High order shock capturing methods with compact stencils for use with adaptive mesh refinement and mapped grids

Scott Moe

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2017

Reading Committee:

Randall J. LeVeque, Chair

Ulrich L. Hetmaniuk

Peter Mackenzie-Helnwein

James A. Rossmanith

Program Authorized to Offer Degree:
Applied Mathematics

University of Washington

Abstract

High order shock capturing methods with compact stencils for use with adaptive mesh refinement and mapped grids

Scott Moe

Chair of the Supervisory Committee:
Professor Randall J. LeVeque
Applied Mathematics

This thesis focuses on several developments toward creating a high order shock capturing method that can be used on mapped grids with block-structured adaptive mesh refinement (AMR). The discontinuous Galerkin (DG) method is used as a starting point for the construction of this method.

A high order mapped grid DG scheme is implemented and tested on several hyperbolic PDEs. It is shown that even on highly-skewed meshes these DG schemes can illustrate high order convergence. Additionally a family of limiters is developed that is extremely flexible with respect to geometry. This flexibility originates from the fact that these limiters use a minimal stencil and do not require directional information. The performance of this family of limiters is explored on structured, unstructured and mapped grids.

Lax-Wendroff time stepping schemes have a very compact stencil and they can easily be used with local time stepping because they produce a local space-time Taylor series of the solution. A positivity limiter is developed to allow the use of high order Lax-Wendroff time stepping on PDEs, such as the Euler equations, that require the positivity of pressure and density. Additionally a new type of Lax-Wendroff time stepping, known as the differential transform method, is adapted to both a WENO finite difference method and DG. The differential transform method uses tools from the automatic differentiation literature to automate the computation of space-time Taylor series. A high order DG scheme using the differential transform method is developed to use block-structured AMR and local time stepping. This method is implemented in one dimension and found to be very effective at maintaining the accuracy of the high order DG method while reducing its computational cost.

The accuracy and convergence rates of the methods developed in this thesis are established by comparing to analytical or very highly refined numerical solutions. All of the methods developed, with the exception of the positivity limiter, are tested on the advection equations, the acoustic equations and the Euler equations on a variety of standard test problems found in the literature.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to express my gratitude to all of the people who have made this thesis possible. First I would like to thank my co-advisors Randy LeVeque and James Rossmanith for there continuous encouragement and guidance. Randy has always been supremely helpful, supportive and patient throughout my whole graduate career. He taught me so much and I could not have made it here without his help. James introduced me to computational mathematics in the first place, and has continued to provide guidance or commiseration whenever I have faced challenges. I am very grateful for all of the hours both of them have put into teaching and advising me throughout my academic career.

I am deeply indebted to David Seal at the US Naval Academy for years of friendship and collaboration. He has helped me in too many ways to list. I especially appreciate our many expansive conversations, they have usually led to good ideas and a great deal of learning.

I would also like to thank Ulrich Hetmaniuk, Peter Mackenzie-Helnwein and Uri Shumlak for providing feedback and helpful comments at different stages of this work, and for sacrificing their time to serve on my committee.

Thank you to the many professors I have had over the years who inspired my curiosity, especially the professors in the Applied Mathematics department. I have learned a great deal from many of you. Also thank you to Lauren Lederer for helping me countless times to meet deadlines (many that I had forgotten about).

Thank you to my peers in the Applied Mathematics department for the many years of friendship. A special thanks to Devin Light, Donsub Rim, Jakob Kotas and Niket Thakkar for all of the good times and the entertaining and often enlightening discussions.

Finally I am extremely grateful to my family. Thank you to my mother and father for always supporting me in any way possible in all of my en devours leading up to this point. And thank you to my sister Olivia for always being there to talk to. Lastly I would like to express my immeasurable gratitude to my wife Chelsee for always being there through this long journey. I would have gotten nowhere without your love.

DEDICATION

To Chelsee.

Chapter 1

INTRODUCTION

This work represents a sequence of projects focused on developing, piece by piece, a compact stencil high order shock-capturing method to model hyperbolic PDEs. The term compact in this case is meant to convey that the computational stencil used to update a given degree of freedom in the method consists only of its direct neighbors. There are several reasons that this is desirable, which will be established by way of a brief overview of the current dominant shock-capturing methods. The contending methods are finite volume schemes, weighted essentially non-oscillatory (WENO) schemes, and discontinuous Galerkin schemes with either Runge-Kutta or Lax-Wendroff time stepping. We will outline the comparative advantages and disadvantages of each of these methods.

1.1 Finite volume methods

Finite volume methods are a broad class of methods that can be derived from the integral form of conservation laws [79]. The canonical finite volume solver is Godunov's method [49, 104]. In Godunov's method the spatial domain is divided up into cells and on each cell the solution is approximated by a constant value. The integral form of a hyperbolic conservation law evaluated on a cell involves surface integrals evaluated on the faces of the cell. In the finite volume method these integrals are evaluated using a numerical flux function that is essentially an upwind flux computed by solving a Riemann problem or using an approximate Riemann solver. The Godunov method is robust, but it is at most first order accurate (in regions where the solution is smooth, less so elsewhere). However, it has become the base of many higher order numerical methods for solving hyperbolic PDEs.

The Clawpack software package [30] is built on top of the wave propagation form of Godunov's method [49]. Unlike flux-differencing schemes, the wave propagation form of Godunov's method can automatically handle non-conservative hyperbolic PDES where no flux function exists.

Clawpack creates a second order approximation of derivative terms using information from neighboring cells. This approximation, combined with a higher order Lax-Wendroff time-stepping procedure, allows Clawpack to be provably second order on smooth problems. Additionally, because Clawpack's reconstruction can be defined from a cell's immediate neighbors (with the limiters requiring an additional cell on each side), the stencil for Clawpack is extremely compact. This, combined with the the use of Riemann solvers to

compute upwind waves, often allows it to achieve second order convergence even when an interface causes the solution to become locally non-differentiable.

Due to Godunov's theorem the second order method must be nonlinear in some way in order to avoid spurious oscillations. Clawpack introduces this nonlinearity through wave limiters, which essentially adjust the reconstructed slope value in order to clamp down on oscillations when they appear. Limiters have been an important area of study for many years now and the limiters for second order methods are currently very well developed and understood [79].

Because Clawpack has evolved to be used with block-structured adaptive mesh refinement, it always assumes a structured grid. It is possible to handle geometry with the structured grid solver by creating a mapping function to map a structured grid to a quadrilateral (in 2D) or hexahedral (in 3D) grid. The resultant mesh is still logically Cartesian despite no longer being composed of regular cells [22, 10, 21]. This is a common technique in numerics, but typically the mapping function must be very smooth in order to achieve high order convergence. However because of its compact stencil (and some of its other features), Clawpack has performed well even when the mapping function is not smooth. Clawpack has been extensively tested on a family of very difficult square to circle mappings that create cells that are nearly triangular. The mappings that produce these cells are not smooth. It has been observed that Clawpack does have its accuracy truncated at these cells, but that generally first order convergence is still observed [76]. Typically when implemented with a less extreme mapping function convergence rates very near second order are observed. Additionally, Clawpack has been implemented as the PDE solving kernel of a block-structured adaptive mesh refinement scheme that can use local time-stepping. Because of all of these features, Clawpack is widely used and robust.

1.2 Weighted essentially non-oscillatory (WENO) methods

Clawpack is limited to second order because its reconstruction is only linear. One option to create a higher order method is to create a higher order reconstruction by interpolating more cell average values. Unfortunately interpolating across a shock will produce spurious oscillations. In order to avoid this, WENO schemes introduce a weighted combination of stencils where the weighting is set, using smoothness indicators, in order to avoid interpolating across non-smooth regions of the solution [116].

This is a very clever idea and it works well. However, this means that high order WENO schemes have very wide stencils. Various WENO implementations exist but the most common forms are probably the method of lines based methods that use Runge-Kutta time stepping. A version of this type of WENO method is implemented in PyClaw [63, 64] so any Clawpack user can test these methods out and compare them to Clawpack very easily. The other large family of WENO schemes contains the Lax-Wendroff based ADER schemes

[125]. One disadvantage of the typical ADER scheme has been that for nonlinear PDEs it requires explicit computation of Jacobians and Hessians of the flux functions (and possibly even higher order derivative terms). This is very cumbersome, requiring a new code every time one would like to change the order.

More recently a new Lax-Wendroff WENO scheme has been introduced that uses ideas from automatic differentiation to automatically compute all of the time-derivative terms needed for high-order Lax Wendroff schemes. This technique is known as a differential transform [89, 90]. The properties of this family of schemes will match other Lax-Wendroff schemes except that these methods can be implemented to arbitrary order automatically. Chapter 5 will discuss the differential transform method in detail. Chapter 6 will illustrate the how the differential transform can be used to solve hyperbolic PDEs using a WENO finite difference scheme.

Because of their wide stencils, WENO schemes are prone to losing accuracy in some situations where methods with a more compact stencil would not. For example it has been shown that on any problem with interfaces at which the solution loses regularity, the convergence rate of a WENO scheme can be truncated [65]. Additionally when WENO is used with a mapping function, if the mapping function is not smooth, its convergence rate will be severely impacted. In [51] the authors attempted to use a non-smooth mapping function that Clawpack has been tested on and found the high order WENO schemes were truncated to first order.

1.3 Discontinuous Galerkin methods

1.3.1 Runge-Kutta DG

Discontinuous Galerkin schemes can be thought of as another extension of the Godunov method to higher order. These methods work by defining a high order basis on each cell and performing a Galerkin projection onto this basis. These methods first became widely known through a series of papers by Cockburn and Shu [33, 36, 31, 32]. The discontinuous Galerkin method has the same stencil (defined in terms of cells) as the Godunov method. Typically DG is used in a method of lines (MOL) formulation with Runge-Kutta time-stepping. This method avoids many of the disadvantages of the WENO methods discussed above.

Given that this method possesses a high-order representation on each cell it can achieve high-order as long as the solution is sufficiently smooth inside every cell. In particular that means it handles non-smooth mappings and discontinuous coefficients extremely well. But the method still has a handful of disadvantages.

The Runge-Kutta schemes typically used with DG are often required to be low-storage (for computational reasons) and also strong stability preserving (SSP). SSPRK schemes are Runge-Kutta schemes where each stage has the form of a forward Euler time-step. Because the stages take this simple form it is possible to prove that these methods possess certain

TVD/TVB properties for small enough time steps [50, 68]. Also, because of the special form these RK methods take, it is possible to show that using this type of time-stepping scheme can preserve the positivity of the cell average values of density and pressure in the Euler equations modeled with a DG scheme using an appropriate numerical flux [142]. These methods typically end up using many stages. For example a widely used 4th order low-storage SSP Runge-Kutta method has 10 stages [62]. This doesn't affect the methods applicability to any particular problems, but it does introduce extra communications that may be costly when parallelizing the scheme. Additionally it can be shown that there are no traditional SSPRK methods of order higher than 4 [108].

Another problem with Runge-Kutta methods is that one cannot use a typical Runge-Kutta scheme to obtain an accurate approximation of the solution at any time in-between the times t^n and t^{n+1}. This is normally not an issue but when using adaptive mesh refinement it is beneficial to be able to use local time-stepping (LTS) so that much larger cells can take much larger time-steps. This typically requires interpolating in time on the coarse cell to obtain a flux to pass to a fine cell. There is a class of methods known as continuous extension Runge-Kutta (CERK) methods, that produce an accurate interpolation polynomial after completing each Runge-Kutta step. CERK methods have been used to implement local time-stepping with RKDG [48]. However these schemes are not SSP and they are not low storage and they also require even more stages than a normal Runge-Kutta method.

1.3.2 Lax-Wendroff DG

A truly high-order numerical method must use a high order spatial discretization and also advance in time using a high order scheme. RK methods are one way to construct a high order accurate approximation of a Taylor series in time, however RK schemes have certain disadvantages pointed out in Section 1.3.1. They require multiple stages (possibly very many stages depending on if you require SSP or CERK schemes) introducing more communication and a wider effective computational stencil. They are difficult to implement with LTS, and they would be very difficult to implement with LTS for PDEs that require certain positivity constraints (fluids, MHD, etc.).

A Lax-Wendroff DG (Lax-DG) scheme produces a high order approximation of a Taylor series in time by using the Cauchy-Kowalevski procedure to explicitly compute terms in the series from the solution at a given time. The DT-DG method (which is a type of Lax-DG scheme) approximates terms of a Taylor series in time using series expansion techniques commonly known in the automatic differentiation literature [131]. In the author's view Lax-Wendroff DG using differential transforms (we will call it DT-DG) solves many of the problems associated with RKDG without losing DG's advantages. DT-DG still has an extremely compact stencil (if anything it is more compact). It still handles interfaces, and should perform well on mapped grids. The method is single-stage single-step, and when it is implemented it automatically produces a Taylor series expansion that can be evaluated

at any time when performing local time-stepping. Additionally we will introduce a limiter in this thesis that can modify the DT-DG solution to give it the properties that are usually guaranteed by using an SSPRK method. This limiter will work for arbitrary order. Unlike Runge-Kutta schemes or typical Lax-Wendroff schemes the DT-DG method will be truly arbitrary order in time and space. Once the code has been written for a specific PDE it can be implemented to any order in time.

To summarize, the DT-DG method:

- Is single-stage single-step;
- Is arbitrary order in space and time;
- Has a compact stencil which should mean good performance on mapped grids;
- Can achieve high-order on interface problems;
- Is easy to implement with local time-stepping;
- Uses minimal communications;
- Has the same stencil as Clawpack (modulo the limiter used).

1.4 Limiters

There have been many high-order limiters developed for DG over the years with a wide variety of properties. Principally these limiting schemes can be broken into three categories: slope limiters, WENO limiters and artificial viscosity limiters. Slope limiters tend to be very specialized to each geometry and it is not clear how to implement them for methods that are higher than 3rd order. The WENO limiters have wide stencils and involve tunable parameters that must be set by trial and error. The artificial viscosity limiters reduce the maximum time step allowed and this reduction gets worse as the order is increased. Much more will be said about the current state of the limiter literature in Chapter 4.

In Chapter 4 we develop a limiter that is a hybrid between a slope limiter and a WENO limiter. It will work on arbitrary geometry but it has a compact stencil like a slope limiter. However the limiter will have a tunable parameter that serves the same purpose as the tunable parameter found in the WENO limiters.

In Chapter 7 we introduce a positivity preserving limiter for the Euler equations modeled with Lax-DG. This limiter is necessary because when using Lax-DG schemes one can no loner rely on the stability properties associated with the SSPRK methods to maintain positivity of cell average values of pressure or density. Unlike SSPRK methods this limiter will preserve positivity for any order DG scheme and for any numerical flux function.

1.5 Adaptive mesh refinement

In this work when we say AMR we are implicitly referring to block-structured adaptive mesh refinement (AMR) as described in Chapter 8.2. AMR sometimes seems conceptually simple, but in practice it has many important complications and can be much harder than expected to implement. The AMRClaw software package has an AMR scheme that is implemented to use Clawpack's stencil. The DT-DG method has exactly the same stencil as Clawpack's finite volume method and so it has been possible to re-use much of AMRClaw's code to implement the DT-DG method with AMR and local time-stepping in 1-d, whereas other high-order DG or WENO methods would not be compatible with AMRCLAW without much more extensive modifications. This will be discussed in Chapter 8. The ideas developed in Chapter 8 should extend to more space dimensions, but the required modifications to the AMR code have not yet been implemented.

Chapter 2

HYPERBOLIC PDES AND DISCONTINUOUS GALERKIN METHODS

2.1 Hyperbolic PDEs and the Clawpack software package

Many of the algorithms that will be discussed in this thesis borrow heavily from algorithms designed to be used with second order finite volume methods using wave limiters as found in the Clawpack software package [30]. In this section we briefly introduce hyperbolic PDEs and describe the algorithms used in Clawpack. We will give only a very brief introduction here. The interested reader is directed to [79] for a very thorough discussion.

2.1.1 Hyperbolic PDEs

Clawpack is designed to solve hyperbolic PDEs in conservative

$$q_t + f(q)_x + g(q)_y = 0, \tag{2.1}$$

or non-conservative

$$q_t + A(x,y)q_x + B(x,y)q_y = 0,$$

form. Consider one dimensional problems of the form in Equation (2.1)

$$q_t + f(q)_x = 0. \tag{2.2}$$

In order for this equation to be hyperbolic it must be true that the Jacobian matrix $f^{'}(q) = \frac{\partial f}{\partial q}$ is diagonalizable with real eigenvalues. The fact that this diagonalization is possible means that one can always perform the decomposition

$$f^{'}(q) := A = R\Lambda R^{-1},$$

where Λ is the diagonal matrix of real eigenvalues of A and R is the matrix of eigenvectors. Given the definition $\omega = R^{-1}q$ one can re-write the linearized version of Equation (2.2) as

$$\omega_t + \Lambda\omega_x = 0. \tag{2.3}$$

The system can be locally decomposed into separate non-interacting advection equations on the variables ω which are known as characteristic variables.

2.1.2 A word on units

The PDEs that will be presented in this work will be non-dimensionalized. What is meant by this is that the variables being solved for will be dimensionless until multiplied by an appropriate scaling. For example a distance variable x would be written as $x = r_0\tilde{x}$ where r_0 is a positive constant that has units of meter and $\tilde{x}$ is unit-less. Similarly time would be written as $t = t_0\tilde{t}$ where t_0 is a positive constant with units of second and $\tilde{t}$ is unit-less and a density would be written as $\rho = \frac{m_0}{r_0^3}\tilde{\rho}$ where m_0 has units of kilogram. The equations presented in this thesis will actually evolve the dimensionless variables like $\tilde{\rho}$, thus units will not be discussed. Since all of the variables are non-dimensionalized, the tildes will be dropped and variables will just be referred to as if they were the corresponding quantity with units (density instead of non-dimensionalized density for example). This is standard practice in numerical analysis.

2.1.3 Finite volume methods

Let us assume that we are on a 2d space-time domain decomposed into a uniform grid so that there are cells centered at each point $(x,t) = (i\Delta x, j\Delta t)$ for $(i,j) \in \mathbb{Z}\times\mathbb{Z}$. The Godunov method for this equation is

$$Q_i^{n+1} = Q_i^n - \frac{\Delta t}{\Delta x}\left[F(Q_i^n, Q_{i+1}^n) - F(Q_{i-1}^n, Q_i^n)\right]. \tag{2.4}$$

Here Q_i^n represents the average of the solution vector q on a cell i of width Δx at time step n (so at time $\Delta t n + t_0$). $F(Q_{i-1}^n, Q_i^n)$ is a numeral approximation to the value

$$F(Q_{i-1}^n, Q_i^n) \approx \frac{1}{\Delta t}\int_{t^n}^{t^{n+1}} f\left(q(x_{i-\frac{1}{2}})\right).$$

This is the simplest scheme that can be used to solve hyperbolic PDEs. In Clawpack the methods actually work in wave propagation form. Let us define r^p as one of the m eigenvectors of the matrix $f'(q)$. Then one can write

$$Q_i - Q_{i-1} = \sum_{p=1}^{m}\alpha_{i-\frac{1}{2}}^p r^p = \sum_{p=1}^{m}\mathcal{W}_{i-\frac{1}{2}}^p.$$

In particular, if one thinks of approximating $F(Q_{i-1}^n, Q_i^n)$ by $F(Q^m)$ or AQ^m, Q^m can be written as $Q^m = Q_{i-1} + \sum_{p=1}^{m'}\mathcal{W}_{i-\frac{1}{2}}^p$ for some m'. Each eigenvector has an associated eigenvalue (wave speed s) that determines which cell (the left or right cell) it contributes to. One can then write the method as Equation (2.5).

$$Q_i^{n+1} = Q_i^n - \frac{\Delta t}{\Delta x}\left[\sum_{p=1}^{m}(s_{i-\frac{1}{2}}^p)^+\mathcal{W}_{i-\frac{1}{2}}^p + \sum_{p=1}^{m}(s_{i+\frac{1}{2}}^p)^-\mathcal{W}_{i+\frac{1}{2}}^p)\right]. \tag{2.5}$$

There are higher resolution correction terms that can be added to this in order to achieve second order to produce

$$Q_i^{n+1} = Q_i^n - \frac{\Delta t}{\Delta x}\left[\sum_{p=1}^{m}(s^p_{i-\frac{1}{2}})^+\mathcal{W}^p_{i-\frac{1}{2}} + \sum_{p=1}^{m}(s^p_{i+\frac{1}{2}})^-\mathcal{W}^p_{i+\frac{1}{2}})\right] - \frac{\Delta t}{\Delta x}\left(\tilde{F}_{i+\frac{1}{2}} - \tilde{F}_{i-\frac{1}{2}}\right). \quad (2.6)$$

The main thing to know about $\tilde{F}_{i+\frac{1}{2}}$ in the context of this thesis is that these high order correction terms also depend on the neighboring cells plus one more cell on either side (this extra cell is often used by the limiting scheme). This defines the minimal stencil a limited DG scheme could have. Additionally if the algorithms constructed in this thesis have this same stencil then many tools designed for these finite volume methods may also be applicable to our discontinuous Galerkin methods. This will be important in Chapter 8.2.

2.2 DG-FEM framework

We first provide a brief description of the discontinuous Galerkin (DG) method, which serves to define the notation used throughout the remainder of this thesis. The interested reader is referred to [35, 57, 112] and references therein for further details of the DG method.

Consider a conservation law of the form

$$q_{,t} + \nabla \cdot \mathbf{F}(q) = 0, \quad \text{in } \Omega \subset \mathbb{R}^d, \quad (2.7)$$

with appropriate boundary conditions, where $q(t, \mathbf{x}) : \mathbb{R}_{\geq 0} \times \mathbb{R}^d \mapsto \mathbb{R}^{M_{\mathrm{E}}}$ are the conserved variables, $\mathbf{F}(q) : \mathbb{R}^{M_{\mathrm{E}}} \mapsto \mathbb{R}^{M_{\mathrm{E}} \times d}$ is the flux function, d is the spatial dimension, and M_{E} is the number of equations in the system. Assume that the system is *hyperbolic*, meaning that the flux Jacobian matrix, $\mathbf{n} \cdot \mathbf{F}_{,q}$, is diagonalizable with real eigenvalues for all q in the domain of interest and for all spatial vectors $\mathbf{n}$ such that $\|\mathbf{n}\| = 1$.

Let $\Omega \subset \mathbb{R}^d$ be a polygonal domain with boundary $\partial\Omega$. The domain Ω is discretized via a finite set of non-overlapping elements, $\mathcal{T}_i$, such that $\Omega = \cup_{i=1}^N \mathcal{T}_i$. Let $P^{M_{\mathrm{D}}}\left(\mathbb{R}^d\right)$ denote the set of polynomials from $\mathbb{R}^d$ to $\mathbb{R}$ with maximal polynomial degree M_{D}. Typically one will define the degree of a polynomial so that either $P^{M_{\mathrm{D}}} = \mathbb{Q}^d_{M_{\mathrm{D}}}$ (the tensor function polynomial space) or $P^{M_{\mathrm{D}}} = \mathbb{P}^d_{M_{\mathrm{D}}}$ (the total order polynomial space). See Section 3.3 for more details on these two spaces. Let $\mathcal{W}^h$ denote the *broken* finite element space on the mesh:

$$\mathcal{W}^h := \left\{ w^h \in [L^\infty(\Omega)]^{M_{\mathrm{E}}} : \, w^h\big|_{\mathcal{T}_i} \in \left[P^{M_{\mathrm{D}}}\right]^{M_{\mathrm{E}}}, \forall \mathcal{T}_i \in \mathcal{T}^h \right\}, \quad (2.8)$$

where h is the grid spacing. The above expression means that $w^h \in \mathcal{W}^h$ has M_{E} components, each of which when restricted to some element $\mathcal{T}_i$ is a polynomial of degree at most M_{D} and no continuity is assumed across element edges (or faces in 3D).

The approximate solution on each element $\mathcal{T}_i$ at time $t = t^n$ is of the form

$$q^h(t^n, \mathbf{x}(\boldsymbol{\xi}))\Big|_{\mathcal{T}_i} = \sum_{\ell=1}^{M_\mathrm{L}} Q_i^{(\ell)n}\, \varphi^{(\ell)}\left(\boldsymbol{\xi}\right), \tag{2.9}$$

where M_L is the number of basis functions, which depends on d and M_D, and $\varphi^{(\ell)}\left(\boldsymbol{\xi}\right) : \mathbb{R}^d \mapsto \mathbb{R}$ are the basis functions defined on the reference element $\mathcal{T}_0$ in terms of the reference coordinates $\boldsymbol{\xi} \in \mathcal{T}_0$. The value $Q_i^{(\ell)n}$ is the coefficient of the basis function $\varphi^{(\ell)}$ at time-step n on cell i. Typically on structured grids we will use a modal basis defined by the Legendre polynomials, which are orthonormal with respect to the following inner product:

$$\frac{1}{|\mathcal{T}_0|}\int_{\mathcal{T}_0} \varphi^{(k)}(\boldsymbol{\xi})\, \varphi^{(\ell)}(\boldsymbol{\xi})\, d\boldsymbol{\xi} = \begin{cases} 1 & \text{if} \quad k = \ell, \\ 0 & \text{if} \quad k \neq \ell. \end{cases} \tag{2.10}$$

Similarly on unstructured grids made up of simplexes we will use an orthonormal basis computed using the Gram-Schmidt procedure. However on mapped grids we will use a nodal basis made up of Lagrange interpolating polynomials and we will use quadrature points that are co-located at the interpolation points of the Lagrange basis functions (hence this may be referred to as a collocation method). For the rest of this section we will assume we are working with a modal orthonormal basis.

Note that independent of h, d, M_D, and the type of element, the lowest order Legendre polynomial is always $\varphi^{(1)} \equiv 1$. This makes the first Legendre coefficient the cell average:

$$Q_i^{(1)n} = \frac{1}{|\mathcal{T}_0|}\int_{\mathcal{T}_0} q^h(t^n, \mathbf{x}(\boldsymbol{\xi}))\Big|_{\mathcal{T}_i} \varphi^{(1)}\left(\boldsymbol{\xi}\right)\, d\boldsymbol{\xi} =: \bar{q}_i^n. \tag{2.11}$$

2.3 Time-stepping

2.3.1 Runge-Kutta time-stepping

To obtain the semi-discrete DG method, equation (2.7) is multiplied by the test function $\varphi^{(k)}$, the resulting equation is integrated over an element $\mathcal{T}_i$, and integration-by-parts is performed on the flux term. The result when the test function is taken to be an orthonormal basis function is the following system of ODEs:

$$\frac{d}{dt}Q_i^{(k)} = \frac{1}{|\mathcal{T}_i|}\int_{\mathcal{T}_i} \nabla_{\mathbf{x}}\varphi^{(k)} \cdot \mathbf{F}\left(q^h\Big|_{\mathcal{T}_i}\right) d\mathbf{x} - \frac{1}{|\mathcal{T}_i|}\oint_{\partial\mathcal{T}_i} \varphi^{(k)}\mathcal{F} \cdot d\mathbf{s}, \tag{2.12}$$

where the numerical fluxes $\mathcal{F}$ must be determined via (approximate) Riemann solvers. The work in this thesis will often make use of the Rusanov [106] numerical flux, which is often referred to as the local Lax-Friedrichs (LLF) flux in the literature. An exact flux may also be defined for any problem by solving the Riemann problem at a point of discontinuity.

The most common approach for time-advancing DG spatial discretizations is via explicit Runge-Kutta time-stepping; the resulting combination of time and space discretization is often referred to as the "RK-DG" or "RKDG" method [31]. The primary advantage for this choice of time stepping is that explicit RK methods are easy to implement, they can be constructed to be low-storage, and a subclass of these methods are SSP [50], which is important for defining a scheme that is provably positivity-preserving. However, there are no explicit Runge-Kutta methods that are SSP for orders greater than four [66, 107].

The main difficulty with Runge-Kutta methods is that they typically require many stages; and therefore, many communications are needed per time step. One direct consequence of the communication required at each RK stage is that it is difficult to combine RK-DG with locally adaptive mesh refinement strategies that simultaneously refine in both space and time. In this thesis when using RK methods we will make use of strong stability-preserving (SSP) Runge-Kutta methods, such as those found in Gottlieb, Shu, and Tadmor [50] and Ketcheson [62]. We will primarily use RK-DG in Chapters 3 and 4.

2.3.2 Lax-Wendroff time-stepping

Local time stepping is a key piece of technology required for a locally adaptive DG schemes (see e.g., Dumbser et al. [39]). Local time-stepping is easier to accomplish with a single-stage, single-step (Lax-Wendroff) method than with a multi-stage Runge-Kutta scheme. For these reasons there is interest from discontinuous Galerkin theorists and practitioners in developing single-step time-stepping techniques for DG (see e.g., [124, 97, 40, 41, 123, 38, 47, 42]), as well as hybrid multistage multi-derivative alternatives [28]. The Lax-Wendroff schemes all proceed by first producing a time Taylor series approximation of the solution at a later time t as in Equation (2.13).

$$q(\mathbf{x}, t) = q^n + (t - t^n)q_t^n + \frac{(t - t^n)^2}{2!}q_{tt}^n + \frac{(t - t^n)^3}{3!}q_{ttt}^n + \cdots . \tag{2.13}$$

After this one can make use of the fact that $q_t = -\nabla \cdot \mathbf{F}(q)$ to compute the time-derivatives in this series expansion and project this Taylor series onto the set of basis functions.

A main point of this thesis is to construct a numerical scheme that uses a Lax-Wendroff time discretization that is coupled with the discontinuous Galerkin spatial discretization. Additionally, in subsequent discussions in this thesis we will demonstrate the advantages of switching to single-stage and single-step time-stepping in regards to enforcing positivity on arbitrary meshes and implementing local time stepping. We will develop and use Lax-Wendroff time stepping methods in Chapters 5, 6, 7 and 8.

Chapter 3

THE DISCONTINUOUS GALERKIN METHOD ON MAPPED GRIDS

3.1 Introduction

DG is extremely flexible with regards to basis functions and cell shape. Because of this flexibility the DGFEM is very commonly implemented on simplical meshes with the goal of fitting to complicated geometry [57]. However, one could instead attempt to implement the DGFEM on a logically rectangular grid using a mapping to fit complicated geometry in physical space. Finite Volume methods have been very successful in using this type of grid [77, 22]. One major advantage of this approach is that it still allows the use of very efficient block adaptive mesh refinement schemes in the style employed in the CLAWPACK [11] or ForestClaw [20] software packages.

When dealing with higher order methods on mapped cells, cell shape affects the function space spanned by the reference element basis functions. Additionally finite elements on arbitrary cell shapes in general require the construction and inversion of non diagonal mass matrices. This may be acceptable with a fixed grid but when performing AMR this would require the repeated computation and factorization of new mass matrices.

Because of the listed constraints this work will focus on locally bilinear (or trilinear in 3D) mappings. In this regime there is a choice of basis functions that will allow one to achieve high order accuracy very efficiently. In the following chapter we will introduce this efficient mapped grid DG method and demonstrate that it is actually achieving high order accuracy even on meshes that have been problematic for other high order schemes [51].

3.2 Implementation on mapped grids

In this section we will introduce the implementation of the discontinuous Galerkin scheme using a mapping function. In this thesis the methods will be developed for one or two dimensions. In this chapter we will assume that the problems have only two spatial dimensions, though the methods we shall discuss could be extended to three dimensions. We will focus on the Runge-Kutta scheme given the fact that its implementation with a mapping function will be somewhat simpler. Let us assume that we have a computational grid on the domain $(\xi, \eta) \in [-1, 1] \times [-1, 1]$ that is divided up into square cells we will label $\mathcal{T}_i$ for $i = 1 \dots N$ where N is the total number of cells. This computational domain will be associated with a

physical domain $(x, y) \in \Omega$ defined by the locally defined mapping functions

$$x = g_i(\xi, \eta) \text{ and } y = h_i(\xi, \eta)$$

We will then define the Jacobian matrix mapping between these coordinate systems on each cell $\mathcal{T}_i$

$$J_i = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} \end{bmatrix}. \tag{3.1}$$

with $|J_i|$ the determinant of the Jacobian. In this work we will only consider local bilinear mappings defined by mapping the four corners of a cell $\mathcal{T}_i$ to the four corners of a physical cell. These mappings are relatively simple and will take forms like:

$$x = \frac{(1-\xi)(1-\eta)x_{11} + (1+\xi)(1-\eta)x_{12} + (1+\xi)(1+\eta)x_{22} + (1-\xi)(1+\eta)x_{21}}{4},$$

$$y = \frac{(1-\xi)(1-\eta)y_{11} + (1+\xi)(1-\eta)y_{12} + (1+\xi)(1+\eta)y_{22} + (1-\xi)(1+\eta)y_{21}}{4}.$$

where (x_{ij}, y_{ij}) $i, j \in (1, 2)$ are supposed to represent corners of a cell in physical space. This means that the mappings and the Jacobian will always be bilinear. In this situation Equation (2.12) becomes

$$\int_{\mathcal{T}_m} \frac{\partial q^h}{\partial t} \varphi^{(k)} \, |J^m| d\xi = \frac{1}{|\mathcal{T}_m|} \int_{\mathcal{T}_m} \nabla_\xi \varphi^{(k)} \cdot \mathbf{F}\left(q^h\Big|_{\mathcal{T}_m}\right) \, |J^m| d\xi - \frac{1}{|\mathcal{T}_m|} \oint_{\partial \mathcal{T}_m} \varphi^{(k)} \mathcal{F} \cdot d\mathbf{s}. \tag{3.2}$$

Most of the terms in Equation (3.2) are straightforward to evaluate, however there are two problems. First we must decide how to evaluate the term

$$\nabla_\xi \varphi^{(k)} \cdot \mathbf{F}\left(q^h\Big|_{\mathcal{T}_m}\right).$$

In evaluating this term at a point one must project the matrix valued function $\mathbf{F}$ onto the ξ, η directions at each point, respectively. The easiest way to do that is to define

$$\tilde{f} = \frac{\partial \xi}{\partial x} f + \frac{\partial \xi}{\partial y} g \quad \text{and} \quad \tilde{g} = \frac{\partial \eta}{\partial x} f + \frac{\partial \eta}{\partial y} g,$$

so that

$$\nabla_\xi \varphi^{(k)} \cdot \mathbf{F}\left(q^h\Big|_{\mathcal{T}_i}\right) = \frac{\partial \varphi^{(k)}}{\partial \xi} \tilde{f} + \frac{\partial \varphi^{(k)}}{\partial \eta} \tilde{g} = \frac{\partial \varphi^{(k)}}{\partial x} f + \frac{\partial \varphi^{(k)}}{\partial y} g.$$

The second issue has to be dealt with is the term

$$\int_{\mathcal{T}_m} \frac{\partial q^h}{\partial t} \varphi^{(k)} \, |J^m| d\xi.$$

The typical hierarchical Legendre basis will no longer give an orthonormal mass matrix as we can no longer say that $\int_{\mathcal{T}_m} \varphi^{(k)}\varphi^{(j)}\,|J^m|d\xi = \delta_{jk}$ because of the Jacobian term. However, remember that the Jacobian is going to be bilinear. Also remember that the set of $n+1$ one-dimensional Gaussian quadrature points exactly integrates all polynomials of order $2n+1$. We denote such a set as Z_{n+1} and the corresponding set of weights as W_{n+1}

Let us define $deg(\varphi, \xi)$ to be the maximum power of ξ contained in φ. Then we say that we are using an $(n+1)$st order DG scheme if we have $\max\left(\deg(\varphi,\xi), \deg(\varphi,\eta)\right) \leq n$. Additionally let us define ℓ_i for $i = 1, \ldots, n+1$ to be Lagrange polynomials associated with the points $\zeta_i \in Z_{n+1}$. The Lagrange polynomials associated with a set of points are the polynomials defined so that each polynomial takes the value 1 on exactly one point in the set, and zero on all of the other points in the set.

If we define

$$\varphi^{(k)} = \ell_{i^k}(\xi)\ell_{j^k}(\eta) \text{ for } k = 1, \ldots, (n+1)^2 \text{ and } (i,j) \in (1,2,\ldots,n+1) \times (1,2,\ldots,n+1),$$

then we have that

$$\int_{\mathcal{T}_m} \varphi^{(k)}\varphi^{(j)}\,|J^m|d\xi = \omega_{i^k}\omega_{j^k}|J^m|\Big|_{(z_{i^k}, z_{j^k})}\delta_{jk}.$$

This is only true because of the restrictions we have placed on the mapping functions $g(\xi,\eta)$ and $h(\xi,\eta)$. So if we restrict ourselves to this combination of basis and mapping functions we can write Equation (3.2) as

$$\frac{d}{dt}Q_m^{(k)}\,\omega_{i^k}\omega_{j^k}|J^m|\Big|_{(z_{i^k}, z_{j^k})} = \frac{1}{|\mathcal{T}_m|}\int_{\mathcal{T}_m} \nabla_{\mathbf{x}}\varphi^{(k)} \cdot \mathbf{F}\left(q^h\Big|_{\mathcal{T}_m}\right)\,|J^m|d\mathbf{x} - \frac{1}{|\mathcal{T}_m|}\oint_{\partial\mathcal{T}_m} \varphi^{(k)}\mathcal{F}\cdot d\mathbf{s}. \quad (3.3)$$

where $Q_m^{(k)}$ now represents the value of q^h at a point.

3.2.1 Proving conservation

In order for this method to be useful for solving general hyperbolic PDEs we must prove that it is conservative. This is obvious when using a hierarchical basis on an unmapped grid because we can explicitly see the update equation for the cell average. It is less obvious for this method. We can use Equation (3.3) to compute the change in mass in the cell $\mathcal{T}_m$. This

gives

$$\frac{d}{dt}\int_{\mathcal{T}_m} q^h\,|J^m|d\mathbf{x} = \frac{d}{dt}\int_{\mathcal{T}_m} q^h\left(\sum_{k=1}^{(n+1)^2}\varphi^{(k)}\right)|J^m|d\mathbf{x} = \sum_{k=1}^{(n+1)^2}\frac{d}{dt}Q_m^{(k)}\,\omega_{i^k}\omega_{j^k}|J^m|\Big|_{(z_{i^k},z_{j^k})},$$
$$= \frac{1}{|\mathcal{T}_m|}\int_{\mathcal{T}_m}\left(\sum_{k=1}^{(n+1)^2}\nabla_{\mathbf{x}}\varphi^{(k)}\cdot\mathbf{F}\left(q^h\Big|_{\mathcal{T}_m}\right)\right)|J^m|d\mathbf{x} - \frac{1}{|\mathcal{T}_m|}\oint_{\partial\mathcal{T}_m}\left(\sum_{k=1}^{(n+1)^2}\varphi^{(k)}\right)\mathcal{F}\cdot d\mathbf{s},$$
$$= \frac{1}{|\mathcal{T}_m|}\int_{\mathcal{T}_m}\nabla_{\mathbf{x}}\left(\sum_{k=1}^{(n+1)^2}\varphi^{(k)}\right)\cdot\mathbf{F}\left(q^h\Big|_{\mathcal{T}_m}\right)|J^m|d\mathbf{x} - \frac{1}{|\mathcal{T}_m|}\oint_{\partial\mathcal{T}_m}\left(\sum_{k=1}^{(n+1)^2}\varphi^{(k)}\right)\mathcal{F}\cdot d\mathbf{s}.$$

Now we can use the fact that $\left(\sum_{k=1}^{(n+1)^2}\varphi^{(k)}\right) = 1$ to simplify our expression to

$$\frac{d}{dt}\int_{\mathcal{T}_m} q^h\,|J^m|d\mathbf{x} = -\frac{1}{|\mathcal{T}_m|}\oint_{\partial\mathcal{T}_m}\mathcal{F}\cdot d\mathbf{s}. \tag{3.4}$$

Note that the property $\left(\sum_{k=1}^{(n+1)^2}\varphi^{(k)}\right) = 1$ is due to the fact that the functions $\varphi^{(k)}$ are Lagrange interpolating polynomials defined on the reference element so this sum corresponds to interpolating a constant function equal to 1 at each interpolation point. □

3.2.2 Calculating the CFL number

It is somewhat unclear how to calculate the CFL number on a mapped grid. In this chapter we define the CFL number using an approach typically employed for DG on unstructured grids [57]. We look at each cell in the physical domain and divide it up into the set of two triangles (assuming we are in 2 dimensions) that maximizes the minimum area among the two triangles. We call this maximized minimum area A. We also call the maximum wave speed on any of the quadrature points on the edge of the cell s (scaled by the corresponding edge length). Then we calculate the CFL as

$$\text{CFL} = dt\frac{s}{2A}.$$

Note that this is not the unique way to define the CFL number on these cells.

3.3 Convergence theory on mapped grids

The discussion below will reference two standard function spaces used in multivariate approximation. These two spaces are $\mathbb{P}_n^d$, the total order space, and $\mathbb{Q}_n^d$ the tensor product polynomial space:

$$\mathbb{P}_n^d = \{x_1^{n_1}x_2^{n_2}\ldots x_d^{n_d} \mid n_1 + n_2 + \ldots n_d \leq n\}, \tag{3.5}$$

$$\mathbb{Q}_n^d = \{x_1^{n_1} x_2^{n_2} \dots x_d^{n_d} \mid \max(n_1, n_2, \dots, n_d) \leq n\}. \tag{3.6}$$

Think of the $\mathbb{Q}_n^d$ space as a d dimensional cube with edges that are length n and the $\mathbb{P}_n^d$ space as one corner of this cube as in Figure 3.1. It has been shown that approximations using a bilinear mapping function where the approximation space is $\mathbb{P}_n^d$ will not be guaranteed to achieve the optimal $n+1$ order convergence we might expect on a structured grid. This is true no matter how smooth the function being approximated is [5, 4]. The actual observed convergence rate will depend on the dimension d and on the properties of the mapping. However, mapped approximation using a function space that resolves $\mathbb{Q}_n^d$ can be done in a way that achieves $n+1$ order convergence on appropriately smooth functions for these bilinear mappings [5, 4].

The fundamental reason this occurs is that the function space in the computational domain does not resolve all of $\mathbb{P}_n^d$ when mapped to physical space except in very special circumstances. For example consider the mapping

$$x = \xi - \frac{1}{2}\xi\eta \text{ and } y = \eta - \frac{1}{2}\xi\eta.$$

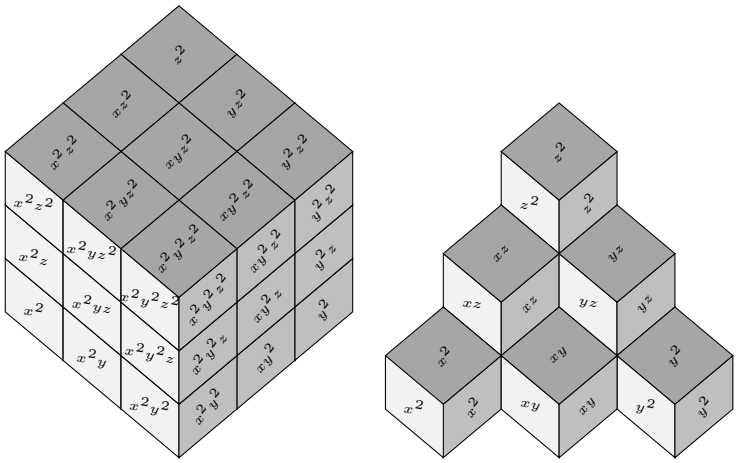
This mapping is bilinear and in fact it maps the square $[0,1] \times [0,1]$ to a triangle defined by the three corners $\{(0,0),(1,0),(0,1)\}$.

If the (x,y) space is the physical domain, in order to resolve x^2 in the physical domain a function space on the computational domain (ξ,η) would need to resolve $x^2 = \xi^2 - \xi^2\eta + \frac{1}{4}\xi^2\eta^2$. This means that $x^2 \in \mathbb{Q}_2^2\Big|_{(\xi,\eta)}$ but $x^2 \notin \mathbb{P}_2^2\Big|_{(\xi,\eta)}$. So, for example, a method implemented on the cell $(\xi,\eta) \in [0,1] \times [0,1]$ with this associated mapping would have no hope of exactly reproducing the values of the function x^2 using the approximation space $\mathbb{P}_2^2\Big|_{(\xi,\eta)}$. More importantly, no approximation of a function on this domain using $\mathbb{P}_2^2\Big|_{(\xi,\eta)}$ could hope to correctly match the Taylor series of a function up to the squared terms.

3.4 Square to circle mapped grids

In this section we will introduce a family of square to circle mappings that have been used successfully with the finite volume method [22, 10]. These mappings provide a good test case for developing the DG method on mapped grids because they are simple to implement but still computationally challenging. For example consider the grids shown in Figure 3.3. Each cell in these grids is mapped to a rectangle in the computational domain. Some of the cells that are very nearly triangular tend to present difficulties for numerical methods [51].

The mappings we will use essentially work by trying to map lines of constant x or y in the computational domain to lines of constant radius as illustrated by the lines of shaded cells in Figure 3.2. Essentially this amounts to mapping concentric squares to concentric



(a) $\mathbb{Q}_2^3$ (b) $\mathbb{P}_2^3$

Figure 3.1: A visual representation of the second degree incarnation of the two most important function spaces in three dimensional approximation theory

circles. However this procedure can be altered in several ways to minimize the number of nearly triangular cells [22].

In Listing 3.1 the mapping literally just maps concentric squares to concentric circles to produce the circular domain grid in Figure 3.3. Essentially the computational domain is divided into sections by its diagonals. In each section a line of constant x or y is mapped to a circular arc centered at the point $(d_1 - \sqrt{R^2 - d_1^2}, 0)$ or $(0, d_1 - \sqrt{R^2 - d_1^2})$ with radius R.

In Listing 3.2 a similar approach is taken to produce a circular inclusion embedded within a rectangular grid as shown in Figure 3.3. In order to transition between cells along circular arcs and cells along lines of constant x or y, the radius used in the mapping is chosen to tend to ∞ as it approaches a certain set of constant x or constant y lines in the computational domain.

Listing 3.1: A C++ code snippet to map a square domain to a circular domain as shown in Figure 3.3

```
double    r1 = 1.5;

double xc1 = fabs(xc);
double yc1 = fabs(yc);
double d = max(max(xc1,yc1), 1.0e-14);
double d1 = d*r1/sqrt(2.0);
double R = r1*d;              // for constant curvature grid lines

double xp2 = d1/d * xc1;
double yp2 = d1/d * yc1;
double center = d1 - sqrt(R*R - d1*d1);

if (fabs(xc1)>fabs(yc1)+1.0e-15){xp2 =center+sqrt(R*R-yp2*yp2);}
if (fabs(yc1)>fabs(xc1)+1.0e-15){yp2 =center+sqrt(R*R-xp2*xp2);}
if ((xc)<1.0e-15){xp2=-fabs(xp2);}
if ((yc)<1.0e-15){yp2=-fabs(yp2);}
xp = xp2;
yp = yp2;
```

Listing 3.2: A C++ code snippet to map a grid on square domain to a grid with a circular inclusion as shown in Figure 3.3

```
double    x0 = x0circ;
double    y0 = y0circ;
double    r1 = r1circ;
double    r2 = r2circ;
double    xc0 = fabs(xc-x0);
double    yc0 = fabs(yc-y0);

   if ((max(xc0,yc0)-r2)<=1.0e-15)
      {

      double xc1 = xc0/r2;
      double yc1 = yc0/r2;
      double d = max(max(xc1,yc1), 1.0e-14);
      d = min(d, 0.9999);
      double d1 = d*r2/sqrt(2.0);
      double d2;
      double rad = r1;              // for constant curvature grid lines
      if (d > r1/r2)
      { d1 = r1/sqrt(2.0)+(d-r1/r2)*(r2-r1/sqrt(2.0))/(1.0 - r1/r2);
        d2 = max(1.0 - d, 1.0e-8);
        rad = r1*pow(((1.0-r1/r2)/d2),(r2/r1+0.5));
      }
      double xp2 = d1/d * xc1;
      double yp2 = d1/d * yc1;
      double center = d1 - sqrt(rad*rad - d1*d1);
      if (abs(xc1-d)<1.0e-15) {xp2 = center + sqrt(rad*rad - yp2*yp2);}
      if (abs(yc1-d)<1.0e-15) {yp2 = center + sqrt(rad*rad - xp2*xp2);}
      if ((xc-x0)<1.0e-15){xp2=-fabs(xp2);}
      if ((yc-y0)<1.0e-15){yp2=-fabs(yp2);}
      xp = x0 + xp2;
      yp = y0 + yp2;
      }
```

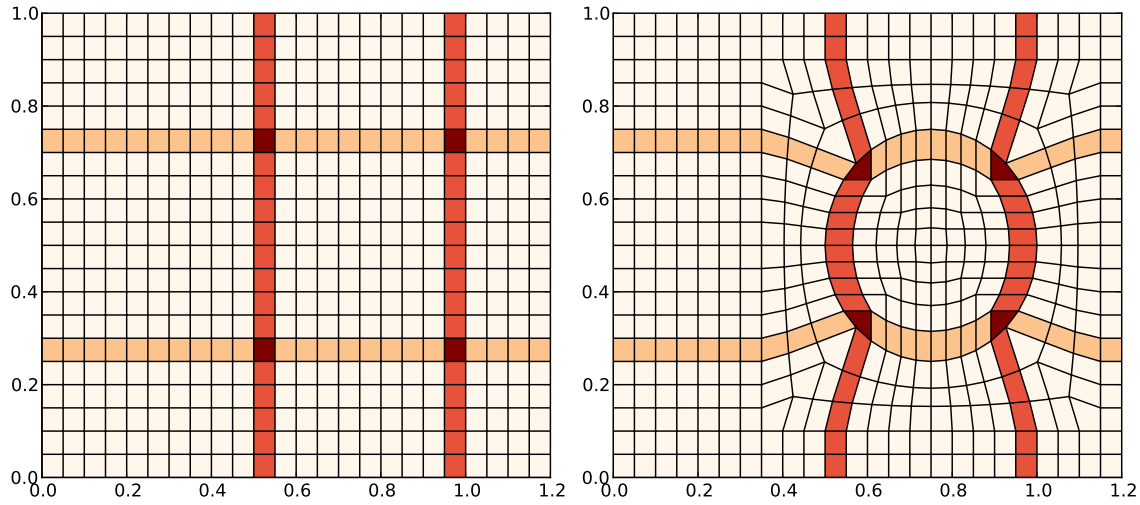
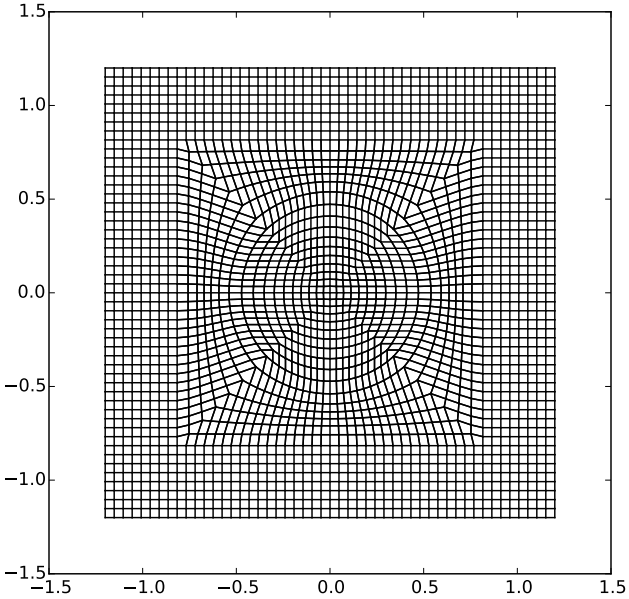
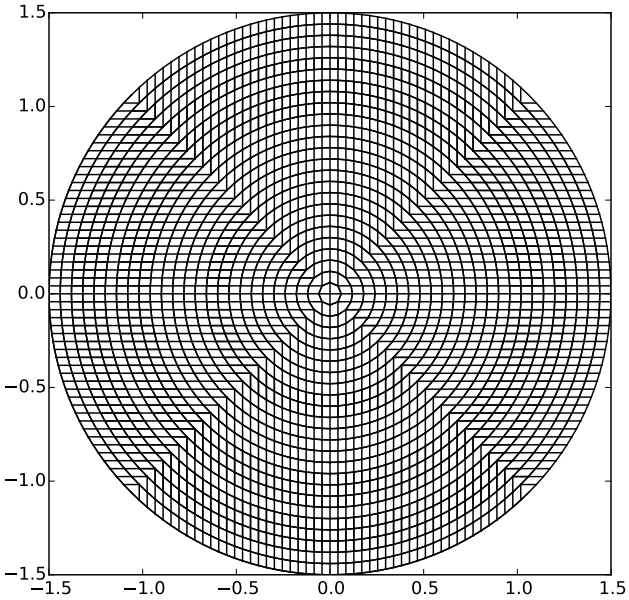


Figure 3.2: A computational domain with a uniform grid mapped to a quadrilateral grid in the physical domain. The shaded cells in the computational domain are mapped to the shaded cells in the physical domain.



(a) Inclusion



(b) Circular domain

Figure 3.3: Two sample logically Cartesian grids created using mapping functions. See Listings 3.1 and 3.2 to see the mapping functions that created these meshes from a uniform grid on the domain $[-1, 1] \times [-1, 1]$.

3.4.1 Convergence of the advection equation

Given the discussion on convergence theory in Section 3.3 we will use the tensor product function space $\mathbb{Q}_n^2$ in the following section. The goal of this section is to illustrate that with this choice of approximation space, a discontinuous Galerkin method can exhibit very high order convergence rates even on very difficult mappings. In order to demonstrate this we will solve the advection equation with a velocity field $u = (1, 0)$ on the domain shown in Figure 3.4. The initial condition used will be

$$q(x, y, t) = \begin{cases} \left[\frac{1}{2}\left(1 + \cos\left(10\pi\left(x - 0.15\right)\right)\right)\right]^6 & \text{for } x \in [0.5, 0.25] \\ 0 & \text{for } (x, y)[0, 1.2] \times [0, 1] \end{cases}.$$

There will be a circular inclusion in this domain centered at $x = 0.5$ and $y = 0.75$ with radius $r = 0.3$. The convergence results for this problem using different approximation spaces $\mathbb{Q}_n^d$ can be found in Table 3.1. The convergence rates observed running this problem for each polynomial order are still above the theoretically possible asymptotic limit during the observed range of resolutions so the observed rates would come down if the tests were run for higher resolutions. These are impressive results given the difficulties that one could expect to see with these quadrilateral meshes.

mx	$n = 1$	**Order**	**mx**	$n = 2$	**Order**	**mx**	$n = 3$	**Order**
40	5.08×10^{-01}	—	80	3.4×10^{-02}	—	40	4.76×10^{-02}	—
72	3.49×10^{-01}	0.64	112	8.82×10^{-03}	4.0	56	1.05×10^{-02}	4.49
129	1.62×10^{-01}	1.3	156	2.0×10^{-03}	4.42	78	1.6×10^{-03}	5.59
233	4.69×10^{-02}	2.1	219	4.23×10^{-04}	4.63	109	2.04×10^{-04}	6.12
419	9.61×10^{-03}	2.7	307	9.66×10^{-05}	4.389	153	2.81×10^{-05}	5.91
755	1.75×10^{-03}	2.9	430	2.56×10^{-05}	3.95	215	5.4×10^{-06}	4.92
—	—	—	602	7.92×10^{-06}	3.48	301	1.3×10^{-06}	4.2

Table 3.1: Convergence for the 2D advection problem in Section 3.4.1. All errors are L_2 norm errors.

3.4.2 Radially symmetric acoustic wave packet

Circular inclusion grid

In this section we run an acoustics equation example with a circular interface. At this interface the sound speed and the impedance will jump discontinuously. Theoretically since we are using polygonal elements the convergence rates we observe for any order of approximation space should be limited to second order. However it is still possible that using higher order basis functions will give other benefits. The problem is as follows:

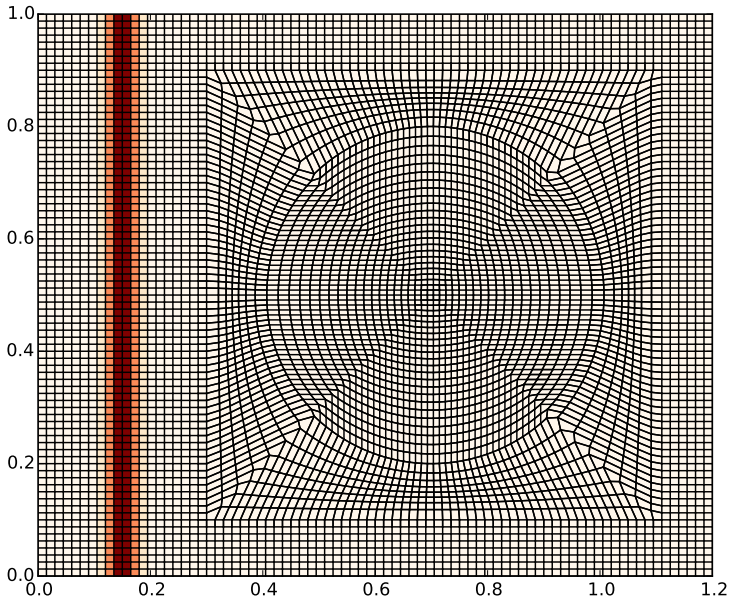


Figure 3.4: An example grid used to demonstrate the convergence rate of the discontinuous Galerkin method on mapped grids in Section 3.4.1.

$$\begin{bmatrix} p \\ u \\ v \end{bmatrix}_t + \begin{bmatrix} 0 & K_0 & 0 \\ \frac{1}{\rho_0} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p \\ u \\ v \end{bmatrix}_x + \begin{bmatrix} 0 & 0 & K_0 \\ 0 & 0 & 0 \\ \frac{1}{\rho_0} & 0 & 0 \end{bmatrix} \begin{bmatrix} p \\ u \\ v \end{bmatrix}_y = 0 \tag{3.7}$$

with

$$K_0 = \begin{cases} 1 \text{ for } \sqrt{x^2 + y^2} < 0.5 \\ 9 \text{ otherwise} \end{cases} \tag{3.8}$$

$$\rho_0 = 1.0 \tag{3.9}$$

We use the initial condition $u(x, y, 0) = v(x, y, 0) = 0.0$ and $p(x, y, 0) = e^{-600(r-0.5)^2} \sin(150r)$ where $r = \sqrt{x^2 + y^2}$ and the domain has outflow boundary conditions. For this problem we must use the exact Riemann solver for a variable coefficient acoustics equation as the Rusanov flux would assume the equation was written in conservative form. Let us define the sound speed c, the impedance $z = c\rho$ and the bulk modulus $k(x) = c^2\rho$. Consider the one-dimensional Riemann problem

$$\begin{bmatrix} p \\ u \end{bmatrix}_t + \begin{bmatrix} 0 & K(x) \\ \frac{1}{\rho(x)} & 0 \end{bmatrix} \begin{bmatrix} p \\ u \end{bmatrix}_x = 0,$$

with

$$\begin{bmatrix} p \\ u \end{bmatrix} = \begin{cases} q_l \text{ for } x < x_i \\ q_r \text{ for } x \geq x_i \end{cases} \quad K(x) = \begin{cases} z_l c_l \text{ for } x < x_i \\ z_r c_r \text{ for } x \geq x_i \end{cases} \text{, and } \rho(x) = \begin{cases} \rho_l \text{ for } x < x_i \\ \rho_r \text{ for } x \geq x_i \end{cases}.$$

The eigenvectors for the matrix $\begin{bmatrix} 0 & K(x) \\ \frac{1}{\rho(x)} & 0 \end{bmatrix}$ are

$$r^1 = \begin{bmatrix} -z \\ 1 \end{bmatrix} \text{ and } \begin{bmatrix} z \\ 1 \end{bmatrix}.$$

Instantaneously after time begins marching forward, the solution at the interface x_i will be

$$q_{x_i} = q_l + \alpha_1 \begin{bmatrix} -z_l \\ 1 \end{bmatrix},$$

where

$$\alpha_1 = \frac{-\Delta_1 + z_r \Delta_2}{z_l + z_r}, \ \Delta_1 := q_r[1] - q_l[1] \text{ and } \Delta_2 := q_r[2] - q_l[2].$$

We will solve this Riemann problem in the normal direction at each point on the edge of our cell to compute the boundary integral term.

The problem starts out with a highly oscillatory ring of pressure centered at radius $r = 0.15$. The ring splits into an inward and and outward going ring. The outward going ring will split again when it encounters the discontinuity in the coefficients at $r = 0.5$. We will compare slice plots of these 2d simulations to a highly refined solution of the radially symmetric problem obtained by taking advantage of the radial symmetry in the problem to produce the system

$$\begin{bmatrix} p \\ u_r \end{bmatrix}_t + \begin{bmatrix} 0 & K_0 \\ \frac{1}{\rho} & 0 \end{bmatrix} \begin{bmatrix} p \\ u_r \end{bmatrix}_x = \begin{bmatrix} -\frac{K_0}{r} u_r \\ 0 \end{bmatrix}. \tag{3.10}$$

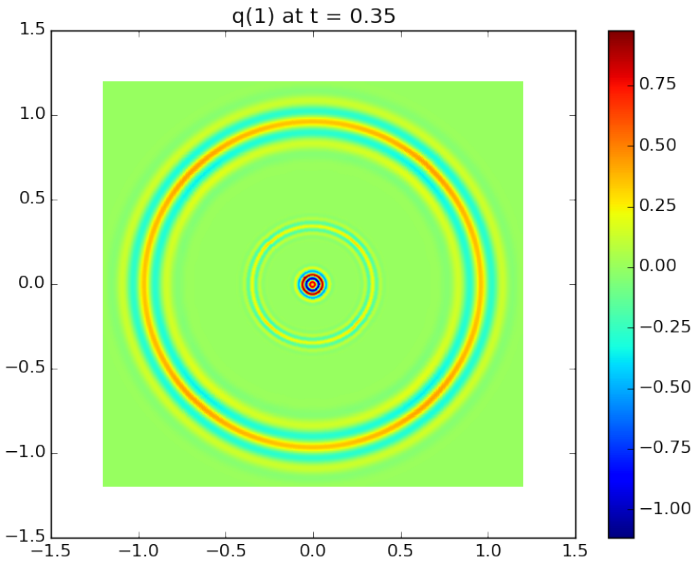


Figure 3.5: The density for the radial acoustics problem in Section 3.4.2 at $t = 0.35$s. This result was produced using the function space $\mathbb{Q}_2^2$ on a 480×480 cell computational grid.

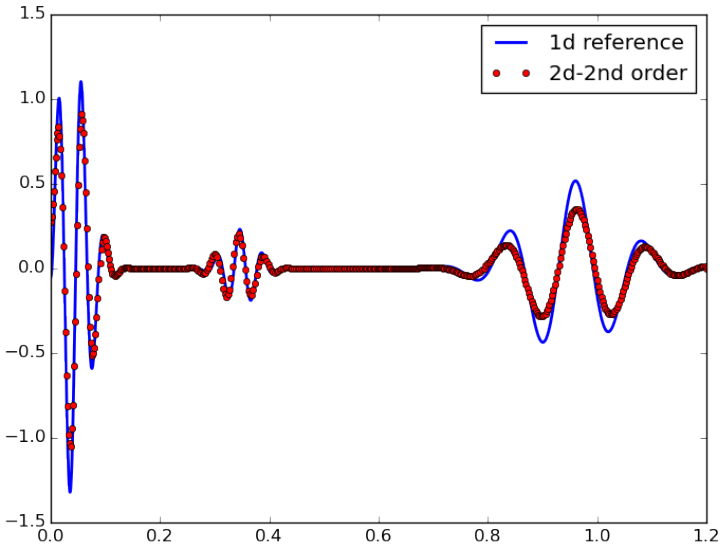


Figure 3.6: A slice of the density for the radial acoustics problem in Section 3.4.2 at $t = 0.35$s. This result was produced using the function space $\mathbb{Q}_2^2$ on a 480×480 cell computational grid.

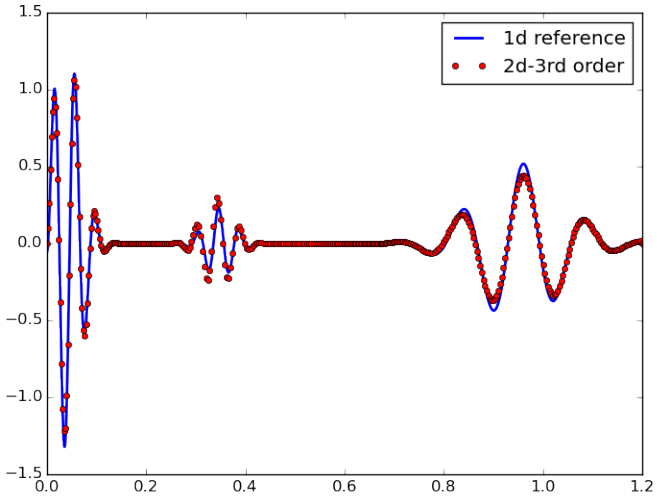


Figure 3.7: A slice of the density for the radial acoustics problem in Section 3.4.2 at $t = 0.35$s. This result was produced using the function space $\mathbb{Q}_3^2$ on a 240×240 cell computational grid.

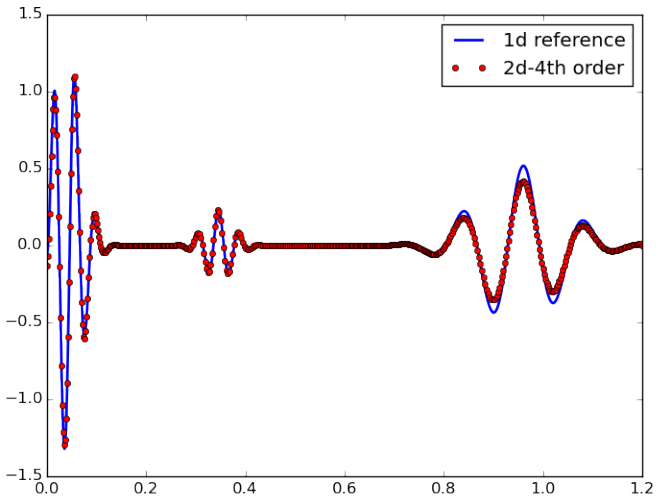


Figure 3.8: A slice of the density for the radial acoustics problem in Section 3.4.2 at $t = 0.35$s. This result was produced using the function space $\mathbb{Q}_4^2$ on a 240×240 cell computational grid.

In the slice plots we take a slice of the 2d density plot along the line $y = 0$. We compare this slice to the solution of the problem defined in Equation (3.10) computed with 2000 cells using Clawpack. We should expect this reference solution to be very accurate given that in 1D it is trivial to resolve the interface and it can easily be highly refined. We can see from Figures 3.5 - 3.8 that there is a very large performance increase when going from second to third order even on a much coarser grid. We chose to run the problem on 480 cells for the second order method because each second order cell only has 4 degrees of freedom and each third order cell has 9 degrees of freedom. There is a smaller performance jump from the third order method to the 4th order method even though the 4th order method has 16 degrees of freedom per cell. However, note that in the oscillatory region near the center of the domain the 4th order method is nearly perfect. The solution in this region has not interacted with the interface at $r = 0.5$. It seems that in this situation the second order accuracy of the mesh fitting the interface is limiting the possible improvement. This problem could theoretically be remedied with adaptive refinement or by using curvilinear meshes.

Circular domain

In this section we run the same problem as in Section 3.4.2, but on the circular domain shown in Figure 3.3. This mesh has more nearly triangular cells than the circular inclusion mesh, and it appears from Figure 3.10 that the poor quality of the mesh has an effect on the solution quality. The solution seems much less sharp.

3.4.3 Shock-entropy wave problem

In this section we run a simple 2d extension of the well known Shu-Osher problem found in [118, 80, 28]. This problem involves a shock interacting with a smooth but oscillatory entropy wave. It is often used as a test case for high resolution methods because it produces very oscillatory behavior and it also requires resolving a shock. The problem is essentially a one-dimensional problem, but we solve it on a mesh like the one shown in Figure 3.11 so the computation is not able to take advantage of the problem's one dimensional nature. We will solve the Euler equations

$$\begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \mathcal{E} \end{bmatrix}_t + \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ (\mathcal{E} + p)u \end{bmatrix}_x + \begin{bmatrix} \rho u \\ \rho uv \\ \rho v^2 + p \\ (\mathcal{E} + p)v \end{bmatrix}_y = 0, \text{ with } \mathbf{u} = \begin{bmatrix} u \\ v \end{bmatrix}, \mathbf{M} = \rho \mathbf{u}, \tag{3.11}$$

$$\text{and } p = (\gamma - 1)\left(\mathcal{E} - \frac{1}{2}\rho\|\mathbf{u}\|^2\right). \tag{3.12}$$

The ratio of specific heats (aka, the *heat capacity ratio*) will be $\gamma = \frac{7}{5}$. This problem has

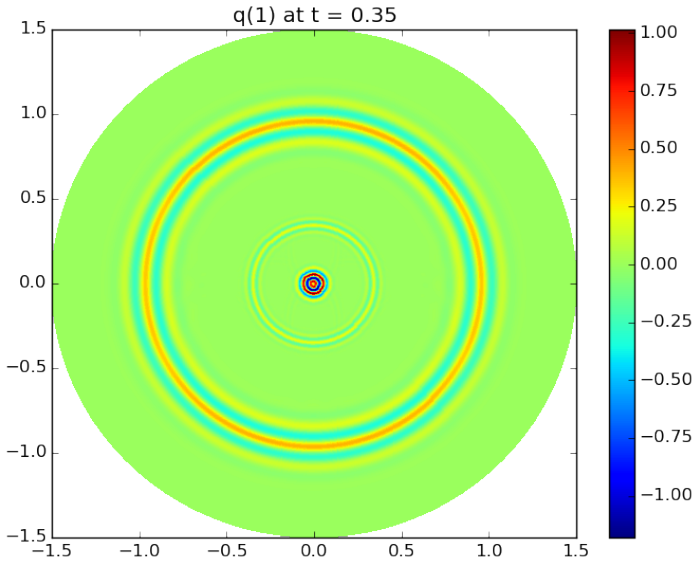


Figure 3.9: The density for the radial acoustics problem in Section 3.4.2 at $t = 0.35$s. This result was produced using the function space $\mathbb{Q}_2^2$ on a 300×300 cell computational grid.

the initial conditions

$$\begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \mathcal{E} \end{bmatrix} = \begin{cases} \begin{bmatrix} 3.857143 \\ 3.857143(2.629369) \\ 0 \\ \frac{10.333333}{\gamma-1} + (2.629369)\frac{(3.857143)^2}{2} \end{bmatrix} & \text{for } x \leq -4.0 \\ \begin{bmatrix} 1+\sin(5x) \\ 0 \\ 0 \\ \frac{1}{\gamma-1} \end{bmatrix} & \text{otherwise,} \end{cases}$$

on the domain $(x, y) \in [-5, 5] \times [-5, 5]$. All of the boundaries are outflow boundaries except the $x = -5$ boundary where the initial condition is used as the boundary condition.

This problem cannot be run without any limiting, and so the results produced here have a very small amount of limiting used to ensure that the density and pressure never go negative. This will be discussed further in Section 4.10.

In Figure 3.12 one can see that the mesh is producing some visible effects when it interacts with the shock wave. This is not surprising given that near the shock one cannot expect the

solution to be converging with high order so changes in mesh orientation could theoretically have a big impact. This was run on a 200×200 mesh, so perhaps this would be less of a problem at smaller mesh sizes. Figure 3.13 shows a 1D slice of the density taken on the line $y = -1.5$. It is compared to a reference solution computed with a 1D 4th order WENO scheme using 10000 grid cells. The two-dimensional DG solution has some undershoots and overshoots, but it seems to still be performing fairly well.

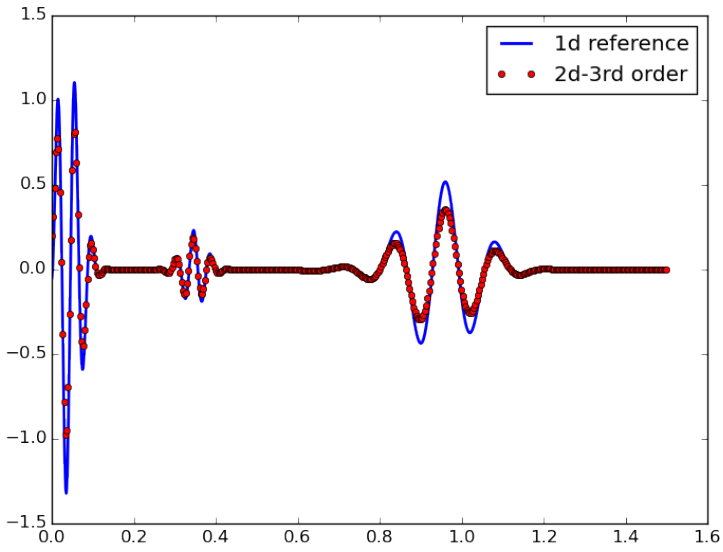


Figure 3.10: A slice of the density for the radial acoustics problem in Section 3.4.2 at $t = 0.35$s. This result was produced using the function space $\mathbb{Q}_2^2$ on a 300×300 cell computational grid.

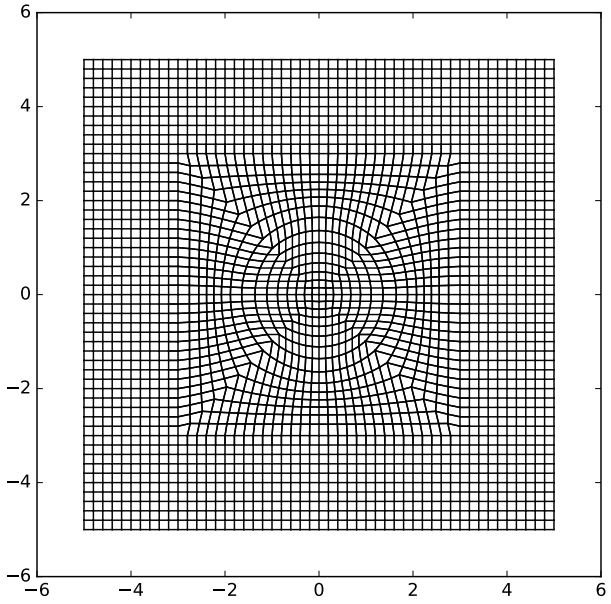


Figure 3.11: The circular inclusion grid used in the Shu-Osher problem simulation in Section 3.4.3.

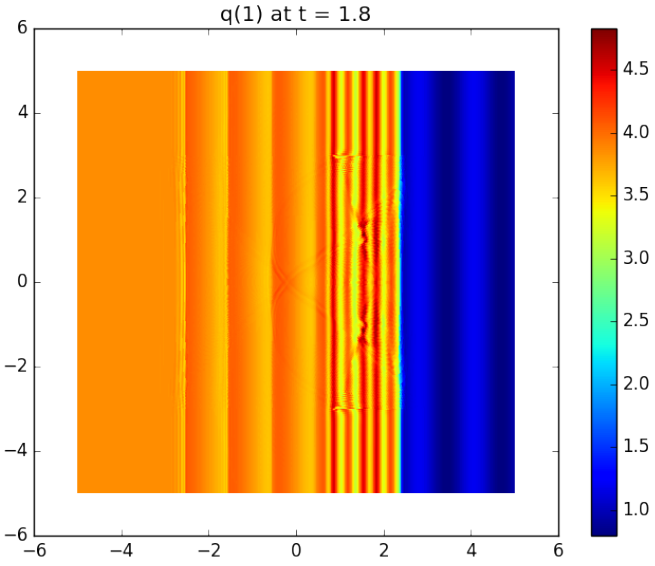


Figure 3.12: The density found for the Shu-Osher problem simulated on a mapped grid in Section 3.4.3 at $t = 1.8$s.

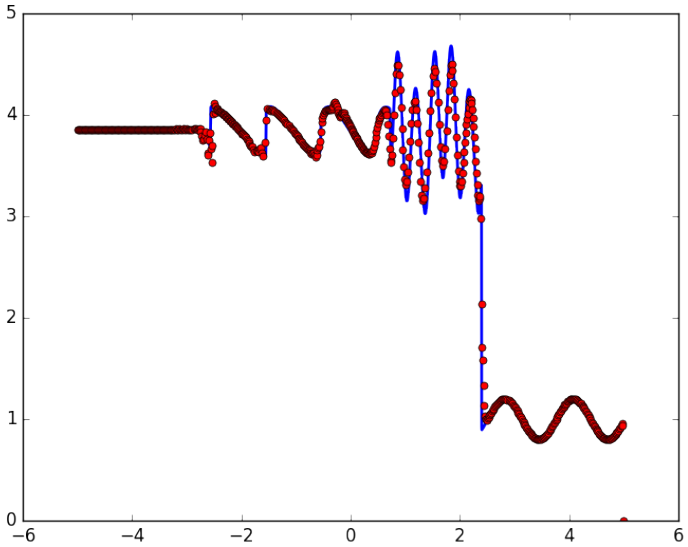


Figure 3.13: A slice plot of the density found for the Shu-Osher problem simulated on a mapped grid in Section 3.4.3 at $t = 1.8$s.

Chapter 4

LIMITING THE DISCONTINUOUS GALERKIN METHOD

4.1 Introduction

This chapter corresponds to a paper written with James Rossmanith and David Seal [86]. In this work we develop a novel limiter for the discontinuous Galerkin method that addresses several issues that the current standard DG limiters posses. We use Runge-Kutta DG with a low storage SSPRK4 [62]. We use this particular time-stepping scheme so that we can maintain positive cell averages of pressure and density in the Euler equation examples found in the chapter.

4.1.1 Some background

Godunov's theorem [49] states that a *linear* numerical method when applied to the one-dimensional, linear, advection equation:

$$\frac{\partial q}{\partial t} + \frac{\partial q}{\partial x} = 0, \quad q(t=0,x) = q_0(x), \tag{4.1}$$

where $x \in \mathbb{R}$, $t \in \mathbb{R}_{\geq 0}$, and $q(t,x) : \mathbb{R}_{\geq 0} \times \mathbb{R} \to \mathbb{R}$, is *monotonicity preserving* (i.e., does not generate new extrema) only if it is at most first-order accurate. In practice, schemes that are not monotone will exhibit large spurious oscillations near discontinuities in the solution. The analog of this result for more complicated hyperbolic equations (i.e., nonlinear systems in multiple dimensions) is that linear numerical methods with a formal accuracy higher than first-order will potentially create spurious oscillations that can lead to catastrophic numerical instabilities (e.g., negative densities and/or pressures). As in the linear scalar case, these instabilities are especially pronounced if the solution becomes discontinuous (i.e., shock waves and contact discontinuities). The additional challenge with nonlinear hyperbolic equations is that generic initial conditions tend to produce shocks in finite time.

The principal remedy for these numerical instabilities in high-order methods is to introduce a *shock-capturing limiter*; such limiters render an otherwise linear method nonlinear, thereby circumventing the restrictions of Godunov's theorem. Typically, limiters are designed to execute two tasks: (1) to monitor the solution quality (at least in some simplified sense); and, when necessary, (2) to reduce the influence of high-order corrections. The development of limiters for hyperbolic equations was pioneered by Harten and Zwas [56], Boris and Book [18, 17] and Van Leer [133, 134]. Additional fundamental contributions came from

Harten [53, 54], Sweby [120], and Tadmor [121] who developed mathematical precise limiting strategies based on the principle that the numerical scheme should be total variation non-increasing. A good review of these limiters can be found in Chapter 6 of LeVeque [79]. In more recent work, many contributions have been made to generalize these limiting strategies to high-order methods, including to weighted non-oscillatory (WENO) schemes (e.g., see Liu et al. [80] and Jiang and Shu [60]) and discontinuous Galerkin schemes (e.g., for a review see Qiu and Shu [100] and Dumbser et al. [43]).

This section focuses on limiters for discontinuous Galerkin (DG) methods [103, 34, 35]. Limiting strategies for DG methods can generally be broken down into two key steps:

Step 1. Detect *troubled-cells* (i.e., cells that contain large gradients);

Step 2. Apply a limiter to each detected troubled cell (i.e., in some way reduce the local gradients or higher derivative analogs thereof without changing the average value in that cell).

In some limiting strategies these two steps are executed with completely different techniques (e.g., see [100]), while in other approaches the detection and limiting happen all in one fell swoop (e.g., the moment limiter of Krivodonova [67]). In either case, the end goal of most limiters is to design a method that simultaneously accomplishes two often conflicting objectives: (1) reduce unphysical oscillations in the presence of shocks and (2) retain high-order accuracy in smooth regimes.

The techniques used for DG methods in the limiting step (**Step 2**) can be subdivided into three broad classes of limiters, which are briefly described below.

Moment or slope methods

These limiters have their roots in the development of classical second-order finite volume methods. The key ingredients are minimum modulus evaluations that compare the slope (or higher derivative analogs) in the current cell to values computed from neighboring cells. Such methods are still widely used (e.g., see [58, 58, 67, 139, 71, 72, 73]). The wave limiters used in Clawpack fit into this family of limiting schemes [79]. There are two well-known difficulties with these limiters: (1) they tend to degrade the of order of accuracy in smooth extrema; and (2) they are difficult to generalize to unstructured meshes.

Hybrid weighted essentially non-oscillatory methods

Methods in this category are motivated by the success of the weighted essentially non-oscillatory (WENO) method in shock-capturing. The classical schemes [99, 101, 102, 148, 147, 149] redefine the polynomial representation of the solution inside each element by considering neighboring elements and matching cell averages. The WENO reconstruction, which

is automatically mass conservative, is then typically applied in order to redefine polynomial representations inside each element. More recently, the so-called Hermite WENO schemes have been used in an effort to reduce the size of the computational stencil [82]. A chief criticism of these methods has been the expense of their application; although recent efforts exist to reduce their computational cost [149].

Artificial viscosity methods

This method is based on the classical idea of adding a small amount of artificial viscosity to hyperbolic equations [135]. The modern incarnation of this limiting strategy was pioneered by Persson and Peraire [93] and further developed in [140, 1]. These methods typically work by using a smoothness indicator to first detect troubled cells, and then directly discretizing a diffusive term that has been artificially added to the equations in an effort to smooth-out oscillations. One drawback of using these limiters with explicit time-stepping schemes is that due to the diffusion operator, the maximum stable time-step is $\Delta t = \mathcal{O}(h/M_{\mathrm{D}}^2)$ (i.e., the viscosity parameter is typically $\mathcal{O}(h)$) [93], where h is the grid spacing and M_{D} is the highest polynomial order of the basis polynomials, rather than the typical $\Delta t = \mathcal{O}(h/M_{\mathrm{D}})$ for hyperbolic systems.

Other variations such as minimum entropy satisfying limiters [144] have also been investigated. A complete description of what options exist for limiting DG solutions is beyond the scope of this work; the interested reader is directed to [43] for a recent and extensive review of the literature.

4.1.2 Brief summary of current work

The purpose of this work is to develop a novel limiter for discontinuous Galerkin methods that has all of the following properties:

- Provable high-order accuracy in smooth regimes;
- Robust reduction in spurious oscillations in the presence of shocks;
- Computationally efficient (i.e., only uses neighboring cells and no characteristic decompositions);
- Easy to implement in existing DG codes;
- Automatic extension to multidimensional settings including Cartesian and unstructured grids; and
- Able to incorporate existing positivity preserving limiters.

The limiter presented in this work can be viewed as a novel extension of two separate methods. First, it can be thought of as extending the finite volume Barth-Jespersen limiter [8] to the discontinuous Galerkin framework. Second, it can be viewed as an extension of the modern maximum principle preserving DG schemes developed by Zhang and Shu [143]. In particular, the limiter is based on finding for each element local upper and lower bounds using only nearest neighbor values, and then limiting the higher-order modes in such a way that the mass conservation is automatic and such that the limited solution remains within the predicted local bounds. Roughly speaking, this limiter can be viewed as belonging to the **moment or slope limiter methods**, although it has important differences with the current methods in this class. Foremost among these differences is the fact that this limiter does not need to perform hierarchical highest-to-lowest moment limiting. This gives the limiter easy extensibility to unstructured meshes.

The remainder of this chapter is organized as follows. The basic limiting strategy for a one-dimensional scalar equation is outlined in Section 4.2. Section 4.3 describes how the proposed limiter obtains high-order accuracy. Extension to multiple dimensions is shown in Section 4.4 and to general systems of hyperbolic conservation laws in Section 4.5, where the compressible Euler equations are used as the prototypical example. Section 4.6 describes how positivity-preserving ideas can be incorporated. The numerical method is validated on several standard cases for the compressible Euler equations in Section 4.8. Section 4.9 introduces a modification of this limiter to allow limiting with Characteristic variables. Section 4.10 introduces a modification of this limiter to allow it to operate on mapped grids. Conclusions are made in Section 4.11.

4.1.3 Implementation details

All of the numerical examples carried out in this work are implemented in the software package DoGPack [105]. DoGPack implements a DG method using a hierarchical modal basis with a variety of time-stepping schemes. There are Runge-Kutta methods implemented up to 5th order, SSPRK methods implemented up to 4th order as well as Lax-Wendroff DG methods implemented up to 3rd order for several important PDEs. This software package is open-source and all the code used in this work can be freely downloaded from the web.

4.2 Proposed limiter: The scalar case

The basic procedure followed to limit the solution on a single element $\mathcal{T}_i$ is given by the following sequence of steps:

Step 0. Define a set of points χ_i that will be used to approximate cell maximum and minimum values. What is meant by this is that the max and min values over this set of

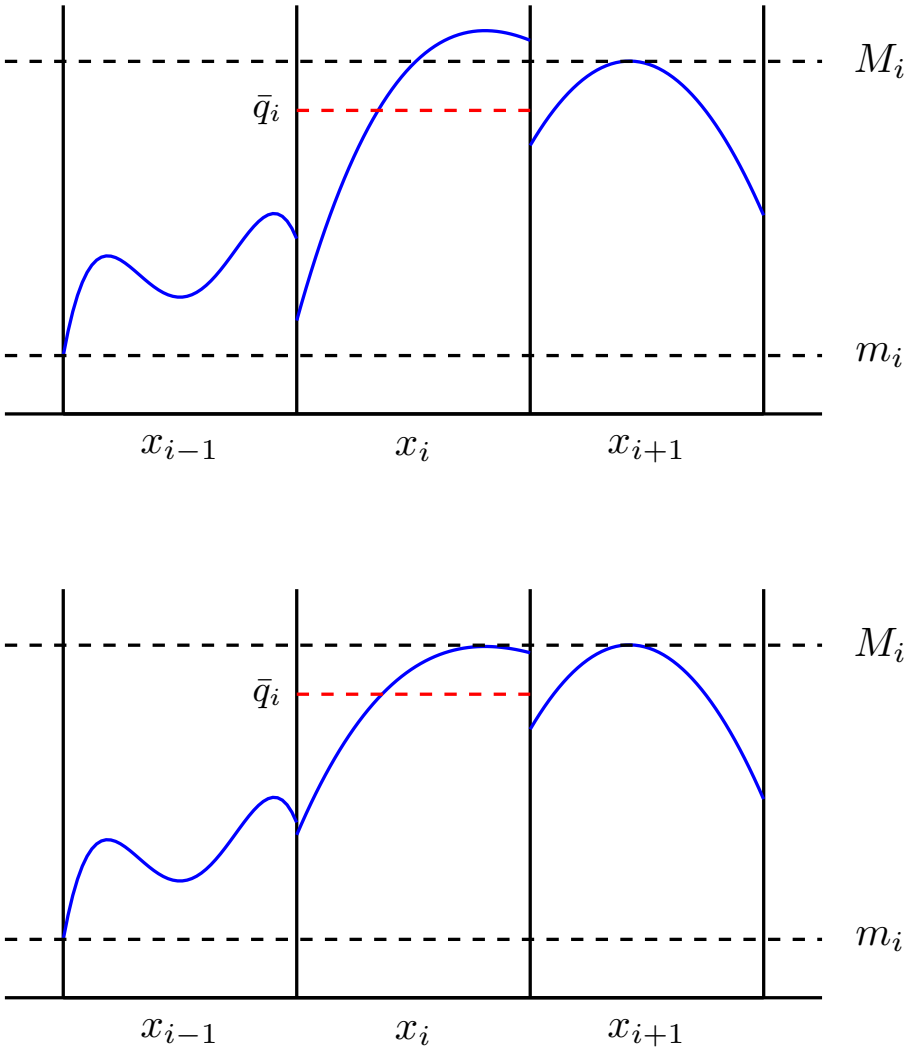


Figure 4.1: An example illustrating the inspiration for the limiter described in Section 4.2.

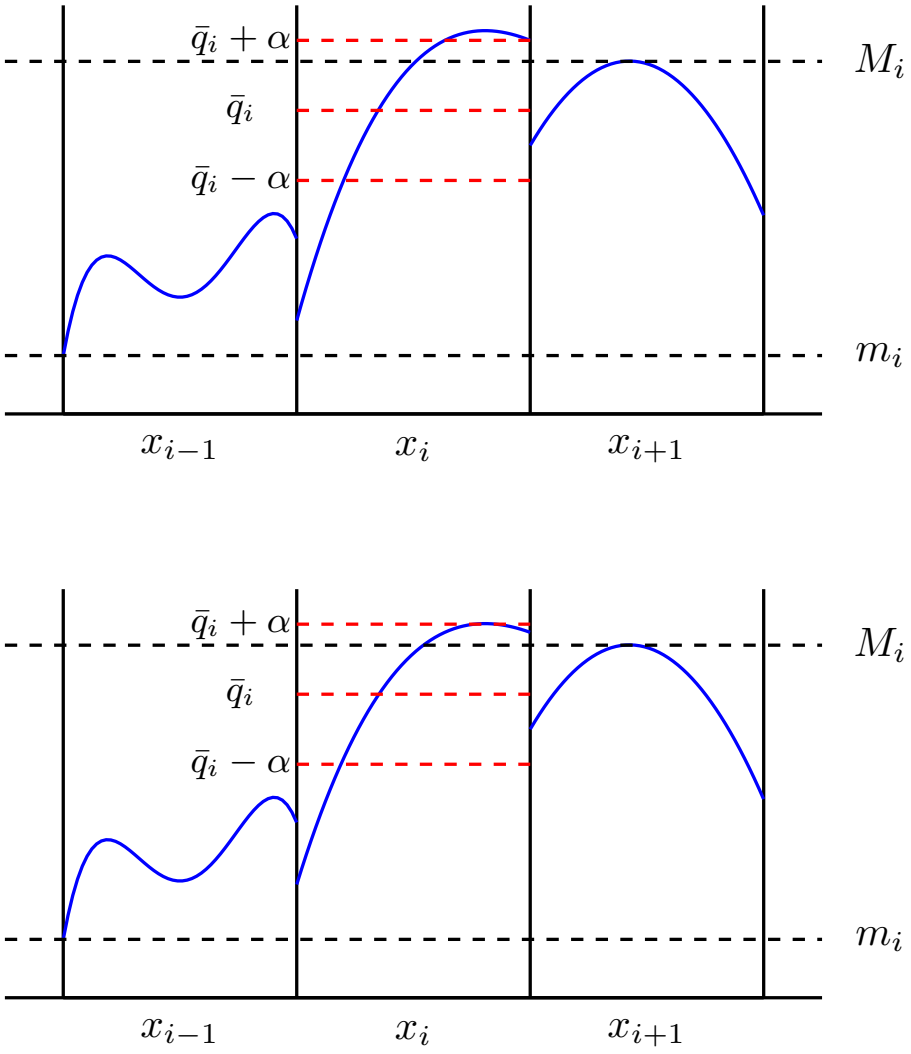


Figure 4.2: An example illustrating the role of α in the limiter described in Section 4.2.

points are computed and that these values are assumed to approximate the actual max and min values over the entire cell.

In the current work, Gaussian quadrature nodes are used. These points are augmented with corner points and quadrature points along the element boundaries (i.e., edge Gaussian quadrature points).

Step 1. For each mesh element, $\mathcal{T}_i$, compute an approximate maximum and minimum:

$$q_{M_i} := \max_{x \in \chi_i} \left\{ q^h(x)\Big|_{\mathcal{T}_i} \right\} \quad \text{and} \quad q_{m_i} := \min_{x \in \chi_i} \left\{ q^h(x)\Big|_{\mathcal{T}_i} \right\}.$$

Step 2. Consider the set $N_{\mathcal{T}_i}$ of all neighbors of $\mathcal{T}_i$, excluding $\mathcal{T}_i$ itself, and compute an approximate upper and lower bound:

$$M_i := \max \left\{ \bar{q}_i + \alpha(h),\ \max_{j \in N_{\mathcal{T}_i}} \left\{ q_{M_j} \right\} \right\}, \tag{4.2}$$

$$m_i := \min \left\{ \bar{q}_i - \alpha(h),\ \min_{j \in N_{\mathcal{T}_i}} \left\{ q_{m_j} \right\} \right\}. \tag{4.3}$$

The scalar function $\alpha(h) \geq 0$ is a tolerance function that will be described shortly. The most aggressive limiter considered sets $\alpha = 0$.

Step 3. Define

$$\theta_{M_i} := \phi\left(\frac{M_i - \bar{q}_i}{q_{M_i} - \bar{q}_i} \right) \quad \text{and} \quad \theta_{m_i} := \phi\left(\frac{m_i - \bar{q}_i}{q_{m_i} - \bar{q}_i} \right), \tag{4.4}$$

where $0 \leq \phi(s) \leq 1$ is a cutoff function. In this work, we use the function

$$\phi(s) := \min\left\{ \frac{s}{1.1}, 1 \right\}, \tag{4.5}$$

which is motivated by the work of [84]. The form of this function becomes important in the multidimensional setting, which will be described shortly.

Step 4. Define the *rescaling parameter* as

$$\theta_i := \min\left\{ 1,\ \theta_{m_i},\ \theta_{M_i} \right\}. \tag{4.6}$$

Step 5. Finally, rescale the approximate solution on the element $\mathcal{T}_i$ as

$$\tilde{q}^h(x)\Big|_{\mathcal{T}_i} := \bar{q}_i + \theta_i \left(q^h(x)\Big|_{\mathcal{T}_i} - \bar{q}_i \right). \tag{4.7}$$

Before delving into the finer details of the method, a few remarks are in order.

Remark 1. *The choice of points used to define the approximate cell maximum and minimum values in* **Step 0** *is not unique.*

Indeed, the purpose of this step is to construct an approximate upper and lower bound for the solution. This limiter was originally implemented using neighboring cell averages to define bounds for each cell, but it was found that those bounds tended to be quite diffusive near shocks. Directly sampling neighboring cells (that are potentially oscillating) allows one to retain sharper features in the solution.

Remark 2. *The presence of a non-zero value of* α *in* **Step 2** *is required to obtain high-order accuracy at local extrema.*

This issue will be elaborated upon in Section 4.3.

Remark 3. *When the proposed scheme is compared to the recent maximum principle preserving (MPP) limiters [143], significant differences come into play in* **Step 2** *and* **Step 3**.

- For the MPP methods, **Step 2** does not exist as the bounds enforced are fixed *a priori* global bounds, whereas the scheme introduced here follows the Barth-Jespersen idea [8] and uses this step to estimate *local* upper and lower bounds for the solution.

- In **Step 3**, the MPP limiters use hard upper and lower bounds for the solution that are known *a priori*, whereas here the results from **Step 2** are used to estimate these.

- In the MPP literature, authors normally set $\phi(s) = \min\{1, s\}$, whereas for the limiter introduced here the form of this function is important when pushing to multiple dimensions. Section 4.4 elaborates on this idea.

Additionally, the newly proposed limiter is designed to capture shocks, whereas the original MPP limiter for DG methods was solely designed to preserve the maximum principle property of hyperbolic problems. In work using the MPP limiter, additional limiting was necessary in order to suppress unphysical oscillations.

Remark 4. *The rescaling in* **Step 5** *does not affect the total mass.*

The consequence of this is that the limiter will be automatically mass conservative, which is an important property for hyperbolic solvers. This can be observed by simply integrating equation (4.7) over a single element.

4.3 On the importance of α for retaining high-order accuracy

This section describes the importance of α for obtaining a method that is genuinely high-order accurate. Many limiters exhibit the so-called "clipping" phenomenon, where smooth extrema are cut off because the limiter turns on and damps out these values. Recent efforts have been put forth to mitigate this clipping phenomenon for other discretizations, including the piecewise parabolic method (PPM) [37].

This section will argue that high-order accuracy will be retained provide that the scalar function $\alpha(h) : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ vanishes slower than $\mathcal{O}(h^2)$. This observation was inspired by recent work on extensions of the Barth-Jespersen limiter to finite volume methods [84], and the performance of this limiter is demonstrated by testing it on a smooth solution in 1D.

In one dimension one can break down any smooth function on a bounded domain into monotonic regions and finitely many points of local extrema. Let us begin with a discussion of monotonic regions, and then turn to isolated local extrema.

4.3.1 Performance in monotonic regions

In regions where the exact solution is monotonic, the proposed limiter does not effect the asymptotic convergence rate, independent of the choice of $\alpha \geq 0$. Without loss of generality, consider a region with the solution is increasing. That is to say, consider a region where the exact solution satisfies $q'(x) > 0$ for all $x \in [x_{i-3/2}, x_{i+3/2}]$. One may then inspect the performance of the limiter on element $\mathcal{T}_i = [x_{i-1/2}, x_{i+1/2}]$. In order to limit this element, first observe that $M_i \geq q(x_{i+3/2})$ and $q_{M_i} = q(x_{i+1/2})$ because the function is non-decreasing. (If $\alpha = 0$, one can conclude that $M_i = q(x_{i+3/2})$.) Next, perform Taylor expansions about $x = x_i$ to compare these values. After doing so, we find that the ratio of the deviation of the predicted and local maximum to the cell averages is given by

$$\begin{aligned}\frac{M_i - \bar{q}_i}{q_{M_i} - \bar{q}_i} \geq \frac{q(x_{i+3/2}) - \bar{q}_i}{q_{M_i} - \bar{q}_i} &= \frac{(q(x_i) - \bar{q}_i) + \frac{3h}{2} q'(x_i) + \mathcal{O}(h^2)}{(q(x_i) - \bar{q}_i) + \frac{h}{2} q'(x_i) + \mathcal{O}(h^2)} \\ &= \frac{\frac{3h}{2} q'(x_i) + \mathcal{O}(h^2)}{\frac{h}{2} q'(x_i) + \mathcal{O}(h^2)} = \frac{3 + \mathcal{O}(h)}{1 + \mathcal{O}(h)} = 3 + \mathcal{O}(h),\end{aligned}$$

where the well-known fact that $\bar{q}_i = q(x_i) + \mathcal{O}(h^2)$ was used, along with the assumption that $q'(x_i) \neq 0$. Similarly, $\frac{m_i - \bar{q}_i}{q_{m_i} - \bar{q}_i} \geq 3 + \mathcal{O}(h)$. Finally, because $\phi(s) = 1$ for all $s \geq 1.1$, one obtains the asymptotic result that $\theta_i = 1$ for this element. This means the limiter has zero effect on the solution, assuming the solution is smooth so the Taylor series manipulations that have been performed are valid.

4.3.2 Performance in regions near smooth extrema

This subsection describes the importance of using a non-zero value of α to maintain genuine high-order accuracy. If α is chosen to be too large, then oscillations in a given cell will never be clamped down, whereas if α vanishes too quickly, then smooth extrema will be clipped. The goal is to find necessary conditions for maintaining high-order accuracy at smooth extrema.

To begin, consider a Taylor expansion of a sufficiently smooth function $q : \mathbb{R} \to \mathbb{R}$ at an extremum $x = \xi_0$:

$$q(x) = q(\xi_0) + q'(\xi_0)(x - \xi_0) + \frac{1}{2}q''(\xi_0)(x - \xi_0)^2 + \cdots .$$

Given that $q'(\xi_0) = 0$, we observe that asymptotically $|q(x) - q(\xi_0)| = \mathcal{O}(h^2)$ if $|x - \xi_0| = \mathcal{O}(h)$. This leads us to consider functions of the form $\alpha(h) = \mathcal{O}(h^r)$, where $r < 2$ in order to maintain

$$|q(x) - q(\xi_0)| = \mathcal{O}(h^2) \leq |\alpha(h)| \quad \text{for all} \quad |x - \xi_0| = \mathcal{O}(h).$$

Note that in the 2D Cartesian case: $h = \max(\Delta x, \Delta y)$, where Δx and Δy are the mesh widths in each coordinate direction.

That is, as long as $\alpha(h)$ goes to zero *slow enough*, then our limiter will eventually turn off and not clip smooth extrema. However, if we were to set $\alpha = 0$, then our limiter will induce clipping at smooth extrema (see the example illustrated below in Section 4.3). We do need α to go to zero as $h \to 0$, because otherwise oscillations would persist for problems with discontinuities.

4.3.3 Maintaining high order accuracy in general

In general if a function being approximated is sufficiently smooth then the solution will have a convergent Taylor series

$$q(x) = q(\xi_0) + q'(\xi_0)(x - \xi_0) + \frac{1}{2}q''(\xi_0)(x - \xi_0)^2 + \cdots .$$

As the solution is refined more and more $q'(\xi_0)$ should stay the same, but the maximum magnitude of $(x - \xi_0) = O(h)$. What this means is that the terms like $(x - \xi_0)^k$ where $k \geq 2$ will shrink faster than $\alpha(h)$.

The choice of α is not unique, and if desired, could be tuned for any given problem. The trade-off is that if α is too large, then true oscillations will only be damped provided the mesh is fine enough. Our choice of $\alpha(h) = \mathcal{O}(h^{1.5})$ is consistent with the recommended switches that turn off similar Barth-Jespersen type limiters for finite volume schemes [84]. Extensive numerical tests indicate this choice finds a good balance between the need to limit near shocks, while being soft enough to allow high-order accuracy on coarse grids. **In the case of smooth problems, our limited solution will always converge to the unlimited solution even near local extrema.** We illustrate this point with a simple case study.

4.3.4 On the importance of α for retaining high-order accuracy: A simple 1D case study

We analyze the performance of the proposed limiter on the linear advection equation (4.1) on $x \in [0, 1]$ with periodic boundary conditions and with the initial condition

$$q(x) = \begin{cases} \cos^6\left(\frac{(x-0.5)\pi}{0.16}\right) & \text{if } |x - 0.5| < 0.08, \\ 0 & \text{otherwise.} \end{cases} \tag{4.8}$$

In Table 4.1 we present a convergence study for various grid resolutions after advecting the solution through one full period (i.e., solve to $t = 1$). For this example we use the 10-stage SSP fourth-order Runge-Kutta scheme (SSPRK4) developed by Ketcheson [62] and hold a constant CFL number of 0.4. We use a hierarchical modal basis of $\mathbb{P}_3^2$ and so we hope to observe 4th order convergence in refinement studies run on this example. We compare the results for the unlimited solution, the solution with $\alpha = 0$, the solution with $\alpha(h) = 50h^{1.5}$, and the solution with $\alpha(h) = 80h^{1.5}$.

We observe that after enough refinement the limited and unlimited solutions agree with each other, whereas for very coarse grids, the order of accuracy is slightly degraded. It is only after the limiter "turns off" that we dial in on high-order accuracy. This cutoff point is a function of the curvature of the exact solution at the local extrema. This example serves to illustrate that after enough refinement, there is no difference between the limited and unlimited solutions, at least for this toy problem.

mx	**No Limiter**	**Order**	$\alpha = 0$	**Order**	$\alpha = 50h^{1.5}$	**Order**	$\alpha = 80h^{1.5}$	**Order**
5	3.78×10^{-1}	—	8.73×10^{-1}	—	3.78×10^{-01}	—	3.78×10^{-01}	—
8	5.61×10^{-1}	-0.84	9.00×10^{-1}	-0.07	5.61×10^{-01}	-0.84	5.61×10^{-1}	-0.84
14	2.32×10^{-1}	1.58	8.22×10^{-1}	0.16	2.34×10^{-01}	1.56	2.32×10^{-1}	1.58
24	5.38×10^{-2}	2.71	7.04×10^{-1}	0.29	1.74×10^{-01}	0.56	5.38×10^{-2}	2.71
42	4.24×10^{-3}	4.54	4.78×10^{-1}	0.69	7.82×10^{-02}	1.43	7.60×10^{-3}	3.50
73	3.65×10^{-4}	4.44	2.24×10^{-1}	1.38	2.19×10^{-02}	2.30	3.65×10^{-4}	5.49
127	3.89×10^{-5}	4.04	8.84×10^{-2}	1.68	2.73×10^{-03}	3.76	3.89×10^{-5}	4.04
222	4.10×10^{-6}	4.03	2.97×10^{-2}	1.95	4.10×10^{-06}	11.64	4.10×10^{-6}	4.03
388	4.38×10^{-7}	4.01	9.37×10^{-3}	2.07	4.38×10^{-07}	4.01	4.38×10^{-7}	4.01
679	4.65×10^{-8}	4.01	2.76×10^{-3}	2.19	4.65×10^{-08}	4.01	4.65×10^{-8}	4.01

Table 4.1: Convergence for the 1D advection problem in Section 4.3.4. All errors are L_2 norm errors. We see that a nonzero value of α is required to allow the limiting to completely turn off when the solution is properly resolved and that a larger value of α allows this to happen sooner during the refinement process.

4.3.5 On the impact of α: An illustrative example

In this section we include an example that shows how well this limiter works on solutions that contain regions of varying degrees of smoothness. Ideally we would like to see fast convergence in smooth regions and no spurious oscillations near regions that have reduced smoothness. This example serves to illustrate that for a fixed mesh size, large values of α are good at capturing smooth extrema but introduce additional oscillations, whereas smaller values of α clamp down on oscillations but also induce clipping at the location of smooth extrema. In practice, our approach has been to try several values of α for a given problem. Our findings indicate that smaller values of α can be used for problems that have small curvature (i.e., $|q''(x)|$ is small) without impacting the solution at smooth extrema. For reference the function $\sin(\pi x)$ on the domain $[-1, 1]$ discretized with 200 cells would be an example of a function with small curvature. If the exact solution has a large amount of curvature, then a larger value of α (or a finer mesh) is required to turn off the limiter at local extrema.

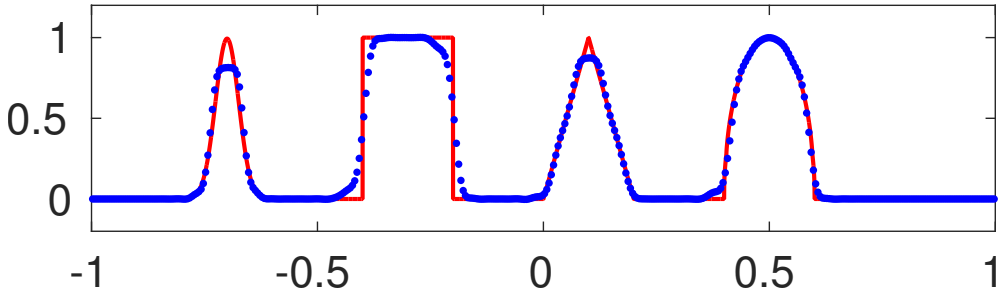
We again consider the linear advection equation (4.1). The domain for this problem is $x \in [-1, 1]$, and we again use periodic boundary conditions. This time we use initial conditions that have four profiles of different regularities [60]:

$$q(x) = \begin{cases} \frac{1}{6}\left[G(x,\beta,z-\delta) + G(x,\beta,z+\delta) + 4G(x,\beta,z)\right], & -0.8 \le x \le -0.6; \\ 1, & -0.4 \le x \le -0.2; \\ 1 - |10(x-0.1)|, & 0 \le x \le 0.2; \\ \frac{1}{6}\left[F(x,\alpha,a-\delta) + F(x,\alpha,a+\delta) + 4F(x,\alpha,a)\right], & 0.4 \le x \le 0.6; \\ 0, & \text{otherwise.} \end{cases} \tag{4.9a}$$

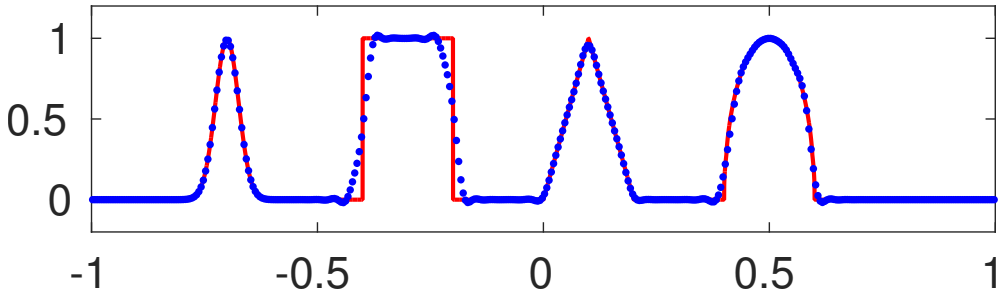
$$G(x,\beta,z) = e^{-\beta(x-z)^2}; \tag{4.9b}$$

$$F(x,\alpha,a) = \sqrt{\max(1-\alpha^2(x-a)^2, 0)}, \tag{4.9c}$$

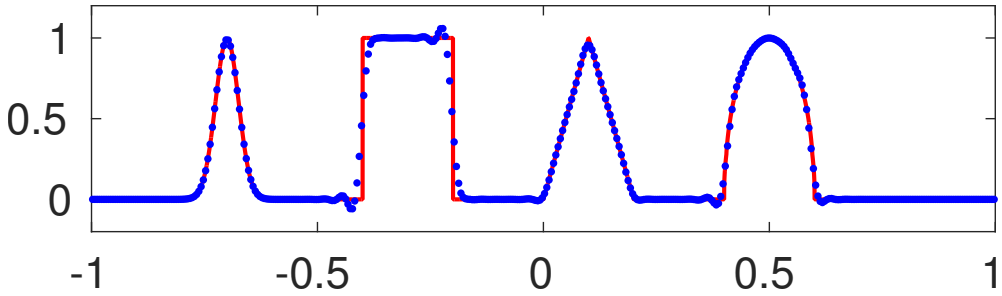
where $a = 0.5$, $z = -0.7$, $\delta = 0.005$, $\alpha = 10$ and $\beta = \log_{10}(2)/(36\,\delta^2)$. The first profile is a smooth Gaussian, the second is a discontinuous square wave, the third is a cone with a discontinuous derivative, and the fourth is a cosine bell with continuous first derivative. We advect these profiles one complete rotation and display our findings in Figure 4.3. Here, we keep a fixed mesh of size $m_x = 100$ and show the solution with three different values of α and one unlimited solution. It appears that $\alpha = 100h^{1.5}$ gives satisfactory results, eliminating the large oscillations that can be seen with $\alpha = 1000h^{1.5}$, and yet it still sharply fits the exact solution (the red line) on the cone and cosine bell.



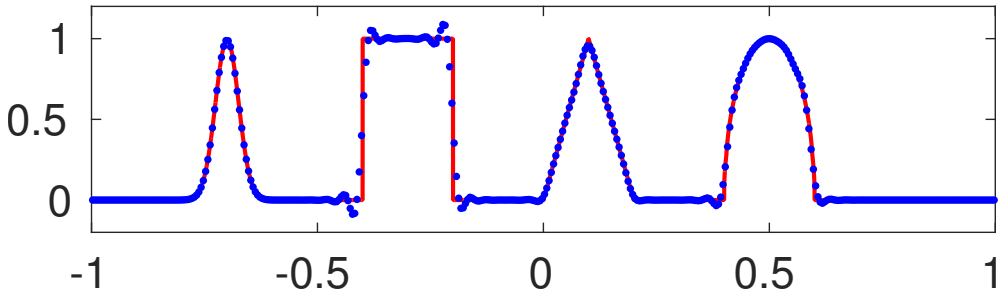
(a) $\alpha = 10h^{1.5}$



(b) $\alpha = 100h^{1.5}$



(c) $\alpha = 300h^{1.5}$



(d) No Limiter

Figure 4.3: Four profiles of different regularity are advected one full rotation around the domain. The red line is the exact solution. Results are shown for three values of α. For these examples $m_x = 100$ ($h = \frac{1}{50}$) and 4 points were used per cell with third order basis functions.

4.4 On the importance of ϕ for multidimensional cases

An important distinction between the single and multidimensional cases is that in higher dimensions, shocks and discontinuities are no longer isolated to single cells. To illustrate this, consider the following initial condition:

$$q(x,y) = \begin{cases} 1, & \text{if } x \leq 0, \\ 0, & \text{otherwise.} \end{cases} \tag{4.10}$$

Now, consider a Cartesian mesh with cells of width Δx and height Δy:

$$C_{ij} := \left[\left(i-\frac{1}{2}\right)\Delta x, \left(i+\frac{1}{2}\right)\Delta x\right] \times \left[\left(j-\frac{1}{2}\right)\Delta y, \left(j+\frac{1}{2}\right)\Delta y\right], \quad i,j \in \mathbb{Z}. \tag{4.11}$$

After projecting the initial conditions onto $\mathcal{W}^h$ with $M_{\mathrm{D}} > 1$ (i.e., see definition (2.8)), every cell $C_{0j}\ \forall j$ will exhibit Gibbs phenomena; this is because the initial discontinuity bisects the cells $C_{0j}\ \forall j$. For an example similar to this situation look at Figure 4.4 (a). We now consider the effect of applying our limiter to this solution. We focus on what happens to the solution on cell C_{00}.

In **Step 2** of the proposed limiter, we compute upper and lower bounds of the solution using nearest neighbors, which in this case satisfy

$$\max\left\{q^h\big|_{C_{00}}\right\} = \max\left\{q^h\big|_{C_{0\,\pm 1}}\right\} \quad \text{and} \quad \min\left\{q^h\big|_{C_{00}}\right\} = \min\left\{q^h\big|_{C_{0\,\pm 1}}\right\}.$$

In **Step 3** of the limiter we will compute $\left(\frac{M_i-\bar{q}_i}{q_{M_i}-\bar{q}_i}\right) = 1$ and $\left(\frac{m_i-\bar{q}_i}{q_{M_i}-\bar{q}_i}\right) = 1$. That is, for these initial conditions, the predicted upper bound for cell C_{00} will not clamp down on any oscillations present in this cell, unless we are careful about deciding on the correct structure of ϕ.

The important observation to make here is that it is *the ratio* of the deviation from cell averages (the value θ in equation (4.4) of **Step 3**) that is the relevant quantity to study. For this problem, it is identically equal to 1 for every mesh element. Indeed, what this tells us is that it is necessary for $\phi(1) < 1$ if we will have any hope of clamping down on the unavoidable oscillations. We observe that even with $\phi(1) < 1$, the oscillations are not completely suppressed after a single time step; however, after several time-steps the artificial oscillations decay exponentially to zero. In practice, we observe that this is sufficient to construct robust simulations.

As a simple demonstration of the importance of this choice of ϕ, we consider linear advection in two-dimensions on a Cartesian grid. Consider the scalar advection equation

$$\frac{\partial q}{\partial t} + \frac{\partial q}{\partial x} + \frac{\partial q}{\partial y} = 0, \quad \text{for} \quad (x,y) \in [0,1] \times [0,1], \tag{4.12}$$

with double-periodic boundary conditions and discontinuous initial conditions:

$$q_0(x,y) = \begin{cases} 1 & \text{if } x \in [0.3, 0.7], \\ 0 & \text{otherwise.} \end{cases} \tag{4.13}$$

In Figure 4.4 we show results for a grid of size 60×60 for a fourth-order ($\mathbb{P}_3$) method, comparing the choices of $\phi(s) = \min\{1, s\}$ and $\phi(s) = \min\{1, \frac{s}{1.1}\}$. This result shows that the second choice is necessary for pushing our results to multiple dimensions. Other choices exist, but we must have that $\phi(1) < 1$. In the remainder of our numerical simulations, we choose $\phi(s) = \min\{1, \frac{s}{1.1}\}$.

4.5 Extensions to systems of equations

As a prototypical example of a hyperbolic system of equations, consider the compressible Euler equations as introduced in Section 3.4.3. In these expressions ρ is the mass density, $\rho\mathbf{u} := (\rho u_1, \rho u_2, \rho u_3)$ is the momentum density, $\mathcal{E}$ is the total energy density, $\mathbf{u} := (u_1, u_2, u_3)$ is the fluid velocity, and p is the pressure.

The simplest implementation of the proposed limiter for this system would be to apply the scalar limiting procedure to each conserved variable independently. Indeed, initial tests employed this approach. However, after extensive testing for this system, it was discovered that using the primitive variables to define one high-order damping parameter θ yields better results. An example is provided in Section 4.7.1 to verify this claim.

The extension of the proposed limiter to systems of equations can be summarized as follows:

Step 0. Select a set of variables to use in the bounds checking: we denote this mapping of the conserved variables $g(q)$. In fluid dynamics, primitive variables are the preferred choice due to their Galilean invariance. Select a set of points χ_i that will be used to approximate cell maximum and minimum values. In this work the limiter always uses: corners, internal, and edge Gaussian quadrature points.

Step 1. For each element i and each component ℓ of g compute:

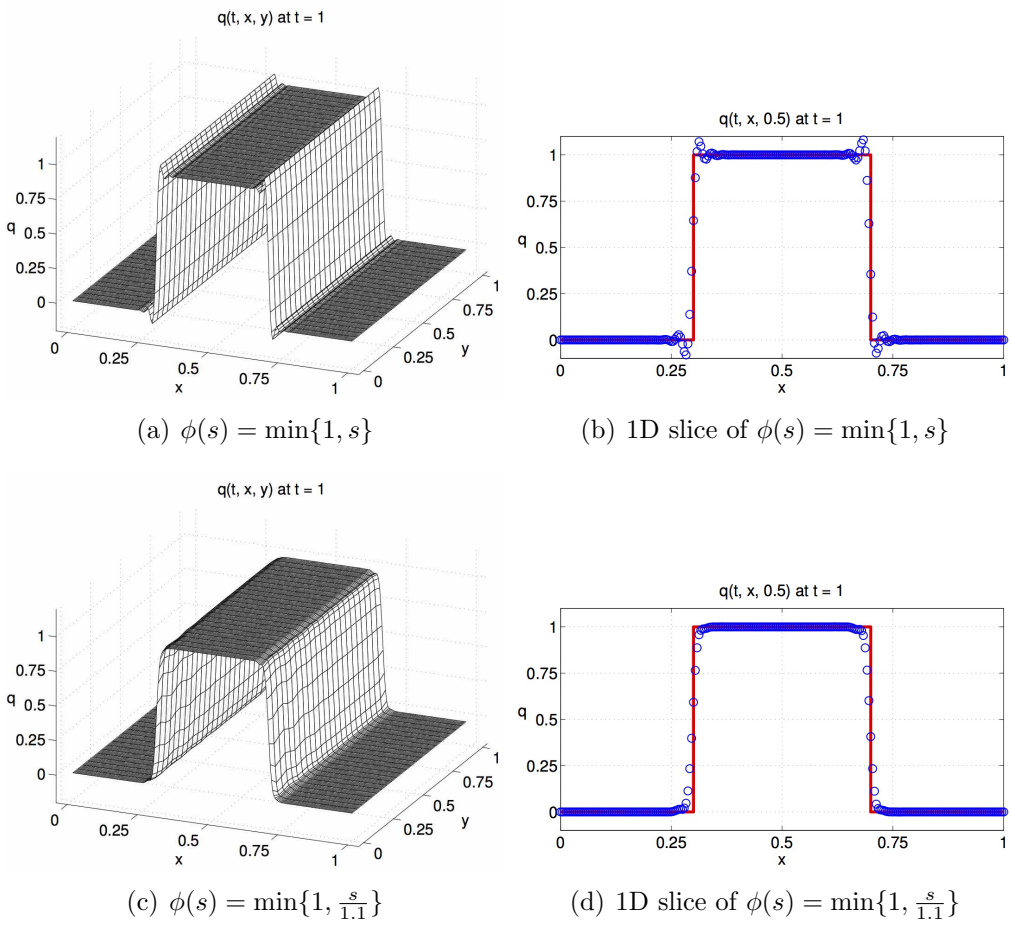
$$g^\ell_{M_i} := \max_{x\in\chi_i} \left\{ g^\ell(q^h(x))\Big|_{\mathcal{T}_i} \right\} \quad \text{and} \quad g^\ell_{m_i} := \min_{x\in\chi_i} \left\{ g^\ell(q^h(x))\Big|_{\mathcal{T}_i} \right\}. \tag{4.14}$$

Step 2. For each element i and each component ℓ of g compute an approximate upper and lower bound over the set of neighbors, $N_{\mathcal{T}_i}$ (excluding the current cell $\mathcal{T}_i$):

$$M^\ell_i := \max\left\{ \bar{g}^\ell_i + \alpha(h),\ \max_{j\in N_{\mathcal{T}_i}} \left\{ g^\ell_{M_j} \right\} \right\}, \tag{4.15}$$

$$m^\ell_i := \min\left\{ \bar{g}^\ell_i - \alpha(h),\ \min_{j\in N_{\mathcal{T}_i}} \left\{ g^\ell_{m_j} \right\} \right\}, \tag{4.16}$$

where $\bar{g}_i^\ell$ is the cell average of g^ℓ over cell $\mathcal{T}_i$. In the Cartesian grid case, we define $N_{\mathcal{T}_i}$ to be the set of cells that share a common edge with $\mathcal{T}_i$. This provides good results in general, and has the nice property that it does not extend the effective stencil size. For unstructured grids, we define $N_{\mathcal{T}_i}$ as the set of all cells that share a common node with $\mathcal{T}_i$. If this choice is replaced by elements that share common edges only, then the

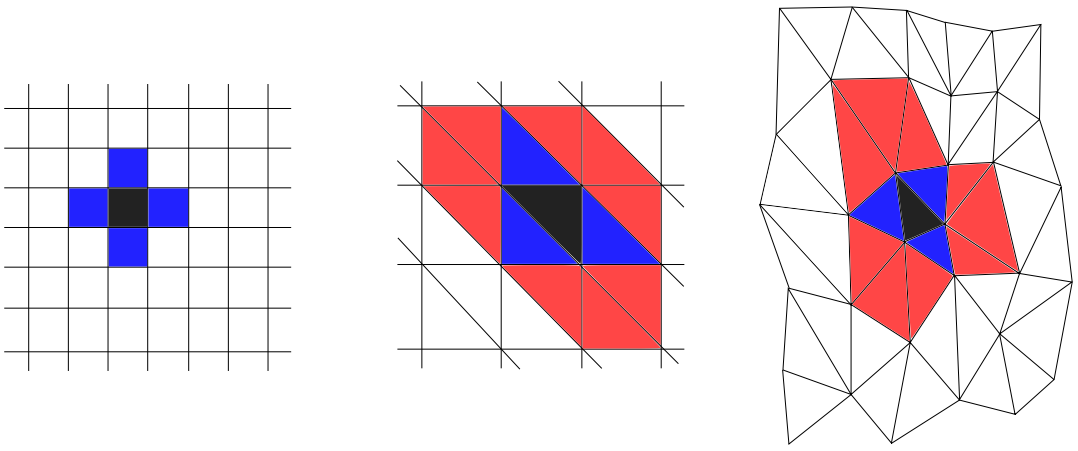


(a) $\phi(s) = \min\{1, s\}$ (b) 1D slice of $\phi(s) = \min\{1, s\}$

(c) $\phi(s) = \min\{1, \frac{s}{1.1}\}$ (d) 1D slice of $\phi(s) = \min\{1, \frac{s}{1.1}\}$

Figure 4.4: Square wave test case for scalar advection with double periodic boundary conditions. The solution in Panels (a) and (b) is computed using $\phi(s) = \min\{1, s\}$, while the solution in Panels (c) and (d) is computed using $\phi(s) = \min\{1, \frac{s}{1.1}\}$. The simulation is run to a final time of $t = 1$, at which point the solution should return to the initial conditions. The choice in (a) and (b) satisfies $\phi(1) = 1$, whereas the choice in (c) and (d) satisfies $\phi(1) < 1$. The first choice permits oscillations to remain in the solution, whereas the second choice eventually suppresses the unphysical oscillations. The remainder of the simulations used in this work will use the function $\phi(s) = \min\{1, \frac{s}{1.1}\}$.

results are more diffusive. A pictorial diagram is displayed in Figure 4.5.



(a) (b) (c)

Figure 4.5: Limiting stencil. The choice of the stencil used to construct local bounds of the solution in **Step 2** is described in the above panels. For structured grids (panel (a)), the limiter only looks at the maximum and minimum values of the neighbors that share a common edge. For unstructured grids (panels (b) and (c)), better solutions are obtained if the limiter also considers neighbors that share common vertices. The larger stencil performs less limiting, and therefore the results are less diffusive and allow more subcell structure to form in the solution.

Step 3. For each element i compute:

$$\theta_{M_i} := \min_{\ell} \left\{ \phi \left(\frac{M_i^{\ell} - \bar{g}_i^{\ell}}{g_{M_i}^{\ell} - \bar{g}_i^{\ell}} \right) \right\} \quad \text{and} \quad \theta_{m_i} := \min_{\ell} \left\{ \phi \left(\frac{m_i^{\ell} - \bar{g}_i^{\ell}}{g_{m_i}^{\ell} - \bar{g}_i^{\ell}} \right) \right\}. \tag{4.17}$$

Step 4. For each element i compute:

$$\theta_i := \min \left\{ 1,\ \theta_{m_i},\ \theta_{M_i} \right\}. \tag{4.18}$$

Step 5. For each element i and each component ℓ of q compute

$$\tilde{q}^h(x)\Big|_{\mathcal{T}_i} := \bar{q}_i + \theta_i \left(q^h(x)\Big|_{\mathcal{T}_i} - \bar{q}_i \right). \tag{4.19}$$

Note that the limiter is using the values of the variable g to determine the amount of limiting that needs to be done (i.e., size of θ_i), but is actually applying the scaling to the conserved variables, thereby retaining mass conservation at the discrete level.

4.5.1 Why not just use characteristic variables?

Typically limiting methods for systems of equations will use some approximation of the systems characteristic variables to limit because these are the variables that satisfy a local maximum principal. However, for the limiter introduced in this chapter it is not straightforward to do this in multiple dimensions. The difficulty originates from the fact that for a two dimensional hyperbolic PDE there is one set of characteristic variables for every direction. Because of this the limiter has been developed without using the characteristic variables. Instead the limiter has been constructed so that it could be used in connection with any set of variables specified by the user. However note that this does not mean that the limiter is guaranteed to perform well with any set of variables. Some work will have to be done to find an appropriate set of variables for the specific PDE. The main system used when developing this limiter was the Euler equations and it has been found that using the primitive variables to limit the solution performs very well for this set of equations. For a different set of equations, however, it will be up to the user to pick which variables to use for limiting. There is also a recently developed modification to this method to allow using characteristic variables to limit in multiple dimensions. That will be discussed in Section 4.9.

4.6 Incorporating positivity-preservation to the limiter

Given the framework of the limiter as outlined in Section 4.5, the addition of the recent and popular positivity-preserving methods developed by Zhang and Shu [143] is relatively straightforward. For example, in order to guarantee positivity of a scalar function that has a positive mean, one may simply replace the computation of the rescaling parameter in equation (4.6) of **Step 4** to include information concerning a *global* as opposed to a *local* bound of the solution (cf. Remark 3). This section describes the necessary details that are required to merge the two limiters.

Recall that the maximum principle preserving (MPP) limiter proposed in [143] chooses a value $\theta_i^p \in [0,1]$ in the expansion

$$\bar{q}_i + \theta_i^p \left(q^h(x)\Big|_{\mathcal{T}_i} - \bar{q}_i \right)$$

in such a way that positivity is retained at a finite set of points that are chosen a priori. The choice of this parameter assumes that a *global* upper bound is already known. In order to fit this MPP limiter to the proposed limiter, one must compute θ_i^p as in [143], and then modify **Step 4** to include a single additional parameter to take the minimum over:

$$\theta_i := \min\left\{1,\ \theta_{m_i},\ \theta_{M_i},\ \theta_i^p\right\}.$$

The limited solution $\tilde{q}^h(x)\Big|_{\mathcal{T}_i}$ defined in such a manner will be positive at all of the sampled points assuming that the cell averages are positive. This is because $\theta_i \leq \theta_i^p$, and therefore

the resulting limited function approximation will have point values that are at least as close to the cell mean as the function approximation produced with only θ_i^p. Finally, it can be proven that cell averages remain positive throughout the simulation because strong stability preserving (SSP) Runge-Kutta methods for time-stepping.

Extensions to the compressible Euler equations to maintain positive pressure require expanding the unknown variables as a quadratic function of θ. This method is used to maintain positivity when working with the Euler equations. The interested reader is referred to [143] for further details on how this is accomplished.

4.7 Compressible Euler Equations

4.7.1 Primitive vs. conservative variables for computing θ: A 1D example

In order to test the limiter performance using either primitive or conservative variables to determine θ_i, consider the one-dimensional shock-entropy problem found in [118, 80, 28] and also used in Section 3.4.3, which highlights the interplay between shock-capturing and feature-preservation. The initial conditions for this problem are given by

$$(\rho, u_1, p) = \begin{cases} (3.857143, 2.629369, 10.3333) & x < -4, \\ (1 + \varepsilon \sin(5x), 0, 1) & x \geq -4. \end{cases}$$

This problem will be run using the common practice to set $\varepsilon = 0.2$ and discretize the domain $[-5, 5]$ using 200 cells. The computation will use a 4th order DG method using SSPRK4. The final time for this simulation is $t = 1.8$. This test problem involves a shockwave interacting with a smooth entropy wave. As this interaction occurs, the result is the formation of highly oscillatory waves after the shock, where high-order accuracy should produce a benefit, yet limiters often smear out these oscillations. Ideally a limiter should be able to pick out and limit only non-physical oscillations, while simultaneously capturing the oscillatory post shock features.

The results of two sets of simulations, one where the θ_i are computed from sampling conserved variables and one from sampling primitive variables, are shown in Figure 4.6, which zooms in on the interesting features of the problem. Additionally, it is important to highlight the effect of the choice of α for each version of the limiter. Identical to what happens with the free "M" parameter in WENO limiters [102], larger values of α retain richer subcell resolution whereas smaller values of α tend to add more diffusion to the system. The trade-off is that some diffusion is necessary to deal with inevitable oscillations, whereas too much can smear out the solution. These results clearly show the advantage of using primitive variables in defining θ_i and they help to illustrate that a substantial range of α values can yield good results. Note that the reference solution (in red) is computed with a 1d 4th order WENO scheme using 10000 grid cells.

4.7.2 On the importance of α: A simple 2D case study

The convergence properties shown in Section 4.3.4 should still be observable in 2D and for systems. To verify this a 2D Euler example that has a smooth exact solution is considered. This example involves an ideal gas with $\gamma = 1.4$, on $(x, y) \in [0, 2] \times [0, 2]$ with double-periodic boundary conditions and the following initial conditions:

$$(\rho, u_1, u_2, p) = \big(1 + 0.2 \sin(\pi(x + y)), 0.7, 0.3, 1.0\big). \tag{4.20}$$

In the exact solution u_1, u_2, and p remain constant for all space and time, while the density, ρ, is advected with the constant fluid velocity given by $(u_1, u_2) = (0.7, 0.3)$. The numerical L_2 errors after one revolution (i.e., time $t = 2$) are shown in Table 4.2. To produce these results one must use a 4th order DG scheme with SSPRK4. Again observe that there must be a nonzero value of α to asymptotically match the unlimited solution's convergence even for this very simple test problem.

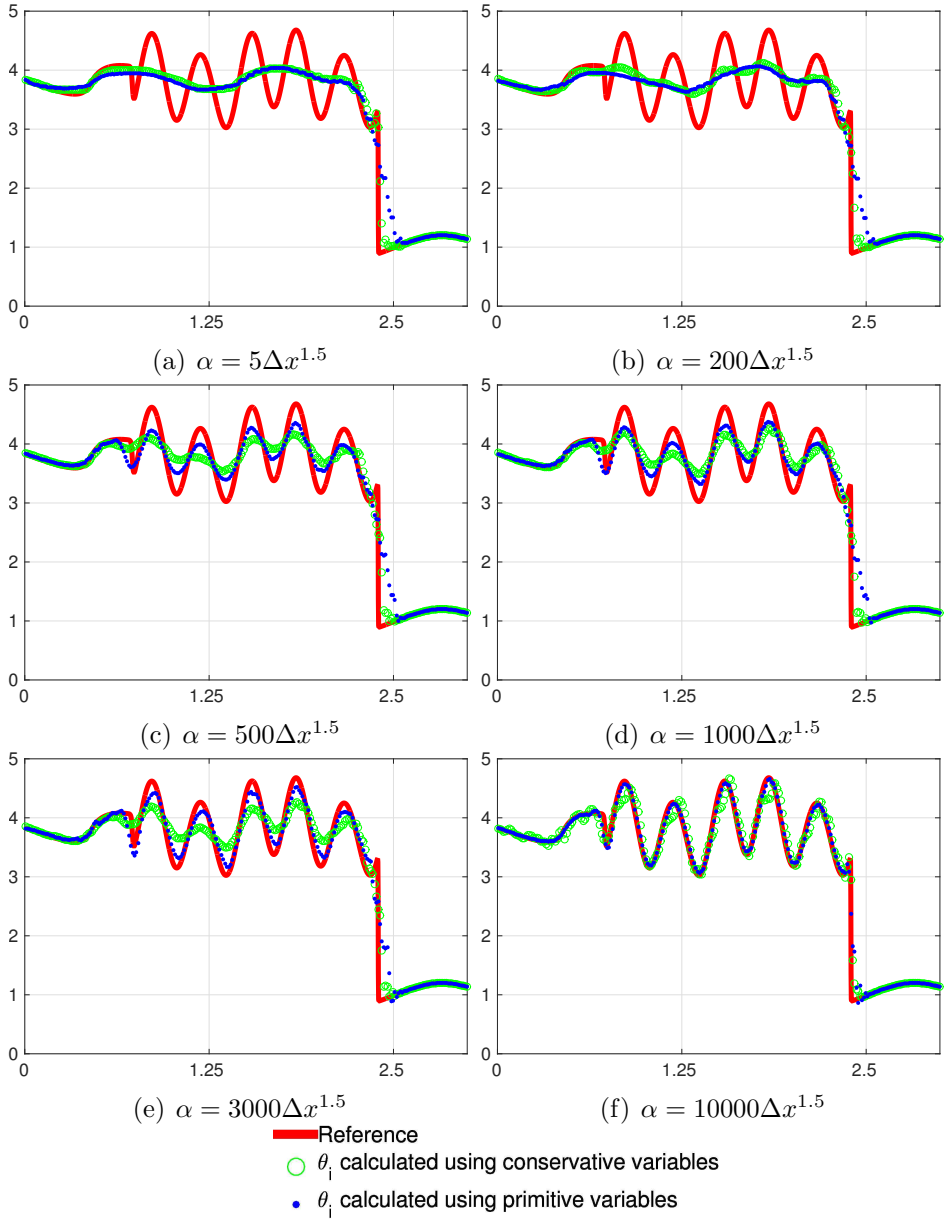


Figure 4.6: Shock-entropy problem solutions computed by a 4th order DG scheme implemented one 200 cells and with 4 points plotted per cell. In these panels the benefit of using the primitive variables to conduct the limiting is highlighted. To highlight the study, different results that are constructing by modifying the subcell tolerance parameter α are compared. Observe that selecting a relatively large value of α combined with primitive variables is the best choice, given that the conserved variables tend to introduce additional oscillations post shock in smooth regimes.

mx	No Limiter	Order	$\alpha = 0$	Order	$\alpha = 0.5h^{1.5}$	Order	$\alpha = 0.75h^{1.5}$	Order
5	1.81×10^{-05}	—	4.24×10^{-2}	—	1.07×10^{-02}	—	1.58×10^{-03}	—
7	3.39×10^{-06}	4.97	3.12×10^{-2}	0.92	6.07×10^{-03}	1.68	1.10×10^{-04}	7.92
10	6.87×10^{-07}	4.48	2.23×10^{-2}	0.94	2.12×10^{-03}	2.95	6.87×10^{-07}	14.23
15	1.33×10^{-07}	4.05	1.75×10^{-2}	0.59	8.56×10^{-05}	7.91	1.33×10^{-07}	4.05
22	2.50×10^{-08}	4.36	1.12×10^{-2}	1.17	2.50×10^{-08}	21.25	2.50×10^{-08}	4.36
33	4.91×10^{-09}	4.02	7.84×10^{-3}	0.88	4.91×10^{-09}	4.02	4.91×10^{-09}	4.02
49	9.52×10^{-10}	4.15	5.11×10^{-3}	1.08	9.52×10^{-10}	4.15	9.52×10^{-10}	4.15

Table 4.2: Convergence for the 2D Euler equation problem from Section 4.7.2. All errors are L_2 norm errors. Observe that with a nonzero value of α, the limited problem eventually matches the unlimited problem's convergence rate, but that without a nonzero value of α, the convergence is first order. One must therefore conclude that nonzero values are required to create a high-order method.

Quadrant	ρ	u_1	u_2	p	ρ	u_1	u_2	p
1	0.5323	1.2060	0.0000	0.3000	2.00	0.75	0.50	1.00
2	0.1380	1.2060	1.2060	0.0290	1.00	−0.75	0.50	1.00
3	0.5323	0.0000	1.2060	0.3000	3.00	−0.75	−0.50	1.00
4	1.5000	0.0000	0.0000	1.5000	1.00	0.75	−0.50	0.00

Table 4.3: Initial conditions for **RP1** (columns 2–5) and **RP3** (columns 6–9).

4.8 Additional numerical results

In this section the numerical scheme is applied to several additional standard numerical test cases. Both Cartesian and unstructured grid examples are included. These benchmark test cases are used to verify the accuracy and robustness of the proposed method.

4.8.1 2D Riemann problems

First a pair of multidimensional Riemann problem test cases are considered. These problems have been used extensively as benchmark test cases for other limiters [69, 43]. The first example, **RP1**, is a well known example that can be found in [110] and [78]. The domain $[-0.5, 0.5] \times [-0.5, 0.5]$ is divided into four rectangular quadrants, all of which meet at the point $(0.3, 0.3)$. The quadrants are numbered starting in the upper left hand corner, and continue in a counterclockwise manner. For example, the upper left hand corner is Quadrant 1, and the upper right hand corner is Quadrant 4. The initial conditions for **RP1** are defined in Table 4.3.

The second two-dimensional Riemann problem considered is called **RP3** [69, 43]. The domain $[-0.5, 0.5] \times [-0.5, 0.5]$ is divided into four equal area quadrants, and the numbering of the quadrants is also defined in a counterclockwise fashion starting at the upper left hand corner. The initial conditions for **RP3** are defined in Table 4.3.

Both of these examples are run on a grid with 400×400 elements and use third-order $\mathbb{P}_2$ elements. Time-stepping is performed with the classical third-order SSPRK3 method using a CFL number of 0.1. Both examples are run using a Rusanov (local Lax Friedrichs (LLF)) flux [106]. Results are plotted in Figure 4.7, which are schlieren plots that show the magnitude of the gradient of the solution density $|\nabla\rho|$. Two figures are shown, one with one point per cell and the other with three points per direction per cell. This comparison is made because the limiter permits tiny oscillations to remain in the solution, since it ignores all variations smaller than a certain threshold (controlled by the parameter α). As the mesh is refined, α also shrinks, and therefore the oscillations also vanish. Note that these oscillations are not visible in the plots showing one point per cell.

4.8.2 Double Mach reflection

For the next two-dimensional example consider the classical double Mach reflection test problem defined by Woodward and Colella [136]. In this test problem a Mach 10 shock approaches a wedge with an angle of 30 degrees. In order to simplify the implementation of this problem on a Cartesian grid, it is typical to rotate the whole problem so that the wedge conforms to the bottom boundary. The problem is run on the domain $[0, 3.2] \times [0, 1]$. For the bottom boundary the exact solution is imposed for $x < \frac{1}{6}$ and solid wall boundary conditions are enforced for $x \geq \frac{1}{6}$. The initial conditions are

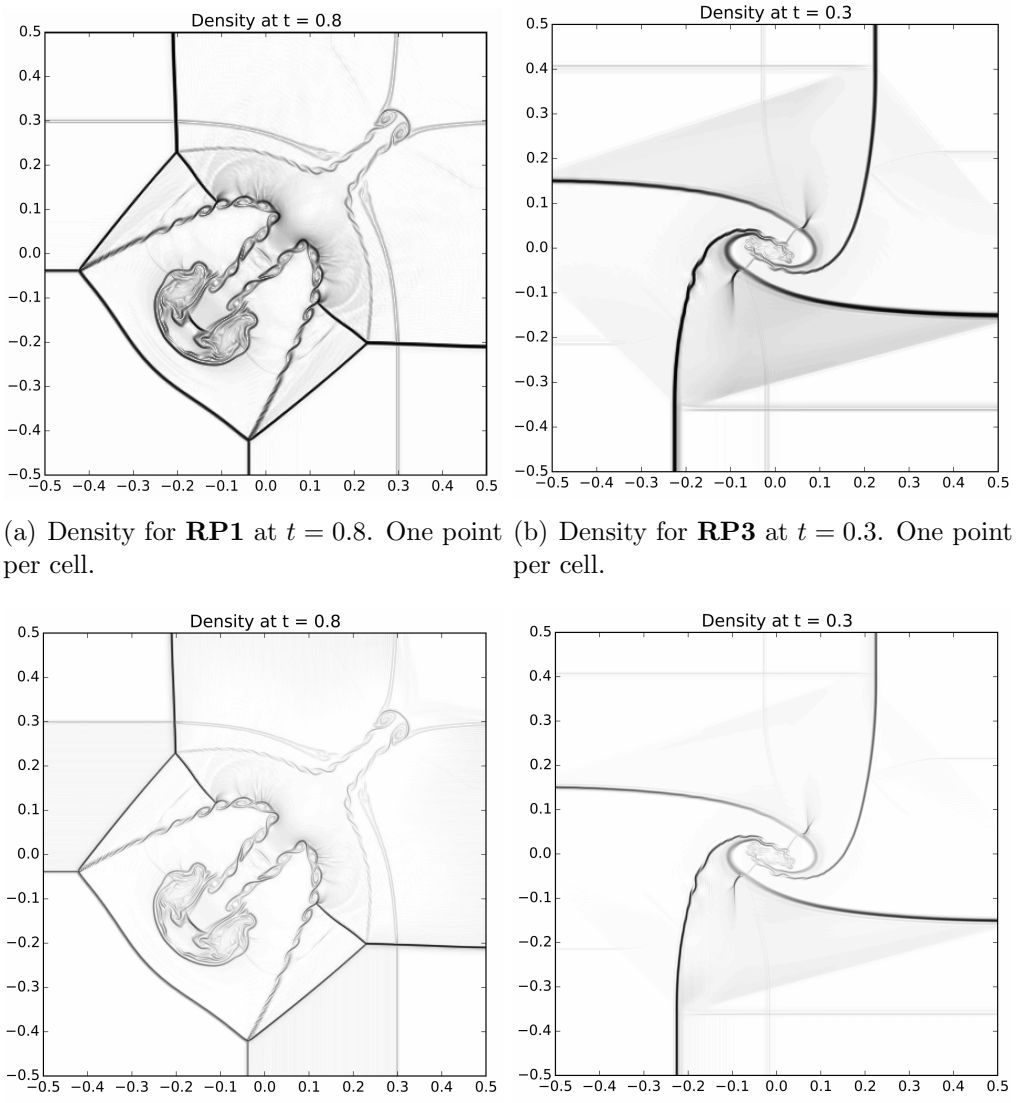
$$(\rho, u_1, u_2, p) = \begin{cases} \left(8.0, 8.25\cos\left(\frac{\pi}{6}\right), 8.25\sin\left(\frac{\pi}{6}\right), 116.5\right) & \text{if} \quad x < \frac{1}{6}, \\ (1.4, 0, 0, 1) & \text{otherwise.} \end{cases}$$

In the Cartesian case this test is run using a 960×240 cell grid matching the second highest resolution in [102]. In the unstructured grid case an example with 204,479 cells is used. Both examples are run with the 10-stage SSPRK4 method with a CFL constant of 0.08 and using $\mathbb{P}_3$ elements so that one should expect 4th order accuracy on sufficiently smooth problems. This smaller CFL number is chosen to guarantee positivity of the solution.

This problem is challenging given that there is a complicated strong-shock structure, as well as a region where there is vortical flow. It can be seen in Figures 4.8 and 4.9 that both limiters qualitatively capture this behavior. The solution on the unstructured grid appears to be much nicer than the Cartesian grid example; this could be due to the fact that the unstructured grid example makes use of all cells touching a current cell, and on the Cartesian grid the limiter only uses cells sharing an edge. The reason for this choice is because ideally one does not want to introduce any extra communication in the limiter (by communicating

across corners). However in the unstructured grid case simply limiting the solution using cells that share common edges introduces additional numerical diffusion, and therefore the trade-off is that less limiting permits more subcell structures to form.

In Figures 4.10 and 4.11 one sees a close of up the solution in the vortical flow region.



(a) Density for **RP1** at $t = 0.8$. One point per cell.

(b) Density for **RP3** at $t = 0.3$. One point per cell.

(c) Density for **RP1** at $t = 0.8$. Nine points per cell.

(d) Density for **RP3** at $t = 0.3$. Nine points per cell.

Figure 4.7: 2D Riemann problem schlieren plots. Here, plots are compared with one and nine points per element in order to enhance the sub-cell resolution of the DG solver.

Both solutions are certainly forming vortices but the unstructured grid again seems to give much better behavior. The interested reader is referred to [102] for solutions with comparable resolutions.

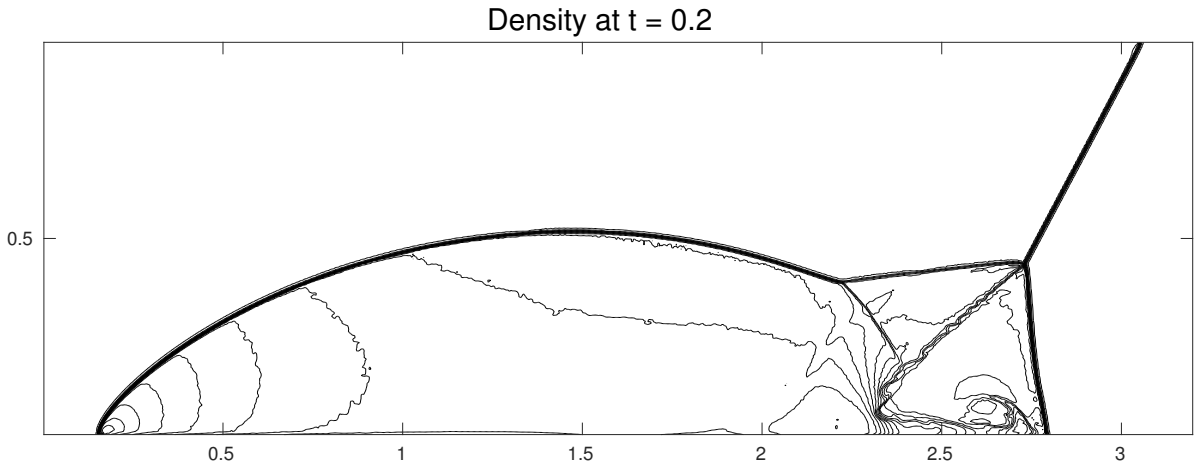
4.8.3 Mach 3 wind tunnel with a step on Cartesian and unstructured meshes

Next consider a classical test problem [136, 60] that is often overlooked, likely due to its difficulty. The problem involves an ideal gas with $\gamma = 1.4$ and a strong shock moving at Mach 3 that passes past a forward facing step. The initial conditions for this problem are given by $(\rho, u_1, u_2, p) = (1.0, 3.0, 0.0, \gamma^{-1})$. The computational domain is the rectangle $[0, 3] \times [0, 1]$ with the region $[0.6, 3.0] \times [0.0, 0.2]$ cut out. In the Cartesian grid case, we use a uniform grid with 480×160 elements, and we use a total of 90,342 cells for the unstructured grid case. The Cartesian grid matches the moderately refined implementation in [34], however we run with the higher-order $\mathbb{P}_3$ elements instead of $\mathbb{P}_2$ elements used in [34]. Both examples are run with SSPRK4 using the Rusanov flux and a CFL of 0.4 (this means our solution is not guaranteed to be positive as for this large of a time-step the SSPRK method does not guarantee a positive cell average, but that doesn't appear to have been a problem for this example).

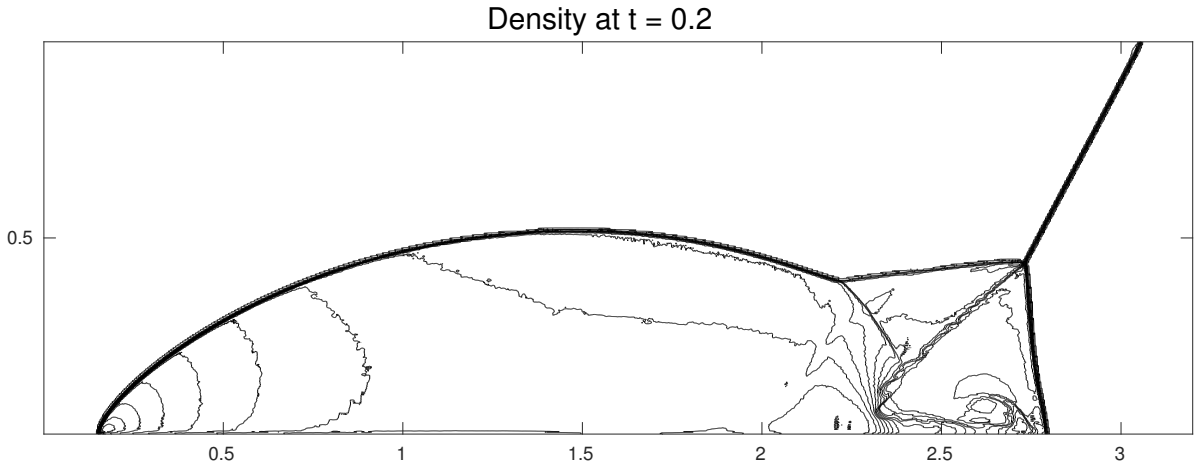
This is a complicated problem featuring a long contact discontinuity. Similar to the previous problems, high-resolution schemes are able to capture the formation of vortices that should exist. An additional and well-known difficult aspect of this problem is that the solution tends to form a spurious entropy layer above the step [136]. This results in the creation of a stem-like structure where there would normally be a shock reflecting off of the step, which is especially noticeable on many solvers that operate on Cartesian grids. Because of these difficulties, authors often introduce a so-called entropy "fix" for their solver. We propose something much simpler: To mitigate this effect, we use a slightly more aggressive implementation of our limiter near the top of the step. In place of using point-wise values of the primitive variables to estimate local upper and lower bounds, we use cell averages of the conserved variables in this edge region only. That is, we replace M_i^ℓ and m_i^ℓ in equations (4.15) and (4.16) by

$$M_i := \max\left\{\bar{q}_i + \alpha(h),\ \max_{j\in N_{\mathcal{T}_i}}\left\{\bar{q}^h_{M_j}\right\}\right\} \ \text{ and } \ m_i := \min\left\{\bar{q}_i - \alpha(h),\ \min_{j\in N_{\mathcal{T}_i}}\left\{\bar{q}^h_{m_j}\right\}\right\},$$

where the ℓ superscript has been suppressed. Note that this means that in this special region we are not using the primitive variables for determining θ_i, but the conserved variables; furthermore, we are using the average values $\bar{q}$ rather than point values at the points defined by χ_i. Additionally we use a much smaller α value in this region to obtain better results. In the majority of the domain we take $\alpha = 500h^{1.5}$, but in the small region above the step we take $\alpha = 100h^{1.5}$ on the structured grid and $\alpha = 20h^{1.5}$ on the unstructured grid. Note that this problem has required a modified treatment in other works as well because the corner is

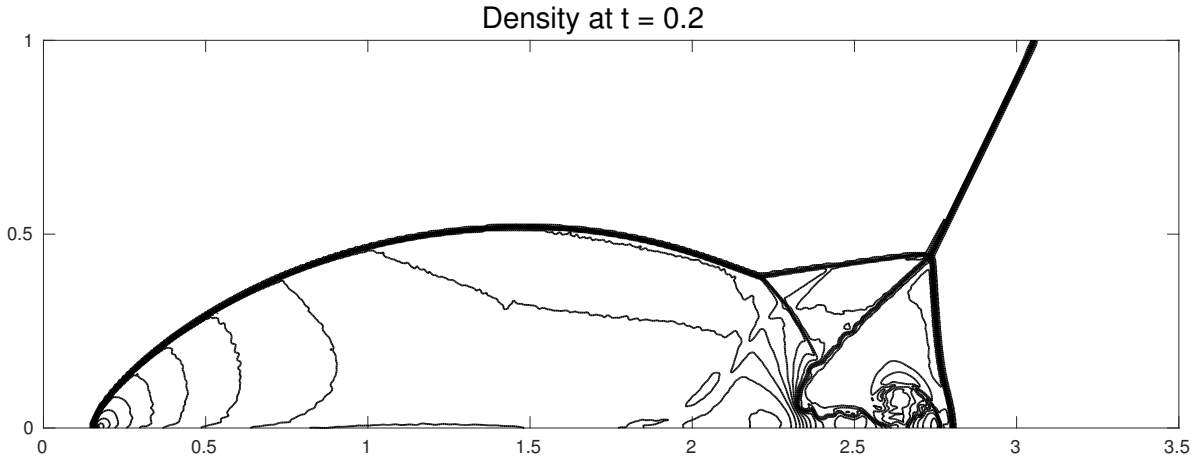


(a) One point plotted per cell.

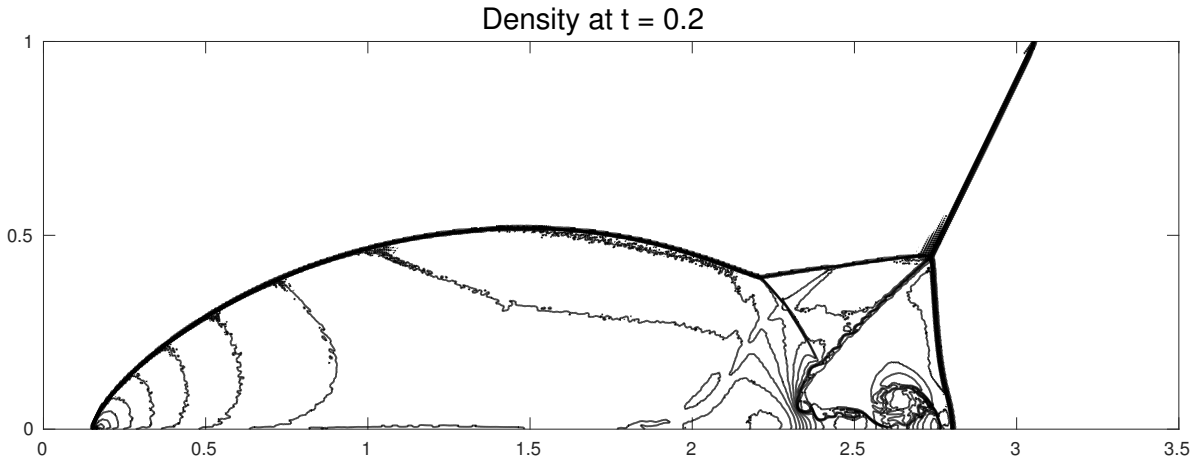


(b) Sixteen points plotted per cell.

Figure 4.8: Double Mach reflection on a structured grid. For this example $\alpha = 1000h^{1.5}$. A total of 30 equispaced contours are plotted ranging from 1.5 to 22.7.



(a) One point plotted per cell.



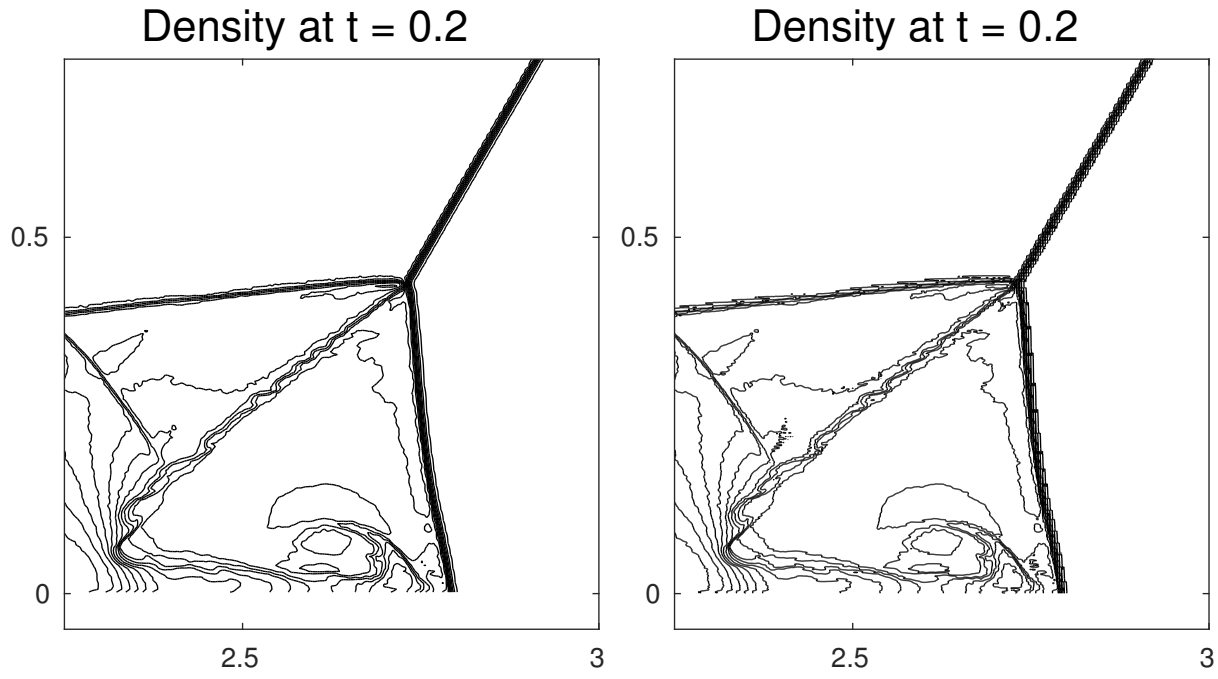
(b) Sixteen points plotted per cell.

Figure 4.9: Double Mach reflection on an unstructured grid. The same contours are plotted as in Figure 4.8

a singular point [136]. The region using the more aggressive limiter extends 6 cells above the step in the Cartesian grid case, and until $y = 0.24$ above the step in the unstructured grid case. These values can certainly be modified, but we found that if the region is taken to be only one cell width, then the entropy layer tends to be much more prominent. Moreover, if the more aggressively limited region is too large the shocks in the problem will be smeared out too heavily.

In Figures 4.12, 4.14, 4.13 and 4.15 we observe that both the structured and unstructured grid limiters do an excellent job of capturing the structure of the contact discontinuity. However it appears that the Cartesian grid case suffers from a pronounced spurious entropy

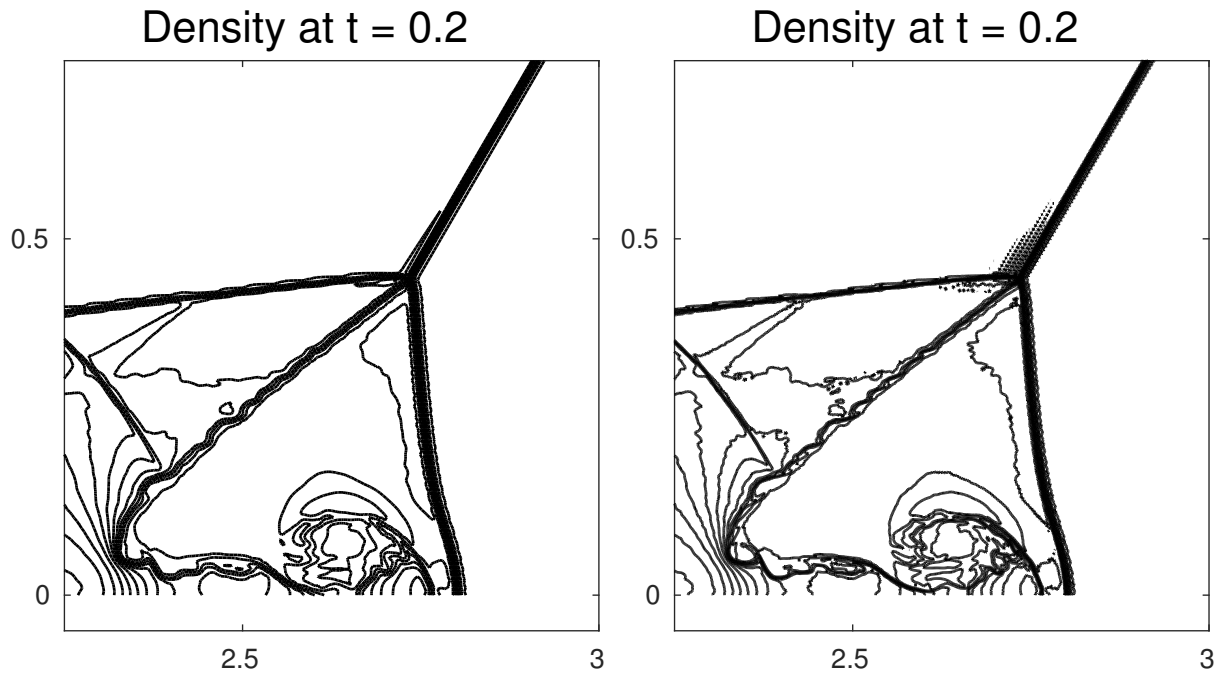
layer; although it should be noted that the entropy layer is not atypically large (e.g., see [34, 149]). Given the relative simplicity of this limiter the results are promising.



(a) One point plotted per cell.

(b) Sixteen points plotted per cell.

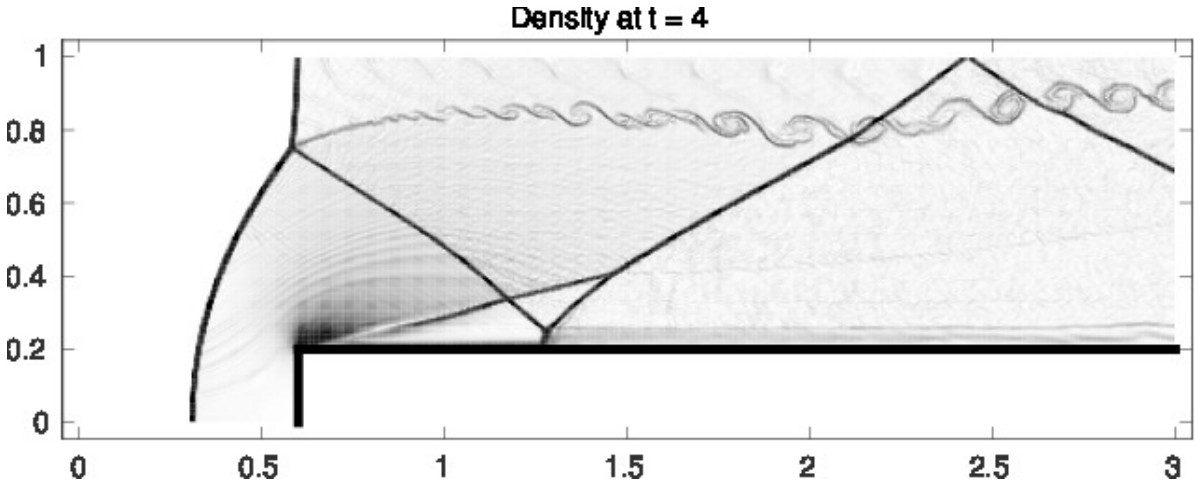
Figure 4.10: The output from Figure 4.8 zoomed in to show the resolution near the contact discontinuity. This contact discontinuity displays the roll up behavior that is expected in high resolution numerical simulations. These simulations use the $\mathbb{P}_3$ basis on a structured grid.



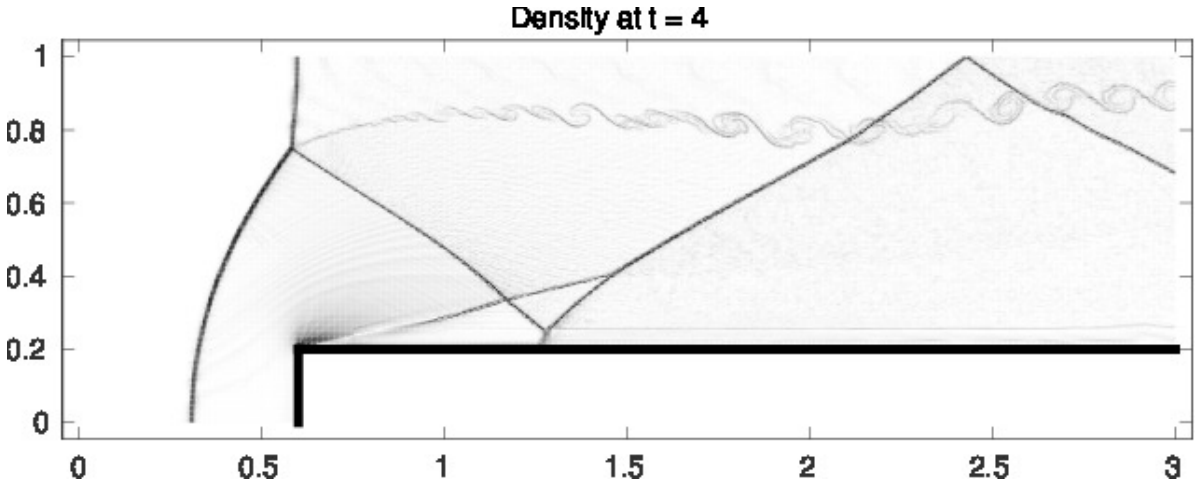
(a) One point plotted per cell.

(b) Sixteen points plotted per cell.

Figure 4.11: The output from Figure 4.9 zoomed in to show the resolution near the contact discontinuity. These simulations use the $\mathbb{P}_3$ basis on an unstructured grid.

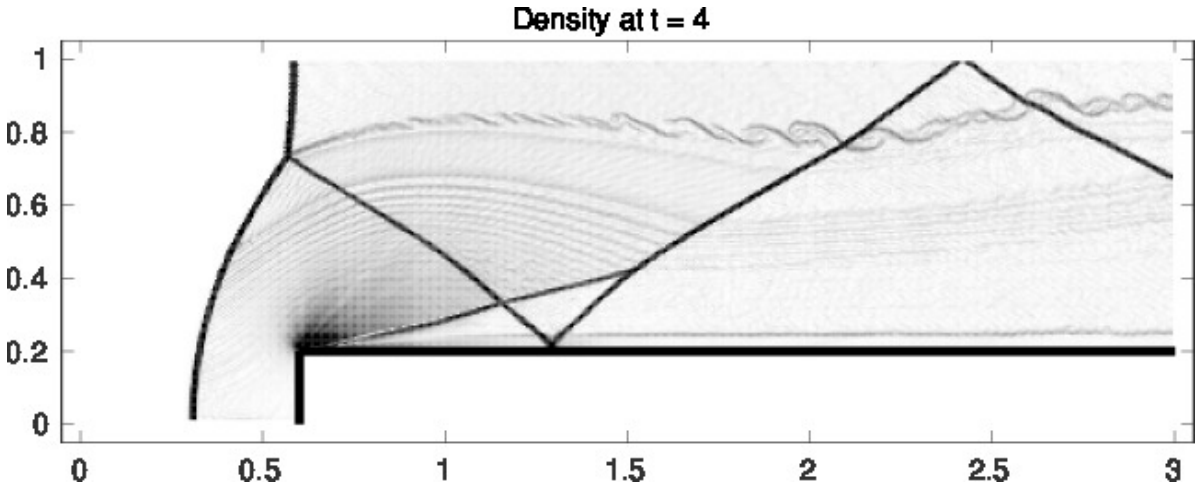


(a) One point plotted per cell.

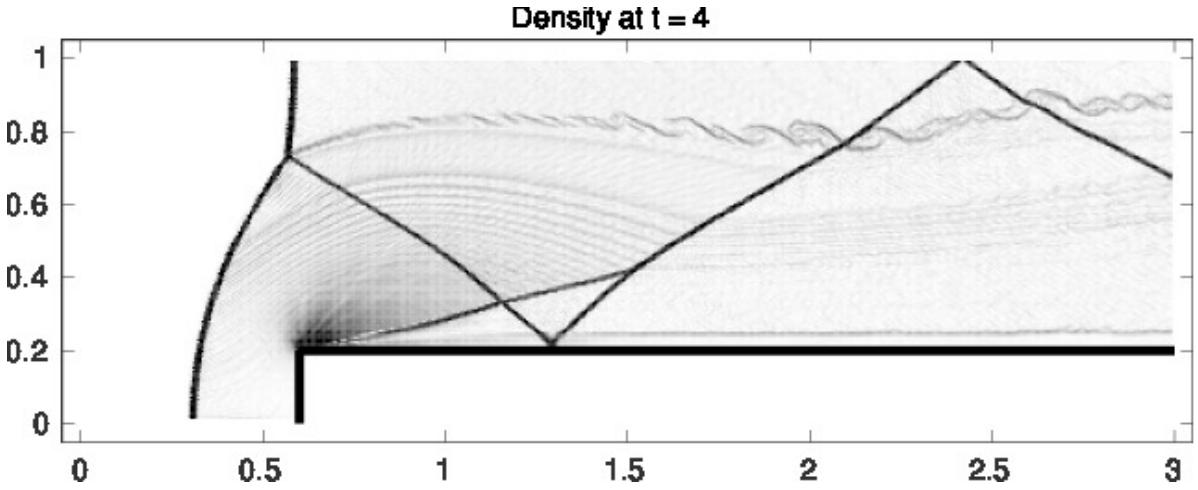


(b) Sixteen points plotted per cell.

Figure 4.12: Density schlieren plots from the forward facing step problem run on Cartesian grid. These simulations use a grid of size $h = \frac{1}{160}$.

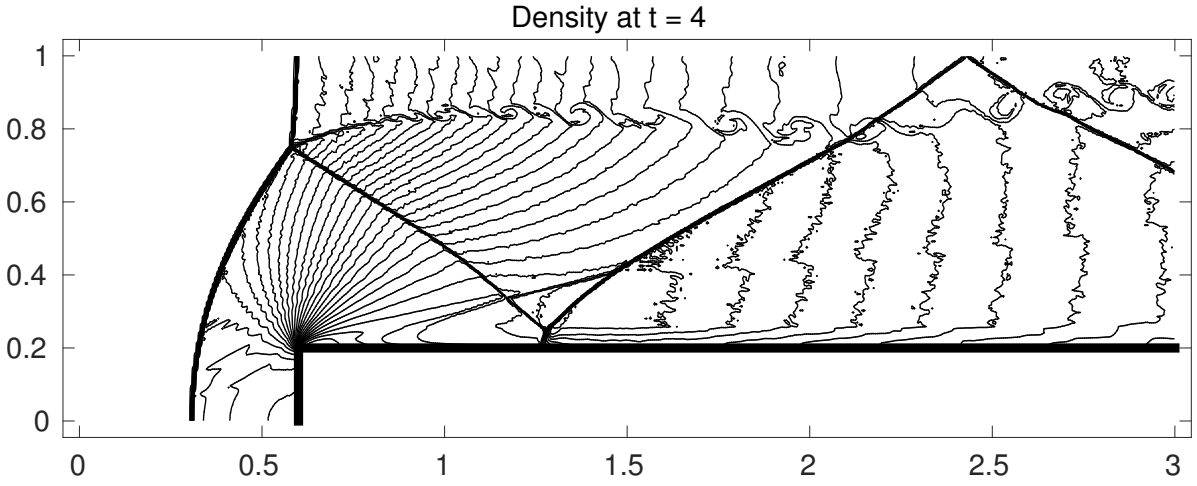


(a) One point plotted per cell.

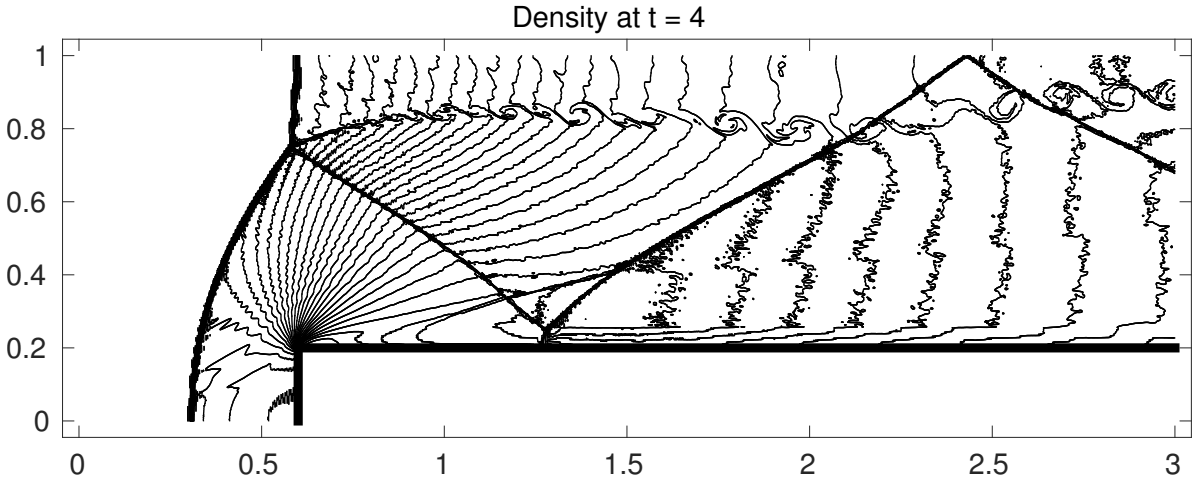


(b) Sixteen points plotted per cell.

Figure 4.13: Density schlieren plots from the forward facing step problem with an an unstructured grid. Here we use 90,342 cells to discretize the domain.

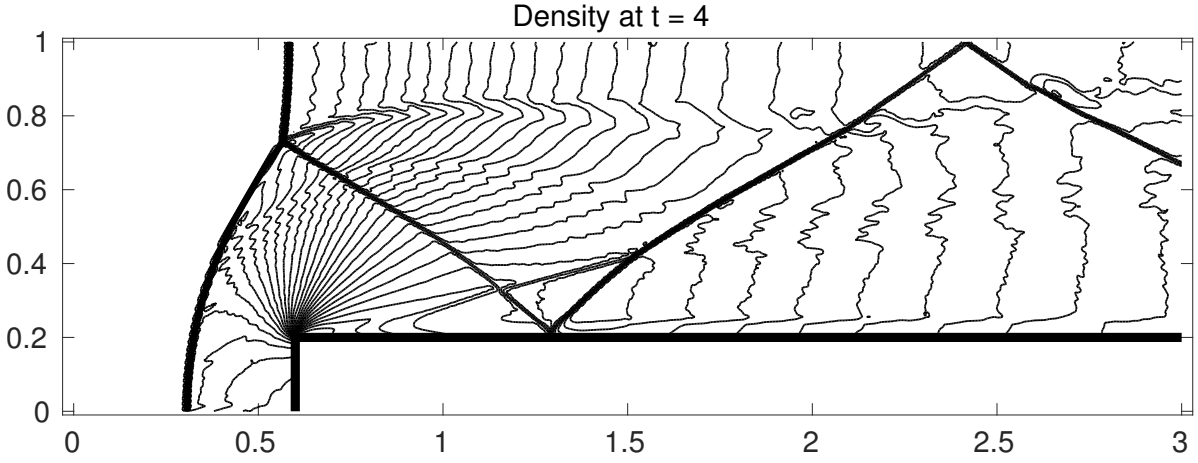


(a) One point plotted per cell.

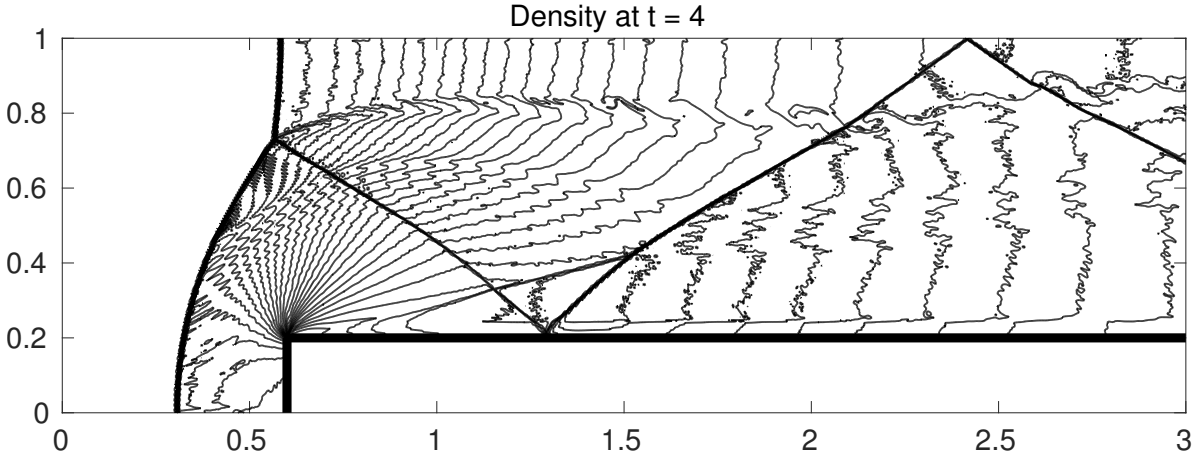


(b) Sixteen points plotted per cell.

Figure 4.14: Density contour plots from the forward facing step problem run on a Cartesian grid. There are 30 contours spaced evenly from 0.1 to 4.54.



(a) One point plotted per cell.



(b) Sixteen points plotted per cell.

Figure 4.15: Density contour plots from the forward facing step problem run on an unstructured grid. The same contours are plotted as in Figure 4.14.).

4.9 Limiting using Characteristic variables

4.9.1 One dimension

In one dimension it is possible to modify the limiter defined in Section 4.5 to use characteristic variables. We will define the characteristic variables of a system to be $w(q)$. In practice we will actually use the characteristic variables defined with respect to cell averages, so $w(q)$ will not be the exact characteristic variables at any give point. Also note that when computing bounds for limiting cell i then the mapping $w(q)$ will need to be defined with respect to cell i even when it is used on neighboring cells to define bounds.

The method can be summarized as follows:

Step 0. Select a set of points χ_i that will be used to approximate cell maximum and minimum values. In this work we always select: corners, internal, and edge Gaussian quadrature points.

Step 1. For each element i and each component ℓ of w compute:

$$w^\ell_{M_i} := \max_{x\in\chi_i}\left\{w^\ell(q^h(x))\Big|_{\mathcal{T}_i}\right\} \quad \text{and} \quad w^\ell_{m_i} := \min_{x\in\chi_i}\left\{w^\ell(q^h(x))\Big|_{\mathcal{T}_i}\right\}. \tag{4.21}$$

Step 2. For each element i and each component ℓ of w compute an approximate upper and lower bound over the set of neighbors, $N_{\mathcal{T}_i}$ (excluding the current cell $\mathcal{T}_i$):

$$M^\ell_i := \max\left\{\bar{w}^\ell_i + \alpha(h),\ \max_{j\in N_{\mathcal{T}_i}}\left\{w^\ell_{M_j}\right\}\right\}, \tag{4.22}$$

$$m^\ell_i := \min\left\{\bar{w}^\ell_i - \alpha(h),\ \min_{j\in N_{\mathcal{T}_i}}\left\{w^\ell_{m_j}\right\}\right\}, \tag{4.23}$$

where $\bar{w}^\ell_i$ is the cell average of w^ℓ over cell $\mathcal{T}_i$.

Step 3. For each element i compute:

$$\theta_{M_i} := \min_\ell\left\{\phi\left(\frac{M^\ell_i - \bar{w}^\ell_i}{w^\ell_{M_i} - \bar{w}^\ell_i}\right)\right\} \text{ and } \theta_{m_i} := \min_\ell\left\{\phi\left(\frac{m^\ell_i - \bar{w}^\ell_i}{w^\ell_{m_i} - \bar{w}^\ell_i}\right)\right\}. \tag{4.24}$$

Step 4. For each element i compute:

$$\theta_i := \min\left\{1,\ \theta_{m_i},\ \theta_{M_i}\right\}. \tag{4.25}$$

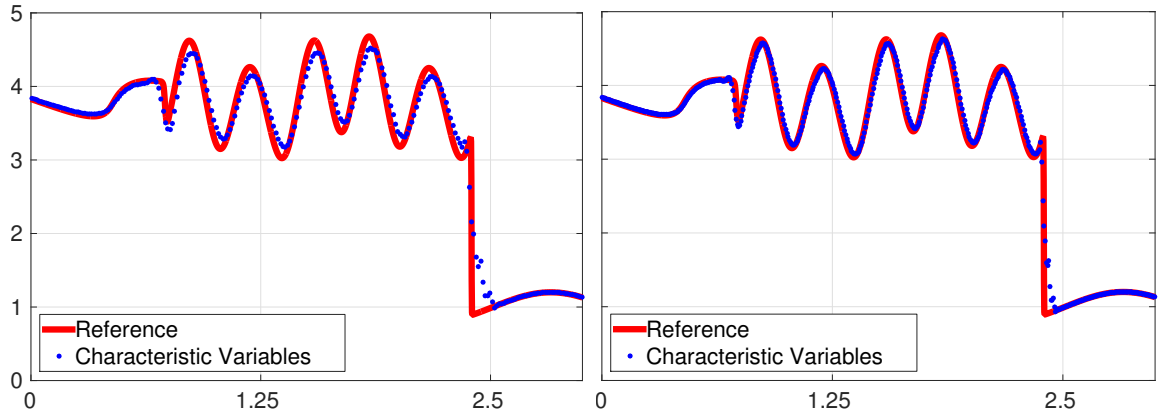
Step 5. For each element i and each component ℓ of q compute

$$\tilde{q}^h(x)\Big|_{\mathcal{T}_i} := \bar{q}_i + \theta_i\left(q^h(x)\Big|_{\mathcal{T}_i} - \bar{q}_i\right). \tag{4.26}$$

Note that we are, as with the primitive variables, using the values of the variable w to determine the amount of limiting that needs to be done (i.e., size of θ_i), but we are actually applying the limiter to the conserved variables, thereby retaining mass conservation at the discrete level.

An Euler equation example-The Shu-Osher problem

Recall the example introduced in Section 4.7.1. We run that same example with characteristic variable limiting. Again we use 4th order DG with SSPRK4. Compare the results in Figure 4.16 to Figure 4.6. The characteristic variable limiter seems to perform somewhat better than the primitive variable limiter but it is hard to tell in the 200 cell case as the solution still appears under-resolved. The 400 cell solution looks very well resolved.



(a) 200 cells (b) 400 cells

Figure 4.16: The Shu-Osher problem solution using characteristic variable limiting. Both problems were run with $\alpha = 1000h^{1.5}$.

4.9.2 Two dimensions

In two dimensions limiting using characteristic variables is more difficult but an idea becomes clear if you consider that a limiting method cannot work unless it is an effective shock detector. In that case, then, the primary goal of using the characteristic variables should be to identify when a jump (which will result in oscillations) is a real shock or contact discontinuity, and when the jump can be made to not look like a jump by transforming variables. One way to accomplish this would be to compute two θ values for each cell i, θ_{ix} and θ_{iy}. The value θ_{ix} would be computed using the characteristic variables in the x direction and θ_{iy} would be computed using the characteristic variables in the y direction. Then we

can take the maximum of those two θ values. This would minimize the limiting that only appears necessary because we are looking in the wrong set of variables. Recall a basic $2d$ hyperbolic PDE

$$q_t + f(q)_x + g(q)_y = 0. \tag{4.27}$$

We will define w_x to be the characteristic variables associated with $f^{'}(q)$ and w_y to be the characteristic variables associated with $g^{'}(q)$.

The method can be summarized as follows:

Step 0. Select a set of points χ_i that will be used to approximate cell maximum and minimum values. In this work we always select: corners, internal, and edge Gaussian quadrature points.

Step 1. For each element i, each direction d (x or y) and each component ℓ of w compute:

$$w^{\ell}_{dM_i} := \max_{x\in\chi_i}\left\{ w^{\ell}_d(q^h(x))\Big|_{\mathcal{T}_i}\right\} \quad \text{and} \quad w^{\ell}_{dm_i} := \min_{x\in\chi_i}\left\{ w^{\ell}_d(q^h(x))\Big|_{\mathcal{T}_i}\right\}. \tag{4.28}$$

Step 2. For each element i, each direction d and each component ℓ of w_d compute an approximate upper and lower bound over the set of neighbors, $N_{\mathcal{T}_i}$ (excluding the current cell $\mathcal{T}_i$)

$$M^{\ell}_{id} := \max\left\{ \bar{w}^{\ell}_{id} + \alpha(h),\ \max_{j\in N_{\mathcal{T}_i}}\left\{ w^{\ell}_{dM_j}\right\}\right\}, \tag{4.29}$$

$$m^{\ell}_{id} := \min\left\{ \bar{w}^{\ell}_{id} - \alpha(h),\ \min_{j\in N_{\mathcal{T}_i}}\left\{ w^{\ell}_{dm_j}\right\}\right\}, \tag{4.30}$$

where $\bar{w}^{\ell}_{id}$ is the cell average of w^{ℓ}_d over cell $\mathcal{T}_i$.

Step 3. For each element i and direction d compute

$$\theta^d_{M_i} := \min_{\ell}\left\{ \phi\left(\frac{M^{\ell}_{id} - \bar{w}^{\ell}_{id}}{w^{\ell}_{dM_i} - \bar{w}^{\ell}_{id}}\right)\right\} \ \text{ and } \ \theta^d_{m_i} := \min_{\ell}\left\{ \phi\left(\frac{m^{\ell}_{id} - \bar{w}^{\ell}_{id}}{w^{\ell}_{dm_i} - \bar{w}^{\ell}_{id}}\right)\right\}. \tag{4.31}$$

Step 4. For each element i and direction d compute:

$$\theta_{id} := \min\left\{1,\ \theta^d_{m_i},\ \theta^d_{M_i}\right\}. \tag{4.32}$$

Step 4. For each element i compute:

$$\theta_i := \max\left\{\theta_{ix}, \theta_{iy}\right\}. \tag{4.33}$$

Step 5. For each element i and each component ℓ of q compute

$$\tilde{q}^h(x)\Big|_{\mathcal{T}_i} := \bar{q}_i + \theta_i\left(q^h(x)\Big|_{\mathcal{T}_i} - \bar{q}_i\right). \tag{4.34}$$

Acoustics radial wave packet

In this section we run a simple acoustics equation example

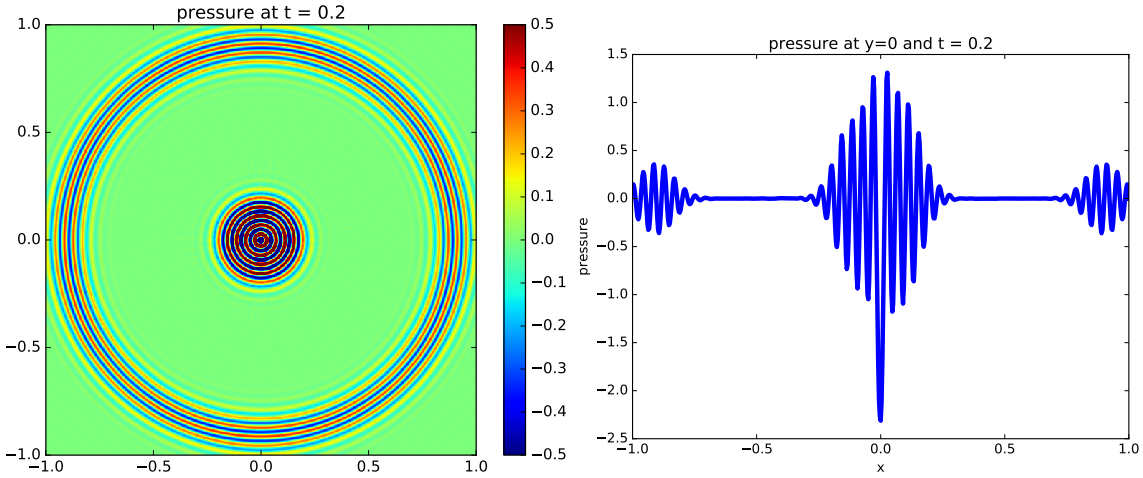
$$\begin{bmatrix} p \\ u \\ v \end{bmatrix}_t + \begin{bmatrix} 0 & K_0 & 0 \\ \frac{1}{\rho_0} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p \\ u \\ v \end{bmatrix}_x + \begin{bmatrix} 0 & 0 & K_0 \\ 0 & 0 & 0 \\ \frac{1}{\rho_0} & 0 & 0 \end{bmatrix} \begin{bmatrix} p \\ u \\ v \end{bmatrix}_y = 0 \tag{4.35}$$

where

$$K_0 = 4,\ \rho_0 = 1.0 \text{ and } c_0 = \sqrt{\frac{K_0}{\rho_0}} = 2.$$

The problem will have the initial condition $u(x, y, 0) = v(x, y, 0) = 0.0$ and $p(x, y, 0) = e^{-100(r-0.5)^2} \sin(150r)$ on a domain with outflow boundary conditions.

The problem starts out with a highly oscillatory ring of pressure centered at radius $r = 0.5$. The ring splits into an inward and outward going ring. At time $t = 0.2$ the inward going ring is colliding on itself to create a negative pressure perturbation at center through constructive interference. The characteristic based limiting scheme handles this fairly well, but limiting on the conserved variables resulted in harsh truncation of the extremum at the point $x = 0$.



(a) Pressure

(b) A 1-d slice of the pressure

Figure 4.17: Pressure given a radially symmetric initial condition in a radially symmetric acoustic medium using characteristic variable limiting. This was done on a 150×150 mesh using a fourth order DG scheme.

4.10 Limiting on mapped grids

The same basic limiting scheme described in the rest of this chapter can be applied to mapped grids with some minor changes. The changes are due to the fact that the hierarchical Legendre modes used on reference elements are no longer orthonormal when combined with a bilinear (or trilinear in 3D) mapping function. The major problem this causes is that it is a bit more work to force the limiter to maintain conservation.

When working on mapped cells it is easier to consider a nodal Lagrange polynomial basis as outlined in Section 3. Assuming we use the tensor product of the 1 dimensional Lagrange polynomials associated with the Gaussian quadrature points as the basis, these sets of nodal points will also provide a Quadrature rule that is valid even with the Jacobian term included in the integral. Say that our solution integrates to V so that

$$\int_{\mathcal{T}_m} q^h \, |J^m| d\mathbf{x} = V.$$

In that case then if we define $\zeta_i \in Z^2_{n+1}$ as the set of tensor product Gaussian quadrature nodes with associated weights ω_i, then $\sum_{\zeta_i \in Z^2_{n+1}} q^h_i |J^m|\Big|_{(z_i)} \omega_i = V$. Then, at every point we can define $\hat{q}^h_i = q^h_i - V$ such that $\sum_{\zeta_i \in Z^2_{n+1}} \hat{q}^h_i |J^m|\Big|_{(z_i)} \omega_i = 0$.

We can then introduce a θ parameter that takes the initial value $\theta = 1$ so our solution at point z_i will be $q^h_i = V + \theta \hat{q}^h_i$. We can then set this θ parameter as we would in the limiting procedure introduced in Section 4.2. Following this we reconstruct the limited solution at each interpolation point i, $\tilde{q}^h_i$ as

$$\tilde{q}^h(z_i) := V + \theta \hat{q}^h(z_i). \tag{4.36}$$

The values $\tilde{q}^h(z_i)$ can then be interpolated to create the new limited solution. This is very similar to the limiter defined earlier in the chapter except that now we cannot work directly with the constant mode and assume the other modes do not contribute to the cell average.

This is conservative because

$$\begin{aligned}\sum_{\zeta_i \in Z^2_{n+1}} q^h_i |J^m|\Big|_{(z_i)} \omega_i &= V \sum_{\zeta_i \in Z^2_{n+1}} |J^m|\Big|_{(z_i)} \omega_i + \theta \sum_{\zeta_i \in Z^2_{n+1}} \hat{q}^h_i |J^m|\Big|_{(z_i)} \omega_i, \\ &= V + \theta \sum_{\zeta_i \in Z^2_{n+1}} \hat{q}^h_i |J^m|\Big|_{(z_i)} = V.\end{aligned}$$

Note that no other limiter that the author is aware of can be so easily applied to problems on mapped grids. On these problems one cannot depend on a hierarchical structure of modes or consistent cell orientations which eliminates most other limiting schemes.

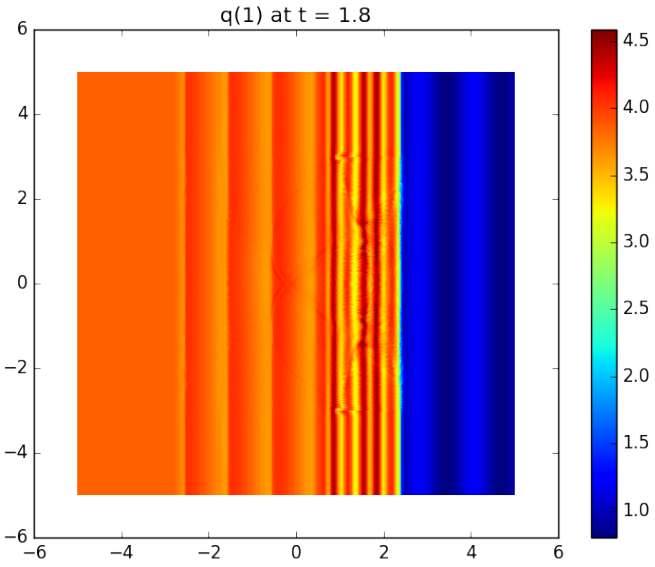


Figure 4.18: The density found for the Shu-Osher problem simulated on a mapped grid in Section 4.10. The limiter introduced in Section 4.10 was used.

Shock-entropy wave problem

In this section we test out the mapped-grid version of the limiter on the shock-entropy problem introduced in Section 3.4.3. As in the positivity limited only version of this problem, we see mesh effects in the solution. It is not surprising that the limiter can not fix this issue. However we see in Figure 4.19 that the limiter has reduced the amount of spurious oscillations in the slice plot while degrading the resolution only minimally. This limiter seems promising.

4.11 Summary of the results of this chapter

In this section the results presented in this chapter will be summarized. First a novel limiter was introduced for the discontinuous Galerkin method. The proposed limiter can be thought of as either an extension of the Barth-Jespersen [8] limiter to the discontinuous Galerkin framework, or as an extension of the Zhang and Shu [143] positivity preserving limiter to be able to accommodate shocks and not just satisfy the positivity principle. The limiter is uniquely flexible because it does not use much geometric data or information about the modal decomposition of the solution. The limiter has been developed and tested on structured, unstructured and mapped grids. Numerical results have been presented demonstrating that this limiter is effective in each of these contexts. The limiter was developed in Sections 4.5

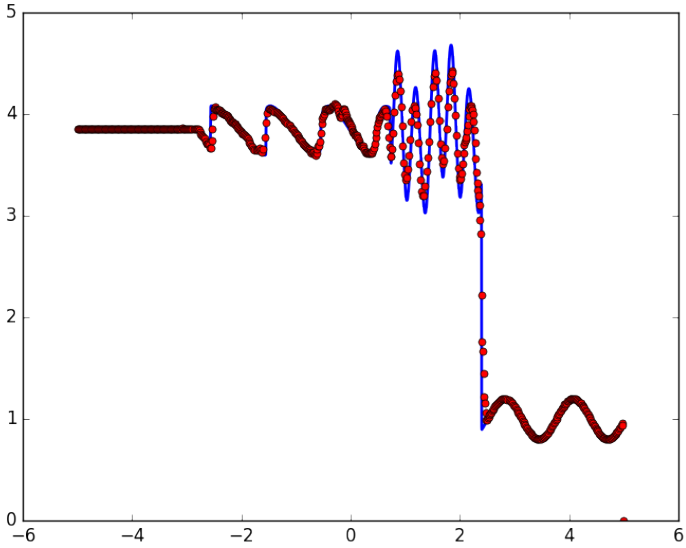


Figure 4.19: A slice plot of the density found for the Shu-Osher problem simulated on a mapped grid in Section 4.10. The slice is taken at time $t = 1.8$. The limiter introduced in Section 4.10 was used.

and 4.2 for scalar equations and systems respectively. In Section 4.9 a modification of the original limiter was introduced that allows limiting using the characteristic variables. This modification was tested on a structured grid example where limiting using the conserved variables fails. In Section 4.10 a modification of the original limiter was introduced that accounts for the fact that it is more difficult to guarantee conservation on mapped grids. This modification was tested on a standard Euler equation test case.

Chapter 5

THE DIFFERENTIAL TRANSFORM METHOD

5.1 The bigger picture

The remainder of this work will involve Lax-Wendroff time-stepping schemes. This chapter introduces the differential transform method as a way to compute a space-time Taylor series of the solution and flux function. This is one way to compute the derivatives necessary in order to implement Lax-Wendroff time-stepping. The subsequent chapters will develop a Lax-DG scheme, complete with the positivity limiter (Chapter 7) necessary to solve many physical problems (CFD or plasma physics for example). Additionally that Lax-DG scheme will be implemented (in particular the DT-DG scheme developed in Chapter 8) with adaptive mesh refinement (introduced in Section 8.2). The fact that this scheme is a single-stage single step scheme that directly constructs a space-time Taylor series of the solution will enable the development of Local time stepping to be used with the AMR.

5.2 Motivation for using differential transforms

The following chapter is based on currently unpublished work done with David Seal aimed at developing a discontinuous Galerkin scheme using the differential transform method for time-stepping, so as to create a method with a spatially compact single-step stencil.

For a hyperbolic conservation law

$$q_t + \nabla_{\mathbf{x}} \cdot \mathbf{F}(q) = 0, \quad q(0, \mathbf{x}) = q_0(\mathbf{x}), \quad \mathbf{x} \in \Omega \subseteq \mathbb{R}^d, \tag{5.1}$$

a method that uses Lax-Wendroff time stepping updates the solution by way of discretizing its Taylor expansion (in time):

$$q(\mathbf{x}, t) = q^n + (t - t^n) q_t^n + \frac{(t - t^n)^2}{2!} q_{tt}^n + \cdots . \tag{5.2}$$

The basic strategy in a Lax-Wendroff procedure is to truncate this series at a desired order of accuracy, and then replace each time derivative with spatial derivatives through the Cauchy-Kowalevski procedure. Each higher derivative follows from the chain rule and appealing to the original PDE in order to convert temporal time derivatives into spatial derivatives. For example, the first two time derivatives are

$$\begin{aligned} q_t &= -\nabla_{\mathbf{x}} \cdot \mathbf{F}(q), \\ q_{tt} &= -\nabla_{\mathbf{x}} \cdot \mathbf{F}(q)_t = -\nabla_{\mathbf{x}} \cdot (\mathbf{F}'(q) q_t) = \nabla_{\mathbf{x}} \cdot (\mathbf{F}'(q) \cdot \nabla_{\mathbf{x}} \cdot \mathbf{F}(q)) . \end{aligned} \tag{5.3}$$

These terms are required to obtain second order accuracy. These expressions become cumbersome as the number of derivatives increases. For example, second order accuracy requires the computation the Jacobian $\mathbf{F}'(q)$, and this is where the original Lax-Wendroff method stopped [75]. Third order accuracy requires the computation of the Hessian $\mathbf{F}''(q)$ of the flux function, and higher orders of accuracy require the computation of tensors that grow drastically in size. Despite this fact, several numerical methods that use Lax-Wendroff time stepping have been constructed to very high orders of accuracy. Currently, the largest class of solvers that use this procedure are the so-called Arbitrary DERivative (ADER) methods [126, 127, 128, 129, 123, 88], which were originally formulated as a finite volume solver for a generalized Riemann problem [104, 9, 83, 126]. Other numerical methods that use Lax-Wendroff time stepping include discontinuous Galerkin [97, 41, 96, 38, 47], as well as finite difference WENO methods [98, 81, 61].

The differential transform method will allow us to automate the process required to construct high order derivatives of the flux function.

5.2.1 Background

Taylor series numerical methods have a long history, having been used as early on as the 1960's for celestial dynamics problems [111]. For an extensive review of their use in analytical and numerical tools, the interested reader is referred to [14]. However, despite this history they are often overlooked as a means of solving PDEs to very high orders of accuracy.

The modern phrase *differential transform* was originally coined by Pukhov in [95], where he used this newfound toolset to model electrical power circuits [44]. Starting in this time period, they have seen rapid development for symbolic differentiation software [109, 131, 130], and have since been applied to solve ordinary differential equations [24, 2, 45], simple partial differential equations [24, 23, 59, 6, 70, 2, 45], linear fractional PDEs [92, 3, 46], non-linear reaction diffusion equations [122], one-dimensional hyperbolic PDEs [91, 90], as well as multidimensional scalar transport equations [89]. The focus on this chapter will be on non-linear multidimensional hyperbolic PDEs.

Differential transforms (DTs) are proposed to automate the complicated rules that are necessary to construct the higher-order terms required for high-order accuracy using Lax-Wendroff time-stepping. The differential transforms take the place of writing code that calls Jacobians and Hessians.

5.3 Differential transforms for functions of two variables

This section reviews the preliminary tools that are required to construct arbitrary order mixed derivatives of nonlinear flux functions using the differential transform method.

Given a sufficiently smooth function $f(q(x,t))$, the *differential transform* of the function

centered at (x^*, t^*) is defined as

$$F(h,k) := \frac{1}{h!k!}\frac{\partial^h}{\partial x^h}\frac{\partial^k}{\partial t^k} f(x,t)|_{(t^*,x^*)}, \quad h,k \in \mathbb{Z}_{\geq 0}. \tag{5.4}$$

The inverse of this transform is simply a rewriting of the Taylor series expansion

$$f(q(x,t)) = \sum_{h=0}^{\infty}\sum_{k=0}^{\infty} F(h,k)(x-x^*)^h(t-t^*)^k. \tag{5.5}$$

There will be a similar series expansion for $q(x,t)$

$$q(x,t) = \sum_{h=0}^{\infty}\sum_{k=0}^{\infty} Q(h,k)(x-x^*)^h(t-t^*)^k. \tag{5.6}$$

The differential transform is actually just a method to compute a series expansion for f from a series expansion that is already known for q. When solving a PDE one will typically know only a spatial Taylor series approximation of q.

Assuming f is a flux function in a hyperbolic PDE, f and q will satisfy a relation such as

$$q_t = -f(q)_x.$$

This means that

$$\sum_{h=0}^{\infty}\sum_{k=1}^{\infty} kQ(h,k)(x-x^*)^h(t-t^*)^{k-1} = -\sum_{h=1}^{\infty}\sum_{k=0}^{\infty} hF(h,k)(x-x^*)^{h-1}(t-t^*)^k. \tag{5.7}$$

Collecting like terms in Equation (5.7) it can be seen that

$$(k+1)Q(h,k+1) = -(h+1)F(h+1,k). \tag{5.8}$$

Equation (5.8) is the key that will allow one to compute the full differential transform. The method will first proceed by using the differential transform method operating on the coefficients $Q(h,0)$ to compute the $F(h,0)$ coefficients. Next one will use Equation (5.8) to compute the $Q(h,1)$ coefficients. Using these coefficients with the differential transform method, again allows the computation of $F(h,1)$. The coefficients $F(h,1)$ can in turn be combined with Equation (5.8) to compute $Q(h,2)$. This process can be repeated indefinitely. The differential transform method can be summarized as follows:

Step 1. First compute the coefficients $F(h,0)$ from the coefficients $Q(h,0)$ using a transform rule.

Step 2. Then use Equation (5.8) to compute the $Q(h,1)$ coefficients from the $F(h,0)$ coefficients.

Step 3. Then repeat steps 1 and 2, this time starting with the $Q(h,1)$ coefficients.

Note that in practice the user will start with a truncated spatial Taylor series so they will produce an approximate space-time Taylor series such as

$$f(q,x,t) = \sum_{k=0}^{H} \sum_{h=0}^{M} F(h,k)(x-x^*)^h (t-t^*)^k.$$

The truncated Taylor series will be treated as if it is exact so that any uninitialized coefficients involved in any expressions to be evaluated will be treated as if they were 0.

Several well known properties of this transform follow. In each of the following, it will be assumed that u and v are real analytic. Short proofs will be provided for several of these properties because the methodology is necessary to derive more general properties.

1. **Linearity.** If $f(x,t) = u(x,t) + cv(x,t)$, where $c \in \mathbb{R}$ is a constant, then $F(h,k) = U(h,k) + cV(h,k)$.

   Proof. This follows immediately from the fact that derivatives are linear operators. □

2. **Recursive definition of derivatives.** If $\frac{\partial f}{\partial t}(x,t) = \frac{\partial u}{\partial x}(x,t)$, then $F(h,k+1) = \frac{h+1}{k+1} U(h+1,k)$.

   Proof. Start with $\frac{\partial f}{\partial t}(x,t) = \frac{\partial u}{\partial x}(x,t)$, and take a total of h-spatial derivatives, and k-temporal derivatives of each side. □

   Remark 5. *This rule extends to identities of the form $\frac{\partial^s}{\partial t^s} f(x,t) = \frac{\partial^r}{\partial x^r} u(x,t)$ for non-negative integers $s, r \in \mathbb{Z}_{\geq 0}$. The end identity requires the use of falling powers of s and r, which are not required for this work, but would be necessary for PDEs of very high order.*

3. **Product rule (convolution).** The differential transform of a product

   $$f(x,t) = u(x,t)v(x,t)$$

   is a convolution given by

   $$F(h,k) = \sum_{r=0}^{h} \sum_{s=0}^{k} U(r,s)V(h-r,k-s).$$

   Proof. The term $F(h,k)$ is the coefficient of $(x-x^*)^h(t-t^*)^k$ in the Taylor expansion of F. The result follows after writing out the product $u(x,t)v(x,t)$, and rearranging the sum after a change of variables. □

4. **Quotient rule.** If $f(x,t) = \frac{v(x,t)}{u(x,t)}$, and $u(x^*,t^*) \neq 0$, then

$$F(h,k) = \frac{1}{U(0,0)} \left(V(h,k) - \sum_{\substack{0 \leq r \leq h \\ 0 \leq s \leq k \\ (r,s) \neq (0,0)}} U(r,s) F(h-r,k-s) \right). \tag{5.9}$$

Proof. Consider the function $v(x,t) = f(x,t)u(x,t)$. Apply the product rule

$$V(h,k) = \sum_{r=0}^{h} \sum_{s=0}^{k} U(r,s)F(h-r,k-s) = U(0,0)F(h,k) + \sum_{\substack{0 \leq r \leq h \\ 0 \leq s \leq k \\ (r,s) \neq (0,0)}} U(r,s)F(h-r,k-s),$$

and solve the resulting identity for $F(h,k)$. □

This quotient rule combined with the previous properties can be used to construct the differential transform of any rational function.

Remark 6. *A special case of the quotient rule is if $f(x,t) = \frac{1}{u(x,t)}$, and $u(x^*,t^*) \neq 0$, then $F(0,0) = 1/U(0,0)$, and*

$$F(h,k) = -\frac{1}{U(0,0)} \sum_{\substack{0 \leq r \leq h \\ 0 \leq s \leq k \\ (r,s) \neq (0,0)}} U(r,s)F(h-r,k-s), \quad (h,k) \neq (0,0). \tag{5.10}$$

This follows from applying the quotient rule to the constant $v(x,t) \equiv 1$, and noting that its transform is itself a delta function.

5. **Power rule.** If $f(x,t) = u(x,t)^s$, then $F(0,0) = U(0,0)^s$, and

$$\begin{aligned} F(h,k+1) = \frac{s}{(k+1)U(0,0)} \Bigg(& \sum_{\substack{0 \leq r \leq h \\ 0 \leq s \leq k}} (k-s+1)F(r,s)U(h-r,k-s+1) \\ & - \sum_{\substack{0 \leq r \leq h \\ 0 \leq s \leq k \\ (r,s) \neq (0,0)}} (k-s+1)U(r,s)F(h-r,k-s+1) \Bigg), \end{aligned} \tag{5.11}$$

and

$$F(h+1,k) = \frac{s}{(h+1)U(0,0)} \left(\sum_{\substack{0\le r\le h\\ 0\le s\le k}} (h-r+1)F(r,s)U(h-r+1,k-s) \right.$$
$$\left. - \sum_{\substack{0\le r\le h\\ 0\le s\le k\\ (r,s)\neq(0,0)}} (h-r+1)U(r,s)F(h-r+1,k-s) \right). \tag{5.12}$$

Proof. First note that the constant term in the Taylor expansion is $F(0,0) = U(0,0)^s$. The starting point in this proof is to first differentiate f with respect to time and observe that

$$\frac{\partial f}{\partial t}(x,t) = su(x,t)^{s-1}\frac{\partial u}{\partial t}(x,t) = s\frac{u(x,t)^s}{u(x,t)}\frac{\partial u}{\partial t}(x,t) = s\frac{f(x,t)}{u(x,t)}\frac{\partial u}{\partial t}(x,t).$$

Next one can apply the product rule followed by the quotient rule to the right hand side. First, define $w = u(x,t)^s \frac{\partial u}{\partial t}(x,t)$. Then one can apply the quotient rule to the right hand side, and observe that the differential transform of $\frac{\partial f}{\partial t}(x,t)$ evaluated at (r,s) is $(s+1)F(r,s+1)$.

This yields

$$(k+1)F(h,k+1) = \frac{s}{U(0,0)} \left(W(h,k) - \sum_{\substack{0\le r\le h\\ 0\le s\le k\\ (r,s)\neq(0,0)}} (k-s+1)U(r,s)F(h-r,k-s+1) \right).$$

One must then define $W(h,k)$ using the product rule. Keep in mind that the differential transform of $\frac{\partial u}{\partial t}(x,t)$ evaluated at (r,s) is $(s+1)U(r,s+1)$. This says that

$$W(k,h) = \sum_{\substack{0\le r\le h\\ 0\le s\le k\\ (r,s)\neq(0,0)}} F(r,s)(k-s+1)U(h-r,k-s+1),$$

which produces the result after dividing both sides of this identity by $k+1$. This procedure can be repeated starting by differentiating with respect to x to obtain the second equation. □

Remark 7. *As an alternative, it is possible to start the derivation of Equation* (5.11) *by taking a derivative with respect to x, but this complicates the definitions for multiple spatial dimensions.*

Many other properties exist. They can be found in [131, 130]. The interested reader is referred to [14] for an extensive review of differential transforms, along with an outline of their utility as analytical and numerical tools.

5.4 Applications to hyperbolic problems: The 1D case

This section derives recursive definitions for the time derivatives of the conserved variables in several hyperbolic PDEs.

A 1D hyperbolic conservation law is a partial differential equation defined by

$$q_t + f(q)_x = 0, \quad q(x,0) = q_0(x), \tag{5.13}$$

where $f : \mathbb{R}^m \to \mathbb{R}^m$ is a flux function whose Jacobian $f'(q)$ is diagonalizable for all $q : [a,b] \times \mathbb{R}^+ \to \mathbb{R}^m$ in the domain of interest. Eqn. (5.13) can be rewritten as

$$q_t = -f_x.$$

One can then take a total of k time derivatives, and h spatial derivatives of both sides of the equation. This allows the construction of higher time derivatives of the conserved variables with the recursive relationship

$$Q(h,k+1) = -\frac{h+1}{k+1} F(h+1,k), \tag{5.14}$$

where Q is the discrete transform of q, and F is the discrete transform of the flux function f. By convention, capital letters are used to denote the differential transform of a function in (h,k) space, and lower case letters to denote the function in (x,t) space. Each problem presented depends on the form of the flux function, together with the rules defined in Section 5.3.

It is useful to work through a few examples demonstrating how this applies to solving hyperbolic differential equations. This illustration will begin with linear and nonlinear scalar examples, then move to linear systems of equations. Finally Euler's system of equations for gas dynamics will be considered. The chief goal of this section is to define the recursive relationship in Eqn. (5.14) for a large class of problems, which in principle could be coupled with many spatial discretizations.

5.4.1 Linear advection

First consider a scalar flux function of the form $f(q) = aq$, where $a \in \mathbb{R}$ is a real constant. Immediately this gives (by linearity of the DT), that $F(h,k) = aQ(h,k)$, and therefore

$$Q(h,k+1) = -a\frac{h+1}{k+1}Q(h+1,k). \tag{5.15}$$

Remark 8. *For linear advection with constant coefficients, this algorithm defines* $\frac{\partial^h}{\partial t^h}q = (-a)^h\frac{\partial^h}{\partial x^h}q$*, which can also be realized by directly identifying that* $\frac{\partial}{\partial_t} = -a\frac{\partial}{\partial_x}$*, and then raising this operator to higher orders.*

5.4.2 Burgers' equation

The next problem to consider is the nonlinear scalar Burgers equation $q_t + (\frac{1}{2}q^2)_x = 0$, where the flux function is defined by $f(q) = \frac{1}{2}q^2$. The DT of this function comes from the product rule and linearity

$$F(h,k) = -\frac{1}{2}\sum_{r=0}^{h}\sum_{s=0}^{k}Q(r,s)Q(h-r,k-s). \tag{5.16}$$

With this definition for the fluxes, the required recursive definition for higher derivatives of the conserved variables is simply

$$Q(h,k+1) = -\frac{1}{2}\sum_{r=0}^{h+1}\sum_{s=0}^{k}Q(r,s)Q(h+1-r,k-s). \tag{5.17}$$

5.4.3 Linear systems of equations

The next example involves a system of differential equations. For example, in a single dimension, Maxwell's equations reduces to solving

$$\begin{pmatrix} q^1 \\ q^2 \end{pmatrix}_t + \begin{pmatrix} q^2 \\ -q^1 \end{pmatrix}_x = 0. \tag{5.18}$$

The differential transform of this system couples both equations through

$$\begin{aligned} Q^1(h,k+1) &= -\frac{h+1}{k+1}Q^2(h+1,k) \\ Q^2(h,k+1) &= \frac{h+1}{k+1}Q^1(h+1,k). \end{aligned}$$

5.4.4 Shallow water equations

The single-layer shallow water equations (in one dimension) are given by the non-linear system

$$\begin{pmatrix} h \\ hu \end{pmatrix}_t + \begin{pmatrix} hu \\ \frac{1}{h}(hu)^2 + \frac{1}{2}gh^2 \end{pmatrix}_x = 0 \tag{5.19}$$

where h is the water height measured from the bottom, u is the wind velocity, and g is the gravitational constant. The variables $(q_1, q_2) = (h, hu)$ are conserved for all time.

The discrete transform for this system (in one dimension) can be found in the recent work of Norman and Finkel [91]. Each component has its own (coupled) recursive relationship given by

$$Q_1(h, k+1) = -\frac{h+1}{k+1} Q^2(h+1, k) \tag{5.20}$$

$$Q_2(h, k+1) = -\frac{h+1}{k+1} G_1(h+1, k) - \frac{1}{2} g \frac{h+1}{k+1} G_2(h+1, k), \tag{5.21}$$

where additional auxiliary functions G_1, and G_2, are the transforms of hu^2, and h^2 respectively. In practice, it is useful to start with building blocks (such as $1/h = 1/q_1$ to compute $(hu)^2/h = (q_2)^2/q_2$) and then build up more complicated terms from those.

5.4.5 Euler equations

The compressible Euler equations have been defined in Equation (3.11). In a single dimension, these equations reduce to the evolution of three conserved variables $q := (\rho, \rho u, \mathcal{E})^T$ through

$$\begin{pmatrix} \rho \\ \rho u \\ \mathcal{E} \end{pmatrix}_t + \begin{pmatrix} \rho u \\ \rho u^2 + p \\ (\mathcal{E} + p)u \end{pmatrix}_x = 0. \tag{5.22}$$

The goal of the DT method is to construct a recursive definition for the differential transforms (DTs) of the conserved variables

$$\begin{aligned} R(h,k) &:= \frac{1}{h!k!} \frac{\partial^h}{\partial x^h} \frac{\partial^k}{\partial t^k} \rho(x,t)|_{(x^*,t^*)}, \\ MX(h,k) &:= \frac{1}{h!k!} \frac{\partial^h}{\partial x^h} \frac{\partial^k}{\partial t^k} \rho(x,t)u(x,t)|_{(x^*,t^*)}, \\ \mathbb{E}(h,k) &:= \frac{1}{h!k!} \frac{\partial^h}{\partial x^h} \frac{\partial^k}{\partial t^k} \mathcal{E}(x,t)|_{(x^*,t^*)}. \end{aligned} \tag{5.23}$$

These transforms are computed by way of local auxiliary variables. First, begin with the transform of $(\rho u)/\rho$, which is

$$GX_r(h,k) = \frac{1}{R(0,0)}\left(MX(h,k) - \sum_{\substack{0\le r\le h\\ 0\le s\le k\\ (r,s)\neq(0,0)}} R(r,s)GX_r(h-r,k-s) \right). \tag{5.24}$$

Next, compute the product with the momentum to construct $(\rho u)^2/\rho$:

$$G_{rmx2}(h,k) = \sum_{r=0}^{h}\sum_{s=0}^{k} GX_r(r,s)MX(h-r,k-s). \tag{5.25}$$

With these variables, compute the DT of the pressure from Equation (3.12), which is simply

$$P(h,k) = (\gamma-1)\left(\mathbb{E}(h,k) - \frac{1}{2}G_{rmx2}(h,k)\right). \tag{5.26}$$

Finally, one may compute the product of the sum of the energy density and pressure against the first component of the velocity $(\mathcal{E}+p)u = (\mathcal{E}+p)\frac{\rho u}{\rho}$:

$$EX(h,k) = \sum_{r=0}^{h}\sum_{s=0}^{k} \left(\mathbb{E}(r,s) + P(r,s)\right) GX_r(h-r,k-s). \tag{5.27}$$

After appealing to the recursive definition of the DT, one can write the required recursive definition of the DT of the conserved variables in Eqn. (5.23) as

$$\begin{aligned} R(h,k+1) &= -\frac{h+1}{k+1}MX(h+1,k),\\ MX(h,k+1) &= -\frac{h+1}{k+1}\left(G_{rmx2}(h+1,k) + P(h+1,k)\right),\\ \mathbb{E}(h,k+1) &= -\frac{h+1}{k+1}EX(h+1,k). \end{aligned} \tag{5.28}$$

5.4.6 Isentropic Euler equations

To illustrate that DTs can be used to handle problems with flux functions that are not rational functions of the conserved variables, the isentropic Euler equations are considered as the final example. These equations evolve the mass density ρ, and momentum ρu through a conservation law

$$\begin{pmatrix} \rho \\ \rho u \end{pmatrix}_t + \begin{pmatrix} \rho u \\ \rho u^2 + p \end{pmatrix}_x = 0, \tag{5.29}$$

where the pressure p is fixed at $p(\rho) = \rho^\gamma$, and $\gamma > 1$ is a constant. By the power rule, its transform is simply $P(0,0) = R(0,0)^\gamma$, and

$$P(h,k+1) = \frac{\gamma}{(k+1)P(0,0)} \left(R(h,k) - \sum_{\substack{0\le r\le h \\ 0\le s\le k \\ (r,s)\neq(0,0)}} (k-s+1)R(r,s)P(h-r,k-s+1) \right), \tag{5.30}$$

where R is the DT of the density ρ. Finally, the recursive relationship is given by

$$\begin{aligned} R(h,k+1) &= -\frac{h+1}{k+1} MX(h+1,k), \\ MX(h,k+1) &= -\frac{h+1}{k+1} \left(G_{rmx2}(h+1,k) + P(h+1,k) \right), \end{aligned} \tag{5.31}$$

where $G_{rmx2}(h+1,k)$ is defined in Eqn. (5.25), and the pressure is now given by Eqn. (5.30).

5.5 Differential transforms for functions of three variables

Extensions to higher dimensions follow from what is presented in Section 5.3. If $f(x,y,t)$, is sufficiently smooth, the *differential transform* of f centered at (x^*, y^*, t^*) is defined as

$$F(h_1,h_2,k) := \frac{1}{h_1!h_2!k!} \frac{\partial^{h_1}}{\partial x^{h_1}} \frac{\partial^{h_2}}{\partial y^{h_2}} \frac{\partial^k}{\partial t^k} f(x,y,t)|_{(x^*,y^*,t^*)}, \quad h_1,h_2,k \in \mathbb{Z}_{\ge 0}. \tag{5.32}$$

Again, the inverse follows from the multidimensional Taylor series of the function:

$$f(x,y,t) = \sum_{h_1=0}^{\infty} \sum_{h_2=0}^{\infty} \sum_{k=0}^{\infty} F(h_1,h_2,k)(x-x^*)^{h_1}(y-y^*)^{h_2}(t-t^*)^k. \tag{5.33}$$

The same properties from the previous section follow, with minor modifications concerning the number of indices. For brevity, their proofs are omitted. Again, assume that u and v are real analytic.

1. **Linearity.** If $c \in \mathbb{R}$ is a constant, and $f(x,y,t) = u(x,y,t) + cv(x,y,t)$, then $F(h_1,h_2,k) = U(h_1,h_2,k) + cV(h_1,h_2,k)$.

2. **Recursive definition of derivatives.** If $\frac{\partial f}{\partial t}(x,y,t) = \frac{\partial u}{\partial x}(x,y,t) + \frac{\partial v}{\partial y}(x,y,t)$, then

$$F(h_1,h_2,k+1) = \frac{h_1+1}{k+1} U(h_1+1,h_2,k) + \frac{h_2+1}{k+1} V(h_1,h_2+1,k).$$

3. **Product rule.** The product rule defines a discrete convolution of two separate DTs. If $f(x,y,t) = u(x,y,t)v(x,y,t)$ is a product of two functions, then its differential transform is

$$F(h_1,h_2,k) = \sum_{r_1=0}^{h_1}\sum_{r_2=0}^{h_2}\sum_{s=0}^{k} U(r_1,r_2,s)V(h_1-r_1,h_2-r_2,k-s),$$

which is independent of the order in which the summation takes place.

4. **Quotient rule.** If $f(x,y,t) = \frac{v(x,y,t)}{u(x,y,t)}$, is a quotient of two functions, and $u(x^*,y^*,t) \neq 0$, then

$$F(h_1,h_2,k) = \frac{1}{U(0,0,0)}\left(V(h_1,h_2,k) - \sum_{\substack{0\leq r_1\leq h_1\\ 0\leq r_2\leq h_2\\ 0\leq s\leq k\\ (r_1,r_2,s)\neq(0,0,0)}} U(r_1,r_2,s)F(h_1-r_1,h_2-r_2,k-s)\right). \tag{5.34}$$

These definitions extend to even higher dimensions, see [131] for many results. Note that multinomial coefficients can be used to compactly express the summands.

5.6 Applications to hyperbolic problems: The 2D case

In two spatial dimensions, a conservation law is defined by a flux function with two components

$$q_t + f(q)_x + g(q)_y = 0, \quad q(x,y,0) = q_0(x,y), \tag{5.35}$$

where q is a vector of conserved variables. This can be rewritten as $q_t = -f_x - g_y$. Taking a total of k time derivatives, and h_1 and h_2 x- and y-derivatives respectively, one obtains a recursive definition for the conserved variables of the form

$$Q(h_1,h_2,k+1) = -\frac{h_1+1}{k+1}F(h_1+1,h_2,k) - \frac{h_2+1}{k+1}G(h_1,h_2+1,k), \tag{5.36}$$

where F is the DT of f and G is the DT of g. As in the 1D case, Equation (5.36) requires one to identify the precise form of F and G for a particular hyperbolic problem.

5.6.1 Linear advection

In two dimensions the scalar advection equation has the form $q_t + (a_1 q)_x + (a_2 q)_y = 0$, where $a_1, a_2 \in \mathbb{R}$ are constants. The differential transform of this system is simply

$$Q(h_1,h_2,k+1) = -a_1\frac{h_1+1}{k+1}Q(h_1+1,h_2,k) - a_2\frac{h_2+1}{k+1}Q(h_1,h_2+1,k). \tag{5.37}$$

5.6.2 Burgers' equation

In 2D, Burgers' equation is $q_t + (q^2/2)_x + (q^2/2)_y = 0$. The differential transform of this system uses a single auxiliary variable for the differential transform of $q^2/2$:

$$Q_2(h_1, h_2, k) = \frac{1}{2} \sum_{r_1=0}^{h_1} \sum_{r_2=0}^{h_2} \sum_{s=0}^{k} Q(r_1, r_2, s) Q(h_1 - r_1, h_2 - r_2, k - s), \tag{5.38}$$

and then the recursive definition is simply

$$Q(h_1, h_2, k+1) = -\frac{h_1 + 1}{k+1} Q_2(h_1 + 1, h_2, k) - \frac{h_2 + 1}{k+1} Q_2(h_1, h_2 + 1, k). \tag{5.39}$$

5.6.3 Euler equations

In two dimensions, the Euler equations reduce to the evolution of four conserverd variables $q := (\rho, \rho u, \rho v, \mathcal{E})^T$ through Equation (3.11). One must construct recursive definitions for the differential transforms (DTs) of the conserved variables

$$\begin{aligned}
R(h_1, h_2, k) &:= \frac{1}{h_1! h_2! k!} \frac{\partial^{h_1}}{\partial x^{h_1}} \frac{\partial^{h_2}}{\partial y^{h_2}} \frac{\partial^k}{\partial t^k} \rho(x, y, t)|_{(x^*, y^*, t^*)}, \\
MX(h_1, h_2, k) &:= \frac{1}{h_1! h_2! k!} \frac{\partial^{h_1}}{\partial x^{h_1}} \frac{\partial^{h_2}}{\partial y^{h_2}} \frac{\partial^k}{\partial t^k} \rho(x, y, t) u(t, x)|_{(x^*, y^*, t^*)}, \\
MY(h_1, h_2, k) &:= \frac{1}{h_1! h_2! k!} \frac{\partial^{h_1}}{\partial x^{h_1}} \frac{\partial^{h_2}}{\partial y^{h_2}} \frac{\partial^k}{\partial t^k} \rho(x, y, t) v(t, x)|_{(x^*, y^*, t^*)}, \\
\mathbb{E}(h_1, h_2, k) &:= \frac{1}{h_1! h_2! k!} \frac{\partial^{h_1}}{\partial x^{h_1}} \frac{\partial^{h_2}}{\partial y^{h_2}} \frac{\partial^k}{\partial t^k} \mathcal{E}(x, y, t)|_{(x^*, y^*, t^*)}.
\end{aligned} \tag{5.40}$$

This, again, requires a handful of auxiliary variables. Begin with the transform of $(\rho u)/\rho$ and $(\rho v)/\rho$:

$$\begin{aligned}
GX_r(h_1, h_2, k) &= \frac{1}{R(0,0,0)} \left(MX(h_1, h_2, k) - \sum_{\substack{0 \le r_1 \le h_1 \\ 0 \le r_2 \le h_2 \\ 0 \le s \le k \\ (r_1, r_2, s) \ne (0,0,0)}} R(r_1, r_2, s) G_r(h_1 - r_1, h_2 - r_2, k - s) \right), \\
GY_r(h_1, h_2, k) &= \frac{1}{R(0,0,0)} \left(MY(h_1, h_2, k) - \sum_{\substack{0 \le r_1 \le h_1 \\ 0 \le r_2 \le h_2 \\ 0 \le s \le k \\ (r_1, r_2, s) \ne (0,0,0)}} R(r_1, r_2, s) G_r(h_1 - r_1, h_2 - r_2, k - s) \right).
\end{aligned} \tag{5.41}$$

Next, compute products against the momentum variables ρu and ρv to construct $(\rho u)^2/\rho$, and $(\rho v)^2/\rho$:

$$\begin{aligned} G_{rmx2}(h_1,h_2,k) &= \sum_{r_1=0}^{h_1}\sum_{r_2=0}^{h_2}\sum_{s=0}^{k} GX_r(r_1,r_2,s)MX(h_1-r_r,h_2-r_2,k-s), \\ G_{rmy2}(h_1,h_2,k) &= \sum_{r_1=0}^{h_1}\sum_{r_2=0}^{h_2}\sum_{s=0}^{k} GY_r(r_1,r_2,s)MY(h_1-r_r,h_2-r_2,k-s). \end{aligned} \tag{5.42}$$

With these variables, the DT of the pressure is simply

$$P(h_1,h_2,k) = (\gamma-1)\left(\mathbb{E}(h_1,h_2,k) - \frac{1}{2}\left(G_{rmx2}(h_1,h_2,k) + G_{rmy2}(h_1,h_2,k)\right)\right). \tag{5.43}$$

Finally, compute products of the sum of the energy density and pressure against the first and second component of the velocity $(\mathcal{E}+p)u = (\mathcal{E}+p)\frac{\rho u}{\rho}$ and $(\mathcal{E}+p)v = (\mathcal{E}+p)\frac{\rho v}{\rho}$:

$$\begin{aligned} EX(h_1,h_2,k) &= \sum_{r_1=0}^{h_1}\sum_{r_2=0}^{h_2}\sum_{s=0}^{k} \left(\mathbb{E}(r_1,r_2,s)+P(r_1,r_2,s)\right) GX_r(h_1-r_1,h_2-r_2,k-s), \\ EY(h_1,h_2,k) &= \sum_{r_1=0}^{h_1}\sum_{r_2=0}^{h_2}\sum_{s=0}^{k} \left(\mathbb{E}(r_1,r_2,s)+P(r_1,r_2,s)\right) GY_r(h_1-r_1,h_2-r_2,k-s). \end{aligned} \tag{5.44}$$

The final term required for the two dimensional Euler equations is the transform of $\rho uv = (\rho u)(\rho v)/\rho$, which is denoted by GXY_r, and can be computed by first applying the product rule to obtain $(\rho u)(\rho v)$, and then applying the quotient rule to the result to construct $(\rho u)(\rho v)/\rho$.

After appealing to the recursive definition of the DT, one can now write the required DT of the conserved variables in Eqn. (5.40) as

$$\begin{aligned} R(h_1,h_2,k+1) &= -\frac{h_1+1}{k+1}MX(h_1+1,h_2,k) - \frac{h_2+1}{k+1}MX(h_1,h_2+1,k), \\ MX(h_1,h_2,k+1) &= -\frac{h_1+1}{k+1}\left(G_{rmx2}(h_1+1,h_2,k)+P(h_1+1,h_2,k)\right) \\ &\quad - \frac{h_2+1}{h+1}GXY_r(h_1,h_2+1,k), \\ MY(h_1,h_2,k+1) &= -\frac{h_1+1}{h+1}GXY_r(h_1+1,h_2,k) \\ &\quad - \frac{h_2+1}{k+1}\left(G_{rmy2}(h_1,h_2+1,k)+P(h_1,h_2+1,k)\right), \\ \mathbb{E}(h_1,h_2,k+1) &= -\frac{h_1+1}{k+1}EX(h_1+1,h_2,k) - \frac{h_2+1}{k+1}EY(h_1,h_2+1,k). \end{aligned} \tag{5.45}$$

Chapter 6

THE PIF-WENO METHOD WITH DT TIME STEPPING

6.1 Introduction

This Chapter is also drawn from the same currently unpublished work done with David Seal as in Chapter 5. As in Chapter 5, this chapter is focused on illustrating how differential transforms may be used to numerically solve PDEs. The numerical method used in the chapter is a basic central finite difference scheme stabilized by the PIF-WENO method [26]. The point of this chapter is to illustrate the use of DTs to construct a high-order numerical method. Using this numerical scheme allows the differential transform method to be tested while avoiding complications like solving Riemann problems. The particular numerical method was selected because central finite differences provide a very straightforward and accurate way to create a high order Taylor series of a function. PIF-WENO is used because something must be done to stabilize the central finite difference discretization as such discretizations are not stable for hyperbolic PDEs.

6.2 PIF WENO with differential transforms: The 1D case

This chapter develops a simple numerical method that will need further modification in order to truly perform well on problems involving strong shocks. The method developed here is only meant to serve as an example illustrating the effectiveness of DT based time-stepping. In developing this method it is important to keep in mind the following properties that numerical methods for hyperbolic conservation laws should satisfy:

- **Conservation.** In order to produce the correct entropy solution for a problem that contains shocks, mass conservation (at the discrete level) is necessary (for most problems).

- **Shock capturing.** For hyperbolic problems that develop shocks, high derivatives introduce spurious oscillations that pollute the numerical solution. Without shock capturing technology, these oscillations can cause the codes to fail.

This work makes use of the so-called Picard integral formulation of finite difference WENO methods [26] in order to retain these two properties. This method starts by conducting formal integration of Eqn. (5.13) over a single time interval $[t^n, t^{n+1}]$. This defines

the exact solution

$$q(x,t^{n+1}) = q(x,t^n) - \Delta t F_x^n. \tag{6.1a}$$

as an integral equation coupled through a *time-averaged flux*

$$F^n(x) := \frac{1}{\Delta t}\int_{t^n}^{t^{n+1}} f(q(x,t))\ dt. \tag{6.1b}$$

The basic idea of the PIF-WENO scheme is to a) construct a discretization of the time averaged fluxes, and then b) apply the nonlinear WENO reconstruction in order to define the derivative in Eqn. (6.1a). This results in a conservative and essentially non-oscillatory update for the solution.

Remark 9. *In [26], the authors approximated* (6.1b) *through a finite Taylor series*

$$F^n(x) \approx f(q(x,t^n)) + \frac{\Delta t}{2} f_t(q(x,t^n)) + \frac{\Delta t^2}{3!} f_{tt}(q(x,t^n)) + \cdots . \tag{6.2}$$

which requires a temporal expansion of the fluxes. They discretized Eqn. (6.2) *with a Taylor series containing only three terms due in part to the increasing complexity of the additional terms that would be required to go higher-order. The aim of this chapter is to make use of DTs to construct these additional higher-order terms.*

The full details are outlined in the following Algorithm.

The PIF-WENO algorithm

Let us first assume that the spatial domain $[a,b]$ is discretized by a uniform grid $x_i = a + (i - \frac{1}{2})\Delta x$, where $\Delta x = (b-a)/m_x$ defines the cell width. The approximate solution at a discrete time level t^n is then $q_i^n \approx q(x_i, t^n)$. The following is a method that attains order $M = 2r+1, r \in \mathbb{Z}_{\geq 0}$.

1. Obtain approximate values for q^n at the points x_i and then apply central finite differences on the stencil $\left\{q_{i-r}^n, q_{i-r+1}^n, \ldots, q_{i+r}^n\right\}$ to construct spatial derivatives $\left\{q_i^n, q_{i,x}^n, q_{i,xx}^n, \ldots, \partial_x^M q_i^n\right\}$ of the current solution q_i^n at grid point x_i. After rescaling, these spatial derivatives define starting values

   $$Q(0,0), Q(1,0), \ldots, Q(M,0)$$

   that can be later converted to temporal derivatives. This work makes use of a single finite difference stencil to compute all of the derivatives. The procedure is outlined in Section 6.3. Because of the increasing powers of Δt in the Taylor expansion of the update, high-order accuracy is retained even as each derivative loses an order of accuracy.

2. Sequentially, use the recursive definition in Eqn. (5.14) to extract time derivatives

$$Q(0,0), Q(0,1), Q(0,2), \ldots Q(0,M).$$

3. Construct a continuous Taylor expansion of the conserved variables at each point $x = x_i$:

$$q_i(t) = q_i^n + (t-t^n)q_{i,t}^n + \frac{(t-t^n)^2}{2!}q_{i,tt}^n + \cdots + \frac{(t-t^n)^M}{M!}\frac{\partial^M}{\partial t^M}q_i^n = \sum_{m=0}^{M}(t-t^n)^m Q(0,m). \tag{6.3}$$

4. Define the time averaged flux through numerical quadrature in time

$$F^n(x_i) \approx \sum_{m=1}^{M} \omega_m f\left(q_i\left(t^n + c_m \Delta t\right)\right), \tag{6.4}$$

where c_m and ω_m are the (rescaled) Gaussian quadrature points, and weights, respectively, on the interval $[0,1]$. The function values are found by evaluating Eqn. (6.3) at each intermediate time point.

5. Apply nonlinear WENO reconstruction to the resulting time averaged fluxes from Eqn. (6.4), and update the solution through a time step that looks like a forward Euler time step. The update will use the procedure described in [26]. The WENO methods used will be the methods of orders $5, 7, 9$ and 11 that can be found in [7].

This completes the 1D method. Before describing how to obtain the central difference coefficients, there are two important remarks to make.

Remark 10. *It is tempting to use Equation* (6.3) *to update the solution. Despite the fact that this is a high-order update, this update would lack conservation, and fail to limit the solution. The process in this chapter simply uses this expansion to construct fluxes, which are then limited through the nonlinear WENO reconstruction procedure in* **Step 5**.

Remark 11. *The numerical integration used in Eqn.* (6.4) *of* **Step 4** *is similar to the one carried out by Harten et. al [55] in their original ENO work from the 1980's. However in the current case, expansions are carried out for finite difference, and not finite volume discretizations.*

6.3 Central difference coefficients

Central difference coefficients for a single stencil can be constructed by inverting a single matrix. The matrix inversion yields high-order accuracy for the first derivative, and it decreases orders of accuracy for each higher derivative.

Consider a $2r+1$ point central stencil for a discrete function u_i at grid points x_i

$$u_{i+s} \approx u(x_{i+s}), \quad -r \le s \le r,$$

where $u(x_{i+s})$ is the exact solution. With a uniform grid length of h, Taylor expansions and truncation of the higher order terms yield a system of equations

$$\begin{aligned}
u(x_{i-s}) &\approx u|_{x_i} - (sh)\frac{d}{dx}u|_{x_i} + \frac{(sh)^2}{2!}\frac{d^2}{dx^2}u|_{x_i} - \frac{(sh)^3}{3!}\frac{d^3}{dx^3}u|_{x_i} + \cdots + \frac{(-sh)^{2r+1}}{(2r)!}\frac{d}{dx}^{2r}u|_{x_i} \\
u(x_{i-s+1}) &\approx u|_{x_i} - ((s-1)h)\frac{d}{dx}u|_{x_i} + ((s-1)h)^2\frac{d^2}{dx^2}u|_{x_i} - ((s-1)h)^3\frac{d^3}{dx^3}u|_{x_i} \\
&\quad + \cdots + ((s-1)h)^{2r}\frac{d^{2r}}{dx^{2r}}u|_{x_i} \\
&\vdots \\
u(x_{i+s-1}) &\approx u|_{x_i} + ((s-1)h)\frac{d}{dx}u|_{x_i} + ((s-1)h)^2\frac{d^2}{dx^2}u|_{x_i} + ((s-1)h)^3\frac{d^3}{dx^3}u|_{x_i} \\
&\quad + \cdots + ((s-1)h)^{2r}\frac{d^{2r}}{dx^{2r}}u|_{x_i} \\
u(x_{i+s}) &\approx u|_{x_i} + (sh)\frac{d}{dx}u|_{x_i} + (sh)^2\frac{d^2}{dx^2}u|_{x_i} + (sh)^3\frac{d^3}{dx^3}u|_{x_i} + \cdots + (sh)^{2r}\frac{d^{2r}}{dx^{2r}}u|_{x_i}
\end{aligned} \tag{6.5}$$

This yields a linear system

$$\mathbf{u} = M\mathbf{v}$$

where M is the Vandermonde matrix

$$M = \begin{pmatrix}
1 & -sh & (-sh)^2 & (-sh)^3 & \dots & (-sh)^{2r} \\
1 & -(s-1)h & (-(s-1)h)^2 & (-(s-1)h)^3 & \dots & (-(s-1)h)^{2r} \\
\vdots & & & & & \\
1 & (s-1)h & ((s-1)h)^2 & ((s-1)h)^3 & \dots & ((s-1)h)^{2r} \\
1 & sh & (sh)^2 & (sh)^3 & \dots & (sh)^{2r}
\end{pmatrix},$$

and $\mathbf{v}$ is the vector containing all the derivatives of u evaluated at $x = x_i$:

$$\mathbf{v} = \left(u|_{x_i}, \frac{d}{dx}u|_{x_i} \frac{d^2}{dx^2}u|_{x_i} \frac{d^3}{dx^3}u|_{x_i} + \dots \frac{d^{2r}}{dx^{2r}}u|_{x_i} \right).$$

These matrices can be inverted using any symbolic software package to yield high-order numerical values for a total of $2r$ derivatives of u.

6.3.1 Examples

The case $r = 1$ defines the linear system

$$\begin{aligned} u(x_{i-1}) &\approx u|_{x_i} - h\frac{d}{dx}u|_{x_i} + \frac{s^2}{2!}\frac{d^2}{dx^2}u|_{x_i}, \\ u(x_i) &\approx u|_{x_i}, \\ u(x_{i-1}) &\approx u|_{x_i} + h\frac{d}{dx}u|_{x_i} + \frac{s^2}{2!}\frac{d^2}{dx^2}u|_{x_i}. \end{aligned} \tag{6.6}$$

Solving this system for each derivative yields

$$\begin{aligned} u|_{x_i} &\approx u(x_i), \\ \frac{d}{dx}u|_{x_i} &\approx \frac{1}{2h}\left(u(x_{i+1}) - u(x_{i-1})\right), \\ \frac{d^2}{dx^2}u|_{x_i} &\approx \frac{1}{h^2}\left(u(x_{i+1}) - 2u(x_i) + u(x_{i-1})\right). \end{aligned} \tag{6.7}$$

The case $r = 2$ defines the linear system

$$\begin{aligned} u(x_{i-2}) &\approx u|_{x_i} - (2h)\frac{d}{dx}u|_{x_i} + \frac{(2h)^2}{2!}\frac{d^2}{dx^2}u|_{x_i} - \frac{(2h)^3}{3!}\frac{d^3}{dx^3}u|_{x_i} + \frac{(2h)^4}{4!}\frac{d^4}{dx^4}u|_{x_i}, \\ u(x_{i-1}) &\approx u|_{x_i} - h\frac{d}{dx}u|_{x_i} + \frac{h^2}{2!}\frac{d^2}{dx^2}u|_{x_i} - \frac{h^3}{3!}\frac{d^3}{dx^3}u|_{x_i} + \frac{h^4}{4!}\frac{d^4}{dx^4}u|_{x_i}, \\ u(x_i) &\approx u|_{x_i}, \\ u(x_{i+1}) &\approx u|_{x_i} + h\frac{d}{dx}u|_{x_i} + \frac{h^2}{2!}\frac{d^2}{dx^2}u|_{x_i} + \frac{h^3}{3!}\frac{d^3}{dx^3}u|_{x_i} + \frac{h^4}{4!}\frac{d^4}{dx^4}u|_{x_i}, \\ u(x_{i+2}) &\approx u|_{x_i} - (2h)\frac{d}{dx}u|_{x_i} + \frac{(2h)^2}{2!}\frac{d^2}{dx^2}u|_{x_i} - \frac{(2h)^3}{3!}\frac{d^3}{dx^3}u|_{x_i} + \frac{(2h)^4}{4!}\frac{d^4}{dx^4}u|_{x_i}. \end{aligned} \tag{6.8}$$

Inverting this system for each derivative yields the following approximations

$$\begin{aligned} u|_{x_i} &\approx u(x_i), \\ \frac{d}{dx}u|_{x_i} &\approx \frac{1}{12h}\left(-u(x_{i+2}) + 8u(x_{i+1}) - 8u(x_{i-1}) + u(x_{i-2})\right), \\ \frac{d^2}{dx^2}u|_{x_i} &\approx \frac{1}{12h^2}\left(-u(x_{i+2}) + 16u(x_{i+1}) - 30u(x_i) + 16u(x_{i-1}) - u(x_{i-2})\right), \\ \frac{d^3}{dx^3}u|_{x_i} &\approx \frac{1}{2h^3}\left(u(x_{i+2}) - 2u(x_{i+1}) + 2u(x_{i-1}) - u(x_{i-2})\right), \\ \frac{d^4}{dx^4}u|_{x_i} &\approx \frac{1}{h^4}\left(u(x_{i+2}) - 4u(x_{i+1}) + 6u(x_i) - 4u(x_{i-1}) + u(x_{i-2})\right). \end{aligned} \tag{6.9}$$

6.4 Numerical Results: 1D examples

The purpose of the numerical results presented here are to demonstrate the suitability of differential transforms for hyperbolic problems. Our aim is to indicate how they can be used to define high-order numerical methods.

6.4.1 Linear advection

Our first numerical example is linear advection

$$q_t + q_x = 0, \quad x \in [0, 1], \tag{6.10}$$

with initial conditions of varying degrees of smoothness.

Linear advection: Smooth initial conditions

First consider (6.10) with initial conditions prescribed by $q_0(x) = 1.0 + 0.2\sin(4\pi x)$. This solution will be integrated to a final time of $T = 1$, at which point the exact solution is identical to the initial conditions. Relative errors will be computed using

$$\text{Error} := \frac{\|q - q_{\text{exact}}\|}{\|q_{\text{exact}}\|} = \frac{\sum_i |q_i - q(x_i)|}{\sum_i |q(x_i)|}. \tag{6.11}$$

In Table 6.1, a convergence study for the fifth, seventh, and ninth-order solvers are presented. Results for the eleventh order solver hit machine precision before being able to observe the formal accuracy of the method.

6.4.2 Isentropic Euler equations

In order to demonstrate a more complicated differential transform, we next consider the isentropic Euler equations from Eqn. (5.29). Consider initial conditions defined by

$$(\rho, u)^T = (1.0 + 0.2\sin 4\pi x, 1.0)^T, \tag{6.12}$$

and set $\gamma = 1.4$.

It is difficult to obtain initial conditions that have an exact solution for this set of equations and so in order to show convergence we will compare the 5th order DT solution to a 5th order Runge Kutta (RK) scheme. Since the Runge Kutta scheme only relies on having the correct flux, we can be sure that it is converging to the correct solution. Let us define the errors as

$$\text{Error} := \frac{\|q_{\text{dt}} - q_{\text{rk}}\|}{\|q_{\text{rk}}\|}. \tag{6.13}$$

The solution is integrated from the stated initial conditions to a final time of $T = 0.15$ with CFL $= 0.1$. The results can be seen in Table 6.2.

Mesh	M5 error	Order	M7 error	Order	M9 error	Order
50	2.63×10^{-05}	—	3.54×10^{-07}	—	4.94×10^{-09}	—
65	7.11×10^{-06}	4.98	5.68×10^{-08}	6.97	4.71×10^{-10}	8.96
84	1.98×10^{-06}	4.87	9.48×10^{-09}	6.82	4.71×10^{-11}	8.78
109	5.39×10^{-07}	4.96	1.53×10^{-09}	6.94	4.53×10^{-12}	8.92
142	1.44×10^{-07}	5.04	2.41×10^{-10}	7.05	4.20×10^{-13}	9.07
185	3.84×10^{-08}	5.04	3.79×10^{-11}	7.05	4.13×10^{-14}	8.84
241	1.02×10^{-08}	5.04	5.96×10^{-12}	7.05	6.16×10^{-15}	7.25
313	2.77×10^{-09}	4.98	9.59×10^{-13}	6.96	6.90×10^{-14}	−9.21
407	7.45×10^{-10}	5.00	1.56×10^{-13}	6.92	3.56×10^{-14}	2.52
530	1.99×10^{-10}	5.03	8.45×10^{-14}	2.34	8.50×10^{-14}	−3.31
689	5.36×10^{-11}	5.00	1.45×10^{-13}	−2.06	1.45×10^{-13}	−2.03
896	1.44×10^{-11}	5.00	6.09×10^{-14}	3.30	6.09×10^{-14}	3.30
1164	3.91×10^{-12}	4.98	6.16×10^{-15}	8.73	6.22×10^{-15}	8.69
1514	1.05×10^{-12}	5.02	8.05×10^{-14}	−9.80	8.05×10^{-14}	−9.76

Table 6.1: Convergence of the 5th, 7th and 9th order differential transform methods, labeled M5, M7 and M9 respectively, on the linear advection problem. All errors are computed using the L2 norm. Note that the accuracy is affected by rounding when the error approaches machine precision, causing convergence to stop.

Mesh	L2 error	Order
400	5.27×10^{-07}	—
480	2.10×10^{-07}	5.05
576	8.45×10^{-08}	5.00
690	3.38×10^{-08}	5.03
828	1.34×10^{-08}	5.08
994	5.30×10^{-09}	5.08
1194	2.08×10^{-09}	5.13
1432	8.30×10^{-10}	5.04
1718	3.36×10^{-10}	4.96
2062	1.37×10^{-10}	4.90

Table 6.2: Convergence of the 5th order differential transform method on the isentropic Euler equations. All errors are computed using the L2 norm, and the relative error defined in (6.13) is used to determine the error of the solution.

6.4.3 Euler Equations

The WENO reconstruction used in this chapter should help maintain the stability even when the problems involve shocks. Here this method is applied to the Shu-Osher problem defined in Section 4.7.1. The results can be seen in Figure 6.1.

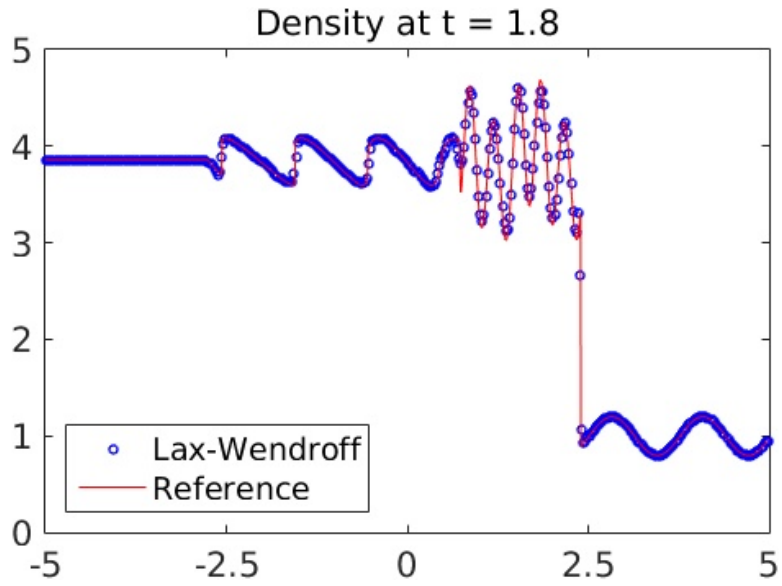


Figure 6.1: A shock wave interacting with a smooth entropy wave. The Lax-Wendroff solution was computed using a 9th order differential transform on a grid with 400 cells. The solution matches the reference, which is computed using a 10000 cell finite volume WENO simulation, very well.

6.5 PIF WENO with differential transforms: The 2D case

The Picard integral formulation of a two-dimensional conservation law

$$q_t + f(q)_x + g(q)_y = 0,$$

requires the expansion of each component of the flux through

$$F^n(x,y) := \frac{1}{\Delta t}\int_{t^n}^{t^{n+1}} f(q(x,y,t))\,dt, \quad G^n(x,y) := \frac{1}{\Delta t}\int_{t^n}^{t^{n+1}} g(q(x,y,t))\,dt. \tag{6.14a}$$

These definitions yield (exact) updates through

$$q_{ij}^{n+1} = q_{ij}^n - \Delta t\left(F^n(x_i,y_j)\right)_x - \Delta t\left(G^n(x_i,y_j)\right)_y, \tag{6.14b}$$

that require identities for the fluxes. The basic procedure is similar to the 1D case, with a handful of modifications:

The PIF-WENO method with DT time stepping: The 2D case

Step 1. Obtain approximate values of q^n at each grid point (x_i, y_j). At each grid point (x_i, y_j), construct mixed spatial derivatives of the solution. That is, apply finite differences to construct finitely many (mixed) spatial derivatives

$$q^n, q^n_x, q^n_y, q^n_{xx}, q^n_{xy}, q^n_{yy}, \dots$$

of the current solution $q = q^n$. Note that here, one must compute *mixed derivatives*, such as q^n_{xyy}, in order to seed this expansion. After rescaling, these (spatial) derivatives define starting values

$$\{Q(h_1, h_2, 0) : 0 \leq h_1, h_2 \leq M\} \tag{6.15}$$

that are used to construct temporal derivatives.

Step 2. Sequentially, use the recursive definition in Eqn. (5.36) to extract time derivatives

$$Q(0,0,0), Q(0,0,1), Q(0,0,2), \dots, Q(0,0,M).$$

Step 3. Extract a continuous (in time) expansion of the conserved variables through their Taylor expansions

$$q(x_i, y_j t) = q^n_{ij} + \Delta t q^n_{ij,t} + \frac{\Delta t^2}{2!} q^n_{ij,tt} + \cdots + \frac{\Delta t^M}{M!} \frac{\partial^M}{\partial t^M} q^n_{ij} = \sum_{m=0}^{M} \Delta t^m Q(0,0,m). \tag{6.16}$$

Step 4. Define the time averaged fluxes through numerical quadrature in time

$$F^n(x_i, y_j) \approx \sum_{m=1}^{M} \omega_m f\left(q_{ij}\left(t^n + c_m \Delta t\right)\right), \quad G^n(x_i, y_j) \approx \sum_{m=1}^{M} \omega_m g\left(q_{ij}\left(t^n + c_m \Delta t\right)\right), \tag{6.17}$$

where c_m and ω_m are the (rescaled) Gaussian quadrature points, and weights, respectively, on the interval $[0, 1]$. The function values are found by evaluating Eqn. (6.16) at each intermediate time point.

Step 5. Apply nonlinear WENO reconstruction to the resulting time averaged fluxes from Eqn. (6.17), and update the solution with Eqn. (6.14b).

This completes the description of the 2D Algorithm. Full details concerning the finite difference WENO reconstruction can be found in [26].

$mx = my$	**L2 error**	**Order**
5	1.14×10^{-01}	—
7	5.60×10^{-03}	7.43
11	9.30×10^{-05}	10.11
16	1.58×10^{-06}	10.05
25	1.68×10^{-08}	11.20
37	3.37×10^{-10}	9.64
56	5.85×10^{-12}	10.0
85	1.12×10^{-13}	9.75
128	5.22×10^{-14}	1.89

Table 6.3: Convergence of the 11th order differential transform method on the linear advection equation. All errors are computed using the L2 norm.

6.5.1 Numerical Results: 2D examples

Advection

Let us again start with the advection equation. In order to demonstrate high order convergence one may run a simple example:

$$q_t + q_x + q_y = 0$$

with initial conditions prescribed by $q_0(x, y) = \sin(2\pi x)\sin(2\pi y)$ on the domain $(x, y) \in [-1, 1] \times [-1, 1]$ (with periodic boundary conditions). Convergence results of an 11th order DT method have been recorded in Table 6.3. The convergence rate only briefly reaches the theoretical rate, but that is likely because rounding error is interfering with the convergence rate.

Euler equations-double Mach reflection

See Section 5.6.3 for an outline of how the differential transform method works for the Euler equations. Here the standard double Mach reflection example outlined in Section 4.8.2 is presented. Figure 6.2 collects the results of using a 9th order DT method on this problem on an 800×200 grid.

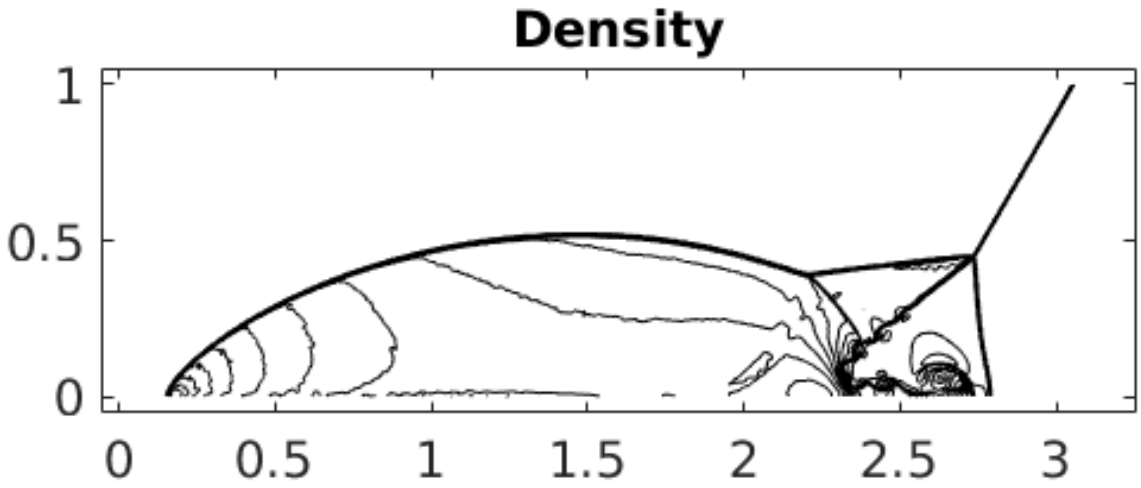


Figure 6.2: Double Mach reflection on a structured grid using a 9th version of the PIF-WENO scheme with differential transform based time-stepping. 30 equispaced contours are plotted ranging from 1.5 to 22.7.

Chapter 7

A POSITIVITY PRESERVING LIMITER FOR DG LAX-WENDROFF TIME-STEPPING SCHEMES

7.1 introduction

This chapter is based on a paper written with Rossmanith and Seal [87], in which we develop a positivity preserving limiter for Lax-Wendroff DG. In our numerical examples we use DG with hard coded high-order fluxes (not the differential transform method) since this work was published before the author had begun working on differential transform methods. However, the limiting scheme developed here is expected to perform equally well when DTs are used to calculate the high order accurate flux functions.

This chapter will focus on developing a positivity-preserving version of the Lax-Wendroff discontinuous Galerkin method for the compressible Euler equations. The actual Lax-Wendroff method used in this chapter is described in Section 7.2.

7.1.1 Positivity preservation

In simulations involving strong shocks, schemes that are higher than first-order applied to the compressible Euler equations generally create nonphysical undershoots (below zero) in the density and/or pressure. These undershoots cause catastrophic numerical instabilities due to a loss of hyperbolicity. These undershoots can occur even when a TVD or TVB limiter is applied to the solution. The chief goal of the limiting scheme developed in this chapter is to prevent positivity violations in density and pressure, and in particular, to accomplish this task with a high-order Lax-Wendroff time-stepping scheme.

In the DG literature, the most widely used strategy to maintain positivity was developed by Zhang and Shu in a series of influential papers [142, 143, 145]. The basic strategy of Zhang and Shu for a positivity-preserving RK-DG method can be summarized as follows:

Step 0. Write the current solution in the form

$$q^h\big|_{\mathcal{T}_i} = \overline{q}_i + \theta\left(q^h\big|_{\mathcal{T}_i} - \overline{q}_i\right), \tag{7.1}$$

where θ is yet-to-be-determined. $\theta = 1$ represents the unlimited solution.

Step 1. Find the largest value of θ, where $0 \leq \theta \leq 1$, such that $q^h\big|_{\mathcal{T}_i}$ satisfies the appropriate positivity conditions at some appropriately chosen quadrature points and limit the solution.

Step 2. Find the largest stable time-step that guarantees that, with a forward Euler time step, the cell average of the new solution remains positive.

Step 3. Rely on the fact that strong stability-preserving Runge-Kutta methods are convex combinations of forward Euler time steps; and therefore, the full method preserves the positivity of cell averages (under some slightly modified maximum allowable time-step).

This strategy is not adequate for a Lax-Wendroff time discretization. The strategy developed in this chapter will still contain an equivalent Step 1; however, in place of Step 2 and Step 3 above, the strategy will make use of a parameterized flux to maintain positive cell averages after taking a single time step. This strategy is sometimes called a flux corrected transport (FCT) scheme. This strategy has the advantage that it avoids introducing additional time step restrictions that often appear (e.g., in Step 2 above) when constructing a positivity-preserving scheme based on Runge-Kutta time stepping.

This idea of computing modified fluxes by combining a stable low-order flux with a less robust high-order flux is relatively old, and perhaps originates with Harten and Zwas and their *self adjusting hybrid scheme* [56]. The basic idea is the foundation of the related *flux corrected transport* (FCT) schemes of Boris, Book and collaborators [18, 17, 19, 16], where fluxes are adjusted in order to guarantee that average values of the unknown are constrained to lie within locally defined upper and lower bounds. This family of methods is used in an extensive variety of applications, ranging from seismology to meteorology [141, 74, 146, 132]. A thorough analysis of some of the early methods is conducted in [119]. Identical to modern maximum principle preserving (MPP) schemes, FCT can be formulated as a global optimization problem where a "worst case" scenario is assumed in order to decouple the previously coupled degrees of freedom [15]. Here FCT is used only in order to retain positivity of the density and pressure associated to $q^h(t^n, \vec{x})$; no local bounds will be enforced. This approach for obtaining positivity for the Euler equations borrows heavily from the finite difference WENO method developed in [137] for incompressible flows, as well as the extension of this approach to a discontinuous Galerkin scheme for scalar convection-diffusion equations [138].

To summarize, this limiting scheme draws on ideas from the two aforementioned families of techniques that are well established in the literature. First, it starts with the now well known (high-order) point-wise limiting developed for discontinuous Galerkin methods [142, 143, 145], and second, this is coupled this with the very large family of flux limiters [29, 114, 25]. (developed primarily for finite-difference (FD) and finite-volume (FV) schemes).

7.1.2 An outline of the proposed positivity-preserving method

The mathematics in this chapter will work with the compressible Euler equations as defined in Section 3.4.3. The basic positivity limiting strategy proposed in this chapter is summa-

rized below. Some important details are omitted here, but these details elaborated on in subsequent sections.

Step 0. On each element the solution is written as

$$q^h(t^n, \vec{x}(\vec{\xi}))\Big|_{\mathcal{T}_i} := \overline{q}_i^n + \theta \sum_{k=2}^{M_{\mathrm{L}}(M_{\mathrm{D}})} Q_i^{(k)}(t)\varphi^{(k)}(\vec{x}), \tag{7.2}$$

where θ is yet-to-be-determined. $\theta = 1$ represents the *unlimited* solution.

Step 1. Assume that this solution is positive in the mean. That is, assume for all i that $\overline{\rho}_i > 0$ and

$$\overline{p}_i := (\gamma - 1)\overline{\mathcal{E}}_i - \frac{1}{2}\frac{\|\overline{\mathbf{M}}_i\|^2}{\overline{\rho}_i} > 0. \tag{7.3}$$

A consequence of these assumptions is that $\overline{\mathcal{E}}_i > 0$.

Step 2. Find the largest value of θ, where $0 \leq \theta \leq 1$, such that the density and pressure are positive at some suitably defined quadrature points.

This step is elaborated upon in Section 7.3.2.

Step 3. Construct time-averaged fluxes through the Lax-Wendroff procedure. That is, we start with the exact definition of the time-average flux:

$$\overline{\mathbf{F}}^n(\vec{x}) := \frac{1}{\Delta t}\int_{t^n}^{t^{n+1}} \mathbf{F}(q(t, \vec{x}))\, dt = \frac{1}{\Delta t}\int_0^{\Delta t} \mathbf{F}(q(t^n + s, \vec{x}))\, ds \tag{7.4}$$

and approximate this via a Taylor series expansion around $s = 0$:

$$\mathbf{F}_T^n(\vec{x}) := \mathbf{F}(q(t^n, \vec{x})) + \frac{\Delta t}{2!}\frac{d\mathbf{F}}{dt}(q(t^n, \vec{x})) + \frac{\Delta t^2}{3!}\frac{d^2\mathbf{F}}{dt^2}(q(t^n, \vec{x})) = \overline{\mathbf{F}}^n(\vec{x}) + \mathcal{O}(\Delta t^3). \tag{7.5}$$

All time derivatives in this expression are replaced by spatial derivatives using the chain rule and the governing PDE (3.11). The approximate time-averaged flux (7.5) is first evaluated at some appropriately chosen set of quadrature points – in fact, the same quadrature points as used in Step 2 – and then, using appropriate quadrature weights, summed together to define a high-order flux, at both interior and boundary quadrature points.

Thanks to the moment limiting procedure in Step 2, all quantities of interest used to construct this expansion are positive at each quadrature point.

Step 4. Time step the solution so that cell averages are guaranteed to be positive. That is, we update the *cell averages* via a formula of the form

$$Q_i^{(1)n+1} = Q_i^{(1)n} - \frac{\Delta t}{|\mathcal{T}_i|} \sum_{e \in \mathcal{T}_i} \vec{F}_e^{h*} \cdot \vec{n}_e, \tag{7.6}$$

where $\vec{n}_e$ is an outward-pointing (relative to $\mathcal{T}_i$) normal vector to edge e with the property that $\|\vec{n}_e\|$ is the length of edge $e \in \mathcal{T}_i$, and the numerical flux on edge, $\vec{F}_e^{h*}$, is a convex combination of a high-order flux, $\mathcal{F}_e^H$, and a low-order flux $\mathcal{F}_e^L$:

$$\vec{F}_e^{h*} := \theta \mathcal{F}_e^{\mathrm{H}} + (1-\theta)\, \mathcal{F}_e^{\mathrm{L}}. \tag{7.7}$$

The low-order flux, $\mathcal{F}_e^{\mathrm{L}}$, is based on the (approximate) solution to the Riemann problem defined by cell averages only. The "high-order" flux, $\mathcal{F}_e^{\mathrm{H}}$, is constructed after integrating, via Gaussian quadrature, the (approximate) Riemann solutions at quadrature points along the edge $e \in \mathcal{T}_i$:

$$\mathcal{F}_e^{\mathrm{H}} = \frac{1}{2} \sum_{k=1}^{M_{\mathrm{Q}}} \omega_k \mathcal{F}_{ek}^{\mathrm{H}}. \tag{7.8}$$

Here ω_k are the Gaussian quadrature weights for quadrature with M_{Q} points and $\mathcal{F}_{ek}^{\mathrm{H}}$ are the numerical fluxes at each of the M_{Q} quadrature points. Note that this sum has only a single summand in the one-dimensional case. The selection of θ is described in more detail in Section 7.3.2. This step guarantees that the solution retains positivity (in the mean) for a single time step.

Step 5. Apply a shock-capturing limiter. The positivity-preserving limiter is designed to preserve positivity of the solution, but it fails at reducing spurious oscillations, and therefore a shock-capturing limiter needs to be added (see for example Shu [117]). There are many choices of limiters available; we use the limiter recently discussed in Chapter 4 and [86] because of its ability to retain genuine high-order accuracy, and its ability to push the polynomial order to arbitrary degree without modifying the overall scheme.

Step 6. Repeat all of these steps to update the solution for the next time step.

Each step of this process is elaborated upon throughout the remainder of this chapter. The end result is that the proposed method is the first scheme to simultaneously obtain all of the following properties:

- **High-order accuracy.** The proposed method is third-order in space and time, and can be extended to arbitrary order.

- **Positivity-preserving.** The proposed limiter is provably positivity-preserving for the density and pressure, at a finite set of point values, for the entire simulation.

- **Single-stage, single-step.** We use a Lax-Wendroff discretization for time stepping the PDE, and therefore we only need one communication per time step.

- **Unstructured meshes.** The proposed limiter can be applied on either structured or unstructutred meshes.

- **No additional time-step restrictions.** Because we do not rely on a SSP Runge-Kutta scheme, we do not have to introduce additional time-step restrictions to retain positivity of the solution. This differentiates us from popular positivity-preserving limiters based on RK time discretizations [143].

7.2 The Lax-Wendroff discontinuous Galerkin scheme

7.2.1 The base scheme: A method of modified fluxes

This section developes a Lax-Wendroff discontinuous Galerkin (LxW-DG) method based on the discussion of Lax-Wendroff time stepping methods in Section 5.2. Let us begin by formally integrating (5.1) over an interval $[t^n, t^{n+1}]$ to get

$$q(t^{n+1}, \vec{x}) = q(t^n, \vec{x}) - \Delta t \, \nabla \cdot \overline{\mathbf{F}}^n(\vec{x}), \tag{7.9}$$

where the *time-averaged* flux [26] is defined as

$$\overline{\mathbf{F}}^n(\vec{x}) := \frac{1}{\Delta t} \int_{t^n}^{t^{n+1}} \mathbf{F}(q(t, \vec{x})) \, dt. \tag{7.10}$$

Moreover, a Taylor expansion of $\mathbf{F}$ and a change of variables yields

$$\begin{aligned} \overline{\mathbf{F}}^n &= \frac{1}{\Delta t} \int_0^{\Delta t} \left(\mathbf{F}(q^n) + \tau \, \mathbf{F}(q^n)_t + \frac{1}{2} \tau^2 \, \mathbf{F}(q^n)_{tt} + \cdots \right) d\tau \\ &= \mathbf{F}(q^n) + \frac{1}{2!} \Delta t \, \mathbf{F}(q^n)_t + \frac{1}{3!} \Delta t^2 \, \mathbf{F}(q^n)_{tt} + \cdots , \end{aligned} \tag{7.11}$$

which can be inserted into (7.9). In a numerical discretization of (7.9), the Taylor series in (7.11) is truncated after a finite number of terms.

Remark 12. *If* $\overline{\mathbf{F}}^n = \mathbf{F}(q(t^n, \vec{x}))$*, then* (7.9) *reduces to a forward Euler time discretization for the hyperbolic conservation law* (5.1)*.*

This observation allows us to incorporate the positivity-preserving limiters that are presented in Section 7.3.1 and Section 7.3.2, because we view the LxW-DG method as a method of modified fluxes.

7.2.2 Construction of the time-averaged flux

Note that no specific method has been prescribed for computing time derivative terms. The numerical results in this chapter were computed using a traditional Lax-DG methodology, but these derivatives could be computed using a differential transform method. For completeness a description will now be provided of how the temporal derivative terms were actually computed in the numerical examples presented in this chapter:

The terms

$$\mathbf{F}(q^n)_t, \quad \mathbf{F}(q^n)_{tt}, \quad \mathbf{F}(q^n)_{ttt}, \quad \dots \tag{7.12}$$

are required to define the time-averaged flux in (7.11). A single application of the chain rule to compute the time derivative of the flux function yields

$$\frac{\partial \mathbf{F}}{\partial t} = \mathbf{F}'(q) \cdot q_t = -\mathbf{F}'(q) \cdot (\nabla \cdot \mathbf{F}), \tag{7.13}$$

where the flux Jacobian is

$$\mathbf{F}'(q)_{ij} := \frac{\partial \mathbf{F}_i}{\partial q_j}, \quad 1 \leq i, j \leq M. \tag{7.14}$$

The matrix-vector products in (7.13) can be compactly written using the Einstein summation convention (where repeated indices are assumed to be summed over), which produces a vector whose i^{th}-component is

$$\frac{\partial \mathbf{F}_i}{\partial t} = \frac{\partial \mathbf{F}_i}{\partial q_j}\frac{\partial q_j}{\partial t} = -\frac{\partial \mathbf{F}_i}{\partial q_j}\left(\nabla \cdot \mathbf{F}\right)_j. \tag{7.15}$$

A derivative of (7.13) yields

$$\frac{\partial^2}{\partial t^2}\mathbf{F} = \frac{\partial}{\partial t}\left(-\mathbf{F}'(q)\nabla \cdot \mathbf{F}\right) = \mathbf{F}''(q) \cdot (\nabla \cdot \mathbf{F}(q), \nabla \cdot \mathbf{F}(q)) + \mathbf{F}'(q) \cdot \nabla\left(\mathbf{F}'(q)\left(\nabla \cdot \mathbf{F}\right)\right), \tag{7.16}$$

where $\mathbf{F}''(q)$ is the Hessian with elements given by

$$\mathbf{F}_{ijk} := \frac{\partial^2 \mathbf{F}_i}{\partial q_j \partial q_k} = \left(\frac{\partial^2 f_i}{\partial q_j \partial q_k}, \frac{\partial^2 g_i}{\partial q_j \partial q_k}\right). \tag{7.17}$$

Equations (7.13) and (7.16) are generic formulae; the equalities are appropriate for any two-dimensional hyperbolic system, and similar identities exist for three dimensions. The first product in the right hand side of (7.16) is understood as a Hessian-vector product. Scripts that compute these derivatives, as well as the matrix, and Hessian vector products that are necessary to implement a third-order Lax-Wendroff scheme for multidimensional Euler equations can be found in the open source software FINESS [113].

Finally, these two time derivatives are sufficient to construct a third-order accurate method by defining the time-averaged flux through

$$\mathbf{F}_T^n(q) := \mathbf{F}(q^n) + \frac{1}{2!}\Delta t\, \mathbf{F}(q^n)_t + \frac{1}{3!}\Delta t^2\, \mathbf{F}(q^n)_{tt}, \tag{7.18}$$

and then updating the solution through

$$q^{n+1}(\vec{x}) = q^n(\vec{x}) - \Delta t \, \nabla \cdot \mathbf{F}_T^n \tag{7.19}$$

in place of (7.9).

7.2.3 Fully-discrete weak formulation

The final step is to construct a fully discrete version of (7.19). The LxW-DG scheme follows the following process [28, 52]:

Step 1. At each quadrature point evaluate the numerical flux, $\mathbf{F}(q^n)$, and then integrate this numerical flux against basis functions to obtain a Galerkin expansion of $\mathbf{F}^h$ inside each element.

Step 2. Using the Galerkin expansions of q^h and $\mathbf{F}^h$, evaluate all required spatial derivatives to construct the time expansion $\mathbf{F}_T^n$ in (7.18) at each quadrature point.

Step 3. Multiply (7.19) by a test function $\varphi^{(\ell)}$, integrate over a control element $\mathcal{T}_i$, and apply the divergence theorem to yield

$$\int_{\mathcal{T}_i} q^{n+1}\varphi^{(\ell)} d\mathbf{x} = \int_{\mathcal{T}_i} q^n \varphi^{(\ell)} d\mathbf{x} - \Delta t \int_{\mathcal{T}_i} \nabla\varphi^{(\ell)} \cdot \mathbf{F}_T(q^n)\, d\mathbf{x} + \Delta t \oint_{\partial\mathcal{T}_i} \varphi^{(\ell)}\, \mathbf{F}_T(q^n) \cdot \hat{\mathbf{n}}\, ds, \tag{7.20}$$

where $\hat{\mathbf{n}}$ is the outward pointing unit normal to element $\mathcal{T}_i$, which reduces to

$$Q_i^{(\ell)\,n+1} = Q_i^{(\ell)\,n} - \frac{\Delta t}{|\mathcal{T}_i|} \underbrace{\int_{\mathcal{T}_i} \nabla\varphi^{(\ell)} \cdot \mathbf{F}_T^h \, d\mathbf{x}}_{\text{Interior}} + \frac{\Delta t}{|\mathcal{T}_i|} \underbrace{\oint_{\partial\mathcal{T}_i} \varphi^{(\ell)}\, \mathbf{F}_T^{h*} \cdot \hat{\mathbf{n}}\, ds}_{\text{Edges}} \tag{7.21}$$

by orthogonality of the basis φ. In practice, both the interior and edge integrals are approximated by appropriate numerical quadrature rules. The flux values, $\mathbf{F}_T^{h*}$, in the edge integrals still need to be defined.

Step 4. Along each edge solve Riemann problems at each quadrature point by using the left and right interface values. In this chapter we use the well-known Lax-Friedrichs flux:

$$\mathbf{F}_T^{h*}\left(q_-^h,\, q_+^h\right) \cdot \hat{\mathbf{n}} = \frac{1}{2}\left[\hat{\mathbf{n}} \cdot \left(\mathbf{F}_T\left(q_+^h\right) + \mathbf{F}_T\left(q_-^h\right)\right) - s\left(q_+^h - q_-^h\right)\right], \tag{7.22}$$

where s is an estimate of the maximum global wave speed, q_-^h is the approximate solution evaluated on the element boundary on the interior side of $\mathcal{T}_i$, and q_+^h is the approximate solution evaluated on the element boundary on the exterior side of $\mathcal{T}_i$.

7.2.4 Boundary conditions

In order to achieve high-order accuracy at the boundaries of the computational domain, a careful treatment of the solution in each boundary element is required. In particular, all simulations in this chapter require either reflective (hard surface) or transparent (outflow) boundary conditions.

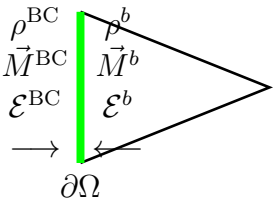


Figure 7.1: Interior, q^b, and exterior, q^{BC}, solution values on either side of the boundary $\partial\Omega$.

Suppose the solution takes on the value

$$q^b = \left(\rho^b,\, \vec{M}^b,\, \mathcal{E}^b\right) \tag{7.23}$$

at a quadrature point $\mathbf{x}^b$ on the boundary $\partial\Omega$, and we wish to define a boundary value, q^{BC}, on the exterior side of the boundary $\partial\Omega$ that yields a flux with one of two desired boundary conditions. This is depicted in Figure 7.1. Let $\hat{\mathbf{t}}$ and $\hat{\mathbf{n}}$ be the unit tangent and unit normal vectors to the boundary at the boundary point $\mathbf{x}^b$, respectively. In both the reflective and transparent boundary conditions, we enforce continuity of the tangential components:

$$\vec{M}^{\mathrm{BC}} \cdot \hat{\mathbf{t}} = \vec{M}^b \cdot \hat{\mathbf{t}}. \tag{7.24}$$

The only difference between the two types of boundary conditions we consider lie in the normal direction. We set

$$\vec{M}^{\mathrm{BC}} \cdot \hat{\mathbf{n}} = \mp \vec{M}^b \cdot \hat{\mathbf{n}}, \tag{7.25}$$

where the minus sign corresponds to the reflective boundary condition and the plus sign corresponds to the transparent boundary condition.

From this we can easily write out the full boundary conditions at the point $\mathbf{x}^{\mathrm{BC}}$:

$$\begin{pmatrix} \rho^{\mathrm{BC}} \\ \vec{M}^{\mathrm{BC}} \\ \mathcal{E}^{\mathrm{BC}} \end{pmatrix} = \begin{pmatrix} \rho^b \\ \left(\vec{M}^b \cdot \hat{\mathbf{t}}\right) \hat{\mathbf{t}} \mp \left(\vec{M}^b \cdot \hat{\mathbf{n}}\right) \hat{\mathbf{n}} \\ \mathcal{E}^b \end{pmatrix}, \tag{7.26}$$

where again the minus sign corresponds to the reflective boundary condition and the plus sign corresponds to the transparent boundary condition.

In order to achieve high-order time accuracy we need to apply the above boundary conditions to the time derivatives on the boundary:

$$\begin{pmatrix} \rho_t^{\mathrm{BC}} \\ \vec{M}_t^{\mathrm{BC}} \\ \mathcal{E}_t^{\mathrm{BC}} \end{pmatrix} = \begin{pmatrix} \rho_t^b \\ \left(\vec{M}_t^b \cdot \hat{\mathbf{t}}\right)\hat{\mathbf{t}} \mp \left(\vec{M}_t^b \cdot \hat{\mathbf{n}}\right)\hat{\mathbf{n}} \\ \mathcal{E}_t^b \end{pmatrix}, \quad \begin{pmatrix} \rho_{tt}^{\mathrm{BC}} \\ \vec{M}_{tt}^{\mathrm{BC}} \\ \mathcal{E}_{tt}^{\mathrm{BC}} \end{pmatrix} = \begin{pmatrix} \rho_{tt}^b \\ \left(\vec{M}_{tt}^b \cdot \hat{\mathbf{t}}\right)\hat{\mathbf{t}} \mp \left(\vec{M}_{tt}^b \cdot \hat{\mathbf{n}}\right)\hat{\mathbf{n}} \\ \mathcal{E}_{tt}^b \end{pmatrix}, \tag{7.27}$$

where all time derivatives must be replaced by spatial derivatives using the PDE.

7.3 Positivity preservation

The temporal evolution described in the previous section fails to retain positivity of the solution, even in the simple case of linear advection with smooth solutions that are near zero. In fact, in extreme cases the projection of the initial conditions can fail to retain positivity of the solution due to the Gibbs phenomenon. The positivity-preserving limiter we present follows a two step procedure:

Step 1. Limit the moments in the expansion so that the solution is positive at each quadrature point.

Step 2. Limit the fluxes so that the cell averages retain positivity after a single time step.

We now describe the first of these two steps.

7.3.1 Positivity at interior quadrature points via moment limiters

This section describes a procedure that implements the following: if the cell averages are positive, then the solution is forced to be positive at a preselected and finite collection of quadrature points. We select only the quadrature points that are actually used in the numerical update; this includes internal Gauss quadrature points as well as face/edge Gauss quadrature points. Unlike other positivity limiting schemes [145, 142] in the SSP Runge-Kutta framework, this step is not strictly necessary to guarantee positivity of the cell average at the next time-step; however, the main reason for applying the limiter at quadrature points is to guarantee that each term in the update is physical, which will reduce the total amount of additional limiting of the cell average updated needed in Section 7.3.2. The process to maintain positivity at quadrature points is carried out in a series of three simple steps. Because this part of the limiter is entirely local, we focus on a single element $\mathcal{T}_k$, and therefore drop the subscript for ease of notation.

Step 0: Assume positivity of the cell averages. We assume that the cell averages for the density satisfies $\overline{\rho}^n \geq \epsilon_0$, where $\epsilon_0 > 0$ is a cutoff parameter that guarantees hyperbolicity of the system. In this chapter, we set $\epsilon_0 = 10^{-12}$ in all simulations. Furthermore, we assume

that the cell average for the pressure satisfies $\overline{p}^n > 0$, where the (average) pressure $\overline{p}^n$ is defined through the averages of the other conserved quantities in (7.3). Note that these two conditions are sufficient to imply that the average energy density $\overline{\mathcal{E}}^n$ is positive.

Step 1: Enforce positivity of the density.

In this step, we enforce positivity of the density at each quadrature point $\mathbf{x}_m \in \mathcal{T}_k$. Because $\overline{\rho}^n \geq \epsilon_0$, there exists a (maximal) value $\theta^\rho \in [0, 1]$ such that

$$\rho_m^\theta := \overline{\rho}^n + \theta \sum_{\ell=2}^{M_{\mathrm{L}}(M_{\mathrm{D}})} \rho^{(\ell)n} \, \varphi^{(\ell)}(\mathbf{x}_m) \geq \epsilon_0 \tag{7.28}$$

for all quadrature points $\mathbf{x}_m \in \mathcal{T}_k$ and all $\theta \in [0, \theta^\rho]$. Note that we drop the subscript that indicates the element number to ease the complexity of notation for the ensuing discussion.

Step 2: Enforce positivity of the pressure.

Recall that the pressure is defined through the relation (3.12). We seek to guarantee that $p(\mathbf{x}_m) > 0$ for each quadrature point $\mathbf{x}_m \in \mathcal{T}_i$. In place of working directly with the pressure, we observe that it suffices to guarantee that the product of the density and pressure is positive. To this end, we expand the momentum and energy density in the same free parameter θ. Similar to the density in (7.28), we write the momentum and energy density as

$$\vec{M}_m^\theta := \overline{\vec{M}}^n + \theta \sum_{\ell=2}^{M_{\mathrm{L}}(M_{\mathrm{D}})} \vec{M}^{(\ell)n} \, \varphi^{(\ell)}(\mathbf{x}_m) \quad \text{and} \quad \mathcal{E}_m^n := \overline{\mathcal{E}}^n + \theta \sum_{\ell=2}^{M_{\mathrm{L}}(M_{\mathrm{D}})} \mathcal{E}^{(\ell)n} \, \varphi^{(\ell)}(\mathbf{x}_m). \tag{7.29}$$

We define deviations from cell averages as

$$\left(\widetilde{\rho}_m^n, \, \widetilde{\vec{M}}_m, \, \widetilde{\mathcal{E}}_m^n \right) := \sum_{\ell=2}^{M_{\mathrm{L}}(M_{\mathrm{D}})} \left(\rho^{(\ell)n}, \, \vec{M}^{(\ell)n}, \, \mathcal{E}^{(\ell)n} \right) \varphi^{(\ell)}(\mathbf{x}_m), \tag{7.30}$$

and compactly write the expression for the limited variables as the cell average plus deviations:

$$q_m^\theta := \overline{q} + \theta \widetilde{q}_m, \quad \text{where} \quad \overline{q} := \left(\overline{\rho}^n, \, \overline{\vec{M}}^n, \, \overline{\mathcal{E}}^n \right) \quad \text{and} \quad \widetilde{q}_m = \left(\widetilde{\rho}_m^n, \, \widetilde{\vec{M}}_m^n, \, \widetilde{\mathcal{E}}_m^n \right). \tag{7.31}$$

The product of the density and pressure at each quadrature point is a quadratic function of θ:

$$(\rho p)_m^\theta := \rho_m^\theta \, p_m^\theta = (\gamma - 1) \left(\mathcal{E}_m^\theta \rho_m^\theta - \frac{1}{2} \| \vec{M}_m^\theta \|^2 \right), \tag{7.32}$$

where ρ_m^θ is defined in (7.28). After expanding each of these conserved variables in the *same*

scaling parameter θ, we observe that

$$\begin{aligned}(\rho p)_m^\theta &= (\gamma-1)\left(\mathcal{E}_m^\theta \rho_m^\theta - \frac{1}{2}\|\vec{M}_m^\theta\|^2\right) \\ &= (\gamma-1)\left[a_m\theta^2 + b_m\theta + \left(\underbrace{\overline{\mathcal{E}}^n\overline{\rho}^n - \frac{1}{2}\|\overline{\vec{M}}^n\|^2}_{>0}\right)\right],\end{aligned} \tag{7.33}$$

where a_m and b_m depend only on the quadrature point and higher-order terms of the expansions of density, energy density, and momentum:

$$a_m = \widetilde{\mathcal{E}}_m^n\,\widetilde{\rho}_m^n - \frac{1}{2}\|\widetilde{\vec{M}}_m^n\|^2 \quad \text{and} \quad b_m = \widetilde{\mathcal{E}}_m^n\,\overline{\rho}_m^n + \overline{\mathcal{E}}_m^n\,\widetilde{\rho}_m^n - \widetilde{\vec{M}}_m^n \cdot \overline{\vec{M}}^n. \tag{7.34}$$

The quadratic function defined by (7.33) is non-negative for at least one value of θ, namely $\theta = 0$. However, if (7.33) is positive at $\theta = 0$, then we are guaranteed that there exists a $\theta_m \in (0,1]$ that guarantees $(\rho p)_m^\theta \geq 0$ for all $\theta \in [0,\theta_m]$. In particular, we are interested in finding the largest such θ (i.e., the least amount of damping).

Instead of exactly computing the optimal θ, which could readily be done, but would require additional floating point operations, we make use of the following lemma [114] to find an approximately optimal θ.

Lemma 1. *The pressure function is a convex function of* θ *on* $[0,\theta^\rho]$*. That is,*

$$p_m^{\alpha\theta_1+(1-\alpha)\theta_2} \geq \alpha p_m^{\theta_1} + (1-\alpha)p_m^{\theta_2} \tag{7.35}$$

for all $\theta_1,\theta_2 \in [0,\theta^\rho]$*, and* $\alpha \in [0,1]$*.*

Proof. We observe that directly from the definition of the limiter for the conserved variables in (7.31) that

$$q_m^{\alpha\theta_1+(1-\alpha)\theta_2} = \alpha q_m^{\theta_1} + (1-\alpha)q_m^{\theta_2}, \tag{7.36}$$

and therefore

$$p_m^{\alpha\theta_1+(1-\alpha)\theta_2} = p\left(q_m^{\alpha\theta_1+(1-\alpha)\theta_2}\right) = p\left(\alpha q_m^{\theta_1} + (1-\alpha)q_m^{\theta_2}\right) \geq \alpha p_m^{\theta_1} + (1-\alpha)p_m^{\theta_2}. \tag{7.37}$$

The final inequality follows because $\rho_m^\theta > 0$ for all $0 \leq \theta \leq \theta^\rho$, and the pressure is a convex function (of the conserved variables) whenever the density is positive. □

As a consequence of (7.35) we can define

$$\theta_m := \min\left(\frac{p_m^0}{p_m^0 - p_m^{\theta^\rho}},\, \theta_m^\rho\right), \tag{7.38}$$

which will guarantee that $p_m^\theta > 0$ for all $\theta \in [0, \theta_m]$. Finally, we define the scaling parameter for the entire cell as

$$\theta := \min_m \{\theta_m\} \tag{7.39}$$

and use this value to limit the higher order coefficients in the Galerkin expansions of the density, momentum, and energy density displayed in Eqns. (7.28) and (7.29).

This definition gives us the property that $\rho_m^n \geq \epsilon_0$ and $p_m^n > 0$ at each quadrature point $\mathbf{x}_m \in \mathcal{T}_i$. This process is repeated (locally) in each element $\mathcal{T}_i$ in the mesh. As a side benefit to guaranteeing that the density and pressure are positive, we have the following remark.

Remark 13. *If ρ_m^θ and p_m^θ are positive at each quadrature point, then $\mathcal{E}_m^\theta$ is also positive at each quadrature point.*

Proof. Divide (7.33) by $(\gamma - 1)\rho_m^\theta$ and add $\frac{1}{2}\|\vec{M}_m^\theta\|^2$ to both sides. □

This concludes the first of two steps for retaining positivity of the solution. We now move on to the second and final step, which takes into account the temporal evolution of the solver.

7.3.2 Positivity of cell averages via parameterized flux limiters

The procedure carried out for the flux limiter presented in this section is very similar to recent work for finite volume [29] as well as finite difference [114, 25] methods. When compared to the finite difference methods, the main difference in this discussion is that the expressions do not simplify as much because quantities such as the edge lengths must remain in the expressions. This makes them more similar to work on finite volume schemes [29]. Overall however, there is little difference between flux limiters on Cartesian and unstructured meshes, and between flux limiters for finite difference, finite volume (FV), and discontinuous Galerkin (DG) schemes. This is because the updates for the cell average in a DG solver can be made to look identical to the update for a FV solver, and once flux interface values are identified, a conservative FD method can be made to look like a FV solver, albeit with a different stencil for the discretization.

All of the aforementioned papers rely on the result of Perthame and Shu [94], which states that a first-order finite volume scheme (i.e., one that is based on a piecewise constant representation with forward Euler time-stepping) that uses the Lax-Friedrichs (LxF) numerical flux is positivity-preserving under the usual CFL condition. Similar to previous work, we leverage this idea and incorporate it into a flux limiting procedure. Here, the focus is on Lax-Wendroff discontinuous Galerkin schemes.

In this chapter we write out the details of the limiting procedure only for the case of 2D triangular elements. However, all of the formulas generalize to higher dimensions and Cartesian meshes.

To begin, we consider the Euler equations as defined in Section 4.5. After integration over a single cell, $\mathcal{T}_i$, and an application of the divergence theorem, we see that the exact evolution equation for the cell average of the density is given by

$$\frac{d}{dt}\int_{\mathcal{T}_i} q\, d\mathbf{x} = -\oint_{\partial\mathcal{T}_i} \vec{F}\cdot\hat{\mathbf{n}}\, ds, \tag{7.40}$$

where $\hat{\mathbf{n}}$ is the outward pointing (relative to $\mathcal{T}_i$) unit normal to the boundary of $\mathcal{T}_i$. Applying to this equation a first-order finite volume discretization using the Lax-Friedrichs flux yields

$$\overline{q}_i^{n+1} = \overline{q}_i^n - \frac{\Delta t}{|\mathcal{T}_i|}\sum_{e\in\mathcal{T}_i} f_e^{\mathrm{LxF}}, \tag{7.41}$$

and the Lax-Friedrichs flux is

$$f_e^{\mathrm{LxF}} := \frac{1}{2}\vec{n}_e\cdot\left(\vec{F}\left(\overline{q}_{e+}^n\right) + \vec{F}\left(\overline{q}_i^n\right)\right) - \frac{1}{2}\|\vec{n}_e\| s\left(\overline{q}_{e+}^n - \overline{q}_i^n\right), \tag{7.42}$$

where $\vec{n}_e$ is an outward-pointing (relative to $\mathcal{T}_i$) normal vector to edge e with the property that $\|\vec{n}_e\|$ is the length of edge $e\in\mathcal{T}_i$, and the $e+$ index refers to the solution on edge e on the exterior side of $\mathcal{T}_i$. We use a *global* wave speed s, because this flux defines a provably positivity-preserving scheme [94]. Other fluxes can be used, provided they are positivity-preserving (in the mean).

Recall that the update for the LxW-DG method is given in (7.21). The numerical edge flux, $\mathbf{F}_T^{h*}$, is based on the temporal Taylor series expansion of the fluxes. The update for the cell averages takes the form:

$$\overline{q}_i^{n+1} = \overline{q}_i^n - \frac{\Delta t}{|\mathcal{T}_i|}\oint_{\partial\mathcal{T}_i} \mathbf{F}_T^{h*}\cdot\vec{\hat{n}}\, ds, \tag{7.43}$$

which in practice needs to be replaced by a numerical quadrature along the edges. Applying Gaussian quadrature along each edge produces the following edge value (or face value in the case of 3D):

$$f_e^{\mathrm{LxW}} := \frac{1}{2}\sum_{k=1}^{M_{\mathrm{Q}}} \omega_k\, \mathbf{F}_T^{h*}(\mathbf{x}_k)\cdot\mathbf{n}_e, \tag{7.44}$$

where $\mathbf{x}_k$ and ω_k are Gaussian quadrature points and weights, respectively, for integration along edge e.

This allows us to write the update for the cell average in the Lax-Wendroff DG method in a similar fashion to that of the Lax-Friedrichs solver, but this time with higher-order fluxes:

$$\overline{q}_i^{n+1} = \overline{q}_i^n - \frac{\Delta t}{|\mathcal{T}_i|}\sum_{e\in\mathcal{T}_i} f_e^{\mathrm{LxW}}. \tag{7.45}$$

Next define a parameterized flux on edge e by

$$\widetilde{f}_e := \theta_e \left(f_e^{\text{LxW}} - f_e^{\text{LxF}} \right) + f_e^{\text{LxF}}, \tag{7.46}$$

where $\theta_e \in [0,1]$ is as free parameter that is yet to be determined. We also define the quantity, Γ_i, which by virtue of the positivity of the Lax-Friedrichs method is positive in both density and pressure:

$$\overline{q}_i^{n+1} = \frac{\Delta t}{|\mathcal{T}_i|}\Gamma_i, \quad \text{where} \quad \Gamma_i := \frac{|\mathcal{T}_i|}{\Delta t}\overline{q}_i^n - \sum_{e\in\mathcal{T}_i} f_e^{\text{LxF}}. \tag{7.47}$$

In order to retain a positive density in the high-order update formula, the following condition must be satisfied:

$$\overline{\rho}_i^{n+1} = \overline{\rho}_i^n - \frac{\Delta t}{|\mathcal{T}_i|}\sum_{e\in\mathcal{T}_i} \tilde{f}_e^{\rho} \geq 0 \quad \Longrightarrow \quad \overline{\rho}_i^{n+1} = \frac{\Delta t}{|\mathcal{T}_i|}\left(\Gamma_i^{\rho} + \sum_{e\in\mathcal{T}_i} \theta_e \Delta f_e^{\rho} \right) \geq 0, \tag{7.48}$$

where

$$\Delta f_e := f_e^{\text{LxF}} - f_e^{\text{LxW}}. \tag{7.49}$$

Note that a ρ superscript is introduced to the flux function in order to denote the first component of the flux, namely the mass flux. Positivity of the density is achieved if

$$\sum_{e\in\mathcal{T}_i} \theta_e \Delta f_e^{\rho} \geq -\Gamma_i^{\rho}. \tag{7.50}$$

The basic procedure for positivity limiting is to reduce the values of θ_e until inequality condition (7.50) is satisfied. For a triangular mesh, there are three values of θ_e that contribute to each cell. In this case there exists a three-dimensional feasible region that contains all admissible θ_e values, one for each $e \in \mathcal{T}_i$, where the new average density, $\overline{\rho}_i^{n+1}$, is positive. Finding the exact boundary of this set is computationally impractical; and therefore, we make an approximation so that the problem becomes much simpler. In particular, we approximate the feasible region by a rectangular cuboid:

$$S_i^{\rho} := [0, \Lambda_{e_{i1}}] \times [0, \Lambda_{e_{i2}}] \times [0, \Lambda_{e_{i3}}] \subseteq [0,1]^3, \tag{7.51}$$

where e_{i1}, e_{i2}, and e_{i3} are the three edges that make up element $\mathcal{T}_i$, over which

$$\overline{\rho}_i^{n+1} = \overline{\rho}_i^n - \frac{\Delta t}{|\mathcal{T}_i|}\left(\widetilde{f}_{e_{i1}}^{\rho} + \widetilde{f}_{e_{i2}}^{\rho} + \widetilde{f}_{e_{i3}}^{\rho} \right) \geq 0, \quad \forall\, (\theta_{e_{i1}}, \theta_{e_{i2}}, \theta_{e_{i3}}) \in S_i^{\rho}. \tag{7.52}$$

Once we have determined the feasible region for each element over which the average density, $\overline{\rho}_i^{n+1}$, remains positive, the next step is to rescale each $\Lambda_{e_{ik}}$ for $k = 1,2,3$ to also guarantee a positive average pressure, $\overline{p}_i^{n+1}$, on the same element:

$$S_i^{\rho p} := [0, \mu_{e_{i1}} \Lambda_{e_{i1}}] \times [0, \mu_{e_{i2}} \Lambda_{e_{i2}}] \times [0, \mu_{e_{i3}} \Lambda_{e_{i3}}] \subseteq [0,1]^3, \tag{7.53}$$

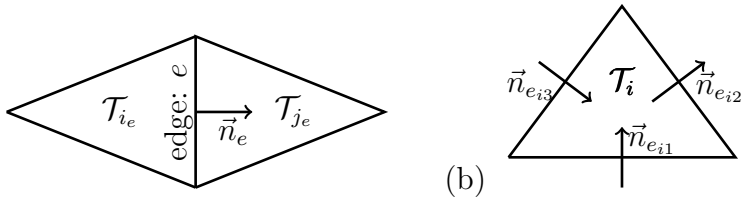


Figure 7.2: Illustrations of various quantities needed in the discontinuous Galerkin update on unstructured grids. Panel (a) illustrates the normal vector $\vec{n}_e$ to the edge e and the two elements i_e (on the outward side of $\vec{n}_e$) and j_e (on the inward side of $\vec{n}_e$). Panel (b) illustrates the three normal vectors $\vec{n}_{e_{i1}}$, $\vec{n}_{e_{i2}}$, and $\vec{n}_{e_{i3}}$ to the three edges of element $\mathcal{T}_i$. In the case shown in Panel (b), the values of ϵ_{ik} as defined in Equation (7.56) are $\epsilon_{i1} = -1$, $\epsilon_{i2} = 1$, and $\epsilon_{i3} = -1$.

where $0 \leq \mu_{e_{ik}} \leq 1$ for each i and for each $k = 1, 2, 3$.

We leave the details of the procedure to determine the feasibility region to the next subsection, in which we also summarize the full algorithm.

7.3.3 Putting it all together: An efficient implementation of the positivity-preserving limiter

An efficient implementation of the positivity preserving limiter should avoid communication with neighboring cells as much as possible. Indeed, this is one advantage of single-stage, single-step methods such as the Lax-Wendroff discontinuous Galerkin method. In order to avoid additional communication overhead, we suggest the implementation described below.

In the formulas below we make use of the following quantities:

$$\mathbf{n}_e := \text{normal vector to edge } e \text{ such that } \|\mathbf{n}_e\| \text{ is equal to the length of edge } e, \tag{7.54}$$

$$e_{ik} := \text{label of } k^{\text{th}} \text{ edge } (k = 1, 2, 3) \text{ of element } \mathcal{T}_i, \tag{7.55}$$

$$\epsilon_{ik} := \begin{cases} +1 & \text{if } \mathbf{n}_{e_{ik}} \text{ is outward pointing relative to } \mathcal{T}_i, \\ -1 & \text{if } \mathbf{n}_{e_{ik}} \text{ is inward pointing relative to } \mathcal{T}_i, \end{cases} \tag{7.56}$$

$$i_e := \text{the element that has } e \text{ as an edge and for which } \mathbf{n}_e \text{ is outward pointing}, \tag{7.57}$$

$$j_e := \text{the element that has } e \text{ as an edge and for which } \mathbf{n}_e \text{ is inward pointing}. \tag{7.58}$$

These quantities are illustrated in Figure 7.2.

Loop over all elements, $i = 1, 2, \ldots, M_{\text{elems}}$:

Step 1. Enforce positivity of the density and pressure at internal and boundary quadrature points using the limiter described in detail in Section 7.3.1.

Step 2. Compute the time-averaged fluxes, $\mathbf{F}_T^n$, defined in Equation (7.18).

Loop over all edges, $e = 1, 2, \ldots, M_{\text{edges}}$:

Step 3. For each quadrature point, $\mathbf{x}_\ell$, on the current edge compute the high-order numerical flux:

$$\mathbf{F}_T^{h*}(\mathbf{x}_\ell) \cdot \mathbf{n}_e := \frac{\mathbf{n}_e}{2} \cdot \left[\mathbf{F}_T\big(q^h(\mathbf{x}_k^+)\big) + \mathbf{F}_T\big(q^h(\mathbf{x}_k^-)\big)\right] - \frac{s\|\mathbf{n}_e\|}{2}\big(q^h(\mathbf{x}_k^+) - q^h(\mathbf{x}_k^-)\big)\,, \tag{7.59}$$

where s is an estimate of the maximum global wave speed. From these numerical flux values, compute the edge-averaged high-order flux:

$$f_e^{\text{LxW}} := \frac{1}{2}\sum_{\ell=1}^{M_{\text{Q}}} \omega_\ell\, \mathbf{F}_T^{h*}(\mathbf{x}_\ell) \cdot \mathbf{n}_e, \tag{7.60}$$

where ω_k are the weights in the Gauss-Legendre numerical quadrature with M_{Q} points. Still on the same edge, also compute the edge-averaged low-order flux:

$$f_e^{\text{LxF}} := \frac{\vec{n}_e}{2} \cdot \left[\vec{F}\left(\overline{q}_{j_e}^h\right) + \vec{F}\left(\overline{q}_{i_e}^h\right)\right] - \frac{s\|\vec{n}_e\|}{2}\left(\overline{q}_{j_e}^h - \overline{q}_{i_e}^h\right). \tag{7.61}$$

Finally, set

$$\Lambda_e = 1. \tag{7.62}$$

Loop over all elements, $i = 1, 2, \ldots, M_{\text{elems}}$:

Step 4. Let $\Delta f_k = \epsilon_{ik}\left(f_{e_{ik}}^{\text{LxF}} - f_{e_{ik}}^{\text{LxW}}\right)$ for $k = 1, 2, 3$ represent the high-order flux contribution on the three edges of element $\mathcal{T}_i$. We choose a reordering of the indices 1, 2, 3 to the indices a, b, c such that the three flux differences are ordered as follows:

$$\Delta f_a^\rho \le \Delta f_b^\rho \le \Delta f_c^\rho. \tag{7.63}$$

We now define the three values Λ_{ia}, Λ_{ib}, and Λ_{ic} and consider four distinct cases.

Case 1. If $0 \le \Delta f_a^\rho \le \Delta f_b^\rho \le \Delta f_c^\rho$, then

$$\Lambda_{ia} = \Lambda_{ib} = \Lambda_{ic} = 1. \tag{7.64}$$

Case 2. If $\Delta f_a^\rho < 0 \le \Delta f_b^\rho \le \Delta f_c^\rho$, then set

$$\Lambda_{ia} = \min\left\{1, \frac{\Gamma_i}{|\Delta f_a^\rho|}\right\}, \quad \Lambda_{ib} = \Lambda_{ic} = 1. \tag{7.65}$$

Case 3. If $\Delta f_a^\rho \le \Delta f_b^\rho < 0 \le \Delta f_c^\rho$, then set

$$\Lambda_{ia} = \Lambda_{ib} = \min\left\{1, \frac{\Gamma_i}{|\Delta f_a^\rho + \Delta f_b^\rho|}\right\}, \quad \Lambda_{ic} = 1. \tag{7.66}$$

Case 4. If $\Delta f_a^\rho \le \Delta f_b^\rho \le \Delta f_c^\rho < 0$, then set

$$\Lambda_{ia} = \Lambda_{ib} = \Lambda_{ic} = \min\left\{1, \frac{\Gamma_i}{|\Delta f_a^\rho + \Delta f_b^\rho + \Delta f_c^\rho|}\right\}. \tag{7.67}$$

Note that in each of these cases the ratios used in the relevant formulas are found by setting the positive contributions on the left-hand side of inequality (7.50) equal to zero, and then solving for the remaining elements. This is equivalent to only looking at the worst case scenario where mass is only allowed to flow out of element $\mathcal{T}_i$.

Step 5. Define the Lax-Friedrichs average solution:

$$\overline{q}_i^{\,\mathrm{LxF}} := \overline{q}_i^n - \frac{\Delta t}{|\mathcal{T}_i|}\sum_{k=1}^{3} f_k^{\mathrm{LxF}}, \tag{7.68}$$

and do the following.

(a) Loop over all seven cases, $c = 111, 110, 101, 100, 011, 010, 001$, shown in Figure 7.3 and for each case construct the average solution:

$$\overline{q}_i^{\,c} := \overline{q}_i^n - \frac{\Delta t}{|\mathcal{T}_i|}\sum_{k=1}^{3} \alpha_k \Lambda_{ik} f_k^{\mathrm{LxW}}. \tag{7.69}$$

The seven cases studied here enumerate all of the possible values of $\alpha_k \in \{0, 1\}$, with the exception of $c = 000$, which reduces to updating the element with purely a Lax-Friedrichs flux.

(b) For each c, determine the largest value of $\mu_c \in [0, 1]$ such that the average pressure as defined by (7.3) and based on the average state:

$$\mu_c\, \overline{q}_i^{\,c} + (1 - \mu_c)\, \overline{q}_i^{\,\mathrm{LxF}} \tag{7.70}$$

is positive. Note that the average pressure is always positive when $\mu_c = 0$, and if the average pressure is positive for $0 \le r \le 1$, then the average pressure will be positive for any $0 \le \mu_c \le r$. This is because the pressure is a convex function of μ_c.

(c) Rescale the edge Λ values (when compared to the neighboring elements) based on μ_c as follows:

$$\Lambda_{e_{i1}} = \min\left\{\Lambda_{e_{i1}},\, \Lambda_{i1}\cdot\min\Big\{\mu_{111},\, \mu_{110},\, \mu_{101},\, \mu_{100}\Big\}\right\}, \tag{7.71}$$

$$\Lambda_{e_{i2}} = \min\left\{\Lambda_{e_{i2}},\, \Lambda_{i2}\cdot\min\Big\{\mu_{111},\, \mu_{110},\, \mu_{011},\, \mu_{010}\Big\}\right\}, \tag{7.72}$$

$$\Lambda_{e_{i3}} = \min\left\{\Lambda_{e_{i3}},\, \Lambda_{i3}\cdot\min\Big\{\mu_{111},\, \mu_{101},\, \mu_{011},\, \mu_{001}\Big\}\right\}. \tag{7.73}$$

Loop over all elements, $i = 1, 2, \ldots, M_{\text{elems}}$:

Step 6. For each of the three edges that make up element $\mathcal{T}_i$ determine the damping coefficients, θ_k for $k = 1, 2, 3$, as follows:

$$\theta_k = \Lambda_{e_{ik}} \quad \text{for} \quad k = 1, 2, 3. \tag{7.74}$$

Update the cell averages:

$$Q_i^{(1)\,n+1} = Q_i^{(1)\,n} - \frac{\Delta t}{|\mathcal{T}_i|}\sum_{k=1}^{3}\Big[\theta_k\, f_{e_{ik}}^{\text{LxW}} + (1-\theta_k)\; f_{e_{ik}}^{\text{LxF}}\Big], \tag{7.75}$$

as well as the high-order moments

$$Q_i^{(\ell)\,n+1} = Q_i^{(\ell)\,n} - \frac{\Delta t}{|\mathcal{T}_i|}\int_{\mathcal{T}_i}\nabla\varphi^{(\ell)}\cdot\mathbf{F}_T^h\,d\mathbf{x} + \frac{\Delta t}{|\mathcal{T}_i|}\oint_{\partial\mathcal{T}_i}\varphi^{(\ell)}\,\mathbf{F}_T^{h*}\cdot\hat{\mathbf{n}}\,ds \tag{7.76}$$

for $2 \le \ell \le M_{\text{L}}$, where exact integration is replaced by numerical quadrature.

Remark 14. *Extensions to 2D Cartesian, 3D Cartesian, and 3D tetrahedral mesh elements follow directly from what is presented here, and require considering flux values along each of the edges/faces of a given element.*

7.4 Numerical results

7.4.1 Implementation details

All of the results presented in this section are implemented in the open-source software package DoGPack [105]. In addition, the positivity limiter described thus far is not designed to handle shocks, and therefore an additional limiter needs to be applied in order to prevent spurious oscillations from developing (e.g., in problems that contain shocks but have large densities). There are many options available for this step, but in this chapter, we supplement the positivity-preserving limiter presented here with the shock-capturing limiter discussed in

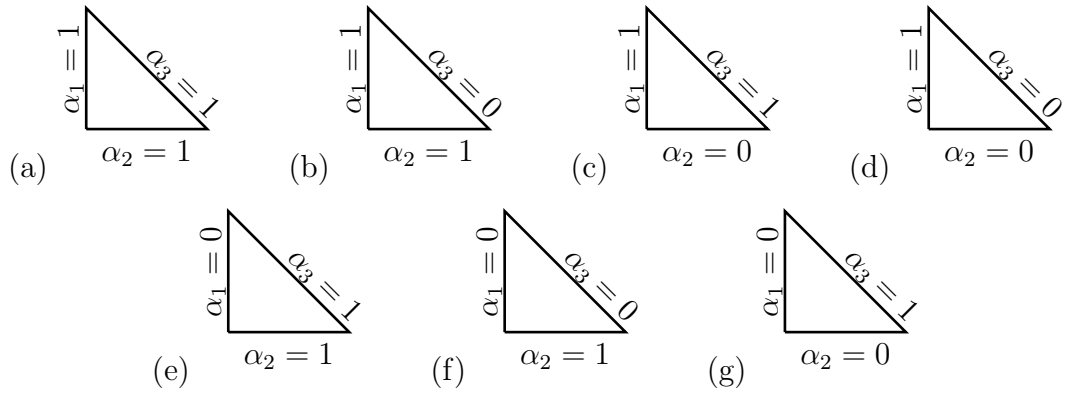


Figure 7.3: Seven cases used to enforce the positivity of the average pressure on each element: (a) 111, (b) 110, (c) 101, (d) 100, (e) 011, (f) 010, and (g) 001.

Chapter 4 and [86] in order to navigate shocks that develop in the solution. Specifically we use the version of this limiter that works with the primitive variables, and we set the parameter $\alpha = 500\Delta x^{1.5}$. In our experience, this limiter with these parameters offers a good balance between damping oscillations while maintaining sharply refined solutions. Additionally, we point out that extra efficiency can be realized by locally storing quantities computed for the aforementioned positivity-limiter as well as this shock-capturing limiter.

Unless otherwise noted, these examples use a CFL number of 0.12 with a 3$^{\text{rd}}$ order Lax-Wendroff time discretization, where the CFL number is defined as follows:

$$\text{CFL} := \max_{i \in [1,N]} \left(\frac{\Delta t |S_i|}{|\mathcal{T}_i|} \right), \tag{7.77}$$

where $|\mathcal{T}_i|$ is the area of element $\mathcal{T}_i$ and $|S_i|$ is the maximum wave speeds times the edge length over all three edges that make up element $\mathcal{T}_i$:

$$|S_i| := \max_{e \in \mathcal{T}_i} \left(|e| |s^e_{\max}| \right), \tag{7.78}$$

where $|e|$ is the length of edge e and $|s^e_{\max}|$ is the maximum wave speed on edge e. The CFL number of 0.12 is comparable to the CFL numbers presented in the original Lax-Wendroff DG paper of Qiu, Dumbser, and Shu [97]. All of the examples have the positivity-preserving and shock-capturing limiters turned on.

7.4.2 One-dimensional examples

This section presents some standard one-dimensional problems that can be found in [114] and references therein. These problems are designed to break codes that do not have a

mechanism to retain positivity of the density and pressure, but with this limiter, we are able to successfully simulate these problems.

Double rarefaction problem

Our first example is the double rarefaction problem that can be found in [142, 143, 114]. This is a Riemann problem with initial conditions given by $(\rho_L, u_{1L}, p_L) = (7, -1, 0.2)$ and $(\rho_R, u_{1R}, p_R) = (7, 1, 0.2)$. The solution involves two rarefaction waves that move in opposite directions that leave near zero density and pressure values in the post shock regime. The solution is presented on a mesh with a course resolution of $\Delta x = \frac{1}{100}$, as well as a highly refined solution with $\Delta x = \frac{1}{1000}$. The results, shown in Figure 7.4, are comparable to those obtained in other works, however the shock-capturing limiter we use is not very diffusive and therefore there is a small amount of oscillation visible in the solution at the lower resolution. However this oscillation vanishes for the more refined solution.

7.4.3 Sedov blast wave

This example is a simple one-dimensional model of an explosion that is difficult to simulate without aggressive (or positivity-preserving) limiting. The initial conditions involve one central cell with a large amount of energy buildup that is surrounded by a large area of undisturbed air. These initial conditions are supposed to approximate a delta function of energy density. As time advances, a strong shock wave emanates from this central region and they move in opposite directions. This leaves the central post-shock regime with near zero density.

The initial conditions are uniform in both density and velocity, with $\rho = 1$ and $u_1 = 0$. The energy density takes on the value $\mathcal{E} = \frac{3200000}{\Delta x}$ in the central cell and $\mathcal{E} = 1.0 \times 10^{-12}$ in every other cell. This problem is explored extensively by Sedov, who in his classical text gives an exact solution that is used to construct the exact solution plotted with the simulation [115]. The solution is shown in Figure 7.5. The results are quite good especially since given the a coarse resolution used $\Delta x = \frac{1}{100}$.

7.4.4 Two dimensional examples

Here, we highlight the fact that this solver is able to operate on both Cartesian and unstructured meshes.

Convergence Results

We first verify the high-order accuracy of the proposed scheme. For problems where the density and pressure are smooth and far away from zero, the limiters proposed in this chapter "turn off", and therefore have no effect on the solution. In order to investigate the effect

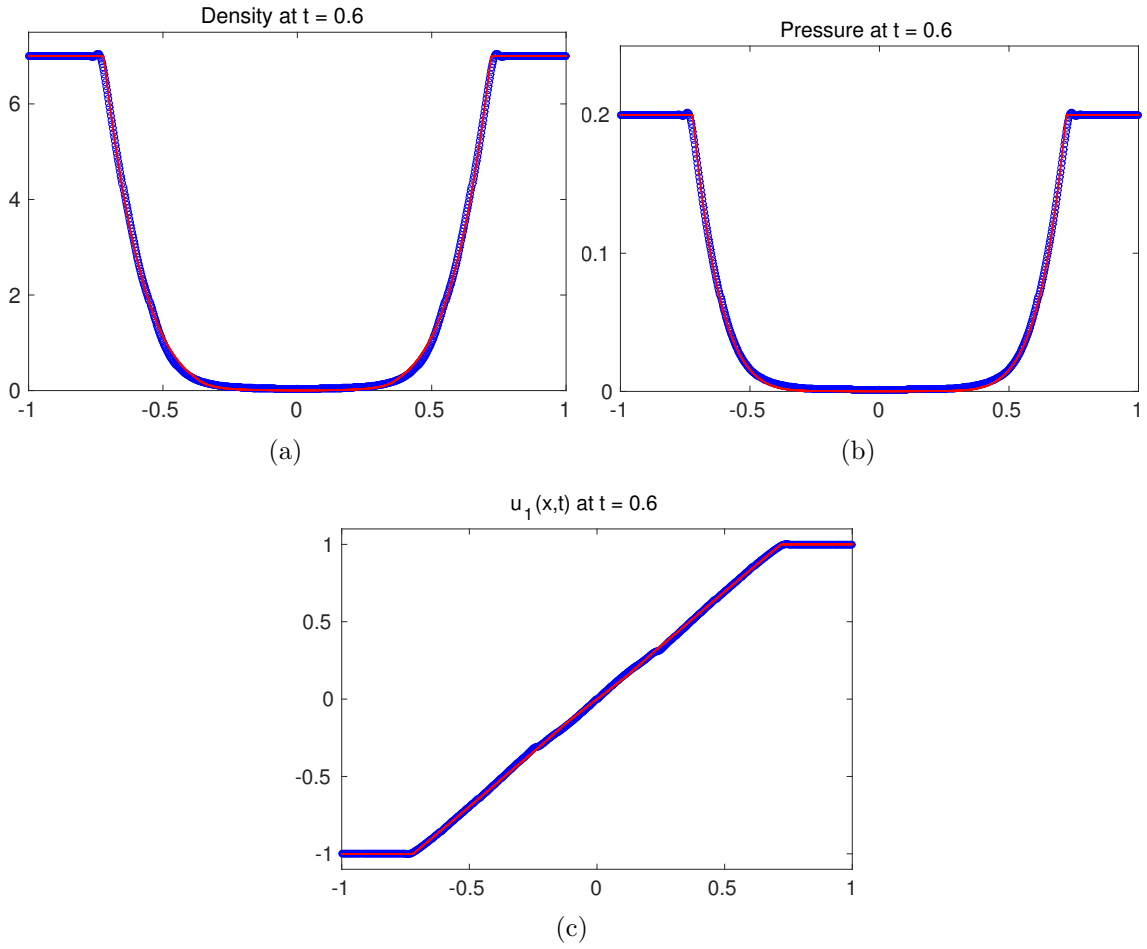


Figure 7.4: The solution for the double-rarefaction problem. This is a standard example that fails for methods that are not positivity-preserving. The blue dots correspond to the computed numerical solution, and the red line corresponds to the computed solution on a highly refined mesh.

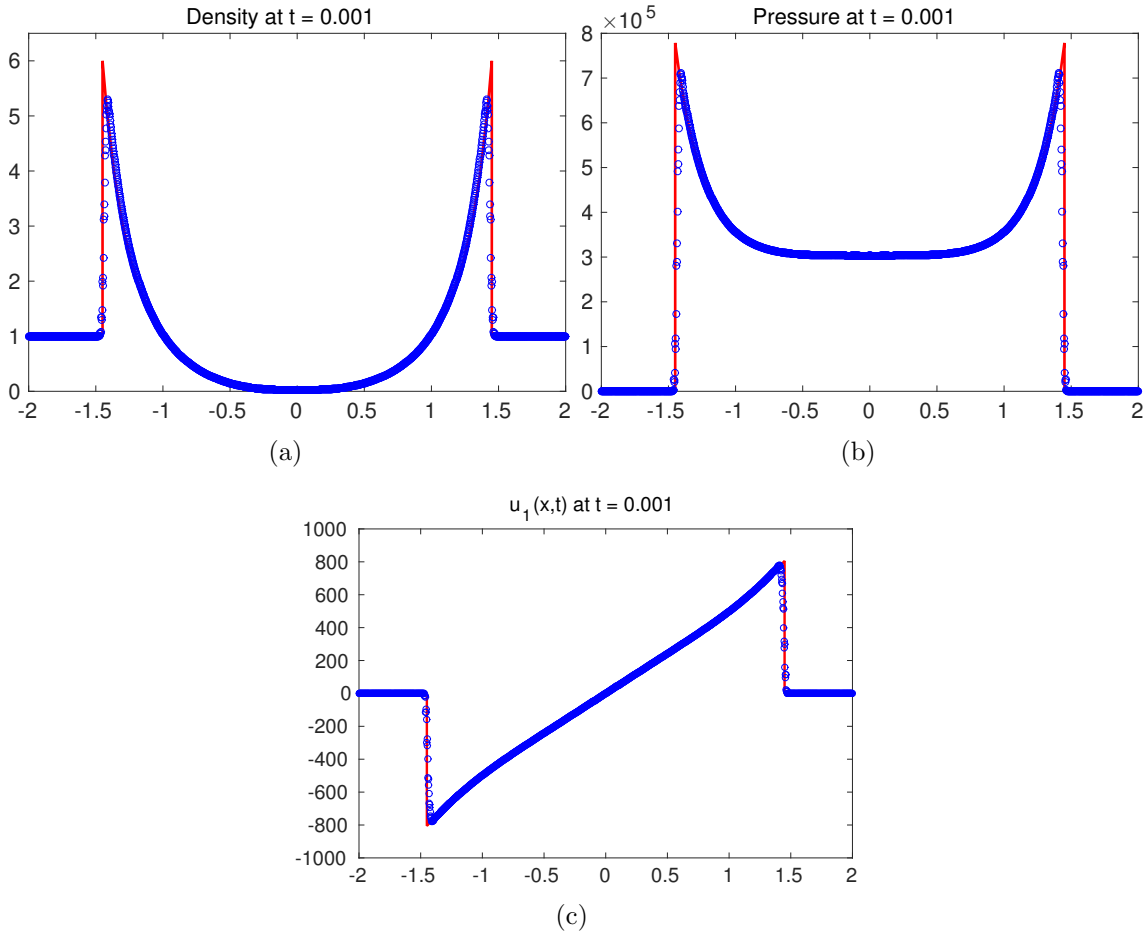


Figure 7.5: Sedov Blast-Wave. This is another standard example that fails without the positivity limiting. The blue dots correspond to the computed numerical solution, and the red line corresponds to the exact solution.

of this positivity preserving limiter, we simulate a smooth problem where the solution has regions that are nearly zero. This is similar to the smooth test case considered by [142, 114, 27].

To this end, we consider initial conditions defined by

$$\begin{pmatrix} \rho_0 \\ u_1 \\ u_2 \\ p \end{pmatrix} (t = 0, \mathbf{x}) = \begin{pmatrix} 1 - 0.9999 \sin(2\pi x) \sin(2\pi y) \\ 1 \\ 0 \\ 1 \end{pmatrix} \tag{7.79}$$

on a computational domain of $[0, 1] \times [0, 1]$. We integrate this problem up to a final time of $t = 0.02$, and compute L^2-norm errors against the exact solution given by

$$\begin{pmatrix} \rho \\ u_1 \\ u_2 \\ p \end{pmatrix} (t, \mathbf{x}) = \begin{pmatrix} 1 - 0.9999 \sin(2\pi\,(x - t)) \sin(2\pi y) \\ 1 \\ 0 \\ 1 \end{pmatrix}. \tag{7.80}$$

Results for Cartesian as well as unstructured meshes are presented in Table 7.1. These indicate that the high-order (3rd order) accuracy of the method is not sacrificed when the limiters are turned on.

As a final note, we observe that it appears that in general much higher resolution is required on unstructured meshes before the numerical results enter the asymptotic regime.

# Cartesian cells	Error	Order	# triangular cells	Error	Order
784	1.79×10^{-04}	—	20400	1.03×10^{-05}	—
1764	7.08×10^{-06}	7.966	29280	2.36×10^{-06}	8.148
3969	2.10×10^{-06}	2.994	42048	2.43×10^{-07}	12.560
8836	6.22×10^{-07}	3.045	60550	1.39×10^{-07}	3.084
19881	1.86×10^{-07}	2.971	86526	8.11×10^{-08}	2.999
44944	5.48×10^{-08}	3.000	124998	4.67×10^{-08}	3.003
—	—	—	179998	2.70×10^{-08}	3.013

Table 7.1: Convergence results for the 2D Euler problem in Section 7.4.4. All errors are L^2 norm errors. We see that the positivity preserving limiter does not affect the asymptotic convergence rate of the method. Also note that this solution was run only to short time, as in [142], [114] and [27], because for this example we must resolve a positive, yet very small density leading to a very large wave-speed and thus a very small permissible time-step.

Sedov blast on an unstructured mesh

In this example we implement a two-dimensional version of the Sedov blast wave on the circular domain

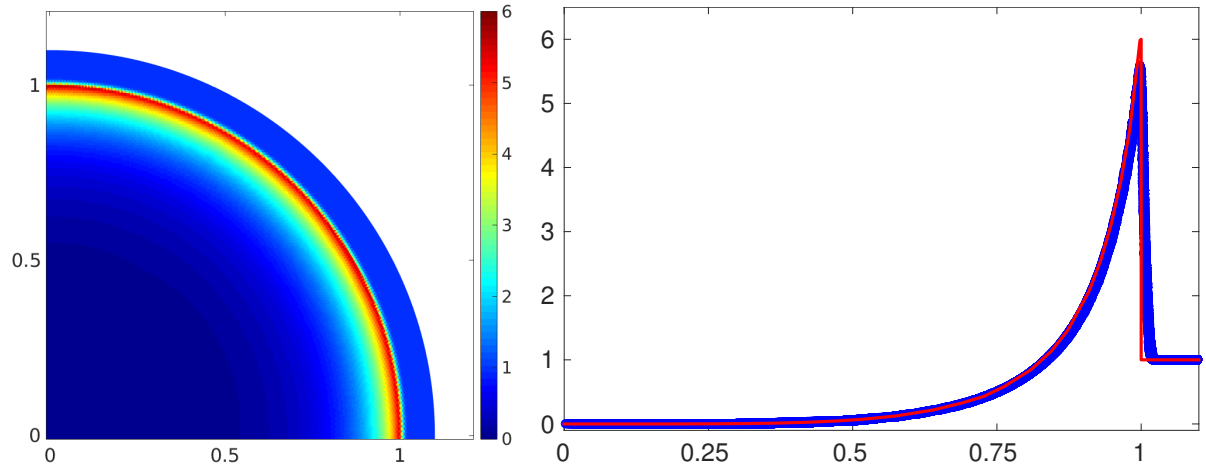
$$\Omega = \left\{(x, y) : x^2 + y^2 \leq 1.1\right\}.$$

The bulk of the domain begins with an undisturbed gas, $\vec{u} \equiv \vec{0}$, with uniform density $\rho \equiv 1$, and near-zero energy density $\mathcal{E} = 10^{-12}$. Only the cells at the center of the domain contain a large amount of energy. This is supposed to approximate a delta function. To simulate this, we introduce a small region at the center of the domain that is radially symmetric (because the simulation should be radially symmetric) of the form

$$\mathcal{E} = \begin{cases} \frac{0.979264}{\pi r_d^2} & \sqrt{x^2 + y^2} < r_d, \\ 10^{-12} & \text{otherwise}, \end{cases}$$

where $r_d = \sqrt{\frac{\pi 1.1^2}{\#\text{cells}}}$ is a characteristic length of the mesh.

We present results in Figure 7.6 where we simulate our solution with a total of 136270 mesh cells.

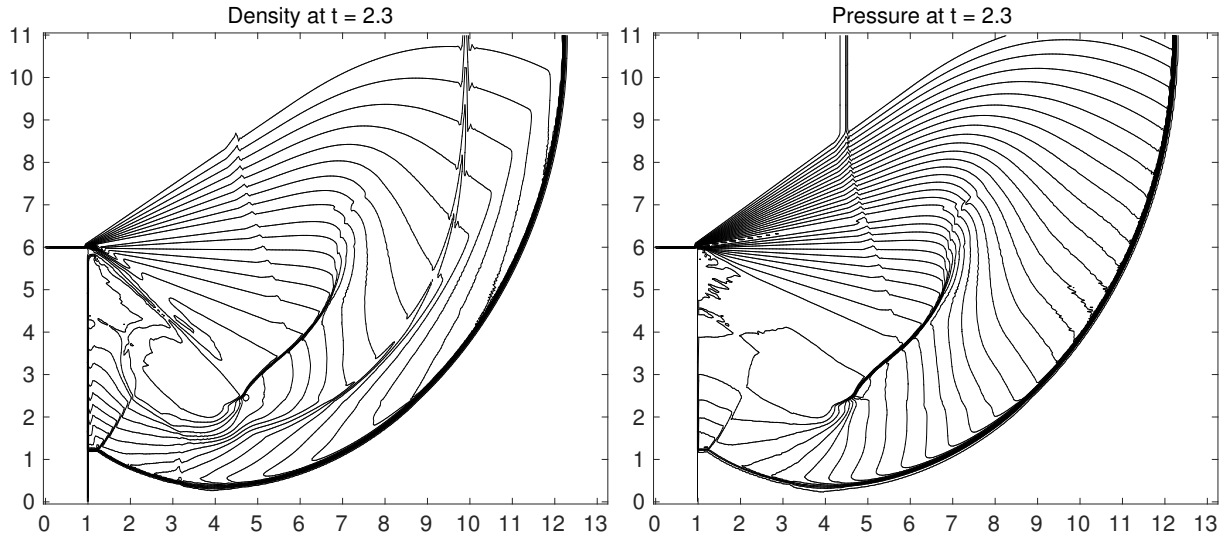


(a) (b)

Figure 7.6: Sedov blast problem. Here, we show density plots of the two dimensional Sedov problem we introduce in Section 7.4.4 . The clear regions in the upper right part of subfigure (a) are due to the fact that we only mesh the interior of the circular domain $\{(x, y) : x^2 + y^2 \leq 1.1\}$. Also note that we ran this problem on the entire circular region but only plot the upper right region of the solution.

Shock-diffraction over a backward facing step: Cartesian mesh

This is a common example used to test positivity limiters [114, 142, 143]. It involves a Mach 5.09 shock located above a step moving into air that is at rest with $\rho = 1.4$ and $p = 1.0$. The domain this problem is typically solved on is $[0, 1] \times [6, 11] \cup [1, 13] \times [0, 11]$. The step is the region $[0, 1] \times [0, 6]$. Our boundary conditions are transparent everywhere except above the step where they are inflow and on the surface of the step where we used solid wall. Our initial conditions have the shock located above the step at $x = 1$. The problem is typically run out to $t = 2.3$, and if a positivity limiter is not used the solution develops negative density and pressure values, which causes the simulation to fail. The solution shown in Figure 7.7 is run on a 390×330 Cartesian mesh.

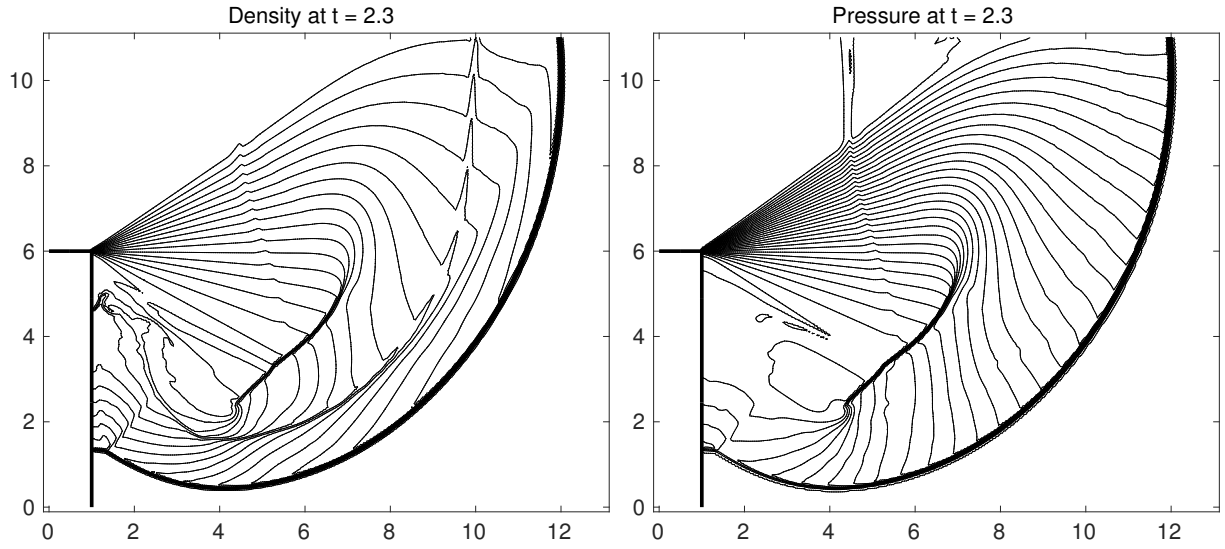


(a) density (b)

Figure 7.7: The Mach 5.09 shock-diffraction test problem on a 390×330 Cartesian mesh. For the density, we plot a total of 20 equally spaced contour lines ranging from $\rho = 0.066227$ to $\rho = 7.0668$. For the pressure, we plot a total of 40 equally spaced contour lines ranging from $p = 0.091$ to $p = 37$ to match the figures in [142].

Shock-diffraction over a block step: unstructured mesh

Next, we run the same problem from Section 7.4.4, but we discretize space using an unstructured triangular mesh with 126018 cells. The results are shown in Figure 7.8, and indicate that the unstructured solver behaves similarly to the Cartesian one.



(a) Density (b) Pressure

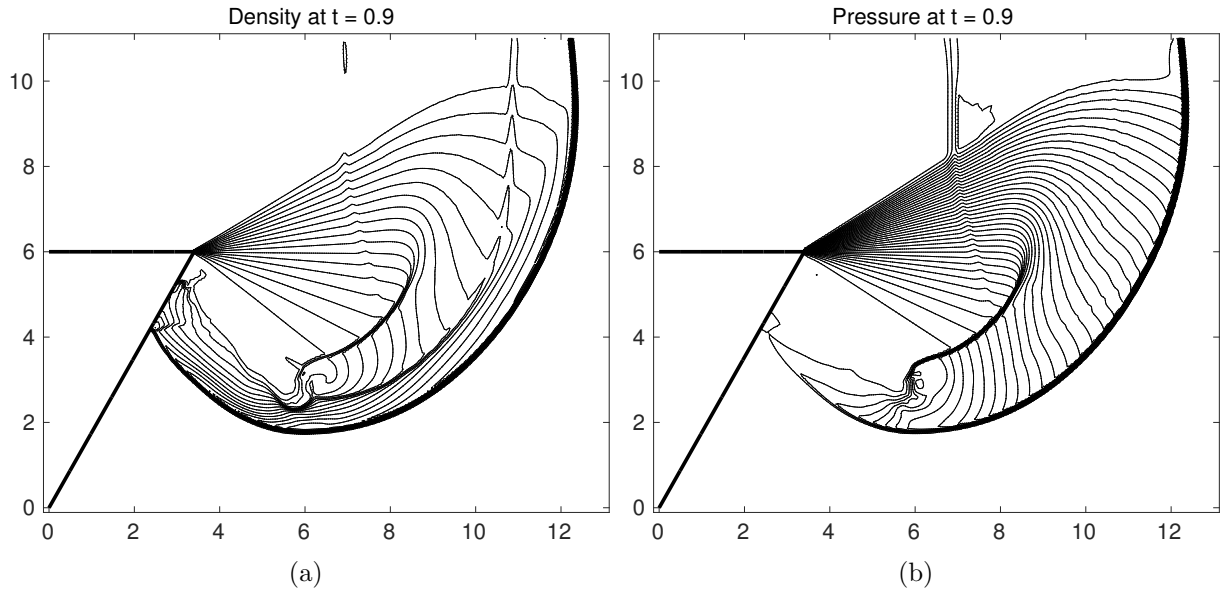
Figure 7.8: The shock-diffraction test problem on an unstructured triangular mesh. For the density, we plot a total of 20 equally spaced contour lines ranging from $\rho = 0.066227$ to $\rho = 7.0668$. For the pressure, we plot a total of 40 equally spaced contour lines ranging from $p = 0.091$ to $p = 37$, again to match the figures in [142].

Shock-diffraction over a 120 *degree wedge*

This final shock-diffraction test problem is very similar to the previous test problems, however it must be run on an unstructured triangular mesh because the wedge involved in this problem is triangular (with a 120 degree angle) [145]. Our domain for this problem is given by

$$[0, 13] \times [0, 11] \setminus [0, 3.4] \cup [0, 3.4] \times \left[\frac{6.0}{3.4}x, 6.0\right].$$

Again the boundary conditions are transparent everywhere except above the step where they are inflow and on the surface of the step where they are reflective solid wall boundary conditions. In addition, the initial conditions for this problem are also slightly different that those in the previous example. Here, we have a Mach 10 shock located above the step at $x = 3.4$, and undisturbed air in the rest of the domain with $\rho = 1.4$ and $p = 1.0$. This problem is run on an unstructured mesh with a total of 122046 cells. These results are presented in Figure 7.9.



(a) (b)

Figure 7.9: Shock diffraction problem with a wedge. Here, we present numerical results for the Mach 10 shock diffraction problem, where the shock passes over a 120 degree angular step. For the density, we plot a total of 20 equally spaced contour lines ranging from $\rho = 0.0665$ to $\rho = 8.1$. For the pressure, we plot a total of 40 equally spaced contour lines ranging from $p = 0.5$ to $p = 118$.

Chapter 8

A DIFFERENTIAL TRANSFORM DISCONTINUOUS GALERKIN METHOD WITH ADAPTIVE MESH REFINEMENT

8.1 Introduction

This chapter develops a high-order DT-DG algorithm with block structured adaptive mesh refinement (AMR) and local time stepping. Section 8.2 first describes block structured adaptive mesh refinement as it is implemented in AMRClaw. Section 8.3 introduces the basic DT-DG method on a single uniformly refined grid. Following these introductions, discussion turns to the modifications that need to be made to the DT-DG scheme in order to implement AMR with local time-stepping. The AMR algorithm is presented in one-dimension, however it is written in such a way that it will generalize to higher dimensions just as the AMRClaw software generalizes to up to three dimensions. Finally several 1D numerical results are presented.

8.2 Introduction to AMR

This section outlines the basics of the block-structured AMR algorithms for wave-propagation problems developed in [13] and [12]. This will only serve as a brief outline and readers are encouraged to explore those references. Also it is useful to recall that the methods discussed in this chapter are designed to be used with the numerical scheme introduced in Section 2.1.

Any procedure that strategically distributes degrees of freedom in a problem domain in a way that changes with time could be described as a type of AMR. The goal of AMR is to adapt the resolution of a numerical method in space and time in a way that minimizes the amount of computation that is performed. In the hyperbolic PDE context this means coarsening or refining the grid depending on the resolution necessary to resolve the solution in a region.

8.2.1 The AMR algorithm

The term block-structured refers to the fact that in these types of AMR algorithms the domain is split up into properly-nested patches or blocks. On these patches computation will proceed as if the patch is the entire domain. The way to think about the methods under discussion is to consider the domain as having multiple patches of grids stacked on top of

each other. The entire domain is covered by a uniform coarse patch. This is the level 1 grid. On top of that grid in some regions is another grid that is refined by some constant factor m_1. This is the level 2 grid. Then there may be a level 3 grid refined by some constant factor m_2 from the level 2 grid. This grid can only exist on top of the level 2 grid, so it cannot be directly above a level 1 grid. Each grid is the union of several smaller rectangular patches. Each patch is given a halo of extra cells around it that are known as ghost cells. These ghost cells are used to communicate boundary condition information to each patch. In some cases the boundary conditions are defined by actual physical boundary conditions, in other cases the ghost cells are filled by values pulled from other patches that they intersect. Given the information about the algorithm discussed in Section 2.1, the halo is typically 2 cells wide on each side.

Whatever the number of levels, the algorithm begins by populating these grids with values just by interpolating from known initial conditions. The method for updating the solution is as follows (assume that there are only 2 grid levels but it is easy to extend to more).

Step 1. Take a time step Δt_1 using a numerical method such as the one defined in Equation (2.6) on the coarsest grid.

Step 2. Populate the ghost cells on the level 2 patches. If a level 2 ghost cell overlaps another level 2 patch one may just copy the values between the patches. Ghost cells that do not intersect a patch on level 2 will be populated by interpolating from cell values on level 1. If $\Delta t_2 < \Delta t_1$ a space-time interpolation must be performed. An interpolation scheme is used that takes the cell values on the level 1 grid at time t^n and $t^n + \Delta t_1$, along with the neighboring physical cell value on level 2, to produce an approximate ghost cell value at $t^n + \Delta t_2$. If $\Delta t_2 = \Delta t_1$ then an interpolation scheme is used that takes the cell values on the level 1 grid and the neighboring physical cell value on level 2 to approximate the ghost cell value.

Step 3. Take a time step Δt_2 on the next coarsest grid. Note that $\Delta t_2 \leq \Delta t_1$

Step 4. Repeat steps 2 and 3 until level 2 and level 1 are at the same time.

Step 5. Update the level 1 grid values using the more accurate level 2 values. In the finite volume case this is done by averaging the level 2 solution onto the level 1 cells.

Step 6. Repeat steps 1-5 until it is time to adapt the mesh. Use some criteria to determine if any level 1 cells should be divided and flag them if they should be.

Step 7. Use the flagged cells to recreate a new patch structure. Populate the newly refined cells by interpolating their values from the underlying coarse grid or copying from an existing level 2 patch if available.

Note that in the algorithm defined above there will be cells where the method is no longer conservative. For example whenever you perform local time stepping the sum of the fluxes on the fine mesh will not necessarily match the single flux on the coarse grid. In these situations one must use a method known as a conservation fix up to maintain conservation [12]

Given this outline of the AMR algorithm one can see the basic requirements for a DG scheme to fit into a software package using block-structured AMR of this form. The main constraint is that the method has to have a numerical stencil that matches the stencil of the finite volume method spatially (the DG scheme will of course have more degrees of freedom per cell). Section 8.4 will outline how to adapt this algorithm to a DT-DG scheme.

8.3 The DT-DG method

The derivation of the DT-DG scheme starts with the typical Lax-DG weak form

$$\int_{\mathcal{T}_i} q^{n+1}\varphi^{(\ell)}d\mathbf{x} = \int_{\mathcal{T}_i} q^n\varphi^{(\ell)}d\mathbf{x} - \Delta t \int_{\mathcal{T}_i} \nabla\varphi^{(\ell)} \cdot \mathbf{F}_T(q^n)\, d\mathbf{x} + \Delta t \oint_{\partial\mathcal{T}_i} \varphi^{(\ell)}\, \mathbf{F}_T(q^n)\cdot\hat{\mathbf{n}}\, ds. \quad (8.1)$$

With

$$\mathbf{F}_T^n(q) := f_T^n(x,y,t)\hat{\mathbf{x}} + g_T^n(x,y,t)\hat{\mathbf{y}}, \quad (8.2)$$

where

$$f_T^n(x,y,t) = \sum_{h_1=0}^{M_1}\sum_{h_2=0}^{M_2}\sum_{k=0}^{H} F(h_1,h_2,k)(x-x^*)^{h_1}(y-y^*)^{h_2}(t-t^*)^k, \quad (8.3)$$

and

$$g_T^n(x,y,t) = \sum_{h_1=0}^{M_1}\sum_{h_2=0}^{M_2}\sum_{k=0}^{H} G(h_1,h_2,k)(x-x^*)^{h_1}(y-y^*)^{h_2}(t-t^*)^k. \quad (8.4)$$

The functions $f_T^n(x,y,t)$ and $g_T^n(x,y,t)$ can be computed with the differential transform method. Use a hierarchical orthonormal Legendre polynomial basis so that

$$q^h(t^n,\mathbf{x}(\boldsymbol{\xi}))\Big|_{\mathcal{T}_i} = \sum_{\ell=1}^{M_\mathrm{L}} Q_i^{(\ell)n}\,\varphi^{(\ell)}(\boldsymbol{\xi}), \quad (8.5)$$

where M_L is the number of basis functions and $\varphi^{(\ell)}(\boldsymbol{\xi}) : \mathbb{R}^d \mapsto \mathbb{R}$ are the basis functions defined on the reference element $\mathcal{T}_0$ in terms of the reference coordinates $\boldsymbol{\xi} \in \mathcal{T}_0$.

The first step of this derivation involves performing a Galerkin projection by multiplying by a test function that is chosen as a Legendre polynomial $\varphi^{(k)}$. Integrating by parts yields

$$Q_i^{(\ell)\,n+1} = Q_i^{(\ell)\,n} - \frac{\Delta t}{|\mathcal{T}_i|}\int_{\mathcal{T}_i} \nabla\varphi^{(\ell)}\cdot\mathbf{F}_T^h\, d\mathbf{x} + \frac{\Delta t}{|\mathcal{T}_i|}\oint_{\partial\mathcal{T}_i} \varphi^{(\ell)}\,\mathbf{F}_T^{h*}\cdot\hat{\mathbf{n}}\, ds. \quad (8.6)$$

The flux values, $\mathbf{F}_T^{h*}$, in the edge integrals still need to be defined. Consider the fact that for a one dimensional linear PDE where A is independent of t

$$q_t + Aq_x = 0,$$

implies that

$$q_{tt} + Aq_{tx} = 0,$$

and

$$q_{ttt} + Aq_{ttx} = 0.$$

This means that for a linear 1D PDE the boundary integral contribution of the modified flux function can be computed by solving the 1D Riemann problem

$$\tilde{q}_t + A\tilde{q}_x = 0$$

with $\tilde{q}_l = \left(q + \Delta t \frac{q_t}{2} + \Delta t^2 \frac{q_{tt}}{6} + \cdots\right)_l$ and $\tilde{q}_r = \left(q + \Delta t \frac{q_t}{2} + \Delta t^2 \frac{q_{tt}}{6} + \cdots\right)_r$. For a nonlinear conservation law one can use the local Lax Friedrichs (LLF) or Rusanov flux with the modified flux function as in Equation 7.22, or plug the modified flux function into other approximate Riemann solvers. In summary the DT-DG method is very similar to the method outlined in Chapter 7 except that the Taylor series of the flux function is computed using a differential transform. Also note that because the method creates an explicit space-time series expansion of the flux function it is possible to implement this method in a quadrature free fashion.

8.4 High-order AMR for DG

This section outlines how the block structured AMR algorithm discussed in Section 8.2.1 could be modified to work with the DT-DG scheme. The basic outline is the same as in Section 8.2.1 except that step 5 will change. In step 5 the level 1 grid values will be updated by performing a L2 projection of the level 2 solution onto the level 1 cell basis functions.

8.4.1 Populating fine-level ghost cells (and local time-stepping)

For fine grid ghost cells that intersect other fine grid cells, the ghost cells can be populated just by copying values. For cells that only intersect coarser levels it is necessary to compute an approximation to the value the fine grid ghost cell should take, possibly at a slightly later time. However, when populating the ghost cells that only intersect a coarser level, it is only actually necessary to guarantee that the solution on the ghost cell will be accurate at the edge of the cell bordering the physical domain of the patch (ignoring the potential limiting for now). This inspires the following method.

Say that we need to populate q_0^{h2} (the first ghost cell on a level 2 patch) from q_i^{h1} (cell i on a level 1 patch) at time $t_n + \Delta t_2$ Use the differential transform procedure to define (for

an M defined by the order of the method) the prediction of the Taylor series

$$\tilde{q}_i^{h1}(x) = \sum_{h=0}^{M} \sum_{k=0}^{M} Q^{h1}(h,k)(x - x_i)^h (\Delta t_2)^k. \tag{8.7}$$

Then let us define q_0^{h2} to be

$$q_0^{h2} = \tilde{q}_i^{h1}(x_{\frac{1}{2}}), \tag{8.8}$$

where $x_{\frac{1}{2}}$ is the boundary point where q_0^{h2} must be accurate. The reason the ghost cell is given a constant value is that the method does not need that value for anything except to provide the boundary flux value. There is no downside to setting the ghost cell value to the constant that will produce the correct flux to pass into the physical domain. This may change when limiters are introduced as the method will need to get more information from the ghost cells. This version of local time-stepping is more efficient than projecting the space time approximation onto the basis belonging to q_0^{h2}, and it seems to be very accurate.

8.4.2 Updating the coarse grid solution

When updating the coarse grid solution, there doesn't seem to be a way to avoid projecting the fine grid solution onto the coarse grid function space. This is because interpolation across possibly discontinuous cells is unstable and can build up spurious oscillations over time. The L2 projection seems to avoid this. Let us return to using $\varphi^{h1}(\xi(x))$ to define Legendre polynomials on a reference element corresponding to the level 1 grid. If we say the coefficient of the hth Legendre polynomial in cell i of level 1 at time t_n is equal to $Q_i^{n,h1}(h)$ then solution on the coarse grid can be represented as

$$\tilde{q}_i^{h1}(x, t = t^n) = \sum_{h=0}^{M} Q_i^{n,h1}(h) \varphi^{h1,(h)}. \tag{8.9}$$

In the block-structured AMR framework refinement is always by a constant number. For example if the refinement ratio between levels 1 and 2 is m_1 the method will always involve projecting from m_1 cells onto one cell (the refinement ratio can vary between different levels). Say that cell i on level 1 is under the cells I_m for $m \in (1, \ldots, m_1)$. Then the projection to compute the coefficient $Q_i^{n,h1}(h)$ will look like

$$Q_i^{h1}(h) = \sum_{m=1}^{m_1} \int_{I_m} q_{I_m}^{h2} \varphi^{h1,(h)} dx. \tag{8.10}$$

This integral will require computing φ^{h1} at quadrature nodes within each of the m_1 finer level cells. Essentially this will require computing m_1 matrices that will store the values of every $h1$ level basis function at the quadrature points in cells of level $h2$. However, these matrices are the same for every projection between these two levels and can be computed once and stored at the beginning of the simulation.

8.5 Numerical results

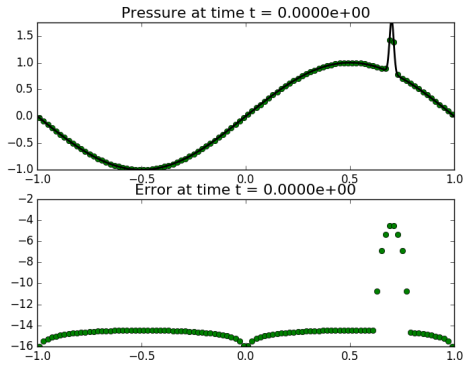
The examples in this section will implement the variable coefficient acoustic equations in one dimension. This equation is chosen because it is a relatively simple hyperbolic system that will focus to be directed toward the properties of the AMR scheme itself.

8.5.1 Motivating using a high order method with AMR

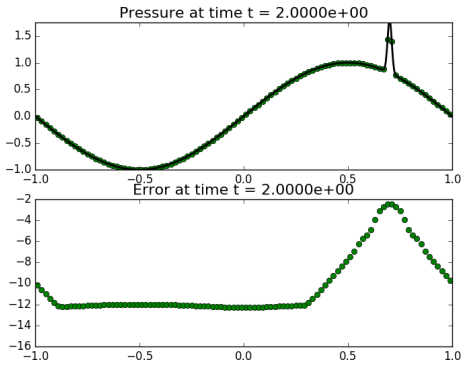
This first test will motivate using high-order with AMR. The one dimensional acoustic equations will be run with $K_0 = \rho = c = 1$ on the domain $[-1, 1] \times [-1, 1]$, with periodic boundary conditions. On this domain the solution returns to the initial condition every 2 time units. The goal of this example is to confirm that the interpolations involved in the AMR scheme do not destroy the accuracy of the numerical methods. Two different examples are run. Both examples are studied with two single grid simulations and an AMR simulation where the two possible levels of refinement have resolutions that match the two coarse grid simulations. In particular one single level simulation is run with $mx = 100$, and the other single level simulation is run with $mx = 200$. The two level AMR simulation is run with the two levels possessing resolutions that match the two single level simulations. Figures 8.1 and 8.2 show the pressure distribution of each simulation, along with the error at the given time. Two initial conditions are investigated. The simulation shown in Figure 8.1 has an initial condition that is made up of the sum of a sine function and a sharply peaked Gaussian. The simulation shown in Figure 8.2 has an initial condition that is made up of the sum of two Gaussian functions, one much sharper than the other. In these figures, when AMR is plotted the blue dots represent solution values on the coarse grid and the red dots represent solution values on the fine grid. The error is computed by sampling both the polynomial approximation and the actual solution at many points and then computing the maximum difference on each cell. This maximum difference is then plotted at the cell center. In both instances the AMR is able to maintain error on the fine patch similar in magnitude to error on the single grid $mx = 200$ simulation. The interpolations being performed as part of the simulation do not seem to be severely impacting the accuracy.

8.5.2 On the long time integration behavior of the high order AMR scheme

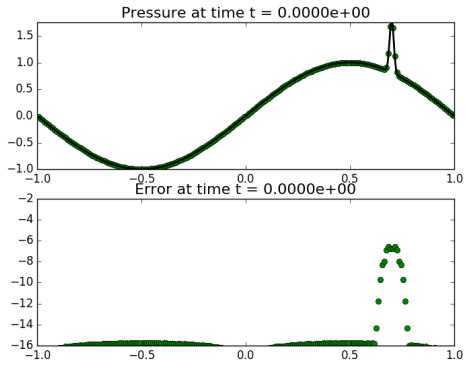
This test will illustrate that the AMR is stable even with discontinuous initial conditions. This test is run on the one dimensional acoustic equations with $K_0 = \rho = c = 1$ on the domain $[-1, 1] \times [-1, 1]$ with periodic boundary conditions. The minimum possible cell size is $h = 0.00625$ with three levels of AMR. Figure 8.3 shows the cell averages plotted after 20 iterations with a 5th order DT-DG method. Clearly some errors have built up and there are some oscillations visible, as is to be expected when using a high order scheme without limiting. However the performance does not seem unusually bad for unlimited DG. For



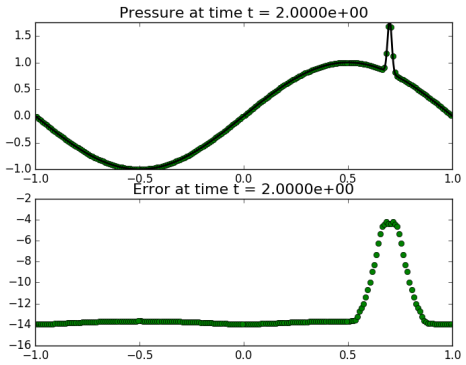
(a) One level $mx = 100$



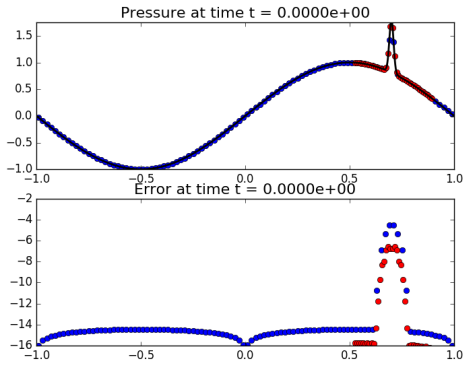
(b) One level $mx = 100$



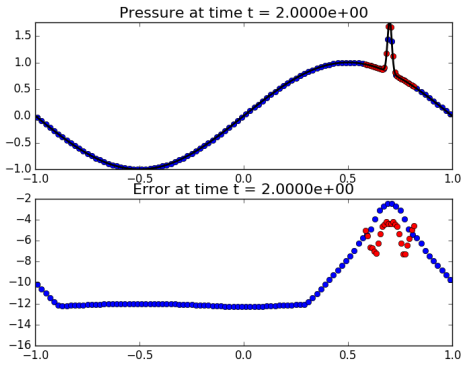
(c) One level $mx = 200$



(d) One level $mx = 200$

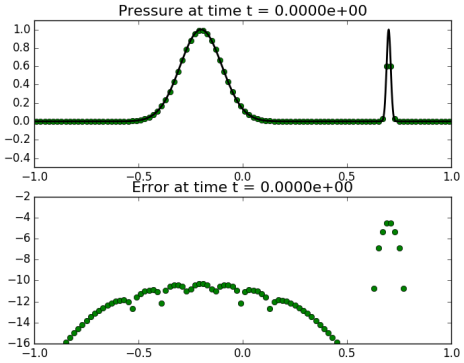


(e) Two levels with $mx = 100$

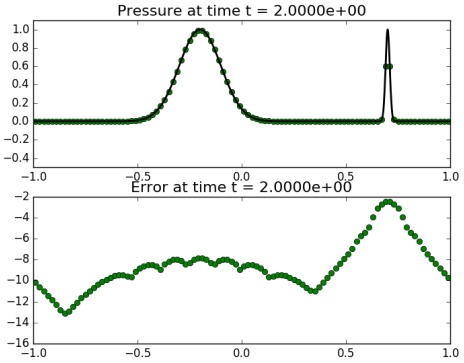


(f) Two levels with $mx = 100$

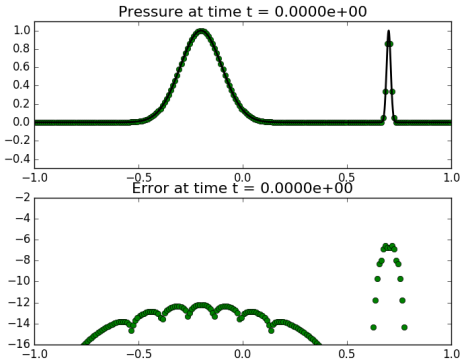
Figure 8.1: The constant coefficient acoustic equation on a periodic domain run until the solution has returned to its initial distribution at $t = 2$. In the AMR plot blue dots are level 1 grid cells and the red dots are level 2 grid cells. The initial pressure profile is a sine wave added to a sharply peaked Gaussian. The pressure and log scale error are plotted. Two uniform grid solutions to are compared to an AMR solution.



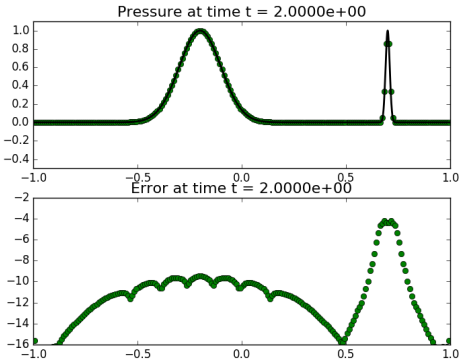
(a) One level $mx = 100$



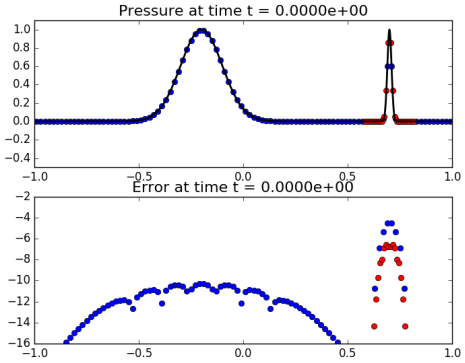
(b) One level $mx = 100$



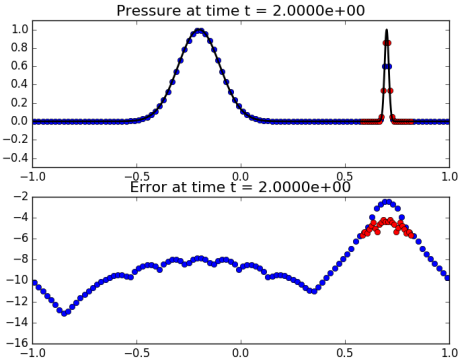
(c) One level $mx = 200$



(d) One level $mx = 200$

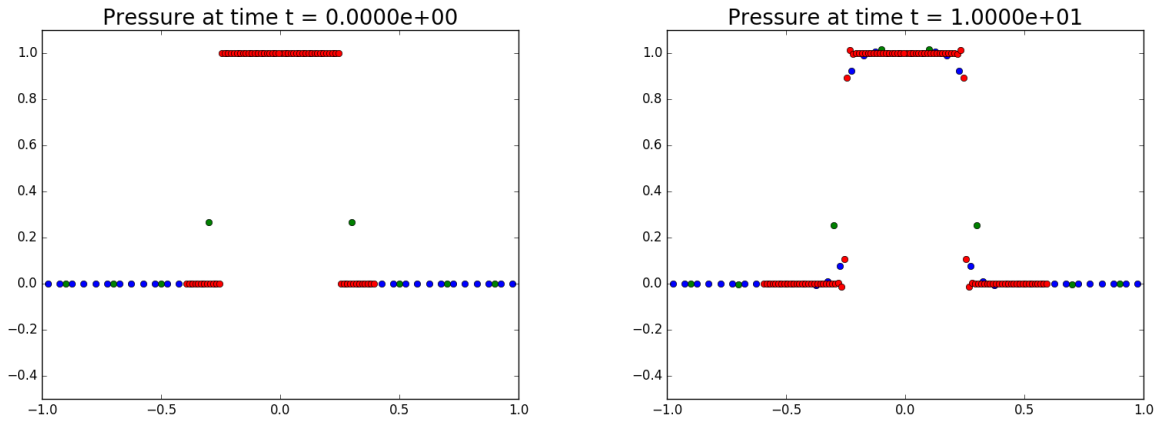


(e) Two levels with $mx = 100$



(f) Two levels with $mx = 100$

Figure 8.2: The same pde solved in Figure 8.1. In the AMR plot blue dots are level 1 grid cells and the red dots are level 2 grid cells. The initial pressure profile is a broader Gaussian added to a sharply peaked Gaussian. The pressure and log scale error are plotted. Two uniform grid solutions to are compared to an AMR solution.



(a) Initial condition

(b) 20 revolutions

Figure 8.3: The constant coefficient acoustic equation on a periodic domain run to time 40, after 20 revolutions through the periodic domain. This was run with a 5th order DT-DG method with 3 possible levels of refinement. Note that in this figure the green dots are cells on the coarse grid, the blue dots are cells on the second level of refinement, and the red dots are on the third level of refinement ($h = 0.00625$).

example we could compare to the unlimited plot in Figure 4.3.

8.5.3 Handling interfaces

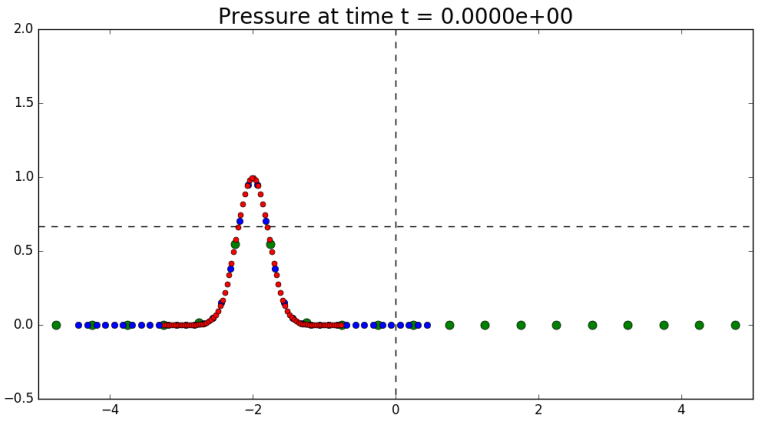
Many DG schemes are written to rely on the existence of a flux function, however that is not a limitation inherent to the method. The implementation used in this chapter (and indeed the entire thesis) creates a flux-like function that can take different values on either side of a cell edge. On problems without a flux function we can use the waves produced by solving the Riemann problem to define this function on either side of a cell edge. This will be important here when running acoustics with piecewise constant coefficient values.

$$z = \begin{cases} 1 \text{ for } x < 0 \\ 2 \text{ otherwise} \end{cases} \quad \text{and } \rho = 1$$

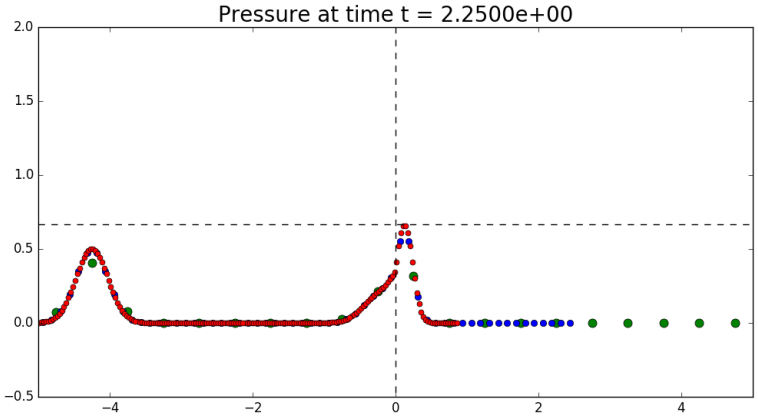
The initial pressure distribution is a Gaussian. For this problem we can compute the transmission and reflection coefficients

$$C_T = \frac{2Z_r}{Z_l + Z_r} = \frac{4}{3} \quad \text{and} \quad C_r = \frac{Z_r - Z_l}{Z_r + Z_l} = \frac{1}{3}$$

This means that after impacting the interface the peak of the Gaussian should jump up to $\frac{4}{3}\frac{1}{2} = \frac{2}{3}$ which is the horizontal line plotted in Figures 8.4 and 8.5.



(a) initial condition

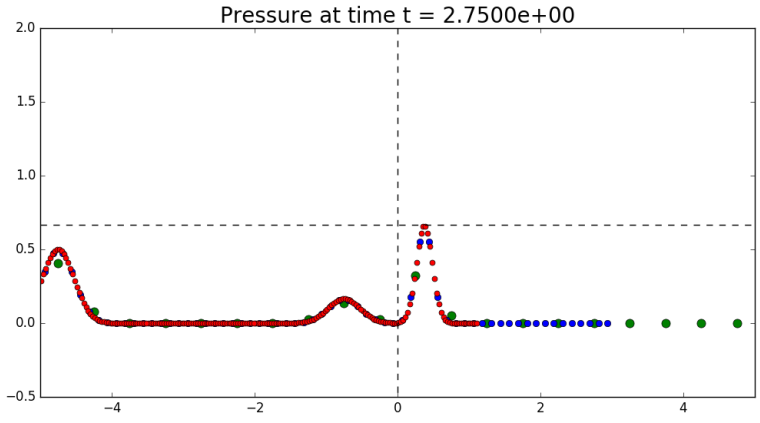


(b) 20 revolutions

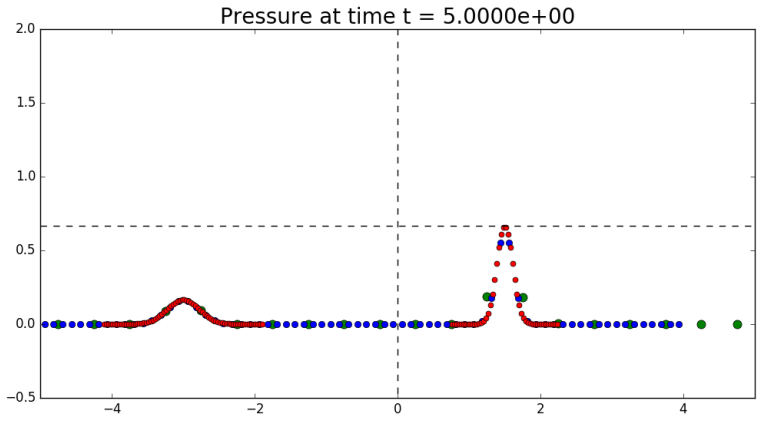
Figure 8.4: An acoustics problem with an interface at $x = 0$ where the impedance doubles.

Accuracy of local time stepping

This is another constant coefficient acoustics example with $K_0 = \rho = c = 1$. The problem has three levels of possible refinement, but only two levels of refinement are allowed to the left of $x = 0$. This forces the scheme to perform the space-time series predictor from Section 8.4.1 in a region where the solution is not perfectly resolved. It can be seen in Figure 8.6 that the accuracy has not been severely truncated. However Table 8.1 shows that the convergence rate does seem to eventually be impacted.



(a) initial condition

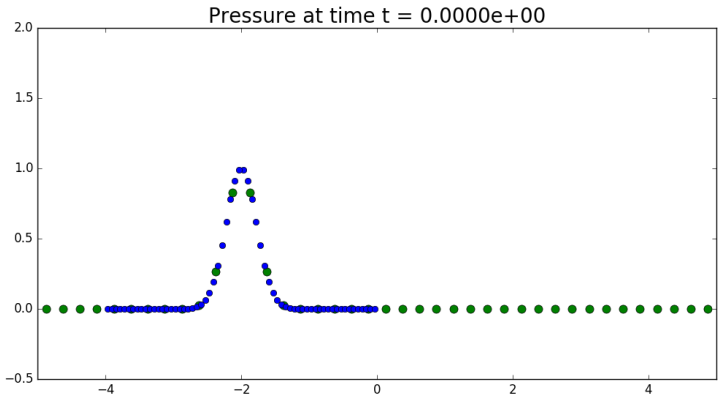


(b) After 20 revolutions

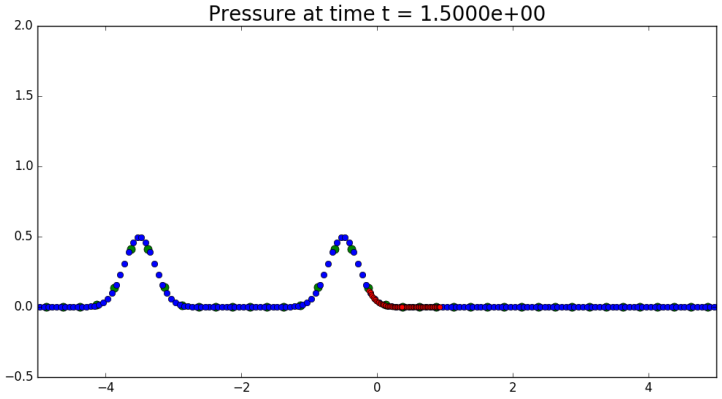
Figure 8.5: An acoustics problem with an interface at $x = 0$ where the impedance doubles.

Coarse Mesh	**L2 error**	**Order**
24	6.2×10^{-04}	—
40	6.4×10^{-06}	6.6
80	1.7×10^{-07}	5.2
160	7.0×10^{-09}	4.6

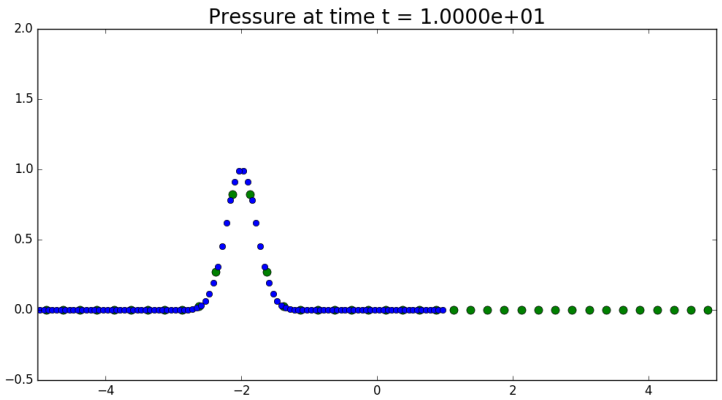
Table 8.1: Convergence of the 5th order differential transform method with AMR as the coarsest level is refined. This table corresponds to the example from Section 8.5.3.



(a) Initial condition

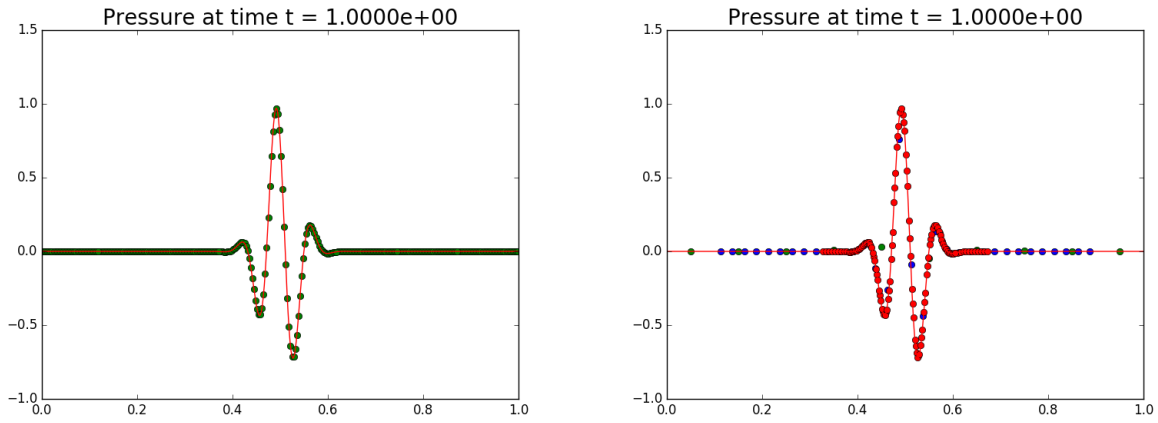


(b) Intersecting $x = 0$



(c) One revolution

Figure 8.6: A constant coefficient acoustics problem where maximum refinement is prevented on the right half of the domain. The goal is to show the accuracy of the local time-stepping method introduced in Section 8.4.1. These plots correspond to the example from Section 8.5.3



(a) Uniform grid with $mx = 300$

(b) AMR grid

Figure 8.7: Using AMR to efficiently resolve an oscillatory wave-packet.

Wave-packet example

Here an example is run to illustrate the potential computational advantage of the AMR scheme. The problem has an oscillatory wave-packet initial condition resolved using a uniform $mx = 300$ grid and an AMR grid with a maximum resolution allowing up to 320 cells (if every region was as refined as possible). Figure 8.7 shows that both methods perform very well, but the fixed grid scheme took 50.076 seconds on an i5-3320M CPU @ 2.60GHz processor and the method with AMR took 30.486 seconds. This difference could be made arbitrarily large by selecting initial conditions that are more localized, causing the uniform grid to waste more degrees of freedom. In higher dimensions this will provide an even larger benefit.

Chapter 9

SUMMARY, CONCLUSIONS AND FUTURE WORK

9.1 Summary of results

The focus of this thesis has ultimately been to develop a numerical scheme that can run on mapped grid geometries with h-adaptivity. These two goals together initially motivated me to look at DG schemes because of their extremely compact stencils and the fact that they only require passing fluxes. However, while the DG scheme provides a very good starting point for the development of such a method, the typical time-stepping and limiting methods used with DG often interfere with this goal.

Of the limiting schemes that are available in the literature the author does not know of any that can be used easily on mapped grids for higher order methods. The WENO schemes could work on mapped grids with significant modification, but it is unclear how well they would perform given the fact that WENO methods have in the past been shown to perform poorly on grids with non-smooth mappings. Slope limiting methods typically rely on finite difference approximations of the derivatives of a function. They then compare these approximations to the moments of the solution. However on mapped grids it is very difficult to say exactly what any particular moment corresponds to. This is because the moments are defined on a rectangular or cubic reference element that is mapped with a general bilinear or trilinear mapping.

Additionally the available limiting schemes sacrifice some of the very nice properties possessed by DG. The WENO limiting schemes give up DG's extreme locality, requiring users to look at wide stencils. Slope limiting schemes typically restrict DG to structured grids. Artificial viscosity limiters require DG's already small time step be reduced by another factor of n where n is the polynomial order of the basis functions.

DG is typically implemented with Runge-Kutta time stepping schemes. These methods are well developed; the literature is full of a menagerie of exotic RK schemes. However there are two aspects in which the RK methods are unsatisfactory. First, the RK methods require one to choose between adding many more stages or storing more data. Storing multiple stages is undesirable, particularly in three space dimensions, but so is adding more stages as each stage is an added communication and often an added step that must be limited. Additionally it is very difficult to perform local time stepping with RK schemes. Local time-stepping usually provides a very large computational improvement when using AMR so this is a significant drawback.

In this work I began by introducing, in a very general way, the DG method. Chap-

ter 2 discussed both RKDG and Lax-DG and briefly outlined how one could proceed with each method. Chapter 3 then introduced the mapped grid DG scheme. This scheme involves working on a computational domain where the cells are rectangles that are mapped to quadrilaterals in physical space. The specifics of this mapping become very important for understanding the performance of this method. I discussed why the desire to achieve $(n+1)$st order convergence on general mapped grids requires one to work with the function space known as $\mathbb{Q}_n^d$ (or the nth order tensor product function space). In addition I presented my study of a very challenging family of square to circle mappings that tend to produce cells that are nearly triangular. It was shown that even on these grids DG is able to achieve excellent convergence for problems where the solution should be smooth. Things are more complicated, however, when one introduces PDEs with discontinuous coefficients. The mapped grid DG method was implemented on an acoustics equation problem with a discontinuous jump in coefficients across a circular interface. Because of the second order error due to approximating the interface one cannot observe high order convergence on this problem. However it was apparent that the higher order methods perform better than a second order method.

In Chapter 4 I discussed and extended a limiting scheme introduced in joint work with Rossmanith and Seal [86]. This limiting scheme only uses an approximation of the extrema values on neighboring cells. These quantities are used to develop a very rudimentary shock-detector and limit cells that appear troubled. The amount of limiting is also determined by this extremal value information. The limiter acts on all moments except the cell average identically. Additionally it does not use directional information in any way. These properties make it a good candidate for mapped grids. This limiter can use any set of variables (characteristic variables, primitive variables, etc.) to define the cell extremal values. It seems that the characteristic variables offer the best performance but satisfactory results were also obtained using primitive variables on the Euler equations. The limiting scheme was examined on a variety of standard test problems and performance comparable to other limiting schemes was observed. One drawback of this limiter is that the shock detector makes use of a variable α that must be set by trial and error. This is actually fairly common in the limiter literature, however. The author is not aware of a high-order limiter that can avoid this for nonlinear problems.

In Section 4.9 the limiter was modified somewhat to use characteristic variables in more than one dimension since the characteristic variables for one-dimensional waves depend on the direction. Essentially the modification involves limiting twice, once using the x- characteristic variables and once using the y-characteristic variables. This modification still needs more testing, especially on unstructured and mapped grids. Additionally, in Section 4.10 I outlined the minor modifications that must be made to the limiter so it can handle mapped grids. The modification is just meant to deal with the fact that the constant mode is no longer orthogonal to the higher order modes due to the Jacobian of the mapping function. Up

to this point computations have been done with RKDG implemented using the DoGPack software package [105]. After Chapter 5 all computations use Lax-Wendroff time-stepping.

Chapter 5 introduced the Lax-Wendroff DG scheme and the differential transform method for computing Taylor series of flux functions. I presented several differential transform identities that are useful for important nonlinear PDEs such as Burgers' equation and the Euler equations. To test the differential transform method a central finite difference scheme was developed in previously unpublished joint work with David Seal. This scheme was stabilized using a PIF-WENO method as discussed in Chapter 6. The relevance of this chapter to the overall thesis was the verification that the differential transform method actually works for automatically computing arbitrary order space-time Taylor series approximations of flux functions from spatial Taylor series. The code in this section is implemented using Finesse [113].

In Chapter 7 I present and discuss a positivity preserving limiter for Lax-Wendroff DG from joint work with Rossmanith and Seal [87]. When simulating the Euler equations (and other PDEs) it is necessary to maintain positive density and pressure. The methods used to maintain positivity in RKDG depend on the SSPRK time stepping schemes. The work in this chapter draws heavily on the technique known as flux corrected transport (FCT). The limiter uses modified flux functions that are a linear combination of a low order flux that guarantees positivity and a high order flux. When using this limiter provable high-order convergence is lost, however the appropriate order of convergence is still observed on smooth problems. Additionally it is important to note that SSPRK methods only exist to fourth order. For higher order time-stepping one would need to use a method similar to the one developed in this chapter. The code in this section is implemented in DoGPack using the Lax-Wendroff discontinuous Galerkin discretization.

In Section 8.2 the block-structured AMR algorithm was outlined as implemented in AMRClaw. In Section 8.3 I introduced the DT-DG scheme as a type of Lax-DG scheme that relies on differential transforms. Chapter 8 also discussed how this method can be implemented with a simple local time-stepping scheme, using a modified version of the AMRClaw code. The local time-stepping scheme involves computing a space-time Taylor series on coarse grid cells that can be interpolated to provide boundary conditions to finer cells. This scheme was tested on a variety of problems in one-dimension as the two-dimensional implementation is incomplete.

In this thesis I have developed the building blocks to create an arbitrary order DT-DG scheme that can be implemented in AMRClaw with local time stepping on mapped grids. RKDG was implemented on mapped grids and shown to perform extremely well even on mappings that produce skewed meshes. Shock-capturing limiters were developed that work on the limited stencil available in AMRClaw to dampen spurious oscillations. Positivity-preserving limiters were developed to maintain positivity when using Lax-Wendroff type DG schemes to solve the Euler equations. The limiters developed also work on mapped grids.

Finally an arbitrary order DT-DG scheme was developed and it was shown that, at least in one dimension, it can be used with block structured AMR with local time stepping. The code used for the computations in this thesis can be found in [85].

9.2 Conclusions

In this thesis I developed a scheme that has the same numerical stencil as the finite volume schemes used in Clawpack. Because of this numerical stencil DG is able to perform well even on skewed mapped grids. When used on non-smooth mappings DG is still able to obtain high-order accuracy because its convergence rate depends on the smoothness of the mapping defined within each cell. In order to develop this minimal stencil DG scheme it was necessary to create a highly-localized high-order limiting scheme. This limiting scheme performs rather well despite using only direct neighbor information, showing that it is possible to obtain good numerical results without limiting schemes that expand a methods numerical stencil.

I also developed a Lax-Wendroff DG scheme using the differential transform method for time stepping (the DT-DG method). The differential transform method has proven to be an effective way to implement Lax-Wendroff time stepping. This DT-DG method has also proven to perform well when used with block-structured AMR as used in AMRClaw. The results obtained in one dimension are extremely promising and this is an approach that should be explored further.

9.3 Future work

9.3.1 Limiters

The shock-capturing limiting scheme introduced in this work has not been fully explored. In particular more exhaustive tests on mapped grids are needed. Additionally, further analysis must be done on how the method performs in multiple dimensions using characteristic variables, especially on unstructured grids. The DT-DG method should also be implemented with the positivity and shock-capturing limiters to study the performance of these limiters on higher order Lax-DG schemes.

9.4 Multidimensional AMR

Implementing DT-DG with AMR in two and three dimensions would be very useful. Theoretically the benefit of AMR goes up even more in higher dimensions. The local time stepping scheme developed in Chapter 8 will have to be modified in two dimensions because the ghost cells will need to provide values at multiple points in space so the ghost cell value cannot be just a constant. However, every other aspect of the DT-DG scheme with AMR seems applicable to multiple dimensions. The series expansions that the differential transform method

depends on can be used in higher dimensions, as seen in the PIF-WENO method introduced in Chapter 6. The AMR should generalize to higher dimensions because AMRClaw was originally written for two dimensional and three dimensional PDEs and only recently adapted to one dimension. Additionally a high-order conservation fix up must be developed. Currently the AMR scheme is nonconservative and it could perform poorly in some situations where there are strong shocks.

9.4.1 Mapped grids with two dimensional hp adaptivity

Once the DT-DG scheme has been implemented in two dimensions it should be adapted to work on mapped grids. This might be difficult because it will likely require performing the DT expansion using the variables ξ and η. This means the PDE must be transformed into mapped coordinates, meaning that the Cauchy-Kowalevski procedure will depend on the mapping itself.

BIBLIOGRAPHY

[1] H. Abbassi, F. Mashayek, and G. Jacobs, *Shock capturing with entropy-based artificial viscosity for staggered grid discontinuous spectral element method*, Comput. & Fluids, 98 (2014), pp. 152–163.

[2] A. Arikoglu and I. Ozkol, *Solution of differential-difference equations by using differential transform method*, Appl. Math. Comput., 181 (2006), pp. 153–162.

[3] A. Arikoglu and I. Ozkol, *Solution of fractional integro-differential equations by using fractional differential transform method*, Chaos Solitons Fractals, 40 (2009), pp. 521–529.

[4] D. Arnold, D. Boffi, and R. Falk, *Approximation by quadrilateral finite elements*, Mathematics of computation, 71 (2002), pp. 909–922.

[5] D. N. Arnold, D. Boffi, and F. Bonizzoni, *Finite element differential forms on curvilinear cubic meshes and their approximation properties*, Numerische Mathematik, (2012), pp. 1–20.

[6] F. Ayaz, *On the two-dimensional differential transform method*, Appl. Math. Comput., 143 (2003), pp. 361–374.

[7] D. S. Balsara and C.-W. Shu, *Monotonicity preserving weighted essentially non-oscillatory schemes with increasingly high order of accuracy*, J. Comput. Phys., 160 (2000), pp. 405–452.

[8] T. Barth and D. Jespersen, *The design and application of upwind schemes on unstructured meshes*, AIAA-89-0366, (1989).

[9] J. B. Bell, P. Colella, and J. A. Trangenstein, *Higher order Godunov methods for general systems of hyperbolic conservation laws*, J. Comput. Phys., 82 (1989), pp. 362–397.

[10] M. J. Berger, D. A. Calhoun, C. Helzel, and R. J. LeVeque, *Logically rectangular finite volume methods with adaptive refinement on the sphere*, Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, 367 (2009), pp. 4483–4496.

[11] M. J. Berger, D. L. George, R. J. LeVeque, and K. T. Mandli, *The geoclaw software for depth-averaged flows with adaptive refinement*, Advances in Water Resources, 34 (2011), pp. 1195–1206.

[12] M. J. Berger and R. J. LeVeque, *Adaptive mesh refinement using wave-propagation algorithms for hyperbolic systems*, SIAM Journal on Numerical Analysis, 35 (1998), pp. 2298–2316.

[13] M. J. Berger and J. Oliger, *Adaptive mesh refinement for hyperbolic partial differential equations*, Journal of computational Physics, 53 (1984), pp. 484–512.

[14] C. Bervillier, *Status of the differential transformation method*, Appl. Math. Comput., 218 (2012), pp. 10158–10170.

[15] P. Bochev, D. Ridzal, G. Scovazzi, and M. Shashkov, *Formulation, analysis and numerical study of an optimization-based conservative interpolation (remap) of scalar fields for arbitrary Lagrangian-Eulerian methods*, J. Comput. Phys., 230 (2011), pp. 5199–5225.

[16] D. Book, *Finite-difference techniques for vectorized fluid dynamics calculations*, New York and Berlin, Springer-Verlag, 1981. 233 p, 1 (1981).

[17] D. Book, J. Boris, and K. Hain, *Flux-corrected transport II: Generalizations of the method*, Journal of Computational Physics, 18 (1975), pp. 248–283.

[18] J. Boris and D. Book, *Flux-corrected transport. I: SHASTA, A fluid transport algorithm that works*, Journal of computational physics, 11 (1973), pp. 38–69.

[19] J. Boris and D. Book, *Flux-corrected transport. III: Minimal-error FCT algorithms*, Journal of Computational Physics, 20 (1976), pp. 397–431.

[20] C. Burstedde, D. Calhoun, K. Mandli, and A. R. Terrel, *Forestclaw: Hybrid forest-of-octrees AMR for hyperbolic conservation laws*, arXiv preprint arXiv:1308.1472, (2013).

[21] D. A. Calhoun and C. Helzel, *A finite volume method for solving parabolic equations on logically cartesian curved surface meshes*, SIAM Journal on Scientific Computing, 31 (2009), pp. 4066–4099.

[22] D. A. Calhoun, C. Helzel, and R. J. LeVeque, *Logically rectangular grids and finite volume methods for PDEs in circular and spherical domains*, SIAM review, 50 (2008), pp. 723–752.

[23] C. K. Chen and S. H. Ho, *Solving partial differential equations by two-dimensional differential transform method*, Applied Mathematics and Computation, 106 (1999), pp. 171 – 179.

[24] C.-L. Chen and Y.-C. Liu, *Differential transformation technique for steady nonlinear heat conduction problems*, Appl. Math. Comput., 95 (1998), pp. 155–164.

[25] A. Christlieb, X. Feng, D. Seal, and Q. Tang, *A high-order positivity-preserving single-stage single-step method for the ideal magnetohydrodynamic equations*, arXiv preprint arXiv:1509.09208, (2015).

[26] A. Christlieb, Y. Güçlü, and D. Seal, *The Picard integral formulation of weighted essentially nonoscillatory schemes*, SIAM J. Numer. Anal., 53 (2015), pp. 1833–1856.

[27] A. Christlieb, Y. Liu, Q. Tang, and Z. Xu, *Positivity-preserving finite difference weno schemes with constrained transport for ideal magnetohydrodynamic equations*. http://arxiv.org/abs/1406.5098, 2014.

[28] A. J. Christlieb, Y. Güçlü, and D. C. Seal, *The Picard integral formulation of weighted essentially nonoscillatory schemes*, SIAM Journal on Numerical Analysis, 53 (2015), pp. 1833–1856.

[29] A. J. Christlieb, Y. Liu, Q. Tang, and Z. Xu, *High order parametrized maximum-principle-preserving and positivity-preserving WENO schemes on unstructured meshes*, J. Comput. Phys., 281 (2015), pp. 334–351, https://doi.org/10.1016/j.jcp.2014.10.029.

[30] Clawpack Development Team, *Clawpack software*, 2017, http://www.clawpack.org. Version 5.4.0.

[31] B. Cockburn, S. Hou, and C.-W. Shu, *The Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws. IV: The multidimensional case*, Math. Comp., 54 (1990), pp. 545–581.

[32] B. Cockburn, S. Lin, and C.-W. Shu, *The Runge-Kutta discontinuous Galerkin method for conservation laws V: Multidimensional systems*, J. Comp. Phys., 141 (1998), pp. 199–224.

[33] B. Cockburn and C.-W. Shu, *TVB Runge-Kutta local projection discontinuous Galerkin finite element method for scalar conservation laws II: General framework*, Math. Comp., 52 (1989), pp. 411–435.

[34] B. Cockburn and C.-W. Shu, *The Runge-Kutta discontinuous Galerkin method for conservation laws. V. Multidimensional systems*, J. Comput. Phys., 141 (1998), pp. 199–224.

[35] B. Cockburn and C.-W. Shu, *Runge-Kutta discontinuous Galerkin methods for convection-dominated problems*, J. Sci. Comput., 16 (2001), pp. 173–261.

[36] Cockburn, B. and Lin, S. Y. and Shu, C.-W., *TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws. III: One-dimensional systems*, J. Comput. Phys., 84 (1989), pp. 90–113.

[37] P. Colella and M. Sekora, *A limiter for PPM that preserves accuracy at smooth extrema*, J. Comput. Phys., 227 (2008), pp. 7069–7076.

[38] M. Dumbser, D. S. Balsara, E. F. Toro, and C.-D. Munz, *A unified framework for the construction of one-step finite volume and discontinuous Galerkin schemes on unstructured meshes*, J. Comput. Phys., 227 (2008), pp. 8209–8253.

[39] M. Dumbser, M. Käser, and E. Toro, *An arbitrary high-order discontinuous Galerkin method for elastic waves on unstructured meshes-V: Local time stepping and p-adaptivity*, Geophysical Journal International, 171 (2007), pp. 695–717.

[40] M. Dumbser and C.-D. Munz, *ADER discontinuous Galerkin schemes for aeroacoustics*, Comptes Rendus Mécanique, 333 (2005), pp. 683–687.

[41] M. Dumbser and C.-D. Munz, *Building blocks for arbitrary high order discontinuous Galerkin schemes*, J. Sci. Comput., 27 (2006), pp. 215–230.

[42] M. Dumbser, O. Zanotti, A. Hidalgo, and D. Balsara, *ADER-WENO finite volume schemes with space-time adaptive mesh refinement*, J. Comput. Phys., 248 (2013), pp. 257–286.

[43] M. Dumbser, O. Zanotti, R. Loubère, and S. Diot, *A posteriori subcell limiting of the discontinuous Galerkin finite element method for hyperbolic conservation laws*, J. Comput. Phys., 278 (2014), pp. 47–75.

[44] G. G. Ev Pukhov, *Differential transforms and circuit theory*, International Journal of Circuit Theory and Applications, 10 (1982), pp. 265–276.

[45] H. Fatoorehchi and H. Abolghasemi, *Improving the differential transform method: A novel technique to obtain the differential transforms of nonlinearities by the Adomian polynomials*, Appl. Math. Model., 37 (2013), pp. 6008–6017.

[46] M. Garg and P. Manohar, *Three-dimensional generalized differential transform method for space-time fractional diffusion equation in two space variables with variable coefficients*, Palest. J. Math., 4 (2015), pp. 127–135.

[47] G. Gassner, M. Dumbser, F. Hindenlang, and C.-D. Munz, *Explicit one-step time discretizations for discontinuous Galerkin and finite volume schemes based on local predictors*, J. Comput. Phys., 230 (2011), pp. 4232–4247.

[48] G. Gassner, F. Hindenlang, and C.-D. Munz, *A Runge-Kutta based discontinuous Galerkin method with time accurate local time stepping*, Advances in Computational Fluid Dynamics, 2 (2011), pp. 95–118.

[49] S. Godunov, *A difference method for numerical calculation of discontinuous solutions of the equations of hydrodynamics*, Mat. Sb., 47 (1959), pp. 271–306.

[50] S. Gottlieb, C.-W. Shu, and E. Tadmor, *Strong stability-preserving high-order time discretization methods*, SIAM review, 43 (2001), pp. 89–112.

[51] H. Grimm-Strele, F. Kupka, and H. J. Muthsam, *Curvilinear grids for WENO methods in astrophysical simulations*, Computer Physics Communications, 185 (2014), pp. 764–776.

[52] W. Guo, J.-M. Qiu, and J. Qiu, *A new Lax–Wendroff discontinuous Galerkin method with superconvergence*, J. Sci. Comput., 65 (2015), pp. 299–326.

[53] A. Harten, *High resolution schemes for hyperbolic conservation laws*, J. Comput. Phys., 49 (1983), pp. 357–393.

[54] A. Harten, *On a class of high resolution total-variation-stable finite-difference schemes*, SIAM Journal on Numerical Analysis, 21 (1984), pp. 1–23.

[55] A. Harten, B. Engquist, S. Osher, and S. R. Chakravarthy, *Uniformly high-order accurate essentially nonoscillatory schemes. III*, J. Comput. Phys., 71 (1987), pp. 231–303.

[56] A. Harten and G. Zwas, *Self-adjusting hybrid schemes for shock computations*, Journal of Computational Physics, 9 (1972), pp. 568–583.

[57] J. S. Hesthaven and T. Warburton, *Nodal discontinuous Galerkin methods: algorithms, analysis, and applications*, vol. 54, Springer, 2007.

[58] H. Hoteit, P. Ackerer, R. Mosé, J. Erhel, and B. Philippe, *New two-dimensional slope limiters for discontinuous Galerkin methods on arbitrary meshes*, Internat. J. Numer. Methods Engrg., 61 (2004), pp. 2566–2593.

[59] M.-J. Jang, C.-L. Chen, and Y.-C. Liu, *Two-dimensional differential transform for partial differential equations*, Appl. Math. Comput., 121 (2001), pp. 261–270.

[60] G.-S. Jiang and C.-W. Shu, *Efficient implementation of weighted ENO schemes*, J. Comput. Phys., 126 (1996), pp. 202–228.

[61] Y. Jiang, C.-W. Shu, and M. Zhang, *An alternative formulation of finite difference weighted ENO schemes with Lax-Wendroff time discretization for conservation laws*, SIAM J. Sci. Comput., 35 (2013), pp. A1137–A1160.

[62] D. Ketcheson, *Highly efficient strong stability-preserving Runge-Kutta methods with low-storage implementations*, SIAM J. Sci. Comput., 30 (2008), pp. 2113–2136.

[63] D. I. Ketcheson and R. J. LeVeque, *Wenoclaw: A higher order wave propagation method*, Hyperbolic problems: theory, numerics, applications, (2008), pp. 609–616.

[64] D. I. Ketcheson, K. Mandli, A. J. Ahmadia, A. Alghamdi, M. Q. de Luna, M. Parsani, M. G. Knepley, and M. Emmett, *Pyclaw: Accessible, extensible, scalable tools for wave propagation problems*, SIAM Journal on Scientific Computing, 34 (2012), pp. C210–C231.

[65] D. I. Ketcheson, M. Parsani, and R. J. LeVeque, *High-order wave propagation algorithms for hyperbolic systems*, SIAM Journal on Scientific Computing, 35 (2013), pp. A351–A377.

[66] J. Kraaijevanger, *Contractivity of Runge-Kutta methods*, BIT, 31 (1991), pp. 482–528.

[67] L. Krivodonova, *Limiters for high-order discontinuous Galerkin methods*, J. Comput. Phys., 226 (2007), pp. 879–896.

[68] E. J. Kubatko, B. A. Yeager, and D. I. Ketcheson, *Optimal strong-stability-preserving Runge-Kutta time discretizations for discontinuous Galerkin methods*, Journal of Scientific Computing, 60 (2014), pp. 313–344.

[69] A. Kurganov and E. Tadmor, *Solution of two-dimensional Riemann problems for gas dynamics without Riemann problem solvers*, Numerical Methods for Partial Differential Equations, 18 (2002), pp. 584–608.

[70] A. Kurnaz, G. Oturanç, and M. E. Kiris, *n-dimensional differential transformation method for solving PDEs*, Int. J. Comput. Math., 82 (2005), pp. 369–380.

[71] D. Kuzmin, *A vertex-based hierarchical slope limiter for p-adaptive discontinuous Galerkin methods*, J. Comput. Appl. Math., 233 (2010), pp. 3077–3085.

[72] D. Kuzmin, *Slope limiting for discontinuous Galerkin approximations with a possibly non-orthogonal Taylor basis*, Internat. J. Numer. Methods Fluids, 71 (2013), pp. 1178–1190.

[73] D. Kuzmin, *Hierarchical slope limiting in explicit and implicit discontinuous Galerkin methods*, J. Comput. Phys., 257 (2014), pp. 1140–1162.

[74] D. Kuzmin and R. Löhner, eds., *Flux-corrected transport*, Scientific Computation, Springer-Verlag, Berlin, 2005. Principles, algorithms, and applications.

[75] P. Lax and B. Wendroff, *Systems of conservation laws*, Comm. Pure Appl. Math., 13 (1960), pp. 217–237.

[76] G. Lemoine, *Numerical modeling of poroelastic-fluid systems using high-resolution finite volume methods*, PhD thesis, 2013.

[77] G. I. Lemoine, M. Y. Ou, and R. J. LeVeque, *High-resolution finite volume modeling of wave propagation in orthotropic poroelastic media*, SIAM Journal on Scientific Computing, 35 (2013), pp. B176–B206.

[78] R. LeVeque, *Wave propagation algorithms for multidimensional hyperbolic systems*, J. Comput. Phys., 131 (1997), pp. 327–353.

[79] R. LeVeque, *Finite volume methods for hyperbolic problems*, Cambridge University Press, 2002.

[80] X.-D. Liu, S. Osher, and T. Chan, *Weighted essentially non-oscillatory schemes*, J. Comput. Phys., 115 (1994), pp. 200–212.

[81] C. Lu and J. Qiu, *Simulations of shallow water equations with finite difference Lax-Wendroff weighted essentially non-oscillatory schemes*, J. Sci. Comput., 47 (2011), pp. 281–302.

[82] H. Luo, J. Baum, and R. Löhner, *A Hermite WENO-based limiter for discontinuous Galerkin method on unstructured grids*, J. Comput. Phys., 225 (2007), pp. 686–713.

[83] I. Men'shov, *Increasing the order of approximation of Godunov's scheme using solutions of the generalized Riemann problem*, USSR Computational Mathematics and Mathematical Physics, 30 (1990), pp. 54 – 65.

[84] K. Michalak and C. Ollivier-Gooch, *Limiters for unstructured higher-order accurate solutions of the Euler equations*, AIAA 2008-776, (2008).

[85] S. Moe, 2017, `https://doi.org/10.5281/zenodo.803506`.

[86] S. A. Moe, J. A. Rossmanith, and D. C. Seal, *A simple and effective high-order shock-capturing limiter for discontinuous Galerkin methods*, arXiv preprint arXiv:1507.03024, (2015), `http://arxiv.org/abs/1507.03024`.

[87] S. A. Moe, J. A. Rossmanith, and D. C. Seal, *Positivity-preserving discontinuous Galerkin methods with Lax-Wendroff time discretizations*, Journal of Scientific Computing, (2016), pp. 1–27.

[88] G. Montecinos, C. E. Castro, M. Dumbser, and E. F. Toro, *Comparison of solvers for the generalized Riemann problem for hyperbolic systems with source terms*, J. Comput. Phys., 231 (2012), pp. 6472–6494.

[89] M. R. Norman, *A WENO-limited, ADER-DT, finite-volume scheme for efficient, robust, and communication-avoiding multi-dimensional transport*, J. Comput. Phys., 274 (2014), pp. 1–18.

[90] M. R. Norman, *Hermite WENO limiting for multi-moment finite-volume methods using the ADER-DT time discretization for 1-D systems of conservation laws*, J. Comput. Phys., 282 (2015), pp. 381–396.

[91] M. R. Norman and H. Finkel, *Multi-moment ADER-Taylor methods for systems of conservation laws with source terms in one dimension*, J. Comput. Phys., 231 (2012), pp. 6622–6642.

[92] Z. Odibat and S. Momani, *A generalized differential transform method for linear partial differential equations of fractional order*, Appl. Math. Lett., 21 (2008), pp. 194–199.

[93] P.-O. Persson and J. Peraire, *Sub-cell shock capturing for discontinuous Galerkin methods*, AIAA 2006-112, (2006).

[94] B. Perthame and C.-W. Shu, *On positivity preserving finite volume schemes for Euler equations*, Numer. Math., 73 (1996), pp. 119–130.

[95] G. È. Pukhov, *Expansion formulas for differential transforms*, Cybernetics, 17 (1981).

[96] J. Qiu, *A numerical comparison of the Lax-Wendroff discontinuous Galerkin method based on different numerical fluxes*, J. Sci. Comput., 30 (2007), pp. 345–367.

[97] J. Qiu, M. Dumbser, and C.-W. Shu, *The discontinuous Galerkin method with Lax-Wendroff type time discretizations*, Comput. Methods Appl. Mech. Eng., 194 (2005), pp. 4528–4543.

[98] J. Qiu and C.-W. Shu, *Finite difference WENO schemes with Lax-Wendroff-type time discretizations*, SIAM J. Sci. Comput., 24 (2003), pp. 2185–2198.

[99] J. Qiu and C.-W. Shu, *Hermite WENO schemes and their application as limiters for Runge-Kutta discontinuous Galerkin method: One-dimensional case*, J. Comput. Phys., 193 (2004), pp. 115–135.

[100] J. Qiu and C.-W. Shu, *A comparison of troubled-cell indicators for Runge-Kutta discontinuous Galerkin methods using weighted essentially nonoscillatory limiters*, SIAM J. Sci. Comput., 27 (2005), pp. 995–1013.

[101] J. Qiu and C.-W. Shu, *Hermite WENO schemes and their application as limiters for Runge-Kutta discontinuous Galerkin method. II: Two dimensional case*, Comput. & Fluids, 34 (2005), pp. 642–663.

[102] J. Qiu and C.-W. Shu, *Runge-Kutta discontinuous Galerkin method using WENO limiters*, SIAM J. Sci. Comput., 26 (2005), pp. 907–929.

[103] W. Reed and T. Hill, *Triangular mesh methods for the neutron transport equation*, Tech. Report LA-UR-73-479, Los Alamos Scientific Laboratory, 1973.

[104] A. Rodionov, *Methods of increasing the accuracy in Godunov's scheme*, USSR Computational Mathematics and Mathematical Physics, 27 (1987), pp. 164 – 169.

[105] J. Rossmanith, DoGPack *software*, 2015. Available from `http://www.dogpack-code.org`.

[106] V. Rusanov, *Calculation of interaction of non-steady shock waves with obstacles*, J. Comp. Math. Phys. USSR, 1 (1961), pp. 267–279.

[107] S. Ruuth and R. Spiteri, *Two barriers on strong-stability-preserving time discretization methods*, in Proceedings of the Fifth International Conference on Spectral and High Order Methods (ICOSAHOM-01) (Uppsala), vol. 17, 2002, pp. 211–220.

[108] S. J. Ruuth and R. J. Spiteri, *High-order strong-stability-preserving Runge-Kutta methods with downwind-biased spatial discretizations*, SIAM Journal on Numerical Analysis, 42 (2004), pp. 974–996.

[109] V. L. Rvachëv, G. P. Man′ko, and V. V. Fed′ko, *Technology of differentiating functions of many variables on an electronic computer*, Kibernetika (Kiev), (1983), pp. 31–33, https://doi.org/10.1007/BF01068756.

[110] C. Schulz-Rinne, J. Collins, and H. Glaz, *Numerical solution of the Riemann problem for two-dimensional gas dynamics*, SIAM J. Sci. Comput., 14 (1993), pp. 1394–1414.

[111] P. Sconzo, *Contribution to the solution of the three-body problem in power series form*, Astronomische Nachrichten, 290 (1967), pp. 163–170.

[112] D. Seal, *Discontinuous Galerkin methods for Vlasov models of plasma*, PhD thesis, University of Wisconsin-Madison, Madison, WI, 2012.

[113] D. Seal, FINESS *software*, 2015. Available from https://bitbucket.org/dseal/finess.

[114] D. C. Seal, Q. Tang, Z. Xu, and A. J. Christlieb, *An explicit high-order single-stage single-step positivity-preserving finite difference WENO method for the compressible Euler equations*, arXiv preprint arXiv:1411.0328, (2014).

[115] L. Sedov, *Similarity and dimensional methods in mechanics*, Academic Press, New York-London, 1959.

[116] C.-W. Shu, *High-order finite difference and finite volume WENO schemes and discontinuous Galerkin methods for CFD*, International Journal of Computational Fluid Dynamics, 17 (2003), pp. 107–118.

[117] C.-W. Shu, *High order WENO and DG methods for time-dependent convection-dominated PDEs: A brief survey of several recent developments*, J. Comput. Phys., 316 (2016), pp. 598–613.

[118] C.-W. Shu and S. Osher, *Efficient implementation of essentially nonoscillatory shock-capturing schemes. II*, J. Comput. Phys., 83 (1989), pp. 32–78.

[119] G. Sod, *A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws*, J. Computational Phys., 27 (1978), pp. 1–31.

[120] P. Sweby, *High resolution schemes using flux limiters for hyperbolic conservation laws*, SIAM J. Numer. Anal., 21 (1984), pp. 995–1011.

[121] E. Tadmor, *Convenient total variation diminishing conditions for nonlinear difference schemes*, SIAM J. Numer. Anal., 25 (1988), pp. 1002–1014.

[122] A. Taghavi, A. Babaei, and A. Mohammadpour, *Application of reduced differential transform method for solving nonlinear reaction-diffusion-convection problems*, Appl. Appl. Math., 10 (2015), pp. 162–170.

[123] A. Taube, M. Dumbser, D. S. Balsara, and C.-D. Munz, *Arbitrary high-order discontinuous Galerkin schemes for the magnetohydrodynamic equations*, J. Sci. Comput., 30 (2007), pp. 441–464.

[124] V. A. Titarev and E. F. Toro, *ADER: arbitrary high order Godunov approach*, in Proceedings of the Fifth International Conference on Spectral and High Order Methods (ICOSAHOM-01) (Uppsala), vol. 17, 2002, pp. 609–618.

[125] E. Toro and M. Dumbser, *ADER schemes for time-dependent PDEs: review and applications to astrophysics*, in Numerical Modeling of Space Plasma Flows, Astronum-2009, vol. 429, 2010, p. 281.

[126] E. F. Toro and V. A. Titarev, *Solution of the generalized Riemann problem for advection-reaction equations*, R. Soc. Lond. Proc. Ser. A Math. Phys. Eng. Sci., 458 (2002), pp. 271–281.

[127] E. F. Toro and V. A. Titarev, *ADER schemes for scalar non-linear hyperbolic conservation laws with source terms in three-space dimensions*, J. Comput. Phys., 202 (2005), pp. 196–215.

[128] E. F. Toro and V. A. Titarev, *TVD fluxes for the high-order ADER schemes*, J. Sci. Comput., 24 (2005), pp. 285–309.

[129] E. F. Toro and V. A. Titarev, *Derivative Riemann solvers for systems of conservation laws and ADER methods*, J. Comput. Phys., 212 (2006), pp. 150–165.

[130] I. Tsukanov and M. Hall, *Data structure and algorithms for fast automatic differentiation*, International Journal for Numerical Methods in Engineering, 56 (2003), pp. 1949–1972.

[131] I. Tsukanov, M. Hall, and I. T. . M. Hall, *Fast forward automatic differentiation library (FFADLib): A user manual*, tech. report, 2000.

[132] P. Ullrich and M. Norman, *The flux-form semi-Lagrangian spectral element (FF–SLSE) method for tracer transport*, Quarterly Journal of the Royal Meteorological Society, 140 (2014), pp. 1069–1085.

[133] B. van Leer, *Towards the ultimate conservative difference scheme I. The quest of monotonicity*, in Proceedings of the Third International Conference on Numerical Methods in Fluid Mechanics, H. Cabannes and R. Temam, eds., vol. 18, 1973, pp. 163–168.

[134] B. van Leer, *Towards the ultimate conservative difference scheme. II: Monotonicity and conservation combined in a second-order scheme*, J. Comput. Phys., 14 (1974), pp. 361–370.

[135] J. von Neumann and R. Richtmyer, *A method for the numerical calculation of hydrodynamic shocks*, J. Applied Physics, 21 (1950), pp. 232–237.

[136] P. Woodward and P. Colella, *The numerical simulation of two-dimensional fluid flow with strong shocks*, J. Comput. Phys., 54 (1984), pp. 115–173.

[137] T. Xiong, J.-M. Qiu, and Z. Xu, *A parametrized maximum principle preserving flux limiter for finite difference RK-WENO schemes with applications in incompressible flows*, J. Comput. Phys., 252 (2013), pp. 310–331.

[138] T. Xiong, J.-M. Qiu, and Z. Xu, *High-order maximum-principle-preserving discontinuous Galerkin method for convection-diffusion equations*, SIAM J. Sci. Comput., 37 (2015), pp. 583–608.

[139] M. Yang and Z. Wang, *A parameter-free generalized moment limiter for high-order methods on unstructured grids*, Adv. Appl. Math. Mech., 1 (2009), pp. 451–480.

[140] J. Yu and C. Yan, *An artificial diffusivity discontinuous Galerkin scheme for discontinuous flows*, Comput. & Fluids, 75 (2013), pp. 56–71.

[141] S. Zalesak, *The design of flux-corrected transport (FCT) algorithms for structured grids*, in Flux-corrected transport, Sci. Comput., Springer, Berlin, 2005, pp. 29–78.

[142] X. Zhang and C.-W. Shu, *On positivity preserving high order discontinuous Galerkin schemes for compressible Euler equations on rectangular meshes*, J. Comp. Phys., 229 (2010), pp. 8918—8934.

[143] X. Zhang and C.-W. Shu, *Maximum-principle-satisfying and positivity-preserving high-order schemes for conservation laws: survey and new developments*, Proc. R. Soc. A, 467 (2011), pp. 2752–2776.

[144] X. Zhang and C.-W. Shu, *A minimum entropy principle of high order schemes for gas dynamics equations*, Numerische Mathematik, 121 (2012), pp. 545–563.

[145] X. Zhang, Y. Xia, and C.-W. Shu, *Maximum-principle-satisfying and positivity-preserving high order discontinuous Galerkin schemes for conservation laws on triangular meshes*, J. Sci. Comput., 50 (2012), pp. 29–62.

[146] H. Zheng, Z. Zhang, and E. Liu, *Non-linear seismic wave propagation in anisotropic media using the flux-corrected transport technique*, Geophysical Journal International, 165 (2006), pp. 943–956.

[147] J. Zhu and J. Qiu, *WENO schemes and their application as limiters for RKDG methods based on trigonometric approximation spaces*, J. Sci. Comput., 55 (2013), pp. 606–644.

[148] J. Zhu, J. Qiu, C.-W. Shu, and M. Dumbser, *Runge-Kutta discontinuous Galerkin method using WENO limiters. II: Unstructured meshes*, J. Comput. Phys., 227 (2008), pp. 4330–4353.

[149] J. Zhu, X. Zhong, C.-W. Shu, and J. Qiu, *Runge-Kutta discontinuous Galerkin method using a new type of WENO limiters on unstructured meshes*, J. Comput. Phys., 248 (2013), pp. 200–220.