Implementation of Two Numerical Solvers for the Study of Non-Equilibrium Gas Dynamics on GPU-Accelerated Platforms using SYCL.

Osman El-Ghotmi

Thesis submitted to the University of Ottawa
in partial Fulfillment of the requirements for the
Master of Applied Science

Department of Mechanical Engineering
Faculty of Engineering
University of Ottawa

Implementation of Two Numerical Solvers for the Study of Non-Equilibrium Gas Dynamics on GPU-Accelerated Platforms using SYCL.

Osman El-Ghotmi
Master of Applied Science

Department of Mechanical Engineering
Faculty of Engineering
University of Ottawa
2024

Abstract

The application of GPUs has extended beyond traditional graphics rendering because their parallel processing capabilities can accelerate many general-purpose tasks, such as machine learning and scientific computing. This thesis presents the implementation of two numerical solvers for the solution of non-equilibrium gas flows. It also demonstrates the computational performance of the two solvers when developed to target GPU-based supercomputers using the SYCL programming model.

The first solver incorporates a novel ray-tracing technique and accurate mathematical relations to efficiently compute any observable property of free-molecular flow past convex shapes (FMFC). It computes integrals of the Maxwell-Boltzmann distribution function to create an algorithm that quickly evaluates any moment of the local particle-velocity distribution. This highly efficient technique is extended for GPUs to accelerate the computation of accurate results. Results produced with the solver serve as robust benchmarks in the validation of other scientific models that describe fluid motion in non-equilibrium regimes.

The second solver extends a CPU-based implementation of the discontinuous Galerkin Hancock (DGH) method into an efficient GPU code. The DGH scheme is a high-order numerical method that solves hyperbolic partial differential equations (PDEs) with stiff source terms. This class of equations is common in many models that are used to describe non-equilibrium gas flows. The GPU implementation of the DGH solver that is presented in this work provides a computationally efficient and numerically accurate method to compute the solution for these models.

Results produced by the FMFC and DGH solvers showcase their accuracy and parallel scalability as efficient GPU algorithms. Furthermore, the effectiveness of the FMFC solver as a validation tool is demonstrated by producing benchmarks to confirm the accuracy of scientific models that are solved with numerical schemes such as DGH.

Acknowledgments

I would like to extend my sincerest appreciation to my advisor, Dr. James McDonald, whose guidance and insight have played an extremely important role throughout this research project. I am also very grateful to my parents and my brother for their unwavering love and encouragement throughout my academic endeavors. I began my graduate studies while recovering from three years of cancer treatment and a stem cell transplant. Without the support that I received from my advisor and my family during this challenging time, I certainly would not have made it this far in my academic journey.

I also owe a great deal of thanks to my friends and team members. It has been good fun to have you all along for the ride.

When I was at my worst, in immense pain, and bedridden at the Ottawa Hospital, I had little hope that I would ever finish my undergraduate degree. Yet, here we are, wrapping up my master's degree and preparing to start my PhD. How neat is that, eh?

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

1.1.1 Computational Fluid Dynamics

The field of computational fluid dynamics (CFD) is the study of fluid flow with the use of numerical algorithms. Simulating different aspects of fluid motion, such as turbulence, can be very demanding on computational resources and largely requires CFD code to be designed to target modern high-performance supercomputers. Many popular algorithms and legacy software in the field of CFD have been heavily optimized for CPU-based platforms. This presents a major issue because there is an ever increasing demand to make a complete move towards GPU-based systems for supercomputing tasks [1]. Migrating towards the use of GPU architectures requires a large investment into the development, implementation, and optimization of new and existing algorithms.

1.1.2 General Purpose GPU Programming

The application of GPUs has extended beyond traditional image rendering, becoming essential in many general purpose tasks, such as machine learning and high-performance scientific computing. GPUs scale very well to massively parallel processing workloads, making them well-suited for solving large numerical problems that require a high degree of parallel computations.

Originally, GPUs were designed to render high fidelity graphics for video game applications. Advancements to the design of GPUs in the early 2000s was heavily focused on performance optimizations for rendering 3D graphics. This mainly required improvements to the GPUs ability to compute floating-point operations very quickly and with a high degree of parallelizability. Such capabilities were immediately sought after by the scientific community [2] because CPUs are designed to handle few complex operations in sequence; whereas, GPUs are designed to handle many simple operations in parallel. This fundamental difference in paradigms pushed scientific researchers to exploit and re-purpose GPUs to accelerate the highly parallel numerical computations that are present in many different scientific applications.

In response to this interest, software developers and hardware manufacturers that specialize

in GPUs, such as Nvidia, created frameworks for general purpose GPU programming (GPGPU). In 2006, Nvidia released CUDA, a proprietary language model for GPU targeted programming. This enabled developers in the scientific community to write highly parallel code in CUDA that can target Nvidia GPUs for general-purpose tasks such as molecular dynamics, astrophysical simulations, and CFD [2, 3]. Currently, Nvidia is the leading supplier of GPUs in the rapidly growing Artificial Intelligence (AI) and High Performance Computing (HPC) markets. This industry dominance is pushing other hardware manufacturers, such as AMD and Intel, to focus their efforts on developing GPUs that can compete with this growing demand. Since CUDA is a proprietary language, exclusive to Nvidia GPUs, software developers are working on open-source alternatives to CUDA, such as OpenCL and, more recently, SYCL. The SYCL open-standard is actively being developed to offer a unified programming model for heterogeneous systems. Furthermore, SYCL is a framework that can be imported directly into a C++ environment and enables developers to write code that can target GPUs from any vendor [4]. The present work makes use of SYCL to implement GPU-based numerical solvers for the solution of non-equilibrium gas flows.

1.1.3 Non-Equilibrium Gas Dynamics

In fluid dynamics, when studying the behaviour of gas particles, in general there are three regimes to consider [5]. There is the continuum regime, where gas particle collisions with one another frequently occur. In this regime, local thermodynamic equilibrium is assumed and traditional models such as the Navier-Stokes equations are used to describe the behavior of the gas flow. There is also the transition regime, where particle collisions in the domain of interest occur less frequently. In this regime, assumptions for local thermodynamic equilibrium start to breakdown, the Navier-Stokes equations can no longer properly describe the gas flow, and specialized models, such as moment closures, are required to describe the properties of the flow. Finally, there is the free-molecular regime, where inter-particle collisions can often be entirely neglected. In this regime, local thermodynamic equilibrium is no longer valid and particle-based methods are required to describe the gas flow. An important non-dimensional number to consider when distinguishing between these regimes is the Knudsen number,

$$\mathrm{Kn} = \frac{\lambda}{l}. \tag{1.1}$$

In this expression, λ is the mean free path of the gas and l is the relevant length scale. If the length scale of a given situation is much greater than the mean free path, then particle collisions frequently occur and the gas is considered to be in the continuum regime. If the mean free path of a given situation is much greater than the length scale, then particle collisions rarely occur and the gas flow is considered to be in the free-molecular regime. It is common to assume that if $\mathrm{Kn} < 0.01$, the gas flow is in the continuum regime, if $0.01 < \mathrm{Kn} < 10$, the gas flow is in the transition regime, and if $10 < \mathrm{Kn}$, the gas flow is in the free-molecular regime. This thesis focuses on non-equilibrium gas flows that are in the transition and free-molecular regimes.

1.2 Motivation

The study of non-equilibrium gas flows is critical in many scientific and engineering applications, which include orbital satellites, particle accelerators, micro-electromechanical systems, and the manufacturing of semi-conductor devices. The work presented in this thesis discusses the implementation of two highly efficient and highly parallel numerical solvers that are used to describe non-equilibrium gas flows. Each of the two solvers have been implemented with SYCL to target GPU-accelerated platforms.

It is desirable to develop GPU versions of the two solvers for a number of reasons. One reason for this approach is because GPU implementations of numerical solvers often show substantial speedups when compared to their CPU counterparts. This is because CPUs typically have few powerful cores that are optimized for complex operations; while, GPUs have many cores that are optimized for simple operations. Studies show substantial performance speedups when using GPUs for workloads that involve a high degree of parallelizability, such as the numerics in CFD applications [6]. Another reason to use GPUs is that new supercomputers are largely being developed with many GPUs. This includes Narval, a supercomputer in Montreal that houses 636 Nvidia A100 GPUs [7], and Aurora, a new exascale supercomputer in Chicago that is equipped with 60,000 Intel Ponte Vecchio GPUs. Additionally, the Aurora exascale platform only supports GPU programs developed with the SYCL programming model [8]. This further incentivizes the development of GPU code with SYCL. Unlike CUDA, which is limited to Nvidia GPUs, SYCL can be used to target upcoming platforms like Aurora, which is Intel-based, as well as platforms like Narval, which has Nvidia-based GPUs, providing the freedom to target any computing facility with GPUs from any vendor.

The work presented in this thesis serves the long-term goal of developing new fluid models that remain valid in thermal non-equilibrium. Currently, computational methods exist that successfully simulate non-equilibrium gas flows; however, all existing methods suffer from certain limitations. The most successful method in these regimes is Direct Simulation Monte Carlo (DSMC) [9]. This probabilistic particle method is fairly efficient for very high-speed flows but becomes expensive and inaccurate at lower speeds. There also exist discrete-velocity models for numerical simulations, such as the one developed by Mieussens [10]. This approach is relatively accurate at lower speed flows, but requires the solution of the Boltzmann equation in six spatial dimensions and one time dimension, which is exceedingly and prohibitively expensive.

1.3 Objective

The objective of the present work is to develop and implement two efficient and accurate numerical solvers to study non-equilibrium gas flows. Another important aspect of this work is to reduce computational runtimes by implementing the two solvers to target large-scale GPU-accelerated systems with SYCL. The first of the two solvers is used as a validation tool that quickly produces a wide range of robust benchmarks for free-molecular flow near convex shapes. The second solver is a discontinuous Galerkin (DG) scheme that is mainly used to compute the solution of models

that describe non-equilibrium gas flows. The two solvers are meant to be used in conjunction with one another such that the free-molecular solver produces accurate benchmarks to validate models that are solved with the DG implementation.

1.3.1 Free-Molecular Flow Near Convex Shapes (FMFC) Solver

The first solver presented in this research project incorporates a novel ray-tracing technique and efficient mathematical relations to compute any observable property for free-molecular flow near convex shapes (FMFC). The objective of the work pertaining to this solver is to conduct highly accurate integrations of sections of the traditional Maxwell-Boltzmann distribution function that describe an ideal gas in local thermodynamic equilibrium, to create an efficient method for evaluating any moment of the distribution. Furthermore, by accelerating this highly efficient solver to target GPUs with SYCL, it can be leveraged as a tool that very quickly produces accurate results. These results can serve as robust benchmarks to validate new and existing models that aim to describe non-equilibrium gas flows.

1.3.2 Discontinuous Galerkin Hancock (DGH) Method

The second goal is to extend an existing CPU-based implementation of the discontinuous Galerkin Hancock (DGH) method [11] into an efficient GPU code. The DGH scheme, originally developed by Suzuki and van Leer [12], is a higher-order numerical technique used to solve systems of hyperbolic-relaxation partial differential equations (PDEs). Equations of this type take the form

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}_i}{\partial x_i} = \mathbf{S}, \tag{1.2}$$

where $\mathbf{U}$ is the solution vector, $\mathbf{F}$ is the flux diad, and $\mathbf{S}$ is the source term. These types of equations are commonly used to study various important scientific problems, such as the models that are used to describe non-equilibrium gas flows [13]. Generally, the relaxation time in the source term of these PDEs are very stiff. That is to say, the source term of these problems can occur on time scales that are very fast compared to other aspects of the situation. Local processes such as drag and ionisation can severely restrict timestep sizes to maintain stable and accurate solutions, when using explicit time marching. The DGH scheme addresses these issues by treating the local time step implicitly, without sacrificing accuracy. This efficient technique extends upon Huynh's one-step, fully discrete, "upwind moment scheme" for application to hyperbolic-relaxation equations [14]. The DGH scheme achieves third-order accuracy in both space and time, enabling the ability to accurately study a wide range of scientific problems. The objective of the work pertaining to this solver is to use SYCL for the development of an efficient GPU-based implementation of the DGH method. DGH presents an inherently parallelizable problem that requires a large amount of local work to be computed. As such, it scales very well to GPUs and reduces the overall computational runtime of the scheme.

1.3.3 Original Contributions

The work presented in this thesis involves many original contributions that include:

- Derivation of exact solutions for the drag and lift coefficients of a rotating cylinder in free-molecular flow.
- Development and implementation of FMFC, a solver that incorporates a novel ray-tracing technique and efficient mathematical relations to compute any observable property of free-molecular flow near convex shapes.
- Extension of the FMFC solver to target GPU platforms using SYCL, further enabling the tools ability to rapidly produce a wide range of benchmarks for model validation.
- Demonstration of substantial speedups for the GPU-based implementation of FMFC, relative to the CPU version.
- Extending the existing CPU-based implementation of the DGH scheme into an efficient GPU code for large-scale heterogeneous computing facilities with SYCL.
- Presentation of strong scaling studies for the GPU implementation of DGH, by distributing the workload across multiple GPUs on heterogeneous computing facilities.
- Demonstration of significant speedups for the GPU version of DGH, relative to its existing CPU counterpart.

1.4 Research Outline

The thesis first introduces, in Chapter 2, a brief explanation of the kinetic theory of gases, the importance of the Maxwell-Boltzmann distribution, a brief discussion on moment-closures, and relevant models that are derived with moment methods. It then proceeds to describe, in Chapter 3, the derivation of analytical solutions for drag and lift, important boundary conditions, relevant geometric relations, and development of the FMFC solver. The accuracy of the solver is confirmed against the analytical solutions for drag and lift. The FMFC solver is then used to produce studies in the free-molecular regime with performance comparisons for the CPU and GPU implementations. Chapter 4 provides an overview of the DGH scheme that highlights the weak formulation of the solution and the derivation of discrete update formulas. It also describes the GPU-based implementation of the scheme and presents scaling studies, performance comparisons against the existing CPU version, and interesting scientific results produced with the implementation. In Chapter 5, benchmark results produced with FMFC are used to investigate the accuracy of moment closure models that are solved with DG schemes. Specifically, the ten-moment equations are used to describe free-molecular flow near a circular cylinder. These equations are solved using DG schemes and compared against benchmarks produced by FMFC. This approach highlights the significance of FMFC as a robust validation tool that can confirm a models level of accuracy in the free-molecular regime.

Chapter 2

Kinetic Theory of Gases and Moment Closures

2.1 Kinetic Theory

When analysing the flow of gas particles, it is particularly interesting to calculate their properties at various times and positions in space. In theory, this can be done with Newton's laws to evaluate the physical motions and dynamics of each individual particle in the gas flow. However, this is not a realistic or reasonable approach because there are too many individual particles that would need to be considered and, in general, it is only the macroscopic observable properties of the gas flow that are actually of interest. The kinetic theory of gases provides the tools necessary to quantitatively calculate the behaviour of a gas flow as a collection of particles [5]. There are a few assumptions that have to be made in order to to use the various methods highlighted in the kinetic theory of gases. From the molecular hypothesis, all matter is composed of discrete particles and all particles of the same species are identical. For ideal monatomic gases, particles are considered as points with no internal structure, they only exert forces over very small distances, and collisions between particles are at most binary. These assumptions also state that all gaseous particles are moving in statistically uncorrelated directions and the collection of gaseous particles are not dense. Any collisions between two particles are also assumed to be completely elastic and so no energy or momentum is lost.

2.1.1 Velocity Distribution Function

Having made assumptions pertaining to the kinetic theory of gases, it is essential to define a velocity distribution function $\mathcal{F}(\vec{x}, \vec{v}, t)$ that describes the statistical behavior of the gas flow as a collection of individual particles. This function provides the probability density of particles with velocity $\vec{v}$ for a time t and position $\vec{x}$. A great way to accurately represent the velocity distribution of particles is the Maxwell-Boltzmann distribution function. The velocity distribution function

for a monatomic gas that is in local thermodynamic equilibrium takes the form

$$\mathcal{F}(\vec{x},\vec{v},t) = \frac{\rho(\vec{x},t)}{m}\left(\frac{\rho(\vec{x},t)}{2\pi P(\vec{x},t)}\right)^{3/2} e^{-\frac{\rho(\vec{x},t)}{2P(\vec{x},t)}(\vec{v}-\vec{u}(\vec{x},t))^2}. \tag{2.1}$$

Here, m is the particle mass, ρ is the density, P is the pressure, and $\vec{u}$ is the local average velocity. Inter-particle collisions drive the gas to reach this state of local thermodynamic equilibrium. Once this state is reached, the distribution function is no longer affected by inter-particle collisions. In other words, the second law of thermodynamics causes all distribution functions to approach the Maxwell Boltzmann distribution function, as shown in Equation (2.1). In Cartesian coordinates, the distribution takes the form

$$\mathcal{F}(\vec{x},\vec{v},t) = \frac{\rho(\vec{x},t)}{m}\left(\frac{\rho(\vec{x},t)}{2\pi P(\vec{x},t)}\right)^{3/2} e^{-\frac{\rho(\vec{x},t)}{2P(\vec{x},t)}((v_x-u_x(\vec{x},t))^2+(v_y-u_y(\vec{x},t))^2+(v_z-u_z(\vec{x},t))^2)}, \tag{2.2}$$

where v_x, v_y, and v_z represent the x-, y-, and z-direction velocities, respectively. Additionally, u_x, u_y, and u_z represent the bulk velocities in each of their respective Cartesian directions. Furthermore, in tensor notation, the distribution is written out to be

$$\mathcal{F}(x_i,v_i,t) = \frac{\rho(x_i,t)}{m}\left(\frac{\rho(x_i,t)}{2\pi P(x_i,t)}\right)^{3/2} e^{-\frac{\rho(x_i,t)}{2P(x_i,t)}((v_i-u_i)(v_i-u_i))}. \tag{2.3}$$

2.1.2 Moments of the Distribution

Any observable property of the gas flow can be calculated by computing velocity moments of the velocity distribution function. These moments are evaluated by first raising the velocities in each direction, v_x, v_y, and v_z, to the powers n_x, n_y, and n_z, respectively. The moment is then calculated by integrating the product of the velocities with the distribution function through all velocity space. In this study, these moments are expressed with notation

$$\left\langle m{v_x}^{n_x}{v_y}^{n_y}{v_z}^{n_z}\mathcal{F}\right\rangle = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} m{v_x}^{n_x}{v_y}^{n_y}{v_z}^{n_z}\mathcal{F}\,\mathrm{d}v_x\,\mathrm{d}v_y\,\mathrm{d}v_z \tag{2.4}$$

and the applied velocity weights are represented by M, where

$$M = {v_x}^{n_x}{v_y}^{n_y}{v_z}^{n_z}. \tag{2.5}$$

This simplifies Equation (2.4) to

$$\langle mM\mathcal{F}\rangle = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} mM\mathcal{F}\,\mathrm{d}v_x\,\mathrm{d}v_y\,\mathrm{d}v_z. \tag{2.6}$$

Computing these moments enables the study of any observable property of the gas flow that may be of interest. For example, using these moments, the local mass density, ρ, of particles is

$$\rho = \langle m\mathcal{F} \rangle \tag{2.7}$$

and the momentum density is

$$\rho u_i = \langle m v_i \mathcal{F} \rangle \,, \tag{2.8}$$

which leads to the bulk velocity

$$u_i = \frac{\rho u_i}{\rho} = \frac{\langle m v_i \mathcal{F} \rangle}{\langle m\mathcal{F} \rangle}. \tag{2.9}$$

It is also interesting to compute high-order velocity moments. Second-order velocity moments are used to construct the energy tensor

$$E_{ij} = \left\langle m v_i v_j \mathcal{F} \right\rangle = \rho u_i u_j + P_{ij}. \tag{2.10}$$

Here P_{ij} is the pressure tensor and it is related to the hydrostatic pressure

$$P = \frac{P_{xx} + P_{yy} + P_{zz}}{3}. \tag{2.11}$$

Using the computed pressure values in Equation (2.10) and Equation (2.11), the deviatoric stress tensor is

$$\vec{\tau} = \begin{bmatrix} \tau_{xx} & \tau_{xy} & \tau_{xz} \\ \tau_{xy} & \tau_{yy} & \tau_{yz} \\ \tau_{xz} & \tau_{yz} & \tau_{zz} \end{bmatrix} = \begin{bmatrix} P & 0 & 0 \\ 0 & P & 0 \\ 0 & 0 & P \end{bmatrix} - \begin{bmatrix} P_{xx} & P_{xy} & P_{xz} \\ P_{xy} & P_{yy} & P_{yz} \\ P_{xz} & P_{yz} & P_{zz} \end{bmatrix} \tag{2.12}$$

and the scalar temperature of the gas is defined as

$$T = \frac{mP}{\rho k}, \tag{2.13}$$

where $k = 1.38054 \times 10^{-23}$ J/K is the Boltzmann constant. Another interesting high-order velocity moment is the third-order moment

$$\left\langle m v_i v_j v_k \mathcal{F} \right\rangle = \rho u_i u_j u_k + u_i P_{ik} + u_j P_{jk} + u_k P_{ij} + Q_{ijk}, \tag{2.14}$$

where Q_{ijk} is the generalized heat-flux tensor. By isolating for heat-flux terms in Equation (2.14), the directional heat-flux vector is computed as

$$q_i = \frac{Q_{ijj}}{2}. \tag{2.15}$$

Here, Q_{ijj} is the trace of the generalized heat-flux tensor. The heat-flux vector in each x-, y-, and z-direction is

$$q_x = \frac{Q_{xxx} + Q_{xyy} + Q_{xzz}}{2}, \tag{2.16}$$

$$q_y = \frac{Q_{yxx} + Q_{yyy} + Q_{yzz}}{2}, \tag{2.17}$$

and

$$q_z = \frac{Q_{zxx} + Q_{zyy} + Q_{zzz}}{2}. \tag{2.18}$$

The entropy density can also be obtained by applying the velocity weight

$$M_S = -\frac{k}{m} \ln \frac{\mathcal{F}}{y}, \tag{2.19}$$

where y is a constant value used to non-dimensionalize the distribution function. The entropy density, which was proved by Boltzmann, is

$$S = \langle M_S \mathcal{F} \rangle . \tag{2.20}$$

Entropy is maximized for $y = ez$, yielding

$$M_{\max} = -\frac{k}{m} \left(\ln \frac{\mathcal{F}}{z} - 1 \right). \tag{2.21}$$

This velocity weight is used to obtain a form of the distribution function that maximizes the entropy. The moments of the distribution function described in this section of the chapter are the macroscopic observable properties of the gas flow that are particularly interesting to study. It is also important to know how these properties change with time.

2.1.3 The Boltzmann Equation

The Boltzmann equation is used to determine the evolution of the distribution function over time, such that

$$\frac{\partial \mathcal{F}}{\partial t} + v_i \frac{\partial \mathcal{F}}{\partial x_i} + a_i \frac{\partial \mathcal{F}}{\partial v_i} = \frac{\delta \mathcal{F}}{\delta t}, \tag{2.22}$$

where a_i is the acceleration of particles that result from external forces. If the acceleration term is neglected, a_i is taken to be zero and Equation (2.22) is simplified to

$$\frac{\partial \mathcal{F}}{\partial t} + v_i \frac{\partial \mathcal{F}}{\partial x_i} = \frac{\delta \mathcal{F}}{\delta t}. \tag{2.23}$$

Additionally, $\frac{\delta \mathcal{F}}{\delta t}$ represents the collision operator for collisions between particles. Traditionally, this term follows the assumptions that the only collisions that take place on scales much smaller than the mean free path are binary, the distribution function is constant across this range, individual collisions do not significantly change the distribution function, and particle velocities are uncorrelated. It is largely impractical to directly solve the Boltzmann equation. This is because numerically solving the equation requires a huge amount of information to be computed with high resolution discretizations of the physical space and the velocity space. Quickly, this becomes prohibitively expensive for modern systems to compute. In general, only the solutions

for specific macroscopic moments of the gas flow need to be computed. This can be done using moment methods, which were originally presented by Grad [15].

2.1.4 The Collision Operator

The collision operator is used to describe two colliding particles and given by the integral expression

$$\frac{\delta \mathcal{F}}{\delta t} = \left\langle \int_0^{2\pi} \int_0^{\pi} \left(\mathcal{F}' \mathcal{F}^1 - \mathcal{F} \mathcal{F}^{1'} \right) g \sigma \sin \chi \, \mathrm{d}\chi \, \mathrm{d}\epsilon \right\rangle_{v_i'} . \tag{2.24}$$

In this equation, the velocities for each of the two colliding particles are v_i and v_i^1. The distribution functions $\mathcal{F}$ and $\mathcal{F}^1$ are evaluated at the respective velocities v_i and v_i^1 of the colliding particles. Variables marked with $'$ represent post-collision states. The term, g, is the relative speed of the two particles before the collision and is given by

$$g = \left| v_i - v_i^1 \right| . \tag{2.25}$$

Furthermore, σ is the differential collision cross section, χ is the deflection angle, and ϵ is the relevant angle for the particle collision. The collision operator is very complex and difficult to compute. To address this issue, simpler models, such as the BGK operator [16], are generally used. This simplified model is represented by

$$\frac{\delta \mathcal{F}}{\delta t} = -\frac{\mathcal{F}(x_i, v_i, t)}{\tau_{\mathcal{F}}(x_i, t)} + \frac{\mathcal{M}(x_i, v_i, t)}{\tau_{\mathcal{M}}(x_i, t)}. \tag{2.26}$$

The BGK model represents particles that are being removed from non-equilibrium with a characteristic time $\tau_{\mathcal{F}}$ and particles that are being added to equilibrium with a characteristic time $\tau_{\mathcal{M}}$. The rate at which particles are being removed from non-equilibrium and added to equilibrium is assumed to occur over the same characteristic time τ. This means that

$$\tau = \tau_{\mathcal{F}} = \tau_{\mathcal{M}}. \tag{2.27}$$

This assumption simplifies Equation (2.26) to

$$\frac{\delta \mathcal{F}}{\delta t} = \frac{-\mathcal{F}(x_i, v_i, t) + \mathcal{M}(x_i, v_i, t)}{\tau}. \tag{2.28}$$

From this simplified model, it is evident that when the distribution of all particles approach equilibrium, the collision operator approaches zero. So when all particles reach thermodynamic equilibrium, $\frac{\delta \mathcal{F}}{\delta t} = 0$ as expected.

2.1.5 Maxwell's Equation of Change

Maxwell's equation of change provides a way to understand how observable quantities of the gas flow develop. This is achieved by taking moments of the Boltzmann equation that appear in

Equation (2.23), such that

$$\frac{\partial}{\partial t}\langle mM\mathcal{F}\rangle + \frac{\partial}{\partial x_i}\langle mv_iM\mathcal{F}\rangle = \left\langle mM\frac{\delta\mathcal{F}}{\delta t}\right\rangle. \tag{2.29}$$

The collision operator is also simplified with notation

$$\Delta\left[M\mathcal{F}\right] = \left\langle mM\frac{\delta\mathcal{F}}{\delta t}\right\rangle, \tag{2.30}$$

which further leads Equation (2.29) to be expressed as

$$\frac{\partial}{\partial t}\langle mM\mathcal{F}\rangle + \frac{\partial}{\partial x_i}\langle mv_iM\mathcal{F}\rangle = \Delta\left[M\mathcal{F}\right]. \tag{2.31}$$

This equation is represented in the general balance law form of the hyperbolic PDE that was introduced in Equation (1.2).

2.2 Moment Methods

As previously discussed, it is computationally inefficient and unrealistic to directly solve the Boltzmann equation. Moment methods are used to derive models in the form of hyperbolic PDEs that can be solved numerically and with a much higher degree of computational efficiency.

2.2.1 Moment Closures

Equation (2.31) can be extended for a vector of weights

$$\mathbf{M} = [\mathbf{M_0}, \mathbf{M_1}, \mathbf{M_2}, ..., \mathbf{M_N}] \tag{2.32}$$

and expressed as

$$\frac{\partial}{\partial t}\langle m\mathbf{M}\mathcal{F}\rangle + \frac{\partial}{\partial x_i}\langle mv_i\mathbf{M}\mathcal{F}\rangle = \Delta\left[\mathbf{M}\mathcal{F}\right]. \tag{2.33}$$

From the balance law in Equation (2.33), it is immediately noticeable that the system is not closed. This is because the flux term,

$$\mathbf{F} = \langle mv_i\mathbf{M}\mathcal{F}\rangle, \tag{2.34}$$

always has an entry that is one order higher than any term in the solution vector,

$$\mathbf{U} = \langle m\mathbf{M}\mathcal{F}\rangle. \tag{2.35}$$

This presents an issue because the observable properties of the gas flow, represented by the solution vector $\mathbf{U}$, develop over time as described by the flux term $\mathbf{F}$. Moment methods are used to address this issue and close the system. Some methods that are used to close the system were originally developed and presented by Grad [15]. Grad's method closes the system by assuming a

polynomial expansion of the distribution function around the equilibrium Maxwellian, $\mathcal{M}$, such that

$$\mathcal{F} = \mathcal{M}(\mathbf{a}^T\mathbf{M}), \tag{2.36}$$

where $\mathbf{a}$ is a vector of free parameters with the same length as $\boldsymbol{U}$. Unfortunately, closures of this form can introduce physical impossibilities, such as negative mass and pressure, because the free parameters that are applied to the distribution are not strictly positive. This issue led to the development of other moment methods, one of which is the maximum-entropy hierarchy.

Maximum-entropy moment closures are a class of moment methods that have been developed with multiple contributions [17, 18, 19]. This method works by selecting a distribution function that maximizes the entropy of the system, while remaining consistent with the known moments. From Equation (2.20) and Equation (2.21), the described approach for maximizing entropy is

$$\max_{\mathcal{F}} \langle -(\mathcal{F}\ln\mathcal{F} - \mathcal{F})\rangle \tag{2.37}$$

subject to

$$\mathbf{U} = \langle m\mathbf{M}\mathcal{F}\rangle\,. \tag{2.38}$$

In order to find a maximization to the entropy that adheres to Equation (2.38), Lagrange multipliers are used to locate the maxima. In this augmented problem, the function $\mathbf{J}$ is defined as

$$J = \langle \mathcal{F}\ln\mathcal{F} - \mathcal{F}\rangle + \lambda^T\left(\langle m\mathbf{M}\mathcal{F}\rangle - \mathbf{U}\right), \tag{2.39}$$

where λ represents a vector of Lagrange multipliers and the extrema of the problem is found by taking $\frac{\partial J}{\partial \mathcal{F}} = 0$, leading to

$$\left\langle \ln\mathcal{F} + \lambda^T m\mathbf{M}\right\rangle = 0. \tag{2.40}$$

It becomes clear that the distribution function that maximizes the entropy of the system equates to

$$\mathcal{F} = e^{-\lambda^T m\mathbf{M}}. \tag{2.41}$$

Some models developed from the maximum-entropy moment method, such as the ten-moment equations, have proven to be very successful. The ten-moment model is globally hyperbolic and accurately models viscous gas flows in the continuum and transition regimes, without consideration for the heat-flux. However, there are drawbacks and restrictions with maximum-entropy moment methods. One main issue arises when trying to model higher-order moments which leads to moments of the distribution function that cannot be evaluated in closed form. This roadblock has pushed research into the development of new moment methods. One of these methods is the φ-divergence closure, a method that is based on the approximation of an exponential function and inspired from the maximum-entropy moment closure [20]. Another set of moment closures are quadrature-based moment methods which were first introduced by McGraw [21] and have been further developed and advanced by Fox [22]. Though these other methods can have advantages, this research work focuses exclusively on maximum-entropy moment closures. It

should be noted that all mentioned moment-closure methods lead to PDEs in balance-law form and are thus amenable to numerical solutions with the DGH method.

2.3 Models Derived from Maximum-Entropy Moment Methods

Two important models that are derived from the maximum-entropy moment closure and used in the present work are the Euler equations and the ten-moment equations.

2.3.1 Euler Equations

The compressible Euler equations can be derived from the maximum-entropy moment method to study the behaviour of gas flows in the continuum regime. A monatomic gas in local thermodynamic equilibrium has a Maxwellian velocity distribution function

$$\mathcal{F}(x_i, v_i, t) = \frac{\rho(x_i,t)}{m}\left(\frac{\rho(x_i,t)}{2\pi P(x_i,t)}\right)^{3/2} e^{-\frac{\rho(x_i,t)}{2P(x_i,t)}((v_i-u_i)(v_i-u_i))}. \tag{2.42}$$

This distribution function is used to obtain the Euler equations from the maximum-entropy moment closure. The Euler equations are used to describe the density ρ, pressure P, and bulk velocity u_i of the gas. The evolution of these macroscopic properties are described by

$$\frac{\partial}{\partial t}(\rho) + \frac{\partial}{\partial x_i}(\rho u_i) = 0, \tag{2.43}$$

$$\frac{\partial}{\partial t}(\rho u_i) + \frac{\partial}{\partial x_j}\left(\rho u_i u_j + P\delta_{ij})\right) = 0, \tag{2.44}$$

and

$$\frac{\partial}{\partial t}(3P + \rho u_i u_i) + \frac{\partial}{\partial x_j}\left(u_j\left(5P + \rho u_i u_i\right)\right) = 0. \tag{2.45}$$

In these equations, δ_{ij} is the Kronecker delta, where

$$\delta_{ij} = \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{otherwise} \end{cases}. \tag{2.46}$$

In two-dimensional space, the equations follow that the solution vector $\boldsymbol{U}$ is

$$\boldsymbol{U} = \begin{bmatrix} \rho \\ \rho u_x \\ \rho u_y \\ 3P + \rho(u_x^2 + u_y^2) \end{bmatrix}, \tag{2.47}$$

the x-direction flux term $\boldsymbol{F}_x$ is

$$\mathbf{F}_x = \begin{bmatrix} \rho u_x \\ \rho u_x^2 + P \\ \rho u_x u_y \\ u_x \left(5P + \rho(u_x^2 + u_y^2)\right) \end{bmatrix}, \tag{2.48}$$

the y-direction flux term $\boldsymbol{F}_y$ is

$$\mathbf{F}_y = \begin{bmatrix} \rho u_y \\ \rho u_x u_y \\ \rho u_y^2 + P \\ u_y \left(5P + \rho(u_x^2 + u_y^2)\right) \end{bmatrix}, \tag{2.49}$$

and the source term $\boldsymbol{S}$ is

$$\boldsymbol{S} = 0, \tag{2.50}$$

such that the conservation form of the system follows

$$\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial \boldsymbol{F}_x}{\partial x} + \frac{\partial \boldsymbol{F}_y}{\partial y} = 0. \tag{2.51}$$

This work uses the two-dimensional compressible Euler equations to produce scientific results that describe gas flows in the continuum regime. The three-dimensional form for these equations are obtained similar to the two-dimensional set of equations.

2.3.2 Ten-Moment Equations

The ten-moment equations are also derived from the maximum-entropy moment closure. This closure is obtained with a Gaussian velocity distribution function

$$\mathcal{F}(x_i, v_i, t) = \frac{\rho(x_i, t)}{m(2\pi)^{3/2}(\det \Theta_{ij})^{1/2}} e^{-\frac{1}{2}\Theta_{ij}^{-1}((v_i - u_i)(v_j - u_j))}, \tag{2.52}$$

where Θ_{ij} is the anisotropic temperature that is given by

$$\Theta_{ij} = \frac{\rho(x_i, t)}{P_{ij}}. \tag{2.53}$$

In two spacial dimensions, the ten-moment equations are a set of seven PDEs and in three spatial dimensions, they are a set of ten PDEs. The observable properties that are described in this moment-closure is the density ρ, bulk velocity u_i, and pressure tensor P_{ij}. When BGK is used, the evolution of these macroscopic properties follow that

$$\frac{\partial}{\partial t}(\rho) + \frac{\partial}{\partial x_i}(\rho u_i) = 0, \tag{2.54}$$

$$\frac{\partial}{\partial t}(\rho u_i) + \frac{\partial}{\partial x_j}(\rho u_i u_j + P_{ij}) = 0, \tag{2.55}$$

and

$$\frac{\partial}{\partial t}\left(\rho u_i u_j P_{ij}\right) + \frac{\partial}{\partial x_k}\left(\rho u_i u_j u_k + u_k P_{ij} + u_j P_{ik} + u_i P_{jk}\right) = -\frac{1}{\tau}\left(P_{ij} - \frac{1}{3}P_{kk}\delta_{ij}\right). \tag{2.56}$$

This closure does not capture the heat-flux of the system; however, it has still shown to be very successful at accurately describing the evolution of gas flows in the continuum and transition regimes. These equations can be used as alternatives to the Navier-Stokes equations, especially in the transition regime where the Navier-Stokes equations fail and the ten-moment equations can remain accurate. In two spatial dimensions, the solution vector $\boldsymbol{U}$ is

$$\boldsymbol{U} = \begin{bmatrix} \rho \\ \rho u_x \\ \rho u_y \\ \rho u_x^2 + P_{xx} \\ \rho u_x u_y + P_{xy} \\ \rho u_y^2 + P_{yy} \\ P_{zz} \end{bmatrix}, \tag{2.57}$$

the x-direction flux diad $\boldsymbol{F}_x$ is

$$\mathbf{F}_x = \begin{bmatrix} \rho u_x \\ \rho u_x^2 + P_{xx} \\ \rho u_x u_y + P_{xy} \\ \rho u_x^3 + 3u_x P_{xx} \\ \rho u_x^2 u_y + 2u_x P_{xy} + u_y P_{xx} \\ \rho u_x u_y^2 + u_x P_{yy} + 2u_y P_{xy} \\ u_x P_{zz} \end{bmatrix}, \tag{2.58}$$

the y-direction flux diad $\boldsymbol{F}_y$ is

$$\boldsymbol{F}_y = \begin{bmatrix} \rho u_y \\ \rho u_x u_y + P_{xy} \\ \rho u_y^2 + P_{yy} \\ \rho u_x^2 u_y + 2u_x P_{xy} + u_y P_{xx} \\ \rho u_x u_y^2 + u_x P_{yy} + 2u_y P_{xy} \\ \rho u_y^3 + 3u_y P_{yy} \\ u_y P_{zz} \end{bmatrix}, \tag{2.59}$$

and the source term $\boldsymbol{S}$ is

$$\boldsymbol{S} = -\frac{1}{\tau}\begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{2P_{xx}-P_{yy}-P_{zz}}{3} \\ P_{xy} \\ \frac{2P_{yy}-P_{xx}-P_{zz}}{3} \\ \frac{2P_{zz}-P_{xx}-P_{yy}}{3} \end{bmatrix}, \tag{2.60}$$

where τ is the relaxation time and evaluated to be the ratio between the viscosity, μ, and the hydrostatic pressure, P, such that

$$\tau = \frac{\mu}{P}. \tag{2.61}$$

In balance-law form, the system follows

$$\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial \boldsymbol{F}_x}{\partial x} + \frac{\partial \boldsymbol{F}_y}{\partial y} = \boldsymbol{S}. \tag{2.62}$$

The two-dimensional ten-moment equations are used in this thesis to study gas flows in the transition and free-molecular regimes. The three-dimensional set of equations can be obtained in a similar way.

The following chapters present the development and implementation of two highly efficient solvers to study the observable properties of non-equilibrium gas flows. The FMFC solver serves as an important validation tool that quickly produces robust benchmarks to confirm the accuracy of scientific models for gas flows in the free-molecular regime. The second solver implements the DGH method, a highly efficient DG scheme that is used to compute the solution of first-order hyperbolic PDEs with stiff source terms. Scientific models derived with moment closures, such as the ten-moment equations, take the form of these hyperbolic PDEs. In this work, DGH is used as the numerical scheme that solves these models. Together, FMFC and DGH form a comprehensive approach to studying non-equilibrium gas flows. Benchmarks generated with FMFC are used to validate models that are solved numerically with DG schemes. Additionally, by implementing the two solvers to target GPUs, highly accurate benchmarks and efficient large-scale simulations can be produced very quickly. This provides robust tools for the accurate and validated study of non-equilibrium gas flows.

Chapter 3

A Novel Ray-Tracing Technique for Free-Molecular Flow near Convex Shapes

This chapter presents the implementation of a numerical solver to study free-molecular flow near convex shapes. The solver incorporates a novel ray-tracing technique and highly accurate mathematical relations to compute any observable property of the gas flow at any location. The objective of the work in this chapter is to develop a numerical technique that computes accurate integrations of sections of the traditional Maxwell-Boltzmann distribution functions, which describe an ideal gas in local thermodynamic equilibrium, and integrate them in a computationally efficient way. The idea is to create a reliable and accurate method for evaluating any moment of the distribution at any point near the convex shape. This solver is developed with SYCL to target GPUs, providing a tool that can very quickly and efficiently produce accurate results for use as benchmarks. The FMFC solver is a tool that is used to validate the accuracy of scientific models in their ability to describe gas flows in the free-molecular regime.

The current chapter highlights explanations for the boundary conditions that are used, analytical derivations for drag and lift, development of the ray-tracing technique, relevant geometric relations, and the construction of highly efficient relations used to create the FMFC solver. An overview of the implementation is also presented with computational results for scientific problems and scaling comparisons between the CPU and GPU versions of the code.

3.1 Overview for the Functionality of the FMFC Solver

In free-molecular flow near convex shapes, the dominant interaction is particle collisions with surfaces. Furthermore, when the flow is passing near the convex shape, it can be said with certainty that any particle impacting the surface of the shape is coming from the free-stream flow. This means that, when evaluating moments of the distribution function that are describing the flow at any point near the convex shape, the distribution is a combination of particles that are from the free-stream, FS, and particles that have undergone a single reflection from the wall, W. The presented technique works with any convex shape; however, for simplicity, the geometric relations presented in this thesis only consider circular cylinders. Figure 3.1 highlights a point

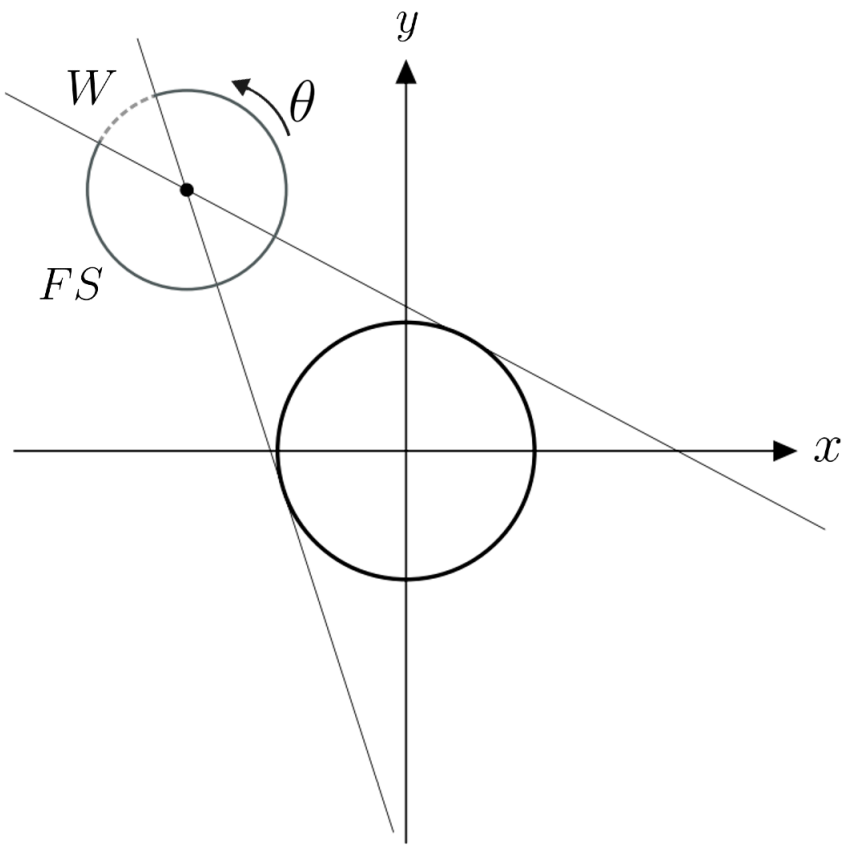


Figure 3.1: Distribution of free-stream and reflected particles around a convex shape in free-molecular flow

in space that is situated near a circular cylinder. The figure depicts the directions from which particles passing through the point are described by either the velocity distribution of particles from the free-stream or the velocity distribution of particles that have been reflected from the cylinder surface. The distribution of particles from the free-stream $\mathcal{M}_{FS}$ is assumed to be

$$\mathcal{M}_{FS} = \frac{\rho_{\mathrm{FS}}}{m}\left(\frac{\rho_{\mathrm{FS}}}{2\pi P_{\mathrm{FS}}}\right)^{3/2} e^{-\frac{\rho_{\mathrm{FS}}}{2P_{\mathrm{FS}}}((v_x-u_x)^2+(v_y-u_y)^2+(v_z-u_z)^2)} \tag{3.1}$$

and the distribution of particles that are reflected and fully accommodated by the cylinder wall $\mathcal{M}_W$ is

$$\mathcal{M}_W = \frac{\rho_{\mathrm{W}}}{m}\left(\frac{\rho_{\mathrm{W}}}{2\pi P_{\mathrm{W}}}\right)^{3/2} e^{-\frac{\rho_{\mathrm{W}}}{2P_{\mathrm{W}}}((v_x-u_x)^2+(v_y-u_y)^2+(v_z-u_z)^2)}. \tag{3.2}$$

Since the solver deals with angular integrations, it is convenient and advantageous to transition to a cylindrical coordinate system. The cylindrical coordinate system introduces the radial velocity term, v_r, and the angular term, θ. The velocity terms, v_x and v_y, can be expressed in terms of the radial velocity, where

$$v_x = v_r \cos\theta \tag{3.3}$$

and

$$v_y = v_r \sin\theta. \tag{3.4}$$

Additionally, the differential element for spatial integrations becomes

$$\mathrm{d}v_x\,\mathrm{d}v_y\,\mathrm{d}v_z = v_r\,\mathrm{d}\theta\,\mathrm{d}v_r\,\mathrm{d}v_z, \tag{3.5}$$

where

$$0 \leq \theta \leq 2\pi, \tag{3.6}$$

$$0 \leq v_r < \infty, \tag{3.7}$$

and

$$-\infty < v_z < \infty. \tag{3.8}$$

In cylindrical coordinates, the distribution of particle velocities from the free-stream is expressed as

$$\mathcal{M}_{FS} = \frac{\rho_{\mathrm{FS}}}{m}\left(\frac{\rho_{\mathrm{FS}}}{2\pi P_{\mathrm{FS}}}\right)^{3/2} e^{-\frac{\rho_{\mathrm{FS}}}{2P_{\mathrm{FS}}}((v_r\cos\theta-u_x)^2+(v_r\sin\theta-u_y)^2+(v_z-u_z)^2)} \tag{3.9}$$

and the distribution of particle velocities that are reflected and fully accommodated by the cylinder wall is

$$\mathcal{M}_{W} = \frac{\rho_{\mathrm{W}}}{m}\left(\frac{\rho_{\mathrm{W}}}{2\pi P_{\mathrm{W}}}\right)^{3/2} e^{-\frac{\rho_{\mathrm{W}}}{2P_{\mathrm{W}}}((v_r\cos\theta-u_x)^2+(v_r\sin\theta-u_y)^2+(v_z-u_z)^2)}. \tag{3.10}$$

Equation (2.6) represents moments of a distribution function that are computed with the Cartesian coordinate system. Applying cylindrical coordinates to these moments yields

$$\left\langle mM\mathcal{F}_p\right\rangle = \int_{-\infty}^{\infty}\int_{0}^{\infty}\int_{0}^{2\pi} mM_{\mathrm{cyl}}\mathcal{F}_p v_r\,\mathrm{d}\theta\,\mathrm{d}v_r\,\mathrm{d}z, \tag{3.11}$$

where

$$M_{\mathrm{cyl}} = (v_r\cos\theta)^{n_x}(v_r\sin\theta)^{n_y}(v_z)^{n_z} \tag{3.12}$$

and $\mathcal{F}_p$ is the distribution function at a selected point near the circular cylinder.

3.1.1 Accommodating Boundary Conditions from the Cylinder Wall

At any point near the cylinder, the velocity distribution of particles passing through that point will be a combination of free-stream particles and reflected particles. Particles that are being reflected from the wall can experience a combination of specular and diffuse reflections [5]. In completely specular reflections, the reflected particles leave the surface with the same speed and an angle equal to the angle of impact. The distribution of incoming particles is simply reflected by flipping the unit normal to the wall. Figure 3.2 provides an example for the velocity distribution at a wall with a unit normal in the negative x direction for a specular reflection. In diffuse reflections, the reflected particles are fully accommodated by the cylinder. The particles come to thermal and momentum equilibrium with the cylinder wall and are re-emitted following Maxwell-Boltzmann statistics at the wall temperature and velocity. Figure 3.3 provides an example for the velocity profile of a diffuse reflection. Unlike the velocity profiles presented in the figures, the FMFC solver

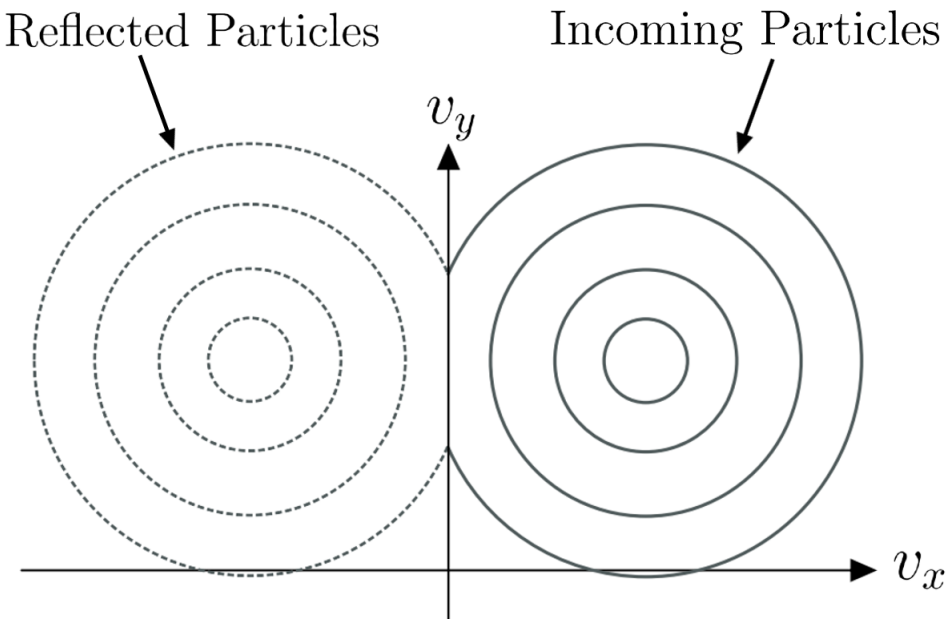


Figure 3.2: Velocity profile of specular reflections

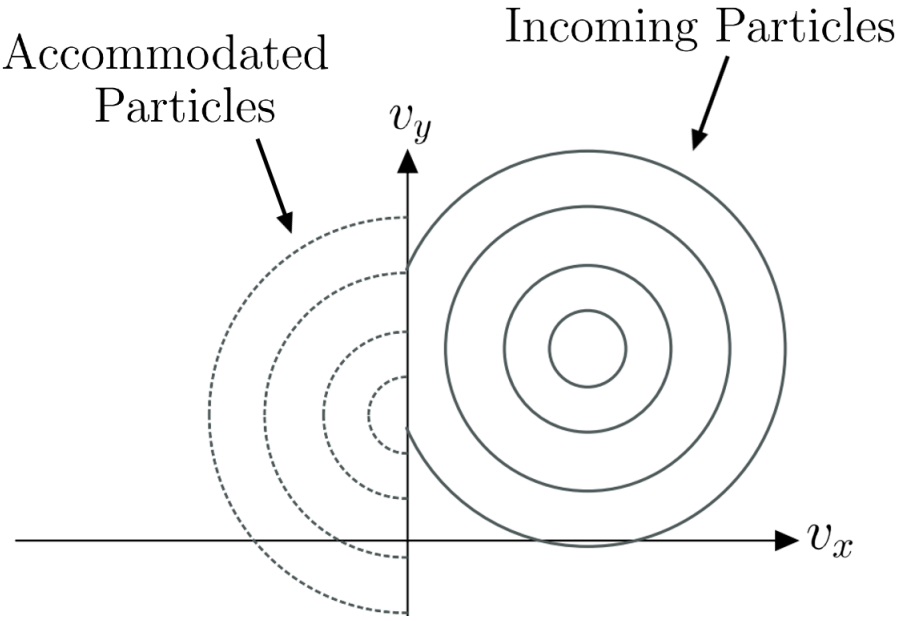


Figure 3.3: Velocity profile of diffuse reflections

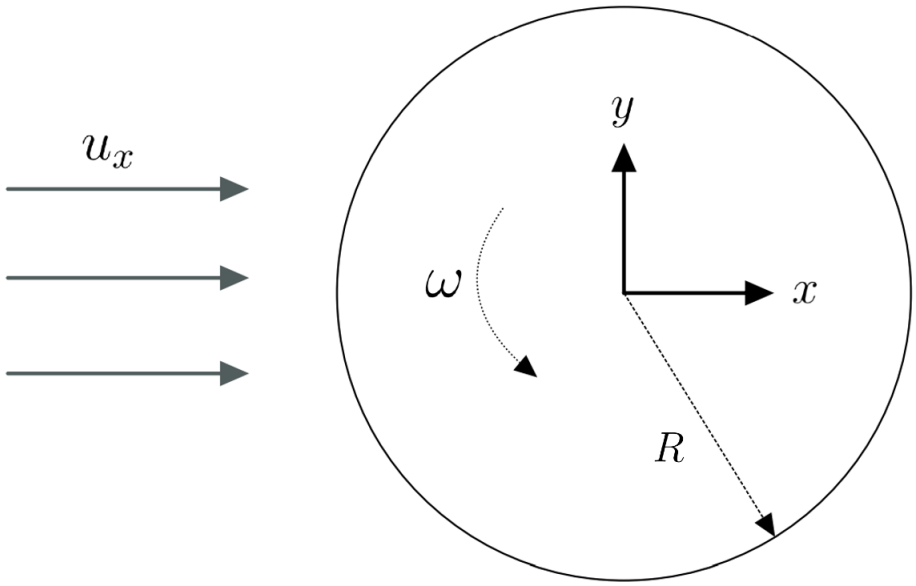


Figure 3.4: Free-molecular flow near a rotating cylinder

assumes an entirely x-direction incoming flow. The distribution function at any point near the cylinder is a combination of the free-stream and reflected particles. At a point on the cylinder wall normal to the incoming flow, the distribution function is

$$\mathcal{F}_P = \begin{cases} \mathcal{M}_{\mathrm{FS}}(v_x, v_y, v_z) & v_x > 0 \\ \alpha \mathcal{M}_{\mathrm{W}}(v_x, v_y, v_z) + (1-\alpha)\mathcal{M}_{FS}(-vx, vy, vz) & v_x < 0 \end{cases}, \tag{3.13}$$

where α is the accommodation coefficient, such that $\alpha = 0$ for reflections that are entirely specular and $\alpha = 1$ for reflections that are entirely diffuse.

3.1.2 Analytical Expressions for Lift and Drag

Deriving analytical solutions for the drag and lift produced by free-molecular flow near circular cylinders is important to confirm the accuracy of the FMFC solver that is presented in later sections of this chapter. Such solutions have been known for some time and derived by Patterson [23]. The present work extends these derivations to include rotating cylinders. Figure 3.4 presents the problem where there is a strictly x-direction free-stream flow with average velocity, u_x, that is impacting a cylinder of radius, R. Additionally, the cylinder is possibly rotating about its own axis with an angular velocity, ω. In order to calculate the drag and lift produced by the flow, the normal stress, σ_N, and tangential stress, σ_T, at any point on the cylinder surface must first be calculated. Figure 3.5 depicts a discretized element at some point on the cylinder surface. The figure also introduces a rotated coordinate system, adjusted for angle ψ, which is used to facilitate calculations. The angle ψ defines the direction normal to the point on the cylinder surface. Geometric definitions for this angle are presented in later sections of this chapter. Additionally, there are a few assumptions that must be made. There is no bulk velocity in the z-direction and particles

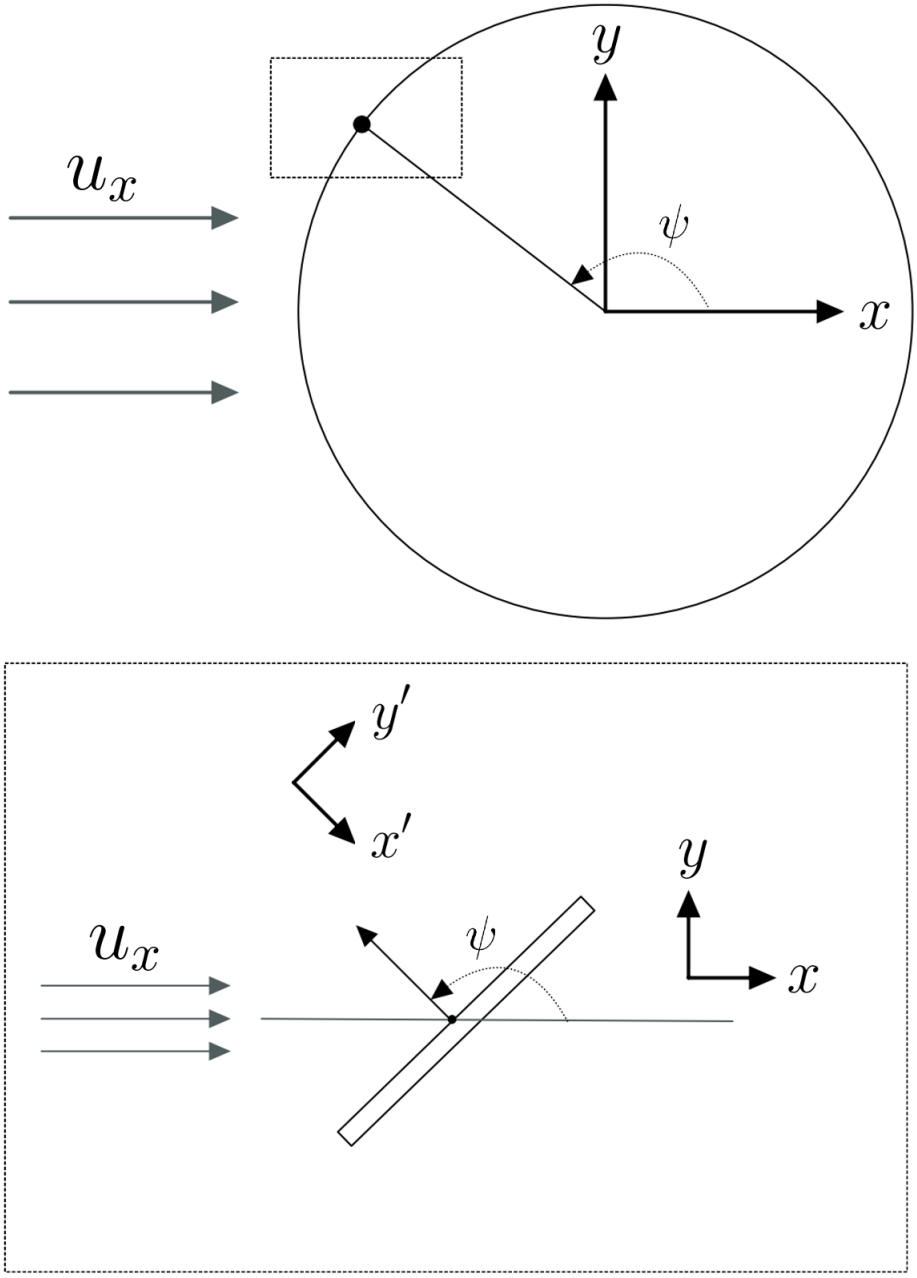


Figure 3.5: Discretized point on the cylinder surface with a rotated coordinated system

that are fully accommodated by the cylinder wall have an assumed x'-direction bulk velocity of zero and a y'-direction bulk velocity that is always equal to ωR. Reflected particles that are not accommodated by the wall are reflected by flipping the unit normal to the wall. Some parameters must also be defined to help simplify the equations. The number density for the free-stream, n_{FS}, and for the wall, n_{W}, are equal to

$$n_{\mathrm{FS}} = \frac{\rho_{\mathrm{FS}}}{m} \tag{3.14}$$

and

$$n_{\mathrm{W}} = \frac{\rho_{\mathrm{W}}}{m}. \tag{3.15}$$

Necessary variables for free-stream particles, β_{FS}, and reflected particles, β_{W}, are equal to

$$\beta_{\mathrm{FS}} = \frac{\rho_{\mathrm{FS}}}{2P_{\mathrm{FS}}} \tag{3.16}$$

and

$$\beta_{\mathrm{W}} = \frac{\rho_{\mathrm{W}}}{2P_{\mathrm{W}}}. \tag{3.17}$$

Here, the pressure of particles accommodated by the wall, P_{W}, is

$$P_{\mathrm{W}} = \frac{\rho_{\mathrm{W}} k T_{\mathrm{W}}}{m}, \tag{3.18}$$

where T_{W} is the temperature of the cylinder surface. Another helpful equation to define is the speed ratio, S, which is given by

$$S = u_x \sqrt{\beta_{\mathrm{FS}}}. \tag{3.19}$$

The speed ratio relates the bulk velocity of the free-stream to the statistically most probable random speed of a particle. It is also important to consider the new bulk velocities, u'_x and u'_y, of the free-stream that are adjusted for the rotated coordinate system. They are

$$u'_x = -u_x \cos\psi \tag{3.20}$$

and

$$u'_y = u_x \sin\psi. \tag{3.21}$$

The distribution of free-stream particles, $\mathcal{M}_{FS}$, in Equation (3.1) and the distribution of reflected particles that are fully accommodated by the wall, $\mathcal{M}_W$, in Equation (3.2) can now be described as

$$\mathcal{M}_{FS} = n_{\mathrm{FS}} \left(\frac{\beta_{\mathrm{FS}}}{\pi}\right)^{3/2} e^{-\beta_{\mathrm{FS}}((v'_x + u_x\cos\psi)^2 + (v'_y - u_y\sin\psi)^2 + v_z^2)} \tag{3.22}$$

and

$$\mathcal{M}_{\mathrm{W}} = n_{\mathrm{W}} \left(\frac{\beta_{\mathrm{W}}}{\pi}\right)^{3/2} e^{-\beta_{\mathrm{W}}((v'_x)^2 + (v'_y - \omega R)^2 + v_z^2)}. \tag{3.23}$$

The only remaining unknown in these expressions is the number density at the wall, n_{W}. For the gas not to penetrate the solid wall, the x'-direction flux through the cylinder wall is zero, such

that

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} v_x' \mathcal{F}_p \mathrm{d}v_x' \mathrm{d}v_y' \mathrm{d}v_z = 0. \tag{3.24}$$

Expanding Equation (3.24) and solving for the number density at the wall, n_W, yields

$$n_\mathrm{W} = \sqrt{\frac{\beta_\mathrm{W}}{\beta_\mathrm{FS}}} n_\mathrm{FS}(e^{-S^2\cos^2\psi} + \sqrt{\pi}(\mathrm{erf}(S\cos\psi) - 1)S\cos\psi). \tag{3.25}$$

The normal stress σ_N on the cylinder surface is the second-order x'-direction velocity moment,

$$\sigma_\mathrm{N} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} v_x'^2 \mathcal{F}_p \, \mathrm{d}v_x' \, \mathrm{d}v_y' \, \mathrm{d}v_z. \tag{3.26}$$

Expanding Equation (3.26) for the normal stress, σ_N, at the wall yields

$$\sigma_\mathrm{N} = \frac{m\alpha n_\mathrm{W}}{4\beta_\mathrm{W}} + \frac{n_\mathrm{FS} m(\alpha-2)(\cos\psi(Se^{-S^2\cos^2\psi}) + (S^2\cos^2\psi + \frac{1}{2})\sqrt{\pi}(\mathrm{erf}(S\cos\psi) - 1))}{2\sqrt{\pi}\beta_\mathrm{FS}}. \tag{3.27}$$

It is important to note that the derivation for the normal stress, σ_N, presented in this work differs from Patterson's original solution [23]. The term from Equation (3.27),

$$W_E = (S^2\cos^2\psi + \frac{1}{2}), \tag{3.28}$$

is different from Patterson's derivation, which has the term,

$$W_P = \frac{1}{2}(S^2\cos^2\psi + 1). \tag{3.29}$$

This misplaced factor of 2 in Patterson's expression introduces slight discrepancies in the final solution for the drag coefficient. Patterson's book is a classic in the field of kinetic theory. The derivation presented in the current work is correct and only aims to address this minor error. The final parameter to calculate, before evaluating drag and lift on the cylinder, is the tangential stress on the cylinder surface. The tangential stress, σ_T, on the cylinder surface is the second-order velocity moment, with x'-direction and y'-direction velocities applied to the distribution, such that

$$\sigma_\mathrm{T} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} v_x' v_y' \mathcal{F}_p \, \mathrm{d}v_x' \, \mathrm{d}v_y' \, \mathrm{d}v_z. \tag{3.30}$$

Expanding Equation (3.30) for the tangential stress, σ_{T}, at the wall yields

$$\begin{aligned}\sigma_{\mathrm{T}} =& \frac{\sin\psi\,(e^{-S^2\cos^2\psi}\sqrt{\pi} + \pi S\cos\psi(\operatorname{erf}(S\cos\psi)-1))\alpha u_x m n_{\mathrm{FS}}}{2\sqrt{\beta_{\mathrm{FS}}}\pi} \\ &+ \frac{R\alpha u_x m n_{\mathrm{FS}}\omega(e^{-S^2\cos^2\psi} + \sqrt{\pi}S\cos\psi(\operatorname{erf}(S\cos\psi)-1))}{2\sqrt{\beta_{\mathrm{FS}}}\sqrt{\pi}}.\end{aligned} \tag{3.31}$$

The drag and lift per unit area are defined to be

$$D = -\sigma_{\mathrm{N}}\cos\psi + \sigma_{\mathrm{T}}\sin\psi \tag{3.32}$$

and

$$L = -\sigma_{\mathrm{N}}\sin\psi - \sigma_{\mathrm{T}}\cos\psi, \tag{3.33}$$

respectively. The total drag force can then be computed as

$$F_D = \int_0^l \int_0^{2\pi} DR\,\mathrm{d}\psi\,\mathrm{d}z, \tag{3.34}$$

where l is the length of the cylinder. The drag force is also known to be

$$F_D = \rho v^2 C_D R l, \tag{3.35}$$

where C_D is the drag coefficient. By equating Equation (3.34) and Equation (3.35), the drag coefficient can be found as

$$C_D = \frac{1}{\rho v^2}\int_0^{2\pi} D\,\mathrm{d}\psi, \tag{3.36}$$

which comes out to be

$$\begin{aligned}C_D = \frac{1}{\rho_{\mathrm{FS}}u_x^2}\Bigg[& n_{\mathrm{FS}} m u_x\sqrt{\frac{\pi}{\beta_{\mathrm{FS}}}}\left(1-\frac{\alpha}{2}\right)\left(I_0\left(-\frac{S^2}{2}\right)+I_1\left(-\frac{S^2}{2}\right)\right) \\ &+ n_{\mathrm{FS}} m u_x^2\left(1-\frac{\alpha}{2}\right)\frac{S\sqrt{\pi}e^{-\frac{S^2}{2}}}{6}\left(9I_0\left(-\frac{S^2}{2}\right)-8I_1\left(-\frac{S^2}{2}\right)-I_2\left(-\frac{S^2}{2}\right)\right) \\ &+ \left(\frac{n_{\mathrm{FS}} m\alpha u_x}{4}\right)\sqrt{\frac{\pi}{\beta_{\mathrm{FS}}}}\pi \\ &+ \frac{n_{\mathrm{FS}} m}{2\beta_{\mathrm{FS}}}\left(1-\frac{\alpha}{2}\right)2S\sqrt{\pi}e^{-\frac{S^2}{2}}\left(I_0\left(-\frac{S^2}{2}\right)-I_1\left(-\frac{S^2}{2}\right)\right) \\ &+ \left(\frac{n_{\mathrm{FS}} m\alpha u_x}{2}\right)\sqrt{\frac{\pi}{\beta_{\mathrm{FS}}}}e^{-\frac{S^2}{2}}\left(I_0\left(-\frac{S^2}{2}\right)-I_1\left(-\frac{S^2}{2}\right)\right) \\ &+ \left(\frac{n_{\mathrm{FS}} m\alpha u_x^2}{2}\frac{S}{6}\sqrt{\pi}e^{-\frac{S^2}{2}}\right)\left(3I_0\left(-\frac{S^2}{2}\right)-4I_1\left(-\frac{S^2}{2}\right)+I_2\left(-\frac{S^2}{2}\right)\right)\Bigg].\end{aligned} \tag{3.37}$$

Here, I_0, I_1, and I_2 are modified Bessel functions of the first kind of orders zero, one, and two, respectively. The details of these integrations are available in Appendix A. Similar to the drag force, the lift force can be computed as

$$F_L = \int_0^l \int_0^{2\pi} LR \, \mathrm{d}\psi \, \mathrm{d}z. \tag{3.38}$$

The lift force is also known to be

$$F_L = \rho v^2 C_L R l, \tag{3.39}$$

where C_L is the lift coefficient. By equating Equation (3.38) and Equation (3.39), the lift coefficient C_L is

$$C_L = \frac{1}{\rho v^2} \int_0^{2\pi} L \, \mathrm{d}\psi. \tag{3.40}$$

From the above equation, C_L equates to

$$C_L = \alpha \left(\frac{R\omega\pi}{2u_x} \right). \tag{3.41}$$

If reflected particles are fully accommodated by the cylinder wall ($\alpha = 1$), the computed lift coefficient is

$$C_{L_{\text{Diffuse}}} = \frac{1}{2} \left(\frac{R\omega\pi}{u_x} \right) \tag{3.42}$$

and if reflected particles undergo completely specular reflections,

$$C_{L_{\text{Specular}}} = 0. \tag{3.43}$$

3.1.3 Steps Required to Construct the FMFC Solver

Chapter 1 discusses the importance of developing an accurate and efficient simulation method that computes the observable properties of free-molecular gas flows near convex shapes. Chapter 2 highlights important assumptions that are made in the kinetic theory of gases and how moments of the gas flow are evaluated. This chapter has presented an overview for the functionality of the FMFC solver, boundary conditions at the cylinder wall, and analytical expressions for the drag and lift coefficients that are used to verify the accuracy of the constructed FMFC solver. The following sections of the chapter present steps that lead to the construction of the FMFC solver. This includes development of the ray-tracing technique, important geometric relations for the solver, and derivations of highly efficient recursive relations that reduce computational runtimes by orders of magnitude. At any point near the cylinder, the solver must be able to accurately compute integrations of any moment of the distribution function using cylindrical coordinates. This includes integrations in the θ, v_r, and v_z directions. Development of this solver requires a set of important steps,

- A ray-tracing technique must be incorporated to define at any point near the cylinder, which particles passing through that point are from the free-stream or reflected from the cylinder wall.
- Geometric relations are derived to determine the point on the cylinder wall from which particles have been reflected. This is used to compute the unit normal at points tangent to the cylinder surface, the direction of velocities for reflected particles, and rotated coordinate systems that facilitate calculations.
- Recursion relations are developed to compute analytical solutions for integration steps in the v_r and v_z directions.
- A 5-point quadrature rule is used to compute numerical solutions for integration steps in the θ direction.

The following sections of this chapter highlight detailed explanations for each of these steps and how they are all used in conjunction with one another to construct the FMFC solver.

3.2 Ray-Tracing Technique

The restriction to convex shapes in the solver is to ensure that, when calculating moments at any point near the shape, there is strictly a combined distribution of particles from either the free-stream or free-stream particles that have undergone a single reflection from the wall. This means that at any point $p(x_p, y_p)$, when integrating the distribution function in θ from $0 \leq \theta \leq 2\pi$, a method is required that can distinguish between free-stream particles and reflected particles. This can be done by incorporating a ray-tracing technique. With this technique, the solver can always differentiate between free-stream and reflected particles. The relations in this work focus only on circular cylinders. Future work should extend the FMFC solver to use relations for any convex shape. Figure 3.6 demonstrates that, at point $p(x_p, y_p)$, all particles travelling in directions between the angles θ_1 and θ_2 have been reflected by the cylinder wall. Given that any particles between these two angles has been reflected from the cylinder wall, it is important to properly define the geometry involved when calculating these angular boundaries. The distance between the cylinder center and some point in space R_2 is

$$R_2 = \sqrt{x_p^2 + y_p^2}, \tag{3.44}$$

where x_p and y_p are the respective x and y positions of the point in reference to the cylinder center. The angular position of the point ϕ is also important and calculated as

$$\phi = \tan^{-1}\left(\frac{y_p}{x_p}\right). \tag{3.45}$$

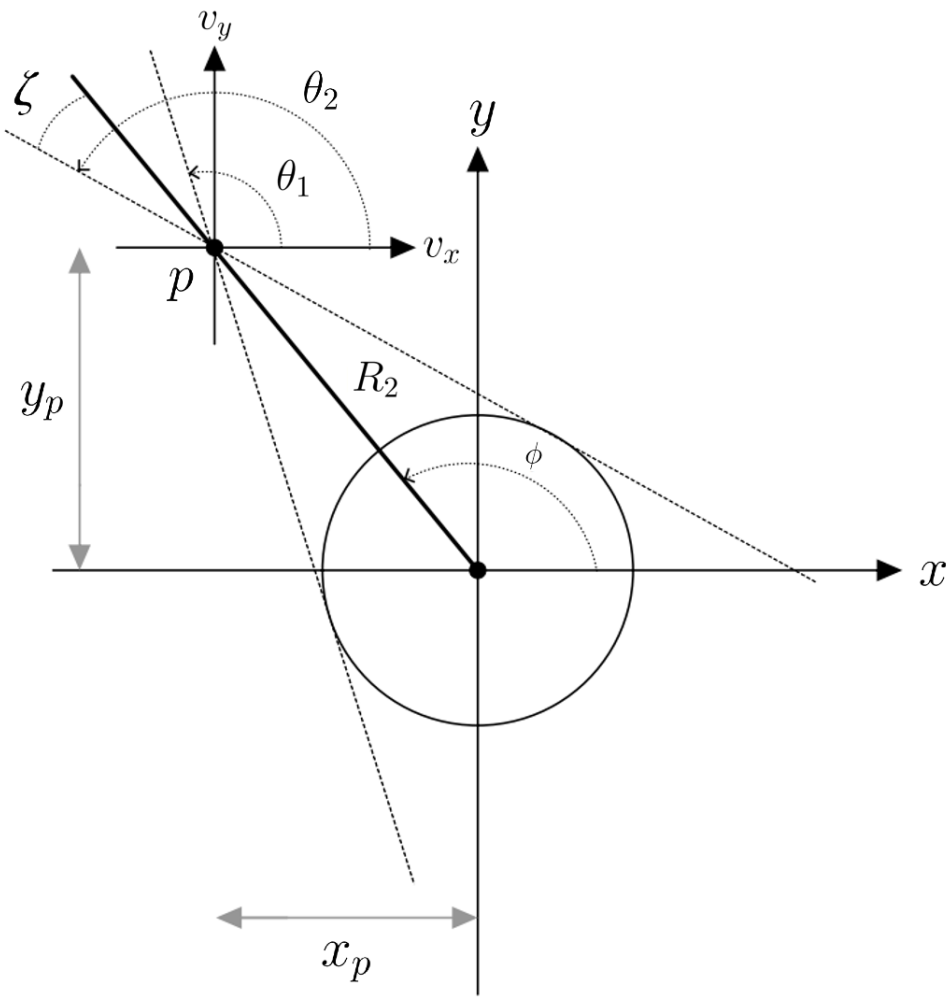


Figure 3.6: Distinguishing between free-stream particles and reflected particles.

Another relevant angle to analyze is the angular distance between the angular position of the point ϕ and the boundaries described by θ_1 and θ_2. This angle ζ is evaluated to be

$$\zeta = \sin^{-1}\left(\frac{R}{R_2}\right). \tag{3.46}$$

The information is used to compute both θ_1 and θ_2, which define the angular bounds between free-stream and reflected particles. As such, these angles are computed as,

$$\theta_1 = \phi - \zeta \tag{3.47}$$

and

$$\theta_2 = \phi + \zeta. \tag{3.48}$$

During the angular integration step, when evaluating moments of the distribution function, any angle θ such that $\theta_1 < \theta < \theta_2$ represents particles that have been reflected from the wall. The ray-tracing technique presented in this section is extremely important to correctly define the distribution function, $\mathcal{F}_P$, during the angular integration step. It is critical to correctly define free-stream and reflected particles to accurately compute moments of the distribution function.

3.3 Important Geometric Relations

When θ is bound between the two angles θ_1 and θ_2, there must exist some line, L_θ, that is defined by this angle and that intersects with the cylinder at the location from which particles are reflected. This angle and the corresponding intersecting line are depicted in Figure 3.7. The intersecting line L_θ is given by the linear function

$$L_\theta = m(x - x_p) + y_p, \tag{3.49}$$

where the slope, m, is equal to

$$m = \tan(\theta). \tag{3.50}$$

The boundaries of the cylinder surface, L_{cyl}, is defined with the function

$$L_{\text{cyl}} = \sqrt{R^2 - x^2}. \tag{3.51}$$

The two x-direction coordinates, x_1 and x_2, of the two intersecting points can be computed by equating Equation (3.49) and Equation (3.51), resulting in

$$x_{1,2} = \frac{-2m(-mx_p + y_p) \pm 2\sqrt{m^2r^2 - m^2x_p^2 + 2mx_py_p + r^2 - y_p^2}}{2m^2 + 2}. \tag{3.52}$$

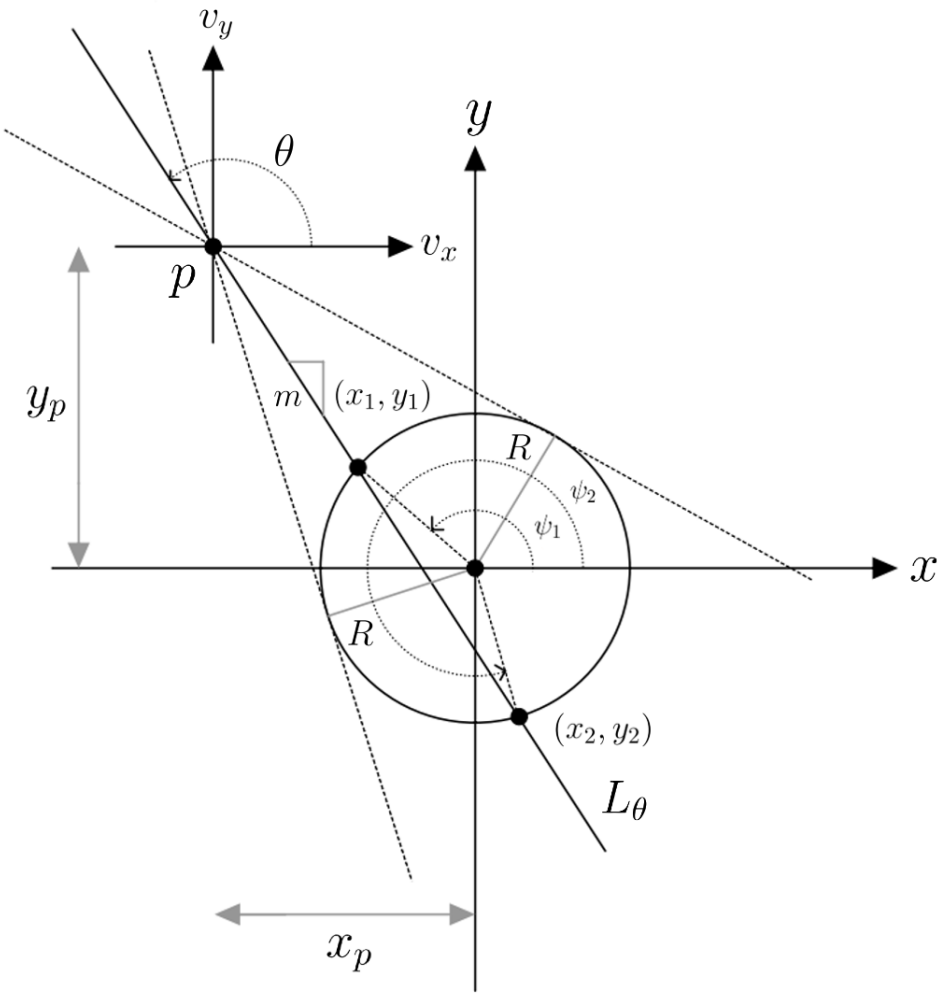


Figure 3.7: Geometric relations that trace reflected particles

Subsequently, the two y-direction coordinates, y_1 and y_2, of the two intersecting points can be computed by substituting Equation (3.52) into Equation (3.51), producing

$$y_1 = m(x_1 - x_p) + y_p \tag{3.53}$$

and

$$y_2 = m(x_2 - x_p) + y_p. \tag{3.54}$$

The angles, ψ_1 and ψ_2, for the two points of intersection are given by

$$\psi_1 = \tan^{-1}\left(\frac{y_1}{x_1}\right) \tag{3.55}$$

and

$$\psi_2 = \tan^{-1}\left(\frac{y_2}{x_2}\right). \tag{3.56}$$

As previously stated, the angle ψ is used to convert to the rotated coordinate system. Both ψ_1 and ψ_2 are angles that reference points of intersection with the cylinder. However, only one of these points is the location on the cylinder surface from which particles have been reflected towards the point of interest. The correct angle, ψ, is selected such that

$$\psi = \begin{cases} \psi_1 & \sqrt{(x_p - x_1)^2 + (y_p - y_1)^2} \leq \sqrt{(x_p - x_2)^2 + (y_p - y_2)^2} \\ \psi_2 & \text{otherwise} \end{cases}. \tag{3.57}$$

Figure 3.8 presents the point of reflection on the cylinder wall, the reference angle ψ, and a representation of the rotated coordinate system. The angle ψ is used to define the rotated coordinate system and the direction of the unit normal, which is perpendicular to the tangent of the cylinder surface.

3.4 Recursion Relations

In this solver, the ray-tracing technique is used to distinguish between free-stream and reflected particles at points near the cylinder. Additional geometric relations are also developed to work in a rotated coordinate system for reflected particles, further simplifying calculations. Moments of the distribution function, $\mathcal{F}_p$, are computed for any point near the circular cylinder to determine specific observable properties of the gas flow. In Cartesian coordinates, the general form of the distribution $\mathcal{M}$, where $u_z = 0$, is

$$\mathcal{M} = n\left(\frac{\beta}{\pi}\right)^{3/2} e^{-\beta((v_x - u_x)^2 + (v_z - u_z)^2 + v_z^2)} \tag{3.58}$$

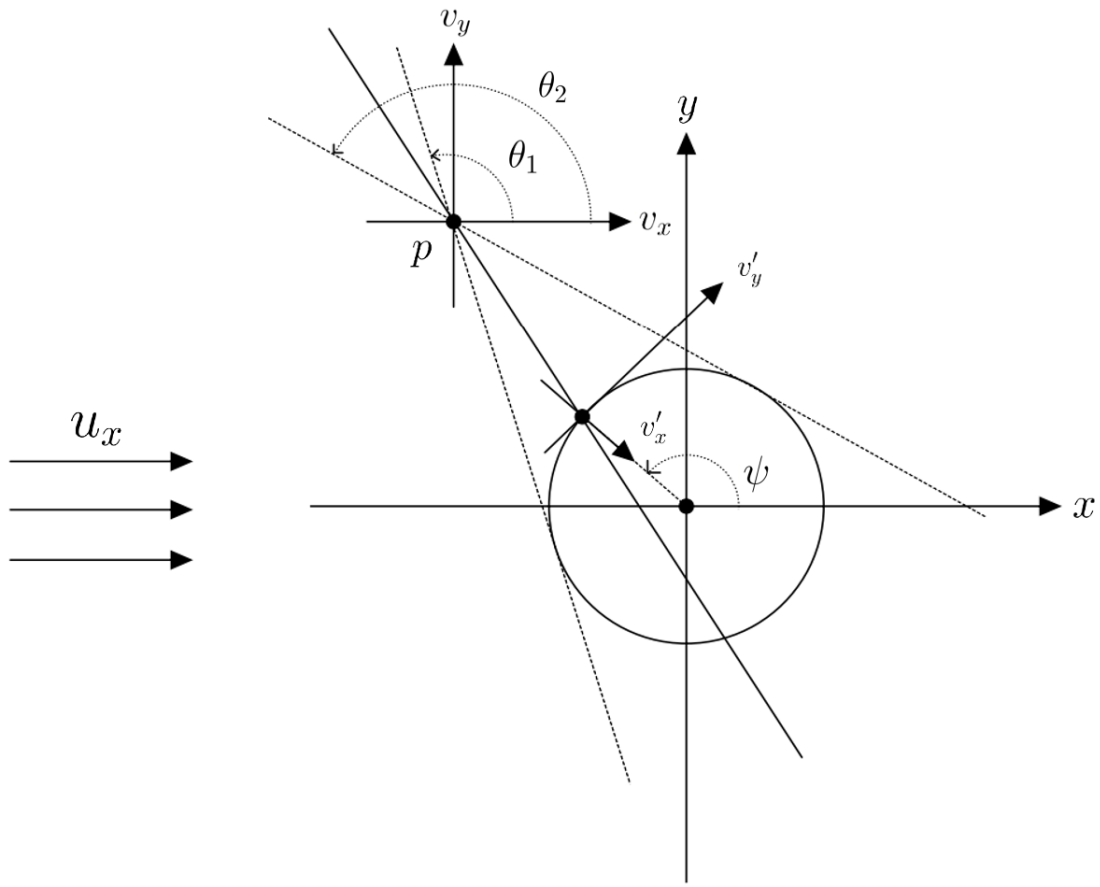


Figure 3.8: Analysis of the parameters in a rotated coordinate system

and the velocity moment of this distribution, for any velocity moment, takes the form

$$\langle m v_x^{n_x} v_y^{n_y} v_z^{n_z} \mathcal{M} \rangle = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} v_x^{n_x} v_y^{n_y} v_z^{n_z} \rho \left(\frac{\beta}{\pi}\right)^{3/2} e^{-\beta((v_x-u_x)^2+(v_y-u_y)^2+(v_z)^2)} \, \mathrm{d}v_x \, \mathrm{d}v_y \, \mathrm{d}v_z. \tag{3.59}$$

In cylindrical coordinates, with reference to Equation (3.11) and Equation (3.12), the moments of the distribution function can be expressed as

$$\langle m v_x^{n_x} v_y^{n_y} v_z^{n_z} \mathcal{M} \rangle = \int_{-\infty}^{\infty} \int_{0}^{\infty} \int_{0}^{2\pi} M_{\mathrm{cyl}} \rho \left(\frac{\beta}{\pi}\right)^{3/2} e^{-\beta\left((v_r \cos\theta-u_x)^2+(v_r \sin\theta-u_y)^2+v_z^2\right)} v_r \, \mathrm{d}\theta \, \mathrm{d}v_r \, \mathrm{d}v_z. \tag{3.60}$$

Various terms in this equation can also be isolated and reorganized in order to simplify the integration steps. Constant terms in the equation are represented by f_{c}, where

$$f_{\mathrm{c}} = \rho \left(\frac{\beta}{\pi}\right)^{3/2} e^{-\beta(u_x^2+u_y^2)}, \tag{3.61}$$

velocity terms in the z-direction are represented by f_{z}, where

$$f_{\mathrm{z}} = \int_{-\infty}^{\infty} v_z^{n_z} e^{-\beta v_z^2} \mathrm{d}v_z, \tag{3.62}$$

velocity terms in the radial direction are represented by f_{r}, where

$$f_{\mathrm{r}} = e^{-\beta(u_x \sin\theta+u_y \sin\theta)} \int_{0}^{\infty} v_r^{n_x+n_y+1} e^{-\beta(v_r-(u_x \sin\theta+u_y \sin\theta))^2} \, \mathrm{d}v_r, \tag{3.63}$$

and angular terms defined by θ are represented by f_θ, where

$$f_\theta = \int_{0}^{2\pi} v_r (\cos\theta)^{n_x} (\sin\theta)^{n_y} f_{\mathrm{r}} \, \mathrm{d}\theta, \tag{3.64}$$

such that f_θ depends on f_{r}. The velocity moments of the distribution in Equation (3.60) can now be expressed as a simplified combination of f_{c}, f_{z}, and f_θ, where

$$\langle m v_x^{n_x} v_y^{n_y} v_z^{n_z} M \rangle = f_{\mathrm{c}} f_{\mathrm{z}} f_\theta. \tag{3.65}$$

The mathematically efficient approach to this free-molecular solver comes from the way that integration is handled for integration steps that involve the v_z and v_r terms. The moments of the distribution function have been generalized for any moment that is selected for any point near

the circular cylinder. Recursion relations are developed to analytically compute integrations in the v_z and v_r directions. By eliminating the need for numerical integrations in these directions, computing any moment of the distribution function becomes a highly efficient computational task.

3.4.1 Centered Maxwell-Boltzmann Recursion Relation

The centered Maxwell-Boltzmann recursion relation is used to analytically solve f_z. This drastically reduces computational runtimes because it eliminates numerical integration for the v_z term. From the fundamental theorem of calculus,

$$\int_{-\infty}^{\infty} \frac{\mathrm{d}}{\mathrm{d}v}[g(v)]\mathrm{d}v = g(\infty) - g(-\infty). \tag{3.66}$$

Here, the function g is

$$g(v) = v^n \cdot F(v), \tag{3.67}$$

where F is a function of v. By setting v to be

$$v = v_z, \tag{3.68}$$

F to be

$$F(v_z) = e^{-\beta v_z^2}, \tag{3.69}$$

and n to be

$$n = n_z, \tag{3.70}$$

Equation (3.66) converges to zero because the selected function, $F(v)$, follows the same statistical probabilities as the distribution f_z. The probability of velocities in the distribution that approach $-\infty$ and $+\infty$ is zero, giving

$$\int_{-\infty}^{\infty} \frac{\mathrm{d}}{\mathrm{d}v_z}[v_z^{n_z} e^{-\beta v_z^2}]\,\mathrm{d}v_z = 0. \tag{3.71}$$

From the product rule for a derivative, the above equation becomes

$$\int_{-\infty}^{\infty} \left[n_z v_z^{n_z-1} e^{-\beta v_z^2} - 2\beta v_z^{n_z+1} e^{-\beta v_z^2}\right]\,\mathrm{d}v_z = 0. \tag{3.72}$$

Solving this equation for a recursion relation results in

$$f_{n_z} = \frac{(n_z - 1) f_{n_z-2}}{2\beta}, \tag{3.73}$$

where

$$f_{\mathrm{n_z}} = \int_{-\infty}^{\infty} v_z^{n_z} e^{-\beta v_z^2} \mathrm{d}v_z. \tag{3.74}$$

In the relation, $f_{\mathrm{n_z}}$ represents a selected velocity moment of v_z that is raised to a power of n_z. The recursion relation depends on the previous two moments to be constructed with the base case being

$$f_0 = \int_{-\infty}^{\infty} F \mathrm{d}v = \sqrt{\frac{\pi}{\beta}} \tag{3.75}$$

and

$$f_1 = \int_{-\infty}^{\infty} vF \mathrm{d}v = 0. \tag{3.76}$$

This completes the recursion relation for the solution of Equation (3.62). The relation can be used to continually find higher-order moments of the distribution function. For example, the second-order moment, f_2, is

$$f_2 = \frac{f_0}{2\beta} = \frac{\sqrt{\pi}}{2\beta^{3/2}} \tag{3.77}$$

and the third-order moment, f_3, is

$$f_3 = \frac{2f_1}{2\beta} = 0. \tag{3.78}$$

The relation can continue to be used recursively to solve the exact analytical solution for any moment of a centered Maxwell-Boltzmann integral.

3.4.2 Shifted Maxwell-Boltzmann Recursion Relation

The shifted Maxwell-Boltzmann recursion relation is used to analytically solve f_{r}. Just like the moment-centered Maxwell Boltzmann recursion relation, this relation is used for computational efficiency and accuracy. Again, from the fundamental theorem of calculus,

$$\int_0^{\infty} \frac{\mathrm{d}}{\mathrm{d}v}[g(v)]\mathrm{d}v = g(\infty) - g(0). \tag{3.79}$$

Here, the function g is

$$g(v) = v^n \cdot F(v), \tag{3.80}$$

where F is a function of v. The term v is set to be

$$v = v_r \tag{3.81}$$

and the variable A is considered to be a constant term relative to v_{r}, where

$$A = u_x \cos\theta + u_y \sin\theta. \tag{3.82}$$

By setting F to be

$$F = e^{-\beta(v_r - A)^2} \tag{3.83}$$

and n to be

$$n = n_x + n_y + 1 = n_r, \tag{3.84}$$

Equation (3.79) takes a form that resembles that of f_r in Equation (3.63). Since the selected function, F, shares the same statistical probabilities as the distribution f_r, the probability of velocities in the distribution that approach $+\infty$ is zero, giving

$$\int_0^\infty \frac{\mathrm{d}}{\mathrm{d}v}[v_r^{n_r} e^{-\beta(v_r - A)^2}]dv = 0. \tag{3.85}$$

From the product rule for a derivative, the above equation becomes

$$\int_0^\infty \left[n_r v_r^{n_r - 1} e^{-\beta(v_r - A)^2} - 2\beta(v_r - A) v_r^{n_r} e^{-\beta(v_r - A)^2} \right] dv_r = 0. \tag{3.86}$$

Solving this equation for a recursion relation results in

$$f_{\mathrm{n_r}} = \frac{2A\beta f_{\mathrm{n_r}-1} + (\mathrm{n_r} - 1) f_{\mathrm{n_r}-2}}{2\beta}, \tag{3.87}$$

where

$$f_{\mathrm{n_r}} = \int_0^\infty v_r^{n_r} e^{-\beta(v_r - A)^2} \mathrm{d}v_z. \tag{3.88}$$

In the relation, $f_{\mathrm{n_r}}$ represents a selected velocity moment of v_r that is raised to a power of n_r. The recursion relation depends on the previous two moments to be constructed with the base case being

$$f_0 = \int_0^\infty F \mathrm{d}v = \frac{\sqrt{\pi}(\mathrm{erf}(\sqrt{\beta}A) + 1)}{2\sqrt{\beta}} \tag{3.89}$$

and

$$f_1 = \int_0^\infty vF \mathrm{d}v = \frac{e^{-\beta A^2}\sqrt{\beta} + A\beta\sqrt{\pi}(\mathrm{erf}(\sqrt{\beta}A) + 1)}{2\beta^{\frac{3}{2}}}. \tag{3.90}$$

This completes the recursion relation for the solution of Equation (3.63). This efficient relation is used to find higher-order moments of the distribution function. For example, the second-order moment, f_2, is

$$f_2 = \frac{2A\beta f_1 + (1) f_0}{2\beta} = \frac{Ae^{-\beta A^2}\sqrt{\beta} + \left(\beta A^2 + \frac{1}{2}\right)\sqrt{\pi}\left(\mathrm{erf}(\sqrt{\beta}A) + 1\right)}{2\beta^{3/2}} \tag{3.91}$$

and the third-order moment, f_3, is

$$f_3 = \frac{2A\beta f_2 + (2) f_1}{2\beta} = \frac{\left(\beta^{3/2}A^2 + \sqrt{\beta}\right) e^{-\beta A^2} + \left(\operatorname{erf}(\sqrt{\beta}A) + 1\right) \beta A \sqrt{\pi} \left(\beta A^2 + \frac{3}{2}\right)}{2\beta^{5/2}}. \tag{3.92}$$

Continuing to apply this recursion relation enables any higher-order moment of a shifted Maxwell-Boltzmann integral to be solved. It is worth noting that the remaining integral for θ is computed numerically as there is no obvious analytical solution for the f_θ term.

3.5 Implementing the FMFC Solver

The FMFC solver is implemented in a C++ programming environment with a standard CPU version of the code and a version of the code that has been developed to target GPUs with SYCL. The solver takes as input positional variables x_p and y_p to define the point $p(x_p, y_p)$. It also requires information about the cylinder, such as the radius, R, wall temperature, T_w, angular velocity, ω, and accommodation coefficient, α. Information about the free-stream is provided as well. This includes the gas density, ρ_{FS}, pressure, P_{FS}, bulk velocity, u_x, and particle mass, m. Inputs for the defined moments n_x, n_y, and n_z are also required to determine which moments of the flow will be evaluated. Geometric inputs are used to compute the angular bounds θ_1 and θ_2 to distinguish between free-stream and reflected particles. Integration steps are completed with the centered and shifted Maxwell-Boltzmann recursion relations for the v_z and v_r directions. The integral in the θ direction is computed numerically with a 5-point Gaussian quadrature rule. As such, the solver also takes an argument for the number of discretized segments in the angular direction, θ. Figure 3.9 illustrates the required inputs to the FMFC solver. All computational results in this section of the thesis are produced for completely diffuse reflections, such that $\alpha = 1$.

3.5.1 Verifying the Solver

In order to verify the accuracy of the solver, comparisons against analytical solutions for the drag and lift coefficients are conducted. This is done by using the solver to compute the solution on a series of points that form a control volume around the rotating cylinder. The described approach is depicted in Figure 3.10. In the figure, G_{left}, G_{right}, G_{top}, and G_{bottom} are relevant momentum fluxes that are integrated along the boundaries of the control volume. From the Eulerian form of Newton's second law, the total force $\sum \overrightarrow{F}$ acting on the control volume is given by

$$\sum \overrightarrow{F} = \frac{\mathrm{d}}{\mathrm{d}t} \iiint_V \rho \overrightarrow{v} \, \mathrm{d}V + \oint_S \rho \overrightarrow{v}\overrightarrow{v} \cdot \hat{n} \, \mathrm{d}S. \tag{3.93}$$

For steady-state flow, the rate of change of momentum within the control volume is zero, such that

$$\frac{d}{dt} \iiint_V \rho \overrightarrow{v} \, \mathrm{d}V = 0, \tag{3.94}$$

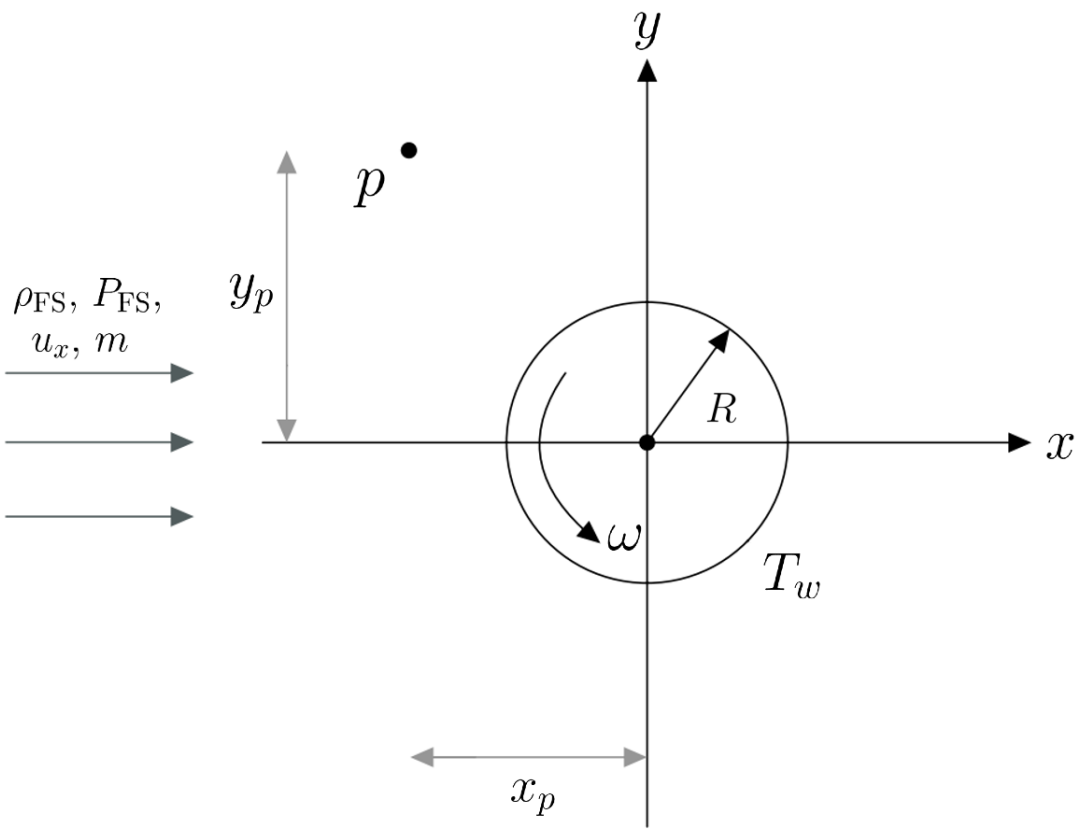


Figure 3.9: Relevant input variables for the FMFC solver

and, in this situation,

$$\sum \overrightarrow{F} = \oint_S \rho \overrightarrow{v}\overrightarrow{v} \cdot \hat{n} \, \mathrm{d}S. \tag{3.95}$$

From the equation for the coefficient of drag,

$$C_D = \frac{\sum \overrightarrow{F}}{\rho u_x^2 R} \tag{3.96}$$

and based on the directions of the forces acting on the control volume,

$$\sum \overrightarrow{F} = G_{\text{left}} - G_{\text{right}} + G_{\text{bottom}} - G_{\text{top}}, \tag{3.97}$$

it is possible to use the solver to calculate drag coefficients. The lift coefficient is similarly computed as

$$C_L = \frac{\sum \overrightarrow{F}}{\rho u_x^2 R} = \frac{\langle v_x v_y m\mathcal{F} \rangle - \langle v_x v_y m\mathcal{F} \rangle + \langle v_y^2 m\mathcal{F} \rangle - \langle v_y^2 m\mathcal{F} \rangle}{\rho u_x^2 R}. \tag{3.98}$$

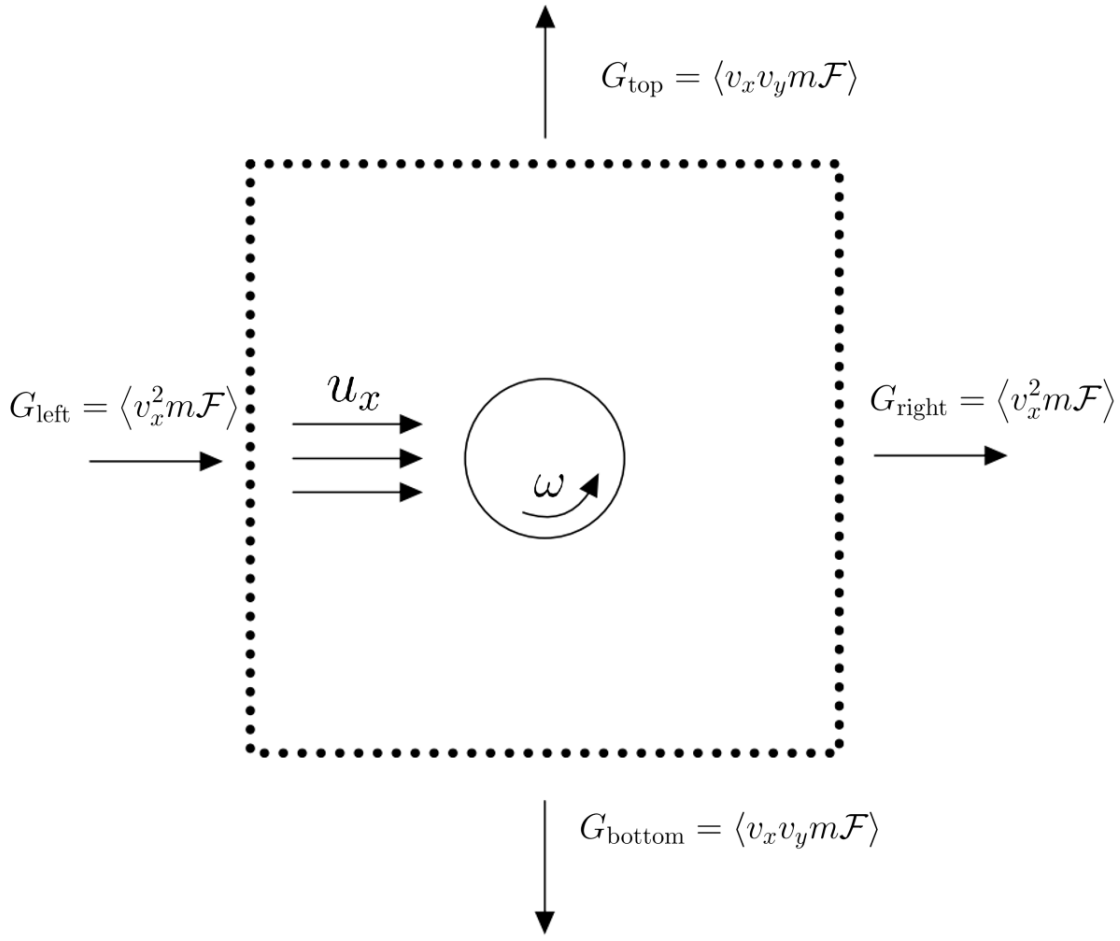


Figure 3.10: Fluxes in the x-direction that are coming into and out of a control volume

Table 3.1: Drag and lift coefficients calculated with the FMFC solver and compared against analytical solutions

θ **Segments**	C_D **Solver**	C_D **Difference**	C_L **Solver**	C_L **Difference**
10	5.1079093488	$8.9441 \cdot 10^{-5}$	0.2617993878	0
50	5.1078278542	$7.9514 \cdot 10^{-6}$	0.2617993878	0
100	5.1078226432	$2.7404 \cdot 10^{-6}$	0.2617993878	0
200	5.1078207988	$8.96 \cdot 10^{-7}$	0.2617993878	0
400	5.1078201464	$2.436 \cdot 10^{-7}$	0.2617993878	0
800	5.1078199157	$1.29 \cdot 10^{-8}$	0.2617993878	0

Table 3.2: Drag and lift coefficients calculated with the FMFC solver and compared against analytical solutions

θ **Segments**	C_D **Solver**	C_D **Difference**	C_L **Solver**	C_L **Difference**
10	5.1079093488	$8.9441 \cdot 10^{-5}$	0.2617993878	0
200	5.1078207988	$8.96 \cdot 10^{-7}$	0.2617993878	0
800	5.1078199157	$1.29 \cdot 10^{-8}$	0.2617993878	0

The verification test problem is conducted for the input conditions,

$$\mathbf{W_1} = \begin{bmatrix} \rho \\ P \\ u_x \\ m \\ R \\ T_w \\ \omega \end{bmatrix} = \begin{bmatrix} 1.7838\,\mathrm{kg/m^3} \\ 101325\,\mathrm{Pa} \\ 300.0\,\mathrm{m/s} \\ 6.63 \times 10^{-26}\,\mathrm{kg} \\ 1.0\,\mathrm{m} \\ 273.15\,\mathrm{k} \\ 50\,\mathrm{rad/s} \end{bmatrix}. \tag{3.99}$$

A 5-point Gaussian quadrature rule is used with 1000 discretized segments on each spatial boundary of the control volume. The angular integral is computed using a 5-point Gaussian quadrature rule and a varying number of discretized θ segments. The analytical result for the drag coefficient in the defined problem is

$$C_D = 5.1078199028 \tag{3.100}$$

and the lift coefficient is

$$C_L = 0.2617993878. \tag{3.101}$$

Table 3.2 shows results computed with the solver in comparison to analytical values.

From the table, it is evident that the drag coefficient, C_D, computed with the solver converges relatively quickly towards the analytical solution. Additionally, the lift coefficient converges independently of the θ resolution because the lift coefficient, C_L, does not depend on the θ component. It is also clear that accurate results are achievable with a small number of θ segments. However, higher accuracies are achieved with an increasing resolution in the θ direction. Interestingly enough, upon further investigation, eliminating cylinder rotation does not affect the resultant

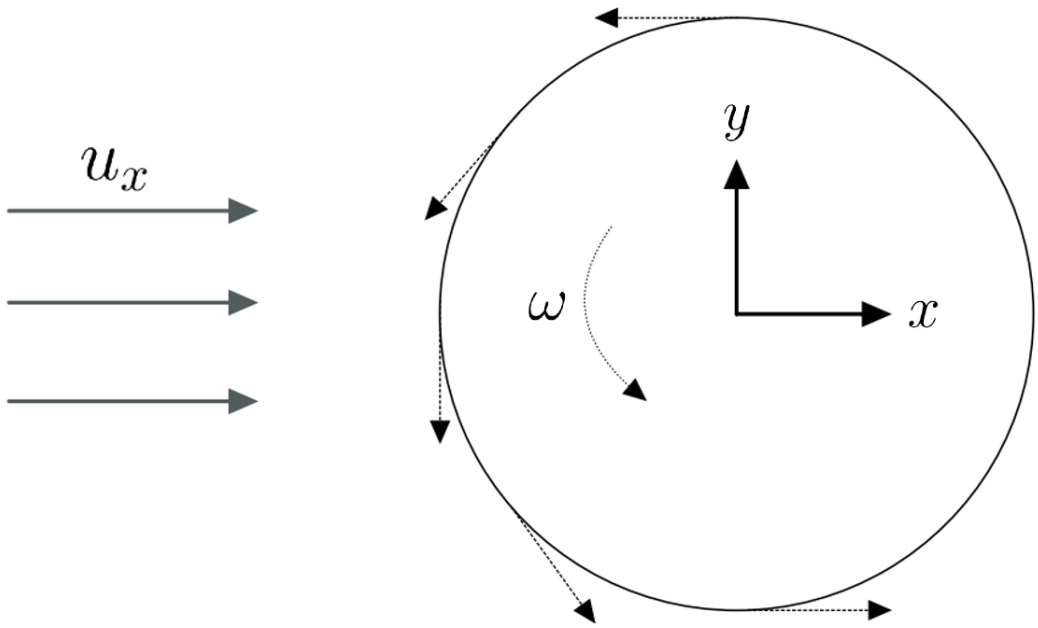


Figure 3.11: Development of the total drag on a rotating cylinder

drag coefficient. Repeating the same problem and removing angular rotation, the resulting drag coefficient is

$$C_D = 5.1078199078 \tag{3.102}$$

and the resulting lift coefficient is

$$C_L = 0.0. \tag{3.103}$$

Rotation produces lift because the higher density of particles that impact the front of the cylinder are reflected in the positive-y and negative-y directions. However, rotation does not affect drag because particles impacting the front of the cylinder are equally being reflected in the positive-x and negative-x directions. As a result, rotation creates flux in the y direction but has a net zero effect in the x direction. The described effect is presented in Figure 3.11.

3.5.2 Field Implementation

While it can be useful to use the FMFC solver to individually calculate moments of the distribution at any point near the cylinder, it is highly desirable to extend the solver to compute an entire field of moments around the circular cylinder. The functionality for entire field calculations was implemented for both a CPU and GPU version of the code. The field calculation sets up a complete circular mesh around the rotating cylinder, with discretizations in the angular and radial directions. In each cell of the discretized mesh, the solver computes all desired observable properties at the cell centroid. Figure 3.12 provides a representation of a circular mesh around a cylinder with radial discretizations, angular discretizations, and highlighted cell centroids. Computing moments of the distribution at each individual cell centroid is completely independent of all other centroids. This means that resolving the entire field is an embarrassingly parallel prob-

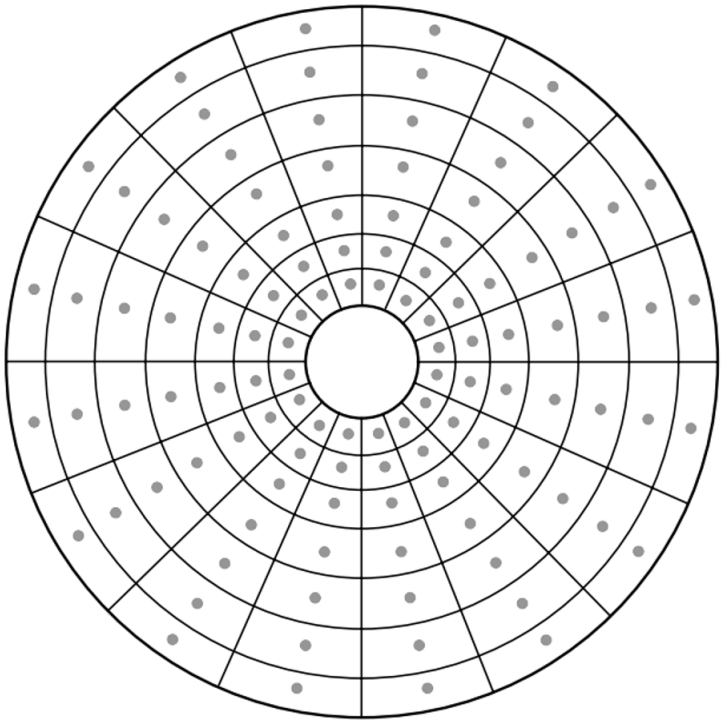
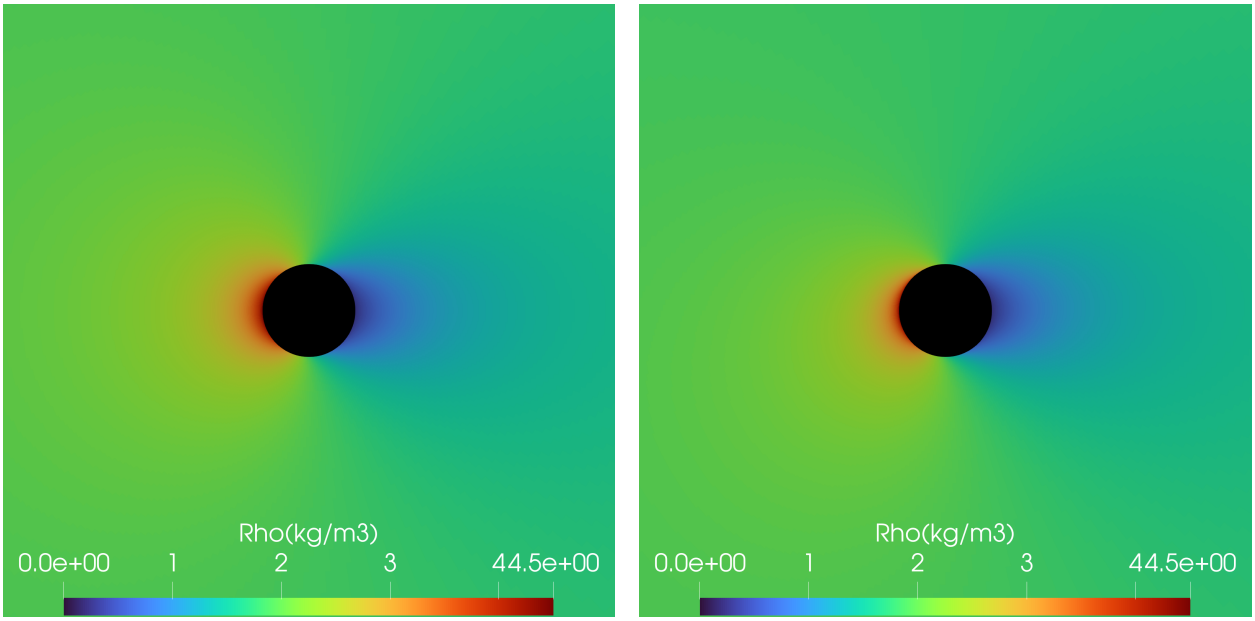


Figure 3.12: Output field for the FMFC solver

lem and should scale very well to parallel platforms. The CPU version of the code makes use of OpenMP, an open-source multi-core processing interface that can be used directly within the C++ programming environment to seamlessly write parallel programming directives for multi-core processors [24]. By leveraging OpenMP functions, the CPU implementation of the FMFC solver can distribute the embarrassingly parallel workload across all available cores. The GPU version of the code is developed in a C++ programming environment with the SYCL open standard to target hardware accelerators. The SYCL framework is used to write GPU kernels that distribute the global problem across all available GPU threads.

3.6 Computational Results from the FMFC Solver

This section of the chapter highlights various computational results produced with the FMFC solver. This is followed with discussions of interesting scientific results and performance comparisons between server-grade CPUs and GPUs. A few distinct scientific cases are produced with the FMFC solver and visualized on a two-dimensional grid with a resolution of 300 cells in the radial direction and 300 cells in the angular direction. This solver is designed for the application of monatomic gasses and all test cases assume this condition.



(a) cylinder with no rotation

(b) cylinder with rotation

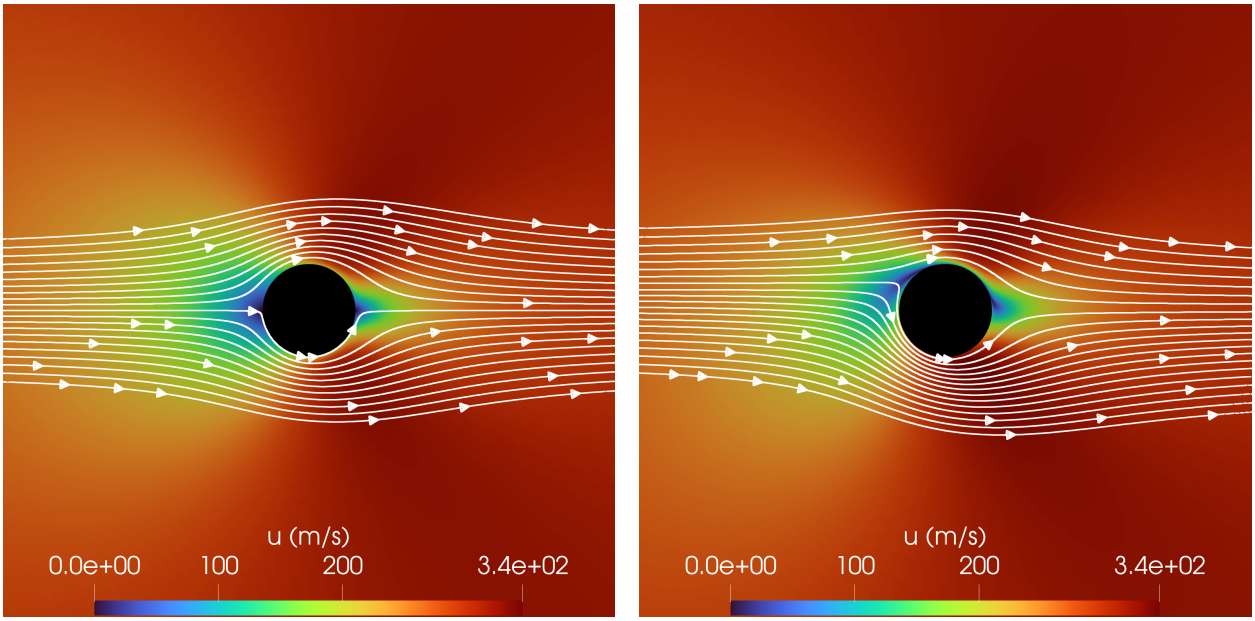
Figure 3.13: Density comparison: cylinder with no rotation vs. cylinder with rotation

3.6.1 High-Speed Flow

Two test cases are studied with the input parameters

$$\mathbf{W_1} = \begin{bmatrix} \rho \\ P \\ u_x \\ m \\ R \\ T_w \\ \omega \end{bmatrix} = \begin{bmatrix} 1.7838\,\text{kg/m}^3 \\ 101325\,\text{Pa} \\ 320\,\text{m/s} \\ 6.63 \times 10^{-26}\,\text{kg} \\ 1\,\text{m} \\ 293.15\,\text{k} \\ 0\,\text{rad/s} \end{bmatrix} \quad \text{and} \quad \mathbf{W_2} = \begin{bmatrix} \rho \\ P \\ u_x \\ m \\ R \\ T_w \\ \omega \end{bmatrix} = \begin{bmatrix} 1.7838\,\text{kg/m}^3 \\ 101325\,\text{Pa} \\ 308\,\text{m/s} \\ 6.63 \times 10^{-26}\,\text{kg} \\ 1\,\text{m} \\ 293.15\,\text{k} \\ 320\,\text{rad/s} \end{bmatrix}. \tag{3.104}$$

Figure 3.13 shows that in both cases, as expected, there are larger particle densities near the point of impact where the flow meets the cylinder wall. It is also interesting to see that, in Figure 3.13b, the distribution of particle densities is not greatly affected by the rotation of the cylinder. The representation of bulk velocities are shown in Figure 3.14. In Figure 3.14a, there is the formation of stagnation points at both the front and back of the cylinder. This effect is expected. In Figure 3.14a, there is also the formation of stagnation points; however, the bulk velocity is no longer symmetric and has been distorted with the cylinder rotation. Another interesting observation to note is that the no-slip condition is no longer valid in free-molecular flow. This is evident because the velocity of the flow does not reach zero at either the top or bottom of the cylinder wall. In free-molecular flow, particles do not closely adhere to the surface and so the no-slip condition loses validity. When analyzing the velocity profiles of the flow, it is also clear that the bulk velocity is significantly affected by cylinder rotation. The heat-flux in each of the two cases



(a) cylinder with no rotation

(b) cylinder with rotation

Figure 3.14: Bulk velocity comparison: cylinder with no rotation vs. cylinder with rotation

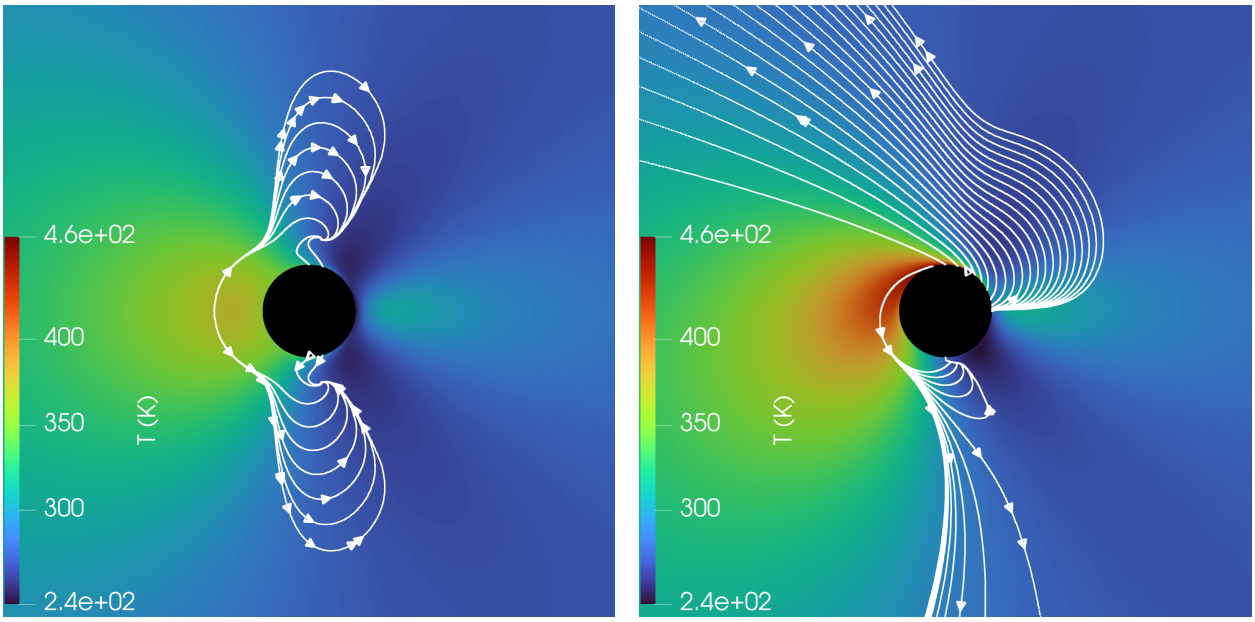
was also studied. Figure 3.15 visualizes the heat-flux as an overlay to the scalar temperature field, as defined in Equation (2.13). Without rotation, the heat-flux is extremely concentrated at the point of impact with the cylinder wall, as shown in Figure 3.15a. In Figure 3.15b, there is some strong concentration at the point of impact with the cylinder. However, when rotation is a factor, there is a broad plume-like structure that follows the direction of rotation.

3.6.2 Counter-Gradient Heat-Flux

Another interesting deviation from continuum fluid dynamics is presented when further investigating the direction of the heat-flux vector as an overlay to the scalar temperature field of the gas flow. The two test cases, $\mathbf{W_1}$ and $\mathbf{W_2}$, follow that

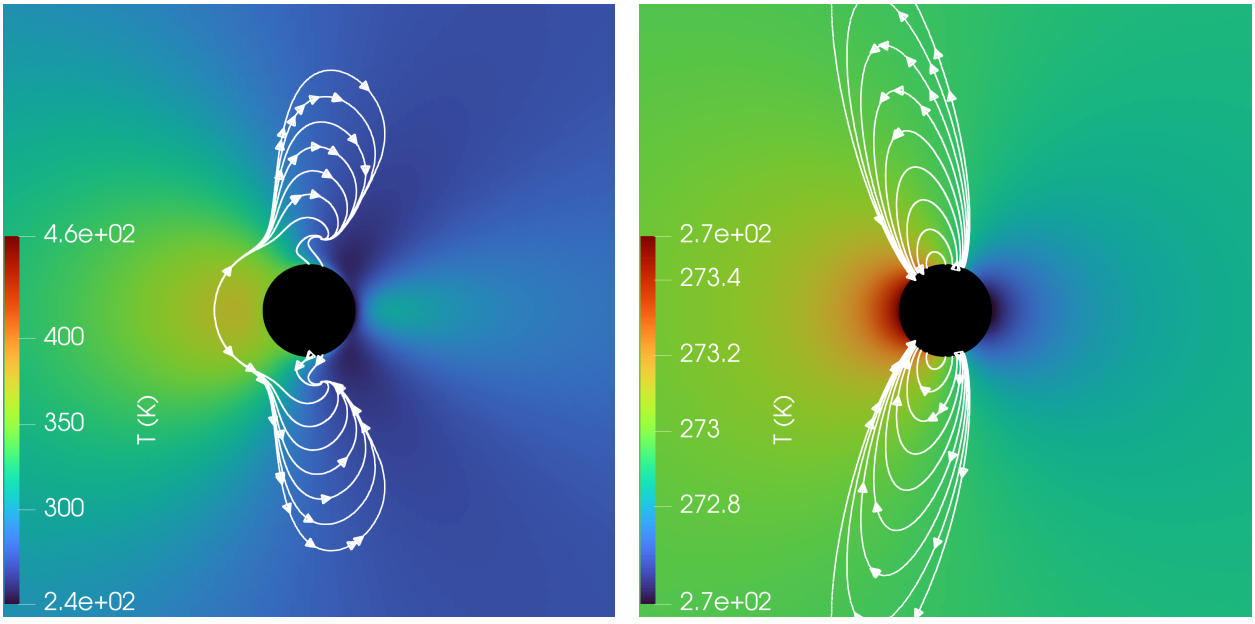
$$\mathbf{W_1} = \begin{bmatrix} \rho \\ P \\ u_x \\ m \\ R \\ T_w \\ \omega \end{bmatrix} = \begin{bmatrix} 1.7838\,\mathrm{kg/m^3} \\ 101325\,\mathrm{Pa} \\ 320\,\mathrm{m/s} \\ 6.63 \times 10^{-26}\,\mathrm{kg} \\ 1\,\mathrm{m} \\ 293.15\,\mathrm{k} \\ 0\,\mathrm{rad/s} \end{bmatrix} \quad \text{and} \quad \mathbf{W_2} = \begin{bmatrix} \rho \\ P \\ u_x \\ m \\ R \\ T_w \\ \omega \end{bmatrix} = \begin{bmatrix} 1.7838\,\mathrm{kg/m^3} \\ 101325\,\mathrm{Pa} \\ 3.20\,\mathrm{m/s} \\ 6.63 \times 10^{-26}\,\mathrm{kg} \\ 1\,\mathrm{m} \\ 273.15\,\mathrm{k} \\ 0\,\mathrm{rad/s} \end{bmatrix}. \tag{3.105}$$

Figure 3.16 presents the direction of heat-flux as an overlay to the temperature field. The heat-flux direction in Figure 3.16a clearly follows the temperature gradient, moving from hot to cold, which agrees with the natural intuition of most people. However, the heat-flux direction in Figure 3.16b flows against the temperature gradient, moving from cold to hot, which is counter-intuitive



(a) cylinder with no rotation (b) cylinder with rotation

Figure 3.15: Heat-flux comparison: cylinder with no rotation vs. cylinder with rotation



(a) With gradient heat-flux (b) Counter gradient heat-flux

Figure 3.16: Direction of heat-flux overlaid on top of the temperature gradient

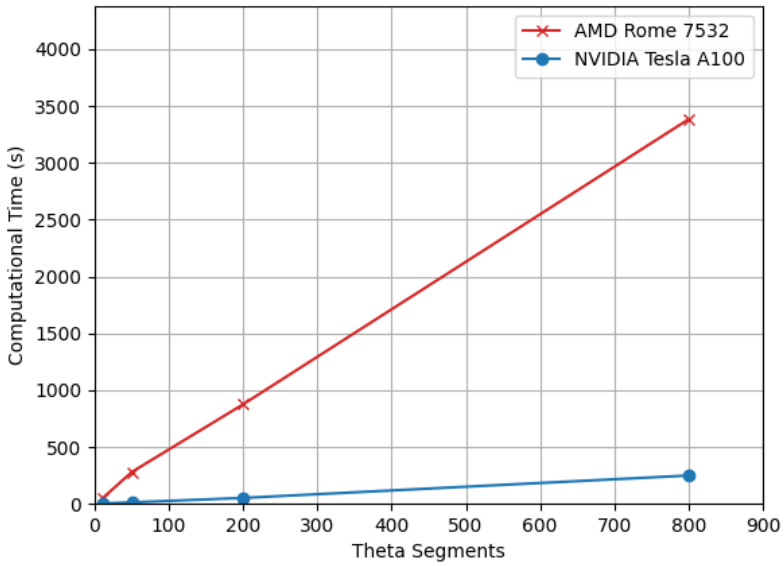


Figure 3.17: Computational runtime benchmarking for CPUs and GPUs using the FMFC solver

to the behaviour that most people would expect. This is an interesting observation because in the free-molecular regime, gas flows do not come to local thermodynamic equilibrium. As a result, behaviours such as a counter-gradient heat-flux are something that can occur. This demonstrates that violations of equilibrium models, such as Fourier's law, are expected in these extreme flows [25].

3.6.3 Performance Benchmarking

At any point near a possibly rotating cylinder, this highly efficient FMFC solver is capable of computing any observable property of the free-molecular gas flow. The purpose of developing this technique is to drastically improve upon the computational time required to produce accurate and high-resolution results. Additionally, to further improve upon these time requirements, the solver has been developed with the SYCL framework to enable the targeting of hardware accelerators such as GPUs. This further improves upon the computational time efficiency by exploiting the highly parallel nature of GPUs for this embarrassingly parallel problem. Table 3.3 presents four angular resolution studies, completed with varying spatial resolutions. Performance benchmarks in the table are provided for the AMD Rome 7532, a server-grade CPU, and the Nvidia Tesla A100, a server grade GPU. These studies were performed for the rotating cylinder parameters from Equation (3.104). The performance benchmarks presented in the table indicate that both the CPU and GPU are capable of quickly producing relatively accurate results with low θ segments. However, by increasing the number of θ segments to achieve greater accuracy, the CPU begins to require significantly longer computational runtimes relative to the GPU. Performance comparisons for a 500×500 grid are presented in Figure 3.17. The GPU implementation demonstrates speedups with a factor of 10-15 times the speed of the CPU version of the code,

Table 3.3: Comparison of computational runtimes using the FMFC solver for different CPUs and GPUs with a varying number of segments in the angular direction

(a) 10 segments in the θ direction

Resolution	**AMD Rome 7532**	**NVIDIA Tesla A100**
100x100	2s	1s
200x200	11s	1s
500x500	46s	3s

(b) 50 segments in the θ direction

Resolution	**AMD Rome 7532**	**NVIDIA Tesla A100**
100x100	12s	1s
200x200	36s	3s
500x500	280s	13s

(c) 200 segments in the θ direction

Resolution	**AMD Rome 7532**	**NVIDIA Tesla A100**
100x100	43s	5s
200x200	166s	10s
500x500	876s	52s

(d) 800 segments in the θ direction

Resolution	**AMD Rome 7532**	**NVIDIA Tesla A100**
100x100	180s	19s
200x200	626s	47s
500x500	3384s	249s

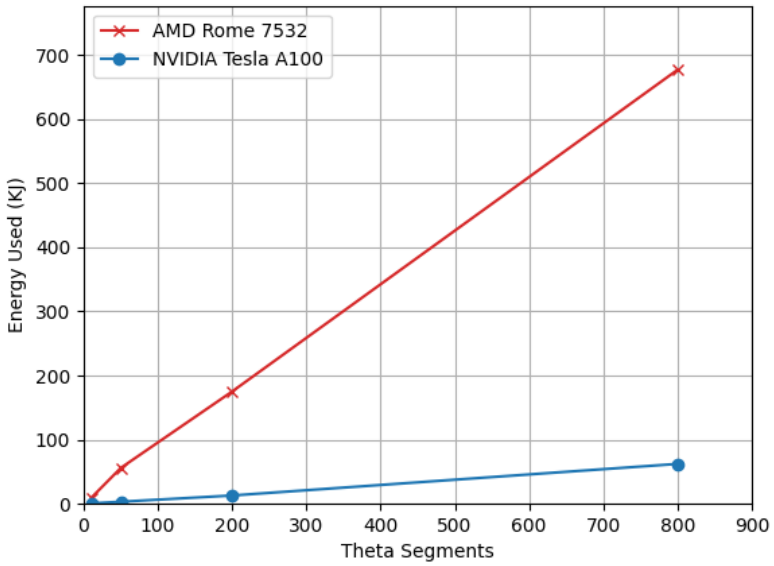


Figure 3.18: Energy consumption benchmarking for CPUs and GPUs using the FMFC solver

showing that the GPU version can be extremely advantageous for reduced computational runtimes. A somewhat crude estimate for the power consumption required from the CPU and GPU can be assessed by using their rated thermal design power (TDP). The AMD Rome 7532 has a rated TDP of 200 W and the Nvidia Tesla A100 has a rated TDP of 250 W. Energy consumption, E, is computed as

$$E = \text{TDP} \times \text{runtime}. \tag{3.106}$$

The comparison of energy consumption is highlighted in Figure 3.18. Reducing energy consumption and energy expenditures is of major concern in the field of computational sciences. Figure 3.18 demonstrates that, in addition to reduced computational runtimes, the GPU is also able to achieve reduced energy consumption for the same scientific computation as the CPU. Energy consumption for large calculations can be costly. These initial results are promising because they show both a reduced computational runtime and reduced energy expenditures by a factor of over 10.

The FMFC solver is able to quickly produce relatively accurate results on both CPUs and GPUs. Producing accurate results with this solver is possible for anyone with either consumer-grade or server-grade hardware. This can be highly advantageous for scientists and engineers that want to produce reliable benchmarks or interesting academic problems to study.

Chapter 4

The Discontinuous Galerkin Hancock Method

The Discontinuous Galerkin Hancock (DGH) method is a numerical scheme that was originally developed by Suzuki and van Leer [12]. This high-order numerical technique was designed to solve systems of hyperbolic relaxation PDEs with stiff relaxation source terms. This class of equations is commonly used to model various important scientific problems, such as multi-phase flows [26, 22], radiation-transport prediction [27], ionized plasma flows [28], and non-equilibrium gas flows [13]. Often, the relaxation time in the source term of these PDEs are very stiff. That is to say, it can describe processes that occur on time scales that are very fast compared to other aspects of the situation. Local processes, such as drag and ionization, can severely restrict timestep sizes to maintain stable and accurate computations if purely explicit time marching is used. The DGH scheme addresses these issues by treating the time-step for the source term implicitly, without sacrificing accuracy. This efficient technique extends upon Huynh's one-step, fully discrete, "upwind moment scheme" for application to hyperbolic-relaxation equations [14]. The DGH scheme achieves third-order accuracy for both space and time, which is advantageous because of the wide range of scientific problems that it can solve both accurately and efficiently. This numerical method has been implemented for large-scale CPU-based systems in one, two, and three dimensions [11]. The present work extends upon this existing contribution in order to produce a DGH implementation that can target GPU-accelerated platforms using the SYCL open-standard. This chapter presents an overview for the numerical structure of DGH, a high-level demonstration discussing how problems are distributed across multiple GPUs on large heterogeneous systems, parallel scaling studies for the GPU implementation, and performance benchmarking between the CPU and GPU versions of the code.

4.1 First-Order DG Method

In addition to the high-order DGH scheme that is highlighted in this chapter, the present work has also implemented a first-order DG scheme. It is advantageous to have a first-order scheme because they are relatively robust and can produce smooth solutions in areas that can be difficult

for high-order schemes to resolve, such as regions of shocks. The DG scheme is used to solve hyperbolic PDEs that take the form

$$\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial \boldsymbol{F}_i}{\partial x_i} = \boldsymbol{S}. \tag{4.1}$$

In this scheme, the spatial domain is discretized into multiple cells and the global solution is approximated in each cell using constant discontinuous functions. The approximate solution in each cell, U_h, is the product of the solution average value in each cell, α_0, and the basis function, $\phi_0 = 1$, which results in

$$\boldsymbol{U}_h = \alpha_0 \phi_0 = \alpha_0. \tag{4.2}$$

The weak formulation is obtained by projecting the basis function onto the PDE. This projection yields

$$\iint \phi_0 \frac{\partial}{\partial t} \boldsymbol{U}_h \, \mathrm{d}x_i \, \mathrm{d}t + \iint \phi_0 \frac{\partial}{\partial x_i} \boldsymbol{F}_i \, \mathrm{d}x_i \, \mathrm{d}t = \iint \phi_0 \boldsymbol{S} \, \mathrm{d}x_i \, \mathrm{d}t \tag{4.3}$$

and simplifies to

$$\iint \frac{\partial}{\partial t} \alpha_0 \, \mathrm{d}x_i \, \mathrm{d}t + \iint \frac{\partial}{\partial x_i} \boldsymbol{F}_i \, \mathrm{d}x_i \, \mathrm{d}t = \iint \boldsymbol{S} \, \mathrm{d}x_i \, \mathrm{d}t. \tag{4.4}$$

By treating the integrals of the solution average term with an integration by parts and applying the divergence theorem to the flux term, an update formula is obtained for the solution average value in each cell that takes the form

$$\alpha_0^{n+1} = \alpha_0 + \frac{1}{V} \left(- \int \oint F_i \cdot \hat{n}_i \, \mathrm{d}\ell \, \mathrm{d}t + \iint \boldsymbol{S} \, \mathrm{d}x_i \, \mathrm{d}t \right), \tag{4.5}$$

where V is the cell volume. The flux terms are evaluated at the current timestep, t^n. The surface integral for the flux term is computed with quadrature rules on each cell face. The time integral and surface integral for the flux term become

$$\int \oint F_i \cdot \hat{n}_i \, \mathrm{d}\ell \, \mathrm{d}t = \Delta t \sum_{\xi} w_\xi \boldsymbol{F}_\xi \,, \tag{4.6}$$

where ξ represents the quadrature points on each cell face and w_ξ is each contributing weight. These flux terms are computed as the result to a Riemann problem with neighbouring cells. The source term is evaluated with implicit-Euler time marching at the cell centroids for the next time step, t^{n+1}. The time integral and spatial integral for the source term become

$$\iint \boldsymbol{S} \, \mathrm{d}x_i \, \mathrm{d}t = \Delta t V \boldsymbol{S}(\alpha_0^{n+1}) \,. \tag{4.7}$$

The discrete update formula for the solution average value in each cell of this first-order scheme takes the form

$$\alpha_0^{n+1} = \alpha_0 + \Delta t \left(-\frac{1}{V} \sum_{\xi} w_\xi \boldsymbol{F}_\xi + \boldsymbol{S}(\alpha_0^{n+1}) \right) . \tag{4.8}$$

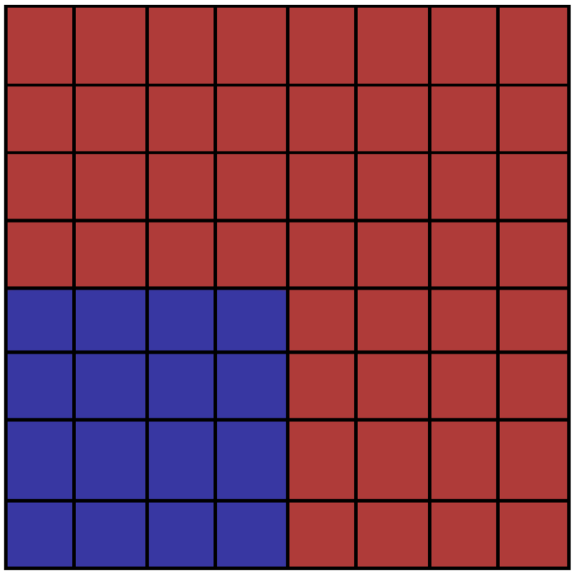


Figure 4.1: Discretization of cells for an initial value problem

This section of the chapter only briefly introduces the first-order DG scheme. The following sections of the chapter provide an in-depth overview of the high-order DGH scheme, highlighting the weak formulation of DGH, update formulas for the weak solution, and the discrete quadrature rules that are used in this numerical method. Explanations for the development of DGH provide the reader with enough context to thoroughly understand the required steps to implement the scheme.

4.2 The Weak Formulation

As previously stated, the DGH scheme is used to numerically approximate the solution for hyperbolic PDEs with stiff local source terms. These equations take the form

$$\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial \boldsymbol{F}_i}{\partial x_i} = \boldsymbol{S}. \tag{4.9}$$

Application of the DGH method requires discretizations in both space and time. Figure 4.1 presents a two-dimensional initial-value problem with a spatial discretization of cells. To contextualize the problem presented in the figure, it is important to note that the cells highlighted in blue depict a low-density and low-pressure gas, while the cells highlighted in red represent a high-density and high-pressure gas. Each individual cell in the discretized spatial domain is assigned its own unique set of basis functions ϕ_υ and test functions ϕ_η. In this scheme, the basis functions and test functions are equal to one another, so the linear set of functions ϕ can be expressed as

$$\phi = \phi_\upsilon = \phi_\eta \tag{4.10}$$

such that

$$\phi = [1, x - x_c, y - y_c, z - z_c]\,. \tag{4.11}$$

The basis functions represent a set of 4 polynomial functions that are denoted with Greek characters. The variables x_c, y_c, and z_c are the respective x, y, and z direction centroids in each cell. These centroids are given by

$$x_c = \frac{\iiint_{\Omega} x \, \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z}{\iiint_{\Omega} \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z}, \tag{4.12}$$

$$y_c = \frac{\iiint_{\Omega} y \, \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z}{\iiint_{\Omega} \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z}, \tag{4.13}$$

and

$$z_c = \frac{\iiint_{\Omega} z \, \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z}{\iiint_{\Omega} \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z}, \tag{4.14}$$

where Ω represents the spatial boundaries of integration for each individual cell. Cell averages, α_υ, are also uniquely attributed to each cell, such that

$$\alpha_\upsilon = \left(\overline{u}, \overline{\frac{\mathrm{d}u}{\mathrm{d}x}}, \overline{\frac{\mathrm{d}u}{\mathrm{d}y}}, \overline{\frac{\mathrm{d}u}{\mathrm{d}z}} \right). \tag{4.15}$$

The first component, $\overline{u}$, is the solution average state in each cell. The next three components represent the cell-specific slope average values in the x, y, and z direction. The approximate solution in each cell is defined as the weighted sum of the cell averages, α_υ, and the basis functions, ϕ_υ. This approximation is referred to as the trial solution, U_h, and given by

$$\boldsymbol{U}_h = \sum_{\eta=0}^{3} \alpha_\eta \phi_\eta, \tag{4.16}$$

which expands out to

$$\boldsymbol{U}_h = \overline{u} + \overline{\frac{\mathrm{d}u}{\mathrm{d}x}}(x - x_c) + \overline{\frac{\mathrm{d}u}{\mathrm{d}y}}(y - y_c) + \overline{\frac{\mathrm{d}u}{\mathrm{d}z}}(z - z_c). \tag{4.17}$$

In each cell, the basis functions only depend on space and the cell averages only depend on time. The scheme follows a coupled space-time approach, which means it also requires discretizations in time. Figure 4.2 demonstrates a single cell in two-dimensional space that is being stepped forward in time. In Figure 4.2, t^n is the current timestep and t^{n+1} is the next timestep. These boundaries define the discretized time domain, T, where,

$$T = [t^n, t^{n+1}]. \tag{4.18}$$

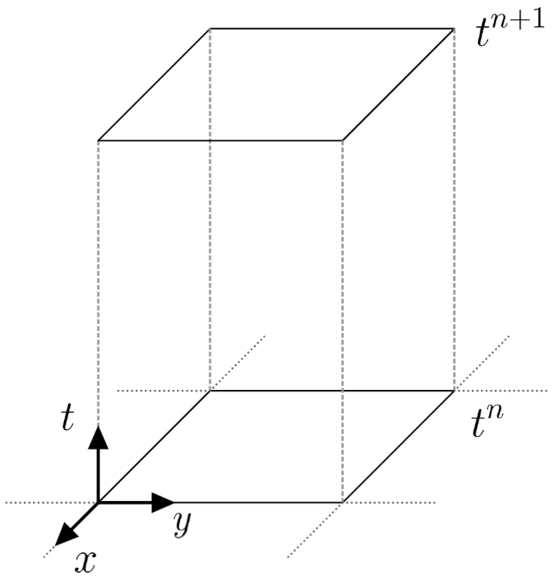


Figure 4.2: Representation of a two dimensional space-time cell

Analysis in the DGH scheme, as depicted in Figure 4.2, largely treats each cell individually for integrations in space and time. This approach to the analysis is because basis functions uniquely exist in each cell and are zero everywhere else. The weak formulation is obtained by projecting the test function, ϕ_η, onto the PDE from Equation (4.9). By taking the product of the PDE with the test function and integrating it across the space and time domains, the weak formulation takes the form

$$\iint\limits_{T\times\Omega} \phi_\eta \frac{\partial}{\partial t} \boldsymbol{U}_h \,\mathrm{d}x_i \,\mathrm{d}t + \iint\limits_{T\times\Omega} \phi_\eta \frac{\partial}{\partial x_i} \boldsymbol{F}_i \,\mathrm{d}x_i \,\mathrm{d}t = \iint\limits_{T\times\Omega} \phi_\eta \boldsymbol{S} \,\mathrm{d}x_i \,\mathrm{d}t. \tag{4.19}$$

Projecting the test function onto the PDE ensures that the error between the approximate solution and the exact solution is orthogonal to the trial solution in the selected function space. This forces the residual to be orthogonal to the selected function space of the approximate solution. The weak formulation is used to develop update formulas that describe how the solution evolves. First, the trial solution from Equation (4.16) is substituted into the weak formulation to get

$$\iint\limits_{T\times\Omega} \phi_\eta \frac{\partial}{\partial t} \left(\sum_{\upsilon=0}^{3} \alpha_\upsilon \phi_\upsilon \right) \,\mathrm{d}x_i \,\mathrm{d}t + \iint\limits_{T\times\Omega} \phi_\eta \frac{\partial}{\partial x_i} \left(\boldsymbol{F}_i \right) \,\mathrm{d}x_i \,\mathrm{d}t = \iint\limits_{T\times\Omega} \phi_\eta \boldsymbol{S} \,\mathrm{d}x_i \,\mathrm{d}t. \tag{4.20}$$

4.3 Update Formulas

The update formulas are required to describe the evolution of the solution and slope averages in each cell for every timestep. These update formulas are obtained by applying an integration by parts to the time-evolution term and the flux term.

4.3.1 Integration by Parts for the Time Evolution Term

The time integral of the time-evolution term in the weak formulation is handled with an integration by parts, where

$$\iint\limits_{T\times\Omega} \phi_\eta \frac{\partial}{\partial t}\left(\sum_{\upsilon=0}^{3} \alpha_\upsilon \phi_\upsilon\right) \mathrm{d}x_i \,\mathrm{d}t = \left[\int\limits_{\Omega} \phi_\eta \left(\sum_{\upsilon=0}^{3} \alpha_\upsilon \phi_\upsilon\right) \mathrm{d}x_i\right]_{t^n}^{t^{n+1}} - \iint\limits_{T\times\Omega} \sum_{\upsilon=0}^{3} \alpha_\upsilon \phi_\upsilon \frac{\partial \phi_\eta}{\partial t} \,\mathrm{d}x_i \,\mathrm{d}t. \tag{4.21}$$

Since the test functions ϕ_η are constant with respect to time,

$$\frac{\partial \phi_\eta}{\partial t} = 0 \tag{4.22}$$

and the first term becomes

$$\iint\limits_{T\times\Omega} \phi_\eta \frac{\partial}{\partial t}\left(\sum_{\upsilon=0}^{3} \alpha_\upsilon \phi_\upsilon\right) \mathrm{d}x_i \,\mathrm{d}t = \left[\int\limits_{\Omega} \phi_\eta \left(\sum_{\upsilon=0}^{3} \alpha_\upsilon \phi_\upsilon\right) \mathrm{d}x_i\right]^{t^{n+1}} - \left[\int\limits_{\Omega} \phi_\eta \left(\sum_{\upsilon=0}^{3} \alpha_\upsilon \phi_\upsilon\right) \mathrm{d}x_i\right]^{t^n}, \tag{4.23}$$

which updates the weak form of the PDE to be

$$\begin{aligned} &\left[\int\limits_{\Omega} \phi_\eta \left(\sum_{\upsilon=0}^{3} \alpha_\upsilon \phi_\upsilon\right) \mathrm{d}x_i\right]^{t^{n+1}} - \left[\int\limits_{\Omega} \phi_\eta \left(\sum_{\upsilon=0}^{3} \alpha_\upsilon \phi_\upsilon\right) \mathrm{d}x_i\right]^{t^n} \\ &\qquad = -\iint\limits_{T\times\Omega} \phi_\eta \frac{\partial}{\partial x_i} \boldsymbol{F}_i \,\mathrm{d}x_i \,\mathrm{d}t + \iint\limits_{T\times\Omega} \phi_\eta \boldsymbol{S} \,\mathrm{d}x_i \,\mathrm{d}t. \end{aligned} \tag{4.24}$$

4.3.2 Integration by Parts for the Flux Term

The product rule states that

$$\frac{\partial}{\partial x_i}\left(\boldsymbol{F}_i \phi_\eta\right) = \phi_\eta \frac{\partial}{\partial x_i} \boldsymbol{F}_i + \boldsymbol{F}_i \frac{\partial}{\partial x_i} \phi_\eta \tag{4.25}$$

and applying the product rule to the flux term in the weak form of the PDE results in

$$\int\limits_{\Omega} \phi_\eta \frac{\partial}{\partial x_i} \boldsymbol{F}_i \,\mathrm{d}x_i = \int\limits_{\Omega} \frac{\partial}{\partial x_i}\left(\boldsymbol{F}_i \phi_\eta\right) \,\mathrm{d}x_i - \int\limits_{\Omega} \boldsymbol{F}_i \frac{\partial}{\partial x_i} \phi_\eta \,\mathrm{d}x_i. \tag{4.26}$$

From the divergence theorem,

$$\int\limits_{\Omega} \frac{\partial}{\partial x_i}\left(\boldsymbol{F}_i \phi_\eta\right) \,\mathrm{d}x_i = \oint\limits_{\partial\Omega} \boldsymbol{F}_i \phi_\eta \cdot \hat{\boldsymbol{n}} \,\mathrm{d}\ell, \tag{4.27}$$

which highlights that the volume integral for the divergence of a vector field is equal to the surface integral of a vector-field through the boundaries of the volume $\partial\Omega$. By applying Equation (4.27)

and Equation (4.26) to the flux term in the weak form of the PDE, Equation (4.24) becomes

$$\begin{aligned}\left[\int_{\Omega} \phi_\eta \left(\sum_{\upsilon=0}^{3} \alpha_\upsilon \phi_\upsilon\right) \mathrm{d}x_i\right]^{t^{n+1}} &= \left[\int_{\Omega} \left(\sum_{\upsilon=0}^{3} \alpha_\upsilon \phi_\upsilon\right) \phi_\eta \,\mathrm{d}x_i\right]^{t^n} \\ - \oiint_{T\times\partial\Omega} \boldsymbol{F}_i \phi_\eta \cdot \hat{\boldsymbol{n}} \mathrm{d}\ell \,\mathrm{d}t + \iint_{T\times\Omega} \boldsymbol{F}_i \frac{\partial}{\partial x_i} \phi_\eta \,\mathrm{d}x_i \,\mathrm{d}t &+ \iint_{T\times\Omega} \boldsymbol{S}\phi_\eta \,\mathrm{d}x_i \,\mathrm{d}t.\end{aligned} \tag{4.28}$$

Expanding out the basis functions and conducting some integration steps produces the solution average update in each cell,

$$\alpha_0^{n+1} = \alpha_0^n - \frac{1}{V} \iint_{T\times\partial\Omega} [F_i \cdot \hat{n}_i] \,\mathrm{d}\ell \,\mathrm{d}t + \iint_{T\times\Omega} [S] \,\mathrm{d}x_i \,\mathrm{d}t, \tag{4.29}$$

where the volume V in each cell is

$$V = \iiint_{\Omega} \mathrm{d}x \,\mathrm{d}y \,\mathrm{d}z. \tag{4.30}$$

The slope average updates in each cell come to be

$$\begin{bmatrix}\alpha_1^{n+1}\\ \alpha_2^{n+1}\\ \alpha_3^{n+1}\end{bmatrix} = \begin{bmatrix}\alpha_1^{n}\\ \alpha_2^{n}\\ \alpha_3^{n}\end{bmatrix} + [K]\left(\iint_{T\times\Omega} \begin{bmatrix}F_1\\ F_2\\ F_3\end{bmatrix} \mathrm{d}x_i \,\mathrm{d}t - \iint_{T\times\partial\Omega} \begin{bmatrix}\phi_1\\ \phi_2\\ \phi_3\end{bmatrix} F_i \cdot \hat{n}_i \,\mathrm{d}l \,\mathrm{d}t + \iint_{T\times\Omega} \begin{bmatrix}\phi_1\\ \phi_2\\ \phi_3\end{bmatrix} S \,\mathrm{d}x_i \,\mathrm{d}t\right), \tag{4.31}$$

where K is the inverse, cell-specific, moment of inertia matrix, such that

$$\boldsymbol{K} = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}^{-1} \tag{4.32}$$

and the elements of the matrix K are the moments of inertia at each cell face, which are

$$I_{xx} = \iiint_{\Omega} (x - x_c)^2 \,\mathrm{d}x_i, \tag{4.33}$$

$$I_{xy} = I_{yx} = \iiint_{\Omega} (x - x_c)(y - y_c) \,\mathrm{d}x_i, \tag{4.34}$$

$$I_{xz} = I_{zx} = \iiint_{\Omega} (x - x_c)(z - z_c) \,\mathrm{d}x_i, \tag{4.35}$$

$$I_{yy} = \iiint_{\Omega} (y - y_c)^2 \,\mathrm{d}x_i, \tag{4.36}$$

$$I_{yz} = I_{zy} = \iiint_{\Omega} (y - y_c)(z - z_c)\,\mathrm{d}x_i, \tag{4.37}$$

and

$$I_{zz} = \iiint_{\Omega} (z - z_c)^2\,\mathrm{d}x_i. \tag{4.38}$$

The remaining integrals in the update formulas need to be evaluated using numerical approximations to produce discrete update formulas that can be computed at each timestep.

4.4 Discrete Update Formulas Using Quadrature Rules

The solution average update formula in Equation (4.29) and the slope average update formula in Equation (4.31) both require numerical approximations for the space and time integrals in their solutions. These integrals are computed with a set of quadrature rules and numerical techniques for the source and flux terms.

4.4.1 Time Integral and Volume Integral for the Source Term

The source term in both update formulas can be very stiff relative to the flux term. This presents an issue because local processes, such as chemical reactions in reactive flows, develop quickly when compared to other aspects of the system that are dictated by the flux terms. To avoid severely restricted timestep sizes, the source term is stepped in time implicitly. The time integral and the volume integral for the source term need to be computed discretely and the method used to treat the time integral is Radau IIA. This is the selected approach because Radau IIA is an implicit Runge-Kutta timestepping method that maintains third-order accuracy and unconditional stability. This method can be used for some function f at timestep t^n that needs to be stepped forward to the timestep t^{n+1}. Radau IIA does this implicitly with a two step approach, given by the system

$$\begin{aligned} f^{n+\frac{1}{3}} &= f^n + \Delta t \left(\frac{5}{12} \left(\frac{\mathrm{d}f}{\mathrm{d}t} \right)^{n+\frac{1}{3}} - \frac{1}{12} \left(\frac{\mathrm{d}f}{\mathrm{d}t} \right)^{n+1} \right) \\ f^{n+1} &= f^n + \Delta t \left(\frac{3}{4} \left(\frac{\mathrm{d}f}{\mathrm{d}t} \right)^{n+\frac{1}{3}} + \frac{1}{4} \left(\frac{\mathrm{d}f}{\mathrm{d}t} \right)^{n+1} \right), \end{aligned} \tag{4.39}$$

where Δt is the size of the timestep,

$$\Delta t = t^{n+1} - t^n. \tag{4.40}$$

The space integral for the source term is computed using Gaussian quadrature points at the centre of each cell. Figure 4.3 provides an example of quadrature points for the source term in a two-dimensional element at the $t^{n+\frac{1}{3}}$ and t^{n+1} positions in time. In the update formulas, the source term is coupled to the solution and slope average values at the updated timesteps. A linear approximation is used to resolve this issue and simplify the source term. The linear treatment of

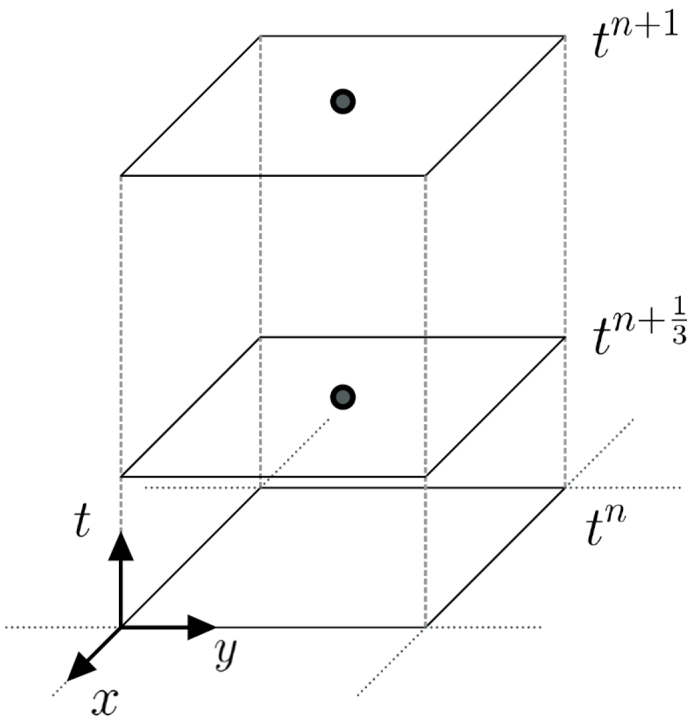


Figure 4.3: Quadrature points used to evaluate source term integrals on a two-dimensional space-time plot

the source term S_L takes the form

$$\boldsymbol{S}_L \approx \boldsymbol{S}(\alpha_0^n) + \left(\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{U}}\right)^n \begin{bmatrix} \alpha_1^n \phi_1^n \\ \alpha_2^n \phi_2^n \\ \alpha_3^n \phi_3^n \end{bmatrix}. \tag{4.41}$$

The space integral of the source term in the solution average update formula then equates to

$$\iint\limits_{T\times\Omega} \boldsymbol{S}_L \phi_0 \mathrm{d}x_i \mathrm{d}t \approx \int\limits_T V \boldsymbol{S}(\alpha_0) \mathrm{d}t \tag{4.42}$$

and the space integral of the source term in the slope average updates then come to be

$$\iint\limits_{T\times\Omega} \boldsymbol{S}_L \phi_1 \mathrm{d}x_i \mathrm{d}t \approx \int\limits_T \frac{\partial \boldsymbol{S}}{\partial \boldsymbol{U}} (I_{xx}\alpha_1 + I_{xy}\alpha_2 + I_{xz}\alpha_3) \mathrm{d}t, \tag{4.43}$$

$$\iint\limits_{T\times\Omega} \boldsymbol{S}_L \phi_2 \mathrm{d}x_i \mathrm{d}t \approx \int\limits_T \frac{\partial \boldsymbol{S}}{\partial \boldsymbol{U}} (I_{yx}\alpha_1 + I_{yy}\alpha_2 + I_{yz}\alpha_3) \mathrm{d}t, \tag{4.44}$$

and

$$\iint\limits_{T\times\Omega} \boldsymbol{S} \phi_3 \mathrm{d}x_i \mathrm{d}t \approx \int\limits_T \frac{\partial \boldsymbol{S}}{\partial \boldsymbol{U}} (I_{zx}\alpha_1 + I_{zy}\alpha_2 + I_{zz}\alpha_3) \mathrm{d}t. \tag{4.45}$$

The time integral in each of the update formulas is evaluated with the Radau IIA implicit timestepping method. Applying this method to the solution average update and the slope average updates

result in

$$\begin{bmatrix} \alpha_0^{n+\frac{1}{3}} \\ \alpha_0^{n+1} \end{bmatrix} = \begin{bmatrix} \alpha_0^{n} \\ \alpha_0^{n} \end{bmatrix} - \frac{1}{V} \begin{bmatrix} \iint\limits_{T_{1/3}\times\partial\Omega} \boldsymbol{F}_i \cdot \hat{n}_i \, \mathrm{d}\ell \, \mathrm{d}t \\ \iint\limits_{T\times\partial\Omega} \boldsymbol{F}_i \cdot \hat{n}_i \, \mathrm{d}\ell \, \mathrm{d}t \end{bmatrix} + \Delta t \begin{bmatrix} \frac{5}{12} & -\frac{1}{12} \\ \frac{3}{4} & +\frac{1}{4} \end{bmatrix} \begin{bmatrix} \boldsymbol{S}(\alpha_0^{n+\frac{1}{3}}) \\ \boldsymbol{S}(\alpha_0^{n+1}) \end{bmatrix} \tag{4.46}$$

and

$$\begin{bmatrix} \alpha_1^{n+\frac{1}{3}} \\ \alpha_2^{n+\frac{1}{3}} \\ \alpha_3^{n+\frac{1}{3}} \\ \alpha_1^{n+1} \\ \alpha_2^{n+1} \\ \alpha_3^{n+1} \end{bmatrix} = \begin{bmatrix} \alpha_1^{n} \\ \alpha_2^{n} \\ \alpha_3^{n} \\ \alpha_1^{n} \\ \alpha_2^{n} \\ \alpha_3^{n} \end{bmatrix} + [K] \begin{bmatrix} \iint\limits_{T_{1/3}\times\Omega} F_i \, \mathrm{d}x_i \, \mathrm{d}t - \iint\limits_{T_{1/3}\times\partial\Omega} \phi_\eta \boldsymbol{F}_i \cdot \hat{n}_i \, \mathrm{d}\ell \, \mathrm{d}t \\ \iint\limits_{T_{1/3}\times\Omega} F_i \, \mathrm{d}x_i \, \mathrm{d}t - \iint\limits_{T_{1/3}\times\partial\Omega} \phi_\eta \boldsymbol{F}_i \cdot \hat{n}_i \, \mathrm{d}\ell \, \mathrm{d}t \\ \iint\limits_{T_{1/3}\times\Omega} F_i \, \mathrm{d}x_i \, \mathrm{d}t - \iint\limits_{T_{1/3}\times\partial\Omega} \phi_\eta \boldsymbol{F}_i \cdot \hat{n}_i \, \mathrm{d}\ell \, \mathrm{d}t \\ \iint\limits_{T\times\Omega} F_i \, \mathrm{d}x_i \, \mathrm{d}t - \iint\limits_{T\times\partial\Omega} \phi_\eta \boldsymbol{F}_i \cdot \hat{n}_i \, \mathrm{d}\ell \, \mathrm{d}t \\ \iint\limits_{T\times\Omega} F_i \, \mathrm{d}x_i \, \mathrm{d}t - \iint\limits_{T\times\partial\Omega} \phi_\eta \boldsymbol{F}_i \cdot \hat{n}_i \, \mathrm{d}\ell \, \mathrm{d}t \\ \iint\limits_{T\times\Omega} F_i \, \mathrm{d}x_i \, \mathrm{d}t - \iint\limits_{T\times\partial\Omega} \phi_\eta \boldsymbol{F}_i \cdot \hat{n}_i \, \mathrm{d}\ell \, \mathrm{d}t \end{bmatrix} \tag{4.47}$$

$$+\Delta t \begin{bmatrix} \frac{5}{12}I & -\frac{1}{12}I \\ \frac{3}{4}I & +\frac{1}{4}I \end{bmatrix} \begin{bmatrix} \left(\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{U}}\right)^{n+\frac{1}{3}} \alpha_1^{n+\frac{1}{3}} \\ \left(\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{U}}\right)^{n+\frac{1}{3}} \alpha_2^{n+\frac{1}{3}} \\ \left(\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{U}}\right)^{n+\frac{1}{3}} \alpha_3^{n+\frac{1}{3}} \\ \left(\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{U}}\right)^{n+1} \alpha_1^{n+1} \\ \left(\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{U}}\right)^{n+1} \alpha_2^{n+1} \\ \left(\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{U}}\right)^{n+1} \alpha_3^{n+1} \end{bmatrix},$$

respectively. Here, $T_{1/3}$ represents the timestep

$$T_{1/3} = [t^n, t^{n+\frac{1}{3}}]. \tag{4.48}$$

4.4.2 Time Integral and Surface Integral for the Interface Flux Term

The approximate solution, U_h, is unique to each individual cell in the DGH method, which means that the global solution is discontinuous at every cell interface. Computing the flux through each interface requires a Riemann problem to be solved because of the discontinuity present at the shared boundaries of adjacent cells. In order to achieve third-order accuracy when evaluating the surface integral of the flux term, Gaussian quadrature rules are used. The approximate solution is composed of polynomials of degree $k = 1$. According to Cockburn and Shu, the quadrature rules over the edge must satisfy a polynomial of degree $2k + 1$ [29]. This means that the number of quadrature points n must satisfy

$$2n - 1 \geq 2k + 1, \tag{4.49}$$

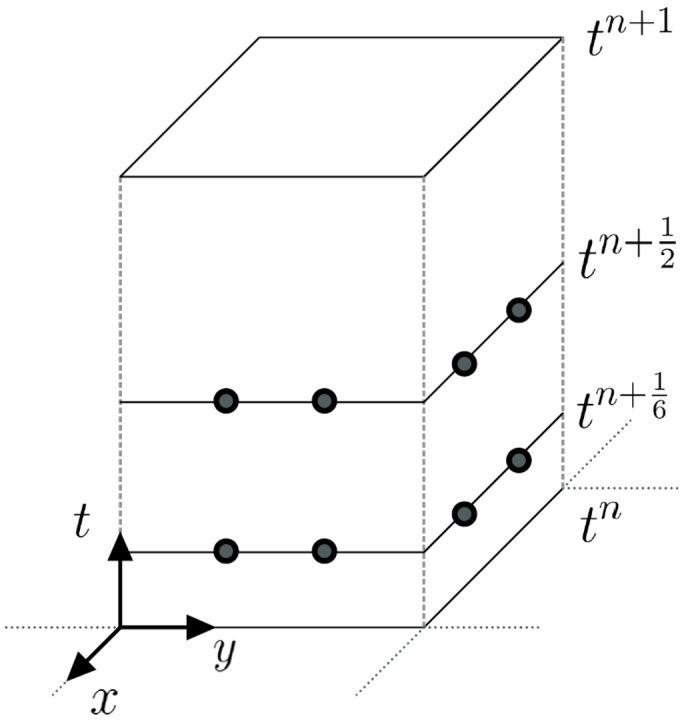


Figure 4.4: Quadrature points used to evaluate flux term integrals, highlighted on a two-dimensional space-time plot

which results in $n = 2$ quadrature points that are used to numerically approximate the surface integral for the interface flux term. This approximation from the solution update formula takes the form

$$\iint_{T\times\partial\Omega} F_i \cdot \hat{n}_i \,\mathrm{d}l \,\mathrm{d}t = \int_T \sum_\xi w_\xi \boldsymbol{F}_\xi \,\mathrm{d}t, \tag{4.50}$$

where ξ represents the selected quadrature points and w_ξ is the weighted contribution of each quadrature point. The slope update formula becomes

$$\iint_{T\times\partial\Omega} \phi_k F_i \cdot \hat{n}_i \,\mathrm{d}\ell \,\mathrm{d}t = \int_T \begin{bmatrix} \sum_\xi w_\xi (x_\xi - x_c) \\ \sum_\xi w_\xi (y_\xi - y_c) \\ \sum_\xi w_\xi (z_\xi - y_c) \end{bmatrix} \boldsymbol{F}_\xi \,\mathrm{d}t. \tag{4.51}$$

Here, the x_ξ, y_ξ, and z_ξ terms represent the x-, y-, and z-direction positions of the quadrature points. The time integral is approximated with the midpoint rule. Since Radau IIA for the source term requires solutions at the $t^{n+\frac{1}{3}}$ and t^{n+1} positions in time, midpoint requires the interface flux calculations at the $t^{n+\frac{1}{6}}$ and $t^{n+\frac{1}{2}}$ intermediate positions in time, respectively. Figure 4.4 provides an example of quadrature points, for the interface flux term, in a two-dimensional element at the $t^{n+\frac{1}{6}}$ and $t^{n+\frac{1}{2}}$ positions in time. With this treatment for the time and space integral of the interface

flux applied to both updates, the solution update formula becomes

$$\begin{bmatrix} \alpha_0^{n+\frac{1}{3}} \\ \alpha_0^{n+1} \end{bmatrix} = \begin{bmatrix} \alpha_0^{n} \\ \alpha_0^{n} \end{bmatrix} - \frac{\Delta t}{V} \begin{bmatrix} \frac{1}{3} \sum_\xi w_\xi \boldsymbol{F}_\xi^{n+\frac{1}{6}} \\ \sum_\xi w_\xi \boldsymbol{F}_\xi^{n+\frac{1}{2}} \end{bmatrix} + \Delta t \begin{bmatrix} \frac{5}{12} & -\frac{1}{12} \\ \frac{3}{4} & \frac{1}{4} \end{bmatrix} \begin{bmatrix} \boldsymbol{S}(\alpha_0^{n+\frac{1}{3}}) \\ \boldsymbol{S}(\alpha_0^{n+1}) \end{bmatrix} \tag{4.52}$$

and the slope update formula becomes

$$\begin{aligned} \begin{bmatrix} \alpha_1^{n+\frac{1}{3}} \\ \alpha_2^{n+\frac{1}{3}} \\ \alpha_3^{n+\frac{1}{3}} \\ \alpha_1^{n+1} \\ \alpha_2^{n+1} \\ \alpha_3^{n+1} \end{bmatrix} = \begin{bmatrix} \alpha_1^{n} \\ \alpha_2^{n} \\ \alpha_3^{n} \\ \alpha_1^{n} \\ \alpha_2^{n} \\ \alpha_3^{n} \end{bmatrix} + [K] \begin{bmatrix} \iint_{T_{1/3}\times\Omega} F_i \,\mathrm{d}x_i \,\mathrm{d}t \\ \iint_{T_{1/3}\times\Omega} F_i \,\mathrm{d}x_i \,\mathrm{d}t \\ \iint_{T_{1/3}\times\Omega} F_i \,\mathrm{d}x_i \,\mathrm{d}t \\ \iint_{T\times\Omega} F_i \,\mathrm{d}x_i \,\mathrm{d}t \\ \iint_{T\times\Omega} F_i \,\mathrm{d}x_i \,\mathrm{d}t \\ \iint_{T\times\Omega} F_i \,\mathrm{d}x_i \,\mathrm{d}t \end{bmatrix} - \Delta t \begin{bmatrix} K & 0 & 0 \\ 0 & K & 0 \\ 0 & 0 & K \end{bmatrix} \begin{bmatrix} \frac{1}{3} \sum_\xi w_\xi (x_\xi - x_c) \boldsymbol{F}_\xi^{n+\frac{1}{6}} \\ \frac{1}{3} \sum_\xi w_\xi (y_\xi - y_c) \boldsymbol{F}_\xi^{n+\frac{1}{6}} \\ \frac{1}{3} \sum_\xi w_\xi (z_\xi - z_c) \boldsymbol{F}_\xi^{n+\frac{1}{6}} \\ \sum_\xi w_\xi (x_\xi - x_c) \boldsymbol{F}_\xi^{n+\frac{1}{2}} \\ \sum_\xi w_\xi (y_\xi - y_c) \boldsymbol{F}_\xi^{n+\frac{1}{2}} \\ \sum_\xi w_\xi (z_\xi - z_c) \boldsymbol{F}_\xi^{n+\frac{1}{2}} \end{bmatrix} \\ + \Delta t \begin{bmatrix} \frac{5}{12} I & -\frac{1}{12} I \\ \frac{3}{4} I & \frac{1}{4} I \end{bmatrix} \begin{bmatrix} \left(\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{U}}\right)^{n+\frac{1}{3}} \alpha_1^{n+\frac{1}{3}} \\ \left(\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{U}}\right)^{n+\frac{1}{3}} \alpha_2^{n+\frac{1}{3}} \\ \left(\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{U}}\right)^{n+\frac{1}{3}} \alpha_3^{n+\frac{1}{3}} \\ \left(\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{U}}\right)^{n+1} \alpha_1^{n+1} \\ \left(\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{U}}\right)^{n+1} \alpha_2^{n+1} \\ \left(\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{U}}\right)^{n+1} \alpha_3^{n+1} \end{bmatrix}. \end{aligned} \tag{4.53}$$

The flux terms calculated at each of the boundary quadrature points is the computed result of a Riemann solver at cell interfaces. The quadrature points on each interface are located $\pm l/(2\sqrt{3})$ from the midpoint, where l is the length of the interface. In DGH, the inputs to the Riemann solver from each cell is determined with the Hancock predictor step.

4.4.3 Hancock Predictor Step

The Hancock predictor step is used to predict the approximate solution, U_h, at the face quadrature points for the fractional timesteps, $t^{n+\frac{1}{6}}$ and $t^{n+\frac{1}{2}}$. Values calculated at these intermediate timesteps are used as inputs to the Riemann solver. Time integrations in this step are computed over the bounds

$$T_H = [t^n, t^{n+H}] \tag{4.54}$$

where $H = \frac{1}{6}$ or $H = \frac{1}{2}$. Predictions for the solution average values $\alpha_0^{n+\frac{1}{6}}$ and $\alpha_0^{n+\frac{1}{2}}$ are evaluated using the solution average update in each cell as

$$\alpha_0^{n+H} = \alpha_0^n - \frac{1}{V} \iint\limits_{T_H \times \partial\Omega} [\hat{F}_i \cdot \hat{n}_i] \, \mathrm{d}\ell \, \mathrm{d}t + \iint\limits_{T_H \times \Omega} [S] \, \mathrm{d}x_i \, \mathrm{d}t. \tag{4.55}$$

The time integral for the flux term $\hat{F}_i$ is computed at the boundary quadrature points, only using local information and the internal solution of each cell. This process is performed independently of neighbouring cells. Additionally, time integration for the source term is treated with implicit Euler and spatial integration for the source term is treated with a one point Gaussian quadrature rule at the cell centroid. This results in

$$\alpha_0^{n+H} = \alpha_0^n + \frac{\Delta t}{V} \left(- \sum_{\xi} w_\xi \boldsymbol{F}_\xi^n + VS(U_h(x_i, t^{n+H}) \right). \tag{4.56}$$

After calculating the solution average values, $\alpha_0^{n+\frac{1}{6}}$ and $\alpha_0^{n+\frac{1}{2}}$, the solution at the quadrature points on each cell interface can be computed using the basis functions and weights, such that

$$U_h^{n+H} = \alpha_0^{n+H} + \Phi \begin{bmatrix} \alpha_1^n (x - x_c) \\ \alpha_2^n (y - y_c) \\ \alpha_3^n (z - z_c) \end{bmatrix} \tag{4.57}$$

where Φ is a vector of slope limiting values. The effective process of this predictor step can be visualized in Figure 4.5 for the Riemann input on a boundary quadrature point at the $t^{n+\frac{1}{6}}$ timestep. Substituting Equation (4.56) into Equation (4.58) produces the complete formulation of the Hancock predictor step and takes the form,

$$U_h^{n+H} = \alpha_0^n + \frac{\Delta t}{V} \left(- \sum_{\xi} w_\xi \boldsymbol{F}_\xi^n + VS(U_h(x_i, t^{n+H}) \right) + \Phi \begin{bmatrix} \alpha_1^n (x - x_c) \\ \alpha_2^n (y - y_c) \\ \alpha_3^n (z - z_c) \end{bmatrix} \tag{4.58}$$

4.4.4 Time Integral and Volume Integral for the Flux Term

The final integral to evaluate is the time and volume integrals for the flux term. The volume integral for the flux term is computed with a 2^D Gaussian quadrature rule. This means that in two-dimensional space, $2^2 = 4$ quadrature points are required per cell volume and in three-dimensional space, $2^3 = 8$ quadrature points are required. Figure 4.6 shows the location of the volume quadrature points for a cell in two spatial dimensions. In three dimensions, the spatial positions of these quadrature points, for a cubic element with lengths

$$(x, y, z) \in \mathbb{R}^3 \mid -1 \leq x \leq 1, \ -1 \leq y \leq 1, \ -1 \leq z \leq 1, \tag{4.59}$$

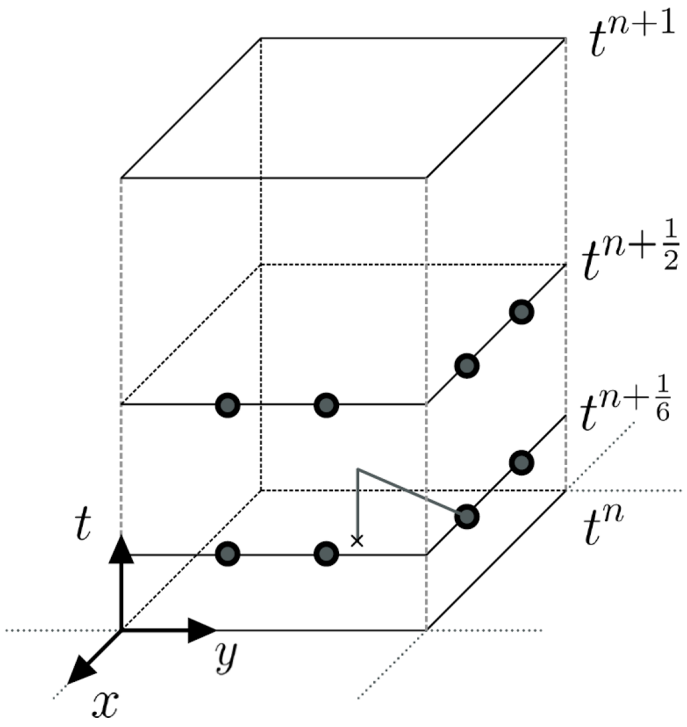


Figure 4.5: Visual representation of the Hancock predictor step

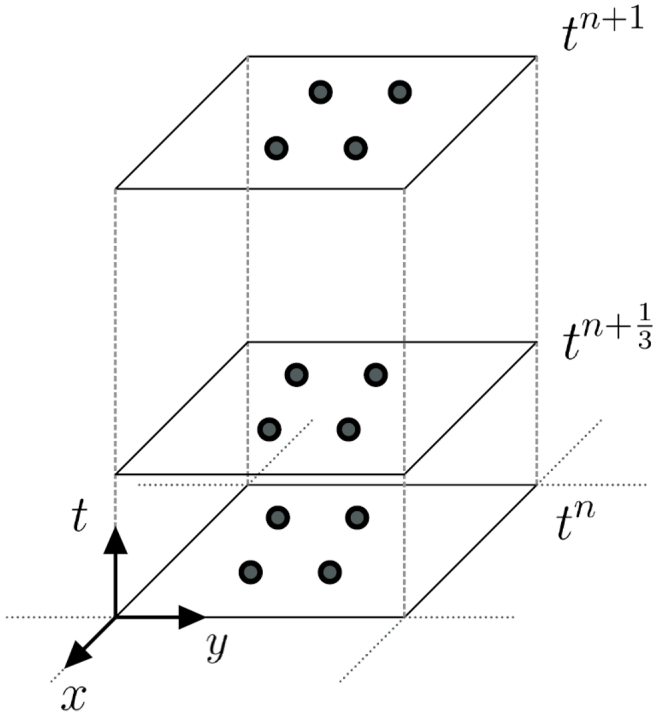


Figure 4.6: Quadrature points used to evaluate the volume integral of the flux term, highlighted on a two-dimensional space-time plot

are

$$\begin{aligned}
\zeta_0 &= \left(-\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}\right) & \zeta_4 &= \left(-\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}\right) \\
\zeta_1 &= \left(\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}\right) & \zeta_5 &= \left(-\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}\right) \\
\zeta_2 &= \left(\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}\right) & \zeta_6 &= \left(-\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}\right) \\
\zeta_3 &= \left(-\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}\right) & \zeta_7 &= \left(\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}\right)
\end{aligned}. \tag{4.60}$$

Cells can be any arbitrary polygon and so the positional definitions of these quadrature points require a trilinear mapping. The mapping for this transformation is defined as

$$\begin{aligned}
\boldsymbol{x}(\zeta, \eta, \mu) = &\frac{1-\zeta}{2}\frac{1-\eta}{2}\frac{1+\mu}{2}\boldsymbol{x}_0 + \frac{1+\zeta}{2}\frac{1-\eta}{2}\frac{1+\mu}{2}\boldsymbol{x}_1 \\
&+ \frac{1+\zeta}{2}\frac{1+\eta}{2}\frac{1+\mu}{2}\boldsymbol{x}_2 + \frac{1-\zeta}{2}\frac{1+\eta}{2}\frac{1+\mu}{2}\boldsymbol{x}_3 \\
&+ \frac{1-\zeta}{2}\frac{1+\eta}{2}\frac{1-\mu}{2}\boldsymbol{x}_4 + \frac{1-\zeta}{2}\frac{1-\eta}{2}\frac{1-\mu}{2}\boldsymbol{x}_5 \\
&+ \frac{1+\zeta}{2}\frac{1-\eta}{2}\frac{1-\mu}{2}\boldsymbol{x}_6 + \frac{1+\zeta}{2}\frac{1+\eta}{2}\frac{1-\mu}{2}\boldsymbol{x}_7.
\end{aligned} \tag{4.61}$$

This transformation represents a mapping from the physical $[x, y, z]$ coordinate system into computational $[\zeta, \eta, \mu]$ coordinates with x_{0-7} representing the nodes that make up the cell. The matrix used for this transformation is the Jacobian matrix $\boldsymbol{J}$ such that

$$\boldsymbol{J} = \frac{\partial x_i}{\partial \xi_j}, \tag{4.62}$$

where

$$\boldsymbol{J}(\zeta_j) = \frac{\partial x}{\partial \zeta}\frac{\partial y}{\partial \eta}\frac{\partial z}{\partial \mu} + \frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \mu}\frac{\partial z}{\partial \zeta} + \frac{\partial x}{\partial \mu}\frac{\partial y}{\partial \zeta}\frac{\partial z}{\partial \eta} - \frac{\partial x}{\partial \mu}\frac{\partial y}{\partial \eta}\frac{\partial z}{\partial \zeta} - \frac{\partial x}{\partial \zeta}\frac{\partial y}{\partial \mu}\frac{\partial z}{\partial \eta} - \frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \zeta}\frac{\partial z}{\partial \mu}. \tag{4.63}$$

The volume integral for the flux term is evaluated based on the points defined by the trilinear mapping. The integral for the volume flux term from Equation (4.53), based on the trilinear mapping, is evaluated as

$$\begin{aligned}
\iint_{T\times\Omega} F_i\left(\boldsymbol{U}(x_j, t)\right)\, \mathrm{d}x_j \mathrm{d}t &= \iint_{T\times\Omega} F_i\left(\boldsymbol{U}(\zeta_j, t)\right) \det\left(J(\zeta_j)\right) \mathrm{d}\zeta_j \mathrm{d}t \\
&\approx \int_T \sum_{\chi=0}^{7} w_\chi \det\left(J(\zeta_j)\right) F_i\left(\boldsymbol{U}(\zeta_j, t)\right) \mathrm{d}t,
\end{aligned} \tag{4.64}$$

such that χ represents each of the volumetric quadrature points, w_χ are the respective contributing weights of the quadrature points, and $\det(J)$ is the determinant of the Jacobian. The time integral for the flux term is computed using Radau IIA and the trapezoidal rule. Radau IIA is used

to compute the integral for time T, which goes from $[t^n, t^{n+1}]$,

$$\int_T \sum_\chi w_\chi \det(J) F_i \mathrm{d}t \approx \sum_\chi w_\chi \det(J) \cdot \left(\frac{3}{4} (F_i)_\chi^n + \frac{1}{4} (F_i)_\chi^{n+\frac{1}{3}} \right) \tag{4.65}$$

and the trapezoidal rule is used to compute the integral for time $T_{1/3}$, which goes from $[t^n, t^{n+\frac{1}{3}}]$,

$$\int_{T_{1/3}} \sum_\chi w_\chi \det(J) F_i \mathrm{d}t \approx \frac{1}{3} \sum_\chi w_\chi \det(J) \cdot \left(\frac{1}{2} (F_i)_\chi^{n+\frac{1}{3}} + \frac{1}{2} (F_i)_\chi^{n+1} \right). \tag{4.66}$$

4.4.5 Discrete Update Formulas

The treatment of both space and time integration steps for the two derived update formulas have been highlighted. The discrete update formula for the solution average update in each cell is

$$\begin{bmatrix} \alpha_0^{n+\frac{1}{3}} \\ \alpha_0^{n+1} \end{bmatrix} = \begin{bmatrix} \alpha_0^n \\ \alpha_0^n \end{bmatrix} - \frac{\Delta t}{V} \begin{bmatrix} \frac{1}{3} \sum_\xi w_\xi \boldsymbol{F}_\xi^{n+\frac{1}{6}} \\ \sum_\xi w_\xi \boldsymbol{F}_\xi^{n+\frac{1}{2}} \end{bmatrix} + \Delta t \begin{bmatrix} \frac{5}{12} & -\frac{1}{12} \\ \frac{3}{4} & \frac{1}{4} \end{bmatrix} \begin{bmatrix} \boldsymbol{S}(\alpha_0^{n+\frac{1}{3}}) \\ \boldsymbol{S}(\alpha_0^{n+1}) \end{bmatrix}, \tag{4.67}$$

and the discrete update formula for the slope average updates in each cell becomes

$$\begin{aligned}
\begin{bmatrix} \alpha_1^{n+\frac{1}{3}} \\ \alpha_2^{n+\frac{1}{3}} \\ \alpha_3^{n+\frac{1}{3}} \\ \alpha_1^{n+1} \\ \alpha_2^{n+1} \\ \alpha_3^{n+1} \end{bmatrix} = \begin{bmatrix} \alpha_1^{n} \\ \alpha_2^{n} \\ \alpha_3^{n} \\ \alpha_1^{n} \\ \alpha_2^{n} \\ \alpha_3^{n} \end{bmatrix} &+ \Delta t \begin{bmatrix} K & 0 & 0 \\ 0 & K & 0 \\ 0 & 0 & K \end{bmatrix} \begin{bmatrix} \frac{1}{3} \sum_{\chi} w_{\chi} \det(J) \cdot \left(\frac{1}{2} (F_1)_{\chi}^{n} + \frac{1}{2} (F_1)_{\chi}^{n+\frac{1}{3}} \right) \\ \frac{1}{3} \sum_{\chi} w_{\chi} \det(J) \cdot \left(\frac{1}{2} (F_2)_{\chi}^{n} + \frac{1}{2} (F_2)_{\chi}^{n+\frac{1}{3}} \right) \\ \frac{1}{3} \sum_{\chi} w_{\chi} \det(J) \cdot \left(\frac{1}{2} (F_3)_{\chi}^{n} + \frac{1}{2} (F_3)_{\chi}^{n+\frac{1}{3}} \right) \\ \sum_{\chi} w_{\chi} \det(J) \cdot \left(\frac{3}{4} (F_1)_{\chi}^{n+\frac{1}{3}} + \frac{1}{4} (F_1)_{\chi}^{n+1} \right) \\ \sum_{\chi} w_{\chi} \det(J) \cdot \left(\frac{3}{4} (F_2)_{\chi}^{n+\frac{1}{3}} + \frac{1}{4} (F_2)_{\chi}^{n+1} \right) \\ \sum_{\chi} w_{\chi} \det(J) \cdot \left(\frac{3}{4} (F_3)_{\chi}^{n+\frac{1}{3}} + \frac{1}{4} (F_3)_{\chi}^{n+1} \right) \end{bmatrix} \\
&- \Delta t \begin{bmatrix} K & 0 & 0 \\ 0 & K & 0 \\ 0 & 0 & K \end{bmatrix} \begin{bmatrix} \frac{1}{3} \sum_{\xi} w_{\xi} (x_{\xi} - x_c) \boldsymbol{F}_{\xi}^{n+\frac{1}{6}} \\ \frac{1}{3} \sum_{\xi} w_{\xi} (y_{\xi} - y_c) \boldsymbol{F}_{\xi}^{n+\frac{1}{6}} \\ \frac{1}{3} \sum_{\xi} w_{\xi} (z_{\xi} - z_c) \boldsymbol{F}_{\xi}^{n+\frac{1}{6}} \\ \sum_{\xi} w_{\xi} (x_{\xi} - x_c) \boldsymbol{F}_{\xi}^{n+\frac{1}{2}} \\ \sum_{\xi} w_{\xi} (y_{\xi} - y_c) \boldsymbol{F}_{\xi}^{n+\frac{1}{2}} \\ \sum_{\xi} w_{\xi} (z_{\xi} - z_c) \boldsymbol{F}_{\xi}^{n+\frac{1}{2}} \end{bmatrix} \\
&+ \Delta t \begin{bmatrix} \frac{5}{12} I & -\frac{1}{12} I \\ \frac{3}{4} I & \frac{1}{4} I \end{bmatrix} \begin{bmatrix} \left(\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{U}} \right)^{n+\frac{1}{3}} \alpha_1^{n+\frac{1}{3}} \\ \left(\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{U}} \right)^{n+\frac{1}{3}} \alpha_2^{n+\frac{1}{3}} \\ \left(\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{U}} \right)^{n+\frac{1}{3}} \alpha_3^{n+\frac{1}{3}} \\ \left(\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{U}} \right)^{n+1} \alpha_1^{n+1} \\ \left(\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{U}} \right)^{n+1} \alpha_2^{n+1} \\ \left(\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{U}} \right)^{n+1} \alpha_3^{n+1} \end{bmatrix}.
\end{aligned} \tag{4.68}$$

4.5 Implementation Overview for the GPU-Based DGH Solver

The GPU-based implementation of DGH is developed within a C++ programming environment to target hardware accelerators with the SYCL programming model.

4.5.1 Load Balancing

In order to design the code to target multiple GPUs on large heterogeneous computing facilities, the implementation follows a block-based approach that distributes the global problem. The discretized initial-value problem presented in Figure 4.1 can be segmented into multiple blocks

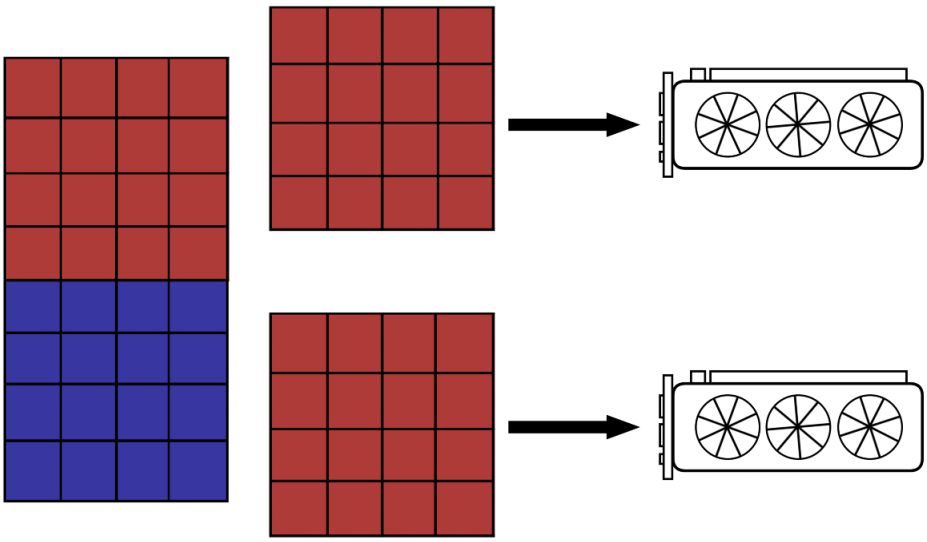


Figure 4.7: Block-based distribution of the global problem

and distributed across different devices. Figure 4.7 provides an example of segmented blocks that are distributed to different GPUs. Currently, the GPU-based implementation is only being used on structured meshes that are evenly divided into blocks with the same number of cells. Load-balancing simply performs an even distribution of blocks across all available GPUs.

4.5.2 Communication with the Message Passing Interface

The message passing interface (MPI) is a library that can be used in the development of C++ programs to provide efficient communication directives [30]. MPI facilitates communication between distinct processes that are working in a parallel computing environment and scales very well to modern supercomputers. In general, large GPU-based computing facilities have many compute nodes that each have a set number of GPUs. Functionalities in the MPI library can be used to transfer data between the GPUs on each of these nodes. This capability is essential to facilitate the exchange of data with an optimized message passing system. In the GPU implementation of DGH, the GPUs that exist on each node have shared data buffers with the CPU process. By using MPI, each CPU process on each node can efficiently communicate data between local GPUs on its own node and global GPUs with other nodes. This enables each GPU to send and receive the solution states at boundary quadrature points that are shared with their neighbours. An example of the solution information that needs to be communicated between the quadrature points at boundary cells is provided in Figure 4.8. Communication must occur between different GPUs that contain neighbouring blocks. Send and receive buffers are established between the GPUs to communicate solution information at the shared boundary quadrature points, as shown in Figure 4.9. Each GPU is designated a “Compute Instance” object that identifies it based on the MPI rank of its host CPU and its local index. Each MPI send and receive message must have it’s own unique tag and each tag is generated based on the Compute Instance of the sending and receiving devices.

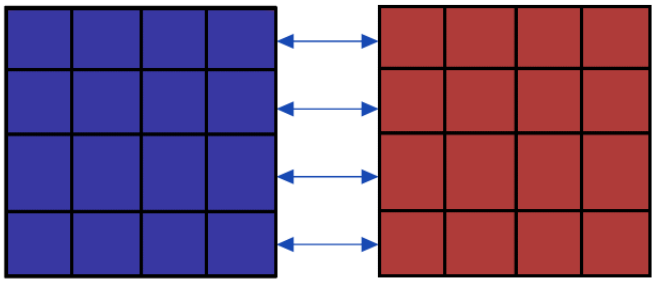


Figure 4.8: Example of boundary communication through quadrature points

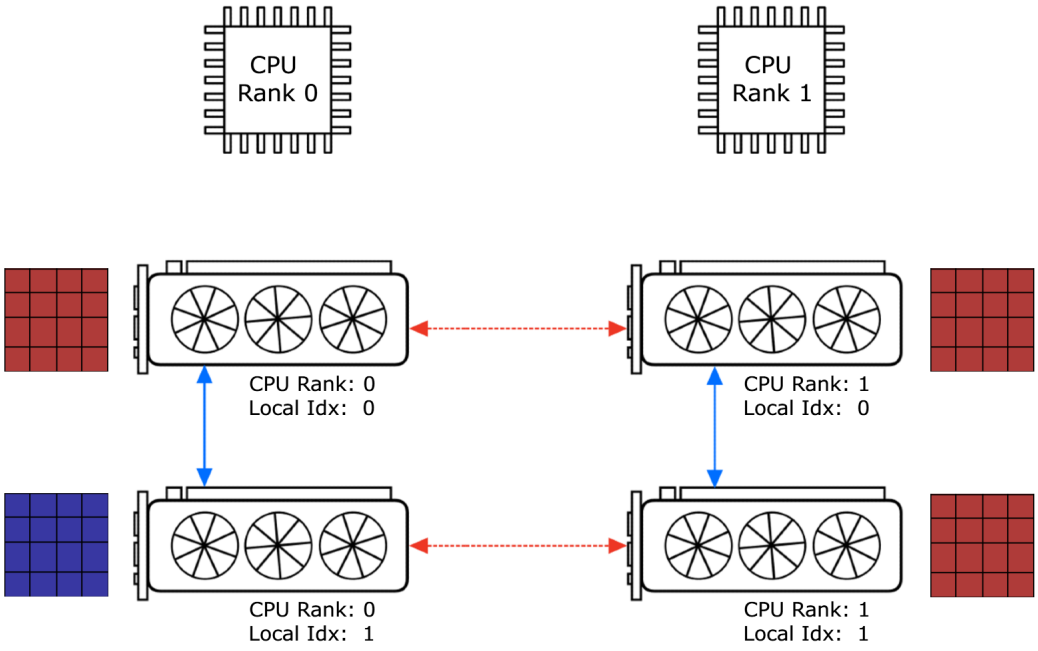


Figure 4.9: Communication on GPU-based platforms

4.5.3 Time-Marching the DGH Implementation

Equation (4.67) and Equation (4.68) are the discrete update formulas that must be computed for each cell. Updates are computed at every timestep when stepping forward in time. Since updates at quadrature points are largely independent of one another, updates at each quadrature point are computed in parallel with GPU kernels using only local information. The updates are stored and then a second GPU kernel performs the updates for each cell in order to avoid race conditions. Information from neighbouring cells is only required when computing the solution to Riemann problems at boundary quadrature points. Since the majority of computations only require local work, DGH is an inherently parallel problem that scales very well on GPU-based platforms. A brief overview of time-marching steps involved in the GPU-based implementation of DGH include:

- Solution states that inputs to Riemann problems are computed with the Hancock predictor step at each of the boundary quadrature points and sent to neighbouring GPUs with non-blocking messages.
- Interface fluxes are computed at the internal quadrature points of each cell and the cells of each block are updated.
- All GPUs wait to receive information from the computed Hancock predictor step. Interface fluxes at the boundary quadrature points are computed and then all boundary cells are updated.
- The solution average value is updated based on the computed interface flux terms and the source term.
- The volume integral for the flux term is computed at each volume quadrature point and added to the slope update in each cell.
- The slope average value is updated based on the computed interface flux terms, the volume integrals for the flux terms, and the source term.

4.6 Numerical Results from DGH

This section of the chapter highlights a selection of numerical results that have been produced with the GPU-based implementation of DGH.

4.6.1 Confirming the Order of Accuracy

One of the main advantages that come with using the DGH scheme is the order of accuracy that can be achieved. It is really important to confirm that third-order accuracy is still maintained with the GPU-based implementation of this numerical method because it helps to confirm that

the implementation is correct. A convergence study is conducted with the linear convection relaxation equation. This PDE takes the form

$$\frac{\partial \rho}{\partial t} + u_i \frac{\partial \rho}{\partial x_i} = -\frac{\rho}{\tau}. \tag{4.69}$$

In this equation, the bulk velocity is constant and the PDE has an exact solution that is known to be

$$\rho(x_i, t) = \rho_0(x_i - u_i t) \exp\left(-\frac{t}{\tau}\right). \tag{4.70}$$

A convergence study is conducted for the linear convection equations in both two and three spatial dimensions. In two-dimensional space, the balance law form follows

$$\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial \boldsymbol{F}_x}{\partial x} + \frac{\partial \boldsymbol{F}_y}{\partial y} = \boldsymbol{S}, \tag{4.71}$$

where the solution term $\boldsymbol{U} = \rho$, the x-direction flux vector $\boldsymbol{F}_x = \rho u_x$, the y-direction flux vector $\boldsymbol{F}_y = \rho u_y$, and the source term $\boldsymbol{S} = -\frac{\rho}{\tau}$. The initial value problem is computed on the domain

$$(x, y) \in \mathbb{R}^2 \mid -10 \leq x \leq 10,\ -10 \leq y \leq 10 \tag{4.72}$$

with the initial conditions

$$\rho_0(x, y) = e^{-0.5(x^2+y^2)}, \tag{4.73}$$

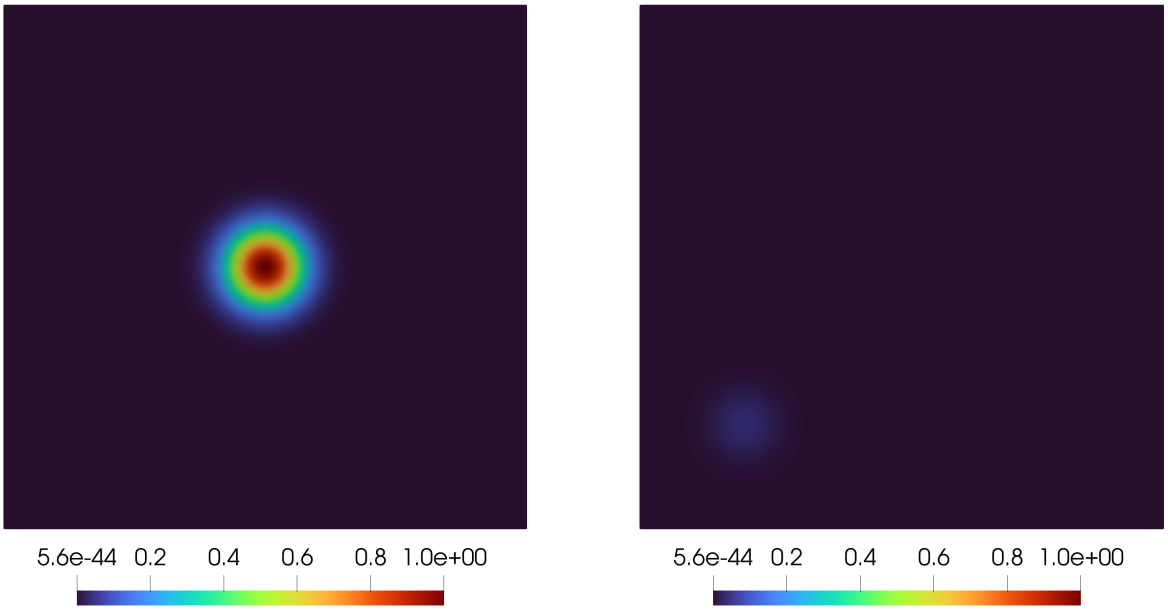
such that the exact solution to the problem is

$$\rho_F(x, y) = e^{-0.5((x-6)^2+(y-6)^2)-3}, \tag{4.74}$$

for constant velocity components $u_x = u_y = -2$, a relaxation time $\tau = 1$, and a final time $t = 3$. Figure 4.10 provides a visualization for the initial conditions at $t = 0$ and the final solution at $t = 3$. The visualized results demonstrate expected behaviours for the flow. The velocities, u_x and u_y, move the flow towards the negative x and y directions, respectively. The results also demonstrate the dissipating effects from the relaxation term. Determining the convergence depends on the ℓ^2 Error, given by

$$\ell^2 \text{ Error} = \sqrt{\sum_{i=0}^{N} \left(\frac{1}{V_i} \sum_{\chi} w_\chi \rho_{F_{(\chi)i}} - \rho_{\text{DGH}_i} \right)^2 \times V_i}, \tag{4.75}$$

where N is the number of cells in the discretized domain and V_i is the volume of each cell. The variable $\boldsymbol{U}_{\text{DGH}_i}$ is the solution average in each cell and $\boldsymbol{U}_{F_{(\chi)i}}$ is the exact solution computed at the volume quadrature points in each cell. It is important to note that the exact average solution is computed as an integrated quantity at the volume quadrature points in each cell. The order of



(a) Initial condition (b) Final solution

Figure 4.10: Linear convection 2D relaxation initial conditions and solution.

Table 4.1: Linear convection 2D convergence results

# Elements	ℓ^2 **Error**	**Order**
300×300	1.43374×10^{-5}	—
400×400	6.06895×10^{-6}	2.99
500×500	3.11309×10^{-6}	2.99
600×600	1.80296×10^{-6}	3.00

convergence is computed as

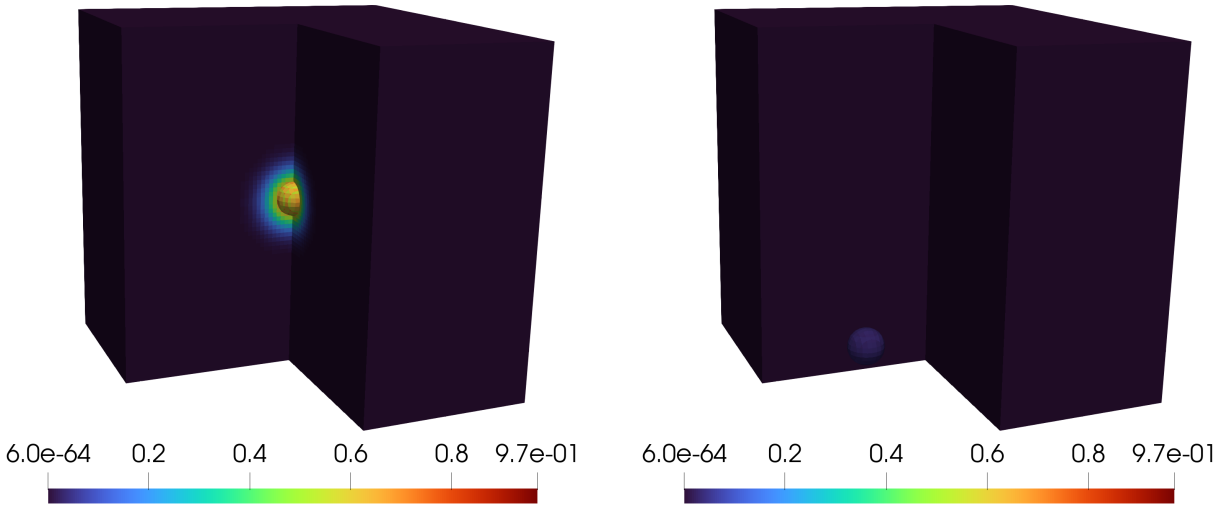
$$\text{Order of Convergence} = \frac{\ln\left(\frac{\text{Error}_1}{\text{Error}_2}\right)}{\ln\left(\frac{N_1}{N_2}\right)}, \tag{4.76}$$

where Error_1 and Error_2 are the computed ℓ^2 Errors for lower and higher resolutions, respectively. N_1 and N_2 are the number of elements along a single dimension on the grid. Convergence results for the two-dimensional linear convection relaxation equations are presented in Table 4.1. Results in the table demonstrate that third-order accuracy is recovered in the GPU-based implementation of the DGH scheme. Similarly, in three-dimensional space, the balance law form follows

$$\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial \boldsymbol{F}_x}{\partial x} + \frac{\partial \boldsymbol{F}_y}{\partial y} + \frac{\partial \boldsymbol{F}_z}{\partial z} = \boldsymbol{S}. \tag{4.77}$$

where the z-direction flux vector $\boldsymbol{F}_z = \rho u_z$. The initial value problem is computed on the domain

$$(x, y, z) \in \mathbb{R}^3 \mid -10 \leq x \leq 10,\ -10 \leq y \leq 10,\ -10 \leq z \leq 10. \tag{4.78}$$



(a) Initial condition (b) Final solution

Figure 4.11: Linear convection 3D relaxation initial conditions and solution.

Table 4.2: Linear convection 3D convergence results

# Elements	ℓ^2 Error	Order
$40 \times 40 \times 40$	9.973186×10^{-3}	—
$80 \times 80 \times 80$	1.780187×10^{-3}	2.5
$120 \times 120 \times 120$	5.674075×10^{-4}	2.82
$160 \times 160 \times 160$	2.454975×10^{-4}	2.91
$200 \times 200 \times 200$	1.271323×10^{-4}	2.95

The initial conditions for the solution term is

$$\rho_0(x, y, z) = e^{-0.5(x^2+y^2+z^2)}, \tag{4.79}$$

such that the exact solution to the problem is

$$\rho_F(x, y, z) = e^{-0.5((x-6)^2+(y-6)^2+(z-6)^2)-3}, \tag{4.80}$$

given an added z-direction velocity component of $u_z = -2$. Figure 4.11 provides a visualization for the initial conditions at $t = 0$ and the final solution at $t = 3$. Again, as expected from the figure, the flow moves towards the negative x, y, and z directions with dissipating effects. Convergence results for the three-dimensional linear convection relaxation equations are presented in Table 4.2. The table confirms third-order accuracy based on the presented orders of convergence. These results demonstrate that third-order accuracy is recovered in the GPU-based implementation of the DGH scheme in both two and three spatial dimensions.

4.6.2 Strong Scaling Study

Parallel scaling studies are also performed for the GPU-based implementation of DGH in order to analyze the performance of the code when distributed across massively parallel computing platforms. These scaling studies are performed for a two-dimensional shockbox test case, using the ten-moment equations presented in Chapter 2. The shockbox test case in this section follows the previously introduced initial condition of a low density and low pressure gas, $\mathbf{W}_{\text{low}}$, that is interfaced with a high density and high pressure gas, $\mathbf{W}_{\text{high}}$. The initial conditions for the two states are

$$\mathbf{W}_{\text{low}} = \begin{bmatrix} \rho \\ u_x \\ u_y \\ P \end{bmatrix} = \begin{bmatrix} 0.30625\,\text{kg/m}^3 \\ 0\,\text{m/s} \\ 0\,\text{m/s} \\ 25\,331.25\,\text{Pa} \end{bmatrix} \tag{4.81}$$

and

$$\mathbf{W}_{\text{high}} = \begin{bmatrix} \rho \\ u_x \\ u_y \\ P \end{bmatrix} = \begin{bmatrix} 1.225\,\text{kg/m}^3 \\ 0\,\text{m/s} \\ 0\,\text{m/s} \\ 101\,325\,\text{Pa} \end{bmatrix}. \tag{4.82}$$

Initial conditions for the problem are set up on the domain

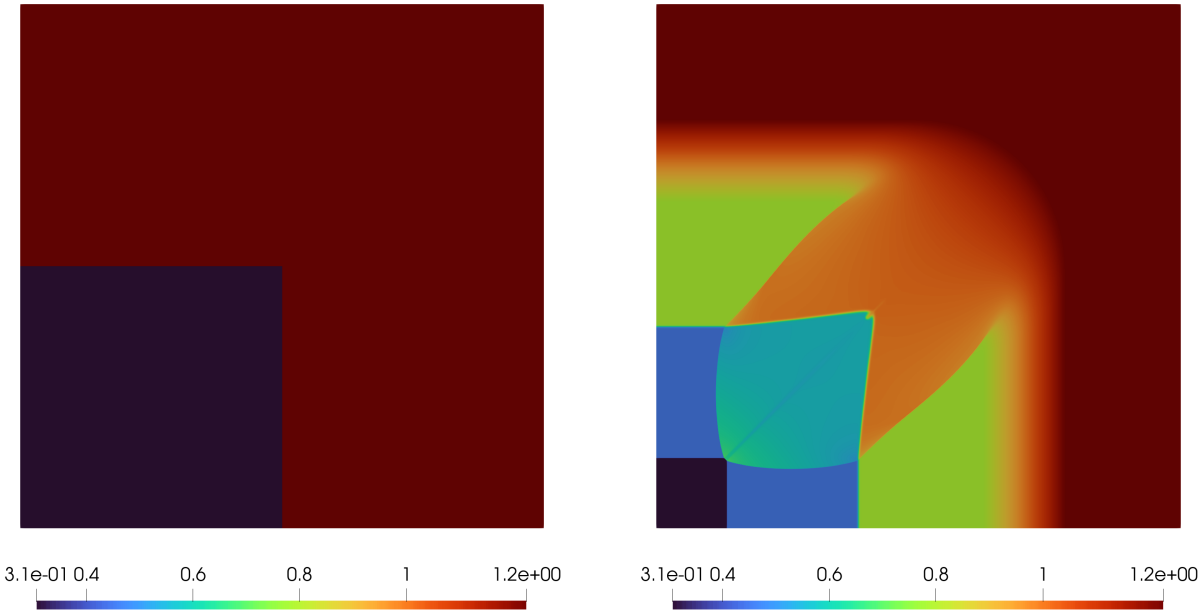
$$(x, y) \in \mathbb{R}^2 \mid -0.5 \leq x \leq 0.5,\ -0.5 \leq y \leq 0.5, \tag{4.83}$$

such that

$$\mathbf{W_0}(\mathbf{x}, \mathbf{y}) = \begin{cases} \mathbf{W}_{\text{low}} & \text{for } x < 0 \text{ and } y < 0, \\ \mathbf{W}_{\text{high}} & \text{otherwise.} \end{cases} \tag{4.84}$$

The solution for this initial value problem is computed with $\tau = 1.0 \times 10^{-3}$ and the test case was time-marched to a final time of $t = 0.75 \times 10^{-3}$s. The CFL number is 0.3, the slope limiter is Venkatakrishnan [31], and the Riemann solver is HLLE [32]. A visual representation of the initial condition and the final solution is provided in Figure 4.12. The figure demonstrates the development of shock waves at the interface of the two initial states. Two strong scaling studies were conducted for this initial-value problem. The first strong scaling study was conducted for a fixed problem size of 4 million elements and the second was conducted for a fixed problem size of 16 million elements. The parallel speedup S_n is measured as the time required for a single process to complete the computation t_1 relative to the the time required for multiple processes to complete the computation t_n. The speed-up is given by

$$S_n = \frac{t_1}{t_n}. \tag{4.85}$$



(a) Initial condition (b) Final solution

Figure 4.12: Shockbox initial conditions and final solution for the Ten Moment 2D equations

Table 4.3: Performance metrics using DGH for multiple GPUs (4 million elements)

# GPUs	Time (s)	Speedup	Efficiency
2	660.014	2.00	1.00
4	334.25	3.95	0.99
8	170.52	7.75	0.97
16	88.12	15.00	0.94

The parallel efficiency E_n is the ratio of the speed-up S_n against the number of processes n, where

$$E_n = \frac{S_n}{n}. \tag{4.86}$$

These values are used to help determine how well the implementation scales to large heterogeneous computing facilities for GPU targeted simulations. An ideal parallel efficiency is $E_n = 1$. Table 4.3 shows the parallel scaling across a different number of GPUs for the 4 million element problem size. Figure 4.13 visualizes the results. This first study demonstrates near perfect parallel scaling across the GPUs with a relatively linear increase in the parallel speed-up and ideal parallel efficiency. The parallel efficiency when distributed up to 16 GPUs is 0.94, which demonstrates desirable results. Scaling the problem size up to 16 million elements can help expose any potential bottlenecks that might arise from sending and receiving larger data buffers during the communication step. Table 4.4 presents the tabulated results for the 16 million element problem size. The visualized results are shown in Figure 4.14. When performed on the larger problem size, the scaling study continues to demonstrate near perfect parallel efficiency with almost ideal parallel speed-ups. This shows that the implementation is not affected by larger communication

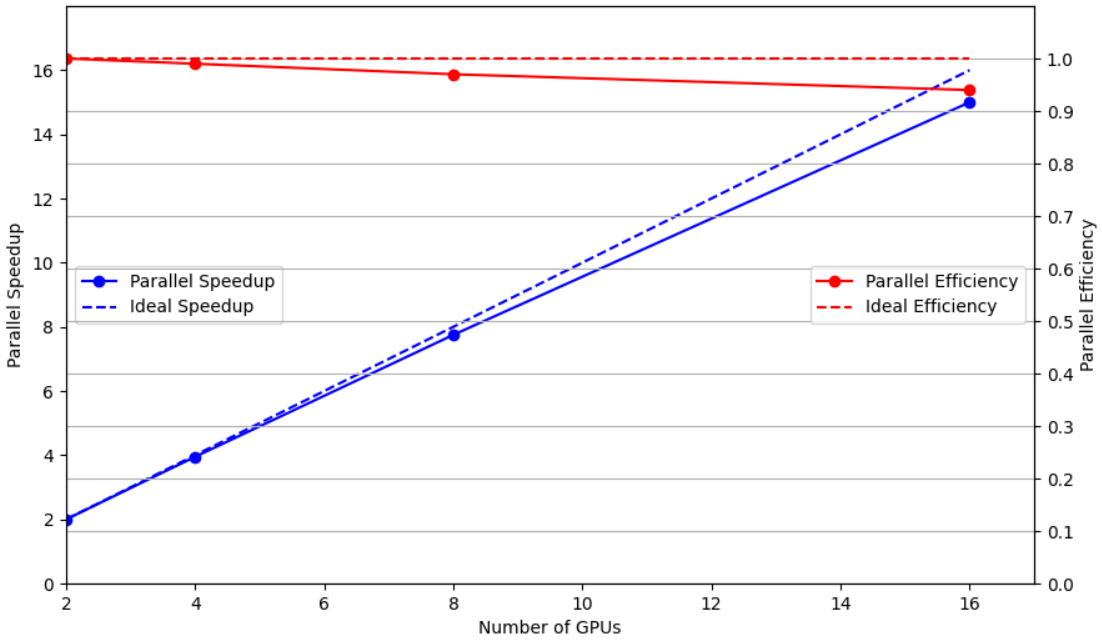


Figure 4.13: Parallel scaling study on multiple GPUs for a problem-size of 4 million elements

Table 4.4: Performance metrics using DGH for multiple GPUs (16 million elements)

# GPUs	Time (s)	Speedup	Efficiency
4	2723.11	4.00	1.00
8	1388.59	7.84	0.98
16	718.3	15.17	0.95

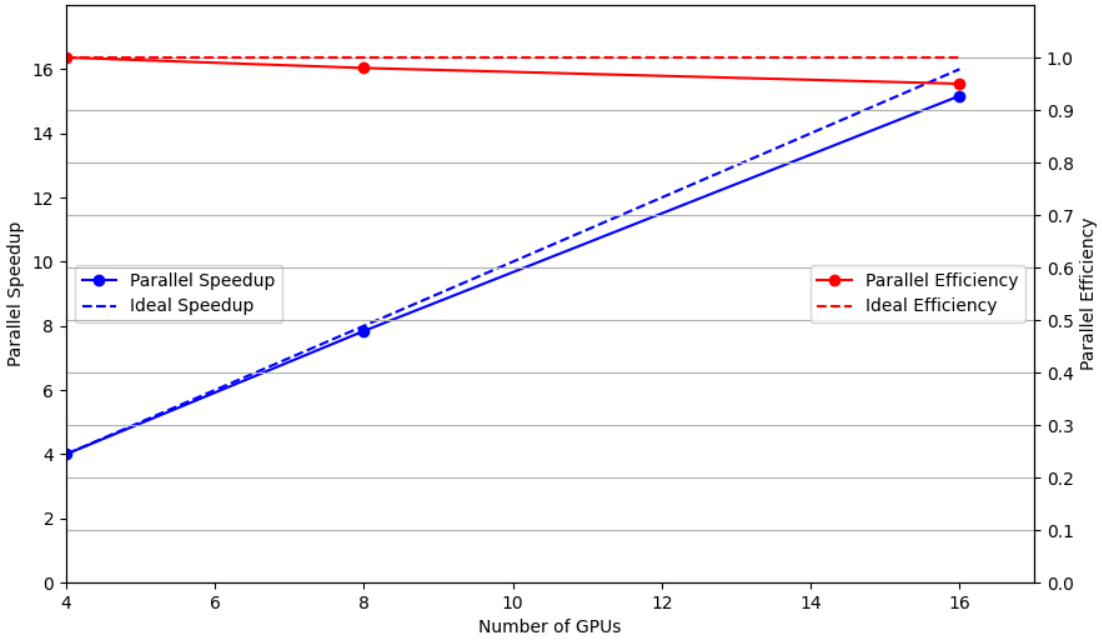


Figure 4.14: Parallel scaling study on multiple GPUs for a problem-size of 16 million elements

Table 4.5: Computational runtimes using DGH for multiple CPUs (16 million elements)

# CPU Cores	Time (s)
256	4893.99
512	2915.71
1024	1648.94

buffers and continues to present great scalability with larger problem sizes.

4.6.3 Performance Comparisons

The computational performance between the CPU and GPU versions of DGH were compared using the 16 million element ten-moment shockbox test case. Performance results for the CPU version of the code, with the AMD Rome 7532, are presented in Table 4.5. It is also worth noting that every 32 CPU cores represents a single AMD Rome 7532 processor. Performance comparisons across multiple CPUs and GPUs are presented in Figure 4.15. Additionally, the estimated energy expenditures for the CPU and GPU implementations are presented in Figure 4.16. The GPU implementation of the code presents significant parallel speedups and very desirable values for energy expenditures. Demonstrating both speedups in computational runtimes and a decrease in energy consumption, the GPU-based implementation for DGH can be leveraged as a very time and energy efficient numerical algorithm for solving a wide range of scientific problems.

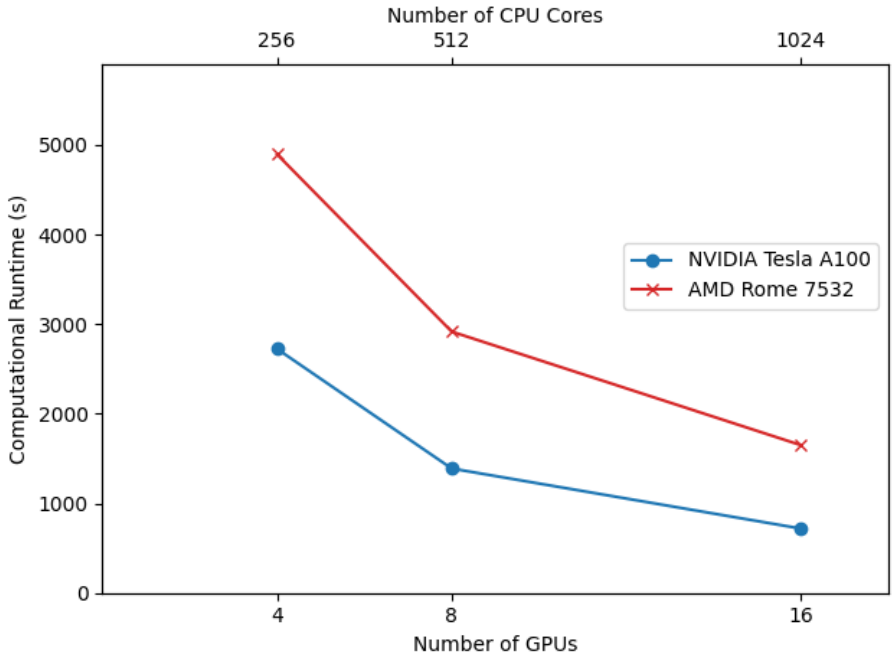


Figure 4.15: Computational runtime comparison between CPUs and GPUs using DGH for a 16 million element study

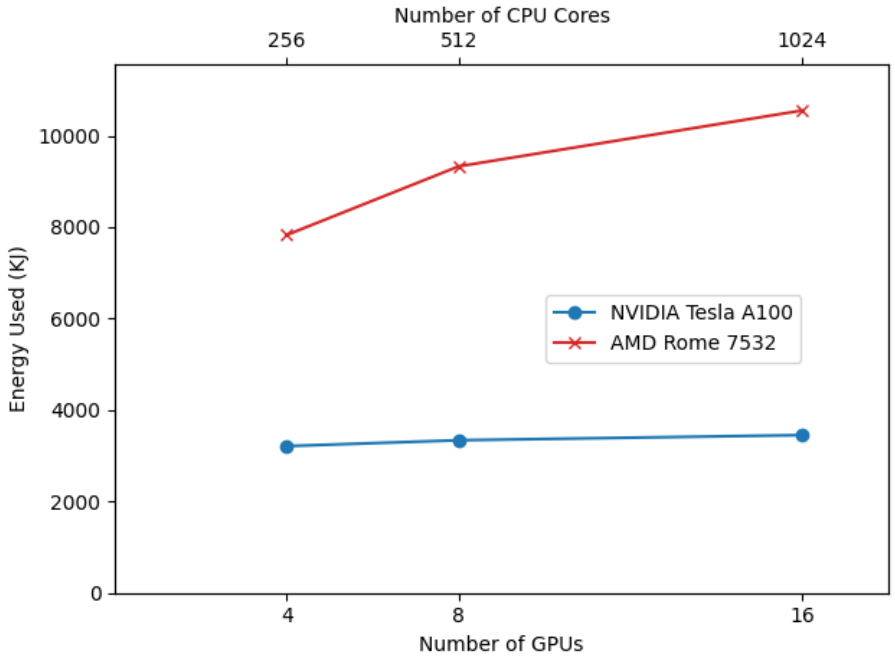


Figure 4.16: Energy expenditure comparison between CPUs and GPUs using DGH for a 16 million element study

4.6.4 Kelvin-Helmholtz Instability

A Kelvin-Helmholtz instability occurs when two fluids that each have a different uniform density are interfaced with one another. The two fluids also have different velocities and sinusoidal perturbations at their shared interface. The dense gas $\mathbf{W}_{\text{middle}}$ has an initial state

$$\mathbf{W}_{\text{middle}} = \begin{bmatrix} \rho \\ u_x \\ u_y \\ P \end{bmatrix} = \begin{bmatrix} 2.0\,\text{kg/m}^3 \\ 0.5\,\text{m/s} \\ 0\,\text{m/s} \\ 2.5\,\text{Pa} \end{bmatrix}, \tag{4.87}$$

and the less dense gas $\mathbf{W}_{\text{outer}}$ has initial conditions,

$$\mathbf{W}_{\text{outer}} = \begin{bmatrix} \rho \\ u_x \\ u_y \\ P \end{bmatrix} = \begin{bmatrix} 1.0\,\text{kg/m}^3 \\ -0.5\,\text{m/s} \\ 0\,\text{m/s} \\ 2.5\,\text{Pa} \end{bmatrix}. \tag{4.88}$$

The lower interface y_{lower} and the upper interface y_{upper} are defined to be

$$y_{\text{lower}} = 0.25 \cdot y_{\text{max}} + 0.015 \cdot \cos\left(\frac{4\pi}{3} \cdot x\right) \tag{4.89}$$

and

$$y_{\text{upper}} = 0.75 \cdot y_{\text{max}} + 0.015 \cdot \cos\left(\frac{4\pi}{3} \cdot x\right). \tag{4.90}$$

where y_{max} is the maximum y value in the range. The initial condition $\mathbf{W_0}$ is such that

$$\mathbf{W_0}(\mathbf{x}, \mathbf{y}) = \begin{cases} \mathbf{W}_{\text{middle}}, & \text{if } y_{\text{lower}} < y < y_{\text{upper}} \\ \mathbf{W}_{\text{outer}}, & \text{otherwise} \end{cases} \tag{4.91}$$

on the domain

$$(x, y) \in \mathbb{R}^2 \mid 0.0 \leq x \leq 3.0,\ 0.0 \leq y \leq 1.5. \tag{4.92}$$

The test case was run with periodic boundary conditions at the x-direction limits, a final time of $t = 2\,\text{s}$, and a resolution of 128 million elements. There are 16000 cells in the x-direction and 8000 cells in the y-direction. The initial conditions are presented in Figure 4.17 for reference. The final results for the density of the gas flow are shown in Figure 4.18. The final results show that the amplitude of the perturbations has increased with the development of detailed structure. These results were produced with a computational runtime of 75 minutes when scaled to target 64 Nvidia A100 GPUs. This test case serves to further demonstrate the ability of the GPU implementation of DGH to quickly produce high resolution results when distributed across large GPU-based supercomputers.

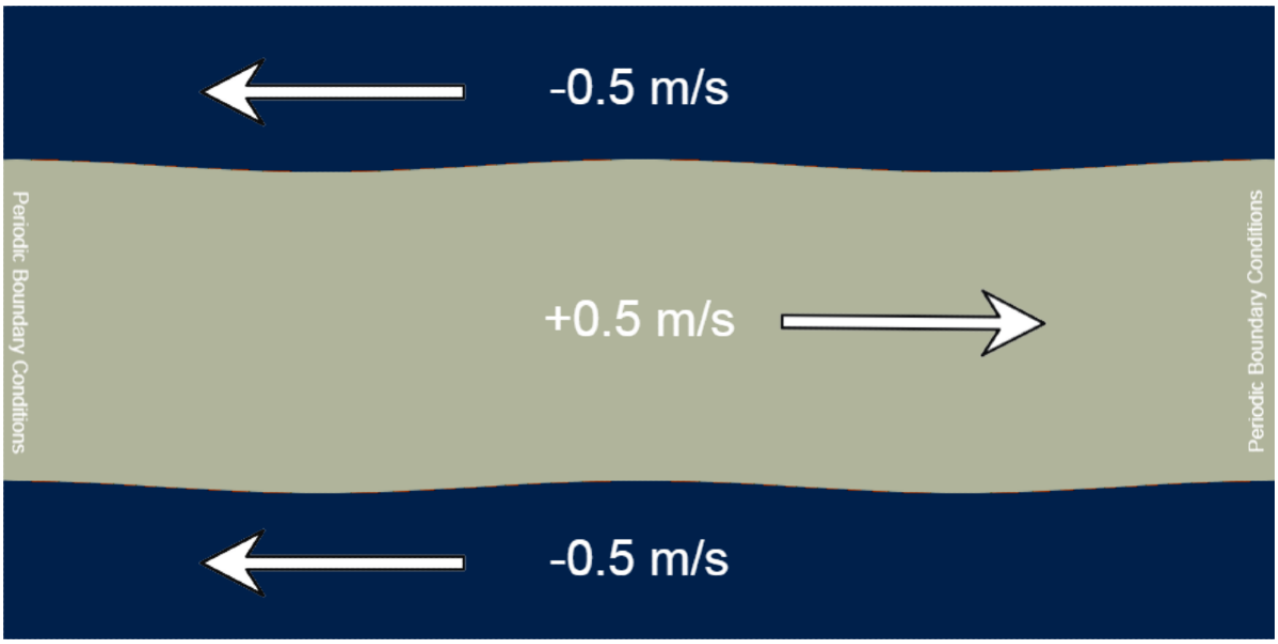


Figure 4.17: Initial conditions for a Kelvin-Helmholtz instability case

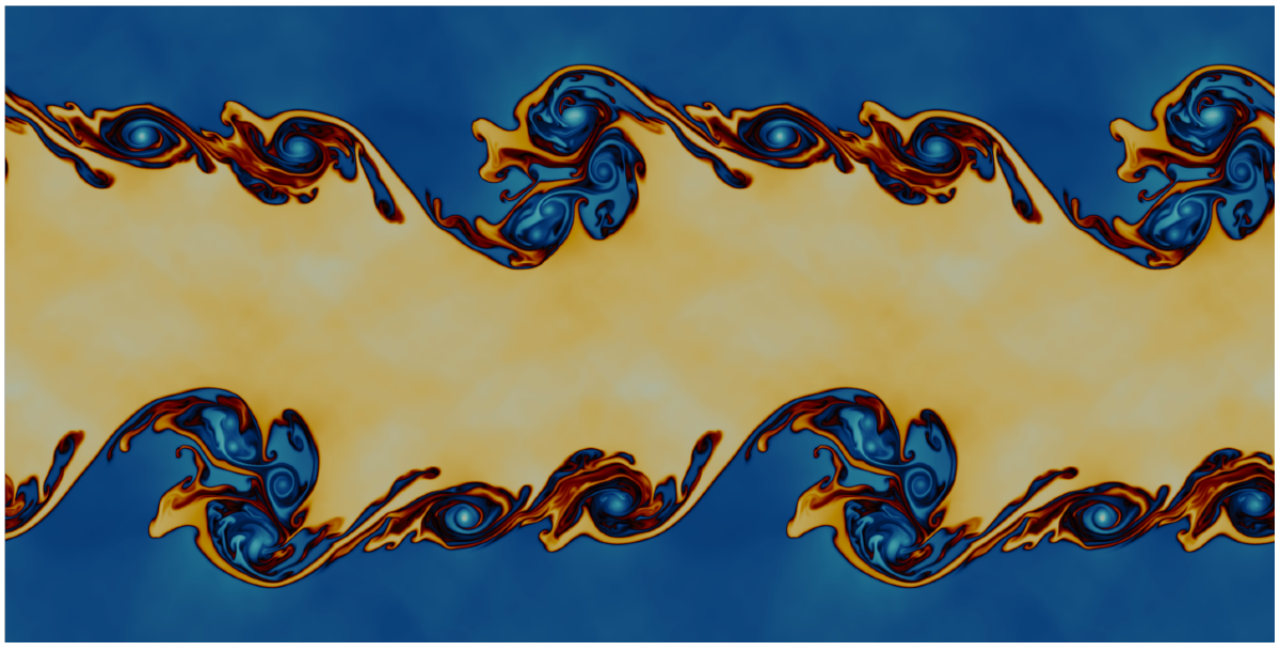


Figure 4.18: Structure development in a Kelvin-Helmholtz instability case

Chapter 5

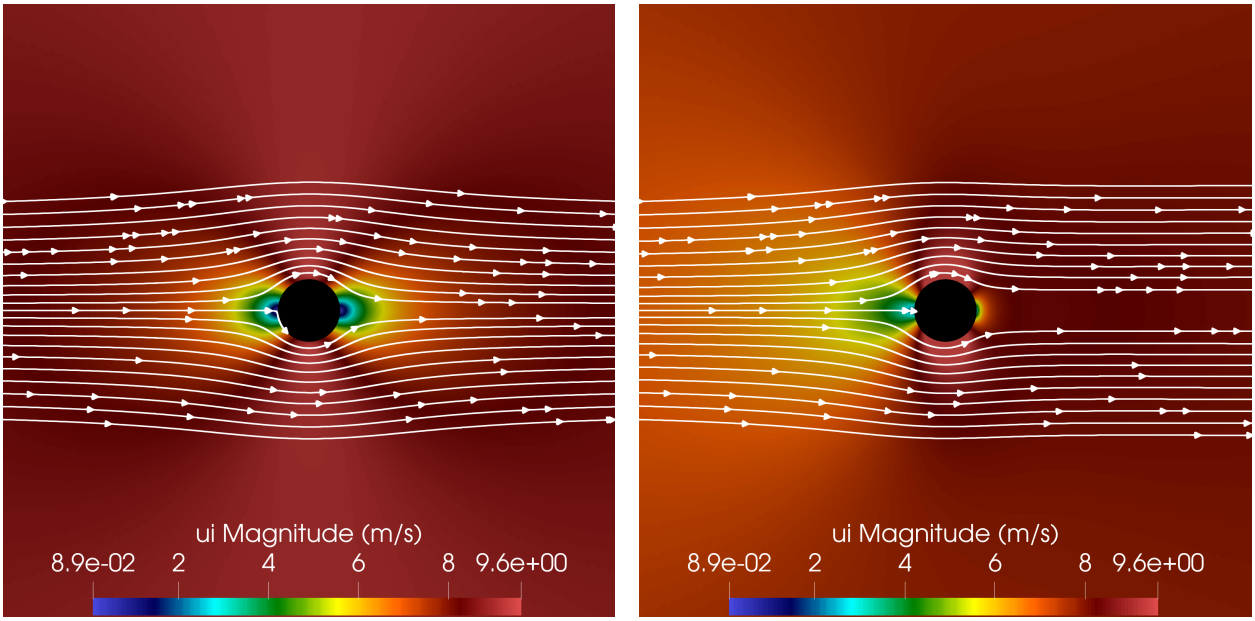
Model Validation Demonstration

As previously mentioned in the introduction of this thesis, one of the main objectives is to leverage the FMFC solver as a validation tool to benchmark the accuracy of models derived from moment closure methods. This section of the report presents a validation study, in the free-molecular regime, for the two-dimensional ten-moment equations. The ten-moment equations are known to be inaccurate in the free-molecular regime. The purpose of this study is simply to show the effectiveness of the FMFC solver as a validation tool that can eventually be used to benchmark the accuracy of moment methods that are currently being developed when solved with numerical schemes such as DG and DGH.

5.1 Benchmarking the Ten-Moment Equations in the Free-Molecular Regime

In this section of the chapter, benchmarks are produced using the FMFC solver and compared against the ten-moment equations that are solved with the first-order DG scheme that was briefly introduced in Chapter 5. The first-order DG scheme is used in place of DGH for this study because regions of low density can be challenging to accurately resolve with DGH for the ten-moment equations. It is also worth noting that results from the FMFC solver and the ten-moment equations are mostly visualized with colour maps of different scales because of the significant discrepancies between the two solutions. This chapter does not present a true scientific validation study but simply a demonstration for the intended use of the FMFC solver. In this demonstration, initial conditions for the free-stream flow are

$$\mathbf{W} = \begin{bmatrix} \rho \\ P \\ u_x \\ m \\ R \\ \omega \end{bmatrix} = \begin{bmatrix} 1.7838\,\text{kg/m}^3 \\ 101325\,\text{Pa} \\ 9.1\,\text{m/s} \\ 6.63 \times 10^{-26}\,\text{kg} \\ 1\,\text{m} \\ 0\,\text{rad/s} \end{bmatrix}. \tag{5.1}$$



(a) benchmark result from the FMFC solver (b) ten-moment equations solved with a DG scheme

Figure 5.1: Average velocity comparison: FMFC solver and the ten-moment equations.

The relaxation time for the ten-moment equations is set to

$$\tau \to \infty \tag{5.2}$$

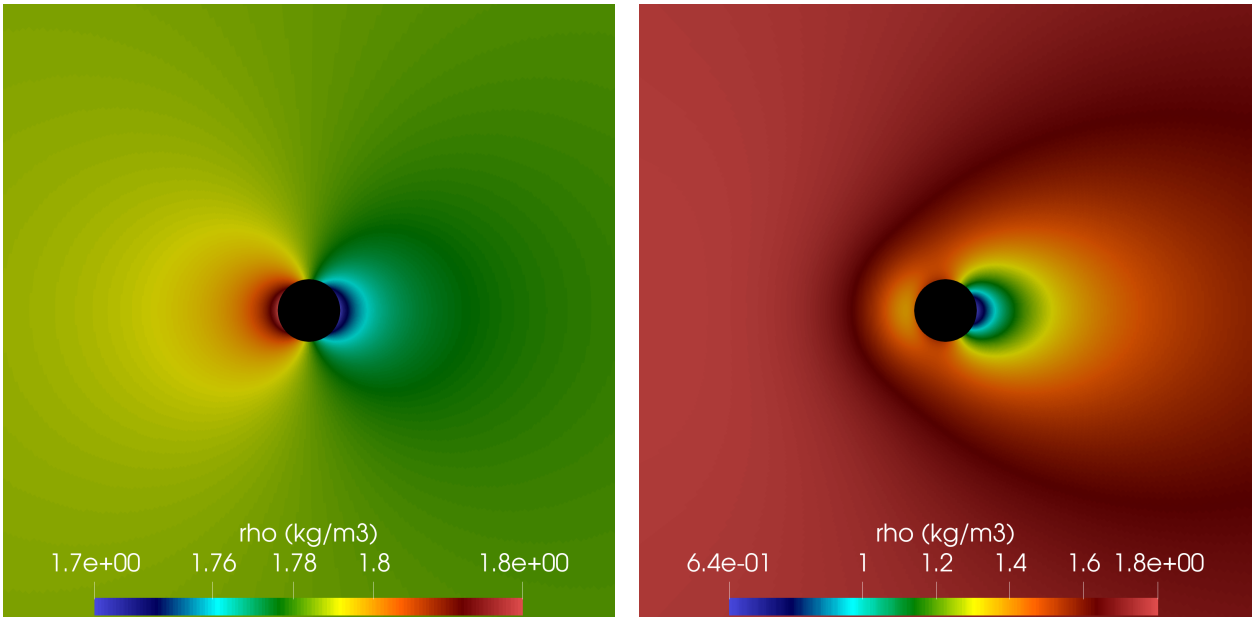
such that the ten-moment equations are modelling free-molecular flow. The radial domain around the cylinder is $r \in [1, 100]$ m with a grid size 250000 elements. Boundary conditions with the cylinder are completely specular so the accommodation coefficient for the FMFC solver is

$$\alpha = 0 \tag{5.3}$$

and the ten-moment equations are solved with reflection boundary conditions at the cylinder surface. Furthermore, the ten-moment equations are run to a final time of $t = 25$ s where the result has converged towards a steady-state solution.

5.1.1 Comparing the Velocity Field

A comparison for the average velocities of the FMFC solver and the ten-moment equations is shown in Figure 5.1. Both the benchmark results from the FMFC solver and the solution from the ten-moment equations present average velocities of similar magnitude. The two gas flows show stagnation points at the front and back of the cylinder. The two results also show a similar behavior at the top and bottom of the cylinder where the average velocity is the free-stream velocity of the gas flow. Despite the similar magnitudes and behaviors of the average velocities in the two solutions, the profiles do not exactly match. This demonstrates the limits of the ten-moment equations when attempting to accurately capture gas flows in the free-molecular regime.



(a) benchmark result from the FMFC solver (b) ten-moment equations solved with a DG scheme

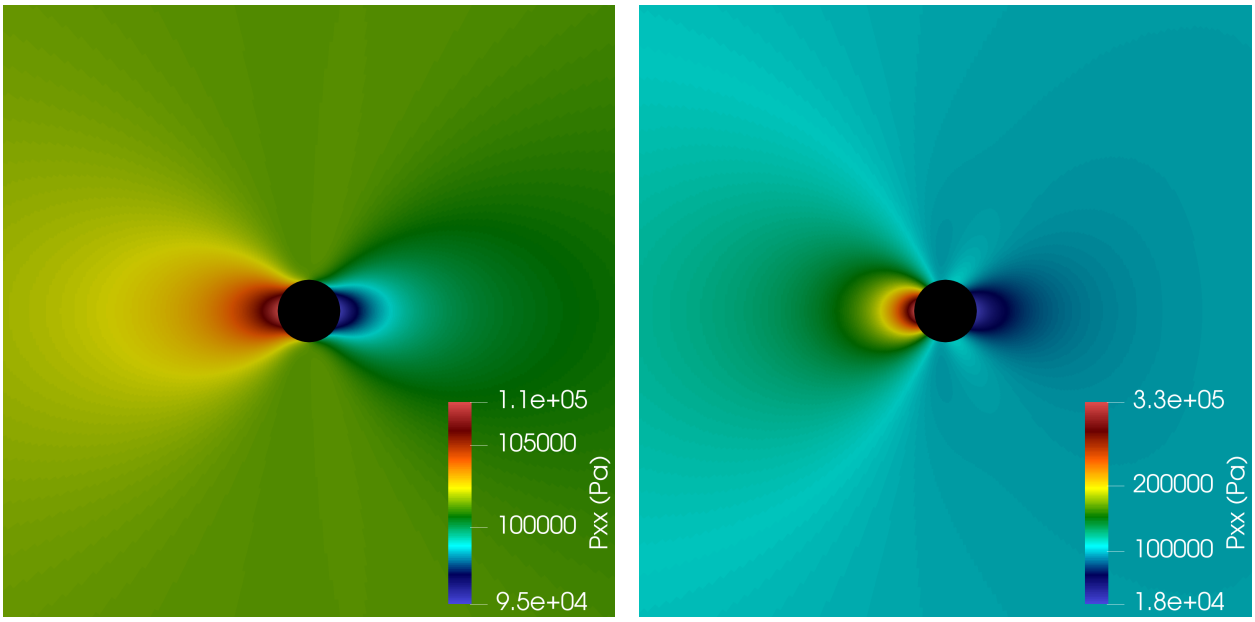
Figure 5.2: Density comparison: FMFC solver and the ten-moment equations.

5.1.2 Comparing the Density Field

The density field for the two gas flows have also been studied. Figure 5.2 presents the comparison for the density of the benchmark result produced with the FMFC solver and the ten-moment equations that are solved with the DG scheme. The visual output shows that the ten-moment equations have areas of low density at the front of the cylinder and areas of much lower density behind the cylinder. The FMFC solver demonstrates an expected behaviour of high density at the front of the cylinder and low density behind the cylinder. It is clear that the ten-moment equations do not accurately capture the density field, demonstrating that these equations fail in the free-molecular regime.

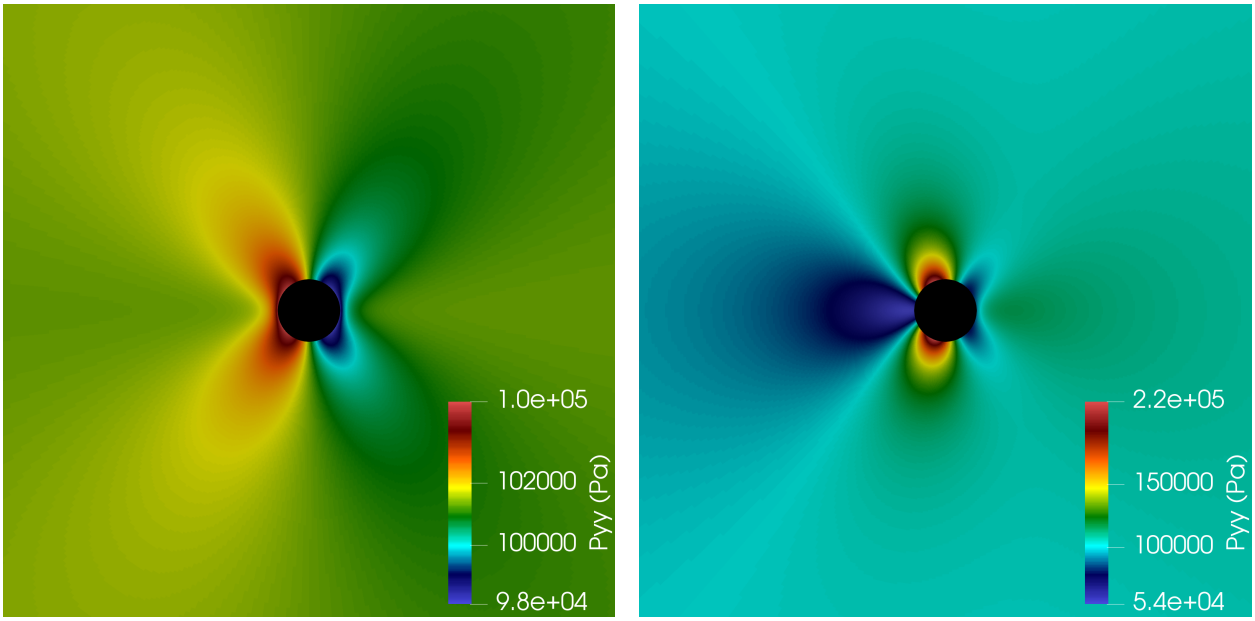
5.1.3 Comparing the Pressure Fields

Another set of comparisons that can be made is in the computed pressure tensor from the two solvers. Figure 5.3 compares the normal stress along the x-axis, P_{xx}. Results from both the FMFC solver and the ten-moment equations show regions of high pressure at the front of the cylinder and regions low pressure behind the cylinder. This behaviour is expected because P_{xx} should be highest at the point of impact with the cylinder. The main difference between the two results is in the magnitude of the two quantities. Another comparison is provided in Figure 5.4 for the normal stress along the y-axis, P_{yy}. The two results show that the pressure term P_{yy} is greater in similar regions of the gas flow. While the two results show slightly similar behaviours, the ten-moment equations are failing to accurately capture the properties of the flow. Figure 5.5 compares the shear stress for the two solvers. The ten-moment equations show a visually similar trend to the benchmark produced in the FMFC solver. In both cases, the magnitude of the shear



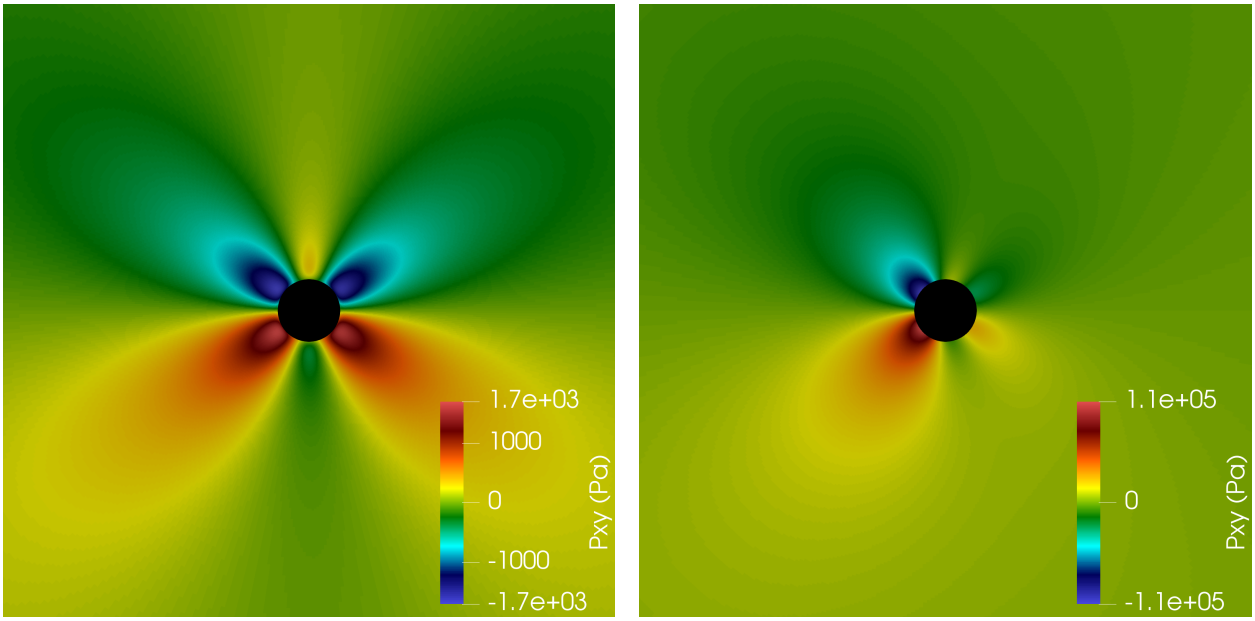
(a) benchmark result from the FMFC solver (b) ten-moment equations solved with a DG scheme

Figure 5.3: Pressure tensor P_{xx}: FMFC solver and the ten-moment equations.



(a) benchmark result from the FMFC solver (b) ten-moment equations solved with a DG scheme

Figure 5.4: Pressure tensor P_{yy}: FMFC solver and the ten-moment equations.



(a) benchmark result from the FMFC solver (b) ten-moment equations solved with a DG scheme

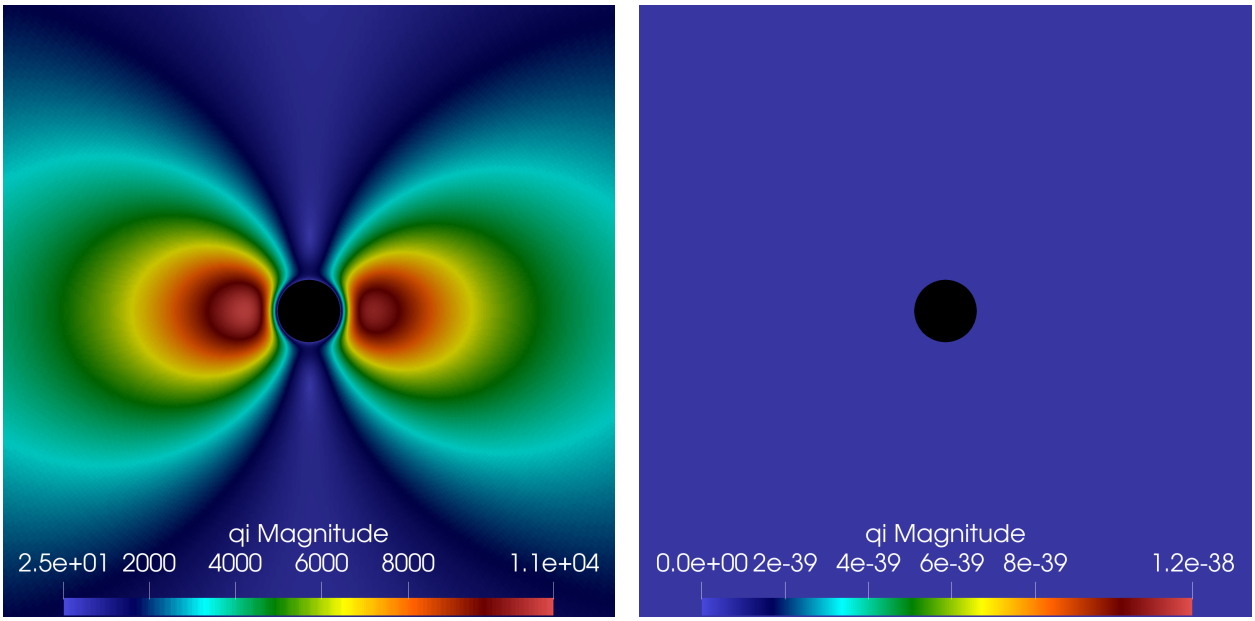
Figure 5.5: Pressure tensor P_{xy}: FMFC solver and the ten-moment equations.

stress component increases in the same regions of the flow. However, the ten-moment equations fail to properly capture the properties of the flow.

5.1.4 Comparing the Heat-Flux

Figure 5.6 highlights the magnitude of the heat-flux from the FMFC solver and the ten-moment equations. The ten-moment equations do not consider the heat-flux of the gas flow and so it is clear that the output for this quantity is zero throughout the entire field. This simply demonstrates that the FMFC solver is capable of computing higher-order moments, such as heat-flux, which can make its applicability as a validation tool for newer models very desirable.

The ten-moment equations are known to be inaccurate in the free-molecular regime. The purpose of this section is not to highlight the inaccuracies of the ten-moment equations but rather to demonstrate how the FMFC solver is a tool that can be used to confirm the accuracy of new models in the free-molecular regime.



(a) benchmark result from the FMFC solver

(b) ten-moment equations solved with a DG scheme

Figure 5.6: Magnitude of the heat-flux: FMFC solver and the ten-moment equations.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

The work presented in this thesis successfully demonstrates the implementation of two numerical solvers for the study of non-equilibrium gas flows on GPU-based platforms using SYCL. Numerical results highlight the computational efficiency and accuracy of the two solvers when simulating gas flows in non-equilibrium regimes.

The FMFC solver incorporates a novel ray-tracing technique, geometric relations, and accurate recursion relations that work together to compute the solution for gas flows near convex shapes in the free-molecular regime. Analytical solutions for the drag and lift produced for free-molecular gas flows near a possibly rotating circular cylinder have also been derived and used to confirm the accuracy of FMFC. The solver was also extended to target GPUs with SYCL to further accelerate the computational runtime by at least an order of magnitude. FMFC can be used to compute any observable property of the gas flow at any location, such as the density, bulk velocity, and heat-flux. This capability enables academic studies for high-speed flows, counter-gradient heat-flux, and other interesting phenomena in the free-molecular regime.

The existing CPU-based implementation of the DGH scheme has been extended into an efficient GPU code using SYCL. The GPU-based implementation was also extended to target multiple GPUs on multiple compute nodes across large-scale supercomputers. A strong scaling study for the GPU version of the code demonstrates near perfect parallel scalability and massive computational speedups for large problem sizes. Third-order accuracy is recovered which enables the code to be used to solve models for the study of non-equilibrium gas flows with a high-degree of numerical accuracy.

One of the objectives in this research project is to use the FMFC solver as a tool that can produce a wide range of accurate benchmarks to validate models for gas flows in the free-molecular regime. The FMFC solver was used to generate benchmarks for gas flows in the free-molecular regime and compared against solutions for the ten-moment equations. The ten-moment equations are already known to be inaccurate in the free-molecular regime. However, this study served as a demonstration for the effectiveness of the FMFC solver as a validation tool that can confirm the accuracy of new models in the free-molecular regime.

6.2 Future Work

Currently, the FMFC solver is only capable of computing the solution for free-molecular gas flows near circular cylinders because the geometric relations that have been developed are restricted to these shapes. However, the recursion relations in the solver work for any convex shape. Future work should extend the capability of the FMFC solver to work with any convex geometry because it would allow for a broad range of benchmarks to be produced and academic studies to be performed.

The GPU implementation of DGH has been developed with rudimentary optimizations. Future work should include an in-depth profiling of the code and targeted optimizations. This can include both general optimizations in the GPU kernels and hardware-specific optimizations for different computing platforms. Currently, the GPU-based version of DGH does not have adaptive mesh refinement (AMR), which could be used to resolve solutions with a much higher level of accuracy and efficiency.

Bibliography

[1] F. D. Witherden and A. Jameson. "Future Directions of Computational Fluid Dynamics". In: *American Institute of Aeronautics and Astronautics* 16.1 (2024), pp. 1–16.

[2] R. Vuduc and J. Choi. "A Brief History and Introduction to GPGPU". In: *Modern Accelerator Technologies for Geographic Information Science*. Springer, 2013, pp. 9–23. DOI: 10.1007/978-1-4614-8745-6_2.

[3] NVIDIA. *CUDA C++ Programming Guide Release 12.5*. May 20, 2024. 2024.

[4] J. Reinders, B. Ashbaugh, J. Brodman, M. Kinsner, J. Pennycook, and X. Tian. "Introduction". In: *Data Parallel C++*. Berkeley, CA: Apress, 2021. URL: https://doi.org/10.1007/978-1-4842-5574-2_1.

[5] H. Struchtrup. *Macroscopic Transport Equations for Rarefied Gas Flows: Approximation Methods in Kinetic Theory*. Dordrecht: Springer-Verlag Berlin and Heidelberg, 2006.

[6] C. W. Jackson, D. Appelhans, J. M. Derlaga, and P. G. Buning. "GPU Implementation of the OVERFLOW CFD Code". In: *NASA Langley Research Center* (2021).

[7] Narval. "Narval - Alliance Doc". In: *Alliance Documentation* (n.d.). URL: https://docs.alliancecan.ca/wiki/Narval/en.

[8] Intel Corporation. "Argonne National Laboratory: Boosting Performance and Power Efficiency of Applications". In: *Intel Case Study* (2020). URL: https://www.intel.com/content/dam/www/central-libraries/us/en/documents/gov-argonne-national-laboratory-case-study-1020.pdf.

[9] G. A. Bird. *Molecular Gas Dynamics and the Direct Simulation of Gas Flows*. Oxford: Clarendon Press, 1994.

[10] P. Charrier, B. Dubroca, L. Mieussens, and R. Turpault. "Discrete-Velocity Models for Numerical Simulations in Transitional Regime for Rarefied Flows and Radiative Transfer". In: *Transport in Transition Regimes*. Ed. by N. B. Abdallah et al. Vol. 135. The IMA Volumes in Mathematics and its Applications. New York, NY: Springer, 2004.

[11] W. Kaufmann. "Extended Hydrodynamics using the Discontinuous-Galerkin Hancock Method". PhD thesis. 2021.

[12] Y. Suzuki. "Discontinuous Galerkin Methods for Extended Hydrodynamics". PhD thesis. University of Michigan, 2008.

[13] J. G. McDonald. “Approximate Maximum-Entropy Moment Closures for Gas Dynamics”. In: *30th International Symposium on Rarefied Gas Dynamics*. Victoria, Canada, 2016.

[14] H. T. Huynh. “An Upwind Moment Scheme for Conservation Laws”. In: *Computational Fluid Dynamics 2004*. 2006, pp. 761–766.

[15] H. Grad. “On the kinetic theory of rarefied gases”. In: *Communications on Pure and Applied Mathematics* 2 (1949), pp. 331–407.

[16] P. L. Bhatnagar, E. P. Gross, and M. Krook. “A Model for Collision Processes in Gases. I. Small Amplitude Processes in Charged and Neutral One-Component Systems”. In: *Physical Review* 94.3 (1954), pp. 511–525.

[17] C. D. Levermore. “Moment closure hierarchies for kinetic theories”. In: *Journal of Statistical Physics* 83 (1996), pp. 1021–1065.

[18] I. Müller and T. Ruggeri. *Rational Extended Thermodynamics*. New York: Springer-Verlag, 1998.

[19] P. L. Tallec and J. P. Perlat. *Numerical analysis of Levermore’s moment system*. Technical Report 3124. INRIA Rocquencourt, 1997.

[20] M. R. A. Abdelmalik and E. H. van Brummelen. “Moment closure approximations of the Boltzmann equation based on -divergences”. In: *Journal of Statistical Physics* 164 (2015).

[21] R. McGraw. “Description of aerosol dynamics by the quadrature method of moments”. In: *Aerosol Science and Technology* 27.2 (1997), pp. 255–265.

[22] D. Marchisio and R. Fox. *Computational Models for Polydisperse Particulate and Multiphase Systems*. Cambridge: Cambridge University Press, 2013.

[23] G. N. Patterson. *Introduction to the kinetic theory of gas flows: By Gordon N. Patterson*. University of Toronto Press, 1971.

[24] OpenMP Architecture Review Board. *OpenMP Application Program Interface Version 3.0*. May 2008. URL: http://www.openmp.org/mp-documents/spec30.pdf.

[25] Mengbo Zhu, Ehsan Roohi, and Amin Ebrahimi. “Computational Study of Rarefied Gas Flow and Heat Transfer in Lid-driven Cylindrical Cavities”. In: *arXiv preprint arXiv:2306.13765* (2023). Physics of Fluids, 35(5), 052012 (2023). URL: https://arxiv.org/abs/2306.13765.

[26] F. Forgues and J. G. McDonald. “Higher-Order Moment Models for Multiphase Flows With Accurate Particle-Stream Crossing”. In: *International Journal of Multiphase Flow* 114 (2019), pp. 28–38.

[27] W. Morin and J. G. McDonald. “A New Moment Model For Radiative-Transport Prediction”. In: *25th Annual Conference of the CFD Society of Canada*. Windsor, Ontario, Canada, 2017.

[28] S. Boccelli, F. Giroux, T. E. Magin, C. P. T. Groth, and J. G. McDonald. “14-Moment Maximum-Entropy Description of Electrons in Crossed Electric and Magnetic Fields”. In: *Physics of Plasmas* 27 (2020).

[29] B. Cockburn, S. Hou, and C. W. Shu. “The Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws. IV: The multidimensional case”. In: *Mathematics of Computation* 54.190 (1990), pp. 545–581.

[30] Message Passing Interface Forum. *MPI: A Message-Passing Interface Standard Version 4.0*. June 2021. URL: https://www.mpi-forum.org/docs/mpi-4.0/mpi40-report.pdf.

[31] V. Venkatakrishnan. “Convergence to Steady State Solutions of the Euler Equations on Unstructured Grids with Limiters”. In: *Journal of Computational Physics* 118.1 (1995), pp. 120–130.

[32] B. Einfeldt. “On Godunov-Type Methods for Gas Dynamics”. In: *SIAM Journal on Numerical Analysis* 25.2 (1988), pp. 294–318.

Appendix A

Integrating the Drag term using Bessel Functions

This section of the appendix highlights important integrations that are used to compute the drag produced by a rotating cylinder. The drag per unit area is

$$D = -\sigma_{\mathrm{N}} \cos\psi + \sigma_{\mathrm{T}} \sin\psi, \tag{A.1}$$

where σ_{N} and σ_{T} are the normal force and tangential force on the cylinder surface, respectively. The normal stress σ_{N} is the second-order x'-directional velocity moment

$$\sigma_{\mathrm{N}} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} v_x'^2 \mathcal{F}_p \, \mathrm{d}v_x' \, \mathrm{d}v_y' \, \mathrm{d}v_z, \tag{A.2}$$

which equates to

$$\sigma_{\mathrm{N}} = \frac{m\alpha n_{\mathrm{W}}}{4\beta_{\mathrm{W}}} + \frac{n_{\mathrm{FS}} m(\alpha-2)(\cos\psi(Se^{-S^2\cos^2\psi}) + (S^2\cos^2\psi + \frac{1}{2})\sqrt{\pi}(\mathrm{erf}(S\cos\psi)-1))}{2\sqrt{\pi}\beta_{\mathrm{FS}}}. \tag{A.3}$$

The tangential stress σ_{T} is the second-order velocity moment of an x'-direction and y'-direction velocity moment, such that

$$\sigma_{\mathrm{T}} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} v_x' v_y' \mathcal{F}_p \, \mathrm{d}v_x' \, \mathrm{d}v_y' \, \mathrm{d}v_z, \tag{A.4}$$

which equates to

$$\begin{aligned}\sigma_{\mathrm{T}} =& \frac{\sin\psi(e^{-S^2\cos^2\psi}\sqrt{\pi} + \pi S\cos\psi(\mathrm{erf}(S\cos\psi)-1))\alpha u_x m n_{\mathrm{FS}}}{2\sqrt{\beta_{\mathrm{FS}}}\pi} \\ &+ \frac{R\alpha u_x m n \omega(e^{-S^2\cos^2\psi} + \sqrt{\pi}S\cos\psi(\mathrm{erf}(S\cos\psi)-1))}{2\sqrt{\beta_{\mathrm{FS}}}\sqrt{\pi}}.\end{aligned} \tag{A.5}$$

The coefficient of drag C_D is

$$C_D = \frac{1}{\rho v^2} \int_0^{2\pi} D \, \mathrm{d}\psi \tag{A.6}$$

and so the integral that needs to be evaluated is

$$\int_0^{2\pi} D \, \mathrm{d}\psi, \tag{A.7}$$

which is equal to

$$\int_0^{2\pi} -\sigma_\mathrm{N} \cos\psi + \sigma_\mathrm{T} \sin\psi \, \mathrm{d}\psi. \tag{A.8}$$

The integral for the drag term expands to be,

$$\begin{aligned} D = \int_0^{2\pi} \Bigg[& \frac{nmu_x}{(B_\mathrm{FS}\pi)^{\frac{1}{2}}} \left(1 - \frac{\alpha}{2}\right) \frac{\cos^2\psi}{e^{u_x^2 B_\mathrm{FS}\cos^2\psi}} - \left(\frac{nm\alpha}{4(B_\mathrm{FS}B_\mathrm{W})^{\frac{1}{2}}}\right) \frac{\cos\psi}{e^{u_x^2 B_\mathrm{FS}\cos^2\psi}} \\ & + nmu_x^2 \left(1 - \frac{\alpha}{2}\right) \cos^3\psi \cdot \mathrm{erf}(B_\mathrm{FS}^{\frac{1}{2}} u_x \cos\psi) \\ & - nmu_x^2 \left(1 - \frac{\alpha}{2}\right) \cos^3\psi - \left(\frac{nm\alpha u_x}{4}\right) \left(\frac{\pi}{B_\mathrm{W}}\right)^{\frac{1}{2}} \cos^2\psi \cdot \mathrm{erf}(B_\mathrm{FS}^{\frac{1}{2}} u_x \cos\psi) \\ & + \left(\frac{nm\alpha u_x}{4}\right) \left(\frac{\pi}{B_\mathrm{W}}\right)^{\frac{1}{2}} \cos^2\psi + \frac{nm}{2B_\mathrm{FS}} \left(1 - \frac{\alpha}{2}\right) \cos\psi \cdot \mathrm{erf}(B_\mathrm{FS}^{\frac{1}{2}} u_x \cos\psi) \\ & - \frac{nm}{2B_\mathrm{FS}} \left(1 - \frac{\alpha}{2}\right) \cos\psi + \left(\frac{nm\alpha u_x}{2(B_\mathrm{FS}\pi)^{\frac{1}{2}}}\right) \frac{\sin^2\psi}{e^{u_x^2 B_\mathrm{FS}\cos^2\psi}} \\ & + \left(\frac{nm\alpha u_x^2}{2}\right) \sin^2\psi \cdot \mathrm{erf}(B_\mathrm{FS}^{\frac{1}{2}} u_x \cos\psi) \cos\psi - \left(\frac{nm\alpha u_x^2}{2}\right) \sin^2\psi \cos\psi \\ & - \left(\frac{nm\alpha R_w}{2(B_\mathrm{FS}\pi)^{\frac{1}{2}}}\right) \frac{\sin\psi}{e^{u_x^2 B_\mathrm{FS}\cos^2\psi}} - \left(\frac{nm\alpha u_x R_w}{2}\right) \sin\psi \cdot \mathrm{erf}(B_\mathrm{FS}^{\frac{1}{2}} u_x \cos\psi) \cos\psi \\ & + \left(\frac{nm\alpha u_x R_w}{2}\right) \sin\psi \cos\psi \Bigg] \mathrm{d}\psi. \end{aligned} \tag{A.9}$$

There are 14 individual terms in the integral that need to be evaluated. Table A.1 highlights these terms and excludes constant variables. The following sections in this chapter of the appendix show how results are obtained for each of the integral terms in Table A.1. Many of the integrals that are solved in the following sections make use of modified Bessel functions of the first kind. The zeroth modified Bessel function of the first kind is

$$I_0(x) = \frac{1}{\pi} \int_0^{\pi} e^{x\cos(u)} \, \mathrm{d}u \tag{A.10}$$

No.	Term	No.	Term
1	$\frac{\cos^2\psi}{e^{u_x^2 B_{\mathrm{FS}}\cos^2\psi}}$	8	$\cos\psi$
2	$\frac{\cos\psi}{e^{u_x^2 B_{\mathrm{FS}}\cos^2\psi}}$	9	$\frac{\sin^2\psi}{e^{u_x^2 B_{\mathrm{FS}}\cos^2\psi}}$
3	$\cos^3\psi\cdot\mathrm{erf}(B_{\mathrm{FS}}^{\frac{1}{2}}u_x\cos\psi)$	10	$\sin^2\psi\cdot\mathrm{erf}(B_{\mathrm{FS}}^{\frac{1}{2}}u_x\cos\psi)\cos\psi$
4	$\cos^3\psi$	11	$\sin^2\psi\cos\psi$
5	$\cos^2\psi\cdot\mathrm{erf}(B_{\mathrm{FS}}^{\frac{1}{2}}u_x\cos\psi)$	12	$\frac{\sin\psi}{e^{u_x^2 B_{\mathrm{FS}}\cos^2\psi}}$
6	$\cos^2\psi$	13	$\sin\psi\cdot\mathrm{erf}(B_{\mathrm{FS}}^{\frac{1}{2}}u_x\cos\psi)\cos\psi$
7	$\cos\psi\cdot\mathrm{erf}(B_{\mathrm{FS}}^{\frac{1}{2}}u_x\cos\psi)$	14	$\sin\psi\cos\psi$

Table A.1: Terms in the integral for drag

and the n^{th} modified Bessel function of the first kind is

$$I_n(x)=\frac{1}{\pi}\int_0^{\pi}\cos(nu)\,e^{x\cos(u)}\,\mathrm{d}u. \tag{A.11}$$

The speed ratio S is also used to simplify each of the terms, where

$$S=u_x\sqrt{\beta_{\mathrm{FS}}}. \tag{A.12}$$

A.1 Integral 1

The first integral to be evaluated is

$$\int_0^{2\pi}\frac{\cos^2\psi}{e^{u_x^2 B_{\mathrm{FS}}\cos^2\psi}}\,\mathrm{d}\psi. \tag{A.13}$$

Using the trigonometric identity

$$\cos^2\psi=\frac{1+\cos(2\psi)}{2}, \tag{A.14}$$

the integral equation becomes

$$\frac{1}{2}e^{-\frac{S^2}{2}}\int_0^{2\pi}(1+\cos(2\psi))\,e^{-\frac{S^2}{2}\cos(2\psi)}\,\mathrm{d}\psi. \tag{A.15}$$

Substituting $\kappa=2\psi$ into the integral equation results in

$$e^{-\frac{S^2}{2}}\left(\int_0^{\pi}e^{-\frac{S^2}{2}\cos\kappa}d\kappa+\int_0^{\pi}(\cos\kappa)e^{-\frac{S^2}{2}\cos\kappa}d\kappa\right). \tag{A.16}$$

By using the modified Bessel functions of the first kind,

$$\int_0^{2\pi}\frac{\cos^2\psi}{e^{u_x^2 B_{\mathrm{FS}}\cos^2\psi}}\,\mathrm{d}\psi=e^{-\frac{S^2}{2}}\left(\pi I_0\left(\frac{S^2}{2}\right)+\pi I_1\left(\frac{S^2}{2}\right)\right). \tag{A.17}$$

A.2 Integral 2

The second integral to be evaluated is

$$\int_0^{2\pi} \frac{\cos\psi}{e^{u_x^2 B_{\mathrm{FS}} \cos^2\psi}} \,\mathrm{d}\psi \tag{A.18}$$

and always converges to zero such that

$$\int_0^{2\pi} \frac{\cos\psi}{e^{u_x^2 B_{\mathrm{FS}} \cos^2\psi}} \,\mathrm{d}\psi = 0. \tag{A.19}$$

A.3 Integral 3

The third integral to be evaluated is

$$\int_0^{2\pi} \cos^3\psi \cdot \mathrm{erf}(B_{\mathrm{FS}}^{\frac{1}{2}} u_x \cos\psi) \,\mathrm{d}\psi. \tag{A.20}$$

By applying an integration by parts, the integral term becomes

$$\frac{2S}{\pi^{\frac{1}{2}}} \int_0^{2\pi} \left(\sin^2\psi - \frac{\sin^4\psi}{3}\right) e^{-S^2\cos^2\psi} \,\mathrm{d}\psi. \tag{A.21}$$

Using the trigonometric identity

$$\sin^2\psi = \frac{1-\cos(2\psi)}{2}, \tag{A.22}$$

the integral equation becomes

$$\frac{S}{12\pi^{\frac{1}{2}}} e^{-\frac{S^2}{2}} \int_0^{2\pi} (9 - 8\cos(2\psi) - \cos(4\psi))\, e^{-\frac{S^2}{2}\cos(2\psi)} \,\mathrm{d}\psi. \tag{A.23}$$

Substituting $\kappa = 2\psi$ into the integral equation results in

$$\frac{S}{6\pi^{\frac{1}{2}}} e^{-\frac{S^2}{2}} \int_0^{\pi} (9 - 8\cos\kappa - \cos(2\kappa))\, e^{-\frac{S^2}{2}\cos\kappa} \,\mathrm{d}\kappa. \tag{A.24}$$

By using the modified Bessel functions of the first kind,

$$\int_0^{2\pi} \cos^3\psi \cdot \mathrm{erf}(B_{\mathrm{FS}}^{\frac{1}{2}} u_x \cos\psi) \,\mathrm{d}\psi = \frac{S}{6\pi^{\frac{1}{2}}} e^{-\frac{S^2}{2}} \left(9\pi I_0\left(\frac{S^2}{2}\right) - 8\pi I_1\left(\frac{S^2}{2}\right) - \pi I_2\left(\frac{S^2}{2}\right)\right). \tag{A.25}$$

A.4 Integral 4

The fourth integral to be evaluated is

$$\int_0^{2\pi} \cos^3\psi \, \mathrm{d}\psi \tag{A.26}$$

and always converges to zero such that

$$\int_0^{2\pi} \cos^3\psi \, \mathrm{d}\psi = 0. \tag{A.27}$$

A.5 Integral 5

The fifth integral to be evaluated is

$$\int_0^{2\pi} \cos^2\psi \cdot \mathrm{erf}(B_{\mathrm{FS}}^{\frac{1}{2}} u_x \cos\psi) \, \mathrm{d}\psi \tag{A.28}$$

and always converges to zero such that

$$\int_0^{2\pi} \cos^2\psi \cdot \mathrm{erf}(B_{\mathrm{FS}}^{\frac{1}{2}} u_x \cos\psi) \, \mathrm{d}\psi = 0 \tag{A.29}$$

A.6 Integral 6

The sixth integral to be evaluated is

$$\int_0^{2\pi} \cos^2\psi \, \mathrm{d}\psi \tag{A.30}$$

and always converges to π such that

$$\int_0^{2\pi} \cos^2\psi \, \mathrm{d}\psi = \pi \tag{A.31}$$

A.7 Integral 7

The seventh integral to be evaluated is

$$\int_0^{2\pi} \cos\psi \cdot \mathrm{erf}(B_{\mathrm{FS}}^{\frac{1}{2}} u_x \cos\psi) \mathrm{d}\psi. \tag{A.32}$$

By applying an integration by parts, the integral term becomes

$$\frac{2S}{\pi^{\frac{1}{2}}} \int_0^{2\pi} \sin^2(\psi) \left(e^{-S^2 \cos^2(\psi)} \right) \mathrm{d}\psi. \tag{A.33}$$

Using the trigonometric identities from Equation (A.14) and Equation (A.22), the integral equation becomes

$$\frac{S}{\pi^{\frac{1}{2}}} e^{-\frac{S^2}{2}} \int_0^{2\pi} (1 - \cos(2\psi))\, e^{-\frac{S^2}{2}\cos(2\psi)}\, \mathrm{d}\psi. \tag{A.34}$$

Substituting $\kappa = 2\psi$ into the integral equation results in

$$\frac{2S}{\pi^{\frac{1}{2}}} e^{-\frac{S^2}{2}} \left(\int_0^{\pi} e^{-\frac{S^2}{2}\cos\kappa} d\kappa - \int_0^{\pi} (\cos\kappa) e^{-\frac{S^2}{2}\cos\kappa} d\kappa \right). \tag{A.35}$$

By using the modified Bessel functions of the first kind,

$$\int_0^{2\pi} \cos\psi \cdot \mathrm{erf}(B_{\mathrm{FS}}^{\frac{1}{2}} u_x \cos\psi) \mathrm{d}\psi = 2S\pi^{\frac{1}{2}} e^{-\frac{S^2}{2}} \left(I_0\left(\frac{S^2}{2}\right) - I_1\left(\frac{S^2}{2}\right) \right). \tag{A.36}$$

A.8 Integral 8

The eighth integral to be evaluated is

$$\int_0^{2\pi} \cos\psi\, \mathrm{d}\psi \tag{A.37}$$

and always converges to zero such that

$$\int_0^{2\pi} \cos\psi\, \mathrm{d}\psi = 0 \tag{A.38}$$

A.9 Integral 9

The ninth integral to be evaluated is

$$\int_0^{2\pi} \frac{\sin^2\psi}{e^{u_x^2 B_{\mathrm{FS}} \cos^2\psi}}\, \mathrm{d}\psi. \tag{A.39}$$

Using the trigonometric identity from Equation (A.22), the integral equation becomes

$$\frac{1}{2} e^{-\frac{S^2}{2}} \int_0^{2\pi} (1 - \cos(2\psi))\, e^{-\frac{S^2}{2}\cos(2\psi)}\, \mathrm{d}\psi. \tag{A.40}$$

Substituting $\kappa = 2\psi$ into the integral equation results in

$$e^{-\frac{S^2}{2}} \left(\int_0^{\pi} e^{-\frac{S^2}{2}\cos\kappa} d\kappa - \int_0^{\pi} (\cos\kappa) e^{-\frac{S^2}{2}\cos\kappa} d\kappa \right). \tag{A.41}$$

By using the modified Bessel functions of the first kind,

$$\int_0^{2\pi} \frac{\sin^2\psi}{e^{u_x^2 B_{\mathrm{FS}} \cos^2\psi}}\, \mathrm{d}\psi = e^{-\frac{S^2}{2}} \left(\pi I_0\left(\frac{S^2}{2}\right) - \pi I_1\left(\frac{S^2}{2}\right) \right). \tag{A.42}$$

A.10 Integral 10

The tenth integral to be evaluated is

$$\int_0^{2\pi} \sin^2\psi \cdot \operatorname{erf}(B_{\mathrm{FS}}^{\frac{1}{2}} u_x \cos\psi)\cos\psi \,\mathrm{d}\psi. \tag{A.43}$$

By applying an integration by parts, the integral term becomes

$$\frac{2S}{3\pi^{\frac{1}{2}}} \int_0^{2\pi} \left(\sin^4\psi\right) e^{-S^2\cos^2\psi} \,\mathrm{d}\psi. \tag{A.44}$$

Using the trigonometric identities from Equation (A.14) and Equation (A.22), the integral equation becomes

$$\frac{S}{6\pi^{\frac{1}{2}}} e^{-\frac{S^2}{2}} \int_0^{2\pi} \left(1 - 2\cos(2\psi) + \cos^2(2\psi)\right) e^{-\frac{S^2}{2}\cos(2\psi)} \,\mathrm{d}\psi. \tag{A.45}$$

Substituting $\kappa = 2\psi$ into the integral equation results in

$$\frac{S}{6\pi^{\frac{1}{2}}} e^{-\frac{S^2}{2}} \left(\int_0^{2\pi} e^{-\frac{S^2}{2}\cos\kappa} d\kappa - 2\int_0^{2\pi} \cos\kappa \cdot e^{-\frac{S^2}{2}\cos\kappa} d\kappa + \int_0^{2\pi} \cos^2\kappa \cdot e^{-\frac{S^2}{2}\cos\kappa} d\kappa \right). \tag{A.46}$$

By using the modified Bessel functions of the first kind,

$$\begin{aligned} &\int_0^{2\pi} \sin^2\psi \cdot \operatorname{erf}(B_{\mathrm{FS}}^{\frac{1}{2}} u_x \cos\psi)\cos\psi \,\mathrm{d}\psi = \\ &\frac{S\pi^{\frac{1}{2}}}{6} e^{-\frac{S^2}{2}} \left(I_0\left(\frac{S^2}{2}\right) - 2I_1\left(\frac{S^2}{2}\right) + e^{\frac{S^2}{2}} \left(I_0\left(\frac{S^2}{2}\right) + I_1\left(\frac{S^2}{2}\right)\right)\right). \end{aligned} \tag{A.47}$$

A.11 Integral 11

The eleventh integral to be evaluated is

$$\int_0^{2\pi} \sin^2\psi \cos\psi \,\mathrm{d}\psi \tag{A.48}$$

and always converges to zero such that

$$\int_0^{2\pi} \sin^2\psi \cos\psi \,\mathrm{d}\psi = 0. \tag{A.49}$$

A.12 Integral 12

The twelfth integral to be evaluated is

$$\int_0^{2\pi} \frac{\sin\psi}{e^{u_x^2 B_{\mathrm{FS}}\cos^2\psi}} \,\mathrm{d}\psi \tag{A.50}$$

and always converges to zero such that

$$\int_0^{2\pi} \frac{\sin\psi}{e^{u_x^2 B_{\mathrm{FS}} \cos^2\psi}} \, \mathrm{d}\psi = 0. \tag{A.51}$$

A.13 Integral 13

The thirteenth integral to be evaluated is

$$\int_0^{2\pi} \sin\psi \cdot \mathrm{erf}(B_{\mathrm{FS}}^{\frac{1}{2}} u_x \cos\psi) \cos\psi \, \mathrm{d}\psi \tag{A.52}$$

and always converges to zero such that

$$\int_0^{2\pi} \sin\psi \cdot \mathrm{erf}(B_{\mathrm{FS}}^{\frac{1}{2}} u_x \cos\psi) \cos\psi \, \mathrm{d}\psi = 0. \tag{A.53}$$

A.14 Integral 14

The fourteenth integral to be evaluated is

$$\int_0^{2\pi} \sin\psi \cos\psi \, \mathrm{d}\psi \tag{A.54}$$

and always converges to zero such that

$$\int_0^{2\pi} \sin\psi \cos\psi \, \mathrm{d}\psi = 0. \tag{A.55}$$

A.15 Final Integral of the Drag Term

Using the computed integral terms in each of the previous sections of this Appendix chapter, the integral of the drag term is

$$\begin{aligned}\int_0^{2\pi} D\,\mathrm{d}\psi = \Bigg[& n_{\mathrm{FS}} m u_x \sqrt{\frac{\pi}{\beta_{\mathrm{FS}}}} \left(1-\frac{\alpha}{2}\right)\left(I_0\left(-\frac{S^2}{2}\right)+I_1\left(-\frac{S^2}{2}\right)\right) \\ & + n_{\mathrm{FS}} m u_x^2 \left(1-\frac{\alpha}{2}\right)\frac{S\sqrt{\pi}e^{-\frac{S^2}{2}}}{6}\left(9I_0\left(-\frac{S^2}{2}\right)-8I_1\left(-\frac{S^2}{2}\right)-I_2\left(-\frac{S^2}{2}\right)\right) \\ & + \left(\frac{nm\alpha u_x}{4}\right)\sqrt{\frac{\pi}{\beta_{\mathrm{FS}}}}\pi \\ & + \frac{n_{\mathrm{FS}} m}{2\beta_{\mathrm{FS}}}\left(1-\frac{\alpha}{2}\right)2S\sqrt{\pi}e^{-\frac{S^2}{2}}\left(I_0\left(-\frac{S^2}{2}\right)-I_1\left(-\frac{S^2}{2}\right)\right) \\ & + \left(\frac{nm\alpha u_x}{2}\right)\sqrt{\frac{\pi}{\beta_{\mathrm{FS}}}}e^{-\frac{S^2}{2}}\left(I_0\left(-\frac{S^2}{2}\right)-I_1\left(-\frac{S^2}{2}\right)\right) \\ & + \left(\frac{nm\alpha u_x^2}{2}\frac{S}{6}\sqrt{\pi}e^{-\frac{S^2}{2}}\right)\left(3I_0\left(-\frac{S^2}{2}\right)-4I_1\left(-\frac{S^2}{2}\right)+I_2\left(-\frac{S^2}{2}\right)\right)\Bigg].\end{aligned} \tag{A.56}$$