A Hopf-Cole transformation based asymptotic method for kinetic equations with a BGK collision operator in the large scale hyperbolic limit

Nicholas Anthony Payne

Iowa State University

Follow this and additional works at: https://lib.dr.iastate.edu/etd

Part of the Applied Mathematics Commons

Recommended Citation

This Thesis is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Graduate Theses and Dissertations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.
A Hopf-Cole transformation based asymptotic method for kinetic equations with
a BGK collision operator in the large scale hyperbolic limit

by

Nicholas Anthony Payne

A thesis submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE

Major: Applied Mathematics

Program of Study Committee:
Songting Luo, Major Professor
   James Rossmanith
   Justin Peters

Iowa State University
Ames, Iowa
2016
Copyright © Nicholas Anthony Payne, 2016. All rights reserved.
DEDICATION

I would like to dedicate this thesis to my wife Kassia; without her support I would not have been able to complete this work. I would also like to thank my friends and peers for their encouragement and guidance during the writing of this thesis.
TABLE OF CONTENTS

LIST OF FIGURES ................................................................. v
LIST OF TABLES ................................................................. vi
ACKNOWLEDGEMENTS ............................................................ ix
ABSTRACT ................................................................. x

CHAPTER 1. OVERVIEW ......................................................... 1
  1.1 Introduction .............................................................. 1
    1.1.1 Boltzmann .......................................................... 1
    1.1.2 BGK ................................................................. 2
    1.1.3 Numerical methods ............................................... 3
  1.2 Problem Statement .................................................... 5
    1.2.1 Formulation of $\varphi^0$ ....................................... 7
    1.2.2 Formulation of $\varphi^1$ ....................................... 8
    1.2.3 Method overview ................................................ 11

CHAPTER 2. NUMERICAL METHOD ............................................ 13
  2.1 Introduction .............................................................. 13
    2.1.1 Finite difference methods ..................................... 13
    2.1.2 Runge-Kutta techniques ....................................... 13
    2.1.3 Gaussian quadrature ............................................ 14
    2.1.4 General Gaussian quadrature .................................. 15
  2.2 Numerical Schemes .................................................... 16
    2.2.1 Property preservation .......................................... 17
    2.2.2 Numerical Hamiltonian ......................................... 18
2.2.3 WENO finite difference method ........................................ 20
2.2.4 Total Variation Diminishing Runge-Kutta technique ............. 21
2.2.5 Algorithm .............................................................. 21
2.2.6 Recovery of the effective Hamiltonian ................................. 21

2.3 The Method ............................................................... 23
2.3.1 Recovering $\varphi^0$ .................................................. 23
2.3.2 Recovering $\varphi^1$ .................................................. 25

CHAPTER 3. RESULTS ......................................................... 27
3.1 Introduction ............................................................... 27
3.2 1D ................................................................. 28
  3.2.1 Computation of the effective Hamiltonian ....................... 28
  3.2.2 Computation of $\varphi^0$ .......................................... 28
  3.2.3 Computation of $\varphi^1$ .......................................... 32
  3.2.4 Approximating $f^\epsilon$ .......................................... 33
3.3 2D ................................................................. 39
  3.3.1 Computation of the effective Hamiltonian ....................... 39
  3.3.2 Computation of $\varphi^0$ .......................................... 39
  3.3.3 Computation of $\varphi^1$ .......................................... 42
  3.3.4 Approximating $f^\epsilon$ .......................................... 47

CHAPTER 4. SUMMARY AND DISCUSSION ................................. 53
4.1 Summary ............................................................... 53
4.2 Future Work .......................................................... 54

BIBLIOGRAPHY ............................................................. 55
LIST OF FIGURES

2.1 Simple stencil of the WENO scheme ........................................... 20

3.1 Constant Maxwellian, $M(v) = \frac{1}{2}$ with (Left) computed effective Hamiltonian in red diamonds with exact Hamiltonian in green circles. (Right) Solution, $\varphi^0$ ................................................................. 31

3.2 Non-constant Maxwellian, $M(v) = \frac{e^{-v^2}}{\sqrt{\pi}\text{erf}(1)}$ with (Left) computed effective Hamiltonian in red circles. (Right) Solution, $\varphi^0$ ................................................................. 32

3.3 Constant Maxwellian, $M(v) = \frac{1}{2}$ with (Top Left) $\varphi^0$, (Top Right) $\lambda$, (Middle Left) Reference Density, (Middle Right) Density with proposed method, (Bottom Left) $f^\epsilon$ from IMEX-RK and (Bottom Right) $M(v)e^{-\frac{\varphi^0 + \epsilon \varphi^1}{\epsilon}}$ ................................................................. 35

3.4 Constant Maxwellian, $M(v) = \frac{1}{2}$ with (Top Left) $\varphi^0$, (Top Right) $\lambda$, (Middle Left) Reference Density, (Middle Right) Density with proposed method, (Bottom Left) $f^\epsilon$ from IMEX-RK and (Bottom Right) $M(v)e^{-\frac{\varphi^0 + \epsilon \varphi^1}{\epsilon}}$ ................................................................. 36

3.5 Non-constant Maxwellian, $M(v) = \frac{e^{-v^2}}{\sqrt{\pi}\text{erf}(1)}$ with (Top Left) $\varphi^0$, (Top Right) $\lambda$, (Middle Left) Reference Density, (Middle Right) Density with proposed method, (Bottom Left) $f^\epsilon$ from IMEX-RK and (Bottom Right) $M(v)e^{-\frac{\varphi^0 + \epsilon \varphi^1}{\epsilon}}$ ................................................................. 37

3.6 Non-constant Maxwellian, $M(v) = \frac{e^{-v^2}}{\sqrt{\pi}\text{erf}(1)}$ with (Top Left) $\varphi^0$, (Top Right) $\lambda$, (Middle Left) Reference Density, (Middle Right) Density with proposed method, (Bottom Left) $f^\epsilon$ from IMEX-RK and (Bottom Right) $M(v)e^{-\frac{\varphi^0 + \epsilon \varphi^1}{\epsilon}}$ ................................................................. 38
3.7 Constant Maxwellian, \( M(u,v) = \frac{1}{4} \) with (Left) \( \lambda \) (Right) \( \varphi^0 \) and (Bottom) \( H \). .......................................................... 44

3.8 Constant Maxwellian, \( M(u,v) = \frac{1}{4} \) with (Left) \( \lambda \) (Right) \( \varphi^0 \) and (Bottom) \( H \). .......................................................... 45

3.9 Non-constant Maxwellian, \( M(u,v) = \frac{e^{-u^2-v^2}}{\pi \text{erf}(1)^2} \) with (Left) \( \lambda \) (Right) \( \varphi^0 \) and (Bottom) \( H \). .......................................................... 46

3.10 Non-constant Maxwellian, \( M(u,v) = \frac{e^{-u^2-v^2}}{\pi \text{erf}(1)^2} \) with (Left) \( \lambda \) (Right) \( \varphi^0 \) and (Bottom) \( H \). .......................................................... 47

3.11 Constant Maxwellian, \( M(u,v) = \frac{1}{4} \) with (Top Left) \( \varphi^0 \), (Top Right) \( \lambda \), (Bottom Left) Reference Density, (Bottom Right) Density with proposed method .......................................................... 49

3.12 Constant Maxwellian, \( M(u,v) = \frac{1}{4} \) with (Top Left) \( \varphi^0 \), (Top Right) \( \lambda \), (Bottom Left) Reference Density, (Bottom Right) Density with proposed method .......................................................... 50

3.13 Non-constant Maxwellian, \( M(u,v) = \frac{e^{-u^2-v^2}}{\pi \text{erf}(1)^2} \) with (Top Left) \( \varphi^0 \), (Top Right) \( \lambda \), (Bottom Left) Reference Density, (Bottom Right) Density with proposed method .......................................................... 51

3.14 Non-constant Maxwellian, \( M(u,v) = \frac{e^{-u^2-v^2}}{\pi \text{erf}(1)^2} \) with (Top Left) \( \varphi^0 \), (Top Right) \( \lambda \), (Bottom Left) Reference Density, (Bottom Right) Density with proposed method .......................................................... 52
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>$L^\infty$ error of Hamiltonian based on number of Gaussian nodes and quadrature rule</td>
<td>28</td>
</tr>
<tr>
<td>3.2</td>
<td>Error and convergence rates when $M(v) = \frac{1}{2}$ using Gauss-Legendre quadrature with $T = 16$</td>
<td>29</td>
</tr>
<tr>
<td>3.3</td>
<td>Error and convergence rates when $\omega(v) = M(v) = \frac{1}{2}$ using the Golub and Welsch algorithm with $T = 16$</td>
<td>29</td>
</tr>
<tr>
<td>3.4</td>
<td>Error and convergence rates when $M(v) = \frac{1}{\sqrt{\pi}erf(1)} e^{-v^2}$ using Gauss-Legendre with $T = 16$</td>
<td>29</td>
</tr>
<tr>
<td>3.5</td>
<td>Error and convergence rates when $\omega(v) = M(v) = \frac{1}{\sqrt{\pi}erf(1)} e^{-v^2}$ using the Golub and Welsch algorithm with $T = 16$</td>
<td>30</td>
</tr>
<tr>
<td>3.6</td>
<td>Computational time, in seconds, for the 1D code to compute a solution to $\varphi^0$</td>
<td>30</td>
</tr>
<tr>
<td>3.7</td>
<td>Error and convergence rates of $\lambda$ with $M(v) = \frac{1}{2}$</td>
<td>33</td>
</tr>
<tr>
<td>3.8</td>
<td>Error and convergence rates of $\lambda$ with $M(v) = \frac{e^{-v^2}}{\sqrt{\pi}erf(1)}$</td>
<td>33</td>
</tr>
<tr>
<td>3.9</td>
<td>Error and convergence rates with $\varphi^0_0 = (1e-4)(1+\sin(2x\pi))$ and $M(v) = \frac{1}{2}$</td>
<td>34</td>
</tr>
<tr>
<td>3.10</td>
<td>Error and convergence rates with $\varphi^0_0 = (1e-4)(1+\sin(2x\pi))$ and $M(v) = \frac{e^{-v^2}}{\sqrt{\pi}erf(1)}$</td>
<td>34</td>
</tr>
<tr>
<td>3.11</td>
<td>$L^\infty$ error of Hamiltonian based on number of Gaussian nodes and quadrature rule</td>
<td>39</td>
</tr>
<tr>
<td>3.12</td>
<td>Error and convergence rates when $M(u, v) = \frac{1}{4}$ using Gauss-Legendre with $T = 12$</td>
<td>40</td>
</tr>
<tr>
<td>3.13</td>
<td>Error and convergence rates when $\omega(u, v) = M(u, v) = \frac{1}{4}$ using the Golub and Welsch algorithm with $T = 12$</td>
<td>40</td>
</tr>
</tbody>
</table>
3.14 Error and convergence rates when $M(u, v) = \frac{e^{-u^2 - v^2}}{\pi \text{erf}(1)^2}$ using Gauss-Legendre with $T = 12$ .............. 40

3.15 Error and convergence rates when $\omega(u, v) = M(u, v) = \frac{e^{-u^2 - v^2}}{\pi \text{erf}(1)^2}$ using the Golub and Welsch algorithm with $T = 12$ .............. 41

3.16 Computational time, in seconds, for varying $T$’s in 2D .............. 42

3.17 Error and convergence rates of $\lambda$ with $M(u, v) = \frac{1}{4}$ .............. 42

3.18 Error and convergence rates of $\lambda$ with $M(u, v) = \frac{e^{-u^2 - v^2}}{\pi \text{erf}(1)^2}$ .............. 43

3.19 Error and convergence rates with $\varphi_0 = 0.0125(2 + \sin(2x\pi) + \sin(2y\pi))$
and $M(u, v) = \frac{e^{-u^2 - v^2}}{\pi \text{erf}(1)^2}$ .............. 48
ACKNOWLEDGEMENTS

I would like to take this opportunity to express my thanks to those who helped me with various aspects of conducting research and the writing of this thesis. First and foremost, Dr. Songting Luo for his guidance, patience, and support throughout this research and the writing process. His insights and guidance were absolutely necessary for the completion of my graduate education. I would also like to thank my committee members for their efforts and contributions to this work: Dr. James Rossmanith and Dr. Justin Peters. I would additionally like to thank Joshua Carlson for inspiring me to begin a graduate career.
We propose an efficient asymptotic approach to approximating the density function of kinetic equations with Bhatnagar-Gross-Krook (BGK) relaxation operators. These types of equations are relevant in the study of particle dynamics in thermodynamic systems; gas dynamics for example. We consider a simplified BGK operator in this thesis for the purposes of explaining the proposed method which has the potential to deal with more general BGK operators. We transform the density function as the Hopf-Cole transformation and expand the phase function in a power series about the Knudsen number. This is similar to moment closure methods for similar equations but we make no assumptions about the moments of the kinetic equation. The leading order term of the power series is the viscosity solution of a particular Hamilton-Jacobi equation. The Hamilton-Jacobi equation involves an implicitly-defined Hamiltonian which is embedded in an integral with respect to the velocity variables. The Hamiltonian is the root of a nonlinear integral equation so Gaussian quadrature and Newton’s method are used to recover it. The first order term in the power series is related to a transport equation. Both the Hamilton-Jacobi equation and the transport equation are formulated in the physical space with necessary components defined as integrals with respect to the velocity variables. The integrals can be evaluated efficiently using Gaussian quadrature. We use well-established techniques for time-dependent Hamilton-Jacobi equations to solve these two equations and recover an estimate of the phase function. We then transform the phase function back to a faithful estimate of the original density function.
CHAPTER 1. OVERVIEW

1.1 Introduction

A specific area of interest in kinetic theory is to estimate the density of particles at specific locations and points in time in a given system. The density of particles, denoted as \( f(t, x, v) \) is called the probability density function and gives the probability that the particles at position \( x \) are moving with velocity \( v \) at time \( t \). The motivation behind this thesis is the development of an efficient numerical method to solve kinetic equations with a BGK relaxation operator when considered in the large scale hyperbolic limit. That is, we are interested in studying the hyperbolic scaling \((\frac{t}{\epsilon}, \frac{x}{\epsilon})\) as \( \epsilon \to 0 \), where \( \epsilon \) is the Knudsen number. The Knudsen number is defined to be the ratio of the molecular mean free path to some characteristic length. It is related to the molecular collision frequency in a thermodynamic system.

1.1.1 Boltzmann

Kinetic equations with BGK operators refers to a family of equations that are simplified forms of the Boltzmann equation. The Boltzmann equation is used to model particle collisions and particle density in a thermodynamic system. The equation is important because it describes the evolution of flow of particles within a fluid. It has applications in scales ranging from gas dynamics to galactic dynamics. Kinetic equations with BGK relaxation operators are specifically relevant in gas dynamics and thermodynamics. The Boltzmann equation involves a complicated quadratically nonlinear collision operator on the right hand side. Perthame [1] offers a quick derivation of the Boltzmann equation which reads

\[
\partial_t f(t, x, v) + v \cdot \nabla f(t, x, v) = Q(f), \quad (t, x, v) \in \mathbb{R}_+ \times \mathbb{R}^d \times \mathbb{V},
\]  

(1.1)
where $Q(f)$ is the collision operator given by:

$$
Q(f) = \int \int_{\mathbb{V}S^{d-1}} (f'f'_* - ff_*)B(|\omega \cdot (v - v_*)|, |v - v_*|)dv_*d\omega,
$$

with the notation $f'_* = f(t, x, v'_*)$, where $v'_*$ denotes the post-collision velocities. The function $B$ is a collision kernel. When the Boltzmann equation is considered in 3-space, the problem is large and difficult to solve numerically. For each spatial coordinate $x = [x, y, z]^T$, there are three momentum components, $p = [p_x, p_y, p_z]^T$, giving rise to a 6-dimensional problem.

1.1.2 BGK

In 1954 Bhatnagar, Gross, and Krook proposed the BGK operator [2] as a means of relaxing the collision operator in the Boltzmann equation due to its complexity. The BGK relaxation operator forgoes securing an exact solution to the Boltzmann equation in general in order to more easily model some of the qualitative behavior of a thermodynamic system that is near equilibrium. The retention of the collision invariants, that is the conservation of mass, momentum, and energy, is one characteristic which makes the BGK operator so useful [3, 4]. Kinetic equations with a BGK operator are still difficult to solve numerically due to the high-dimensional structure of the particle density and the presence of multiple scales.

In this thesis, we consider a simplified BGK operator, aiming to propose and explain an effective asymptotic approach to approximating the density function. The proposed method has the potential to deal with more general BGK operators.

We begin with a kinetic equation with a simplified BGK operator:

$$
\partial_t f(t, x, v) + v \cdot \nabla_x f(t, x, v) = M(v)\rho(t, x) - f(t, x, v),
$$

(1.2)

where $f(t, x, v)$ denotes a probability density function. The function $\rho(t, x)$ is called the macroscopic density and is given by:

$$
\rho(t, x) = \int f(t, x, v)dv, \quad (t, x) \in \mathbb{R}_+ \times \mathbb{R}^d.
$$

$M(v)$ is the Maxwellian and is symmetric with the following moment properties:

$$
\int v M(v)dv = 0, \quad \int v^2 M(v)dv = \theta^2, \quad \theta \in \mathbb{R}.
$$
We consider the hyperbolic scaling of equation (1.2) by replacing $(t, x)$ with $(\frac{t}{\epsilon}, \frac{x}{\epsilon}) = (\tau, \xi)$. We then have that $\frac{d}{dt} = \epsilon \frac{d}{d\tau}$ and similarly that $\frac{d}{dx} = \epsilon \frac{d}{d\xi}$. Then equation (1.2) reads in the $(\tau, \xi)$ space:

$$\epsilon \partial_\tau f + \epsilon \mathbf{v} \cdot \nabla_\xi f = M(\mathbf{v}) \rho - f \Rightarrow \partial_\tau f + \mathbf{v} \cdot \nabla_\xi f = \frac{1}{\epsilon} (M(\mathbf{v}) \rho - f)$$

We finally consider the large scale hyperbolic limit by considering the limit as $\epsilon \to 0$. Now for ease of reading simply relabel the equation as $(\tau, \xi) = (t, x)$ and use an $\epsilon$ superscript to denote the fact that we are looking at the large scale hyperbolic limit.

Thus the kinetic equation with a simplified BGK operator in the large scale hyperbolic limit considered in this thesis is given as:

$$\partial_t f^\epsilon(t, x, \mathbf{v}) + \mathbf{v} \cdot \nabla_x f^\epsilon(t, x, \mathbf{v}) = \frac{1}{\epsilon} (M(\mathbf{v}) \rho^\epsilon(t, x) - f^\epsilon(t, x, \mathbf{v})),$$  \hspace{1cm} (1.3)

More complex and realistic BGK operators include Maxwellians which have dependence $(t, x, v)$ [1, 2, 5, 6].

### 1.1.3 Numerical methods

There are existing numerical methods and tools available which can be utilized to solve these types of kinetic equations such as simple finite difference or finite volume schemes, moment methods, semi-Lagrangian schemes, discrete-velocity models, spectral methods, asymptotic preserving methods, and implicit-explicit Runge-Kutta type methods [1, 3, 4, 5, 7, 8, 9, 10, 11]. These references and the references therein offer detailed discussions of these mentioned methods and techniques. These existing methods have their strengths and weaknesses; the primary weaknesses being their complexity and dependence on $\epsilon$. We propose a method to combat these weaknesses and balance accuracy and efficiency in a desirable way. Standard methods for solving kinetic equations with a BGK operator often involve discretizations directly in phase space. A simple scheme consists of covering the physical domain and velocity domain with a uniform mesh and then using a simple or high order finite difference scheme to approximate $f^\epsilon$. One specific example is a one-step upwinding/downwinding scheme in which the physical space and velocity space are covered with a uniform mesh and then upwinding (or downwinding) is
used in order to approximate the gradient. The notation used below is the same as is discussed in Section 2.2.

\[
F_{kj}^{n+1} = F_{kj}^n - \Delta t \hat{F}_{kj}^n + \frac{\Delta t}{\epsilon} \left( M_k \rho_j^n - f_{kj}^n \right)
\]

where \( \hat{F}_{kj}^n = \begin{cases} 
\frac{v_k}{\Delta x} (f_{j+1}^n - f_j^n) & \text{if } v_k \leq 0 \\
\frac{v_k}{\Delta x} (f_j^n - f_{j-1}^n) & \text{if } v_k > 0 
\end{cases} \) (1.4)

In cases with small values of \( \epsilon \) the problem becomes very stiff and so classical methods such as (1.4) require impractically small time steps to attain any reliable solutions. Therefore, more involved tools (such as the ones listed previously) must be used to efficiently compute a reliable solution. A specific, well-established, method for solving kinetic equations with a BGK relaxation operator is the aforementioned implicit-explicit Runge-Kutta (IMEX-RK) method [5]. The collision term is treated implicitly to combat the stiffness. The first three moments of the kinetic equation are required at each intermediate stage of the Runge-Kutta procedure and these can be computed efficiently using Gaussian quadrature. The implicit treatment of the collision term allows for larger time steps to be taken, which is advantageous for efficiency. In principle, any reliable scheme used to solve a kinetic equation with a BGK operator directly in phase space given in the form of equation (1.3) must be constructed using the techniques discussed in [5]. This method uses a high order implicit-explicit Runge-Kutta scheme for the temporal discretization which has a non-trivial implementation (for a more robust discussion see [7, 8, 9]). In [5] it is proposed in conservative form with a simpler, but nonconservative scheme, also outlined. This is the method that we implement to calculate a reference solution to the kinetic equation with BGK operator considered in this thesis to verify the accuracy of the proposed method. The strength of this method is that it can, in principle, be used to solve problems with infinite stiffness, and it can also be implemented to ensure conservation of the collision invariants of the kinetic equation. Another strength is that even in the nonconservative form, the scheme demonstrates that the loss of conservation of the collision invariants is very small [5]. The primary weaknesses of the IMEX-RK method that we attempt to specifically address are the dependence on \( \epsilon \) as well as the dependence on the velocity discretization. We remark that the dependence on \( \epsilon \) is considered a weakness in the context of considering the
limit as \( \epsilon \to 0 \), and not a weakness in general.

In the proposed method we transform the density function of the kinetic equation with a BGK relaxation operator using a Hopf-Cole transformation and expand the phase function in a power series about \( \epsilon \). We attempt to recover an accurate approximation to the phase function by solving a Hamilton-Jacobi equation and a transport equation, both of which are formulated only in the physical space with necessary components defined as integrals with respect to the velocity variables. We implement a high order finite difference method to achieve high order accuracy in space, a Runge-Kutta scheme to achieve high order accuracy in time, and Gaussian quadrature to decrease the complexity in the velocity variables. Standard methods for solving kinetic equations with a BGK relaxation operator are on the order of \( O(N^{2d}) \) complexity if \( O(N) \) points are used in the space and velocity discretizations and \( d \) is the dimension. We construct a method of complexity \( O(T^d N^d) \), where \( T \) is the number of Gaussian abscissas/weights used in the velocity discretization, which is significantly more efficient than \( O(N^{2d}) \) if \( T \ll N \). Our approximation is based on solutions to equations which are formulated only in the physical space which allows for cheaper computations than solving equations in phase space. We remark that the proposed method is not dependent on \( \epsilon \) or the velocity discretization.

The rest of this thesis is organized as follows:

In Section 2, we discuss the numerical tools and schemes used in the proposed method in this work. In Section 3, we present numerical results to verify the effectiveness of the proposed method. In Section 4, we summarize the results as well as discuss areas of future research.

### 1.2 Problem Statement

As \( \epsilon \to 0 \) on the right hand side of equation (1.3), it is clear that the velocity distribution relaxes towards the Maxwellian distribution rapidly and so we are motivated to introduce the following Hopf-Cole transformation [12]:

\[
f(t, x, v)^\epsilon = M(v)e^{-\varphi^\epsilon(t, x, v) \epsilon},
\]

(1.5)

where \( \varphi^\epsilon \) is the phase function and has some useful uniform estimates with respect to \( \epsilon > 0 \) [12].
Proposition 1.2.1. Let $\mathcal{V} \subset \mathbb{R}^d$ be a bounded subset. Assume $M(v) \in L^1(\mathcal{V})$, and is nonnegative and symmetric, $\varphi^\epsilon(t = 0, x, v) = \varphi_0(x) \geq 0$, and $\varphi_0 \in W^{1,\infty}(\mathbb{R}^d)$, then

\begin{align*}
0 \leq \varphi^\epsilon(t, \cdot, \cdot) & \leq \|\varphi_0\|_{\infty}, \\
\|\nabla_x \varphi^\epsilon(t, \cdot, \cdot)\|_{\infty} & \leq \|\nabla_x \varphi_0\|_{\infty}, \\
\|\nabla_v \varphi^\epsilon(t, \cdot, \cdot)\|_{\infty} & \leq t\|\nabla_x \varphi_0\|_{\infty}, \\
\|\partial_t \varphi^\epsilon(t, \cdot, \cdot)\|_{\infty} & \leq V_{\text{max}}\|\nabla_x \varphi_0\|_{\infty}
\end{align*}

where $V_{\text{max}}$ is the maximum velocity.

We consider the power series expansion of $\varphi^\epsilon(t, x, v)$ as

\[ \varphi^\epsilon(t, x, v) = \varphi_0^\epsilon(t, x, v) + \epsilon \varphi_1^\epsilon(t, x, v) + \epsilon^2 \varphi_2^\epsilon(t, x, v) + \cdots + \epsilon^m \varphi_m^\epsilon(t, x, v) + \cdots \quad (1.6) \]

The first two terms in the expansion (1.6) can be found by solving a Hamilton-Jacobi equation and a transport equation respectively. Both equations are formulated in the physical space with necessary components defined as integrals with respect to the velocity variables (the formulations follow in the next sections). In the proposed method we will ultimately truncate after the first order term as a means of recovering an approximation of $\varphi^\epsilon(t, x, v)$ and therefore a faithful approximation to $f^\epsilon(t, x, v)$ by:

\begin{align*}
\varphi^\epsilon(t, x, v) & \approx \varphi_0^\epsilon(t, x, v) + \epsilon \varphi_1^\epsilon(t, x, v) \\
f^\epsilon(t, x, v) & \approx M(v)e^{-\frac{\varphi_0^\epsilon(t, x, v) + \epsilon \varphi_1^\epsilon(t, x, v)}{\epsilon}}.
\end{align*}
1.2.1 Formulation of $\varphi^0$

Substituting (1.5) into equation (1.3) yields the following:

$$\partial_t \left(M(v)e^{-\frac{\varphi^\epsilon}{\epsilon}}\right) + v \cdot \nabla_x \left(M(v)e^{-\frac{\varphi^\epsilon}{\epsilon}}\right) = M(v)\rho - M(v)e^{-\frac{\varphi^\epsilon}{\epsilon}}$$

$$-\frac{1}{\epsilon} \varphi^\epsilon_t M(v)e^{-\frac{\varphi^\epsilon}{\epsilon}} + v \cdot \left(-\frac{1}{\epsilon} M(v)\nabla_x \varphi^\epsilon\right) = \frac{1}{\epsilon} \left(\int_M f^\epsilon dv - M(v)e^{-\frac{\varphi^\epsilon}{\epsilon}}\right)$$

$$-\varphi_t^\epsilon e^{-\frac{\varphi^\epsilon}{\epsilon}} - v \cdot \nabla_x \varphi^\epsilon e^{-\frac{\varphi^\epsilon}{\epsilon}} = \int_M M(v)e^{-\frac{\varphi^\epsilon}{\epsilon}}dv - e^{-\frac{\varphi^\epsilon}{\epsilon}}$$

$$\varphi_t + v \cdot \nabla_x \varphi^\epsilon = 1 - e^{-\frac{\varphi^\epsilon}{\epsilon}} \int_M M(v)e^{-\frac{\varphi^\epsilon}{\epsilon}}dv$$

We then have

$$\varphi_t + v \cdot \nabla_x \varphi^\epsilon = \int_M M(v') \left(1 - e^{-\frac{\varphi^\epsilon - \varphi^\epsilon'}{\epsilon}}\right) d\nu', \quad (1.7)$$

and finally

$$1 - \varphi_t^\epsilon - v \cdot \nabla_x \varphi^\epsilon = \int_M M(v') e^{-\frac{\varphi^\epsilon - \varphi^\epsilon'}{\epsilon}} d\nu'. \quad (1.8)$$

From equation (1.8) we notice the following:

$$\frac{M(v)}{1 - \varphi_t^\epsilon - v \cdot \nabla_x \varphi^\epsilon} = \frac{M(v)}{\int_M M(v') e^{-\frac{\varphi^\epsilon - \varphi^\epsilon'}{\epsilon}} d\nu'} = \frac{M(v) e^{-\frac{\varphi^\epsilon}{\epsilon}}}{\int_M M(v') e^{-\frac{\varphi^\epsilon}{\epsilon}} d\nu'} = \frac{f^\epsilon}{\rho^\epsilon}$$

By integrating both sides with respect to $v$ we get that:

$$\int_M \frac{M(v)}{1 - \varphi_t^\epsilon - v \cdot \nabla_x \varphi^\epsilon} dv = \int_M f^\epsilon \rho^\epsilon = 1 \quad (1.9)$$

Now passing the limit as $\epsilon \to 0$ we attain:

$$\int_M \frac{M(v)}{1 - \partial_t \varphi^0(t, x, v) - v \cdot \nabla_x \varphi^0(t, x, v)} dv = 1, \quad (t, x) \in \mathbb{R}_+ \times \mathbb{R}^d$$

**Theorem 1.2.2.** (Kinetic Eikonal Equation) Let $\mathbb{V} \subset \mathbb{R}^d$ be a bounded symmetric velocity space, and $M(v) \in L^1(\mathbb{V})$ be nonnegative and symmetric. Then $\varphi^\epsilon$ converges (locally) uniformly
towards $\varphi^0$, where $\varphi^0$ does not depend on $v$. Moreover $\varphi^0$ is the viscosity solution of the following Hamilton-Jacobi equation, referred to as the kinetic eikonal equation

$$
\int_{\mathbb{V}} \frac{M(v)}{1 - \partial_t \varphi^0(t, x) - v \cdot \nabla_x \varphi^0(t, x)} dv = 1, \quad (t, x) \in \mathbb{R}_+ \times \mathbb{R}^d.
$$

(1.10)

The denominator of the integrand is positive for all $v \in \mathbb{V}$. (Proof available in [12])

Equation (1.10) can be expressed as $G(\varphi^0_t, \nabla_x \varphi^0) = 0$ [12]. We observe that $G$ is increasing with respect to $\varphi^0_t$ and so it is rewritten as

$$
\varphi^0_t + H(\nabla_x \varphi^0) = 0,
$$

(1.11)

with the effective Hamiltonian, $H(p)$, being defined implicitly in the integral equation

$$
\int_{\mathbb{V}} \frac{M(v)}{1 + H(p) - v \cdot p} dv = 1, \quad p = \nabla_x \varphi^0(t, x).
$$

(1.12)

1.2.2 Formulation of $\varphi^1$

Here we will notate $\varphi^{\epsilon} = \varphi(t, x, v')$. Equation (1.6) can be rewritten as

$$
\varphi^{\epsilon}(t, x, v) = \varphi^0(t, x) + \epsilon\varphi^1,\epsilon(t, x, v)
$$

(1.13)

where $\varphi^1,\epsilon(t, x, v) = \varphi^1(t, x, v) + \epsilon\varphi^2(t, x, v) + \cdots + \epsilon^{m-1}\varphi^m(t, x, v) + \cdots$.

Substituting (1.13) into equation (1.8) yields:

$$
1 - (\varphi^0_t + \epsilon\varphi^1,\epsilon) - v \cdot \nabla_x (\varphi^0 + \epsilon\varphi^1,\epsilon) = \int_{\mathbb{V}} M(v') e^{(\varphi^0 + \epsilon\varphi^1,\epsilon - \varphi^0 - \epsilon\varphi^1,\epsilon')} dv'
$$

$$
(1 - \varphi^0_t - v \cdot \nabla_x \varphi^0) - \epsilon(\varphi^1,\epsilon + v \cdot \nabla_x \varphi^1,\epsilon) = \int_{\mathbb{V}} M(v') e^{(\varphi^1,\epsilon - \varphi^1,\epsilon')} dv',
$$

and finally,

$$
e^{-\varphi^1,\epsilon} [(1 - \varphi^0_t - v \cdot \nabla_x \varphi^0) - \epsilon(\varphi^1,\epsilon + v \cdot \nabla_x \varphi^1,\epsilon)] = \int_{\mathbb{V}} M(v') e^{-\varphi^1,\epsilon'} dv'.
$$

(1.14)
Also, consider the difference between equations (1.10) and (1.9)

\[
0 = \int V M(v) \left( 1 - \varphi^0_k - v \cdot \nabla x \varphi^0 \right) d\nu - \int V M(v) \left( 1 - \varphi^0_k - v \cdot \nabla x \varphi^0 \right) d\nu
\]

\[
= \int V M(v) \left[ \frac{(1 - \varphi^0_k - v \cdot \nabla x \varphi^0) - (1 - \varphi^0_k - \epsilon \varphi^1_k - v \cdot \nabla x (\varphi^0 + \epsilon \varphi^1_k))}{(1 - \varphi^0_k - v \cdot \nabla x \varphi^0)(1 - \varphi^0_k - v \cdot \nabla x \varphi^0)} \right] d\nu,
\]

which yields:

\[
\int V M(v) \left[ \frac{\varphi^1_k + v \cdot \nabla x \varphi^1_k}{(1 - \varphi^0_k - v \cdot \nabla x \varphi^0)^2} \right] d\nu = 0. \tag{1.15}
\]

Letting \( \epsilon \to 0 \) in equations (1.14) and (1.15) results in

\[
\begin{cases} 
 e^{-\varphi^1}(1 - \varphi^0_k - v \cdot \nabla x \varphi^0) = \int V M(v') e^{-\varphi^1_k} d\nu' \\
\int V M(v) \left[ \frac{\varphi^1_k + v \cdot \nabla x \varphi^1_k}{(1 - \varphi^0_k - v \cdot \nabla x \varphi^0)^2} \right] d\nu = 0. \tag{1.16}
\end{cases}
\]

Let \( \lambda(t, x) \equiv \log \int V M(v') e^{-\varphi^1(t, x, v')} d\nu' \) and let \( G(t, x, v) \equiv 1 - \partial_t \varphi^0(t, x) - v \cdot \nabla x \varphi^0(t, x) \).

Then from the first equation of (1.16) and the definition of \( \lambda(t, x) \) we see that

\[
\varphi^1(t, x, v) = \log G(t, x, v) - \lambda(t, x). \tag{1.17}
\]

From the second equation of (1.16) and equation (1.17) we have the following:

\[
0 = \int V M(v) \left[ \frac{\varphi^1_k + v \cdot \nabla x \varphi^1_k}{(1 - \varphi^0_k - v \cdot \nabla x \varphi^0)^2} \right] d\nu
\]

\[
= \int V M(v) \left[ \frac{\varphi^1_k + v \cdot \nabla x \varphi^1_k}{G^2} \right] d\nu
\]

\[
= \int V M(v) \left[ \frac{\frac{1}{2} G \lambda - \lambda t + v \cdot \nabla x \log G}{G^2} \right] d\nu
\]

\[
= \int V M(v) \frac{G_t + v \cdot \nabla x G}{G^3} d\nu - \int V M(v) \frac{\lambda t + v \cdot \nabla x \lambda}{G^2}, d\nu
\]

which gives rise to the following transport equation:

\[
a(t, x) \partial_t \lambda(t, x) + B(t, x) \cdot \nabla x \lambda(t, x) = r(t, x), \tag{1.18}
\]
where
\[ a(t, x) = \int M(v) \frac{d v}{G^2} \]
\[ B(t, x) = \int \frac{v \cdot M(v)}{G^2} d v \]
\[ r(t, x) = \int M(v) \frac{G_t + v \cdot \nabla_x G}{G^3} d v. \] (1.19)

It is desirable to eliminate the time derivatives of \( G \) and replace them with spatial derivatives.

Recalling the definition of \( G \) we then have the following:
\[ G_t = -\varphi^0_{tt} - v \cdot \nabla_x \varphi^0_t \]
\[ \nabla_x G = -\nabla_x \varphi^0_t - v \cdot D^2_x \varphi^0 \]
\[ G_t + v \cdot \nabla_x G = -\varphi^0_{tt} - 2v \cdot \nabla_x \varphi^0_t - v \cdot D^2_x \varphi^0 \cdot v, \]

where \( D^2_x \) denotes the Hessian. Then by differentiating equation (1.11) with respect to \( t \) and \( x \) we have
\[ \varphi^0_{tt} + \nabla H(\nabla_x \varphi^0) \cdot \nabla_x \varphi^0_t = 0 \Rightarrow \varphi^0_{tt} = -\nabla H(\nabla_x \varphi^0) \cdot \nabla_x \varphi^0_t, \]

and also
\[ \nabla_x \varphi^0_t + \nabla H(\nabla_x \varphi^0) \cdot D^2_x \varphi^0 = 0 \Rightarrow \nabla_x \varphi^0_t = -\nabla H(\nabla_x \varphi^0) \cdot D^2_x \varphi^0. \]

Using the above equalities we have the following:
\[ G_t + v \cdot \nabla_x G = -\varphi^0_{tt} - 2v \cdot \nabla_x \varphi^0_t - v \cdot D^2_x \varphi^0 \cdot v \]
\[ = \nabla H(\nabla_x \varphi^0) \cdot \nabla_x \varphi^0_t + 2v \cdot \nabla H(\nabla_x \varphi^0) \cdot D^2_x \varphi^0 - v \cdot D^2_x \varphi^0 \cdot v \]
\[ = -\nabla H(\nabla_x \varphi^0) \cdot \nabla H(\nabla_x \varphi^0) \cdot D^2_x \varphi^0 + 2v \nabla H(\nabla_x \varphi^0) \cdot D^2_x \varphi^0 - v \cdot D^2_x \varphi^0 \cdot v \]
\[ = (-\nabla H(\nabla_x \varphi^0) + v) \cdot D^2_x \varphi^0 \cdot (\nabla H(\nabla_x \varphi^0) - v). \]

Therefore numerator of \( r(t, x) \) in equation (1.19) can be rewritten as
\[ r(t, x) = \int M(v) \left( -\frac{(\nabla H(\nabla_x \varphi^0) - v) \cdot D^2_x \varphi^0 \cdot (\nabla H(\nabla_x \varphi^0) - v)}{G^3(t, x, v)} \right) d v. \]
Consider the derivative of equation (1.12) with respect to $\mathbf{p}$:

$$
- \int_{\mathcal{V}} M(\mathbf{v}) \frac{\nabla H(\mathbf{p}) \cdot \nabla \mathbf{p} - \mathbf{v} \cdot \nabla \mathbf{p}}{(1 + H(\mathbf{p}) - \mathbf{v} \cdot \mathbf{p})^2} d\mathbf{v} = 0
$$

$$
\nabla H(\mathbf{p}) \cdot \nabla \mathbf{p} \int_{\mathcal{V}} M(\mathbf{v}) \left( \frac{1}{1 + H(\mathbf{p}) - \mathbf{v} \cdot \mathbf{p}} \right) d\mathbf{v} - \nabla \mathbf{p} \int_{\mathcal{V}} \frac{\mathbf{v} M(\mathbf{v})}{1 + H(\mathbf{p}) - \mathbf{v} \cdot \mathbf{p}}^2 d\mathbf{v} = 0,
$$

where $\mathbf{p} = \nabla_x \varphi^0$. This allows us to note that

$$
\nabla H(\nabla_x \varphi^0) = \frac{\int_{\mathcal{V}} \frac{\mathbf{v} M(\mathbf{v})}{G^2} d\mathbf{v}}{\int_{\mathcal{V}} \frac{M(\mathbf{v})}{G^2} d\mathbf{v}} = \frac{B(t, \mathbf{x})}{a(t, \mathbf{x})},
$$

(1.20)

Now using equation (1.20) equations (1.18) and (1.19) can be rewritten as

$$
\partial_t \lambda(t, \mathbf{x}) + \frac{B(t, \mathbf{x})}{a(t, \mathbf{x})} \cdot \nabla_x \lambda(t, \mathbf{x}) = \frac{r(t, \mathbf{x})}{a(t, \mathbf{x})}
$$

$$
a(t, \mathbf{x}) = \int_{\mathcal{V}} \frac{M(\mathbf{v})}{G^2(t, \mathbf{x}, \mathbf{v})} d\mathbf{v}, \quad B(t, \mathbf{x}) = \int_{\mathcal{V}} \frac{\mathbf{v} M(\mathbf{v})}{G^2(t, \mathbf{x}, \mathbf{v})} d\mathbf{v}
$$

$$
r(t, \mathbf{x}) = \int_{\mathcal{V}} M(\mathbf{v}) \left( - \frac{B(t, \mathbf{x})}{a(t, \mathbf{x})} - \mathbf{v} \right) \cdot \frac{D_x^2 \varphi^0}{G^3(t, \mathbf{x}, \mathbf{v})} d\mathbf{v}.
$$

(1.21)

1.2.3 Method overview

Here we present an overview of the method. $\varphi^0(t, \mathbf{x})$ is obtained through solving equation (1.11). Well-established techniques for time-dependent Hamilton-Jacobi equations are used in this computation. The Hamiltonian is defined implicitly in equation (1.12) and is recovered by using Gaussian quadrature to discretize the integral and Newton’s method to solve the resulting non-linear equation for which $H$, the Hamiltonian, is a root. $\varphi^1(t, \mathbf{x}, \mathbf{v})$ is obtained by solving the transport equation (1.21). The integrals defining the coefficients in the transport equation are discretized using Gaussian quadrature. We rely on similar computational techniques as in the computation of $\varphi^0(t, \mathbf{x})$ to obtain $\lambda(t, \mathbf{x})$, the solution of the transport equation. Once $\varphi^0(t, \mathbf{x})$ and $\lambda(t, \mathbf{x})$ are obtained, then $G(t, \mathbf{x}, \mathbf{v})$ is computed at the desired velocity points and $\varphi^1(t, \mathbf{x}, \mathbf{v}) = \log(G(t, \mathbf{x}, \mathbf{v}))-\lambda(t, \mathbf{x})$. Finally we approximate the original density as:

$$
f^\epsilon(t, \mathbf{x}, \mathbf{v}) \approx M(\mathbf{v}) e^{-\frac{\varphi^0(t, \mathbf{x})}{\epsilon} - \varphi^1(t, \mathbf{x}, \mathbf{v})}.
Note that we have

\[
\begin{align*}
f^\varepsilon(t, x, v) &= M(v)e^{-\phi^\varepsilon(t, x, v)} \\
&= M(v)e^{-\psi^0(t, x) + \psi^1(t, x, v) + \sum_{k=2}^{\infty} \varepsilon^{k-1} \varphi_k(t, x, v)} \\
&= M(v)e^{-\psi^0(t, x)} - \psi^1(t, x, v) e^{\sum_{k=2}^{\infty} \varepsilon^{k-1} \varphi_k}.
\end{align*}
\]

Then by Taylor Series we have

\[
e^{-\sum_{k=2}^{\infty} \varepsilon^{k-1} \varphi_k} = \sum_{m=0}^{\infty} \frac{(-\sum_{k=2}^{\infty} \varepsilon^{k-1} \varphi_k(t, x, v))^m}{m!}.
\]

Thus we have

\[
f^\varepsilon(t, x, v) = M(v)e^{-\psi^0(t, x)} - \psi^1(t, x, v) \sum_{m=0}^{\infty} \frac{(-\sum_{k=2}^{\infty} \varepsilon^{k-1} \varphi_k(t, x, v))^m}{m!},
\]

which implies that

\[
\left| \frac{f^\varepsilon(t, x, v) - M(v)e^{-\psi^0(t, x)} - \psi^1(t, x, v)}{M(v)e^{-\psi^0(t, x)} - \psi^1(t, x, v)} \right| = \sum_{m=1}^{\infty} \frac{(-\sum_{k=2}^{\infty} \varepsilon^{k-1} \varphi_k(t, x, v))^m}{m!} = \mathcal{O}(\varepsilon) [13].
\]

We therefore expect that the method will be \(\mathcal{O}(\varepsilon)\) accurate, which is verified in Section 4.
CHAPTER 2. NUMERICAL METHOD

2.1 Introduction

In this chapter we introduce the general topics and tools utilized in this work. Following is a discussion of the numerical method used to approximate $\varphi^0$ and $\varphi^1$.

2.1.1 Finite difference methods

In this thesis we utilize a finite difference method. Finite difference methods consist of choosing an adequate discretization for both the physical and temporal domains of the partial differential equation (PDE) in question. The derivatives of the PDE are then approximated using difference equations, often derived using Taylor Series. The PDE is then discretized and solved numerically. In order to achieve high orders of accuracy in a method, high order approximations to the derivatives must be used. Error in a finite difference method is defined to be the difference between the approximation (numerical solution) and the exact solution. There are two sources of error in a finite difference method; the first is the loss of precision due to computer arithmetic and the second is the truncation error (the difference between the method’s solution and the exact solution assuming computer arithmetic is exact). Both types of error are important to consider when developing and testing new finite difference methods. The order of accuracy of a finite difference method describes how quickly the numerical solution converges to the exact solution when the step-size is taken to 0 in a limit sense.

2.1.2 Runge-Kutta techniques

Runge-Kutta type methods refer to a class of iterative methods used to achieve high orders of accuracy in the temporal discretization for the approximation of solutions to ordinary diff-
ferential equations. In the world of PDEs, Runge-Kutta methods are used specifically for the
temporal discretization to achieve high orders of accuracy in time. Runge-Kutta methods can
be implicit or explicit. In general, if we consider an initial value problem given by:

\[ \frac{d}{dt}y(t) = f(t, y), y(t_0) = y_0, \]

where \( y(t) \) is some unknown function of \( t \), then a \( \nu \)-stage explicit Runge-Kutta method is
generalized by:

\[ y^{n+1} = y^n + \Delta t \sum_{i=1}^{\nu} b_i k_i, \]

where

\[ k_1 = f(t^n, y^n) \]
\[ k_2 = f(t^n + c_2 \Delta t, y^n + \Delta t(a_{21} k_1)) \]
\[ \vdots \]
\[ k_\nu = f(t^n + c_\nu \Delta t, y^n + \Delta t(a_{\nu 1} k_1 + a_{\nu 2} k_2 + \cdots + a_{\nu \nu-1} k_{\nu-1})), \]

where \( y(n\Delta t) = y^n \) and the coefficients \( a_{ij}, b_i, c_i, 0 < j < i < \nu \) are tabulated for varying \( \nu \)
or can also be derived depending on \( \nu \). \( \Delta t \) is the step size. A more robust introduction and
discussion of Runge-Kutta methods can be found in the literature, for example [14].

2.1.3 Gaussian quadrature

Integration is a familiar mathematical operator which shows up in countless applications.
Numerical integration, or quadrature, is an important area of study for any application and
Gaussian quadrature is a widely used method for evaluating integrals numerically. The theory
of Gaussian quadrature is well-established and is built on top of the theory of orthogonal
polynomials. A set of polynomials, \( P_n \) is considered orthogonal with respect to a weight function \( \omega(x) \) if

\[ \int_\Omega \omega(x) p_i(x) p_j(x) dx = 0 \text{ for } p_k(x) \in P_n, 1 \leq k, i, j \leq n, i \neq j. \]

A quadrature rule approximates a definite integral on some domain, \( \Omega \), by a weighted finite
sum. Using Gaussian quadrature, we can construct a finite sum of \( n \) terms to yield exact
integration results for polynomials of up to order $2n - 1$. Say that we are given a function $f(x)$ which can be factored into a product of some weight function, $\omega(x)$ and a polynomial, $g(x)$ of degree $2n - 1$. Then we can write

$$\int_{\Omega} f(x) dx = \int_{\Omega} \omega(x)g(x) dx = \sum_{i=1}^{n} w_i g(x_i),$$

where $\{x_i\}_{i=1}^{n}$ is the set of roots of a polynomial belonging to $P_n$, the set of orthogonal polynomials with respect to $\omega(x)$, and $w_i$ is the $i^{th}$ weight. Gaussian quadrature is also useful if the function we desire to integrate can be closely approximated by the product of a polynomial and a weight function. For well-known weight functions, such as $\omega(x) = 1$ or $\omega(x) = e^{-x^2}$, weights and abscissas are known and tabulated up to varying orders because these weight functions are related to well known sets of orthogonal polynomials; the Legendre polynomials and the Hermite polynomials respectively. We discuss weight and abscissa generation for non-classical weight functions in Section 2.1.4.

In the multi-D cases, Gaussian quadrature can simply be treated dimension by dimension [15], for example:

$$I = \int_{\Omega_1} \int_{\Omega_2} f(x,y) dxdy \approx \int_{\Omega_1} \sum_{i=1}^{n} w_i f(x_i,y) dy = \sum_{i=1}^{n} w_i \int_{\Omega_1} f(x_i,y) dy.$$

Similarly, the inside integral is then discretized and we arrive at a 2D Gaussian quadrature rule:

$$I = \int_{\Omega_1} \int_{\Omega_2} f(x,y) dxdy \approx \sum_{i=1}^{n} \sum_{j=1}^{n} w_i w_j f(x_i,y_j)$$

For 3D a completely analogous computation holds.

2.1.4 General Gaussian quadrature

In this work the Maxwellian, $M(v)$ appears in some integral equations with respect to the velocity variables. Simple Gaussian (Gauss) rules can be implemented to accurately solve the integral equations, such as Gauss-Legendre quadrature. It is advantageous for efficiency to treat $M(v)$ as the weight function and generate Gaussian weights and abscissas specific to the integral in question. This is simply done by following the Golub and Welsch algorithm
given in [16]. Gautschi proved that all sets of orthogonal polynomials, \( P_n \) satisfy a three-term recurrence relation given by
\[
p_{r+1}(x) = (x - a_{r,r})p_r(x) - b_r p_{r-1}(x)
\]
for \( a_r := \frac{\langle x p_r, p_r \rangle}{\langle p_r, p_r \rangle} \) and \( b_r := \frac{\langle p_r, p_r \rangle}{\langle p_{r-1}, p_{r-1} \rangle} \) [14]. This three-term recurrence relation can be expressed in matrix form
\[
J \hat{P} = x \hat{P} - p_n(x) \times e_n
\]
where the \( i \)th row of \( \hat{P} \) is the coefficients of \( p_i(x) \) and \( e_n \) is the \( n \)th standard basis vector. The zeros, \( x_j \), of the polynomials up to degree \( n \) are used as the Gaussian nodes and are given by the eigenvalues of \( J \) which is tridiagonal. The weights then are computed by the first component of the corresponding eigenvector to each \( x_j \). That is, \( w_j = q_{1,j}^2 \times \int_a^b \omega(x) \, dx \), where \( q_{1,j} \) is the first component of the \( j \)th eigenvector of \( J \). See [16] for more a detailed discussion.

2.2 Numerical Schemes

In this work, we will utilize high order finite difference methods in order to solve the kinetic eikonal equation and the transport equation which arise in the study of the transformed kinetic equation with a BGK relaxation operator in the large scale hyperbolic limit [12, 17]. We will consider equations of the general form:
\[
\partial_t \varphi(t, x) + H(t, x, \nabla_x \varphi(t, x)) = s(t, x).
\] (2.1)

For simplicity the schemes will be discussed in 2D. We denote \( x = (x, y) \), \( v = (u, v) \), and \( \nabla_x \varphi = (\varphi_x, \varphi_y) \). The domain \( \Omega = \{x : x_{\min} \leq x \leq x_{\max}, y_{\min} \leq y \leq y_{\max}\} \) and the velocity domain is given by \( \mathbb{V} = \{v : u_{\min} \leq u \leq u_{\max}, v_{\min} \leq v \leq v_{\max}\} \). We assume that periodic boundary conditions in \((x, y)\) are imposed. The domain \( \Omega \) is discretized with a uniform mesh, \( \Omega_h = \{(x_i, y_j) : x_i = x_{\min} + i \Delta x, y_j = y_{\min} + j \Delta y; 0 \leq i \leq N_x, 0 \leq j \leq N_y\} \), where \( N_x \) and \( N_y \) are the number of points used in the \( x \) and \( y \) direction discretizations respectively. In our tests we assume \( N = N_x = N_y \). The step size used is \( \Delta x = \Delta y = \frac{2}{N-1} \). We denote the time step by \( \Delta t \). We denote the numerical approximation to the solution at time step \( n \) by \( \varphi(t^n, x, y_j) = \varphi(n \Delta t, x_{\min} + i \Delta x, y_{\min} + j \Delta y) = \varphi^n_{ij} \). Finally some standard notations are used such as:
\[
\Delta^+ \varphi_{ij} = \varphi_{i+1,j} - \varphi_{ij}, \quad \Delta^- \varphi_{ij} = \varphi_{ij} - \varphi_{i-1,j}
\]
\[
\Delta^+ \psi_{ij} = \psi_{i+1,j} - \psi_{ij}, \quad \Delta^- \psi_{ij} = \psi_{ij} - \psi_{i-1,j}
\]
We build on the starting block of a first order monotone scheme [18]

\[ \varphi^{n+1}_{ij} = \varphi^n_{ij} - \Delta t \hat{H} \left( \frac{\Delta x \varphi^n_{ij}}{\Delta x}, \frac{\Delta y \varphi^n_{ij}}{\Delta x}, \frac{\Delta y \varphi^n_{ij}}{\Delta x} \right) + \Delta t s^n_{ij} \quad (2.2) \]

where \( \hat{H} \) is a Lipschitz continuous monotone flux consistent with the effective Hamiltonian that is

\[ \hat{H}(p, q, q) = H(p, q). \]

### 2.2.1 Property preservation

The kinetic eikonal equation has some properties that need to be satisfied by any numerical scheme proposed to solve it. These properties are:

1. The viscosity solution, \( \varphi^0(t, x) \geq 0, \ (t, x) \in \mathbb{R}_+ \times \mathbb{R}^d \) [12].

2. \( H(p) \) is convex, where \( p = \nabla_x \varphi^0(t, x) \) [12].

3. \( H(0) = 0 \) and \( \nabla_p H(0) = 0 \), which is easily seen from equation (1.12) and its derivative.

4. \( H(p) \geq 0 \), and \( H(p) = 0 \) only if \( p = 0 \), which is obvious from the fact that \( H(p) \) is convex.

In the recovery of the effective Hamiltonian when computing \( \varphi^0(t, x) \) there are two main obstacles which we must overcome. The first is that these properties must be preserved. This is accomplished by using well-established schemes for time-dependent Hamilton-Jacobi equations [17, 18]. The second obstacle is that the effective Hamiltonian is defined as the root of an integral equation with respect to \( v \). The overall efficiency of our method is determined based on how efficiently these integrals, as well as the integrals defining the necessary components of equation (1.21), can be evaluated. This evaluation is done efficiently using Gauss quadrature and then in the case of recovering the Hamiltonian in equation (1.12), Newton’s method is used to solve the non-linear equation.
2.2.2 Numerical Hamiltonian

The effective Hamiltonian from equation (1.12) is monotone, which in the case of the numerical Hamiltonian means that it is non-increasing in the first and third arguments and non-decreasing in the second and fourth arguments.

There are a number of useful numerical Hamiltonians to consider:

(i) Godunov [18]:

\[
\hat{H}^G(p^+, p^-, q^+, q^-) = \text{ext}_{p \in I(p^-, p^+)} \text{ext}_{q \in I(q^-, q^+)} H(p, q),
\]

where

\[
\text{ext}_{p \in I(a, b)} = \begin{cases} 
\min_{a \leq p \leq b} & \text{if } a \leq b, \\
\max_{b \leq p \leq a} & \text{if } a > b,
\end{cases}
\]

and

\[
I(a, b) = [\min(a, b), \max(a, b)], \quad \text{and } H_1(p, q) = \frac{\partial H}{\partial p}, H_2(p, q) = \frac{\partial H}{\partial q}.
\]

(ii) Roe with Lax-Friedrichs entropy correction [18]:

\[
\hat{H}^{NL}(p^+, p^-, q^+, q^-) = \begin{cases} 
H(p^*, q^*) & \text{if } H_1(p, q) \text{ and } H_2(p, q) \text{ do not change signs in} \\
& p \in I(p^-, p^+), q \in I(q^+, q^-), \\
& H\left(\frac{p^+ + p^-}{2}, q^*\right) - \frac{1}{2} \alpha_x(p^+ - p^-) & \text{otherwise and if } H_2(p, q) \text{ does not change sign in} \\
& A \leq p \leq B, q \in I(q^-, q^+), \\
& H\left(p^*, \frac{q^+ + q^-}{2}\right) - \frac{1}{2} \alpha_y(q^+ - q^-) & \text{otherwise and if } H_1(p, q) \text{ does not change sign in} \\
& p \in I(p^-, p^+), C \leq q \leq D, \\
& H\left(\frac{p^+ + p^-}{2}, \frac{q^+ + q^-}{2}\right) - \frac{1}{2} \alpha_x(p^+ - p^-) - \frac{1}{2} \alpha_y(q^+ - q^-) & \text{otherwise},
\end{cases}
\]
where $A \leq p^\pm \leq B, C \leq q^\pm \leq D$ and

$$p^* = \begin{cases} p^+ & \text{if } H_1(p, q) \leq 0 \\ p^- & \text{if } H_1(p, q) \geq 0 \end{cases}, q^* = \begin{cases} q^+ & \text{if } H_2(p, q) \leq 0 \\ q^- & \text{if } H_2(p, q) \geq 0 \end{cases}$$

and

$$\alpha^x = \max_{p \in I(p^-, p^+), C \leq q \leq D} |H_1(p, q)|, \alpha^y = \max_{q \in I(q^-, q^+), A \leq p \leq B} |H_2(p, q)|.$$

The Godunov Hamiltonian, $\hat{H}^G$ is monotone for $A \leq p^\pm \leq B, C \leq q^\pm \leq D$, with $A, B, C, D$ being appropriate constants. Also, if $H(p, q) \geq 0$ for $A \leq p \leq B, C \leq q \leq D$, then $\hat{H}^G(p^+, p^-, q^+, q^-) \geq 0$. $\hat{H}^{RL}$ is advantageous for its simplicity in coding, however if $H(p, q) \geq 0$, $\hat{H}^{RL} \geq 0$ is not guaranteed, which is one property that we desire to preserve as observed in Section 2.2.1. This can be fixed by using a Godunov entropy correction in the Roe scheme rather than a Lax-Friedrichs entropy correction.

(iii) Roe with Godunov entropy correction [19]:

$$\hat{H}^{RG}(p^+, p^-, q^+, q^-) = \begin{cases} H(p^*, q^*) & \text{if } H_1(p, q) \text{ and } H_2(p, q) \text{ do not change signs in} \\
\text{ext}_{p \in I(p^-, p^+)} H(p^*) & p \in I(p^-, p^+), q \in I(q^+, q^-), \\
\text{ext}_{q \in I(q^-, q^+)} H(p^*, q) & \text{otherwise and if } H_2(p, q) \text{ does not change sign in} \\
\hat{H}^G(p^+, p^-, q^+, q^-) & A \leq p \leq B, q \in I(q^-, q^+), \end{cases}$$

where $A \leq p^\pm \leq B, C \leq q^\pm \leq D$ and

$$p^* = \begin{cases} p^+ & \text{if } H_1(p, q) \leq 0 \\ p^- & \text{if } H_1(p, q) \geq 0 \end{cases}, q^* = \begin{cases} q^+ & \text{if } H_2(p, q) \leq 0 \\ q^- & \text{if } H_2(p, q) \geq 0 \end{cases}.$$
In Section 4, all numerical results are computed using $\tilde{H}^G$, however similar results can be attained with $\tilde{H}^{RG}$ [19].

2.2.3 WENO finite difference method

![Figure 2.1: Simple stencil of the WENO scheme](image)

We now consider the spatial discretization that is used. We utilize a third order weighted essentially non-oscillatory (WENO) spatial discretization [17, 20, 21].

In the WENO scheme we construct a high order finite difference approximation to the gradient both in an upwind and downwind fashion. In the $x$ direction then, we have $(\varphi_x)_{i,j}^-$ and $(\varphi_x)_{i,j}^+$ as the left and right derivative approximations respectively.

$$(\varphi_x)_{i,j}^- = (1 - w_-)\left(\frac{\varphi_{i+1,j} - \varphi_{i-1,j}}{2\Delta x}\right) + w_-\left(\frac{3\varphi_{i,j} - 4\varphi_{i-1,j} + \varphi_{i-2,j}}{2\Delta x}\right),$$

where

$$w_- = \frac{1}{1 + 2r_-^2}, \quad r_- = \frac{\epsilon_0 + (\varphi_{i,j} - 2\varphi_{i-1,j} + \varphi_{i-2,j})^2}{\epsilon_0 + (\varphi_{i+1,j} - 2\varphi_{i,j} + \varphi_{i-1,j})^2}$$

Similarly we have in the upwind direction:

$$(\varphi_x)_{i,j}^+ = (1 - w_+)\left(\frac{\varphi_{i+1,j} - \varphi_{i-1,j}}{2\Delta x}\right) + w_+\left(\frac{-3\varphi_{i,j} + 4\varphi_{i+1,j} - \varphi_{i+2,j}}{2\Delta x}\right),$$

where

$$w_+ = \frac{1}{1 + 2r_+^2}, \quad r_+ = \frac{\epsilon_0 + (\varphi_{i,j} - 2\varphi_{i+1,j} + \varphi_{i+2,j})^2}{\epsilon_0 + (\varphi_{i+1,j} - 2\varphi_{i,j} + \varphi_{i-1,j})^2}$$

The gradient approximations, $(\varphi_y)_{i,j}^\pm$ are constructed similarly. $\epsilon_0$ is chosen as $10^{-10}$ to avoid division by zero.

We consider WENO schemes as opposed to simple finite difference schemes because WENO methods perform well when solutions develop kinks, shocks, discontinuities, etc. whereas simple schemes can introduce wild oscillations in the numerical solution [17, 18].
2.2.4 Total Variation Diminishing Runge-Kutta technique

The next part of the numerical method is the temporal discretization that is used. We utilize a third order total variation diminishing Runge-Kutta (TVDRK) scheme [18]. Let \( \hat{H}_{ij} = \hat{H}((\varphi_x)_{ij}^+, (\varphi_x)_{ij}^-, (\varphi_y)_{ij}^+, (\varphi_y)_{ij}^-) \) denote the numerical Hamiltonian used in the method and let \( \varphi^{(\nu)} \) be the intermediate solution at the \( \nu^{th} \) stage of the RK scheme. Then the TVDRK scheme is given thus:

\[
\begin{align*}
\varphi_{ij}^{(1)} &= \varphi_{ij}^{n} - \Delta t (\hat{H}_{ij}^{(1)} - s_{ij}^{(1)}) \\
\varphi_{ij}^{(2)} &= \frac{3}{4} \varphi_{ij}^{n} + \frac{1}{4} \varphi_{ij}^{(1)} - \frac{1}{4} \Delta t (\hat{H}_{ij}^{(2)} - s_{ij}^{(2)}) \\
\varphi_{ij}^{n+1} &= \frac{1}{3} \varphi_{ij}^{n} + \frac{2}{3} \varphi_{ij}^{(2)} - \frac{2}{3} \Delta t (\hat{H}_{ij}^{n+1} - s_{ij}^{n+1})
\end{align*}
\]  

(2.6)

2.2.5 Algorithm

Based on the presented numerical techniques for achieving high order accuracy in space and time, we summarize the algorithm for equations in the form of equation (2.1).

Algorithm 1 WENO-RK method

1: while \( (t^n < t_{final}) \) do
2:     for all \( x_i, y_j \in \Omega^h \) do
3:         Compute \( (\varphi_x^\pm)_{ij}, (\varphi_y^\pm)_{ij} \) with WENO3
4:         Utilize TVDRK3 to attain \( \varphi_{ij}^{n+1} \)
5:     end for
6:     \( n = n + 1 \)
7: end while

2.2.6 Recovery of the effective Hamiltonian

We utilize Gaussian quadrature in order to discretize the integral equation (1.12). We test two Gauss quadrature rules where the weights and abscessas are generated with the Golub and Welsch algorithm briefly mentioned in Section 2.1.4 and presented in detail in [16]. The first rule is simple Gauss-Legendre quadrature in which \( \omega(v) = 1 \), however in the case where \( M(v) \) is non-constant, it is desirable to generate a Gauss rule where \( \omega(v) = M(v) \). Gauss-Legendre is tested to verify that treating \( M(v) \) as the weight function is accurate. After the integral is discretized Newton’s method is implemented to solve the non-linear equation. That is, once
the gradient is approximated, $H(p, q) = H$, and the following integral equation arises:

$$F(H) = \int_{v_{\text{min}}}^{v_{\text{max}}} \int_{u_{\text{min}}}^{u_{\text{max}}} \frac{M(u, v)}{1 + H - u \cdot p - v \cdot q} \, du \, dv - 1 = 0. \tag{2.7}$$

For implementing Newton’s method, $F'(H)$ is required which is given by:

$$F'(H) = -\int_{v_{\text{min}}}^{v_{\text{max}}} \int_{u_{\text{min}}}^{u_{\text{max}}} \frac{M(u, v)}{(1 + H - u \cdot p - v \cdot q)^2} \, du \, dv.$$  

Discretizing the integrals for Gauss quadrature gives the following 2 equations:

$$F(H) = \int_{v_{\text{min}}}^{v_{\text{max}}} \int_{u_{\text{min}}}^{u_{\text{max}}} \frac{M(u, v)}{1 + H - u \cdot p - v \cdot q} \, du \, dv - 1 = \sum_{j=1}^{T} \sum_{i=1}^{T} w_i \cdot w_j \cdot Y_{ij} - 1 = 0, \tag{2.8}$$

$$F'(H) = -\int_{v_{\text{min}}}^{v_{\text{max}}} \int_{u_{\text{min}}}^{u_{\text{max}}} \frac{M(u, v)}{(1 + H - u \cdot p - v \cdot q)^2} \, du \, dv = -\sum_{j=1}^{T} \sum_{i=1}^{T} w_i \cdot w_j \cdot Y_{ij} = 0. \tag{2.9}$$

where $Y_{ij} = 1$ for $0 \leq i, j \leq T$ in the case of Gaussian quadrature where $\omega(u, v) = M(u, v)$, or $Y_{ij} = M(u_j, v_i)$ for $0 \leq i, j \leq T$ in the case where $\omega(u, v) = 1$. $\{w^u, u\}_{j=1}^{T}, \{w^v, v\}_{i=1}^{T}$ are appropriate quadrature weights and abscissas. It is clear that $F(H)$ is decreasing for $H \in [\max\{0, \max_{u,v} \{u \cdot p - v \cdot q - 1\}\}, \infty)$ since $F'(H) < 0$. Therefore $F(H)$ has a unique solution on $[\max\{0, \max_{u,v} \{u \cdot p - v \cdot q - 1\}\}, \infty)$. We pick $H_0 = \max u \cdot |p| + \max v \cdot |q| - 1 + \gamma$ for some $\gamma > 0$ and summarize Newton’s method for equation (2.7):

**Algorithm 2** Newton’s Method for equation (2.7)

1. $H_0 = \max\{0, |p| \cdot u_{\text{max}} + |q| \cdot v_{\text{max}} - 1 + \gamma\}$
2. $m = 0$
3. while $|H_m - H_{m-1}| > \text{TOL}$ & $m \leq \text{Max\_iter}$ do  
   4. $H_{m+1} = H_m - F(H_m)/F'(H_m)$,
   5. where $F(H_m)$ and $F'(H_m)$ are approximated by (2.8) and (2.9).
6. $H_{m+1} = \max\{H_{m+1}, 0, |p| \cdot u_{\text{max}} + |q| \cdot v_{\text{max}} - 1\}$
7. $m = m + 1$
8. end while
9. return $H_m$
2.3 The Method

We use the previously mentioned methods and tools to recover $\varphi^0$ and $\varphi^1$, which are then used to transform the phase function, $\varphi^\epsilon$, back to $f^\epsilon$.

2.3.1 Recovering $\varphi^0$

$\varphi^0(t, x)$ is obtained by implementing Algorithm 1. At each time step, once $\varphi^0_x$ and $\varphi^0_y$ are computed, $H(\varphi^0_x, \varphi^0_y)$ is the root of equation (2.8). We use Algorithm 2 to recover $H$, noting that the positivity of $H$ is maintained [19]. $F(H)$ and $F'(H)$ are computed using Gauss quadrature. Furthermore, in order to maintain the positivity of $\varphi^0$, after the RK procedure in step (4) of Algorithm 1 we set:

$$\varphi^{0,n+1}_{ij} = \max\{\varphi^{0,n+1}_{ij}, 0\}. \quad (2.10)$$

We enforce non-negativity in the solution to equation (1.11) by this step. The solution is in fact always nonnegative (Sec. 2.2.1) and our method must preserve this property. In general, a $P^{th}$-order WENO method is not guaranteed to satisfy the maximum principle, and therefore is not guaranteed to maintain positivity of the solution. Equation (2.10) is enough to guarantee positivity and we claim that simultaneously it maintains high order accuracy (third order with the presented schemes)[19].

We verify this in the following way. For simplicity denote $\varphi^0 = \psi$ and $\psi_0(x) = \psi(t = 0, x)$. First, denote $0 = \psi_{\min} = \min\{\psi_{0_{ij}}\}, \psi_{\max} = \max\{\psi_{0_{ij}}\},$ and then we state the maximum principle as: if $\psi_{\min} \leq \{\dot{\psi}_{ij}\} \leq \psi_{\max},$ then $\psi_{\min} \leq \{\psi^{n+1}_{ij}\} \equiv \{(\psi^n - \Delta t\dot{H}(\psi^n))_{ij}\} \leq \psi_{\max}$, where $\dot{H}$ denotes the numerical Hamiltonian after all stages in the Runge-Kutta procedure. Since the Hamiltonian is nonnegative (Sec. 2.2.1), it is obvious that $\psi_{ij}^{n+1} \equiv \psi_{ij}^n - \Delta t\dot{H}(\{\psi^n_{ij}\}) \leq \psi_{ij}^n \leq \psi_{\max}$. Clearly, by imposing (2.10) $\{\psi^{n+1}_{ij}\} \geq \psi_{\min} = 0$. The maximum principle is therefore satisfied. We also claim that the accuracy is maintained by enforcing equation (2.10). To verify this assume that the solution is smooth, then what we must show is that $|\psi_{ij}^{n+1} - \max\{\psi_{ij}^{n+1}, 0\}| \leq \Delta tO(h^P)$, where $P$ is the order of accuracy of the WENO approximation and $h$ is the step size. This will tell us that over $n$ time steps, where $n\Delta t = t_{final}$, the error is bounded by $O(h^P)$ (ie. the scheme is $P^{th}$-order accurate). Recall the kinetic eikonal equation
(1.10). We then note the following:

$$\frac{1}{1 + H(p) - \mathbf{v} \cdot \mathbf{p}} = \frac{1}{1 + H(p) - (\mathbf{v} \cdot \mathbf{p})^{1+H(p)/1+H(p)}} = \frac{1}{1 + H(p)} \cdot \frac{1}{1 - \frac{\mathbf{v} \cdot \mathbf{p}}{1+H(p)}}.$$ 

Since the integrand of equation (1.10) is positive for all \( \mathbf{v} \), then \( \frac{\mathbf{v} \cdot \mathbf{p}}{1+H(p)} < 1 \) which means that

$$\frac{1}{1 - \frac{\mathbf{v} \cdot \mathbf{p}}{1+H(p)}} \approx \frac{1}{1 - |x|} \text{ for } |x| < 1.$$ 

Recall by Taylor that

$$\frac{1}{1-x} = \sum_{m=0}^{\infty} x^m.$$ 

Then we have the following [19]:

$$\int_{\mathbb{M}^0} \frac{M(\mathbf{v})}{1 + H(p) - \mathbf{v} \cdot \mathbf{p}} d\mathbf{v} = 1,$$

$$\Rightarrow \int_{\mathbb{M}^0} \frac{M(\mathbf{v})}{1 - \frac{\mathbf{v} \cdot \mathbf{p}}{1+H(p)}} d\mathbf{v} = 1 + H(p),$$

$$\Rightarrow \int_{\mathbb{M}^0} M(\mathbf{v}) \sum_{m=0}^{\infty} \left( \frac{\mathbf{v} \cdot \mathbf{p}}{1 + H(p)} \right)^m d\mathbf{v} = 1 + H(p),$$

$$\Rightarrow \int_{\mathbb{M}^0} M(\mathbf{v}) \left( \frac{\mathbf{v} \cdot \mathbf{p}}{1 + H(p)} \right)^2 d\mathbf{v} + \int_{\mathbb{M}^0} M(\mathbf{v}) \sum_{m=3}^{\infty} \left( \frac{\mathbf{v} \cdot \mathbf{p}}{1 + H(p)} \right)^m d\mathbf{v} = H(p)$$

$$\Rightarrow \int_{\mathbb{M}^0} M(\mathbf{v}) \left( \frac{\mathbf{v} \cdot \mathbf{p}}{1 + H(p)} \right)^2 d\mathbf{v} + \int_{\mathbb{M}^0} M(\mathbf{v}) \sum_{m=3}^{\infty} \left( \frac{\mathbf{v} \cdot \mathbf{p}}{1 + H(p)} \right)^m d\mathbf{v} = H(p),$$

which (recalling the moment identities of \( M(\mathbf{v}) \)) implies that \( H(p) \sim \theta^2 p^2 \) as \( p \sim 0, \theta \in \mathbb{R} \).

If \( x_{ij} = (x_i, y_j) \) is a local minimum point then we have \( |\hat{\mathcal{H}}(\{\psi^n_{ij}\})| \sim \theta^2 |\hat{\mathbf{p}}^2_{ij}| \sim \theta^2 |\hat{\mathbf{p}}_{ij} - \mathbf{0}|^2 \sim \mathcal{O}(h^2P) \) with \( \hat{\mathbf{p}}_{ij} \) denoting the P-th order WENO approximation of \( \nabla \psi^n_{ij} \sim 0 \). If \( \psi^{n+1}_{ij} \geq 0 \), then \( |\psi^{n+1}_{ij} - \max\{\psi^n_{ij}, 0\}| = 0 \). If \( \psi^{n+1}_{ij} < 0 \), we have that \( \psi^n_{ij} \geq 0 > \psi^{n+1}_{ij} \), and \( |\psi^{n+1}_{ij} - \max\{\psi^n_{ij}, 0\}| = |\psi^{n+1}_{ij} - 0| < |\psi^n_{ij} - \psi^{n+1}_{ij}| = |\Delta t \hat{\mathcal{H}}(\{\psi^n_{ij}\})| \). Therefore, \( |\psi^{n+1}_{ij} - \max\{\psi^{n+1}_{ij}, 0\}| \leq \Delta t \mathcal{O}(h^{2P}) \). If \( x_{ij} = (x_i, y_j) \) is close to a local minimum point, \( x_* = (x_*, y_*) \), then by L’Hospital’s
Similarly, we have \( \lim_{x \to x_*} \frac{x \partial}{\partial x} \psi(t, x) = \lim_{x \to x_*} \frac{\partial}{\partial t} \psi(t, x) + x \frac{\partial^2}{\partial x^2} \psi(t, x) \)

\[
= \lim_{x \to x_*} \frac{2 \partial^2}{\partial x^2} \psi(t, x) + x \frac{\partial^3}{\partial x^3} \psi(t, x) \\
\]

\[
= \ldots \\
= \lim_{x \to x_*} \frac{K}{x} \frac{\partial^K}{\partial x^K} \psi(t, x) + x \frac{\partial^{K+1}}{\partial x^{K+1}} \psi(t, x) \\
\]

\[
= \frac{\partial^K}{\partial x^K} \psi(t, x) = K,
\]

where \( K \) is the smallest constant such that \( \frac{\partial^k}{\partial x^k} \psi(t, x_*) = 0 \) for \( k \leq K - 1 \), but \( \frac{\partial^K}{\partial x^K} \psi(t, x_*) \neq 0 \). Similarly, we have \( \lim_{x \to x_*} \frac{y \partial}{\partial y} \psi(t, x) = L \), where \( L \) is the smallest constant such that \( \frac{\partial^l}{\partial y^l} \psi(t, x_*) = 0 \) for \( l \leq L - 1 \), but \( \frac{\partial^L}{\partial y^L} \psi(t, x_*) \neq 0 \). Note that we only apply L'Hospital's rule if it is applicable and after the necessary cancelations. Therefore we have that \( \|x\|_\infty \|\psi(t, x)\|_\infty \leq C \) for some constant \( C > 0 \) as \( x \to x_* \). That is, we have a lower bound on \( \psi(t, x) \) as \( x \to x_* \). We know that \( |H(\nabla_x \psi(t, x))| \leq V_{max} \|\nabla_x \psi(t, x)\|_\infty \) (See Section 2.2.1). Hence, if we choose the Courant-Friedrichs-Lewy (CFL) condition such that \( \Delta t \leq \frac{\max(|\Delta x, \Delta y|)}{2C V_{max}} \), then \( \psi_{n+1} = \psi_{n} - \Delta t \hat{H}(\{\psi_{n}^i\}) \geq 0 \). The constant \( C \) can be estimated through the initial condition \( \psi_0(x) \).

### 2.3.2 Recovering \( \varphi^1 \)

From Section 1.2.2, recall that \( \varphi^1(t, x, v) = \log G(t, x, v) - \lambda(t, x) \), with \( G(t, x, v) \equiv 1 - \partial_t \varphi^0(t, x) - v \cdot \nabla_x \varphi^0(t, x) \) and \( \lambda(t, x) \) solves equation (1.18). The coefficients of the transport equation are given in equation (1.19) and can be computed using Gauss quadrature rules. That is:

\[
a(t, x) \approx \sum_{j=1}^{T} \sum_{i=1}^{T} \frac{w_j^u w_i^v Y_{ij}}{(1 + H - u_j \varphi_0^x - v_i \varphi_0^y)^2}, \\
B(t, x) \approx \sum_{j=1}^{T} \sum_{i=1}^{T} \frac{w_j^u w_i^v Y_{ij}}{(1 + H - u_j \varphi_0^x - v_i \varphi_0^y)^2}, \\
r(t, x) \approx \sum_{j=1}^{T} \sum_{i=1}^{T} \frac{w_j^u w_i^v Y_{ij} (- \frac{B_{ij}}{a_{ij}} - u_j - v_i) \cdot D_x^2 \varphi^0(t, x) \cdot (\frac{B_{ij}}{a_{ij}} - u_j - v_i))}{(1 + H - u_j \varphi_0^x - v_i \varphi_0^y)^3},
\]
Where $Y_{ij}, w_j^u, u_j, w_i^v, v_i$ are defined as in Section 2.2.6.

Equation (1.18) is a simple transport equation of the form equation (2.1) with coefficients defined as above, therefore Algorithm 1 is used to compute $\lambda$. In the computation of $r(t, x)$ an approximation to $D_x^2\varphi^0(t, x)$ is required. This is obtained by applying WENO dimension by dimension.

It is in the steps where the necessary components defined as integrals with respect to the velocity variables are computed that the overall efficiency of the method is determined. If $T \ll N$, then the method will be significantly more efficient than standard methods since the overall complexity will be $O(T^d N^d)$ as opposed to $O(N^{2d})$. We discover that in practice, $T = 16$ seems to be enough to faithfully recover $H$. We make one final remark before presenting numerical results that this proposed method is independent of $\epsilon$. 
CHAPTER 3. RESULTS

3.1 Introduction

In this section we present some numerical results of the discussed method. In all cases, the initial data given is periodic. All examples utilize Algorithm 1 with $\tilde{H}^G$ recovered using Algorithm 2. In all experiments, the computational domain is $\Omega = [-1,1]^d$, and the velocity space, $V = [-1,1]^d$. All experiments were performed on a Dell Optiplex 990 with 3.4 GHz 4 core i7 processor, 16 GB 1333MHz DDR3 using MATLAB R2015a and double precision. All plots are generated with MATLAB as well. The Knudsen number, $\epsilon$, is taken to be $1e-2$ in all experiments. Results are presented in 1D and 2D. Reference solutions are computed using the IMEX-RK method [5]. When verifying the accuracy of the proposed method for recovering the probability density function, we include $L^\infty$ as well as relative $L^\infty$ and relative $L^1$ errors between a reference solution and the approximation computed with the proposed method where the relative errors are given by:

$$\frac{\|f^\epsilon(t,x,v) - f^{\epsilon,ref}(t,x,v)\|_\infty}{\|f^{\epsilon,ref}(t,x,v)\|_\infty}, \frac{\|f^\epsilon(t,x,v) - f^{\epsilon,ref}(t,x,v)\|_1}{\|f^{\epsilon,ref}(t,x,v)\|_1},$$

where $f^{\epsilon,ref}(t,x,v)$ is the reference solution to the particle density function, and $f^\epsilon(t,x,v)$ is the solution computed with the proposed method. We remark that Gauss-Legendre quadrature is used to compute the required moments of the kinetic equation with BGK relaxation operator in the IMEX-RK scheme. The Legendre points then serve as the velocity mesh. This is only for the purposes of verifying the accuracy of the proposed method by comparing it to the velocity mesh on which the IMEX-RK scheme calculates a solution. In practice, if $f^\epsilon(t,x,v)$ is required at some velocity point not present in the set of quadrature points then more points must be used in the velocity discretization in the IMEX-RK scheme whereas the proposed method is
independent of the velocity mesh. The velocity mesh is unrelated to the Gaussian quadrature points used for computing the necessary components of equations (1.12) and (1.21).

3.2 1D

3.2.1 Computation of the effective Hamiltonian

In the case of the constant Maxwellian, \( M(v) = \frac{1}{2} \), we have a closed form of the Hamiltonian, 
\[ H(p) = \frac{v - \tanh(p)}{\tanh(p)} \] [12]. We use this to check the \( L^\infty \) norm of the error of the computed Hamiltonian. Periodic initial data is given and exact gradient values are used. In the non-constant case we simply calculate a reference Hamiltonian using a high number of Gaussian nodes for accuracy and compare the error when using fewer nodes. In this case the reference Hamiltonian is calculated using 64 Gaussian nodes. In all cases, the spatial mesh used contains 2049 points. The initial data given is \( \varphi^0(\cdot, x) = 0.1(1 + \sin(x\pi)) \). Table 3.1 shows the accuracy of varying the number of Gaussian Nodes in both Maxwellian cases, using both quadrature rules.

Table 3.1: \( L^\infty \) error of Hamiltonian based on number of Gaussian nodes and quadrature rule

<table>
<thead>
<tr>
<th>T</th>
<th>( \omega(v) = 1, M(v) = \frac{1}{2} )</th>
<th>( \omega(v) = M(v) = \frac{1}{2} )</th>
<th>( \omega(v) = 1, M(v) = \frac{1}{\sqrt{\pi \text{erf}(1)}} )</th>
<th>( \omega(v) = M(v) = \frac{1}{\sqrt{\pi \text{erf}(1)}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>1.0749865e-15</td>
<td>1.0026702e-15</td>
<td>1.8943180e-15</td>
<td>1.6792123e-15</td>
</tr>
<tr>
<td>12</td>
<td>7.9797280e-16</td>
<td>1.2290516e-15</td>
<td>1.1692036e-15</td>
<td>2.0261570e-15</td>
</tr>
<tr>
<td>10</td>
<td>6.5225603e-16</td>
<td>8.0144225e-16</td>
<td>3.3668505e-13</td>
<td>1.1082715e-15</td>
</tr>
</tbody>
</table>

3.2.2 Computation of \( \varphi^0 \)

3.2.2.1 Experiment 1.1

We present a 1D example demonstrating the accuracy of computing \( \varphi^0(t, x) \). The initial data given is \( \varphi^0(\cdot, x) = 0.5(1 + \sin(x\pi)) \). We test \( M(v) = \frac{1}{2} \) and \( M(v) = \frac{e^{-v^2}}{\sqrt{\pi \text{erf}(1)}} \). Reference solutions are computed on a mesh of 32769 points with 32 abscissas and weights. Final time for checking convergence is \( t = 0.125 \), a time when the solution is still smooth.
Third order convergence is achieved until the error of computing the effective Hamiltonian becomes dominant in the solution (bolded).

Table 3.2: Error and convergence rates when $M(v) = \frac{1}{2}$ using Gauss-Legendre quadrature with $T = 16$

<table>
<thead>
<tr>
<th>N</th>
<th>$dx$</th>
<th>$L^1$</th>
<th>$L^\infty$</th>
<th>$L^1$ - rate</th>
<th>$L^\infty$ - rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>1.2500e-01</td>
<td>5.305923e-03</td>
<td>6.275868e-03</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>33</td>
<td>6.2500e-02</td>
<td>1.095231e-03</td>
<td>1.647334e-03</td>
<td>2.276369</td>
<td>1.929682</td>
</tr>
<tr>
<td>65</td>
<td>3.1250e-02</td>
<td>1.150160e-04</td>
<td>1.772433e-04</td>
<td>3.251329</td>
<td>3.21633</td>
</tr>
<tr>
<td>129</td>
<td>1.5625e-02</td>
<td>4.407452e-06</td>
<td>4.752238e-06</td>
<td>4.705746</td>
<td>5.220980</td>
</tr>
<tr>
<td>257</td>
<td>7.8125e-03</td>
<td>1.941064e-07</td>
<td>1.743888e-07</td>
<td>4.505025</td>
<td>4.768228</td>
</tr>
</tbody>
</table>

Table 3.3: Error and convergence rates when $\omega(v) = M(v) = \frac{1}{2}$ using the Golub and Welsch algorithm with $T = 16$

<table>
<thead>
<tr>
<th>N</th>
<th>$dx$</th>
<th>$L^1$</th>
<th>$L^\infty$</th>
<th>$L^1$ - rate</th>
<th>$L^\infty$ - rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>1.2500e-01</td>
<td>5.305923e-03</td>
<td>6.275868e-03</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>33</td>
<td>6.2500e-02</td>
<td>1.095231e-03</td>
<td>1.647334e-03</td>
<td>2.276369</td>
<td>1.929682</td>
</tr>
<tr>
<td>65</td>
<td>3.1250e-02</td>
<td>1.150160e-04</td>
<td>1.772433e-04</td>
<td>3.251329</td>
<td>3.21633</td>
</tr>
<tr>
<td>129</td>
<td>1.5625e-02</td>
<td>4.407452e-06</td>
<td>4.752238e-06</td>
<td>4.705746</td>
<td>5.220980</td>
</tr>
<tr>
<td>257</td>
<td>7.8125e-03</td>
<td>1.941064e-07</td>
<td>1.743888e-07</td>
<td>4.505025</td>
<td>4.768228</td>
</tr>
</tbody>
</table>

Table 3.4: Error and convergence rates when $M(v) = \frac{1}{\sqrt{\pi\text{erf}(1)}}e^{-v^2}$ using Gauss-Legendre with $T = 16$

<table>
<thead>
<tr>
<th>N</th>
<th>$dx$</th>
<th>$L^1$</th>
<th>$L^\infty$</th>
<th>$L^1$ - rate</th>
<th>$L^\infty$ - rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>1.2500e-01</td>
<td>5.106213e-03</td>
<td>6.341210e-03</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>33</td>
<td>6.2500e-02</td>
<td>1.065472e-03</td>
<td>1.712016e-03</td>
<td>2.260762</td>
<td>1.889062</td>
</tr>
<tr>
<td>65</td>
<td>3.1250e-02</td>
<td>1.125089e-04</td>
<td>1.863590e-04</td>
<td>3.243381</td>
<td>3.199540</td>
</tr>
<tr>
<td>129</td>
<td>1.5625e-02</td>
<td>4.407515e-06</td>
<td>5.633374e-06</td>
<td>4.647131</td>
<td>5.047942</td>
</tr>
<tr>
<td>257</td>
<td>7.8125e-03</td>
<td>8.300310e-07</td>
<td>2.107892e-06</td>
<td>2.435527</td>
<td>1.418198</td>
</tr>
</tbody>
</table>
Table 3.5: Error and convergence rates when $\omega(v) = M(v) = \frac{1}{\sqrt{\pi \text{erf}(1)}} e^{-v^2}$ using the Golub and Welsch algorithm with $T = 16$

<table>
<thead>
<tr>
<th>N</th>
<th>$dx$</th>
<th>$L^1$</th>
<th>$L^\infty$</th>
<th>$L^1$ - rate</th>
<th>$L^\infty$ - rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>1.2500e-01</td>
<td>5.106140e-03</td>
<td>6.340978e-03</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>33</td>
<td>6.2500e-02</td>
<td>1.065469e-03</td>
<td>1.711855e-03</td>
<td>2.260745</td>
<td>1.889145</td>
</tr>
<tr>
<td>65</td>
<td>3.1250e-02</td>
<td>1.125527e-04</td>
<td>1.859865e-04</td>
<td>3.242816</td>
<td>3.202290</td>
</tr>
<tr>
<td>129</td>
<td>1.5625e-02</td>
<td>4.665698e-06</td>
<td>5.631632e-06</td>
<td>4.592363</td>
<td>5.045501</td>
</tr>
<tr>
<td>257</td>
<td>7.8125e-03</td>
<td>1.095412e-06</td>
<td>2.862902e-06</td>
<td>2.090619</td>
<td>0.9760748</td>
</tr>
</tbody>
</table>

3.2.2.2 1D Computational Time

Table 3.6 shows differing computational times for varying weights/abscissas and quadrature rules. We remark that the improvements are noticed much more in higher dimensions. We further remark that computational time is highly subjective to the machine, language, and how efficiently the algorithm is implemented, yet it does demonstrate that as $T$ increases, the increase in computational cost is noticeable.

Table 3.6: Computational time, in seconds, for the 1D code to compute a solution to $\varphi^0$.

<table>
<thead>
<tr>
<th>T</th>
<th>$\omega(v) = 1, M(v) = \frac{1}{2}$</th>
<th>$\omega(v) = M(v) = \frac{1}{2}$</th>
<th>$\omega(v) = 1, M(v) = \frac{1}{\sqrt{\pi \text{erf}(1)}}$</th>
<th>$\omega(v) = M(v) = \frac{1}{\sqrt{\pi \text{erf}(1)}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>22.20491</td>
<td>22.36696</td>
<td>16.10435</td>
<td>21.42997</td>
</tr>
<tr>
<td>10</td>
<td>22.92549</td>
<td>23.06833</td>
<td>15.54712</td>
<td>22.15988</td>
</tr>
<tr>
<td>12</td>
<td>23.91023</td>
<td>23.63698</td>
<td>22.56568</td>
<td>22.64293</td>
</tr>
<tr>
<td>14</td>
<td>24.38379</td>
<td>24.10242</td>
<td>22.96163</td>
<td>23.11213</td>
</tr>
<tr>
<td>16</td>
<td>25.23493</td>
<td>24.55474</td>
<td>23.43072</td>
<td>23.60767</td>
</tr>
</tbody>
</table>

3.2.2.3 Experiment 1.2

Consider the initial data $\varphi^0(t = 0, x) = 0.5(1 + \sin(x\pi))$. We plot solutions considering both $M(v) = \frac{1}{2}$ and $M(v) = \frac{1}{\sqrt{\pi \text{erf}(1)}}$. Plots of the solution at $t = 0.125$ and 5 as well as the
computed effective Hamiltonian vs. the exact Hamiltonian (in the constant Maxwellian case) are included. All solutions are computed on a grid of 129 points with 16 abscissas/weights.

Figure 3.1: Constant Maxwellian, $M(v) = \frac{1}{2}$ with (Left) computed effective Hamiltonian in red diamonds with exact Hamiltonian in green circles. (Right) Solution, $\varphi^0$. 
Figure 3.2: Non-constant Maxwellian, $M(v) = \frac{e^{-v^2}}{\sqrt{\pi \text{erf}(1)}}$ with (Left) computed effective Hamiltonian in red circles. (Right) Solution, $\varphi^0$.

3.2.3 Computation of $\varphi^1$

3.2.3.1 Experiment 1.3

We consider the following initial data: $\varphi^0(t = 0, x) = 0.5(1 + \sin(2\pi x))$ and $\lambda(t = 0, x) = -8 \cdot (1 + \cos(4\pi x))$. We consider both $M(v) = \frac{1}{2}$ and $M(v) = \frac{e^{-v^2}}{\sqrt{\pi \text{erf}(1)}}$. In this work we have considered third order accurate schemes in space and time for $\varphi^0$. The computation of $\varphi^1$ involves a second derivative of $\varphi^0$ and thus we lose two orders of accuracy, hence $\varphi^1$ will only be first order accurate even when the third order scheme is utilized. To verify this, a reference solution for $\lambda$ is computed on a grid of 2049 points with final time, $t = 0.03125$, a time when $\lambda$ is still smooth. Once $\varphi^0(t, x)$ and $\lambda(t, x)$ are computed, $\varphi^1(t, x, v)$ can be computed easily by
3.2.4 Approximating \( f^\epsilon \)

Once \( \varphi^0 \) and \( \varphi^1 \) are calculated, we can transform back to the original density function. We consider a 1D example with both the constant and non-constant Maxwellian cases to demonstrate the computation of \( f^\epsilon(t, x, v) \). All solutions are computed on a grid of \((2\epsilon+1)\times16 \in \mathbb{R} \times \mathbb{V}\). \( \lambda(t = 0, x) = -(1 + \cos(x\pi)) \). The final time for checking errors is \( t = 0.25 \) to ensure that solutions are smooth. The relationship \( N = \frac{2}{\epsilon} \) is important because the IMEX-RK solver is dependent on \( \epsilon \), and as \( \epsilon \) is decreased more points are needed to resolve any kinks that appear in the solutions due to the multi-scale nature of the kinetic equation with a BGK relaxation operator being considered. We expect, and do in fact see, that the error is of \( \mathcal{O}(\epsilon) \). Convergence rates in the \( L^\infty \) norm as well as the relative \( L^1 \) and \( L^\infty \) norms are checked.

Table 3.7: Error and convergence rates of \( \lambda \) with \( M(v) = \frac{1}{2} \)

<table>
<thead>
<tr>
<th>N</th>
<th>( dx )</th>
<th>( L^1 )</th>
<th>( L^\infty )</th>
<th>( L^1 ) - rate</th>
<th>( L^\infty ) - rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>65</td>
<td>3.125000e-02</td>
<td>1.170883e-01</td>
<td>1.421991e-01</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>129</td>
<td>1.562500e-02</td>
<td>3.814379e-02</td>
<td>6.647100e-02</td>
<td>1.618077</td>
<td>1.097116</td>
</tr>
<tr>
<td>257</td>
<td>7.812500e-03</td>
<td>2.18655e-02</td>
<td>3.53192e-02</td>
<td>0.8025907</td>
<td>0.9871898</td>
</tr>
<tr>
<td>513</td>
<td>3.906250e-03</td>
<td>1.001081e-02</td>
<td>1.408686e-02</td>
<td>1.127298</td>
<td>1.251185</td>
</tr>
<tr>
<td>1025</td>
<td>1.953125e-03</td>
<td>3.418999e-03</td>
<td>4.739482e-03</td>
<td>1.549913</td>
<td>1.571549</td>
</tr>
</tbody>
</table>

Table 3.8: Error and convergence rates of \( \lambda \) with \( M(v) = \frac{e^{-v^2}}{\sqrt{\pi\text{erf}(1)}} \)

<table>
<thead>
<tr>
<th>N</th>
<th>( dx )</th>
<th>( L^1 )</th>
<th>( L^\infty )</th>
<th>( L^1 ) - rate</th>
<th>( L^\infty ) - rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>65</td>
<td>3.125000e-02</td>
<td>3.185874e-01</td>
<td>4.579963e-01</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>129</td>
<td>1.562500e-02</td>
<td>1.433685e-01</td>
<td>2.403258e-01</td>
<td>1.151962</td>
<td>0.9303444</td>
</tr>
<tr>
<td>257</td>
<td>7.812500e-03</td>
<td>6.803434e-02</td>
<td>1.143455e-01</td>
<td>1.075393</td>
<td>1.071592</td>
</tr>
<tr>
<td>513</td>
<td>3.906250e-03</td>
<td>2.968720e-02</td>
<td>4.904874e-02</td>
<td>1.196422</td>
<td>1.221111</td>
</tr>
<tr>
<td>1025</td>
<td>1.953125e-03</td>
<td>1.000634e-02</td>
<td>1.724578e-02</td>
<td>1.568926</td>
<td>1.507973</td>
</tr>
</tbody>
</table>
3.2.4.1 Experiment 1.4

We consider the initial data given to be \( \varphi_0 = (1 - 4)(1 + \sin(2x\pi)) \). We remark that \( \varphi_0 = (1 - 4)(1 + \sin(2x\pi)) \) implies that the system is initially close to equilibrium.

Table 3.9: Error and convergence rates with \( \varphi_0 = (1 - 4)(1 + \sin(2x\pi)) \) and \( M(v) = \frac{1}{2} \)

<table>
<thead>
<tr>
<th>( \epsilon )</th>
<th>( L^1 )</th>
<th>( L^\infty )</th>
<th>( L^1_{\text{relative}} )</th>
<th>( L^\infty_{\text{relative}} )</th>
<th>( L^\infty ) rate</th>
<th>( L^1_{\text{relative}} ) rate</th>
<th>( L^\infty_{\text{relative}} ) rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.829565</td>
<td>0.009145</td>
<td>0.013181</td>
<td>0.018460</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>9.788161</td>
<td>0.004601</td>
<td>0.006664</td>
<td>0.009375</td>
<td>0.991129</td>
<td>0.984088</td>
<td>0.977613</td>
<td></td>
</tr>
<tr>
<td>9.625535</td>
<td>0.002305</td>
<td>0.003351</td>
<td>0.004781</td>
<td>0.996825</td>
<td>0.991892</td>
<td>0.971385</td>
<td></td>
</tr>
<tr>
<td>9.273200</td>
<td>0.001152</td>
<td>0.001680</td>
<td>0.002464</td>
<td>1.000803</td>
<td>0.995885</td>
<td>0.956315</td>
<td></td>
</tr>
<tr>
<td>8.608471</td>
<td>0.000574</td>
<td>0.000841</td>
<td>0.001287</td>
<td>1.004338</td>
<td>0.997821</td>
<td>0.937235</td>
<td></td>
</tr>
<tr>
<td>7.482127</td>
<td>0.000286</td>
<td>0.000421</td>
<td>0.000676</td>
<td>1.007515</td>
<td>0.998410</td>
<td>0.928472</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.10: Error and convergence rates with \( \varphi_0 = (1 - 4)(1 + \sin(2x\pi)) \) and \( M(v) = \frac{e^{-v^2}}{\sqrt{\text{erf}(1)}} \)

<table>
<thead>
<tr>
<th>( \epsilon )</th>
<th>( L^1 )</th>
<th>( L^\infty )</th>
<th>( L^1_{\text{relative}} )</th>
<th>( L^\infty_{\text{relative}} )</th>
<th>( L^\infty ) rate</th>
<th>( L^1_{\text{relative}} ) rate</th>
<th>( L^\infty_{\text{relative}} ) rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.142981</td>
<td>0.005399</td>
<td>0.010939</td>
<td>0.008218</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>7.108541</td>
<td>0.002723</td>
<td>0.005527</td>
<td>0.004184</td>
<td>0.987587</td>
<td>0.984972</td>
<td>0.974070</td>
<td></td>
</tr>
<tr>
<td>6.988068</td>
<td>0.001365</td>
<td>0.002778</td>
<td>0.002135</td>
<td>0.996106</td>
<td>0.992381</td>
<td>0.970687</td>
<td></td>
</tr>
<tr>
<td>6.731139</td>
<td>0.000681</td>
<td>0.001393</td>
<td>0.001099</td>
<td>1.002336</td>
<td>0.996128</td>
<td>0.957895</td>
<td></td>
</tr>
<tr>
<td>6.248259</td>
<td>0.000339</td>
<td>0.000697</td>
<td>0.000572</td>
<td>1.008492</td>
<td>0.997907</td>
<td>0.941500</td>
<td></td>
</tr>
<tr>
<td>5.431081</td>
<td>0.000168</td>
<td>0.000349</td>
<td>0.000299</td>
<td>1.014720</td>
<td>0.998316</td>
<td>0.935868</td>
<td></td>
</tr>
</tbody>
</table>

Plots are included with a final time extended to \( t = 1 \).
Figure 3.3: Constant Maxwellian, \( M(v) = \frac{1}{2} \) with (Top Left) \( \varphi^0 \), (Top Right) \( \lambda \), (Middle Left) Reference Density, (Middle Right) Density with proposed method, (Bottom Left) \( f^e \) from IMEX-RK and (Bottom Right) \( M(v) e^{-\frac{\varphi^0+\epsilon \varphi^1}{\epsilon}} \)
Figure 3.4: Constant Maxwellian, $M(v) = \frac{1}{2}$ with (Top Left) $\varphi^0$, (Top Right) $\lambda$, (Middle Left) Reference Density, (Middle Right) Density with proposed method, (Bottom Left) $f^c$ from IMEX-RK and (Bottom Right) $M(v)e^{-\frac{\varphi^0 + \epsilon \varphi^1}{c}}$
Figure 3.5: Non-constant Maxwellian, \( M(v) = \frac{e^{-v^2}}{\sqrt{\pi}erf(1)} \) with (Top Left) \( \varphi^0 \), (Top Right) \( \lambda \), (Middle Left) Reference Density, (Middle Right) Density with proposed method, (Bottom Left) \( f^c \) from IMEX-RK and (Bottom Right) \( M(v)e^{-\frac{\varphi^0+\epsilon\phi^1}{\epsilon}} \)
Figure 3.6: Non-constant Maxwellian, \( M(v) = \frac{e^{-v^2}}{\sqrt{\pi} \sigma f'(1)} \) with (Top Left) \( \varphi^0 \), (Top Right) \( \lambda \), (Middle Left) Reference Density, (Middle Right) Density with proposed method, (Bottom Left) \( f^\epsilon \) from IMEX-RK and (Bottom Right) \( M(v)e^{-\frac{\varphi^0+\epsilon^2}{\epsilon}} \)
3.3 2D

Here we present numerical results in the 2D case. Gauss quadrature is treated dimension by dimension. We again consider a constant Maxwellian case and a non-constant Maxwellian case. We again see third order convergence of $\varphi^0$ and first order convergence to $\lambda$. The plots for $H$ are obtained by interpolating the original scattered data onto a uniform mesh using natural neighbor interpolation. It’s clear that $\varphi^0$ and $H$ are nonnegative and that $H$ is convex.

3.3.1 Computation of the effective Hamiltonian

In the case of testing the Hamiltonian in 2D, a reference Hamiltonian is computed using 32 Gaussian nodes and a spatial grid consisting of $1025 \times 1025$ points. The initial data given is $\varphi^0(t = 0, x, y) = 0.2(1 + \cos(x\pi) + \sin(y\pi))$.

Table 3.11: $L^\infty$ error of Hamiltonian based on number of Gaussian nodes and quadrature rule

<table>
<thead>
<tr>
<th>T</th>
<th>$\omega(v) = 1, M(v) = \frac{1}{4}$</th>
<th>$\omega(v) = M(v) = \frac{1}{4}$</th>
<th>$\omega(v) = 1, M(v) = \frac{1}{\pi \text{erf}(1)^2}$</th>
<th>$\omega(v) = M(v) = \frac{1}{\pi \text{erf}(1)^2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1.301736e-14</td>
<td>1.301736e-14</td>
<td>2.509274e-02</td>
<td>1.419004e-14</td>
</tr>
</tbody>
</table>

3.3.2 Computation of $\varphi^0$

3.3.2.1 Experiment 2.1

We present a 2D example demonstrating the accuracy of computing $\varphi^0(t, x)$.

We consider $M(u, v) = \frac{1}{4}$ and $M(u, v) = \frac{e^{-u^2-v^2}}{\pi \text{erf}(1)^2}$ and $\varphi^0(t = 0, x, y) = 0.2(1 + \cos(x\pi) + \sin(y\pi))$. Reference solutions for $\varphi^0$ are computed on a spatial grid of $513 \times 513$ with 16 abscissas/weights. The final time for checking convergence is $t = 0.625$ to ensure the solution is smooth.
Table 3.12: Error and convergence rates when $M(u,v) = \frac{1}{4}$ using Gauss-Legendre with $T = 12$

<table>
<thead>
<tr>
<th>N</th>
<th>$L^1$</th>
<th>$L^\infty$</th>
<th>$L^1$ - rate</th>
<th>$L^\infty$ - rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>4.747251e-04</td>
<td>3.689068e-04</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>33</td>
<td>5.55037e-05</td>
<td>5.250037e-05</td>
<td>3.095224</td>
<td>2.812857</td>
</tr>
<tr>
<td>65</td>
<td>1.713434e-06</td>
<td>1.205384e-06</td>
<td>5.018834</td>
<td>5.444762</td>
</tr>
<tr>
<td>129</td>
<td>7.802628e-08</td>
<td>4.819985e-08</td>
<td>4.456787</td>
<td>4.644321</td>
</tr>
<tr>
<td>257</td>
<td>7.549584e-09</td>
<td>4.621913e-09</td>
<td>3.369491</td>
<td>3.382467</td>
</tr>
</tbody>
</table>

Table 3.13: Error and convergence rates when $\omega(u,v) = M(u,v) = \frac{1}{4}$ using the Golub and Welsch algorithm with $T = 12$

<table>
<thead>
<tr>
<th>N</th>
<th>$L^1$</th>
<th>$L^\infty$</th>
<th>$L^1$ - rate</th>
<th>$L^\infty$ - rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>4.747251e-04</td>
<td>3.689068e-04</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>33</td>
<td>5.55037e-05</td>
<td>5.250037e-05</td>
<td>3.095224</td>
<td>2.812857</td>
</tr>
<tr>
<td>65</td>
<td>1.713434e-06</td>
<td>1.205384e-06</td>
<td>5.018834</td>
<td>5.444762</td>
</tr>
<tr>
<td>129</td>
<td>7.802628e-08</td>
<td>4.819985e-08</td>
<td>4.456787</td>
<td>4.644321</td>
</tr>
<tr>
<td>257</td>
<td>7.549583e-09</td>
<td>4.621912e-09</td>
<td>3.369491</td>
<td>3.382467</td>
</tr>
</tbody>
</table>

Table 3.14: Error and convergence rates when $M(u,v) = e^{-u^2-v^2}$ using Gauss-Legendre with $T = 12$

<table>
<thead>
<tr>
<th>N</th>
<th>$L^1$</th>
<th>$L^\infty$</th>
<th>$L^1$ - rate</th>
<th>$L^\infty$ - rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>8.957487e-05</td>
<td>7.048270e-05</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>33</td>
<td>5.684190e-06</td>
<td>4.098636e-06</td>
<td>3.978067</td>
<td>4.104054</td>
</tr>
<tr>
<td>65</td>
<td>1.654805e-07</td>
<td>1.046167e-07</td>
<td>5.102221</td>
<td>5.291958</td>
</tr>
<tr>
<td>129</td>
<td>1.305934e-08</td>
<td>8.052048e-09</td>
<td>3.663508</td>
<td>3.699614</td>
</tr>
<tr>
<td>257</td>
<td>1.390564e-09</td>
<td>8.559448e-10</td>
<td>3.231340</td>
<td>3.233766</td>
</tr>
</tbody>
</table>
Table 3.15: Error and convergence rates when \( \omega(u, v) = M(u, v) = \frac{e^{-u^2 - v^2}}{\pi \text{erf}(1)} \) using the Golub and Welsch algorithm with \( T = 12 \)

<table>
<thead>
<tr>
<th>N</th>
<th>( L^1 )</th>
<th>( L^\infty )</th>
<th>( L^1 ) - rate</th>
<th>( L^\infty ) - rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>8.957487e-05</td>
<td>7.048270e-05</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>33</td>
<td>5.684190e-06</td>
<td>4.098636e-06</td>
<td>3.978067</td>
<td>4.104054</td>
</tr>
<tr>
<td>65</td>
<td>1.654805e-07</td>
<td>1.046167e-07</td>
<td>5.102221</td>
<td>5.291958</td>
</tr>
<tr>
<td>129</td>
<td>1.305934e-08</td>
<td>8.052048e-09</td>
<td>3.663508</td>
<td>3.699614</td>
</tr>
<tr>
<td>257</td>
<td>1.390564e-09</td>
<td>8.559447e-10</td>
<td>3.231340</td>
<td>3.233766</td>
</tr>
</tbody>
</table>

In 2D we still achieve third order convergence in all cases.

3.3.2.2 2D Computational Time

Computational time for varying \( T_s \) is compared. Table 3.16 demonstrates the increased efficiency of fewer Gaussian nodes. Treating \( M(v) \) as the weight function is not only desirable for efficiency, but also for securing solutions. The experiments recorded in Table 3.16 were computed with a final time, \( t = 0.0625 \) and \( \varphi^0(t = 0, x, y) = 0.2(1 + \cos(x\pi) + \sin(y\pi)) \) on a grid of 129 \( \times \) 129 points. We experienced numerical breakdown of solutions using Gauss-Legendre quadrature in the case of the non-constant Maxwellian when not enough Gaussian nodes were considered. We again remark that computational time is highly subjective to the machine, language, and how efficiently the algorithm is implemented, yet it does demonstrate that as \( T \) increases, the increase in computational cost is noticeable.
Table 3.16: Computational time, in seconds, for varying $T$’s in 2D

<table>
<thead>
<tr>
<th>$T$</th>
<th>$\omega(v) = 1, M(v) = \frac{1}{4}$</th>
<th>$\omega(v) = M(v) = \frac{1}{4}$</th>
<th>$\omega(v) = 1, M(v) = \frac{1}{\pi \text{erf}(1)^2}$</th>
<th>$\omega(v) = M(v) = \frac{1}{\pi \text{erf}(1)^2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>105.2353</td>
<td>107.7222</td>
<td>/</td>
<td>101.9898</td>
</tr>
<tr>
<td>10</td>
<td>125.8303</td>
<td>129.8409</td>
<td>/</td>
<td>122.0378</td>
</tr>
<tr>
<td>12</td>
<td>150.9738</td>
<td>155.1284</td>
<td>145.1641</td>
<td>145.5515</td>
</tr>
<tr>
<td>14</td>
<td>180.2031</td>
<td>185.6468</td>
<td>172.9193</td>
<td>173.5452</td>
</tr>
<tr>
<td>16</td>
<td>215.6332</td>
<td>213.4121</td>
<td>204.4005</td>
<td>204.2738</td>
</tr>
</tbody>
</table>

3.3.3 Computation of $\varphi^1$

3.3.3.1 Experiment 2.2

We consider the following initial data: $\varphi^0(t = 0, x) = 0.025(2 + \sin(2y\pi) + \sin(2x\pi))$ and $\lambda(t = 0, x) = -(2+\cos(2y\pi)+\cos(2x\pi))$. We consider both $M(u, v) = \frac{1}{4}$ and $M(u, v) = \frac{e^{-u^2-v^2}}{\pi \text{erf}(1)^2}$.

To verify first order convergence of $\lambda(t, x)$, a reference solution is computed on a grid $513 \times 513$ points with final time, $t = 0.0625$, a time when $\lambda$ is still smooth.

Table 3.17: Error and convergence rates of $\lambda$ with $M(u, v) = \frac{1}{4}$

<table>
<thead>
<tr>
<th>$N$</th>
<th>$L^1$</th>
<th>$L^\infty$</th>
<th>$L^1$ - rate</th>
<th>$L^\infty$ - rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>1.649195</td>
<td>7.066633e-01</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>33</td>
<td>2.838478e-01</td>
<td>1.774828e-01</td>
<td>2.538573</td>
<td>1.993344</td>
</tr>
<tr>
<td>65</td>
<td>4.822334e-02</td>
<td>3.139357e-02</td>
<td>2.557314</td>
<td>2.499138</td>
</tr>
<tr>
<td>129</td>
<td>8.382973e-03</td>
<td>7.304457e-03</td>
<td>2.524198</td>
<td>2.103620</td>
</tr>
<tr>
<td>257</td>
<td>1.256525e-03</td>
<td>1.453841e-03</td>
<td>2.738023</td>
<td>2.328907</td>
</tr>
</tbody>
</table>
Table 3.18: Error and convergence rates of $\lambda$ with $M(u, v) = \frac{e^{-u^2-v^2}}{\pi \text{erf}(1)^2}$

<table>
<thead>
<tr>
<th>N</th>
<th>$L^1$</th>
<th>$L^\infty$</th>
<th>$L^1$ - rate</th>
<th>$L^\infty$ - rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>1.003860e00</td>
<td>4.375920e-01</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>33</td>
<td>1.740056e-01</td>
<td>1.145178e-01</td>
<td>2.528353</td>
<td>1.934015</td>
</tr>
<tr>
<td>65</td>
<td>2.944812e-02</td>
<td>1.875491e-02</td>
<td>2.562886</td>
<td>2.61023</td>
</tr>
<tr>
<td>129</td>
<td>5.127863e-03</td>
<td>4.400852e-03</td>
<td>2.521746</td>
<td>2.091414</td>
</tr>
<tr>
<td>257</td>
<td>7.944587e-04</td>
<td>9.343894e-04</td>
<td>2.690313</td>
<td>2.235687</td>
</tr>
</tbody>
</table>

We remark that the convergence rates are closer to second order accurate in this particular numerical test. This is because the data is smooth and not much time has passed. In theory, $\lambda(t, x)$ and thus $\varphi^1(t, x, v)$ will only be first order accurate with the methods discussed in this work.

Plots of $\lambda(t, x, v)$, $\varphi^0(t, x)$, and $H$ are included. The solutions are computed on a grid size of $65 \times 65 \times 12 \times 12 \in \mathbb{R}^2 \times \mathbb{V}^2$. 
Figure 3.7: Constant Maxwellian, $M(u, v) = \frac{1}{4}$ with (Left) $\lambda$ (Right) $\varphi^0$) and (Bottom) $H$. 
Figure 3.8: Constant Maxwellian, $M(u,v) = \frac{1}{4}$ with (Left) $\lambda$ (Right) $\varphi^0$ and (Bottom) $H$. 
Figure 3.9: Non-constant Maxwellian, $M(u, v) = e^{-u^2-v^2}/\pi \text{erf}(1)^2$ with (Left) $\lambda$ (Right) $\varphi^0$ and (Bottom) $H$. 
Figure 3.10: Non-constant Maxwellian, $M(u, v) = \frac{e^{-u^2 - v^2}}{\pi \text{erf}(1)^2}$ with (Left) $\lambda$ (Right) $\varphi^0$ and (Bottom) $H$.

### 3.3.4 Approximating $f^\epsilon$

Once $\varphi^0$ and $\varphi^1$ are calculated, we can transform back to the original density function.

#### 3.3.4.1 Experiment 2.3

We present a 2D example demonstrating the accuracy of computing $f^\epsilon(t, x, v)$ with a non-
constant Maxwellian. Consider the initial data given to be $\varphi^0_0(x, y) = 0.0125(2 + \sin(2x\pi) + \sin(2y\pi)), \lambda(t = 0, x, y) = -(2 + \cos(2x\pi) + \cos(2y\pi))$. The computations are done on a grid of $(\frac{2}{\epsilon} + 1) \times (\frac{2}{\epsilon} + 1) \times 12 \times 12 \in \mathbb{R}^2 \times \mathbb{V}^2$. We include errors measured at the final time, $t = 0.005$. 
Table 3.19: Error and convergence rates with $\varphi^0 = 0.0125(2 + \sin(2x\pi) + \sin(2y\pi))$ and $M(u, v) = \frac{e^{-u^2-v^2}}{\pi \text{erf}(1)}$

<table>
<thead>
<tr>
<th>$\epsilon$</th>
<th>$L^1$</th>
<th>$L^\infty$</th>
<th>$L^1_{\text{relative}}$</th>
<th>$L^\infty_{\text{relative}}$</th>
<th>$L^\infty$ rate</th>
<th>$L^1_{\text{relative}}$ rate</th>
<th>$L^\infty_{\text{relative}}$ rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\epsilon$</td>
<td>77.88022</td>
<td>3.19129e-03</td>
<td>1.42273e-02</td>
<td>4.41102e-02</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>$1/2\epsilon$</td>
<td>108.4939</td>
<td>1.96778e-03</td>
<td>1.27290e-02</td>
<td>3.88628e-02</td>
<td>0.697565</td>
<td>0.160544</td>
<td>0.182722</td>
</tr>
<tr>
<td>$1/4\epsilon$</td>
<td>134.1890</td>
<td>1.10941e-03</td>
<td>9.62984e-03</td>
<td>2.68418e-02</td>
<td>0.826770</td>
<td>0.402541</td>
<td>0.533910</td>
</tr>
<tr>
<td>$1/8\epsilon$</td>
<td>145.3053</td>
<td>5.61128e-04</td>
<td>5.83687e-03</td>
<td>1.50322e-02</td>
<td>0.983402</td>
<td>0.722316</td>
<td>0.836421</td>
</tr>
</tbody>
</table>

As $\epsilon$ is halved, the convergence rates approach first order demonstrating that the method is $O(\epsilon)$.

### 3.3.4.2 Experiment 2.4

We consider the initial data given to be $\varphi^0(x, y) = 0.25(2 + \sin(2x\pi) + \sin(2y\pi))$, $\lambda(t = 0, x, y) = -(2 + \cos(2x\pi) + \cos(2y\pi))$. Both the constant and non-constant Maxwellian cases are considered. We include plots of $\varphi^0$ and $\lambda$ as well as $\rho^e(t, x)$, the macroscopic density. Solutions are constructed on a grid of $101 \times 101 \times 8 \times 8 \in \mathbb{R}^2 \times \mathbb{V}^2$. 
Figure 3.11: Constant Maxwellian, $M(u,v) = \frac{1}{4}$ with (Top Left) $\varphi^0$, (Top Right) $\lambda$, (Bottom Left) Reference Density, (Bottom Right) Density with proposed method
Figure 3.12: Constant Maxwellian, $M(u,v) = \frac{1}{4}$ with (Top Left) $\phi^0$, (Top Right) $\lambda$, (Bottom Left) Reference Density, (Bottom Right) Density with proposed method
Figure 3.13: Non-constant Maxwellian, $M(u,v) = \frac{e^{-u^2-v^2}}{\pi \text{erf}(1)^2}$ with (Top Left) $\varphi^0$, (Top Right) $\lambda$, (Bottom Left) Reference Density, (Bottom Right) Density with proposed method
Figure 3.14: Non-constant Maxwellian, $M(u, v) = \frac{e^{-u^2-v^2}}{\pi \text{erf}(1)^2}$ with (Top Left) $\varphi^0$, (Top Right) $\lambda$, (Bottom Left) Reference Density, (Bottom Right) Density with proposed method
CHAPTER 4. SUMMARY AND DISCUSSION

4.1 Summary

In this thesis we have presented an asymptotic Hopf-Cole transformation based method for computing solutions to a kinetic equation with a BGK relaxation operator in the large scale hyperbolic limit. The density function of the kinetic equation with a BGK relaxation operator is transformed as the Hopf-Cole transformation and the phase function is then expanded in a power series about the Knudsen number. This approach is similar to moment closure methods, however we never make any assumptions about the density function which is required in a moment closure method. The phase function is approximated using the first two terms of the power series expansion. The leading term is the viscosity solution of a Hamilton-Jacobi equation in which the Hamiltonian is defined implicitly in an integral with respect to the velocity variables. The first order term is related to a transport equation. Both the Hamilton-Jacobi equation and transport equation are formulated in the physical space with necessary components defined as integrals with respect to the velocity variable. These integrals can be efficiently computed using Gauss quadrature, which is where the efficiency of the proposed method lies. Using well-established techniques for time-dependent Hamilton-Jacobi equations, the leading order term and the first order term of the power series expansion of the phase function can be computed. Using these to estimate the phase function we faithfully recover an estimate to the density function. The proposed method is independent of the velocity discretization and the Knudsen number.
4.2 Future Work

This method has been discussed in the context of a very simple BGK relaxation operator. Future works include extending the method to cases of more realistic Maxwellians with dependence on (t, x, v). Higher order terms will also be considered in the power series expansion of the phase function of the transformed density function to develop more accurate approximations to the density. For example, considering the second order term to obtain an approximation that is $O(\epsilon^2)$ accurate. Finally, there is difficulty in computing solutions in which $\varphi^0$ develops kinks. $\varphi^0$ is a viscosity solution but is not guaranteed to remain smooth. This is an area worthy of deeper thought and research regarding the proposed method.
BIBLIOGRAPHY


