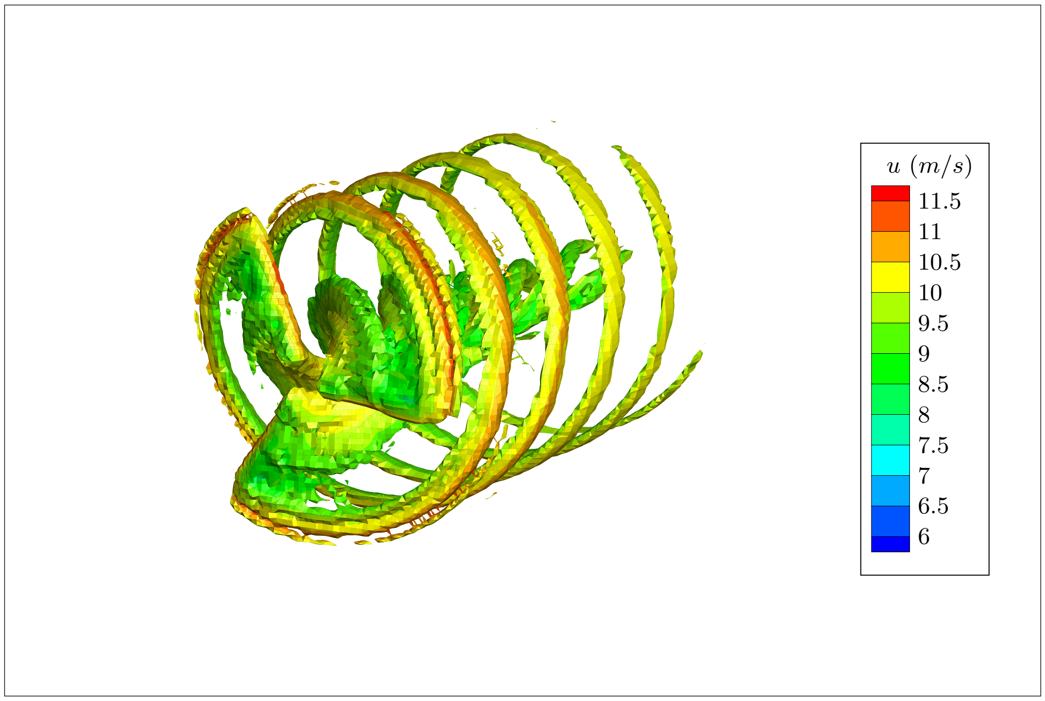
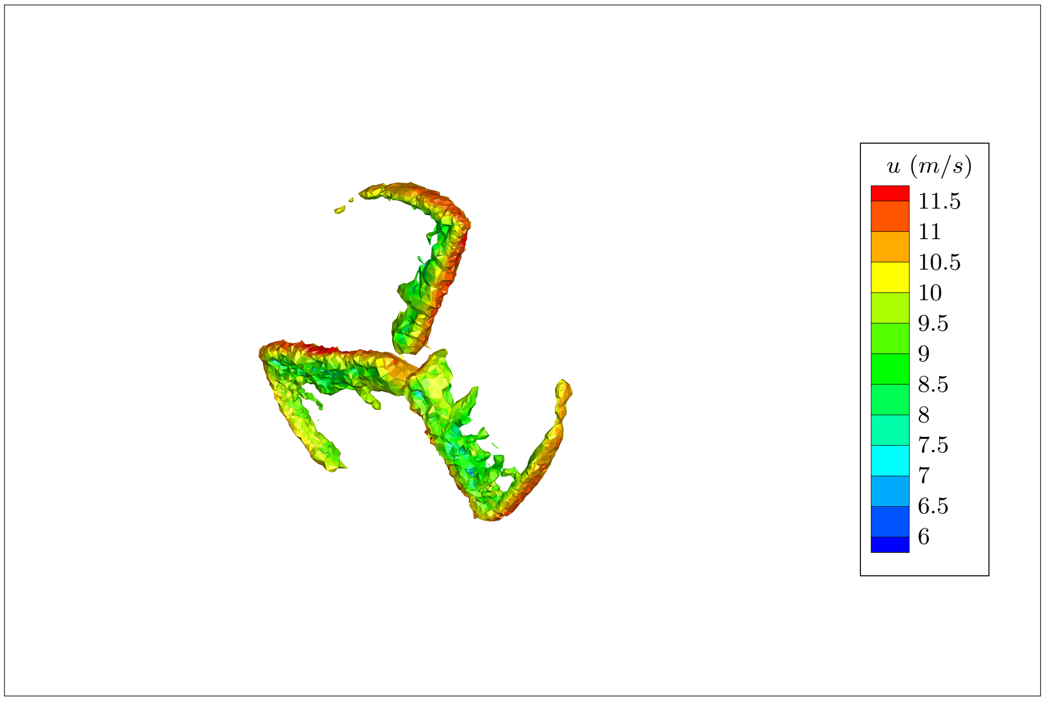


Figure 6.13: Unsteady wake using the structured PL (top) and QUICK (bottom) schemes.



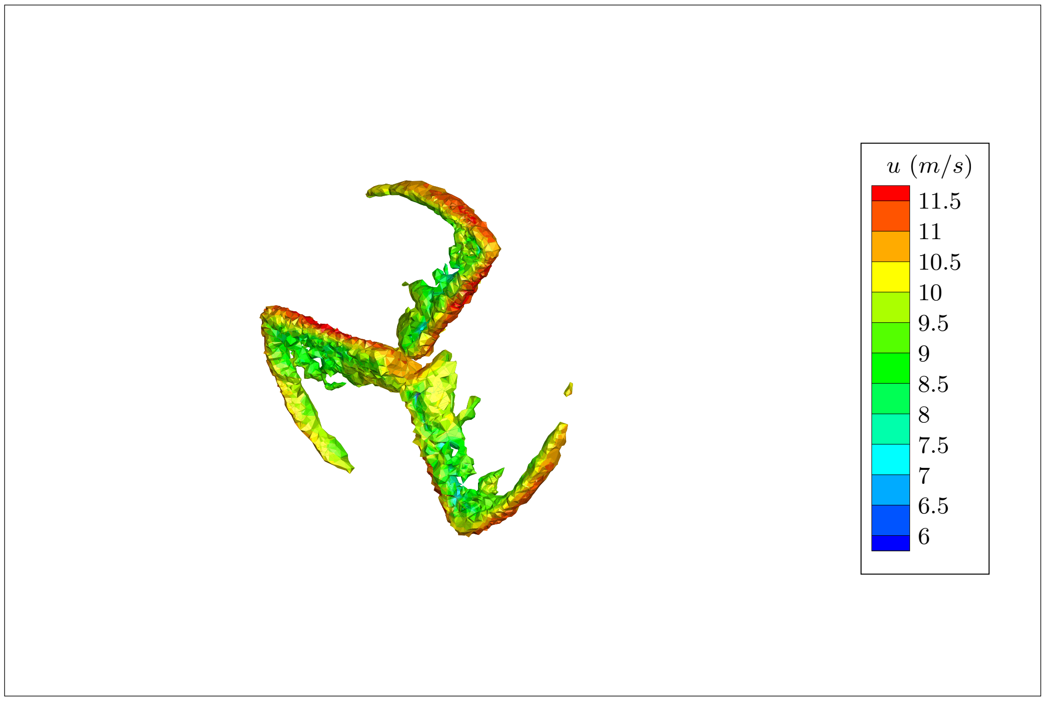


Figure 6.14: Unsteady wake using the unstructured PL (top) and FCM (bottom) schemes.

6.2.4 Turbulent Wake Capturing

Wind turbines and their wakes are impacted by turbulence, both atmospheric and created in the turbine wakes. To accurately simulate turbines, the realizable $k - \epsilon$ model is used to simulate the turbulent flow. The turbine used to test the wake profile with turbulence is the downwind Nibe wind turbine [96]. The Nibe turbine specifications are given in Tables 6.6 and 6.7.

Table 6.6: Nibe rotor specifications.

Diameter	40 m
Number of Blades	3
Rotation Speed	34 RPM
Hub Radius (r/R)	0.10
Blade Pitch	2^o
Cone Angle	6.0^o
Twist (constant)	2^o
Airfoil (constant)	NACA 4412

Table 6.7: Nibe rotor chord distribution (constant taper).

r/R	chord$/R$
0.1	0.116667
1.0	0.027778

The Nibe turbine is simulated without tower and nacelle and in a turbulent boundary layer defined by

$$u(z) = u_{ref} \frac{\ln(z/z_o)}{\ln(z_{ref}/z_o)} \qquad k = \frac{(u^*)^2}{\sqrt{C_\mu}} \qquad \epsilon(z) = \frac{(u^*)^3}{\kappa z} \ , \tag{6.34}$$

where z is the distance above the ground, z_{ref} is the reference height (hub height= 45 m), z_o is the roughness height (0.01 m), u_{ref} is the velocity at hub height (13 m/s), $C_\mu = 0.09$, and $u^* = \kappa u_{ref} / \ln(z_{ref}/z_o)$. These problem specifications follow Murali [44].

The realizable $k - \epsilon$ RANS model simulates the turbulent flow, as discussed in Chapter 5, for the unstructured vertex-based tetrahedral grid. For the windspeed of 13 m/s, the steady rotor model is used with and without the $k - \epsilon$ model with the same boundary layer profiles given for u. Figures 6.15 and 6.16 show the wake profiles for laminar and turbulent simulation, respectively, at four different x/R locations (2.5, 4, 6, and 7.5) downwind of the turbine. Results are compared to the experimental values of Taylor [96] and computation results of Murali [44] (which used the realizable $k - \epsilon$ model on a structured grid with the same steady rotor momentum source model).

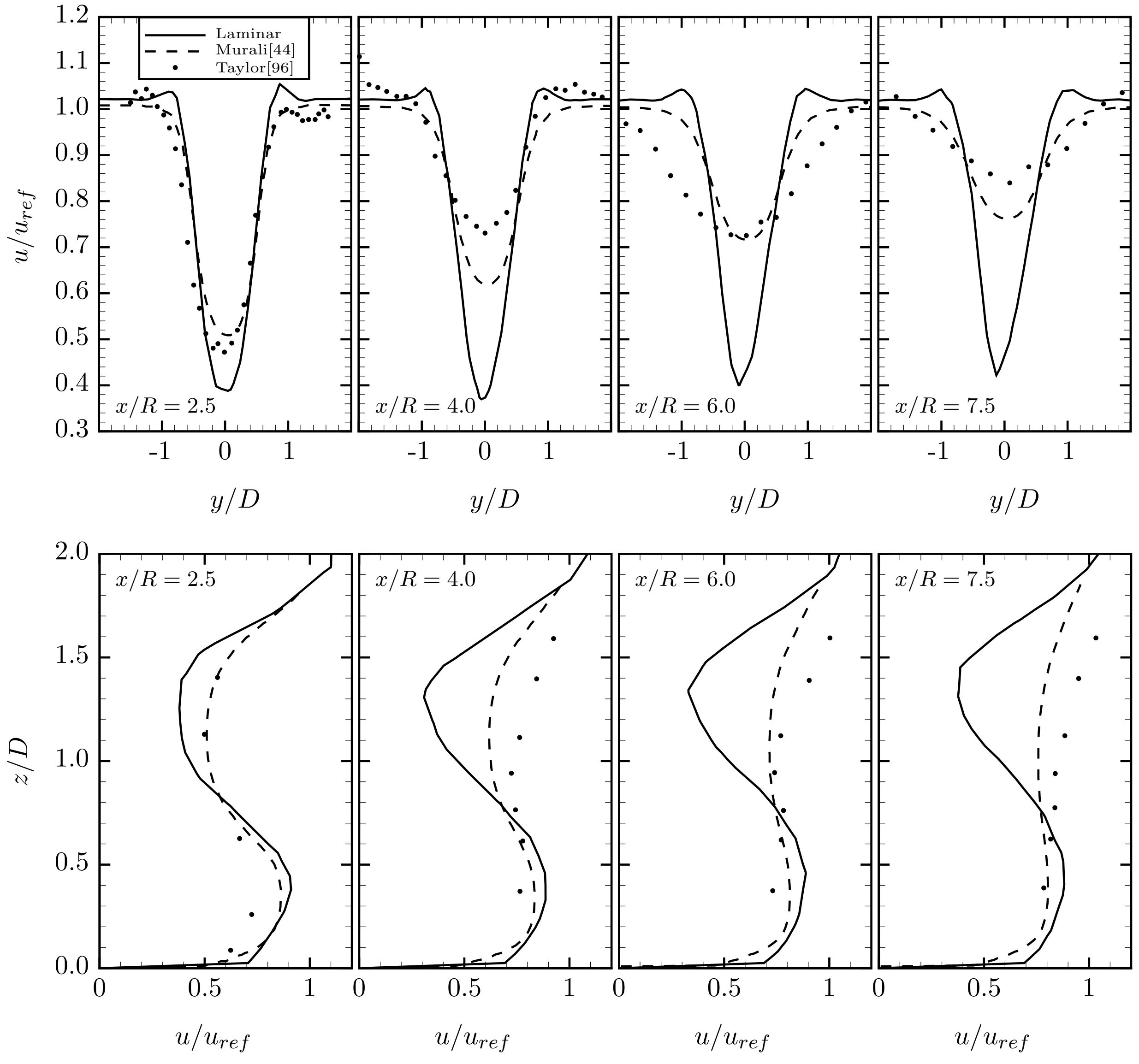


Figure 6.15: Wake profiles at four downwind locations without turbulence modeling (top: lateral profiles at $z = 0$, bottom: vertical profiles at $y = 0$).

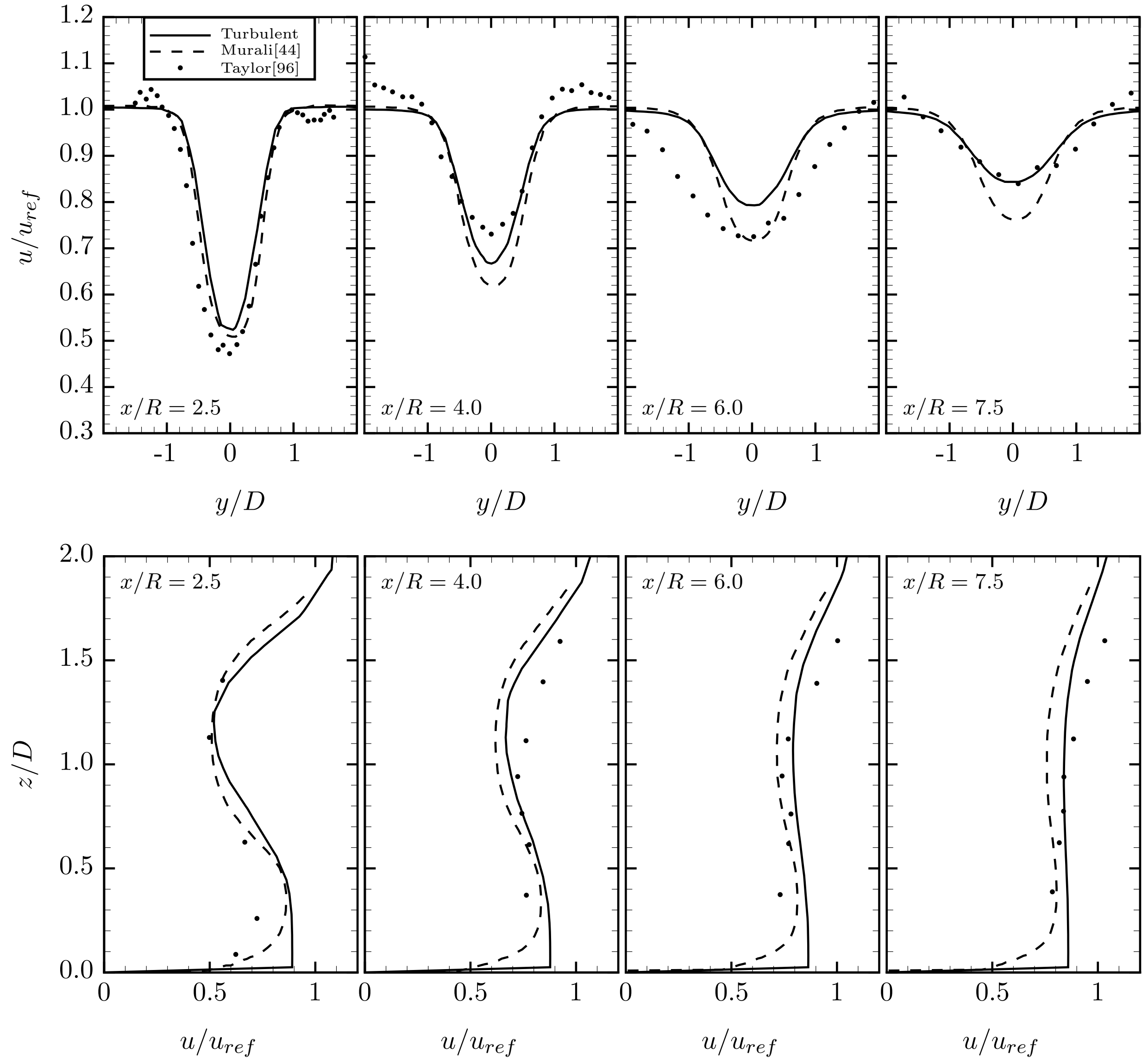


Figure 6.16: Wake profiles at four downwind locations with the realizable $k-\epsilon$ turbulence model (top: lateral profiles at $z=0$, bottom: vertical profiles at $y=0$).

These results show that without the turbulence modeling, the wake profile maintains a similar profile far downstream without much dissipation. With the realizable $k-\epsilon$ model the wake profiles dissipate due to turbulent mixing in the wake region. The wake profiles match both experimental and computational results well along $z/D = 1$ and moderately well along $y/D = 0$. Near the ground, profiles differ significantly due to the coarse grid used near the ground. Grid refinement near the ground and in the wake region would likely better match the computational results of Murali (also using the realizable $k-\epsilon$ model); however, refinement near the ground dramatically increases the number of nodes and runtime. Using tetrahedral elements in the ground boundary layer is not efficient and is not recommended. Other element types should be investigated, such as hexahedron and prisms.

6.3 Conclusions

The IRK-SIMPLER algorithm reduces the runtime to simulate the flow around a wind turbine with momentum source modeling. Using the steady rotor model, IRK-SIMPLER predicted power that matches the legacy SIMPLER algorithm and explicit RK-SIMPLER algorithm, while requiring between 2.3 and 4.6 less runtime than SIMPLER.

Three unsteady rotor models are tested, with a new formulation in method 3 allowing IRK-SIMPLER to achieve accurate results at a higher time step than the two other methods. Using the new unsteady rotor model with IRK-SIMPLER allows up to 447.2 times less runtime than SIMPLER (with the original rotor model) to achieve accurate time averaged power for the structured grid, and 46.5 time less runtime for the unstructured grid. The unsteady rotor model predicts lower power than the steady rotor model and more closely matches experimental power.

The QUICK scheme was most capable of capturing the tip vortex in the wake of the unsteady rotor model on a relatively coarse grid. On unstructured grids, the schemes tested (Power Law and FCM) were both unable to accurately capture the tip vortex and require grid refinement to do so.

The wake profiles downstream are dramatically different between laminar and turbulent simulation, with turbulent modeling matching experimental data better. The turbulent modeling with the unstructured grid improves results over the laminar simulation on the same grid, but further refinement and study is suggested. Also, the use of purely tetrahedral grids of approximately isometric size (which is desirable for the median-dual formulation used) does not allow for efficient grids in a three-dimensional boundary layer.

CHAPTER 7. CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE WORK

The IRK-SIMPLER algorithm proves to be an efficient and accurate algorithm for steady and unsteady flows in two- and three-dimensions. Both structured Cartesian grids and unstructured grids are tested, and IRK-SIMPLER is the most efficient algorithm for all cases. IRK-SIMPLER is capable of up to third order accuracy in time, but (with the exception of the unsteady rotor) the two stage, second order method gives accurate results in less runtime.

The IRK-SIMPLER algorithm is used to test several different flux schemes and is an accurate and stable algorithm for all methods. IRK-SIMPLER is also tested with RANS turbulence modeling. Finally, the IRK-SIMPLER algorithm is tested with wind turbines using the momentum source model. For the wind turbine case, IRK-SIMPLER is faster achieve accurate results than SIMPLER or RK-SIMPLER.

The unstructured FCM scheme provides more accurate results than the unstructured Power Law scheme with very little extra runtime. The end result is a scheme that can give accurate results using a coarser grid and in less runtime. The FCM scheme accurately calculates momentum flux as well as the turbulent k and ϵ fluxes.

The RANS equations are solved using $k-\epsilon$ models (standard and realizable) on unstructured, vertex-centered grids using wall functions. The results for both steady and unsteady cases compare well to experimental and computational results. A disadvantage of unstructured grids for turbulent problems is the lack of control of the grid spacing normal to walls. For more complex body shapes, another form of grid besides purely triangles or tetrahedral cells is recommended.

Wind turbines are simulated using a momentum source model on both structured, Cartesian and tetrahedral, unstructured grids. The power production for both grids match experiments well in the region before stall. The unsteady rotor is modeled with three different intersection and source calculation methods. The method that uses the time integration method (and particularly the DIRK methods) to find the intersections and sources provides the best convergence with time step size. The unsteady rotor model power prediction matches the experiments better than the steady model.

This result opens the question of why the unsteady rotor results appear more accurate. Is the model able to capture the physics of the unsteady rotor better and is more accurate? Refining the grid (or developing more accurate spatial methods) for both the steady and unsteady rotor models (and reducing time step size for the unsteady model) will help gain more insight.

The rotor model applies the sources completely into each grid cell being intersected. Future work should test and compare the methods using Gaussian and elliptic distributions of the source.

Using the QUICK scheme on structured grids allows the tip vortex coming off of a wind turbine to be more accurately captured downstream, while the Power Law scheme in structured and unstructured and the unstructured FCM scheme results in a tip vortex that dissipated immediately. Researching adaptive grid refinement or higher order tetrahedral elements is recommended to improve the unstructured capability.

Another open issue is the pressure oscillations seen for small time step sizes on the vertex-centered unstructured grids. Other researchers find this problem and develop methods to reduce or remove the effects [61],[62],[63]. This line of research should be investigated and applied to the current methods.

BIBLIOGRAPHY

[1] Hairer, E., Lubich, C., and Roche, M., *The Numerical Solution of Differential-Algebraic Systems by Runge-Kutta Methods*, Springer, Berlin, 1989.

[2] Chorin, A. J., "A Numerical Method for Solving Incompressible Viscous Flow Problems," *Journal of Computational Physics*, Vol. 2, 1967, pp. 12–26.

[3] Tannehill, J. C., Anderson, D. A., and Pletcher, R. H., *Computational Fluid Mechanics and Heat Transfer*, Taylor and Francis, Philadelphia, 1997.

[4] Chorin, A. J., "Numerical Solution of the Navier-Stokes Equations," *Mathematics of Computation*, Vol. 22, 1968, pp. 745–762.

[5] Patankar, S. V., *Numerical Heat Transfer and Fluid Flow*, Hemisphere Publishing, Washington, 1980.

[6] Rajagopalan, R. G. and Lestari, A. D., "RK-SIMPLER: An Explicit Time Accurate Algorithm for Incompressible FLows," *AIAA Journal*, Vol. 54, No. 2, 2016, pp. 616–624.

[7] Issa, R. I., "Solution of the implicitly discretised fluid flow equations by operator-splitting," *Journal of Computational Physics*, Vol. 62, No. 1, 1986, pp. 40–65.

[8] Tao, W. Q., Qu, Z. G., and He, Y. L., "A Novel Segregated Algorithm for Incompressible Fluid Flow and Heat Transfer Problems – CLEAR (Coupled and Linked Equations Algorithm Revised) Part I: Mathematical Formulation and Solution Procedure," *Numerical Heat Transfer, Part B: Fundamentals*, Vol. 45, No. 1, 2004, pp. 1–17.

[9] Sun, D. L., Qu, Z. G., He, Y. L., and Tao, W. Q., "An Efficient Segregated Algorithm for Incompressible Fluid Flow and Heat Transfer Problems – IDEAL (Inner Doubly Iterative

Efficient Algorithm for Linked Equations) Part I: Mathematical Formulation and Solution Procedure," *Numerical Heat Transfer, Part B: Fundamentals*, Vol. 53, No. 1, 2008, pp. 1–17.

[10] Jameson, A., Schmidt, W., and Turkel, E., "Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes," *AIAA 14th Fluid and Plasma Dynamic Conference*, AIAA Paper 1981-1259, June 1981.

[11] Jameson, A. and Baker, T. J., "Solution of the Euler equations for complex problems," *AIAA Journal*, Vol. 1929, 1983, pp. 293–302.

[12] Kennedy, C. A., Carpenter, M. H., and Lewis, R. M., "Low-storage, explicit Runge-Kutta schemes for the compressible Navier-Stokes equations," *Applied Numerical Mathematics*, Vol. 35, 2000, pp. 177–219.

[13] Bijl, H., Carpenter, M. H., Vatsa, V. N., and Kennedy, C. A., "Implicit Time Integration Schemes for the Unsteady Compressible Navier-Stokes Equations: Laminar Flow," *Journal of Computational Physics*, Vol. 179, 2002, pp. 313–329.

[14] Jothiprasad, G., Mavriplis, D. J., and Caughey, D. A., "Higher-order time integration schemes for the unsteady Navier-Stokes equations on unstructured meshes," *Journal of Computational Physics*, Vol. 191, 2003, pp. 542–566.

[15] Ijaz, M., *Implicit Runge-Kutta methods to simulate unsteady incompressible flows*, Master's thesis, Texas A&M University, College Station, TX, 2007.

[16] Le, H. and Moin, P., "An Improvement of Fractional Step Methods for the Incompressible Navier-Stokes Equations," *Journal of Computational Physics*, Vol. 92, 1991, pp. 369–379.

[17] Kampanis, N. A. and Ekaterinaris, J. A., "A Staggered Grid, High-Order Accurate Method for the Incompressible Navier-Stokes Equations," *Journal of Computational Physics*, Vol. 215, 2006, pp. 589–613.

[18] Cabuk, H., Sung, C. H., and Modi, V., “Explicit Runge-Kutta Method for Three-Dimensional Internal Incompressible Flows,” *AIAA Journal*, Vol. 30, No. 8, 1992, pp. 2024–2031.

[19] Pereira, J. M. C., Kobayashi, M. H., and Pereira, J. C. F., “A Fourth-Order Accurate Finite Volume Compact Method for the Incompressible Navier-Stokes Solutions,” *Journal of Computational Physics*, Vol. 167, 2001, pp. 217–243.

[20] Morinishi, Y., Lund, T. S., Vasilyev, O. V., and Moin, P., “Fully Conservative Higher Order Finite Difference Schemes for Incompressible Flow,” *Journal of Computational Physics*, Vol. 143, 1998, pp. 90–124.

[21] Nikitin, N., “Third-Order-Accurate Semi-Implicit Runge-Kutta Scheme for Incompressible Navier-Stokes Equations,” *International Journal for Numerical Methods in Fluids*, Vol. 51, 2006, pp. 221–233.

[22] Sanderse, B. and Koren, B., “Accuracy analysis of explicit Runge-Kutta methods applied to the incompressible Navier-Stokes equations,” *Journal of Computational Physics*, Vol. 231, 2012, pp. 3041–3063.

[23] Bashforth, F. and Adams, J. C., *An Attempt to Test the Theories of Capillary Action by Comparing the Theoretical and Measured Forms of Drops of Fluid, with an Explanation of the Method of Integration Employed in Contructing the Tables which Give the Theoretical Forms of Such Drops*, Cambridge, Cambridge University Press, 1883.

[24] Butcher, J. C., *Numerical Methods for Ordinary Differential Equations*, Wiley, West Sussex, UK, 2003.

[25] Price, H. S., Varga, R., and Warrent, J., “Applications of oscillation matrices to diffusion-correction equations,” *Journal of Mathematics and Physics*, Vol. 45, 1966, pp. 301–311.

[26] Leonard, B. P., “A stable and accurate convective modeling procedure based on quadratic upstream interpolation,” *Computer Methods in Applied Mechanics and Engineering*, Vol. 19, 1979, pp. 59–98.

[27] Harten, A., Engquist, B., Osher, S., and Chakravarthy, S., “Uniformly high order accurate essentially non-oscillatory schemes III,” *Journal of Computational Physics*, Vol. 49, 1983, pp. 231–303.

[28] Luo, X., Osher, S., and T., C., “Weighted Essentially Non-Oscillatory Schemes,” *Journal of Computational Physics*, Vol. 115, 1994, pp. 200–212.

[29] Wirogo, S., “Flux Corrected Method: An Accurate Approach to Fluid Flow Modeling,” *13th AIAA CFD Conference*, June 1997.

[30] Baliga, B. and Patankar, S., “A new finite-element formulation for convection-diffusion problems,” *Numerical Heat Transfer*, Vol. 3, 1980, pp. 393–409.

[31] Prakash, C. and Patankar, S., “A control volume based finite element method for solving the Navier-Stokes equations using equal-order velocity-pressure formulation,” *Numerical Heat Transfer*, Vol. 8, 1985, pp. 259–280.

[32] Spalart, P. R. and Allmaras, S. R., “A One-Equation Turbulence Model for Aerodynamic Flows,” *AIAA Paper 92-0439*, AIAA, 1992.

[33] Baldwin, B. S. and Lomax, H., “Thin Layer Approximation and Algebraic Model for Separated Turbulent Flows,” *AIAA Paper 78-257*, AIAA, 1992.

[34] Launder, B. E. and Sharma, B. I., “Application of the Energy-Dissipation Model of Turbulence to the Calculation of Flow Near a Spinning Disk,” *Letters in Heat and Mass Transfer*, Vol. 1, 1974, pp. 131–138.

[35] Wilcox, D. C., *Turbulence Modeling for CFD*, DCW Industries, 1993.

[36] Durbin, P. A. and Pettersson Reif, B. A., *Statistical Theory and Modeling for Turbulent Flows*, Wiley, West Sussex, UK, 2011.

[37] Spalart, P. R., “Comments on the Feasibility of LES for Wing and on a Hybrid RANS/LES Approach,” *1st ASOSR Conference on DNS/LES*, Arlington, Texas, August 1997.

[38] Foster, N., "Accuracy of High-Order CFD and Overset Interpolations in Finite Volume/Difference Codes," *22nd AIAA Computational Fluid Dynamics Conference*, AIAA Paper 2015-3424, Dallas, Texas, June 2015.

[39] Wang, L. and Persson, P., "High-order Discontinuous Galerkin Simulations on Moving Domains using an ALE Formulation and Local Remeshing with Projections," *22nd AIAA Computational Fluid Dynamics Conference*, AIAA Paper 2015-3424, Dallas, Texas, June 2015.

[40] Rajagopalan, R. G. and Fanucci, J. B., "Finite Difference Model for Vertical Axis Wind Turbines," *Journal of Propulsion and Power*, Vol. 1, No. 6, 1985, pp. 432–436.

[41] Rajagopalan, R. G. and Lim, C. K., "Laminar Flow Analysis of a Rotor in Hover," *Journal of the American Helicopter Society*, Vol. 36, No. 1, 1991, pp. 12–23.

[42] Rajagopalan, R. G. and Mathur, S. R., "Three Dimensional Analysis of a Rotor in Forward Flight," *Journal of the American Helicopter Society*, Vol. 38, No. 3, 1993, pp. 14–25.

[43] Zori, L. A. J. and Rajagopalan, R. G., "Navier-Stokes Calculations of Rotor-Airframe Interaction in Forward Flight," *Journal of the American Helicopter Society*, Vol. 40, No. 2, 1995, pp. 57–67.

[44] Murali, A. and Rajagopalan, R. G., "Numerical Simulation of Multiple Interacting Wind Turbines on a Complex Terrain," *Journal of Wind Engineering and Industrial Aerodynamics*, Vol. 162, 2017, pp. 57–72.

[45] Sørensen, J. N. and Shen, W. Z., "Numerical Modeling of Wind Turbine Wakes," *Journal of Fluids Engineering*, Vol. 124, 2002.

[46] Jha, P. K., Churchfield, M. J., Moriarty, P. J., and Schmitz, S., "Guidelines for Volume Force Distributions Within Actuator Line Modeling of Wind Turbines on Large-Eddy Simulation-Type Grids," *Journal of Solar Energy Engineering*, Vol. 136, No. 3, 2014.

[47] Churchfield, M., Schreck, S., Martinez-Tossas, L. A., Meneveau, C., and Spalart, P. R., *An Advanced Actuator Line Method for Wind Energy Applications and Beyond*, NREL/CP-5000-67611, 2017.

[48] Jameson, A., "Application of Dual Time Stepping to Fully Implicit Runge-Kutta Schemes for Unsteady Flow Calculations," *22nd AIAA Computational Fluid Dynamics Conference*, AIAA Paper 2015-2753, Dallas, Texas, June 2015.

[49] Williamson, J. H., "Low-Storage Runge-Kutta Schemes," *Journal of Computational Physics*, Vol. 35, 1980, pp. 48–56.

[50] Alexander, R., "Diagonally Implicit Runge-Kutta Methods for Stiff O.D.E.'s," *SIAM Journal on Numerical Analysis*, Vol. 14, No. 6, 1977, pp. 1006–1021.

[51] Kvæ rnø, A., Nø rsett, S. P., and Owren, B., "Runge-Kutta Research in Trondheim," *Applied Numerical Mathematics*, Vol. 22, 1996, pp. 263–277.

[52] Purohit, P., *Development of an Explicit Time Accurate Scheme for Incompressible Flows*, Master's thesis, Iowa State University, Ames, IA, 2001.

[53] Lestari, A., *Development of unsteady algorithms for pressure-based unstructured solver for two-dimensional incompressible flows*, Master's thesis, Iowa State University, Ames, IA, 2009.

[54] Pearce, J. A., Qasim, A., Maxwell, T. T., and Paramesawaran, S., "A Computational Study of Coharent Wake Structures Behind 2-D Bluff Bodies," *Journal of Wind and Industrial Aerodynamics*, Vol. 41-44, 1992, pp. 2853–2861.

[55] Lisoski, D. L. A., *Nominally 2-Dimensional Flow About a Normal Flat Plate*, Master's thesis, California Institute of Technology, Pasadena, CA, 1993.

[56] Maresca, M. J., *A pressure-based incompressible unstructured solver for general two-dimensional flows.*, Master's thesis, Iowa State University, Ames, IA, 1995.

[57] Rhie, C. M. and Chow, W. L., "Numerical Study of the Turbulent Flow Past an Airfoil with Trailing Edge Separation," *AIAA Journal*, Vol. 21, No. 11, 1983, pp. 1525–1532.

[58] Choi, S. K., "Note on the use of momentum interpolation method for unsteady flows," *Numerical Heat Transfer, Part A*, Vol. 36, 1999, pp. 545–550.

[59] Guntupalli, K., *Development, validation and verification of the Momentum Source Model for discrete rotor blades*, Master's thesis, Iowa State University, Ames, IA, 2011.

[60] Choi, S., "Note on the use of momentum interpolation method for unsteady flows," *Numerical Heat Transfer, Part A: Application*, Vol. 36, 1999, pp. 545–550.

[61] Zhang, S., Zhao, X., and Bayyuk, S., "Generalized formulations for the Rhie-Chow interpolation," *Journal of Computational Physics*, Vol. 258, 2014, pp. 880–914.

[62] Yu, B., Kawaguchi, Y., Wen-Quan, T., and Ozoe, H., "Checkerboard Pressure Predictions Due to the Underrelaxation Factor and Time Step Size for a Nonstaggered Grid with Momentum Interpolation Method," *Numerical Heat Transfer, Part B: Fundamentals*, Vol. 41, 2002, pp. 85–94.

[63] Shen, W. Z., Michelsen, J. A., and Sorensen, J. N., "Improved Rhie-Chow Interpolation for Unsteady Flow Computations," *AIAA Journal*, Vol. 39, No. 12, 2001, pp. 2406–2409.

[64] Luo, H., Luo, L., Nourgaliev, R., Mousseau, V. A., and Dinh, N., "A reconstructed discontinuous Galerkin method for the compressible Navier-Stokes equations on arbitrary grids," *Journal of Computational Physics*, Vol. 229, 2010, pp. 6961–6978.

[65] Wang, Z. J., Fidkowski, K., Abgrall, R., Bassi, F., Caraeni, D., Cary, A., Deconinck, H., Hartmann, R., Hillewaert, K., Huynh, H. T., Kroll, N., May, G., Persson, P., van Leer, B., and Visball, M., "High-order CFD methods: current status and perspective," *International Journal for Numerical Methods in Fluids*, Vol. 72, 2013, pp. 811–845.

[66] Rajagopalan, R. G. and Yu, C., "Use of Lagrange Interpolation in Modeling Convective Kinematics," *Numerical Heat Transfer, Part B: Fundamentals*, Vol. 36, No. 2, 1999, pp. 233–240.

[67] Saha, A. K., Biswas, G., and Muralidhar, K., "Three-Dimensional Study of Flow Past a Square Cylinder at Low Reynolds Numbers," *International Journal of Heat and Fluid Flow*, Vol. 24, 2003, pp. 54–66.

[68] Rajani, B. N., Kandasamy, A., and Majumdar, S., "Numerical Simulation of Laminar Flow Past a Circular Cylinder," *Applied Mathematical Modeling*, Vol. 33, 2009, pp. 1228–1247.

[69] Mittal, R. and Balachandar, S., "On the inclusion of three-dimensional effects in simulation of two-dimensional bluff-body wake flows," *ASME Fluids Engineering Division SUmmer Meeting*, 1997.

[70] Kianifar, A. and Rad, E. Y., "Numerical Simulation of Unsteady Flow with Vortex Shedding Around Circular Cylinder," *Latest Trends on Theoretical and Applied Mechanics*, 2010.

[71] Kim, J., Kim, D., and Choi, H., "An Immersed-Boundary Finite-Volume Method for Simultions of Flow in Complex Geometries," *Journal of Computational Physics*, Vol. 171, 2001, pp. 132–150.

[72] Nowell, P., "Mapping a Cube to a Sphere," mathproofs.blogspot.co.uk/2005/07/mapping-cube-to-sphere.html, July 2005.

[73] Constantinescu, G. S. and Squires, K. D., "LES and DES Investigations of Turbulent Flow over a Sphere," *AIAA Paper 2000-0540*, AIAA, 2000.

[74] Johnson, T. A. and Patel, V. C., "Flow past a sphere up to a Reynolds Number of 300," *Journal of Fluid Mechanics*, Vol. 378, 1999, pp. 19–70.

[75] Fornberg, B., "Steady viscous flow past a sphere at high Reynolds numbers," *Journal of Fluid Mechanics*, Vol. 190, 1988, pp. 471–489.

[76] Jones, W. P. and Launder, B. E., "The prediction of laminarization with a two-equation model," *International Journal of Heat Transfer*, Vol. 15, 1972, pp. 301–314.

[77] Kato, M. and Launder, B. E., "The Modeling of Turbulent Flow Around Stationary and Vibrating Square Cylinders," *Proceedings of the 9th Symposium on Turbulent Shear Flows*, August 1993, pp. 10.4.1–10.4.6.

[78] Shih, T. H., Liou, W. W., Shabbir, A., Yang, Z., and Zhu, J., "A New k-ϵ Eddy Viscosity Model for High Reynolds Number Turbulent Flows - Model Development and Validation," *Computers and Fluids*, Vol. 24, No. 3, 1995, pp. 227–238.

[79] Launder, B. E. and Spalding, D. B., "The Numerical Computation of Turbulent Flows," *Computer Methods in Applied Mechanics and Engineering*, Vol. 3, 1974, pp. 269–289.

[80] Ghosh, S., *Configurational Effect on Dust Cloud Formation and Brownout*, Master's thesis, Iowa State University, Ames, IA, 2010.

[81] Rodi, W., "A New Algebraic Relation for Calculating the Reynolds Stresses," *Journal of Applied Mathematics and Mechanics*, Vol. 56, 1976, pp. T219–T221.

[82] Driver, D. M. and Seegmiller, H. L., "Features of a Reattaching Turbulent SHear Layer in Divergent Channel Flows," *AIAA Journal*, Vol. 23, No. 2, February 1985, pp. 163–171.

[83] Yoder, D. A. and Georgiadis, N. J., *Implementation and Validation of the Chien k-ϵ Turbulence Model in the Wind Navier-Stokes Code*, NASA/TM-1999-209080, 1999.

[84] Shewchuk, J. R., "Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator," *Applied Computational Geometry: Towards Geometric Engineering, volume 1148 of Lecture Notes in Computer Science*, Springer-Verlag, Berlin, May 1996, pp. 203–222.

[85] Iaccarino, G., Ooi, A., Durbin, P. A., and Behnai, M., "Reynolds Averaged Simulation of Unsteady Separated Flow," *International Journal of Heat and Fluid Flow*, Vol. 24, 2003, pp. 147–156.

[86] Lyn, D., Einav, S., Rodi, W., and Park, J., "A Laser-Doppler Velocimetry Study on Ensemble Averaged Characteristics of the Turbulent Near Wake of a Square Cylinder," *Journal of Fluid Mechanics*, Vol. 304, 1995, pp. 205–232.

[87] Lee, B. E., "The Effect of Turbulence on the Surface Pressure Field of a Square Prism," *Journal of Fluid Mechanics*, Vol. 69, 1975, pp. 263–289.

[88] Vickery, B. J., "Fluctuating Lift and Drag on a Long Square Cylinder of Square Cross Section in a Smooth and Turbulent Stream," *Journal of Fluid Mechanics*, Vol. 25, 1966, pp. 481–503.

[89] Rodi, W., Ferziger, J. H., Breuer, M., and Pourquie, M., "Status of Large Eddy Simulation: Result of a Workshop," *Journal of Fluids Engineering*, Vol. 119, 1997, pp. 248–262.

[90] Durbin, P., "Separated Flow Computations with the k-ϵ-v^2 model," *AIAA Journal*, Vol. 33, 1995, pp. 659–664.

[91] Butterfield, C. P., "Aerodynamic Pressure and Flow-Visualization Measurement from a Rotation Wind Turbine Blade," *8th ASME Wind Energy Symposium*, Huston, January 1989.

[92] Duque, E., Johnson, W., van Dam, C., Cortes, R., and Yee, K., "Numerical Predictions of Wind Turbine Power and Aerodynamic Loads for the NREL Phase II Combined Experiment Rotor," *AIAA 38th Aerospace Sciences Meeting*, AIAA-2000-0038, Reno, NV, January 2000.

[93] Laino, D. J. and Hansen, A. C., *User's Guide to the Computer Software Routines AeroDyn Interface for ADAMS*, Windward Engineering, Salt Lake City, Utah, Prepared for the National Renewable Energy Laboratory under Subcontract No. TCX-9-29209-01, September 2001.

[94] Johnson, W., "Rotorcraft Aerodynamics Models for a Comprehensive Analysis," *Presented at the American Helicopter Society Forum*, Washington, D.C., May 1998.

[95] Du, Z. and Selig, M. S., “A 3-D Stall Delay Model for Horizontal Axis Wind Turbines,” *1998 ASME Wind Energy Symposium, Aerospace Sciences Meeting*, AIAA-98-0021, Reno, Nevada, 1998.

[96] Taylor, G. J., Milborrow, D. J., McIntosh, D. N., and Swift-Hook, D. T., “Wake Measurements on the Nibe Windmills,” *7th BWEA Wind Energy Conference*, Oxford, UK, 1985, pp. 67–73.

[97] Fischels, M. V. and Rajagopalan, R. G., “Runge-Kutta Based Algorithms for More Efficient Simulation of Unsteady Incompressible Flows,” *Ninth International Conference on Computational Fluid Dynamics*, ICCFD9-2016-125, Istanbul, Turkey, July 11-15, 2016.

[98] Brandt, A., “Multi-Level Adaptive Solutions to Boundary-Value Problems,” *Mathematics of Computation*, Vol. 31, No. 138, 1977, pp. 333–390.

[99] Briggs, W. L., *A Multigrid Tutorial*, SIAM, Philadelphia, 1987.

[100] Zori, L. A. and Rajagopalan, R. G., “The FAS Multigrid Method for the Segrated Solution of the SIMPLER Algorithm,” *Proceeding of the 6th International Symposium on CFD*, Vol. 3, Tahoe, Nevada, 1995, pp. 1509–1514.

APPENDIX A. INTEGRATION OF UNSTEADY SOURCE TERMS WITH RUNGE-KUTTA METHODS

The Runge-Kutta method integrates an equation with the form of Eq. 2.26, where F is a function of the variable $\phi(t)$, but also is a function of source terms that are unsteady, $b(t)$. As a test case, the following equation is used

$$\frac{d\phi}{dt} = -\phi + \cos(t) \ , \tag{A.1}$$

where $b(t) = \cos(t)$. Using the initial condition $\phi(0) = 1$, the exact solution is $\phi(t) = 0.5[\exp(-t) + \sin(t) + \cos(t)]$. A two stage explicit RK method is used to integrate this equation with coefficients

$$\gamma_2 = 0.5 \qquad \alpha_{2,1} = 0.5 \tag{A.2}$$

$$\beta_1 = 0.0 \qquad \beta_2 = 1.0 \ . \tag{A.3}$$

The integration using these coefficients leads to the stage values and next step value

$$\phi_1 = \phi^n \tag{A.4}$$

$$\phi_2 = \phi^n + 0.5\Delta t\Big[-\phi_1 + \cos(t)\Big] \tag{A.5}$$

$$\phi^{n+1} = \phi^n + \Delta t\Big[-\phi_2 + \cos(t)\Big] \ . \tag{A.6}$$

The unsteady source term $\cos(t)$ is evaluated twice, and the time at which to evaluate it is examined. Three methods are tested. Method 1 uses the time value t^n to evaluate all unsteady source terms

$$\phi_1 = \phi^n \tag{A.7}$$

$$\phi_2 = \phi^n + 0.5\Delta t\Big[-\phi_1 + \cos(t^n)\Big] \tag{A.8}$$

$$\phi^{n+1} = \phi^n + \Delta t\Big[-\phi_2 + \cos(t^n)\Big] \ , \tag{A.9}$$

method 2 uses the time value t^{n+1} to evaluate all source terms

$$\phi_1 = \phi^n \tag{A.10}$$

$$\phi_2 = \phi^n + 0.5\Delta t\Big[-\phi_1 + \cos(t^{n+1})\Big] \tag{A.11}$$

$$\phi^{n+1} = \phi^n + \Delta t\Big[-\phi_2 + \cos(t^{n+1})\Big]\ , \tag{A.12}$$

and method 3 uses the time value for each stage being evaluated

$$\phi_1 = \phi^n \tag{A.13}$$

$$\phi_2 = \phi^n + 0.5\Delta t\Big[-\phi_1 + \cos(t_1)\Big] \tag{A.14}$$

$$= \phi^n + 0.5\Delta t\Big[-\phi_1 + \cos(t^n + \gamma_1\Delta t)\Big] \tag{A.15}$$

$$= \phi^n + 0.5\Delta t\Big[-\phi_1 + \cos(t^n)\Big] \tag{A.16}$$

$$\phi^{n+1} = \phi^n + \Delta t\Big[-\phi_2 + \cos(t_2)\Big] \tag{A.17}$$

$$= \phi^n + \Delta t\Big[-\phi_2 + \cos(t^n + \gamma_2\Delta t)\Big] \tag{A.18}$$

$$= \phi^n + \Delta t\Big[-\phi_2 + \cos(t^n + 0.5\Delta t)\Big]\ . \tag{A.19}$$

Any Runge-Kutta scheme can use this method of evaluating the unsteady sources at each stage time $t_s = t^n + \gamma_s\Delta t$.

The test problem is integrated with these three different methods, all for a time step size of $\Delta t = 0.1$. Figure A.1 shows the resulting $\phi(t)$ for the three methods along with the exact solution. Method 3 falls on top of the exact solution, while methods 1 and 2 are above and below the exact solution with an error that grows with time. This result shows that source terms that are dependent on time are most accurate when evaluated at the stage time (method 3). A key case of this is the unsteady rotor model (Sec. 6.1.4), where the rotor source is moving in time. The most accurate unsteady rotor method uses the stage time to find the rotor intersection and the rotor source.

Within the Runge-Kutta methods, stage values are not necessarily as accurate as the final update value, ϕ^{n+1}. However, Fig. A.2 shows the stage values alongside the final update and exact solution for the same test problem. The stage values are reasonably close to the exact solution.

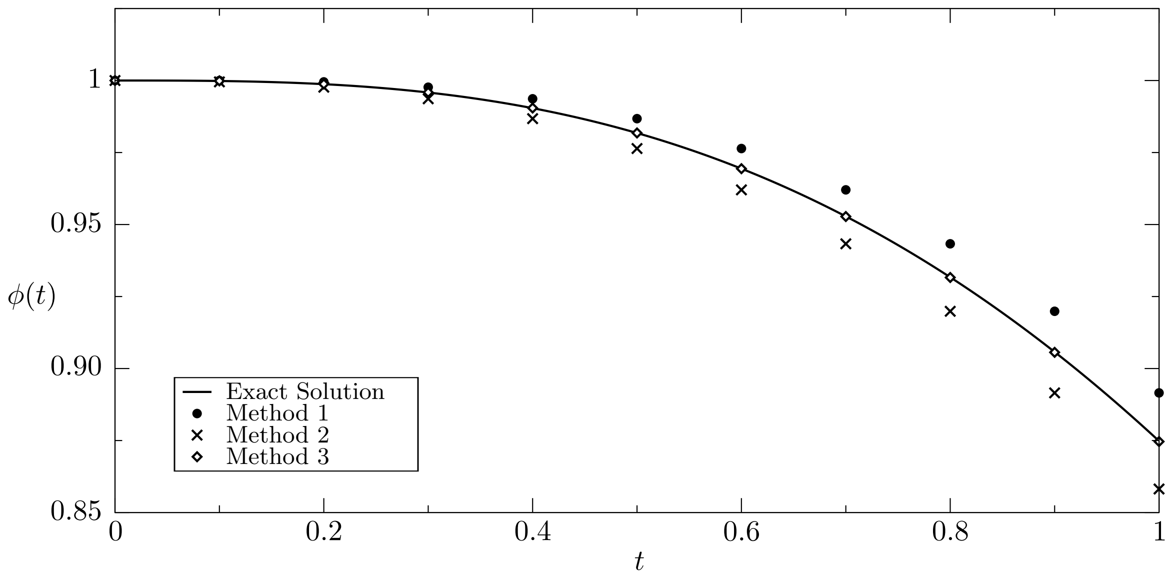


Figure A.1: Solution to the test problem with three different source methods.

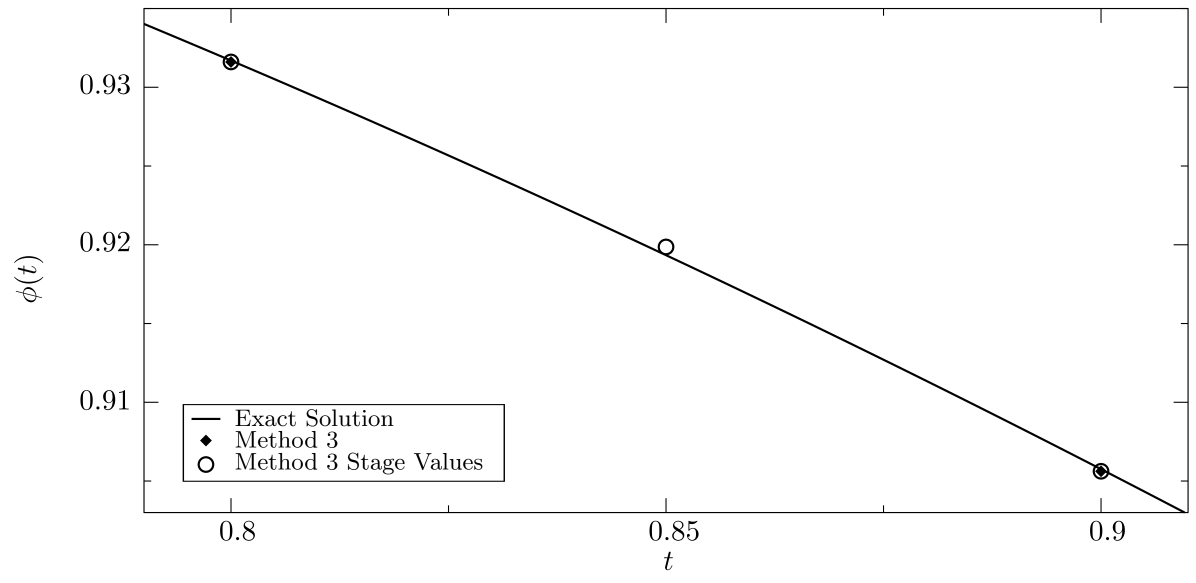


Figure A.2: Solution to the test problem with method 3 and stage values shown.

APPENDIX B. APPROXIMATE AND LU FACTORIZATIONS

The solution of momentum equations using IRK-SIMPLER(A) with ESDIRK methods provides opportunity to use a factorization of the coefficient matrix. For the present research, a combination of approximate factorization and LU factorization are investigated.

Runge-Kutta starts with the discrete momentum equations in the form

$$\frac{d\phi}{dt} = F(\phi, t) \ , \tag{B.1}$$

and for structured grids

$$F(\phi, t) = \frac{a_E\phi_E + a_W\phi_W + a_N\phi_N + a_S\phi_S - a_P\phi_P + b}{\rho\Delta\forall} \ . \tag{B.2}$$

This ODE is integrated from time level n to $n+1$ with a time step size of Δt.

The solution of the equations with EDIRK methods involve S stages to be solved, each with a system of equations. The stage values for ESDIRK methods are

$$\phi_1 = \phi^n \tag{B.3}$$

$$\phi_s = \phi^n + k\Delta t\, F\big(\phi_s, t^n + \gamma_s\Delta t\big) + \Delta t\sum_{l=1}^{s-1}\alpha_{s,l}F\big(\phi_l, t^n + \gamma_l\Delta t\big) \qquad \text{for } 2 \leq s \leq S \ . \tag{B.4}$$

For IRK-SIMPLER(A), the coefficients of the momentum equations are calculated once and held constant for all stages. As each stage is solved, the same coefficient matrix is used at each stage. This means a system of equations is solved at each stage where the coefficients do not change, but only the source term changes. To take advantage of this, a LU factorization and approximate factorization are used.

Approximate factorization splits the x and y derivatives and simplifies the two-dimensional system of equations into two groups of one-dimensional tri-diagonal (tridi) systems, which can be solved with less computational effort. Starting with the discrete momentum equation, new central coefficients are defined for x and y derivatives.

$$\frac{d\phi}{dt} = \frac{(a_E\phi_E + a_W\phi_W - a^x_P\phi_P) + \left(a_N\phi_N + a_S\phi_S - a^y_P\phi_P\right) + b}{\rho\Delta\forall} \tag{B.5}$$

$$a^x_P = a_E + a_W \qquad\qquad a^y_P = a_N + a_S \tag{B.6}$$

Integrating this with ESDIRK methods leads to an equation with the following form for each stage $s > 1$.

$$\phi_s = \phi^n + k\frac{\Delta t}{\rho\Delta\forall}\left[(a_E\phi_E + a_W\phi_W - a^x_P\phi_P) + \left(a_N\phi_N + a_S\phi_S - a^y_P\phi_P\right) + b\right]_s + \tilde{b}_s \ , \tag{B.7}$$

with

$$\tilde{b}_s = \Delta t\sum_{l=1}^{s-1}\alpha_{s,l}F(\phi_l, t^n + \gamma_l\Delta t) \ . \tag{B.8}$$

After defining $\Delta\phi = \phi_s - \phi^n$

$$\begin{aligned}\Delta\phi_P = k\frac{\Delta t}{\rho\Delta\forall}\Big[&(a_E\Delta\phi_E + a_W\Delta\phi_W - a^x_P\Delta\phi_P)\\ &+ \left(a_N\Delta\phi_N + a_S\Delta\phi_S - a^y_P\Delta\phi_P\right)\Big]_i + \tilde{b}_s + k\Delta tF(\phi^n, t^n)\end{aligned} \tag{B.9}$$

By defining two operators f_x and f_y which operate on $\Delta\phi_P$

$$f_x\Delta\phi_P = a_E\Delta\phi_E + a_W\Delta\phi_W - a^x_P\Delta\phi_P \tag{B.10}$$

$$f_y\Delta\phi_P = a_N\Delta\phi_N + a_S\Delta\phi_S - a^y_P\Delta\phi_P \tag{B.11}$$

The following equation is formed.

$$\left[1 - k\frac{\Delta t}{\rho\Delta\forall}f_x - k\frac{\Delta t}{\rho\Delta\forall}f_y\right]\Delta\phi_P = \tilde{b}_s + k\Delta tF(\phi^n, t^n) \tag{B.12}$$

Making an approximate factorization of the form

$$\left[1 - k\frac{\Delta t}{\rho\Delta\forall}f_x - k\frac{\Delta t}{\rho\forall}f_y\right] \approx \left(1 - k\frac{\Delta t}{\rho\Delta\forall}f_x\right)\left(1 - k\frac{\Delta t}{\rho\Delta\forall}f_y\right) \ . \tag{B.13}$$

The error associated with this factorization is $(k\Delta t/\rho\Delta\forall)^2 f_x f_y$. With this factorization, each stage is solved in three steps:

- Step 1: Solve $\Delta\phi_P^*$ from

$$\left(1 - k\frac{\Delta t}{\rho\Delta\forall}f_x\right)\Delta\phi_P^* = \tilde{b}_s + k\Delta t F(\phi^n, t^n)\ . \tag{B.14}$$

- Step 2: Solve $\Delta\phi_P$ from

$$\left(1 - k\frac{\Delta t}{\rho\Delta\forall}f_y\right)\Delta\phi_P = \Delta\phi_P^*\ . \tag{B.15}$$

- Step 3: Find $(\phi_P)_s$

$$(\phi_P)_i = \Delta\phi_P + \phi_P^n\ . \tag{B.16}$$

Step 1 consists of a group of tridi matrix equations, with the unknowns being neighbors in the x-direction. For each grid line of constant y, one tridi matrix equation is solved. Similarly for step 2, for each grid line of constant x, one tridi matrix equation is solved with unknowns being neighbors in the y-direction. Step 3 is simply updating the value of ϕ_s, and requires little computational effort.

Along side the approximate factorization, an LU factorization reduces the computation by taking advantage of the fact that both $\left(1 - k\frac{\Delta t}{\rho\Delta\forall}f_x\right)$ and $\left(1 - k\frac{\Delta t}{\rho\Delta\forall}f_y\right)$ are the same for every stage in a time step. LU factorization is a procedure to decomposed a matrix into lower and upper triangular matrices. For a matrix equation of the form

$$A\vec{x} = \vec{b}\ , \tag{B.17}$$

the LU factorization process involves solving two simpler matrix equations of the form

$$L\vec{y} = \vec{b} \tag{B.18}$$

$$U\vec{x} = \vec{y}\ , \tag{B.19}$$

where $A = LU$. The solution procedure for these two matrix equations is computationally less expensive than solving the original equation.

For the ESDIRK methods, the equivalent A matrices for both step 1 and step 2 are constant for all stages. By using the LU factorization, the L and D matrices for both step 1 and step 2 are found once at the beginning of the time step and used for all stages.

An approximate factorization for three-dimensions is treated in a similar fashion (see [97]).

APPENDIX C. MULTIGRID METHODS FOR IRK-SIMPLER(B)

Multigrid methods allow linear systems of equations to be solved with fewer computations and less runtime than traditional methods [98],[99]. There are a variety of multigrid methods developed for many different problems. For incompressible flow, a multigrid method developed based on the SIMPLER algorithm is called FAS-SIMPLER [100]. A similar multigrid method is developed for unsteady incompressible flow based on the IRK-SIMPLER(B) algorithm to improve the convergence rate at each time step.

In the IRK-SIMPLER algorithm, to reach a converged solution (where all equations have low residual) at each time step, several iterations must be completed. This is where most computations occur in the IRK-SIMPLER algorithm. To speed up the convergence rate, multigrid methods are developed.

The non-linear system of equations to be solved has the form

$$A\mathbf{u} = \mathbf{f} \ . \tag{C.1}$$

When attempting to solve for $\mathbf{u}$, after some iteration an approximate value $\mathbf{v}$ is known. The error is defined as $\mathbf{e} = \mathbf{u} - \mathbf{v}$. This error is not computable without knowing the exact solution for $\mathbf{u}$. A computable measure of how close the equation is to being satisfied is the residual, $\mathbf{r} = \mathbf{f} - A\mathbf{v}$. Equation C.1 is rearranged as

$$A\mathbf{e} = \mathbf{r} \ . \tag{C.2}$$

Using common iterative solution methods like Gauss-Seidel on a single grid quickly removes high frequency errors, but low frequency errors require many iterations to be removed. To remedy this, transferring the problem to coarser grids (called restriction) allows the low frequency errors to be removed much quicker. In FAS multigrid methods the variable being solved for ($\mathbf{u}$) is restricted to coarser grids. This is useful for non-linear problems where the coefficients are dependent on the variables. In non-FAS methods, only the error ($\mathbf{e}$) is restricted to coarser grids. For the non-linear incompressible Navier-Stokes equations, the coefficients are dependent on the variables, so FAS multigrid methods are used in the present research. The notation for the restriction of values from a fine grid k to a coarse grid $k-1$ is $I_k^{k-1}\mathbf{v}^k$. The restriction of a field variable (u, v, or p) in the present research is accomplished by bi-linear interpolation in two dimensions, and tri-linear interpolation in three dimensions. The restriction of the residuals for finite volume methods (which are integrated quantities) requires an addition, not an interpolation, of the fine grid residuals contained in each coarse grid volume.

After some iterations on the fine grid, high frequency errors are removed while low frequency errors still remain. At this point we call the solution $\mathbf{v}_{old}^k$ and calculate the residual $\mathbf{r}^k$. On the coarse grid the error is defined by

$$\mathbf{e}^{k-1} = \mathbf{u}^{k-1} - I_k^{k-1}\mathbf{v}_{old}^k \; . \tag{C.3}$$

Substituting this into Equation C.2 leads to the equation to be solved on the coarse grid $k-1$.

$$A^{k-1}\mathbf{u}^{k-1} = I_k^{k-1}\mathbf{r}^k + A^{k-1}(I_k^{k-1}\mathbf{v}_{old}^k) \tag{C.4}$$

The value of $\mathbf{u}^{k-1}$ is initialized to $I_k^{k-1}\mathbf{v}_{old}^k$ when starting on the next coarser grid level. After some iterations on the coarse grid, the low frequency errors are removed. These values are then restricted to the next coarser grid in the same manner. Once the coarsest grid is reached, the errors are prolonged back up to the finer grids. The notation for prolongation of values from a coarse grid $k-1$ to a fine grid k is $I_{k-1}^k\mathbf{v}^{k-1}$. For the present research, all prolongation is accomplished with bi- or tri-linear interpolation. The correction of fine grid values is found by

$$\mathbf{v}_{new}^k = \mathbf{v}_{old}^k + I_{k-1}^k(\mathbf{v}^{k-1} - I_k^{k-1}\mathbf{v}_{old}^k) \; . \tag{C.5}$$

By prolonging errors to the fine grid, some high frequency errors are introduced, so a few more iterations are typically used to reduce them. The error is then prolonged to the next finer grid until the finest grid level is reached. At this point one multigrid iteration is completed. This form of multigrid iterations is called V-cycle [99], where all grid levels are visited in order from fine to coarse while restricting, and then all grid levels are visited in order from coarse to fine while prolonging. In the presented simulations, the V-cycle is used in the IRK-SIMPLER algorithm with a fixed number of three iterations on each grid level. This fixed number has not been rigorously optimized, and future research could include examining the impact of the number of iterations at each grid level.

Another form of multigrid iterations is called cycle-C [98]. Instead of visiting each grid level in order, cycle-C determines whether to move to a coarser or finer grid by monitoring the residuals. For the present simulations the residual value monitored is the sum of the L2 norm of all the momentum equation residuals, $r = L2(\mathbf{r}_{x-mom}) + L2(\mathbf{r}_{y-mom}) + L2(\mathbf{r}_{z-mom})$. Cycle-C moves up or down grids at any point during iterations as criteria are met.

When determining whether to restrict to a coarser grid, the convergence rate, or the rate that residuals are dropping, is monitored. If the convergence rate is slow on the current grid, the high frequency errors are removed, and the values are restricted to the next coarser grid. Restriction will occur if

$$r^k > \eta r^k_{last} \ , \tag{C.6}$$

where r^k_{last} is the residual value from the last iteration on the current grid level. The value of η is taken from Brandt at 0.6. Whenever moving to a new grid level, the value of r^k_{last} is initialized to a large value, $r^k_{last} \to \infty$.

When determining whether to prolong to a finer gird, the convergence level, or the magnitude of the current residual, is monitored. If the convergence level is below a tolerance for that grid level, its solution is converged and the values will be prolonged to the next finer gird. Prolongation will occur if

$$r^k < r^k_{tol} \ . \tag{C.7}$$

On the finest grid, $k = N$, the value of r^N_{tol} is the overall tolerance level for the problem. If this criteria is met on the finest grid level, then the solution is found. On coarser grid levels, the value of r^k_{tol} is determined by a parameter δ and the next finer grid level residual by

$$r^k_{tol} = \delta r^{k+1}_{last} \ . \tag{C.8}$$

In other words, if the residual on a coarse grid level is less than the residual on the next finer level by a specified amount, δ, that solution is determined to be converged and is prolonged to the next finer grid level. The value of δ is taken from Brandt as 0.3. This cycle-C method of Brandt allows for the most efficient use of multiple grid levels by monitoring the residual and determining the optimal time to restrict and prolong. On the other hand, the V-cycle simply goes up and down the grid levels in order, whether or not the convergence rate is slow or if the solution is converged. Both the V-cycle and cycle-C are applied to the IRK-SIMPLER algorithm to determine if the convergence rate can be improved.

In the IRK-SIMPLER algorithm, iterative loops are preformed, which simultaneously solve pressure and velocity. To obtain a converged solution, many iterations are required each stage. Multigrid methods are applied to reduce the number of computations required to get a converged solution at each RK stage. The initial solution of the momentum equations in IRK-SIMPLER is only solved once and does not use the multigrid method. The multigrid methods are used in the iterative loops to accelerate the convergence rate of pressure and momentum equations. When using multigrid methods with the IRK-SIMPLER algorithm, the pressure and velocity are restricted and prolonged and the residual of the pressure and momentum equations are restricted. Each time a new grid level is reached, the non-linear momentum and pressure coefficients are computed with the latest approximation.

Both V-cycle and cycle-C multigrid methods are applied to IRK-SIMPLER. A diagram of the V-cycle multigrid IRK-SIMPLER algorithm is shown in Fig. C.1, and a diagram of the cycle-C multigrid IRK-SIMPLER algorithm is shown in Fig. C.2.

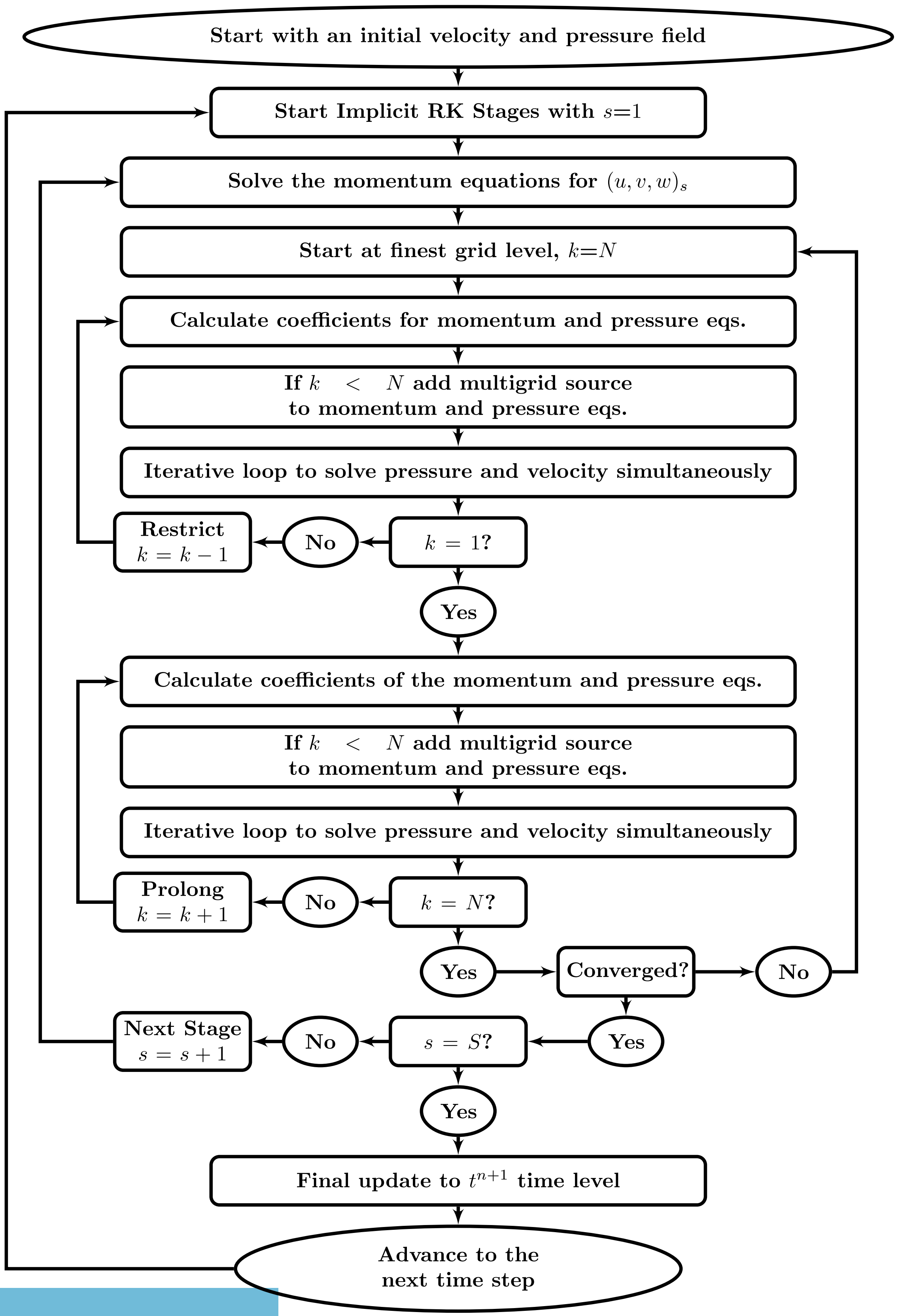

Start with an initial velocity and pressure field
Start Implicit RK Stages with s=1
Solve the momentum equations for $(u, v, w)_s$
Start at finest grid level, k=N
Calculate coefficients for momentum and pressure eqs.
If $k < N$ add multigrid source to momentum and pressure eqs.
Iterative loop to solve pressure and velocity simultaneously
Restrict $k = k - 1$
No
$k = 1$?
Yes
Calculate coefficients of the momentum and pressure eqs.
If $k < N$ add multigrid source to momentum and pressure eqs.
Iterative loop to solve pressure and velocity simultaneously
Prolong $k = k + 1$
No
$k = N$?
Yes
Converged?
No
Next Stage $s = s + 1$
No
$s = S$?
Yes
Yes
Final update to t^{n+1} time level
Advance to the next time step

Figure C.1: Diagram of v-cycle multigrid IRK-SIMPLER algorithm.

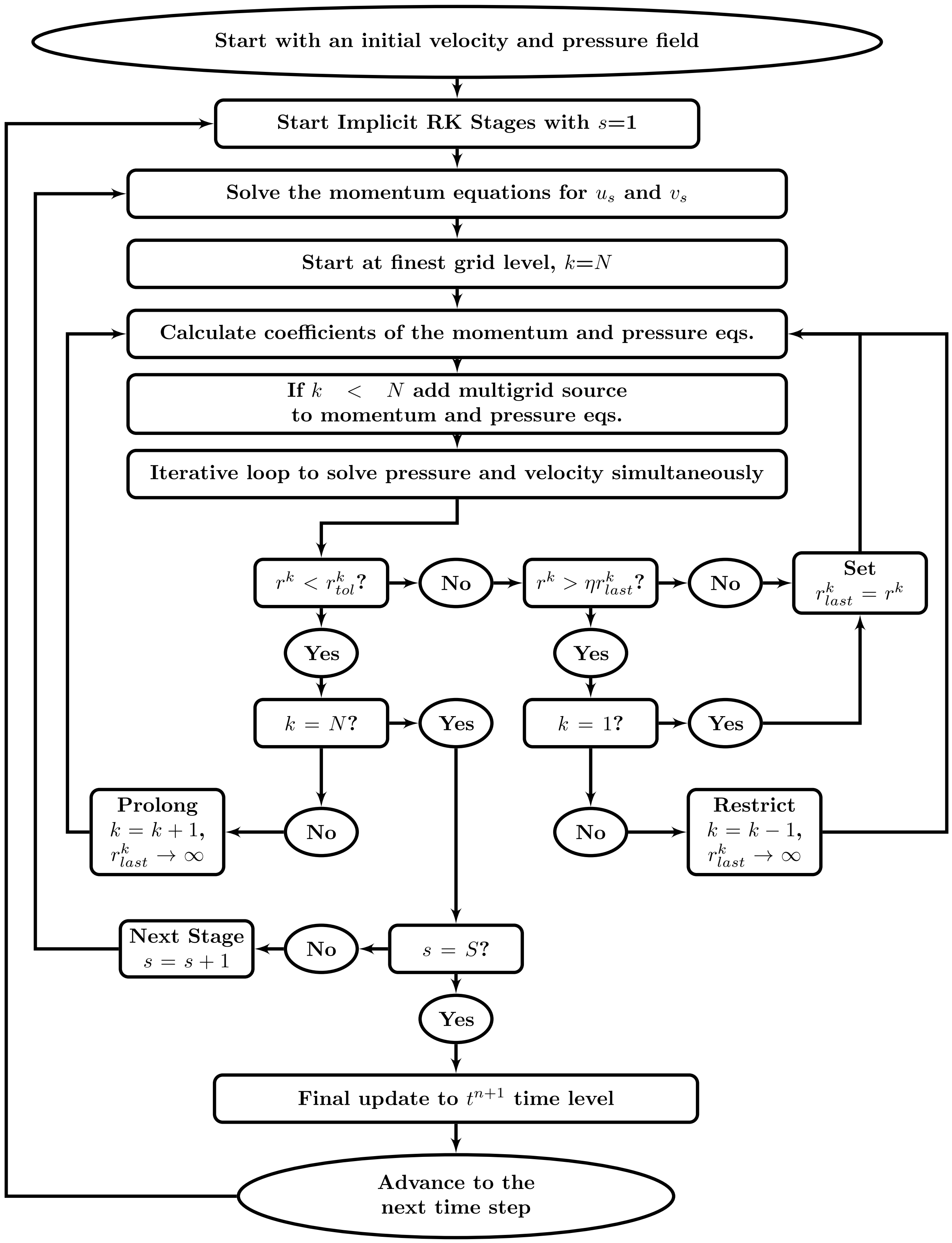

Start with an initial velocity and pressure field
Start Implicit RK Stages with s=1
Solve the momentum equations for u_s and v_s
Start at finest grid level, k=N
Calculate coefficients of the momentum and pressure eqs.
If $k < N$ add multigrid source to momentum and pressure eqs.
Iterative loop to solve pressure and velocity simultaneously
$r^k < r^k_{tol}$?
No
$r^k > \eta r^k_{last}$?
No
Set $r^k_{last} = r^k$
Yes
Yes
$k = N$?
Yes
$k = 1$?
Yes
Prolong $k = k + 1$, $r^k_{last} \to \infty$
No
No
Restrict $k = k - 1$, $r^k_{last} \to \infty$
Next Stage $s = s + 1$
No
$s = S$?
Yes
Final update to t^{n+1} time level
Advance to the next time step

Figure C.2: Diagram of cycle-C multigrid IRK-SIMPLER algorithm.

Results

Multigrid methods are more efficient than single grid methods when a system of equations are solved to a low residual level. In the previous sections, simulations were run with only a few iterations to complete the simulation as fast as possible. Although the results were accurate, the residual values were not guaranteed to reach a certain level each time step. The IRK-SIMPLER algorithm can instead be run with as many iterations as required to reach a given residual tolerance. Figure C.3 shows the mass and momentum residuals versus CPU time for the IRK-SIMPLER algorithm and the multigrid IRK-SIMPLER method for one RK stage. The single grid method residuals drop quickly at first, when the high frequency errors are being removed, but they drop much slower when removing the low frequency errors. The multigrid method allows the residual to drop consistently at a constant rate, but for about the first 25 seconds, the single grid momentum residuals are actually lower than the multigrid method.

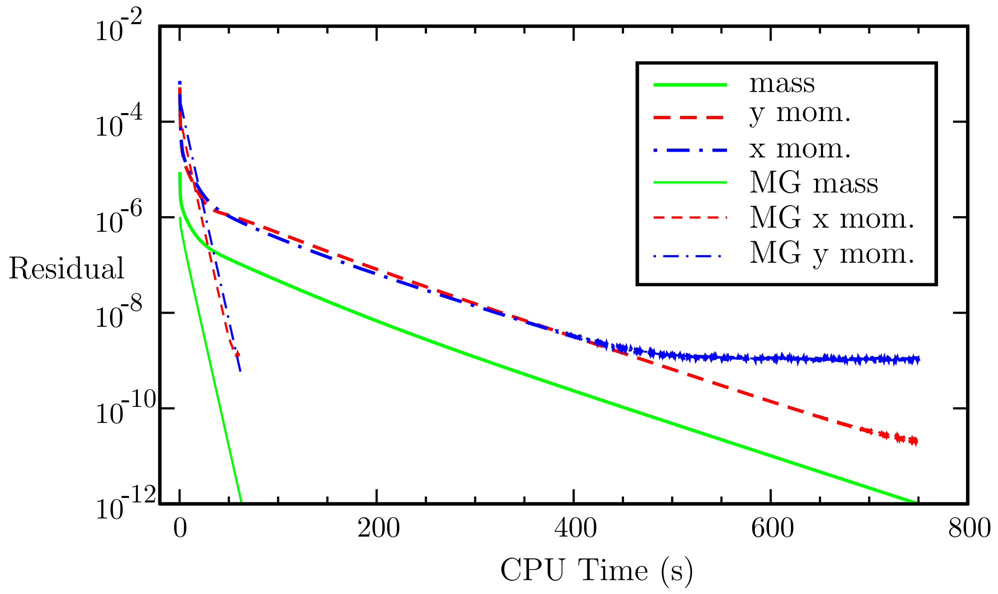


Figure C.3: Residual values with and without multigrid for IRK-SIMPLER.

The multigrid IRK-SIMPLER algorithm is tested on the unsteady flat plate normal to the flow (Sec. 2.4.3.1) and the unsteady infinite square cylinder with the QUICK scheme (Sec. 4.1.2.2). The same structured Cartesian grids are used as in previous tests. The flat plate is run with a time

Table C.1: Two-dimenional flat plate results with multigrid (7 grid levels).

	Multigrid Method	Residual Tolerance	Δt (sec.)	$\overline{C_d}$ at Δt	Sr at Δt	CPU Time at Δt (min.)	Speedup at Δt
IRK	—	1×10^{-6}	4×10^{-5}	2.339	0.150	130.77	1.0
IRK	V-cycle	1×10^{-6}	4×10^{-5}	2.339	0.150	72.26	1.8
IRK	cycle-C	1×10^{-6}	4×10^{-5}	2.339	0.150	34.12	3.8
IRK	—	1×10^{-8}	4×10^{-5}	2.339	0.150	317.83	1.0
IRK	V-cycle	1×10^{-8}	4×10^{-5}	2.339	0.150	170.05	1.9
IRK	cycle-C	1×10^{-8}	4×10^{-5}	2.339	0.150	95.52	3.3

step of $\Delta t' = 0.08$, and the square cylinder with a time step of $\Delta t' = 0.60$. The number of inner iterations for IRK-SIMPLER are as many as needed to yield a specified residual tolerance. Results are given in Tables C.1 and C.2.

For the flat plate, the speedup for both tolerance levels are similar, with cycle-C resulting in faster speedup (up to 3.8). For the square cylinder, multigrid does not decrease the runtime when the residual tolerance is 1×10^{-6}, but for a residual tolerance of 1×10^{-8}, cycle-C reduces runtime by 50% while V-cycle reduces runtime by 24%.

Table C.2: Three-dimensional square cylinder results with multigrid (4 grid levels).

	Multigrid Method	Residual Tolerance	$\overline{C_D}$ at Δt	CPU Time at Δt (min.)	Speedup at Δt
IRK	—	1×10^{-6}	1.684	473.35	1.0
IRK	V-cycle	1×10^{-6}	1.640	886.37	0.5
IRK	cycle-C	1×10^{-6}	1.665	594.05	0.8
IRK	—	1×10^{-8}	1.685	1486.07	1.0
IRK	V-cycle	1×10^{-8}	1.666	1131.68	1.3
IRK	cycle-C	1×10^{-8}	1.683	743.78	2.0

The square cylinder case presented here has a relatively coarse grid. Multigrid methods are most effective for highly refined grids with large systems of equations. More research is suggested to investigate the IRK-SIMPLER algorithm with multigrid for simulations with more refined grids.