A high-order discontinuous Galerkin finite element method for a quadrature-based moment-closure model

Erica Johnson

Iowa State University
A high-order discontinuous Galerkin finite element method for a quadrature-based moment-closure model

by

Erica Renee Johnson

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:
James Rossmanith, Major Professor
Pablo Raúl Stinga
Jue Yan

Iowa State University
Ames, Iowa
2017

Copyright © Erica Renee Johnson, 2017. All rights reserved.
TABLE OF CONTENTS

ABSTRACT ................................................................. iv

CHAPTER 1. HYPERBOLIC CONSERVATION LAWS ............... 1

1.1 Introduction ....................................................... 1

1.2 Examples of conservation laws ................................. 2

1.2.1 Advection equation (a linear scalar example) .......... 2

1.2.2 Wave equation (a linear system example) ............... 3

1.2.3 Burgers equation (a nonlinear scalar example) .......... 5

1.2.4 Shallow water equations (a nonlinear system example) . 7

1.3 Classifications of hyperbolic conservation laws ............ 10

1.4 Scope of this work ............................................... 11

CHAPTER 2. VLASOV EQUATION AND THE MOMENT-CLOSURE PROBLEM ................................................. 13

2.1 The moment-closure problem ................................... 13

2.2 Compressible Euler equations .................................. 14

2.3 Other closures ..................................................... 16

2.3.1 Grad-closure .................................................... 16

2.3.2 Maximum entropy closure ................................... 17

CHAPTER 3. QUADRATURE-BASED MOMENT-CLOSURE METHODS ................................................................. 18

3.1 Existing quadrature-based moment-closures .................. 18

3.1.1 Gaussian quadrature using a bi-delta distribution ........ 18

3.1.2 Quadrature using a bi-Gaussian distribution ............. 22

3.1.3 Quadrature using a bi-B-spline distribution .............. 23
3.2 A new approach - Gauss-Radau quadrature using a tri-delta distribution . . . . 25
3.2.1 Analysis of eigensystem .............................................. 27

CHAPTER 4. DISCONTINUOUS GALERKIN ................................. 33
4.1 Semi-discrete discontinuous Galerkin method .......................... 33
4.2 Total variation diminishing Runge-Kutta time-stepping ................. 34
4.3 Limiting .......................................................................... 35

CHAPTER 5. SOLVING THE TRI-DELTA SYSTEM WITH DISCONTINUOUS GALERKIN AND CONSTRAINT-PRESERVING TECHNIQUES 37
5.1 Constraint-preserving condition ............................................ 37
5.1.1 Constraint-preserving limiter ........................................... 40
5.2 Numerical results ................................................................ 42

CHAPTER 6. CONCLUSIONS .................................................... 47

BIBLIOGRAPHY ..................................................................... 48

APPENDIX A. EXCERPT FROM DOGPACK .............................. 51

APPENDIX B. PARAMETERS FOR FIGURES 5.1-5.4 ......................... 60
The Euler equations are a system of nonlinear partial differential equations that prescribe the evolution of mass density, velocity, and pressure of a gas in thermodynamic equilibrium. In order to extend the validity of the Euler equations beyond thermodynamic equilibrium, equations for higher moments must be added to the system. The core difficulty with expanding the Euler system is that every new moment evolution equation that is added requires knowledge of the next moment. This problem is known as the moment-closure problem. In this work we study a particular strategy for closing the moment hierarchy: quadrature-based moment-closures. In particular, we review existing approaches that close the moment hierarchy by assuming that the underlying distribution is the sum of two delta functions, two Gaussian distributions, or two B-splines. Next we develop a closure based on three delta functions (tri-delta), where one of the delta functions is located at a prescribed location. This leads to a Gauss-Radau-type quadrature rule. We derive exact formulas that relate the positions and weights of the three delta functions to the primitive variables: mass density, velocity, pressure, heat flux, and kurtosis. We also derive exact conditions that simultaneously guarantee that the underlying system of partial differential equations remain hyperbolic and that the inversion problem from primitive variables to Gauss-Radau quadrature weights and points is solvable. Furthermore, we prove that the region in solution space for which these conditions are satisfied is convex. Finally, we develop a high-order discontinuous Galerkin finite element method to solve this system with a moment-realizability limiter that guarantees that the numerical solution remains in this convex hyperbolic/moment-realizable region.
CHAPTER 1. HYPERBOLIC CONSERVATION LAWS

1.1 Introduction

A one dimensional conservation law is a partial differential equation (PDE) of the form

\[ q_t + f(q)_x = 0, \]  

(1.1)

where \( q(t, x) : \mathbb{R}^+ \times \mathbb{R} \rightarrow \mathbb{R}^m \) is the vector of conserved variables (e.g., mass, momentum, energy) and \( f(q) : \mathbb{R}^m \rightarrow \mathbb{R}^m \) is the flux function.

Starting from the form (1.1) written as

\[ q_t + \frac{\partial f}{\partial x} = 0, \]

(1.2)

and using the chain rule we obtain:

\[ q_t + A(q)q_x = 0, \]

(1.3)

where \( A(q) = \frac{\partial f}{\partial q} \) is known as the flux Jacobian. Form (1.3) of the PDE is known as the quasilinear form.

**Definition 1.1.1.** A conservation law of the form (1.1) is said to be hyperbolic if the flux Jacobian, \( A(q) = \frac{\partial f}{\partial q} \), is a matrix with only real eigenvalues and a complete set of eigenvectors.

The fact that the flux Jacobian has only real eigenvalues has an important physical consequence: each eigenvalue corresponds to a wave speed in the system, and the fact that they are real signifies that each wave propagates at a finite speed.

To figure out how the total amount of \( q \) in the domain \([x_1, x_2]\) evolves, we integrate (1.1) and apply the Fundamental Theorem of Calculus to get

\[ \frac{d}{dt} \int_{x_1}^{x_2} q(t, x) \, dx = f(q(x_1, t)) - f(q(x_2, t)). \]

(1.4)
This is known as the integral form of the conservation law \[1\]. The first term on the right hand side of (1.4) represents the flux of \(q\) into the domain at \(x_1\), the second term on the right hand side represents the flux of \(q\) out of the domain at \(x_2\), and the left hand side represents the total amount of \(q\) in the domain. An important and fundamental idea behind conservation laws is that the quantity \(q\) is neither created nor destroyed. The only way the quantity can change is from flux in or out of the boundary points.

1.2 Examples of conservation laws

In this section we give four examples of hyperbolic conservation laws of different types. The types discussed are linear scalar, linear systems, nonlinear scalar, and nonlinear systems.

1.2.1 Advection equation (a linear scalar example)

The simplest form of a hyperbolic conservation law is the advection equation, which can be written as

\[ q_t + u q_x = 0, \quad (1.5) \]

where \(u\) is the constant fluid velocity. Equation (1.5) models the advection of a passive tracer with concentration \(q\) in a fluid that is moving at constant velocity \(u\). In this case, the flux is \(f(q) = u q\), and the flux Jacobian is \(f'(q) = u\), which clearly has a real eigenvalue and is diagonalizable.

In order to illustrate what solutions of this equation look like, we show a smooth example in Figure 1.1 with initial condition

\[ q(t = 0, x) = e^{-12(x - \frac{1}{2})^2}, \quad (1.6) \]

where the wave is propagating with velocity \(u = 1\). This simulation was carried out using the DOGPACK [13] software package.
Figure 1.1: Solution of the advection equation (1.5) with wave speed $u = 1$ and initial condition (1.6). Panel (a) shows the initial condition; Panel (b) shows the solution at time $t = 0.8$.

1.2.2 Wave equation (a linear system example)

The advection equation is an example of a linear scalar hyperbolic equation. A hyperbolic partial differential equation that results in a linear hyperbolic system is the wave equation:

$$p_{tt} - c^2 p_{xx} = 0,$$

where $c > 0$ is a constant. If we define

$$q_1 = p_t \quad \text{and} \quad q_2 = -p_x,$$

then we have the equations

$$(q_1)_t + c^2(q_2)_x = 0,$$

$$(q_1)_x + (q_2)_x = 0,$$

which we can write as

$$q_t + Aq_x = 0,$$

where

$$A = \begin{bmatrix} 0 & c^2 \\ 1 & 0 \end{bmatrix}. $$
It follows that (1.12) has eigenvalues and eigenvectors of the form:

\[
\lambda_1 = -c, \quad \lambda_2 = c,
\]

\[
r_1 = \begin{bmatrix} -c \\ 1 \end{bmatrix}, \quad r_2 = \begin{bmatrix} c \\ 1 \end{bmatrix},
\]

and therefore \( A \) has the following eigenvalue decomposition

\[
A = R\Lambda R^{-1} = \begin{bmatrix} -c & c \\ 1 & 1 \end{bmatrix} \begin{bmatrix} -c & 0 \\ 0 & c \end{bmatrix} \frac{1}{2c} \begin{bmatrix} -1 & c \\ 1 & c \end{bmatrix}.
\] (1.14)

Since the eigenvalues are all real and we have a complete set of eigenvectors, this system is indeed hyperbolic. Using some manipulations we arrive at

\[
R^{-1}q_t + R^{-1}(RAR^{-1})q_x = 0 \quad \Rightarrow \quad R^{-1}q_t + \Lambda R^{-1}q_x = 0.
\] (1.15)

By defining \( w(t, x) = R^{-1}q(x, t) \), we can decouple the system into independent advection equations:

\[
 w_t^p + \lambda^p w_x^p = 0,
\] (1.16)

for each eigenvalue \( \lambda^p \). This form is convenient since the solution to a linear hyperbolic system then consists of linearly independent waves, each traveling with velocity \( \lambda^p \). In the wave equation case we obtain:

\[
 w_1^1 - cw_1^1 = 0 \text{ and } w_2^2 + cw_2^2 = 0.
\] (1.17)

A sample solution of the wave equation is shown in Figure 1.2 with initial conditions

\[
q_1(t = 0, x) = \cos^6 \left( \frac{5\pi}{4} (2x - 1) \right),
\] (1.18)

\[
q_2(t = 0, x) = 0.
\] (1.19)

We see that \( q_1 \) and \( q_2 \) each split into two different waves, with the left wave having a velocity of \( c = -1 \), and the right wave having a velocity of \( c = 1 \). This simulation was carried out using the DOGPACK [13] software package.
Figure 1.2: Solution of the wave equation (1.11)--(1.12) with wave speed $c = 1$ and initial conditions (1.18)--(1.19). Panels (a), (c), and (e) show $q_1$ at various times; Panels (b), (d), and (f) show $q_2$ at various times.

1.2.3 Burgers equation (a nonlinear scalar example)

The two previous examples were both examples of linear equations, which means that the flux function $f(q)$ depends linearly on the solution $q$. In this and the next subsection we
consider nonlinear examples. An example of a nonlinear scalar equation is Burgers equation:

\[ q_t + \left( \frac{1}{2} q^2 \right)_x = 0, \quad (1.20) \]

where \( f'(q) = q \). It follows that the quasilinear form of (1.20) is

\[ q_t + qq_x = 0, \quad (1.21) \]

which implies in smooth regimes that the solution is constant along characteristics traveling at speed \( q \) (see Leveque [1]).

Sample solutions of the Burger’s equation are shown in Figures 1.3 and 1.4. In Figure 1.3
the initial data is piecewise constant with left and right states on either side of \( x = 0 \): \( q_\ell = 1, q_r = -1/2 \). In this case the left state is larger than the right state, which results in compression and therefore a shock wave solution. In Figure 1.4 the initial data is again piecewise constant, but this time with the left and right values reversed from the previous example: \( q_\ell = -1/2, q_r = 1 \). In this case the left state is smaller than the right state, which results in the solution being pulled apart and therefore a rarefaction forms. These simulations were carried out using the DOGPACK [13] software package.

Figure 1.3: A Riemann problem for Burgers equation with piecewise constant initial data: \( q_\ell = 1, q_r = -1/2 \). The resulting solution is a shock propagating at speed \( s = (q_\ell + q_r)/2 = 1/4 \). Panel (a) shows the initial data; Panel (b) shows the solution at time \( t = 0.4 \).
Figure 1.4: A Riemann problem for Burgers equation with piecewise constant initial data: \( q_\ell = -1/2, \ q_r = 1 \). The resulting solution is a rarefaction. Panel (a) shows the initial data; Panel (b) shows the solution at time \( t = 0.4 \).

### 1.2.4 Shallow water equations (a nonlinear system example)

Consider the system of equations:

\[
\begin{bmatrix}
h \\
h u
\end{bmatrix}_t + \begin{bmatrix}
h u \\
h u^2 + \frac{1}{2} g h^2
\end{bmatrix}_x = 0,
\]

which models the dynamics of a thin fluid layer with height \( h(t, x) \) and depth-averaged velocity \( u(t, x) \) (see Leveque [1]). In the above equations, \( g > 0 \) is the acceleration due to gravity. If we define the conserved variables

\[
q_1 = h, \quad q_2 = hu,
\]

then we have a flux function of the form

\[
f(q) = \begin{bmatrix}
h u \\
h u^2 + \frac{1}{2} g h^2
\end{bmatrix} = \begin{bmatrix}
q_2 \\
\frac{q_2^2}{q_1} + \frac{1}{2} g q_1^2
\end{bmatrix}.
\]

From the flux function we can form the flux Jacobian:

\[
A(q) := \frac{\partial f}{\partial q}(q) = \begin{bmatrix}
0 & 1 \\
-u^2 + gh & 2u
\end{bmatrix}.
\]
It follows that (1.26) has eigenvalues and eigenvectors of the form

\[
\lambda_1 = u - \sqrt{gh}, \quad \lambda_2 = u + \sqrt{gh},
\]

where \( h > 0 \) guarantees that (1.22) is a system of hyperbolic conservation laws. The eigenvalues \( \lambda_1 = u - \sqrt{gh} \) and \( \lambda_2 = u + \sqrt{gh} \) represent the velocities of the surface gravity waves.

A sample Riemann (dam-break) problem solution is shown below in Figure 1.5 with piece-wise constant initial data: \((h, u)_\ell = (3, 0)\) and \((h, u)_r = (1, 0)\), on either side of \( x = 0.5 \). We see the solution split into two waves: the left-going wave is a rarefaction and the right-going wave a shock. This simulation was carried out using the DOGPACK [13] software package.
Figure 1.5: A Riemann (dambreak) problem for the shallow water equations with piecewise constant initial data: \((h, u)_{\ell} = (3, 0)\) and \((h, u)_{r} = (1, 0)\). Panels (a), (c), and (e) show the height at various times; Panels (b), (d), and (f) show the velocity at various times.
1.3 Classifications of hyperbolic conservation laws

In this section we briefly categorize various types of hyperbolic conservation laws. These distinctions become important in subsequent chapters.

**Definition 1.3.1.** We say that if \( A(q) \) in (1.3) is symmetric, then the system is symmetric hyperbolic.

**Definition 1.3.2.** If the eigenvalues of \( A(q) \) in (1.3) are real and distinct, then we classify the system as strictly hyperbolic, since it follows that the eigenvectors must also be linearly independent. A simple example of a strictly hyperbolic system are the shallow water equations 1.22 for all \( h > 0 \).

**Definition 1.3.3.** If the eigenvalues of \( A(q) \) in (1.3) are real but not distinct, and \( A(q) \) has a complete set of eigenvectors, then we have a non-strictly hyperbolic system.

**Definition 1.3.4.** If \( A(q) \) in (1.3) has real eigenvalues, but \( A(q) \) is not diagonalizable (i.e., does not have a complete set of eigenvectors), then the system is called weakly hyperbolic.

By computing the gradient of each eigenvalue and computing the inner product of this gradient and the corresponding eigenvector, we can check if the waves are genuinely nonlinear or linearly degenerate.

**Definition 1.3.5.** The \( k^{th} \) wave is linearly degenerate if

\[
\nabla_q \lambda_k \cdot r_k \equiv 0. \tag{1.28}
\]

This condition implies that \( \lambda_k \) is constant along each integral curve of \( r_k \) or in other words the wave translates with a constant speed without changing shape (see Leveque [1]).

Note that if we have a linear system, then it follows \( \lambda_k \) is independent of \( q \), which implies each wave in the system is linearly degenerate (because \( \nabla_q \lambda_k \equiv 0 \)).

**Definition 1.3.6.** The \( k^{th} \) wave is genuinely nonlinear if

\[
\nabla_q \lambda_k \cdot r_k \neq 0. \tag{1.29}
\]
A further discussion of these concepts, along with detailed examples can be found in Leveque [1]. In Chapter 3 we will analyze the hyperbolicity of a quadrature-based moment-closure system and provide a detailed analysis of the linear degeneracy and genuine nonlinearity of waves in the system.

1.4 Scope of this work

The Euler equations are a system of nonlinear partial differential equations that prescribe the evolution of mass density, velocity, and pressure of a gas in thermodynamic equilibrium. In order to extend the validity of the Euler equations beyond thermodynamic equilibrium, equations for higher moments must be added to the system. In this work we study a particular strategy for closing the moment hierarchy: quadrature-based moment-closures. In particular, we review existing approaches that close the moment hierarchy by assuming that the underlying distribution is the sum of two delta functions by Chalons et al. [8], two Gaussian distributions by Chalons et al. [9], or two B-splines by Cheng and Rossmanith [4]. Next we develop a closure based on three delta functions (tri-delta), where one of the delta functions is located at a prescribed location. Using the distribution function of the form

\[ f \sim f^* = \omega_1 \delta(v - \mu_1) + \omega_2 \delta(v - u) + \omega_3 \delta(v - \mu_3), \]

we show in this work (see Chapter 3) that we arrive at the following system of nonlinear hyperbolic partial differential equations:

\[
\begin{bmatrix}
q_1 \\
q_2 \\
q_3 \\
q_4 \\
q_5
\end{bmatrix}
+ \begin{bmatrix}
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 \\
\frac{\partial f_5}{\partial q_1} & \frac{\partial f_5}{\partial q_2} & \frac{\partial f_5}{\partial q_3} & \frac{\partial f_5}{\partial q_4} & \frac{\partial f_5}{\partial q_5}
\end{bmatrix}
\begin{bmatrix}
q_1 \\
q_2 \\
q_3 \\
q_4 \\
q_5
\end{bmatrix}_x = 0,
\]
which in primitive form can be written as

\[
\begin{bmatrix}
\rho \\
u \\
p \\
q \\
r
\end{bmatrix}
+ \begin{bmatrix}
u & \rho & 0 & 0 & 0 \\
0 & u & \frac{1}{\rho} & 0 & 0 \\
0 & 3p & u & 1 & 0 \\
0 & 4q & \frac{-3p}{\rho} & u & 1 \\
0 & 5r & \frac{2q^3}{\rho^3} - \frac{2qr}{\rho^2} - \frac{4q}{\rho} & \frac{-3q^2 + 2pr}{\rho^2} & \frac{2q}{\rho} + u
\end{bmatrix}
\begin{bmatrix}
\rho \\
u \\
p \\
q \\
r
\end{bmatrix}
= 0.
\] (1.31)

In this work we study the hyperbolic structure of this system. We derive conditions for hyperbolicity and establish that the constraint set over which these conditions are satisfied is convex. We derive analytic expressions for all the wave speeds and establish the linear independence of the eigenvectors. These conditions simultaneously guarantee that the underlying system of partial differential equations remain hyperbolic and that the inversion problem from primitive variables to Gauss-Radau quadrature weights and points is solvable. Finally, we develop a high-order discontinuous Galerkin finite element method to solve this system with a moment-realizability limiter that guarantees that the numerical solution remains in the convex hyperbolic/moment-realizable region.

A brief outline of this thesis is discussed next. In Chapter 2 we introduce the Vlasov equation and discuss the moment-closure problem. Chapter 3 details several previous results known as the bi-delta (Chalons et al. [8]), bi-Gaussian (Chalons et al. [9]), and bi-B-spline (Cheng and Rossmanith [4]) approaches, which give motivation for the tri-delta method developed in this work. After reviewing existing quadrature-based moment-closure methods we also fully analyze the tri-delta approach in Chapter 3. Chapter 4 will give a brief overview of the discontinuous Galerkin method used to solve the system (1.30). Finally, in Chapter 5 we develop a high-order discontinuous Galerkin finite element method with a constraint-preserving limiter that guarantees that the discrete solution remains in the solution regime where (1.30) is strictly hyperbolic. We implement this method in the DOGPACK software package [13] and test it on a problem with piecewise constant initial data (i.e., the Riemann or shock tube problem).
CHAPTER 2. VLASOV EQUATION AND THE MOMENT-CLOSURE PROBLEM

In this chapter we describe the moment-closure problem, derive the compressible Euler equations, and describe two commonly used moment-closure techniques: Grad’s moment-closure [24] with a globally hyperbolic correction by Cai et al. [26] and the maximum entropy closure of Levermore [23].

2.1 The moment-closure problem

The conservative form of the 1D1V Vlasov equation or collisionless Botzmann equation is given by

\[ f_t + vf_x = 0, \]

where \( f = f(t, x, v) \) is the probability distribution function that gives the probability of finding a particle at time \( t \), with position \( x \), and velocity \( v \). By integrating \( v^k f \) with respect to \( v \), we can define the mass density as well as higher moments:

\[
\int_{-\infty}^{\infty} v^k f dv = \begin{cases} 
\rho & k = 0 \quad \text{(mass density)}, \\
\rho u & k = 1 \quad \text{(momentum density)}, \\
\mathcal{E} = \rho u^2 + p & k = 2 \quad \text{(energy density)}, \\
\mathcal{F} = \rho u^3 + 3pu + q & k = 3, \\
\mathcal{G} = \rho u^4 + 6pu^2 + 4qu + r & k = 4.
\end{cases}
\]

From the definition of the momentum density, \( \rho u \), we can define the macroscopic velocity:

\[
u = \frac{1}{\rho} \int_{-\infty}^{\infty} v f dv.
\]

We can also define the thermal pressure:

\[ p = \int_{-\infty}^{\infty} (v - u)^2 f dv, \]  

(2.4)

the heat flux:

\[ q = \int_{-\infty}^{\infty} (v - u)^3 f dv, \]  

(2.5)

and Kurtosis:

\[ r = \int_{-\infty}^{\infty} (v - u)^4 f dv. \]  

(2.6)

In order to derive the evolution equations for the macroscopic fluid variables such as mass density, momentum density, and energy density, we multiply (2.1) by \( v^p \) for \( p = 1, 2, \ldots \) and integrate in \( v \):

\[ \left( \int_{-\infty}^{\infty} v^p f dv \right)_t + \left( \int_{-\infty}^{\infty} v^{p+1} f dv \right)_x = 0. \]  

(2.7)

We can write this as

\[ (M)_{p,t} + (M_{p+1})_x = 0, \]  

(2.8)

where \( M_p \) is referred to as the \( p \)th moment of the distribution function. From this equation we see that in order to evolve the \( p \)th moment we would need to know the \((p + 1)\) moment. This means that we have an infinite hierarchy, which is often called the moment-closure problem. In order to obtain a closure, additional assumptions must be made on \( f \).

### 2.2 Compressible Euler equations

If you assume that the gas is in thermodynamic equilibrium, then \( f \) is a Maxwellian distribution

\[ f \sim f^* = \frac{\rho}{\sqrt{2\pi p}} \exp \left( \frac{p}{2\rho} (v - u)^2 \right), \]  

(2.9)

where

\[ \int_{-\infty}^{\infty} v^k f^* dv = \frac{\rho}{\sqrt{2\pi p}} \int_{-\infty}^{\infty} v^k \exp \left( \frac{p}{2\rho} (v - u)^2 \right) dv = \begin{cases} \rho & k = 0, \\ \rho u & k = 1, \\ \rho u^2 + p & k = 2, \\ \rho u^3 + 3pu & k = 3. \end{cases} \]  

(2.10)
Under this assumption, the first three fluid equations become a closed system:

\[
\begin{bmatrix}
\rho \\
\rho u \\
\rho u^2 + p
\end{bmatrix}_t + \begin{bmatrix}
\rho u \\
\rho u^2 + p \\
\rho u^3 + 3pu
\end{bmatrix}_x = 0,
\] (2.11)

since the heat flux, \( q \), is now zero:

\[
q = \int_{-\infty}^{\infty} (v - u)^3 f^* dv = 0.
\] (2.12)

System (2.11) is known as the compressible Euler equations. We can obtain the flux Jacobian in terms of the conserved variables:

\[
A = \frac{\partial f}{\partial q} = \begin{bmatrix}
0 & 1 & 0 \\
0 & 0 & 1 \\
\frac{4q_3^2 - 3q_1 q_2 q_3}{q_1^3} & \frac{3(-2q_2^2 + q_1 q_3)}{q_1^2} & \frac{3q_2}{q_1} \\
\end{bmatrix},
\] (2.13)

which results in the following system of equations (written here in quasilinear form):

\[
\begin{bmatrix}
q_1 \\
q_2 \\
q_3
\end{bmatrix}_t + \begin{bmatrix}
0 & 1 & 0 \\
0 & 0 & 1 \\
\frac{4q_3^2 - 3q_1 q_2 q_3}{q_1^3} & \frac{3(-2q_2^2 + q_1 q_3)}{q_1^2} & \frac{3q_2}{q_1}
\end{bmatrix}\begin{bmatrix}
q_1 \\
q_2 \\
q_3
\end{bmatrix}_x = 0.
\] (2.14)

Notice that system (1.11) can be written as

\[
\frac{\partial q}{\partial P} \frac{\partial P}{\partial t} + \frac{\partial f}{\partial P} \frac{\partial P}{\partial x} = 0,
\] (2.15)

where \( q \) are the conserved variables, \( q = (\rho, \rho u, \mathcal{E}) \), and \( P \) are the primitive variables \( P = (\rho, u, p) \). Therefore, an alternative form to the flux Jacobian in terms of primitive variables is

\[
A_{\text{prim}} := \left( \frac{\partial q}{\partial P} \right)^{-1} \frac{\partial f}{\partial P} = \begin{bmatrix}
u & \rho & 0 \\
0 & u & \frac{1}{\rho} \\
0 & 3p & u
\end{bmatrix}.
\] (2.16)

Therefore system (2.14) can also be written as

\[
\begin{bmatrix}
\rho \\
u \\
p
\end{bmatrix}_t + \begin{bmatrix}
u & \rho & 0 \\
0 & u & \frac{1}{\rho} \\
0 & 3p & u
\end{bmatrix}\begin{bmatrix}
\rho \\
u \\
p
\end{bmatrix}_x = 0.
\] (2.17)
The eigenvalues and eigenvectors of (2.16) are

\[
\begin{align*}
\lambda_1 &= u - c, & r_1 &= \begin{bmatrix} \frac{\rho}{c}, 1, -\rho c \end{bmatrix}^T, \\
\lambda_2 &= u, & r_2 &= \begin{bmatrix} 1, 0, 0 \end{bmatrix}^T, \\
\lambda_3 &= u + c, & r_3 &= \begin{bmatrix} \frac{\rho}{c}, 1, \rho c \end{bmatrix}^T,
\end{align*}
\]

(2.18)

where \( c = \sqrt{\frac{3\rho}{\rho}} \) is the speed of sound of the gas. In the compressible Euler case there are three waves, with each speed represented by one eigenvalue. It is noted that these equations are hyperbolic when \( p > 0 \) and \( \rho > 0 \), since the system will result in three distinct real eigenvalues.

We can determine the linear degeneracy of the waves by doing the following computation

\[
\begin{align*}
\nabla_q \lambda_1 \cdot r_1 &= 2 \quad \text{(sound wave 1)}, \\
\nabla_q \lambda_2 \cdot r_2 &= 0 \quad \text{(entropy wave)}, \\
\nabla_q \lambda_3 \cdot r_3 &= 2 \quad \text{(sound wave 2)},
\end{align*}
\]

(2.19) (2.20) (2.21)

implying that the first and third waves are genuinely nonlinear, while the second is linearly degenerate.

### 2.3 Other closures

In this section we briefly discuss the history of moment-closure methods, which gives motivation for the methods seen in Chapter 3.

#### 2.3.1 Grad-closure

Grad’s moment-closure [24] assumes a distribution on \( f \) of the form

\[
f \sim f^* = (1 + \alpha \cdot \Phi(v)) \frac{\rho}{\sqrt{2\pi\rho^2}} \exp \left( \frac{p}{2\rho} (v - u)^2 \right),
\]

(2.22)

where \( \Phi(v) = [1, v, v^2, \ldots]^T \) and \( \alpha \) are expansion coefficients. This was the first approach to give a systematic extension of the fluid equations beyond thermodynamic equilibrium. This system is hyperbolic near thermodynamic equilibrium, but is not globally hyperbolic. This issue was solved by Cai et al. [26] using a correction that guarantees the system to be globally hyperbolic.
2.3.2 Maximum entropy closure

Another important approach for obtaining moment-closures is maximum entropy method of Levermore [23], in which a distribution of the following form is assumed:

\[ f \sim f^* = e^{\alpha \cdot \Phi(v)}, \quad \text{where} \quad \Phi(v) = [1, v, v^2, \ldots]^T. \quad (2.23) \]

It follows that we can write (2.1) as

\[
\left( \int_{-\infty}^{\infty} \Phi e^{\alpha \cdot \Phi(v)} dv \right)_t + \left( \int_{-\infty}^{\infty} v \Phi e^{\alpha \cdot \Phi(v)} dv \right)_x = 0,
\]

which implies that we have a system of the form (1.1). By defining the entropy to be

\[
\eta(\alpha) := \int_{-\infty}^{\infty} e^{\alpha \cdot \Phi(v)} dv \implies \eta_\alpha = \int_{-\infty}^{\infty} \Phi e^{\alpha \cdot \Phi(v)} dv = q,
\]

and the entropy flux to be

\[
\psi(\alpha) := \int_{-\infty}^{\infty} v e^{\alpha \cdot \Phi(v)} dv \implies \psi_\alpha = \int_{-\infty}^{\infty} v \Phi e^{\alpha \cdot \Phi(v)} dv = F,
\]

we see that

\[
q_\alpha \alpha_t + F_\alpha \alpha_x = 0 \implies \eta_{\alpha \alpha} \alpha_t + \psi_{\alpha \alpha} \alpha_x = 0.
\]

Levermore [23] proved that \( \eta_{\alpha \alpha} \) is a positive semi-definite matrix, and that \( \psi_{\alpha \alpha} \) is symmetric, implying that the system (2.24) is hyperbolic. The difficulty with this approach is that the moment-inversion, is non-trivial (i.e., converting between the \( \alpha \)'s and the primitive variables \( \rho, u, p, q, r, \ldots \)).
CHAPTER 3. QUADRATURE-BASED MOMENT-CLOSURE METHODS

In this chapter we introduce the tri-delta quadrature-based moment-closure that is the central object of interest in this thesis. We start with a related system: the bi-delta approach of Chalons et al. [8] for quadrature-based moment-closure problem, as well as two other methods: the bi-Gaussian approximation of Chalons et al. [9] and the bi-B-spline approximation of Cheng and Rossmanith [4]. We then show how to modify these approaches to obtain the tri-delta quadrature-based moment-closure. Finally, we analyze the resulting equations in detail, showing hyperbolicity and moment-realizability, and obtaining analytic expressions for the eigenvalues and eigenvectors of the appropriate flux Jacobian.

3.1 Existing quadrature-based moment-closures

3.1.1 Gaussian quadrature using a bi-delta distribution

In the moment-closure problem described in Chapter 1, an assumption on \( f(t, x, v) \) must be made in order to result in a closed set of fluid equations. In this section we review the consequences of assuming a bi-delta distribution of the form:

\[
 f \sim f^* = \omega_1 \delta(v - \mu_1) + \omega_2 \delta(v - \mu_2),
\]  

(3.1)

where \( \delta \) is the Dirac delta function, \( \mu_1 \) and \( \mu_2 \) are the locations of the two delta functions in velocity, and \( \omega_1 \) and \( \omega_2 \) are the weights associated to each delta function. In order to obtain a closure, we must solve the moment inversion problem: find \( (\mu_1, \mu_2, \omega_1, \omega_2) \) in terms of \( (\rho, u, p, q) \). The key realization of quadrature-based moment-closure methods is that finding \( (\mu_1, \mu_2, \omega_1, \omega_2) \) is equivalent to deriving a Gaussian quadrature rule. Therefore to obtain suitable weights, \( \omega_k \),
and quadrature points, $\mu_k$, we need to find a set of polynomials that are orthogonal with respect to the inner product

$$\langle P_n(v), P_m(v) \rangle = \int_{-\infty}^{\infty} P_n(v)P_m(v)f^*(v)dv = 0. \tag{3.2}$$

Since Gaussian quadrature is given by

$$\int_{-\infty}^{\infty} g(v)f^*(v)dv \approx \sum_{k=1}^{M} \omega_k g(\mu_k), \tag{3.3}$$

it follows that the maximum degree of precision is $2M-1$, where $M = 2$ in this current case, and therefore the maximum degree of precision is 3. This implies that our quadrature will integrate any cubic polynomial and lower exactly. Our polynomial must be exact for $g(v) = 1, v, v^2, v^3$.

This results in the following system:

$$M_0 = \int_{-\infty}^{\infty} f^*(v)dv = \omega_1 + \omega_2 = \rho, \tag{3.4}$$

$$M_1 = \int_{-\infty}^{\infty} vf^*(v)dv = \omega_1 \mu_1 + \omega_2 \mu_2 = \rho u, \tag{3.5}$$

$$M_2 = \int_{-\infty}^{\infty} v^2 f^*(v)dv = \omega_1 \mu_1^2 + \omega_2 \mu_2^2 = \rho u^2 + p, \tag{3.6}$$

$$M_3 = \int_{-\infty}^{\infty} v^3 f^*(v)dv = \omega_1 \mu_1^3 + \omega_2 \mu_2^3 = \rho u^3 + 3pu + q. \tag{3.7}$$

It follows that we have the system

$$\begin{bmatrix} \rho \\ \rho u \\ \rho u^2 + p \\ \rho u^3 + 3pu + q \end{bmatrix} + \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho u^3 + 3pu + q \\ M_4 \end{bmatrix} = 0, \tag{3.8}$$

where the missing moment, $M_4$ will be replaced by

$$M_4^* := \int_{-\infty}^{\infty} v^4 f^*(v)dv = \omega_1 \mu_1^4 + \omega_2 \mu_2^4, \tag{3.9}$$

after we have solved (3.4)–(3.7) for the quadrature points and weights.

Starting from a polynomial of degree 0, $P_0(v) = 1$, and a polynomial of degree 1, $P_1(v) =$
\( v + b \), we will force the orthogonality condition:

\[
\int_{-\infty}^{\infty} P_0(v)P_1(v)f^*(v)dv = \int_{-\infty}^{\infty} (v + b)f^*(v)dv = 0, \tag{3.10}
\]

\[
= \int_{-\infty}^{\infty} vf^*(v)dv + b \int_{-\infty}^{\infty} f^*(v)dv = 0,
\]

\[
= pu + bp = 0,
\]

by using (2.2). From this we see that \( b = -u \) and \( P_1(v) = v - u \). Next we define \( P_2(v) = v^2 + c(v - u) + d \), and we must force two orthogonality conditions. The first condition is

\[
\int_{-\infty}^{\infty} P_0(v)P_2(v)f^*(v)dv = \int_{-\infty}^{\infty} (v^2 + c(v - u) + d)f^*(v)dv = 0, \tag{3.11}
\]

\[
= \int_{-\infty}^{\infty} v^2 f^*(v)dv + c \int_{-\infty}^{\infty} (v - u)f^*(v)dv + d \int_{-\infty}^{\infty} f^*(v)dv = 0,
\]

\[
= (pu^2 + p) + 0 + \rho d = 0,
\]

where

\[
c \int_{-\infty}^{\infty} (v - u)f^*(v)dv = 0. \tag{3.12}
\]

From this we can find that \( d = u^2 - \frac{p}{\rho} \). The second orthogonality condition is

\[
\int_{-\infty}^{\infty} P_1(v)P_2(v)f^*(v)dv = \int_{-\infty}^{\infty} (v - u)(v^2 + c(v - u) + d)f^*(v)dv = 0, \tag{3.13}
\]

\[
= \int_{-\infty}^{\infty} (v - u)v^2 f^*(v)dv + c \int_{-\infty}^{\infty} (v - u)^2 f^*(v)dv + d \int_{-\infty}^{\infty} (v - u)f^*(v)dv = 0,
\]

\[
= \int_{-\infty}^{\infty} v^3 f^*(v)dv - u \int_{-\infty}^{\infty} v^2 f^*(v)dv + c \int_{-\infty}^{\infty} (v - u)^2 f^*(v)dv + 0 = 0,
\]

\[
= (pu^3 + 3pu + q) - u(\rho u^2 + p) + cp = 0.
\]

From this we find that \( c = \frac{2pu + q}{-\rho} \); plugging in \( c \) and \( d \) into our orthogonal polynomial of degree 2, we obtain

\[
P_2(v) = v^2 + u^2 - 2uv - \frac{p}{\rho} + \frac{qu - qv}{\rho}.	ag{3.14}
\]

This polynomial has the roots

\[
\mu_1 = u + \frac{q}{2\rho} + \sqrt{\frac{p}{\rho} + \left(\frac{q}{2\rho}\right)^2}, \quad \mu_2 = u + \frac{q}{2\rho} - \sqrt{\frac{p}{\rho} + \left(\frac{q}{2\rho}\right)^2}, \tag{3.15}
\]

and by plugging the roots into (3.4)-(3.7) we find that

\[
\omega_1 = \frac{\rho}{2} + \frac{\rho^2 q}{2\sqrt{\rho^2 q^2 + 4\rho p^3}}, \quad \omega_2 = \frac{\rho}{2} - \frac{\rho^2 q}{2\sqrt{\rho^2 q^2 + 4\rho p^3}}. \tag{3.16}
\]
Therefore we can rewrite (3.8) in closed form by replacing $M_4$ with

$$M_4^* = \rho u^4 + 6pu^2 + 4qu + \frac{q^2}{p} + \frac{p^2}{\rho}.$$  

(3.17)

The system in terms of the conserved variables is

$$
\begin{bmatrix}
q_1 \\
q_2 \\
q_3 \\
q_4
\end{bmatrix}_t + 
\begin{bmatrix}
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\frac{\partial M_4^*}{\partial q_1} & \frac{\partial M_4^*}{\partial q_2} & \frac{\partial M_4^*}{\partial q_3} & \frac{\partial M_4^*}{\partial q_4}
\end{bmatrix}
\begin{bmatrix}
q_1 \\
q_2 \\
q_3 \\
q_4
\end{bmatrix} = 0,
$$  

(3.18)

where

$$
\frac{\partial M_4^*}{\partial q_1} = -\frac{(ppu^2 + qpu - p^2)^2}{p^2} 
$$  

(3.19)

$$
\frac{\partial M_4^*}{\partial q_2} = 2(q + 2pu)(ppu^2 + qpu - p^2) 
$$  

(3.20)

$$
\frac{\partial M_4^*}{\partial q_3} = -\frac{q^2}{p^2} + \frac{2p}{\rho} - \frac{6qu}{p} - 6u^2 
$$  

(3.21)

$$
\frac{\partial M_4^*}{\partial q_4} = \frac{2q}{p} + 4u, 
$$  

(3.22)

and in terms of the primitive variables is

$$
\begin{bmatrix}
\rho \\
u \\
p \\
q
\end{bmatrix}_t + 
\begin{bmatrix}
u & \rho & 0 & 0 \\
0 & u & \frac{1}{\rho} & 0 \\
0 & 3p & u & 1 \\
-\frac{p^2}{p^2} & 4q & -\frac{q^2}{p^2} & \frac{2q}{p} + u
\end{bmatrix}
\begin{bmatrix}
\rho \\
u \\
p \\
q
\end{bmatrix} = 0.
$$  

(3.23)

Computing the eigenvalues and eigenvectors of the flux Jacobian, $A(q) = \frac{\partial f}{\partial q}$, reveals that the system is weakly hyperbolic since

$$
\lambda_1 = \lambda_2 = \mu_1, \quad r_1 = r_2 = \begin{bmatrix} 1, \mu_1, \mu_1^2, \mu_1^3 \end{bmatrix}_T, 
$$  

(3.24)

$$
\lambda_3 = \lambda_4 = \mu_2, \quad r_3 = r_4 = \begin{bmatrix} 1, \mu_2, \mu_2^2, \mu_2^3 \end{bmatrix}_T. 
$$

It is also noted that each of these waves is linearly degenerate since

$$
\nabla_q \lambda_1 \cdot r_1 = 0, \quad \nabla_q \lambda_2 \cdot r_2 = 0, 
$$  

$$
\nabla_q \lambda_3 \cdot r_3 = 0, \quad \nabla_q \lambda_4 \cdot r_4 = 0. 
$$  

(3.25)
For a detailed analysis of this system see Chalons et al. [8]. This gives us motivation to find an alternative method to solve the quadrature-based moment-closure problem that will result in a complete set of eigenvectors.

### 3.1.2 Quadrature using a bi-Gaussian distribution

In Chalons et al. [8], the following assumption on \( f \) was made:

\[
f \sim f^* = \frac{\omega_1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(v - \mu_1)^2}{2\sigma}\right) + \frac{\omega_2}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(v - \mu_2)^2}{2\sigma}\right),
\]

where \( \sigma \) is a width parameter. If \( \sigma \to 0 \) we recover the bi-delta distribution (3.1). This results in the following equations

\[
M_0 = \omega_1 + \omega_2 = \rho,
\]

(3.27)

\[
M_1 = \omega_1 \mu_1 + \omega_2 \mu_2 = \rho u,
\]

(3.28)

\[
M_2 = \omega_1 \mu_1^2 + \omega_2 \mu_2^2 = \rho u^2 + \alpha p,
\]

(3.29)

\[
M_3 = \omega_1 \mu_1^3 + \omega_2 \mu_2^3 = \rho u^3 + 3\alpha pu + q,
\]

(3.30)

\[
M_4 = \omega_1 \mu_1^4 + \omega_2 \mu_2^4 = \rho u^4 + 6\alpha pu^2 + 4qu + \frac{3p^2(\alpha^2 - 1)}{\rho},
\]

(3.31)

where

\[
\sigma = \frac{p}{\rho}(1 - \alpha).
\]

(3.32)

Note that if \( \alpha = 1 \) then we have the bi-delta system (3.8). It follows that we have the following system:

\[
\begin{pmatrix}
\rho \\
\rho u \\
\rho u^2 + \alpha p \\
\rho u^3 + 3\alpha pu + q \\
\rho u^4 + 6\alpha pu^2 + 4qu + \frac{3p^2(\alpha^2 - 1)}{\rho}
\end{pmatrix} + \begin{pmatrix}
\rho u \\
\rho u^2 + \alpha p \\
\rho u^3 + 3\alpha pu + q \\
\rho u^4 + 6\alpha pu^2 + 4qu + \frac{3p^2(\alpha^2 - 1)}{\rho} \\
M_5
\end{pmatrix} = 0.
\]

(3.33)

In Chalons et al. [8], the system is closed by solving (3.27)–(3.31) and replacing \( M_5 \) with

\[
M_5^* = \rho u^5 + 10\rho u^3 + \frac{15p^2u}{\rho} + \alpha \left( \frac{q^3}{p^2} + \frac{5q^2u}{p} + 10qu + \frac{10pq}{\rho} \right) - \frac{2p\alpha}{\rho} (4\tilde{q} + 5pu),
\]

(3.34)
where $\tilde{q} = q/\alpha$ (we set $\tilde{q} = 0$ if $\alpha = 0$). The system (3.33) with the flux Jacobian has the form

\[
\begin{bmatrix}
q_1 \\
q_2 \\
q_3 \\
q_4 \\
q_5
\end{bmatrix}
+ \begin{bmatrix}
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
\partial M^*_z/\partial q_1 \\
\partial M^*_z/\partial q_2 \\
\partial M^*_z/\partial q_3 \\
\partial M^*_z/\partial q_4 \\
\partial M^*_z/\partial q_5
\end{bmatrix}
\begin{bmatrix}
q_1 \\
q_2 \\
q_3 \\
q_4 \\
q_5
\end{bmatrix}
= 0,
\]

(3.35)

with eigenvalues and eigenvectors of the form

\[
\lambda_k = z_k \quad \text{and} \quad r_k = [1, z_k, z_k^2, z_k^3, z_k^4]^T,
\]

(3.36)

where $z_k$ cannot be written in closed form. Under physically reasonable assumptions, it can be shown that $z_k$ for $k = 1, \ldots, 5$ are all distinct and thus the system is strictly hyperbolic [8].

### 3.1.3 Quadrature using a bi-B-spline distribution

The bi-Gaussian closure of Chalons et al. [8] results in a strictly hyperbolic system, thus overcoming the weak hyperbolicity of the bi-delta closure. However, it achieves this at a cost of replacing a compactly supported distribution function by one that has infinite extent. One remedy for this was developed by Cheng and Rossmanith [4]. Their approach replaces bi-Gaussians with compactly supported bi-B-splines. Cheng and Rossmanith [4] use a distribution of the form:

\[
f \sim f^* = \omega_1 B^0_\sigma(v - \mu_1) + \omega_2 B^0_\sigma(v - \mu_2),
\]

(3.37)

where

\[
B^0_\sigma = \begin{cases}
\frac{2}{\sigma}(2v + \sqrt{\alpha}) & \text{if } -\sqrt{\alpha} \leq 2v \leq 0, \\
\frac{2}{\sigma}(\sqrt{\alpha} - 2v) & \text{if } 0 \leq 2v \leq \sqrt{\alpha}, \\
0 & \text{otherwise},
\end{cases}
\]

(3.38)

and $\sigma = \frac{24p}{\rho}(1 - \alpha)$ is the width of each B-spline. The distribution (3.37) is forced to have the first five moments (3.27)–(3.31) with $M_4$ slightly modified to be

\[
M_4 = \omega_1 \mu_1^4 + \omega_2 \mu_2^4 = \rho u^4 + 6\alpha \rho u^2 + 4qu + r + \frac{6p^2}{5\rho}(3\alpha + 2)(\alpha - 1).
\]

(3.39)
The moment-closure problem is then solved by replacing $M_5$ with

$$M_5^* = \rho u^5 + 10u^2(pu + q) + \frac{2pq}{\rho}(5 - 4\alpha) + \frac{p^2u}{\rho}(12 + 6\alpha - 13\alpha^2) + \frac{5q^2u}{p\alpha} + \frac{q^3}{p^2\alpha^2}. \quad (3.40)$$

This method also results in a system that is proven to be strictly hyperbolic by Cheng and Rossmanith [4].
3.2 A new approach - Gauss-Radau quadrature using a tri-delta distribution

In this work we develop an alternative to the bi-Gaussian and bi-B-spline approach that uses three delta functions, but where one has a prescribed location. This approach was conceived by Fox [27]. Our purpose here is to rigorously analyze the mathematical structure of this closure and, in the next chapter, to develop a high-order finite element method that is guaranteed to be moment-realizable.

By using Gauss-Radau quadrature and delta distributions the distribution for \( f \) will be

\[
 f \sim f^* = \omega_1 \delta(v - \mu_1) + \omega_2 \delta(v - u) + \omega_3 \delta(v - \mu_3),
\]

(3.41)

where we fix the second delta function at the point \( u \). This will result in a similar system to (3.4)–(3.7):

\[
 M_0 = \int_{-\infty}^{\infty} f(v) dv = \omega_1 + \omega_2 + \omega_3 = \rho, \quad (3.42)
\]

\[
 M_1 = \int_{-\infty}^{\infty} vf(v) dv = \omega_1 \mu_1 + \omega_2 u + \omega_3 \mu_3 = \rho u, \quad (3.43)
\]

\[
 M_2 = \int_{-\infty}^{\infty} v^2 f(v) dv = \omega_1 \mu_1^2 + \omega_2 u^2 + \omega_3 \mu_3^2 = \rho u^2 + p, \quad (3.44)
\]

\[
 M_3 = \int_{-\infty}^{\infty} v^3 f(v) dv = \omega_1 \mu_1^3 + \omega_2 u^3 + \omega_3 \mu_3^3 = \rho u^3 + 3pu + q, \quad (3.45)
\]

\[
 M_4 = \int_{-\infty}^{\infty} v^4 f(v) dv = \omega_1 \mu_1^4 + \omega_2 u^4 + \omega_3 \mu_3^4 = \rho u^4 + 6p^2 + 4qu + r, \quad (3.46)
\]

where

\[
 r = \int_{-\infty}^{\infty} (v - u)^4 f(v) dv. \quad (3.47)
\]

It follows that we have the system

\[
 \begin{bmatrix}
 \rho \\
 \rho u \\
 \rho u^2 + p \\
 \rho u^3 + 3pu + q \\
 \rho u^4 + 6pu^2 + 4qu + r
\end{bmatrix}
 +
 \begin{bmatrix}
 \rho u \\
 \rho u^2 + p \\
 \rho u^3 + 3pu + q \\
 \rho u^4 + 6pu^2 + 4qu + r \\
 M_5
\end{bmatrix}
 = 0, \quad (3.48)
\]
where we will close the system by replacing $M_5$ with

$$M_5^* = \int_{-\infty}^{\infty} v^5 f(v) dv = \omega_1 \mu_1^5 + \omega_2 u^5 + \omega_3 \mu_3^5. \quad (3.49)$$

Similar to (3.14) the orthogonal polynomial of degree 2 that makes this quadrature exact up to degree 3 is

$$P_2(v) = v^2 + u^2 - 2uv - \frac{p}{\rho - \omega_2} + \frac{q u - q v}{p}, \quad (3.50)$$

with roots

$$\mu_1 = \frac{q}{2p} + u + \frac{\sqrt{4p^3 + q^2 p - q^2 \omega_2}}{2p\sqrt{\rho - \omega_2}}, \quad \mu_3 = \frac{q}{2p} + u - \frac{\sqrt{4p^3 + q^2 p - q^2 \omega_2}}{2p\sqrt{\rho - \omega_2}}. \quad (3.51)$$

Since $\mu_1$ and $\mu_2$ are not only in terms of the primitive variables, we need to solve for $\omega_2$ to move on. If we get $\omega_1$ and $\omega_3$ in terms of $\omega_2$ then we can easily solve for all variables in terms of only the primitive variables. Plugging (3.51) into the system (3.42)-(3.46) it follows that

$$\omega_1 = \frac{q}{2p} - \frac{\omega_2}{q^2} - \frac{pq - \omega_2 q}{2\sqrt{\rho^3 + q^2 \omega_3 + 4pr}}, \quad \omega_3 = \frac{q}{2p} - \frac{\omega_2}{q^2} + \frac{pq - \omega_2 q}{2\sqrt{\rho^3 + q^2 \omega_3 + 4pr}}. \quad (3.52)$$

If we take (3.46) and plug in (3.52) it follows that

$$\omega_2 = \rho - \frac{p^3}{pr - q^2}. \quad (3.53)$$

Therefore we have

$$\omega_1 = \frac{q}{2p} + u + \frac{\sqrt{-3q^2 + 4pr}}{2p}, \quad \omega_3 = \frac{q}{2p} + u - \frac{\sqrt{-3q^2 + 4pr}}{2p}, \quad (3.54)$$

Hence we can write (3.48) in closed form by replacing $M_5$ with

$$M_5^* = \rho u^5 + 10pu^3 + 10qu^2 + 5ru + \frac{2qr}{p} - \frac{q^3}{p^2}. \quad (3.55)$$

This system in terms of the conserved variables is

$$\begin{bmatrix}
q_1 \\
q_2 \\
q_3 \\
q_4 \\
q_5
\end{bmatrix} + \begin{bmatrix}
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 \\
\partial M_5^* / \partial q_1 & \partial M_5^* / \partial q_2 & \partial M_5^* / \partial q_3 & \partial M_5^* / \partial q_4 & \partial M_5^* / \partial q_5
\end{bmatrix} \begin{bmatrix}
q_1 \\
q_2 \\
q_3 \\
q_4 \\
q_5
\end{bmatrix} = 0, \quad (3.56)$$
where

\[
\frac{\partial M^*_5}{\partial q_1} = -q^2 u + ru + 2q^3 u^2 - 2qru^2 + 3q^2 u^3 - 2ru^3 + 2qu^4 + u^5, \tag{3.57}
\]

\[
\frac{\partial M^*_5}{\partial q_2} = \frac{q^2}{p^3} - \frac{r}{\rho} - \frac{4q^3 u}{p^3} + \frac{4qru}{p^3} + \frac{9q^2 u^2}{p^2} - \frac{6ru^2}{p} + \frac{8qu^3}{p}, \tag{3.58}
\]

\[
\frac{\partial M^*_5}{\partial q_3} = \frac{2q^3}{p^3} - \frac{2qr}{p^2} + \frac{9q^2 u}{p^2} - \frac{6ru}{p} + \frac{12qu^2}{p} + 10u^3, \tag{3.59}
\]

\[
\frac{\partial M^*_5}{\partial q_4} = -\frac{3q^2}{p^2} + \frac{2r}{p} - \frac{8qu}{p} - 10u^2, \tag{3.60}
\]

\[
\frac{\partial M^*_5}{\partial q_5} = \frac{2q}{p} + 5u, \tag{3.61}
\]

and in terms of the primitive variables is

\[
\begin{bmatrix}
\rho \\
u \\
p \\
q \\
r
\end{bmatrix}
+ \begin{bmatrix}
0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 3p & u & 1 & 0 \\
0 & 4q & -\frac{3p}{\rho} & u & 1 \\
0 & 5r & \frac{2q^3}{p^3} & \frac{2qr}{p^2} & -\frac{2q}{p} & -\frac{3q^2}{p^2} & \frac{2q}{p} & + u
\end{bmatrix}
\begin{bmatrix}
\rho \\
u \\
p \\
q \\
r
\end{bmatrix}
= 0. \tag{3.62}
\]

### 3.2.1 Analysis of eigensystem

The eigenvalues are

\[
\lambda_1 = u, \\
\lambda_2 = u + \frac{q}{2p} - \frac{\sqrt{-3p^4 q^2 \rho^2 + 4p^2 \rho^2 - 4\sqrt{\alpha}}}{2p \rho}, \\
\lambda_3 = u + \frac{q}{2p} + \frac{\sqrt{-3p^4 q^2 \rho^2 + 4p^2 \rho^2 - 4\sqrt{\alpha}}}{2p \rho}, \\
\lambda_4 = u + \frac{q}{2p} - \frac{\sqrt{-3p^4 q^2 \rho^2 + 4p^2 \rho^2 + 4\sqrt{\alpha}}}{2p \rho}, \\
\lambda_5 = u + \frac{q}{2p} + \frac{\sqrt{-3p^4 q^2 \rho^2 + 4p^2 \rho^2 + 4\sqrt{\alpha}}}{2p \rho},
\]

where \(\alpha = -p^8 (-q^2 + pr)p^3 (p^3 + q^2 \rho - pr \rho).\) The corresponding eigenvectors are

\[
r_1 = \begin{bmatrix} 1, \lambda_1, \lambda_1^2, \lambda_1^3, \lambda_1^4 \end{bmatrix}^T, \\
r_2 = \begin{bmatrix} 1, \lambda_2, \lambda_2^2, \lambda_2^3, \lambda_2^4 \end{bmatrix}^T, \\
r_3 = \begin{bmatrix} 1, \lambda_3, \lambda_3^2, \lambda_3^3, \lambda_3^4 \end{bmatrix}^T, \\
r_4 = \begin{bmatrix} 1, \lambda_4, \lambda_4^2, \lambda_4^3, \lambda_4^4 \end{bmatrix}^T, \\
r_5 = \begin{bmatrix} 1, \lambda_5, \lambda_5^2, \lambda_5^3, \lambda_5^4 \end{bmatrix}^T.
\]

(3.64)
Only one of these waves is linearly degenerate since it can be readily shown that

\[ \nabla q \lambda_1 \cdot r_1 = 0, \]
\[ \nabla q \lambda_2 \cdot r_2 \neq 0, \]
\[ \nabla q \lambda_3 \cdot r_3 \neq 0, \quad (3.65) \]
\[ \nabla q \lambda_4 \cdot r_4 \neq 0, \]
\[ \nabla q \lambda_5 \cdot r_5 \neq 0. \]

Lemma 3.2.1. (Moment-realizability condition, modified from Chalons et al. [9] and Cheng and Rossmanith [4]). Under the assumptions on the primitive variables

1. Positive pressure: \( p > 0 \),
2. Positive density: \( \rho > 0 \),
3. Lower bound on \( r \): \( r - \frac{p^2}{\rho} - \frac{q^2}{p} > 0 \),

then the system is strictly hyperbolic.

**Proof.** Recall that for a system to be strictly hyperbolic we need real, distinct eigenvalues and a complete set of eigenvectors. To ensure that the eigenvalues are real and distinct we must satisfy the following two conditions:

1. \( \alpha > 0 \),
2. \(-3p^4q^2\rho^2 + 4p^5r\rho^2 > 4\sqrt{\alpha} \).

First we will analyze \( \alpha = -p^8(-q^2 + pr)\rho^3(p^3 + q^2\rho - pr\rho) \). Since \( p > 0, \rho > 0 \), it follows that

\[ -p^8 < 0, \quad (3.66) \]
\[ \rho^3 > 0, \quad (3.67) \]

therefore to keep \( \alpha \) positive we will force

\[ -q^2 + pr > 0, \quad (3.68) \]

resulting in the necessary condition

\[ r > \frac{q^2}{p}. \quad (3.69) \]
This is consistent with assumption 3, since
\[ r > \frac{p^2}{\rho} + \frac{q^2}{p} > \frac{q^2}{p}. \]  
(3.70)

Next we need that
\[ p^3 + q^2 \rho - pr \rho < 0, \]  
(3.71)
resulting in the necessary condition
\[ r > \frac{p^2}{\rho} + \frac{q^2}{p}, \]  
(3.72)
which is exactly the assumption that was made. Therefore we have proved that \( \alpha > 0 \). Moving to the second condition it follows that we need
\[ r > \frac{16p^3q^2 + 7q^4 \rho}{16p^4 + 8pq^2 \rho}. \]  
(3.73)
If we take (3.73) and subtract \( \frac{q^2}{p} \) we get
\[ -\frac{q^4 \rho}{16p^4 + 8pq^2 \rho} < 0. \]  
(3.74)
This implies that
\[ \frac{q^2}{p} > \frac{16p^3q^2 + 7q^4 \rho}{16p^4 + 8pq^2 \rho}, \]  
(3.75)
and since we know that \( r > \frac{q^2}{p} \) by assumption then we have satisfied (3.73), hence satisfying the second condition. If we assume \( q = 0 \) then it follows that we need
\[ r > 0; \]  
(3.76)
which is satisfied by definition (3.47). Therefore each eigenvalue will be real and distinct, resulting in a complete set of eigenvectors and the system is therefore strictly hyperbolic.

Convexity of the constraints \( \phi_k \) can be proven by showing that \( \phi_k''(q) \) is positive semi-definite. Recall the gradient is defined to be
\[ \phi_k'(q) = \left[ \frac{\partial \phi_k}{\partial q_1}, \frac{\partial \phi_k}{\partial q_2}, \frac{\partial \phi_k}{\partial q_3}, \frac{\partial \phi_k}{\partial q_4}, \frac{\partial \phi_k}{\partial q_5} \right], \]  
(3.77)
and the Hessian is
\[ \phi_k''(q) = \begin{bmatrix} \frac{\partial^2 \phi_k}{\partial q_1 \partial q_1} & \frac{\partial^2 \phi_k}{\partial q_1 \partial q_2} & \cdots & \frac{\partial^2 \phi_k}{\partial q_1 \partial q_5} \\ \frac{\partial^2 \phi_k}{\partial q_2 \partial q_1} & \frac{\partial^2 \phi_k}{\partial q_2 \partial q_2} & \cdots & \frac{\partial^2 \phi_k}{\partial q_2 \partial q_5} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 \phi_k}{\partial q_5 \partial q_1} & \frac{\partial^2 \phi_k}{\partial q_5 \partial q_2} & \cdots & \frac{\partial^2 \phi_k}{\partial q_5 \partial q_5} \end{bmatrix}, \]  
(3.78)
where the conserved variables are

- $q_1 = \rho$,
- $q_2 = \rho u$,
- $q_3 = \rho u^2 + p$,
- $q_4 = \rho u^3 + 3\rho u + q$,
- $q_5 = \rho u^4 + 6\rho u^2 + 4\rho u + r$.

**Lemma 3.2.2.** $G = \{q \in \mathbb{R}^5 : \rho > 0, p > 0, r - \frac{v^2}{\rho} - \frac{q^2}{p} > 0\}$ is a convex set of the conserved variables $q_1, q_2, q_3, q_4, q_5$.

**Proof.** We will prove by cases. Define $\phi_k$ for $k = 1, 2, 3$:

1. $\phi_1 = \rho = q_1$,
2. $\phi_2 = p = q_3 - \frac{q_3^2}{q_1}$,
3. $\phi_3 = r - \frac{v^2}{\rho} - \frac{q^2}{p} = \frac{q_3^3 - 2q_2 q_3 q_4 + q_3 q_4^2 + q_4^3 - q_1 q_3 q_5}{q_1^2}$.

1. Consider $\phi_1 = q_1$, the gradient is given by
   \[
   \phi_1'(q) = \begin{bmatrix} 1, 0, 0, 0, 0 \end{bmatrix}^T ,
   \]
   and the Hessian will simply be the zero matrix which satisfies the convexity condition.

2. Consider $\phi_2 = q_3 - \frac{q_3^2}{q_1}$, the gradient is given by
   \[
   \phi_2'(q) = \begin{bmatrix} \frac{q_3^2}{q_1}, -\frac{q_3}{q_1}, 1, 0, 0 \end{bmatrix}^T ,
   \]
   and the Hessian in terms of the primitive variables is
   \[
   \phi_2''(q) = \frac{-2}{\rho} \begin{bmatrix} u^2 & -u & 0 & 0 & 0 \\
   -u & 1 & 0 & 0 & 0 \\
   0 & 0 & 0 & 0 & 0 \\
   0 & 0 & 0 & 0 & 0 \\
   0 & 0 & 0 & 0 & 0 \end{bmatrix} .
   \]
This results in the eigenvalues \( \lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = 0 \), \( \lambda_5 = -\frac{2(u^2 + 1)}{p} \), which satisfies the convexity condition.

3. Consider \( \phi_3 = \frac{q_3^2 - 2q_2 q_1 q_4 + q_1^2 q_5 + q_2^2 q_5 - q_1 q_5 q_4}{q_2^2 - q_1 q_3} \), the gradient is given by

\[
\phi_3'(q) = \begin{bmatrix}
\frac{(q_3^2 - q_2 q_1)^2}{(q_2^2 - q_1 q_3)^2} \\
\frac{2(0 - q_2 q_1) - (-q_2^2 + q_2 q_4)}{(q_2^2 - q_1 q_3)^2} \\
\frac{3q_3^2 q_1^2 - 2q_3 q_1 + q_1(-2q_3^2 + q_5^2)}{(q_2^2 - q_1 q_3)^2} \\
\frac{-2q_3 q_1 + 2q_2 q_4}{q_2^2 - q_1 q_3} \\
1
\end{bmatrix}.
\] (3.82)

The Hessian is suppressed due to large entries in terms of primitive variables. The Hessian is symmetric which implies that we will have real eigenvalues. The resulting eigenvalues are \( \lambda_1 = 0, \lambda_2 = 0, \lambda_3 = 0, \lambda_4 = -\frac{b - \sqrt{b^2 - 4ac}}{2a}, \lambda_5 = -\frac{b + \sqrt{b^2 - 4ac}}{2a}, \) (3.83)

where \( a = \frac{p^2 q^2}{2r} \),

\[
b = p^6 \rho^4 (p^5 + p^4 (\rho - \rho u^2) + q^2 \rho^3 (1 + 4u^2 + u^4) + p^2 \rho^3 (1 + 9u^2 + 9u^4 + u^6) + p^3 \rho (-2qu - \rho (-1 + 2u^2 + u^4)) + pq \rho^2 (q (1 + u^2) + 2 \rho u (3 + 6u^2 + u^4))),
\]

\[
c = 2p^4 \rho^2 (p^8 - 4p^7 \rho u^2 + q^4 \rho^4 (1 + u^2 + u^4) + 2pq^3 \rho^4 u (3 + 3u^2 + 2u^4) + 2p^6 \rho (\rho - 2qu + 2qu^2 + 3qu^4) + 4p^5 \rho^2 u (q + 3qu^2 - \rho u (-1 + 2u^2 + u^4)) + 2p^3 \rho q^3 (q (1 - 5u^2 - 6u^4) + 2\rho u (1 + 5u^2 + 3u^4 + u^6)) + p^2 q^2 \rho^3 (-2q (u + u^3) + \rho (1 + 16u^2 + 13u^4 + 6u^6)) + p^4 \rho^2 (q^2 (1 + 6u^2) - 4q \rho u (-1 + 4u^2 + 3u^4) + \rho^2 (1 + 4u^2 + 10u^4 + 4u^6 + u^8))).
\]

For \( \phi_3 \) to satisfy the convexity condition we need \( \lambda_4 < 0 \) and \( \lambda_5 < 0 \). This is guaranteed to happen if \( a > 0, b > 0, c > 0 \). It is clear to see that \( a > 0 \) given that \( p > 0 \) and \( \rho > 0 \), therefore we will prove that \( b > 0 \) and \( c > 0 \).

We note if \( p = 0 \) or \( \rho = 0 \), then \( b = 0 \). We also note that the only real solutions to

\[
\nabla b = 0,
\] (3.84)
are \( p = 0 \) and \( \rho = 0 \). This means that there are no local minimums, maximums, or saddle points for any \( p > 0 \) and \( \rho > 0 \). Furthermore, if we set \( u = 0 \) we have that
\[
b = p^6 \rho^4(p + \rho)(p^4 + p^2 \rho^2 + q^2 \rho^2), \tag{3.85}
\]
which is always positive given that \( p > 0 \) and \( \rho > 0 \). Therefore, we are guaranteed that \( b \) must be positive for all choices of the primitive variables such that \( p > 0 \) and \( \rho > 0 \).

In a similar manner, we note that when \( p = 0 \) or \( \rho = 0 \), then \( c = 0 \). If we consider the solutions to
\[
\nabla c \equiv 0, \tag{3.86}
\]
the only real solutions are \( p = 0 \) and \( \rho = 0 \). This means that there are no local minimums, maximums, or saddle points for any \( p > 0 \) and \( \rho > 0 \). Furthermore, if we set \( u = 0 \) we have that
\[
c = 2(p^8 + 2p^6 \rho^2 + 2p^3 q^2 \rho^3 + p^2 q^2 \rho^4 + q^4 \rho^4 + p^4 \rho^2(q^2 + \rho^2)), \tag{3.87}
\]
which is always positive given that \( p > 0 \) and \( \rho > 0 \). Therefore, we are guaranteed that \( c \) must be positive for all choices of the primitive variables such that \( p > 0 \) and \( \rho > 0 \).
CHAPTER 4. DISCONTINUOUS GALERKIN

In the previous chapter we established that under physically reasonable conditions, the tri-delta moment-closure is strictly hyperbolic. Now that we have established this, we move on to the task of developing a high-order discontinuous Galerkin finite element discretization of the resulting fluid equation. In this chapter we review the discontinuous Galerkin (DG) method that was originally developed by Cockburn and Shu [12] and is currently implemented in the DOGPACK software package [13].

4.1 Semi-discrete discontinuous Galerkin method

We first construct an equally spaced grid

$$\mathcal{T}_{\Delta x} = \{x_i = a + (i - 1/2)\Delta x \quad \text{for} \quad i = 1, \ldots, N\},$$

where $\Delta x = (b - a)/N$ is the grid spacing with $N$ elements, and $x_i$ is defined to be the center of the region $[x_{i-1/2}, x_{i+1/2}]$. We next define the broken finite element space

$$V^\Delta x = \left\{ v^\Delta x \in L^\infty(\Omega) : v^\Delta x|_T \in P^k, \forall T \in \mathcal{T}_{\Delta x} \right\},$$

where each $v^\Delta x$ will be a polynomial of degree $k$ on each element $T$. The solution $q^\Delta x \in V^\Delta x$ restricted to the element $T_i$ is

$$q^\Delta x \bigg|_{T_i} = \sum_{\ell=1}^M Q^\ell(t) \phi^{(\ell)}(\xi),$$

where $\xi \in [-1, 1]$ is the canonical variables on each element, and the orthonormal Legendre basis polynomials are

$$\phi^{(\ell)}(\xi) = \left\{ 1, \sqrt{3} \xi, \frac{\sqrt{5}}{2} (3\xi^2 - 1), \ldots \right\}.$$
In order to obtain the semi-discrete DG method we (1) multiply hyperbolic conservation law (1.1) by \( \phi(\ell)(\xi) \), (2) integrate over an element \( T_i \), (3) integrate by part in \( x \) (or \( \xi \)), and lastly (4) replace \( q \) by (4.3). We then arrive at the set of coupled ordinary differential equations in time:

\[
\frac{d}{dt} Q^{(j)}_i = \frac{1}{\Delta x} \int_{-1}^{1} f(Q) \phi^{(j)}_{\xi} d\xi - \frac{1}{\Delta x} \left[ \phi^{(j)}(1) F_{i+1/2} - \phi^{(j)}(-1) F_{i-1/2} \right],
\]

where \( F_{i+1/2} \) is the numerical flux at the interface \( x = x_{i+1/2} \). The numerical flux is ambiguous at the interface, since the solution there is discontinuous. The ambiguous numerical flux is obtained through an approximate Riemann solver such as the local Lax-Friedrichs or Rusanov flux [28]:

\[
F_{i+1/2} = \frac{1}{2} (f(Q^+) - f(Q^-)) - \frac{1}{2} s_{\text{max}} (Q^+ - Q^-),
\]

where \( Q^+ \) and \( Q^- \) are the solution values on either side of the interface \( x_{i+1/2} \), and \( s_{\text{max}} \) is an estimate of the fastest wave speed near \( x_{i+1/2} \).

### 4.2 Total variation diminishing Runge-Kutta time-stepping

In order to time advance the semi-discrete scheme, we use the standard third-order total variation diminishing Runge-Kutta (TVD-RK) described by Gottlieb and Shu [14]. To illustrate these methods consider the initial value problem:

\[
\frac{d}{dt} q = L(q).
\]

The first order TVD-RK method is simply the forward Euler method:

\[
Q^{n+1} = Q^n + \Delta t L(Q^n).
\]

The second order accurate version is

\[
Q^* = Q^n + \Delta t L(Q^n),
\]

\[
Q^{n+1} = \frac{1}{2} Q^n + \frac{1}{2} Q^* + \frac{1}{2} \Delta t L(Q^*).
\]
And finally, the third order accurate version is

\[ Q^* = Q^n + \Delta t \mathcal{L}(Q^n), \]
\[ Q^{**} = \frac{3}{4} Q^n + \frac{1}{4} U^* + \frac{1}{4} \Delta t \mathcal{L}(Q^*), \]
\[ Q^{n+1} = \frac{1}{3} Q^n + \frac{2}{3} Q^{**} + \frac{2}{3} \Delta t \mathcal{L}(Q^{**}). \]

(4.11) \hspace{1cm} (4.12) \hspace{1cm} (4.13)

When the \( O(\Delta t^k) \) TVD Runge-Kutta method is applied to the \( P^{k-1} \) discontinuous Galerkin method, the maximum stable time step satisfies

\[ \text{CFL}_{\text{max}} \equiv O(k^{-1}) = \max \left\{ \lambda^{(p)}(q, x) \right\} \cdot \frac{\Delta t_{\text{max}}}{\Delta x}, \]

(4.14)

where \( \max \{\lambda^p\} \) is the spectral radius of the flux Jacobian, \( \partial f/\partial q \), maximized over the entire computational domain.

### 4.3 Limiting

It is well-known that applying high-order methods to problems where shocks arise leads to unphysical oscillations in the numerical solution. A standard approach for removing these unwanted oscillations is through a post-processing step called a limiter. The basic idea is that if the derivatives of the solution becomes too large inside an element, it is likely due a shock propagating through this element. Since the Legendre coefficients, \( Q^{(\ell)} \) for \( \ell > 1 \), are proportional to derivatives of the solution, the goal is to limit these when needed.

In this work we will follow the procedure of Krivodonova [10], which is described below for a third-order accurate discontinuous Galerkin method (i.e, \( M = 3 \)). The process in each element is:

1. Replace \( Q_i^{(3)} \) by the following limited value:

\[ L_i \tilde{Q}_i^{(3)} \leftarrow \minmod \left\{ L_i Q_i^{(3)}, \frac{\sqrt{3}}{\sqrt{5}} L_i \left( Q_{i+1}^{(2)} - Q_i^{(2)} \right), \frac{\sqrt{3}}{\sqrt{5}} L_i \left( Q_i^{(2)} - Q_{i-1}^{(2)} \right) \right\}, \]

where \( L \) is the matrix of left eigenvectors of the flux Jacobian, \( \partial f/\partial q \), and \( L_i \equiv L(Q_i^{(1)}) \) (i.e., \( L_i \) is the matrix of left eigenvectors evaluated using the solution average in cell \( i \)).
In the above expression we define \( \text{minmod}(a, b, c) \) as follows:

\[
\text{minmod}(a, b, c) = \begin{cases} 
\text{sgn}(a) \min(\{|a|, |b|, |c|\}) & \text{if } \text{sgn}(a) = \text{sgn}(b) = \text{sgn}(c), \\
0 & \text{otherwise}.
\end{cases}
\]

2. If \( \tilde{Q}_i^{(3)} = Q_i^{(3)} \) then \textbf{STOP}; otherwise go to Step (3).

3. Replace \( Q_i^{(2)} \) by the following limited value:

\[
L_i\tilde{Q}_i^{(2)} \leftarrow \text{minmod}\left\{ L_iQ_i^{(2)}, \frac{1}{\sqrt{3}} L_i \left( Q_{i+1}^{(1)} - Q_i^{(1)} \right), \frac{1}{\sqrt{3}} L_i \left( Q_i^{(1)} - Q_{i-1}^{(1)} \right) \right\}.
\]

This limiting strategy is applied in each stage of the Runge-Kutta discontinuous Galerkin method.
CHAPTER 5. SOLVING THE TRI-DELTA SYSTEM WITH DISCONTINUOUS GALERKIN AND CONSTRAINT-PRESERVING TECHNIQUES

In this chapter we develop an additional limiting step, which when applied to each Runge-Kutta discontinuous Galerkin stage is shown to guarantee the exact numerical preservation of the strict hyperbolicity and moment-realizability constraints given in Lemma 3.2.1. This new limiter is an extension of the Zhang and Shu [16] limiter that was previously shown to guarantee exact positive density and pressure for the Euler equations. After developing the constraint limiter, we show numerical results of the Riemann problem.

5.1 Constraint-preserving condition

Lemma 5.1.1. (Constraint-preserving condition, modified from Perthame and Shu [11]). Assume there exists a convex set of constraints \( G \) and eigenvalues of the flux Jacobian that are bounded, then the first order piecewise constant discontinuous Galerkin method with the local Lax-Friedrichs Riemann solver is constraint-preserving by restricting the CFL condition to be

\[
\frac{\Delta t}{\Delta x} s_{\text{max}} < \frac{1}{2},
\]

where \( s_{\text{max}} \) is maximum wave speed over the entire computational domain.

Proof. Consider the modified system from (1.1):

\[
q_t + [f(q) + sq]_x = 0,
\]

\[
q_t + [f(q) - sq]_x = 0,
\]

where \( s \) is a strict upper bound on the maximum eigenvalue of \( f'(q) \). Consider the eigenvalues
of (5.2):

\[
\begin{align*}
\lambda_1 &= u + s, \\
\lambda_2 &= u + \frac{q}{2p} - \sqrt{-3p^2q^2 + 4p^2r - 4\sqrt{\alpha}} + s, \\
\lambda_3 &= u + \frac{q}{2p} + \sqrt{-3p^2q^2 + 4p^2r + 4\sqrt{\alpha}} + s, \\
\lambda_4 &= u + \frac{q}{2p} + \sqrt{-3p^2q^2 + 4p^2r - 4\sqrt{\alpha}} + s, \\
\lambda_5 &= u + \frac{q}{2p} - \sqrt{-3p^2q^2 + 4p^2r + 4\sqrt{\alpha}} + s,
\end{align*}
\]

(5.4)

where \( \alpha = -p^3(-q^2 + pr)^3(p^3 + q^2\rho - pr\rho) \). Since \( s \) is a strict upper bound on the eigenvalues of \( f'(q) \), it follows that each of the above eigenvalues will be positive. If we update (5.2) using the piecewise constant \((M = 0)\) discontinuous Galerkin scheme and an exact Riemann solver, we obtain:

\[
\dot{Q}_{i+1} = Q_i - \frac{\Delta t}{\Delta x} \left[ \hat{F}_{i+\frac{1}{2}} - \hat{F}_{i-\frac{1}{2}} \right],
\]

(5.5)

where

\[
\begin{align*}
\hat{F}_{i+\frac{1}{2}} &= f(Q_i^\alpha) + s_{i+\frac{1}{2}} Q_i^\alpha, \\
\hat{F}_{i-\frac{1}{2}} &= f(Q_{i-1}^\alpha) + s_{i-\frac{1}{2}} Q_{i-1}^\alpha.
\end{align*}
\]

(5.6)  (5.7)

Alternatively, this can be written as

\[
\dot{Q}_{i+1} = Q_i - \frac{\Delta t}{\Delta x} \left[ f(Q_i^{i+1}) + s_{i+\frac{1}{2}} Q_i^{i+1} - f(Q_i^{i-1}) - s_{i-\frac{1}{2}} Q_i^{i-1} \right].
\]

(5.8)

Similarly, to solve the system (5.3), we have the scheme:

\[
\dot{Q}_{i+1} = Q_i - \frac{\Delta t}{\Delta x} \left[ f(Q_i^{i+1}) - s_{i+\frac{1}{2}} Q_i^{i+1} - f(Q_i^{i-1}) + s_{i-\frac{1}{2}} Q_i^{i-1} \right].
\]

(5.9)

By taking the arithmetic average of (5.8) and (5.9) we get

\[
\ddot{Q}_{i+1} = Q_i - \frac{\Delta t}{\Delta x} \left[ \frac{1}{2} \left( f(Q_i^{i+1}) + f(Q_i^{i-1}) \right) - \frac{1}{2} s_{i+\frac{1}{2}} (Q_i^{i+1} - Q_i^{i-1}) - \frac{1}{2} (f(Q_i^{i+1}) + f(Q_i^{i-1})) + \frac{1}{2} s_{i-\frac{1}{2}} (Q_i^{i-1} - Q_i^{i+1}) \right],
\]

(5.10)

which is precisely the local Lax-Friedrichs scheme applied to the original system (1.1):

\[
\ddot{Q}_{i+1} = Q_i - \frac{\Delta t}{\Delta x} \left[ F_{i+\frac{1}{2}} - F_{i-\frac{1}{2}} \right],
\]

(5.11)
where

\[ F_{i+1/2} = \frac{1}{2} (f(Q^n_{i+1}) + f(Q^n_{i})) - \frac{1}{2} s_{i+\frac{1}{2}} (Q^n_{i+1} - Q^n_{i}), \quad (5.12) \]

\[ F_{i-1/2} = \frac{1}{2} (f(Q^n_{i-1}) + f(Q^n_{i})) - \frac{1}{2} s_{i-\frac{1}{2}} (Q^n_{i} - Q^n_{i-1}). \quad (5.13) \]

Since the exact Riemann solver is constraint-preserving, then \( Q^{n+1}_i \), as the convex combination of two constraint-preserving states, must also satisfies the constraints. In order for the Riemann solver to remain exact we impose the condition that \( \frac{\Delta t}{\Delta x} s_{\text{max}} < \frac{1}{2} \) to ensure there are no interactions between neighboring Riemann problems.

\[ \square \]

**Theorem 5.1.2.** (From Zhang and Shu [16]) Consider a finite volume scheme or the scheme satisfied by the cell average of the discontinuous Galerkin method (5.11), if \( q_j(x_\alpha) \in G \) for all \( j \) and \( \alpha \), then \( \bar{Q}^{n+1}_i \in G \) under the CFL condition

\[ \frac{\Delta t}{\Delta x} ||u|| + c||_\infty \leq \hat{\omega}_1 \alpha_0, \quad (5.14) \]

where \( \hat{\omega}_\alpha \) is the Legendre Gauss-Lobatto quadrature weights for the interval \([-\frac{1}{2}, \frac{1}{2}]\) such that \( \sum_{\alpha=1}^{M} \hat{\omega}_\alpha = 1 \).

**Proof.** Start with scheme (5.11). Next, note that the exactness of the quadrature rule for polynomials of degree \( k \) implies

\[ \bar{Q}^n_i = \frac{1}{\Delta x} \int_{\tau_i} q_i(x) dx = \sum_{\alpha=1}^{M} \hat{\omega}_\alpha q_i(\hat{x}_\alpha). \quad (5.15) \]

By adding and subtracting \( F(Q^+_{i-1/2}, Q^-_{i+1/2}) \) where \( Q^+_{i-1/2} = q^{\Delta x} |_{\tau_i} (x_i - 1/2) \) and similarly \( Q^-_{i+1/2} = q^{\Delta x} |_{\tau_i} (x_i + 1/2) \), the scheme (5.11) becomes

\[ \bar{Q}^{n+1}_i = \sum_{\alpha=1}^{N} \hat{\omega}_\alpha q_j(\hat{x}_\alpha) - \frac{\Delta t}{\Delta x} [F(Q^-_{i+1/2}, Q^+_{i+1/2}) - F(Q^+_{i-1/2}, Q^-_{i+1/2})] \]
\[ + F(Q^-_{i-1/2}, Q^-_{i+1/2}) - F(Q^+_{i-1/2}, Q^+_{i+1/2}), \quad (5.16) \]

\[ = \sum_{\alpha=1}^{N-1} \hat{\omega}_\alpha q_j(\hat{x}_\alpha) + \hat{\omega}_N (Q^-_{i+1/2} - \frac{\Delta t}{\Delta x \bar{\omega}_N} [F(Q^-_{i+1/2}, Q^+_{i+1/2}) - F(Q^+_{i-1/2}, Q^-_{i+1/2})]) \]
\[ + \hat{\omega}_1 (Q^-_{i+1/2} - \frac{\Delta t}{\Delta x \bar{\omega}_N} [F(Q^+_{i-1/2}, Q^-_{i+1/2}) - F(Q^-_{i-1/2}, Q^+_{i-1/2})]), \quad (5.17) \]

\[ = \sum_{\alpha=1}^{N-2} \hat{\omega}_\alpha q_j(\hat{x}_\alpha) + \hat{\omega}_N H_N + \hat{\omega}_1 H_1, \quad (5.19) \]
where

\[ H_1 = Q_{i-1/2}^+ - \frac{\Delta t}{\Delta x \hat{\omega}_1} [F(Q_{i-1/2}^+, Q_{i+1/2}^-) - F(Q_{i-1/2}^-, Q_{i+1/2}^+)], \quad (5.20) \]

\[ H_N = Q_{i+1/2}^- - \frac{\Delta t}{\Delta x \hat{\omega}_1} [F(Q_{i+1/2}^-, Q_{i+1/2}^+) - F(Q_{i-1/2}^-, Q_{i+1/2}^+)]. \quad (5.21) \]

Notice that (5.20) and (5.21) are both updates of the type (5.11) and \( \hat{\omega}_1 = \hat{\omega}_N \), therefore \( H_1 \) and \( H_N \) are in the set \( G \) under the CFL condition (5.14). Now, it is easy to conclude that \( \bar{Q}_i^{n+1} \) is in \( G \), since it is a convex combination of elements in \( G \).

5.1.1 Constraint-preserving limiter

The final ingredient in guaranteeing that \( \bar{Q}_i^{n+1} \in G \) is to ensure that the high-order polynomial, \( q_i(x) \), in element \( i \) satisfies \( q_i(x_\alpha) \in G \) at the Gauss-Lobatto points \( x_\alpha \) in element \( i \) – this is assumed in the previous lemma. These conditions are enforced by generalizing the Zhang and Shu [16] limiter to the tri-delta case. Let \( \epsilon > 0 \), and assume that \( \bar{p}^n_i \geq \epsilon, \bar{\rho}^n_i \geq \epsilon, \bar{\rho}^n_i - \frac{(\bar{\rho}^n_i)^3 + (\bar{\rho}^n_i)^2 \bar{\rho}^n_i}{\bar{p}^n_i \bar{\rho}^n_i} \geq \epsilon \) for all \( i \). The update for the solution in element \( i \) is

\[ q^{\Delta x}|_{T_i} = Q_i^{(1)} + \theta \sum_{k=2}^{M} Q_i^{(k)} \phi^{(k)}(\xi). \quad (5.22) \]

1. First we limit the density by finding the \( \theta \) such that

\[ \theta = \min \left\{ \frac{\bar{p}^n_i - \epsilon}{\bar{p}^n_i - \rho_{\min}}, 1 \right\}, \quad \text{where} \quad \rho_{\min} = \min_{\alpha} \rho^{\Delta x}|_{T_i}(x_\alpha). \quad (5.23) \]

We update the solutions which now has a guaranteed positive average density:

\[ Q_i^{(1)} \leftarrow Q_i^{(1)}, \quad (5.24) \]

\[ Q_i^{(k)} \leftarrow \theta Q_i^{(k)}, \quad (5.25) \]

for \( k = 2, \ldots, M \).

2. Next we limit the pressure. Define

\[ p_\alpha(\theta) := p_\alpha \left( Q_i^{(1)} + \theta \sum_{k=2}^{M} Q_i^{(k)} \phi^{(k)}(\xi_\alpha) \right). \quad (5.26) \]

Consider the two points, \( p_\alpha(0) = p_0, p_\alpha(1) = p_1 \), where the line \( p_\alpha^* = p_0 + \theta(p_1 - p_0) \) passes through these two points. This line will intersect the \( \theta \)-axis at one point. By
setting $p_\alpha^* = 0$ we can solve for the $\theta$:

$$\theta_\alpha = \min \left\{ \frac{p_0 - \epsilon}{p_0 - p_1}, 1 \right\}. \quad (5.27)$$

Then let $\theta = \min_\alpha(\theta_\alpha)$ and we again limit

$$Q_i^{(1)} \leftarrow Q_i^{(1)}, \quad (5.28)$$

$$Q_i^{(k)} \leftarrow \theta Q_i^{(k)}, \quad (5.29)$$

for $k = 2, \ldots, M$, which ensures that $\hat{p}_i(x_\alpha) \geq \epsilon$ for all $\alpha$.

3. Next we will limit $r - \frac{r^2}{\rho} - \frac{q^2}{p}$. Define

$$r^* := r - \frac{p^2}{\rho} - \frac{q^2}{p}, \quad (5.30)$$

and

$$r_\alpha^*(\theta) := r_\alpha^* \left( Q_i^{(1)} + \theta \sum_{k=2}^{M} Q_i^{(k)} \phi^{(k)}(\xi_\alpha) \right). \quad (5.31)$$

Consider the two points, $r_\alpha^*(0) = r_0^*$, $r_\alpha^*(1) = r_1^*$, where the line $r^{**} = r_0^* + \theta_\alpha(r_1^* - r_0^*)$ passes through these two points. This line will intersect the $\theta$-axis at one point. By setting $r^{**} = 0$ we can solve for this $\theta$:

$$\theta_\alpha = \min \left\{ \frac{r_0^* - \epsilon}{r_0^* - r_1^*}, 1 \right\}. \quad (5.32)$$

Then set $\theta = \min_\alpha(\theta_\alpha)$ and we again limit

$$Q_i^{(1)} \leftarrow Q_i^{(1)}, \quad (5.33)$$

$$Q_i^{(k)} \leftarrow \theta Q_i^{(k)}, \quad (5.34)$$

for $k = 2, \ldots, M$, which ensures that $\hat{r}_i(x_\alpha) \geq \epsilon$ for all $\alpha$.

The above described constraint-preserving limiter has been implemented in the DOGPACK [13] code. The implementation details can be found in Appendix A.
5.2 Numerical results

In this section we solve two shock-tube problems with different left and right states. For each simulation, in addition to the primitive variables, we also plot the constraint, \( r - \frac{v^2}{p} - \frac{q^2}{p} \), the quadrature weights, \( \omega_1, \omega_2, \omega_3 \), and the quadrature points \( \mu_1, u, \mu_3 \).

In Figures 5.1 and 5.2 we show a shock tube problem for the tri-delta system with initial states

\[
(\rho, u, p, q, r)_L = (1.5, -0.5, 1.5, 1.0, 4.5), \quad (5.35)
\]
\[
(\rho, u, p, q, r)_R = (1.0, -0.5, 1.0, 0.5, 3.0). \quad (5.36)
\]

The panels of Figure 5.1 show the (1) density, (2) pressure, (3) velocity, (4) heat flux, (5) kurtosis, and (6) the condition on \( r \) for moment-realizability. Counting the waves from left to right we have a 1-rarefaction, 2-shock, 3-shock, 4-rarefaction, and 5-shock. In Figure 5.2 we show (1) \( \mu_1 \), (2) \( u \), (3) \( \mu_3 \), (4) \( \omega_1 \), (5) \( \omega_2 \), and (6) \( \omega_3 \) for the same initial conditions. A full list of parameters can be found in Appendix B.

In Figures 5.3 and 5.4 we show a shock tube problem for the tri-delta system with initial states

\[
(\rho, u, p, q, r)_L = (1.0, -0.7, 1.5, 1.5, 5.5), \quad (5.37)
\]
\[
(\rho, u, p, q, r)_R = (0.5, -0.9, 1.0, 1.0, 4.0). \quad (5.38)
\]

The panels of Figure 5.3 show the (1) density, (2) pressure, (3) velocity, (4) heat flux, (5) kurtosis, and (6) the condition on \( r \) for moment-realizability. Counting the waves from left to right we have a 1-shock, 2-shock, 3-shock, 4-rarefaction, and 5-shock. In Figure 5.4 we show (1) \( \mu_1 \), (2) \( u \), (3) \( \mu_3 \), (4) \( \omega_1 \), (5) \( \omega_2 \), and (6) \( \omega_3 \) for the same initial conditions. A full list of parameters can be found in Appendix B.
Figure 5.1: A shock tube problem for the tri-delta system. In this example the initial states are $(\rho, u, p, q, r)_L = (1.5, -0.5, 1.5, 1.0, 4.5)$ and $(\rho, u, p, q, r)_R = (1.0, -0.5, 1.0, 0.5, 3.0)$. The panels show (1) density, (2) pressure, (3) velocity, (4) heat flux, (5) kurtosis, and (6) condition on $r$ for moment-realizability. Counting the waves from left to right we have a 1-rarefaction, 2-shock, 3-shock, 4-rarefaction, and 5-shock. A full list of parameters can be found in Appendix B.
Figure 5.2: A shock tube problem for the tri-delta system. In this example the initial states are $(\rho, u, p, q, r)_\ell = (1.5, -0.5, 1.5, 1.0, 4.5)$ and $(\rho, u, p, q, r)_r = (1.0, -0.5, 1.0, 0.5, 3.0)$. The panels show (1) $\mu_1$, (2) $u$, (3) $\mu_3$, (4) $\omega_1$, (5) $\omega_2$, and (6) $\omega_3$. A full list of parameters can be found in Appendix B.
Figure 5.3: A shock tube problem for the tri-delta system. In this example the initial states are $(\rho, u, p, q, r)_L = (1.0, -0.7, 1.5, 1.5, 5.5)$ and $(\rho, u, p, q, r)_R = (0.5, -0.9, 1.0, 1.0, 4.0)$. The panels show (1) density, (2) pressure, (3) velocity, (4) heat flux, (5) kurtosis, and (6) the condition on $r$ for moment-realizability. Counting the waves from left to right we have a 1-shock, 2-shock, 3-shock, 4-rarefaction, and 5-shock. A full list of parameters can be found in Appendix B.
Figure 5.4: A shock tube problem for the tri-delta system. In this example the initial states are 
$(\rho, u, p, q, r)_l = (1.0, -0.7, 1.5, 1.5, 5.5)$ and $(\rho, u, p, q, r)_r = (0.5, -0.9, 1.0, 1.0, 4.0)$. The panels 
show (1) $\mu_1$, (2) $u$, (3) $\mu_3$, (4) $\omega_1$, (5) $\omega_2$, and (6) $\omega_3$. Counting the waves from left to right we 
have a 1-shock, 2-shock, 3-shock, 4-rarefaction, and 5-shock. A full list of parameters can be 
found in Appendix B.
CHAPTER 6. CONCLUSIONS

In this work we studied a particular strategy for closing the moment hierarchy: quadrature-based moment-closures. In particular, we reviewed existing approaches that close the moment hierarchy by assuming that the underlying distribution is the sum of two delta functions, two Gaussian distributions, or two B-splines. Next we developed a closure based on three delta functions (tri-delta), where one of the delta functions is located at a prescribed location. This leads to a Gauss-Radau-type quadrature rule. We derived exact formulas that relate the positions and weights of the three delta functions to the primitive variables: mass density, velocity, pressure, heat flux, and kurtosis. We also derived exact conditions that simultaneously guarantee that the underlying system of partial differential equations remain hyperbolic and that the inversion problem from primitive variables to Gauss-Radau quadrature weights and points is solvable. Furthermore, we proved that the region in solution space for which these conditions are satisfied is convex. Finally, we developed a high-order discontinuous Galerkin finite element method to solve this system with a moment-realizability limiter that guarantees that the numerical solution remains in this convex hyperbolic/moment-realizable region.
BIBLIOGRAPHY


[27] R. Fox, private communication.

APPENDIX A. EXCERPT FROM DOGPACK

Constraint-preserving code

The following code is the file EnforcePositivity.cpp located in the DOGPACK software package [13] in the following directory:

    $DOGPACK/apps/1d/QMOM_3deltas/lib.

The following code enforces the constraints that ensure the underlying system of partial differential equations remain hyperbolic and that the inversion problem from primitive variables to Gauss-Radau quadrature weights and points is solvable.

```cpp
#include "DogArrays.h"
#include "DogParams.h"
#include "DogMesh.h"
#include "DogBasis.h"
#include "AppData.h"
#include "assert.h"

inline double get_pressure(const double& rho, const double& mom, const double& energy)
{
    return energy - pow(mom,2)/rho;
};
```
inline double get_r_cond(const double& rho,
    const double& mom,
    const double& energy,
    const double& q4,
    const double& q5)
{
    return (pow(energy,3) - 2*mom*energy*q4 + rho*pow(q4,2)
            + pow(mom,2)*q5 - rho*energy*q5)/(pow(mom,2) - rho*energy) ;
}

// This is a user-supplied routine that enforces positivity (or maximum-principle)
// -------------------------------------------------------------------------------
void EnforcePositivity(const double& time,
    const DogParams* dgParams,
    const DogMesh* dgMesh,
    const DogBasis* dgBasis,
    AppData* appData,
    DblArray* qsoIn)
{
    // get some important constants
    assert_eq(dgParams->get_num_dims(),1);
    assert_eq(dgBasis->get_BasisType(),legcart1);

    const int space_order = dgParams->get_space_order();
    if (space_order<=1)
        { return; }
```c++
const int time_order = dgParams->get_time_order();
if (time_order>4)
    { return; }

const int num_elems = qsoln->get_ind_length(1);
const int num_eqns = qsoln->get_ind_length(2);
assert_eq(num_eqns,5);
const int num_basis_cmpts = qsoln->get_ind_length(3);
const int num_pos_points = dgBasis->get_num_pos_points(space_order);

// grab the minimum and maximum values
const double min_val = 1.0e-10;

// set maximum of absolute value of Legendre basis functions
DblArray max_phi(num_basis_cmpts);
dgBasis->set_max_basis_values(max_phi);

// set positivity points
DblArray pos_points(num_pos_points,1);
dgBasis->set_positivity_points(pos_points);

// set value of Legendre basis functions at positivity points
DblArray phi_at_pos_points(num_pos_points,
    num_basis_cmpts);
dgBasis->set_basis_at_points(num_pos_points,
    num_basis_cmpts,
    pos_points,
    phi_at_pos_points);
```
// enforce constraints
for (int i=1; i<=num elems; i++)
{
    // Store average value
    const double rho_av = qsoln->get(i,1,1);
    const double press0 = get_pressure(rho_av,
    qsoln->get(i,2,1),
    qsoln->get(i,3,1));

    const double r_cond0 = get_r_cond(rho_av,
                    qsoln->get(i,2,1),
                    qsoln->get(i,3,1),
                    qsoln->get(i,4,1),
                    qsoln->get(i,5,1));

    if (rho_av>min_val && press0>min_val && r_cond0>min_val)
    {
        // -------------------
        // Part A: Density
        // -------------------

        // Compute density at each positivity point
        DblArray rho_values(num_pos_points);
        rho_values.setall( rho_av );

        for (int j=1; j<=num_pos_points; j++)
            for (int m=2; m<=num basis_cmpts; m++)
            {
            rho_values.fetch(j) += qsoln->get(i,1,m)*phi_at_pos_points.get(j,m);
\}

// min density

double rho_values_min = rho_values.get(1);
for (int j=2; j<=num_pos_points; j++)
{
    rho_values_min = dog_math::Min(rho_values_min, rho_values.get(j));
}

// if min value is smaller than min_val, must limit
if (rho_values_min<min_val)
{
\    // Calculate theta
\    const double frac = fabs(rho_av - min_val)/
\        (fabs(rho_av - rho_values_min)+1.0e-10);
\    const double theta = dog_math::Min( 1.0, frac );
\}

\    // Limit solution
\    for (int meq=1; meq<=num_eqns; meq++)
\    for (int mb=2; mb<=num_basis_cmpts; mb++)
\    {
\        const double tmp = qsoln->get(i,meq,mb);
\        qsoln->fetch(i,meq,mb) = theta*tmp;
\    }
\}

\} // -------------------
\} // Part B: Pressure

\} // Compute density, momentum, energy, and pressure
\} // values at each positivity point
DblArray mom_values(num_pos_points);
DblArray energy_values(num_pos_points);

rho_values.setall(qsoln->get(i,1,1));
mom_values.setall(qsoln->get(i,2,1));
energy_values.setall(qsoln->get(i,3,1));

for (int j=1; j<=num_pos_points; j++)
    for (int m=2; m<=num_basis_cmpts; m++)
    {
        rho_values.fetch(j) += qsoln->get(i,1,m)*phi_at_pos_points.get(j,m);
        mom_values.fetch(j) += qsoln->get(i,2,m)*phi_at_pos_points.get(j,m);
        energy_values.fetch(j) += qsoln->get(i,3,m)*phi_at_pos_points.get(j,m);
    }

DblArray press_values(num_pos_points);
for (int j=1; j<=num_pos_points; j++)
{
    press_values.fetch(j) = get_pressure(rho_values.get(j),
                                          mom_values.get(j),
                                          energy_values.get(j));
}

// min pressure
double press_values_min = press_values.get(1);
for (int j=2; j<=num_pos_points; j++)
{
    press_values_min = dog_math::Min(press_values_min, press_values.get(j));
}

// if min value is smaller than min_val, must limit
if (press_values_min<min_val)
{
    // Calculate theta
    const double frac = fabs(press0 - min_val) /
    (fabs(press0 - press_values_min)+1.0e-10);
    const double theta = dog_math::Min( 1.0, frac );

    // Limit solution
    for (int meq=1; meq<=num_eqns; meq++)
        for (int mb=2; mb<=num_basis_cmpts; mb++)
            {
                const double tmp = qsoln->get(i,meq,mb);
                qsoln->fetch(i,meq,mb) = theta*tmp;
            }

    // ---------------
    // Part C: R Condition
    // ---------------

    // Compute density, momentum, energy, and pressure
    // values at each positivity point
    DblArray q4_values(num_pos_points);
    DblArray q5_values(num_pos_points);

    rho_values.setall(qsoln->get(i,1,1));
    mom_values.setall(qsoln->get(i,2,1));
    energy_values.setall(qsoln->get(i,3,1));
    q4_values.setall(qsoln->get(i,4,1));
    q5_values.setall(qsoln->get(i,5,1));
for (int j=1; j<=num_pos_points; j++)
    for (int m=2; m<=num_basis_cmpts; m++)
    {
        rho_values.fetch(j) += qsoln->get(i,1,m)*phi_at_pos_points.get(j,m);
        mom_values.fetch(j) += qsoln->get(i,2,m)*phi_at_pos_points.get(j,m);
        energy_values.fetch(j) += qsoln->get(i,3,m)*phi_at_pos_points.get(j,m);
        q4_values.fetch(j) += qsoln->get(i,4,m)*phi_at_pos_points.get(j,m);
        q5_values.fetch(j) += qsoln->get(i,5,m)*phi_at_pos_points.get(j,m);
    }

DblArray r_cond_values(num_pos_points);
for (int j=1; j<=num_pos_points; j++)
    {
        r_cond_values.fetch(j) = get_r_cond(rho_values.get(j),
            mom_values.get(j),
            energy_values.get(j),
            q4_values.get(j),
            q5_values.get(j));
    }

// min r_cond
double r_cond_values_min = r_cond_values.get(1);
for (int j=2; j<=num_pos_points; j++)
    {
        r_cond_values_min = dog_math::Min(r_cond_values_min, r_cond_values.get(j));
    }

// if min value is smaller than min_val, must limit
if (r_cond_values_min<min_val)
    {
        // Calculate theta
const double frac = fabs(r_cond0 - min_val)/
(fabs(r_cond0 - r_cond_values_min)+1.0e-10);
const double theta = dog_math::Min( 1.0, frac );

// Limit solution
for (int meq=1; meq<=num_eqns; meq++)
for (int mb=2; mb<=num_basis_cmpts; mb++)
{
    const double tmp = qsoln->get(i,meq,mb);
    qsoln->fetch(i,meq,mb) = theta*tmp;
}
}

else
{
    // Limit solution
    for (int me=1; me<=num_eqns; me++)
    for (int mb=2; mb<=num_basis_cmpts; mb++)
    {
        qsoln->fetch(i,me,mb) = 0.0;
    }
}
APPENDIX B. PARAMETERS FOR FIGURES 5.1-5.4

The following is an excerpt from the DOGPACK software package [13]. Shown here are the parameters used in Figures 5.1–5.4. This code can be found in the file parameters.ini located in the following directory

$DOGPACK/apps/1d/QMOM_3deltas/shock_tube.

Parameters for Figure 5.1/5.2

; Parameters common to all DoGPack applications
[dogParams]
defaults_file = "$DOGPACK/config/dogParams_defaults.ini"

num_dims = 1 ; number of dimensions
mesh_type = cartesian ; (either cartesian or unstructured)
number_of_output_frames = 10 ; number of output times to plot output
final_time = 0.4 ; final time
dt_initial = 0.02 ; initial time step (will be overwritten if too large)
dt_max = 1.0 ; max allowable time step (will be overwritten if too large)
cfl_max = 0.2 ; max allowable Courant number (must be >= cfl_target)
cfl_target = 0.18 ; desired Courant number
max_number_of_time_steps = 500000 ; max number of time steps
time_stepping_method = runge-kutta ; (see documentation for options)
basis_type = space-legendre ; (see documentation for options)
limiter_method = moment ; (see documentation for options)
limiter_parameter = 50.0 ; value of 'alpha' for 'bounds' limiter
                      (otherwise ignored)
riemann_solver = rusanov ; (see documentation for options)
space_order = 3 ; order of accuracy in space
time_order = 3 ; order of accuracy in time
use_limiter = 1 ; use limiter (1-yes, 0-no)
enforce_positivity = 1 ; enforce positivity (aka maximum-principle)
                      (1-yes, 0-no)
verbosity = 1 ; verbosity of output (1-yes, 0-no)
source_term = 0 ; source term (1-yes, 0-no)
num_eqns = 5 ; number of equations
restart_frame = -1 ; (neg==off; 9998 => q9998.dat, a9998.dat,
t9998.ini)
datafmt = 1 ; 1 for ascii, 5 for hdf5.
ic_quad_order = -100 ; quadrature for L2-projection of
                     initial conditions

; Parameters for mesh
[mesh]
xlow = -1.2e0 ; left end point
xhigh = 1.2e0 ; right end point
left_bc = extrapolation ; left boundary condition
                   (extrapolation, periodic, wall, user-defined)
right_bc = extrapolation ; right boundary condition
              (extrapolation, periodic, wall, user-defined)
; for 'wall' boundary condition
left_bc_how_many_cmpts_to_negate = -1 ; how many components should be negated at left boundary condition
left_bc_which_cmpts_to_negate = -1 ; which components should be negated at left boundary condition
right_bc_how_many_cmpts_to_negate = -1 ; how many components should be negated at right boundary condition
right_bc_which_cmpts_to_negate = -1 ; which components should be negated at right boundary condition

; Parameters for initial condition
[initialParams]
shock_location = 0.0 ; shock location
left_state = 1.5, -0.5, 1.5, 1.0, 4.5 ; rhoL,UL,PL,QL,RL
right_state = 1.0, -0.5, 1.0, 0.5, 3.0 ; rhoR,UR,PR,QR,RR

Parameters for Figure 5.3/5.4

; Parameters common to all DoGPack applications
[dogParams]
defaults_file = "$DOGPACK/config/dogParams_defaults.ini"
num_dims = 1 ; number of dimensions
mesh_type = cartesian ; (either cartesian or unstructured)
number_of_output_frames = 10 ; number of output times to plot output
final_time = 0.4 ; final time
dt_initial = 0.02 ; initial time step (will be overwritten if too large)
dt_max = 1.0 ; max allowable time step (will be overwritten if too large)
cfl_max = 0.2 ; max allowable Courant number
cfl_target = 0.18 ; desired Courant number (must be >= cfl_target)
max_number_of_time_steps = 500000 ; max number of time steps

time_stepping_method = runge-kutta ; (see documentation for options)
basis_type = space-legendre ; (see documentation for options)
limiter_method = moment ; (see documentation for options)
limiter_parameter = 50.0 ; value of 'alpha' for 'bounds' limiter (otherwise ignored)

riemann_solver = rusanov ; (see documentation for options)
space_order = 3 ; order of accuracy in space
time_order = 3 ; order of accuracy in time
use_limiter = 1 ; use limiter (1-yes, 0-no)
enforce_positivity = 1 ; enforce positivity (aka maximum-principle) (1-yes, 0-no)

verbosity = 1 ; verbosity of output (1-yes, 0-no)
source_term = 0 ; source term (1-yes, 0-no)
num_eqns = 5 ; number of equations
restart_frame = -1 ; (neg==off; 9998 => q9998.dat, a9998.dat, t9998.ini)
datafmt = 1 ; 1 for ascii, 5 for hdf5.
ic_quad_order = -100 ; quadrature for L2-projection of initial conditions

; Parameters for mesh

[mesh]
mx = 1500 ; number of grid elements in x-direction
xlow = -1.2e0 ; left end point
xhigh = 1.2e0 ; right end point
left_bc = extrapolation ; left boundary condition
right_bc = extrapolation ; right boundary condition

; for 'wall' boundary condition
left_bc_how_many_cmpts_to_negate = -1 ; how many components should be
negated at left boundary condition
left_bc_which_cmpts_to_negate = -1 ; which components should be
negated at left boundary condition
right_bc_how_many_cmpts_to_negate = -1 ; how many components should be
negated at right boundary condition
right_bc_which_cmpts_to_negate = -1 ; which components should be
negated at right boundary condition

; Parameters for initial condition
[initialParams]
shock_location = 0.0 ; shock location
left_state = 1.0, -0.7, 1.5, 1.5, 5.5 ; rhol,ul,pl,quel,rl
right_state = 0.5, -0.9, 1.0, 1.0, 4.0 ; rhor,ur,pr,quer,rr