A parallel, adaptive discontinuous Galerkin method for hyperbolic problems on unstructured meshes

by

Andrew Giuliani

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Applied Mathematics

Waterloo, Ontario, Canada, 2018

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

Supervisor(s):	Lilia Krivodonova Professor, Dept. of Applied Mathematics, University of Waterloo
Internal Member:	Marek Stastna Professor, Dept. of Applied Mathematics, University of Waterloo
Internal Member:	Sander Rhebergen Professor, Dept. of Applied Mathematics, University of Waterloo
Internal-External Member:	Christopher Batty Professor, Dept. of Computer Science, University of Waterloo
External-External(s):	Clinton Groth Professor, Institute for Aerospace Studies, University of Toronto

Abstract

This thesis is concerned with the parallel, adaptive solution of hyperbolic conservation laws on unstructured meshes.

First, we present novel algorithms for cell-based adaptive mesh refinement (AMR) on unstructured meshes of triangles on graphics processing units (GPUs). Our implementation makes use of improved memory management techniques and a coloring algorithm for avoiding race conditions. The algorithm is entirely implemented on the GPU, with negligible communication between device and host. We show that the overhead of the AMR subroutines is small compared to the high-order solver and that the proportion of total run time spent adaptively refining the mesh decreases with the order of approximation. We apply our code to a number of benchmarks as well as more recently proposed problems for the Euler equations that require extremely high resolution. We present the solution to a shock reflection problem that addresses the von Neumann triple point paradox. We also study the problem of shock disappearance and self-similar diffraction of weak shocks around thin films.

Next, we analyze the stability and accuracy of second-order limiters for the discontinuous Galerkin method on unstructured triangular grids. We derive conditions for a limiter such that the numerical solution preserves second order accuracy and satisfies the local maximum principle. This leads to a new measure of cell size that is approximately twice as large as the radius of the inscribed circle. It is shown with numerical experiments that the resulting bound on the time step is tight. We also consider various combinations of limiting points and limiting neighborhoods and present numerical experiments comparing the accuracy, stability, and efficiency of the corresponding limiters.

We show that the theory for strong stability preserving (SSP) time stepping methods employed with the method of lines-type discretizations of hyperbolic conservation laws may result in overly stringent time step restrictions. We analyze a fully discrete finite volume method with slope reconstruction and a second order SSP Runge-Kutta time integrator to show that the maximum stable time step can be increased over the SSP limit. Numerical examples show that this result extends to two-dimensional problems on triangular meshes.

Finally, we propose a moment limiter for the discontinuous Galerkin method applied to hyperbolic conservation laws in two and three dimensions. The limiter works by finding directions in which the solution coefficients can be separated and limits them independently of one another by comparing to forward and backward reconstructed differences. The limiter has a precomputed stencil of constant size, which provides computational advantages in terms of implementation and runtime. We provide examples that demonstrate stability and second order accuracy of solutions.

Acknowledgments

I would like to thank my supervisor, Lilia, who has been a friend and mentor. I also would like to thank my sisters, Alyssa and Sara, for their friendship and support. Finally, I would like to thank the members of the Examining Committee: Marek Stastna, Sander Rhebergen, Christopher Batty, and Clinton Groth.

Dedication

To my parents, Marco and Sandra, who taught me by example the value of hard work. This thesis is a product of their guidance, love, and encouragement.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

This thesis is concerned with the development of robust, parallel algorithms for the numerical approximation of solutions to hyperbolic conservation laws in two and three dimensions using the discontinuous Galerkin (DG) method. Hyperbolic conservation laws are partial differential equations (PDEs) that model wave propagation and have applications in computational fluid dynamics (CFD). Weak solutions of such equations admit discontinuities, which can be difficult to approximate numerically. Therefore, the development of reliable numerical methods for this class of PDEs is important for many applied problems in CFD and engineering. Popular methods to solve hyperbolic PDEs include finite difference and finite volume methods with high order reconstructions, e.g., ENO and WENO schemes. These methods attain high order accuracy using an order-dependent stencil and can be unwieldy on complex geometries and unstructured meshes. In contrast, the DG method presents a number of advantages. First, the method is of arbitrarily high order and has a compact, order-independent stencil. It is also suitable for execution on parallel architectures since an element only requires information about itself and its neighbors. Second, it can be defined on different element geometries, e.g. triangles, quadrilaterals, and on complex domains. Finally, it is straightforward to use this method with adaptive mesh refinement (AMR) algorithms that refine or coarsen elements in the mesh or modify an element's order of approximation.

In this thesis, we have focused on (1) the development and implementation of AMR algorithms parallelized on graphics processing units (GPUs) and on (2) the development and analysis of novel limiting techniques for the DG method on unstructured meshes of triangles and tetrahedra. GPUs are popular for the parallelization of numerical solvers for PDEs [3–6] due to their low cost and impressive compute capabilities. Adaptive, unstructured computational fluid dynamics (CFD) solvers on GPUs must leverage the high floating point operation (FLOP) throughput available by optimizing memory transfers and reducing latencies [3,7]. We propose and implement novel algorithms for cell-based adaptive mesh refinement on unstructured meshes of triangles on graphics processing units. Our implementation makes use of improved memory management techniques and a coloring algorithm for avoiding race conditions. Using our GPU-accelerated AMR algorithm, we solve a number of problems in gas dynamics that are intractable without some form of adaptive mesh refinement. In particular, we provide numerical evidence in support of Guderley Mach reflection, which requires element sizes on the order of 10^{-6} and meshes comprising over 5 million elements. We also solve a problem concerning the interaction of a shock with a thin, reflecting film and determine the location of shock disappearance.

For nonlinear conservation laws, a stabilization procedure, such as slope limiting, is required to prevent instabilities that can occur in the presence of discontinuities in the numerical solution. Limiters from the finite volume framework can sometimes be modified to act on DG solutions, though there are limiters devised to stabilize DG solutions specifically [8,9]. These limiters compare the DG solution values on the edges to values reconstructed from averages on neighboring elements. In this thesis, we analyze the stability and accuracy of second-order limiters for the discontinuous Galerkin method on unstructured triangular meshes. We also present a limiter that can be viewed as a first step in the generalization of the moment limiter in [10,11] to unstructured meshes, or as a standalone second order limiter with proven stability and accuracy properties. It is a lightweight and simple limiting procedure that is composed of two independent one-dimensional limiters. The implementation of the limiter is easily parallelizable and straightforward as it uses the minmod function to compare the solution coefficients to suitable forward and backward differences.

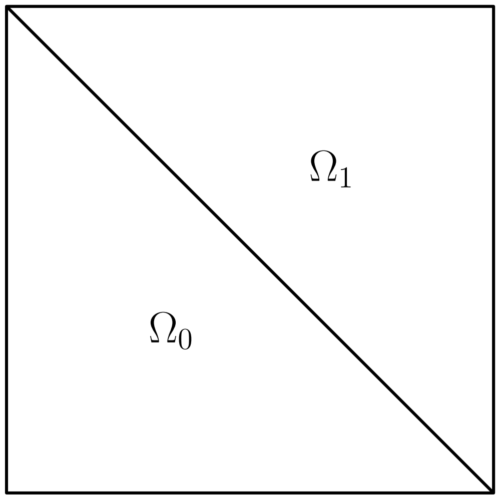
We now describe the discontinuous Galerkin method in two and three dimensions, give an overview of adaptive mesh refinement techniques, provide a description of reflection problems in gas dynamics, and finally, describe limiters in the context of the DG method.

1.1 The discontinuous Galerkin method

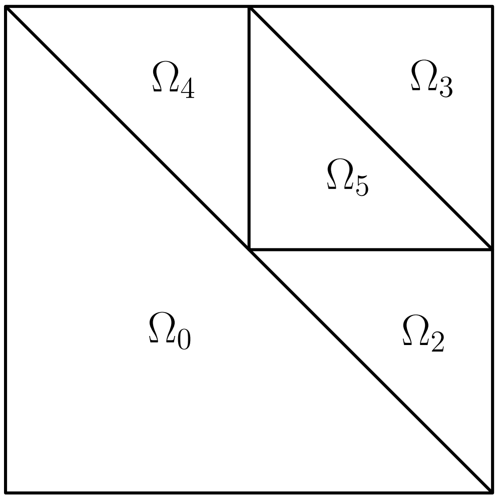
Hyperbolic conservation laws are partial differential equations of the form

$$\mathbf{u}_t + \nabla \cdot \mathbf{F}(\mathbf{u}) = 0, \tag{1.1}$$

with the solution $\mathbf{u}(\mathbf{x}, t) = (u_1, u_2, ..., u_M)^\intercal$ defined on $\Omega \times [0, T]$ such that $\mathbf{x} \in \Omega \subset \mathbb{R}^d$, T is the final time and $\mathbf{F}(\mathbf{u})$ is the flux function. Additionally, the initial condition along



(a) Conforming mesh of two triangles.



(b) Nonconforming mesh of five triangles.

Figure 1.1: Possible meshes of a square domain.

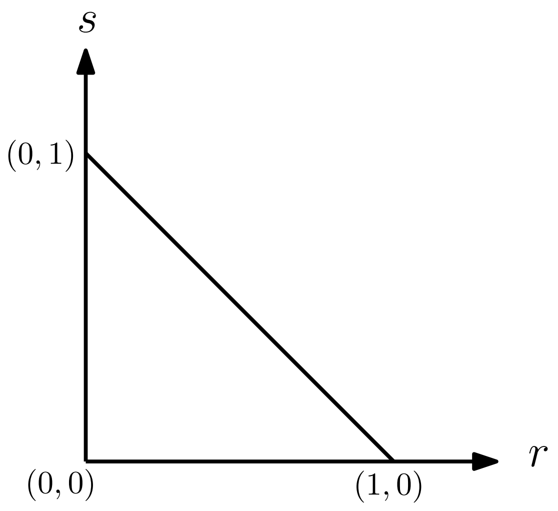
with appropriate boundary conditions are prescribed. In this thesis, we consider two- and three-dimensional hyperbolic PDEs, i.e., $d = 2, 3$.

The discontinuous Galerkin method can be formulated by first dividing the domain Ω into an unstructured mesh of triangles or tetrahedra such that $\Omega = \bigcup_i \Omega_i$. Typically, a mesh produced by a mesh generator is conforming (Figure 1.1a). However, an adaptive mesh refinement algorithm can split an element into four smaller triangles by connecting edges' midpoints, which may produce a nonconforming mesh (Figure 1.1b). The weak form of the conservation law is obtained by multiplying equation (1.1) by a test function $v \in H^1(\Omega_i)$ and integrating on element Ω_i. After applying the divergence theorem, we obtain

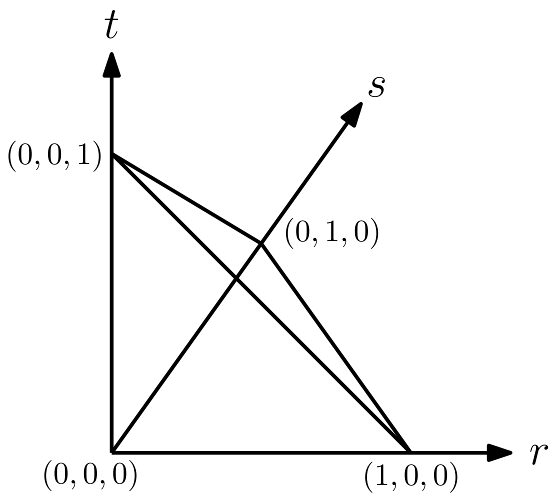
$$\int_{\Omega_i} \mathbf{u}_t v d\mathbf{x} - \int_{\Omega_i} \mathbf{F}(\mathbf{u}) \cdot \nabla v d\mathbf{x} + \int_{\partial\Omega_i} v\mathbf{F}(\mathbf{u}) \cdot \mathbf{n} dl = 0,\ \forall v \in H^1(\Omega_i), \tag{1.2}$$

where $\mathbf{n}$ is the unit outward facing normal on the element's boundary $\partial\Omega_i$. In two dimensions, each element Ω_i is mapped to the canonical triangle Ω_c, having vertices at $(0,0), (1,0), (0,1)$, using the transformation

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x_{i,1} & x_{i,2} & x_{i,3} \\ y_{i,1} & y_{i,2} & y_{i,3} \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1-r-s \\ r \\ s \end{pmatrix}, \tag{1.3}$$



(a) Canonical triangle.



(b) Canonical tetrahedron.

Figure 1.2: Canonical elements Ω_c.

where $(x_i, y_i)_{1,2,3}$ are the vertices of Ω_i in the physical space (Figure 1.2a). We label the edge defined by $(0,0)$ and $(1,0)$ of the canonical triangle edge 1, $(1,0)$ and $(0,1)$ edge 2, and $(0,1)$ and $(0,0)$ edge 3. The Jacobian of the transformation is

$$J_i = \begin{pmatrix} x_{i,2} - x_{i,1} & x_{i,3} - x_{i,1} \\ y_{i,2} - y_{i,1} & y_{i,3} - y_{i,1} \end{pmatrix}. \tag{1.4}$$

In three dimensions, each element Ω_i is mapped to the canonical tetrahedron Ω_c, having vertices at $(0,0,0),(1,0,0),(0,1,0),(0,0,1)$, using the transformation

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x_{i,1} & x_{i,2} & x_{i,3} & x_{i,4} \\ y_{i,1} & y_{i,2} & y_{i,3} & y_{i,4} \\ z_{i,1} & z_{i,2} & z_{i,3} & z_{i,4} \\ 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1-r-s-t \\ r \\ s \\ t \end{pmatrix}, \tag{1.5}$$

where $(x_i, y_i, z_i)_{1,\dots,4}$ are the vertices of Ω_i in the physical space (Figure 1.2b). We label the face opposite vertex $(1,0,0)$ face 1, the face opposite vertex $(0,1,0)$ face 2, the face opposite vertex $(0,0,1)$ face 3, and the face opposite the vertex $(0,0,0)$ face 4 (Figure

1.2b). The Jacobian of the transformation is

$$J_i = \begin{pmatrix} x_{i,2} - x_{i,1} & x_{i,3} - x_{i,1} & x_{i,4} - x_{i,1} \\ y_{i,2} - y_{i,1} & y_{i,3} - y_{i,1} & y_{i,4} - y_{i,1} \\ z_{i,2} - z_{i,1} & z_{i,3} - z_{i,1} & z_{i,4} - z_{i,1} \end{pmatrix}. \tag{1.6}$$

We define $S^p(\Omega_c)$ to be the space of polynomials of order up to p on Ω_c, and $\{\varphi_k\}_{k=0,\cdots,N_p^d-1}$ to be the set of orthonormal basis functions on $S(\Omega_c)$ [12, 13], where the number of basis functions N_p^d for the space of order p in d spatial dimensions is

$$N_p^d = \begin{cases} \frac{1}{2}(p+1)(p+2) \text{ if } d = 2, \\ \frac{1}{6}(p+1)(p+2)(p+3) \text{ if } d = 3. \end{cases}$$

The linear basis in two dimensions is

$$\begin{aligned} \varphi_0 &= \sqrt{2}, \\ \varphi_1 &= -2 + 6r, \\ \varphi_2 &= -2\sqrt{3} + 2\sqrt{3}r + 4\sqrt{3}s, \end{aligned} \tag{1.7}$$

and in three dimensions is

$$\begin{aligned} \varphi_0 &= \sqrt{6}, \\ \varphi_1 &= -\sqrt{10} + 4\sqrt{10}r, \\ \varphi_2 &= -2\sqrt{5} + 2\sqrt{5}r + 6\sqrt{5}s, \\ \varphi_3 &= -2\sqrt{15} + 2\sqrt{15}r + 2\sqrt{15}s + 4\sqrt{15}t. \end{aligned} \tag{1.8}$$

The exact solution on element Ω_i is approximated by $\mathbf{U}_i$, which is a linear combination of the basis functions φ_k, i.e. $\mathbf{U}_i = \sum_{k=0}^{N_p^d-1} \mathbf{c}_{i,k}\varphi_k$, where $\mathbf{c}_{i,k} = [c_{i,k}^1, c_{i,k}^2, \ldots, c_{i,k}^m, \ldots, c_{i,k}^M]^\intercal$ are referred to as the degrees of freedom (DOFs). As continuity between elements is not imposed, the solution is multivalued in the boundary integral. We therefore introduce a numerical flux $\mathbf{F}^*(\mathbf{U}_i, \mathbf{U}_j)$ to allow information exchange between adjacent cells Ω_i and Ω_j. We assume that the numerical flux is consistent, monotone, and differentiable. With

v chosen to be φ_k, equation (1.2) now becomes

$$\frac{d}{dt}\mathbf{c}_{i,k} = \frac{1}{\det J_i}\Bigg(\int_{\Omega_c} \mathbf{F}(\mathbf{U}_i)\cdot(\nabla\varphi_k J_i^{-1})\ \det J_i\, d\mathbf{x} \\ - \sum_{j\in N_i^e, j\neq i}\int_{\partial\Omega_{i,j}} \varphi_k \mathbf{F}^*(\mathbf{U}_i,\mathbf{U}_j)\cdot\mathbf{n}_{i,j}\, dl\Bigg), \quad k = 0,\cdots,N_p^d - 1, \quad (1.9)$$

where N_i^e is the set of indices of Ω_i and of elements that share an interface with Ω_i, $\partial\Omega_{i,j}$ is the interface shared by Ω_i and Ω_j, and $\mathbf{n}_{i,j}$ is the outward pointing unit normal on that interface. In two dimensions, $\partial\Omega_{i,j}$ is a linear segment, i.e. an edge and in three dimensions, it is a triangle, i.e. a face. $\partial\Omega_{i,j}$ is also referred to as e_s, where s is the index of the edge or face. We use numerical quadrature rules of order $2p$ and $2p+1$ to evaluate the volume and surface integrals in (1.9), respectively [9]. The system of equations (1.9) can be solved in time using a standard ordinary differential equation (ODE) solver of order $p+1$, e.g. a Runge-Kutta (RK) method.

As the volume and surface integral contributions can be computed cell-by-cell and face-by-face, respectively, the method is predisposed to applications on highly parallel GPU architectures [3]. We now give an overview of the application programming interface (API), CUDA, with which the DG method can be implemented on NVIDIA GPUs.

1.2 CUDA

NVIDIA's CUDA (Compute Unified Device Architecture) is an API for general purpose GPU computing on NVIDIA GPUs. The CPU, named the host, directs the GPU, named the device, through API calls that transfer memory and execute parallel algorithms, called kernels. Kernels are executed by *threads* in parallel, or in lock-step, i.e. in a Single Instruction Multiple Data (SIMD) fashion. That is, the same instruction set is executed on different data units simultaneously.

There are many considerations that must be taken into account when designing a kernel. The programmer is tasked with mapping data elements to threads such that memory contention (race conditions) is avoided, while also ensuring memory is accessed in an efficient manner for optimal (coalesced) memory transfers.

Warps are collections of 32 threads. They are grouped into *blocks* of a size determined by the programmer. The entire collection of blocks constitutes all the parallel elements of a kernel. Though it is possible to implement a synchronization point across the threads in a specific block, the only way to synchronize across all threads in a kernel is by exiting the kernel. Upon kernel termination, the programmer is guaranteed that all threads have completed their respective work. This is crucial to know when designing kernels that may present race conditions.

There is a complex memory hierarchy on NVIDIA GPUs that we only briefly summarize here. Global memory is shared among threads and is located in GPU video memory (DRAM). Modern NVIDIA GPU platforms, such as the NVIDIA Tesla K40, have 12 GB of global memory available. Thread local memory is located on low-latency registers, or, if the registers are full, there is spillage into global memory. The programmer can minimize the effect of high latency global memory accesses through efficient data access patterns, i.e. coalesced reads and writes, and minimization of the number of reads and writes to global memory. An optimized data flow on the GPU entails a coalesced load of data from global memory, manipulation of these data in the registers, then a coalesced write back to global memory.

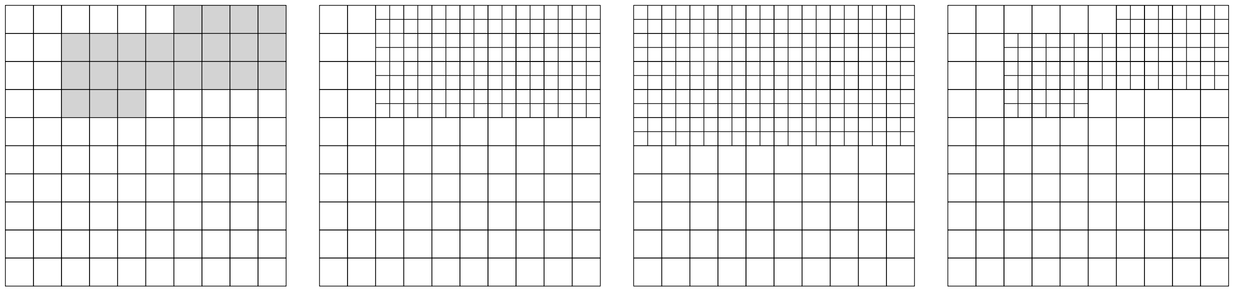
1.3 Adaptive mesh refinement

AMR is a technique that modifies the mesh in order to efficiently distribute computational resources over the domain. Common types of adaptivity include anisotropic adaptivity, p-, and h-refinement. Anisotropic adaptivity spatially relocates, or smooths, the geometrical nodes of the mesh [14]. P-refinement strategies aim to increase the local degree of approximation in smooth regions of the solution [15, 16]. Finally, h-refinement strategies enrich the mesh locally with new elements in order to capture fine structures of the solution or shocks.

Implementations of AMR are numerous and include PARAMESH [17], Chombo [18], deal.ii [19], and AMRClaw [20]. H-adaptivity has been implemented as patch-, block-, or cell-based refinement.

The idea behind patch-based refinement, or component grids, is to superimpose progressively refined Cartesian grids until the desired accuracy is obtained [21] (Figure 1.3b). Subgrids communicate with one another and are advanced in time using local time stepping.

In block-based refinement, a predefined number of elements is grouped together into blocks [22, 23]. Refinement and coarsening operations execute on blocks of cells, rather than individual elements (Figure 1.3c). Only inter-block connectivity is required since the blocks are scaled versions of one another.



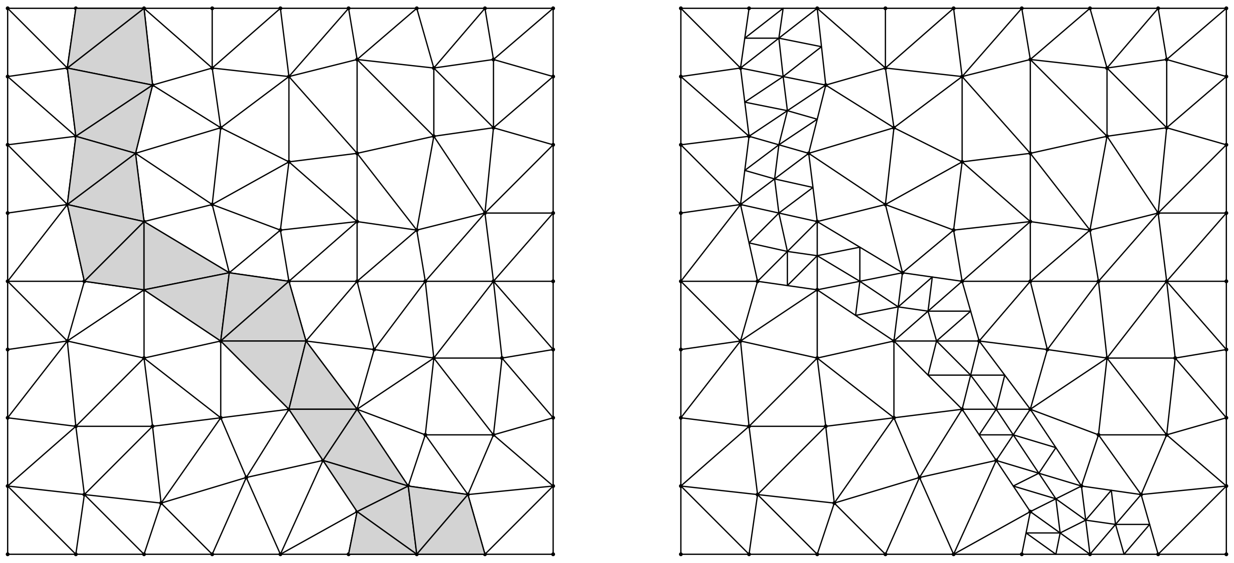
(a) Initial mesh. Shaded elements are flagged for refinement.

(b) Patch-based refinement.

(c) Block-based refinement.

(d) Cell-based refinement.

Figure 1.3: Patch-, block-, and cell-based refinement strategies on regular grids.



(a) Initial unstructured mesh of triangles. Shaded triangles are flagged for refinement.

(b) Cell-based refinement.

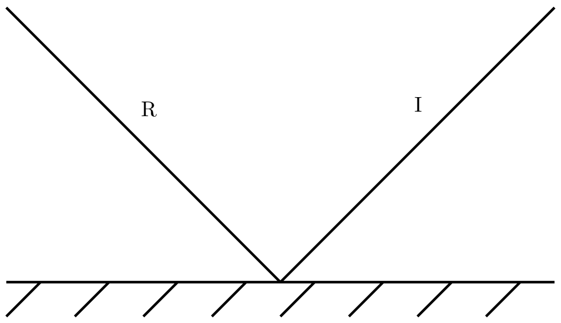
Figure 1.4: Cell-based refinement on an unstructured mesh of triangles.

In cell-based h-refinement, cells are refined independently of one another, which requires more connectivity data than block-based refinement (Figure 1.3d). Usually, parent-child relations between coarse and fine elements are organized into a quadtree or octree data structure for two- and three-dimensional codes [24], respectively. The advantage of this approach is that fewer elements may be needed for a prescribed error tolerance and that it is well suited to unstructured meshes. We focus on cell based h-adaptivity on unstructured meshes of triangles in this thesis (Figure 1.4), though the work we present here generalizes to other AMR strategies.

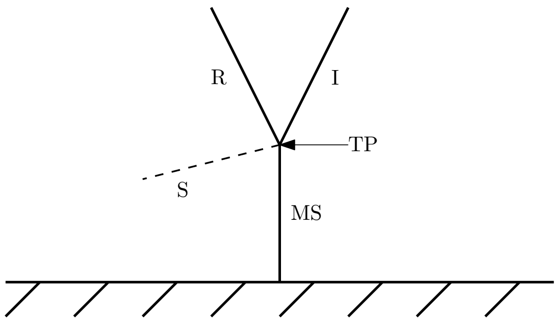
Cell-based h-adaptivity has been used extensively on serial and parallel CPU architectures in CFD codes, e.g., [25–27]. However, GPU architectures present their own challenges. During AMR, elements and their corresponding data may be added or removed from the mesh. Updating the data arrays can lead to memory management issues, e.g., ensuring that arrays contain contiguous information without excessive copying.

1.4 Reflection problems in gas dynamics

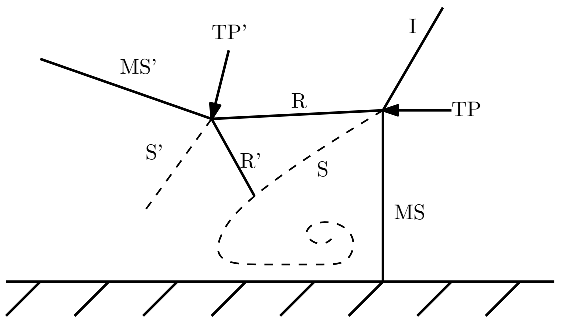
There are a number of reflection problems in gas dynamics that are intractable without some form of adaptive mesh refinement. In this thesis, we consider a number of AMR benchmarks as well as two more challenging problems. The first problem is resolving Guderley Mach reflection and the second is resolving the shock disappearance point in a diffraction problem. Guderley Mach reflection has previously been simulated on manually constructed, logically Cartesian grids. Here we present fully adaptive computations on an unstructured mesh, which allows us to obtain a more accurate position of the triple point and contribute to the body of numerical evidence for Guderley's solution. There are several self-similar reflection patterns that can result from the oblique reflection of a shock against a wedge. Regular reflection occurs when the incident (I) and reflected (R) shocks meet at the wall (Figure 1.5a). Single Mach reflection occurs when the point at which the incident and reflected shocks meet detaches from the wall (Figure 1.5b). This point is called the triple point (TP) and is connected to the wall via the Mach stem (MS). A slipline (S) also originates at the triple point. Under different wedge angles and shock strengths, a more complex reflection pattern is observed, called double Mach reflection. The reflected shock creates a second triple point (TP'), Mach stem (MS'), and slipline (S') (Figure 1.5c). The theory of regular and Mach reflection was developed by von Neumann and allowed the prediction of the type of reflection pattern that occurs (regular or Mach reflection) based on the wedge angle and shock strength. However, some difficulty was encountered when applying the theory to weak shocks. Early experimental evidence seemed



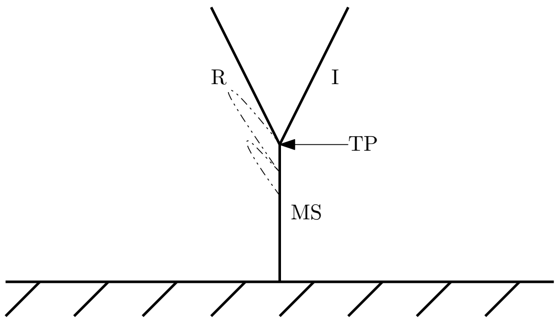
(a) Regular reflection.



(b) Single Mach reflection.



(c) Double Mach reflection.



(d) Guderley Mach reflection.

Figure 1.5: Regular reflection, single and double Mach reflection. The incident (I), primary and secondary reflected shocks (R, R'), Mach stems (MS, MS') and sliplines (S, S') are indicated. The sonic line in the Guderley Mach reflection case is indicated by the dashed-dotted line.

to indicate that in some parameter regimes, Mach reflection was the observed reflection pattern even though this was not allowed in von Neumann's theory. One proposed solution is that a singularity could be present behind the triple point, invalidating assumptions in von Neumann's theory. Another solution was proposed by Guderley, where there is an expansion fan and a supersonic patch behind the triple point [28]. Early experimental and numerical studies were unable to determine the correct solution. This is because the region of interest is very small, on the order of 10^{-4}. To properly resolve this flow feature, cell sizes on the order of 10^{-6} are required in the neighborhood of the triple point. Recently, numerical and experimental evidence has suggested that Guderley's solution is correct [29–31] (Figure 1.5d).

1.5 Limiters

Oscillations in numerical solutions given by high order numerical methods appear due to Gibbs' phenomenon in the presence of discontinuities. We illustrate this on a simple example by solving the linear advection equation, $u_t + u_x = 0$ on the periodic domain $[-1, 1]$, with a finite volume method using the upwind numerical flux and a slope reconstruction. The initial condition is the pulse $u_0(x) = 1$ if $x < 0$ and $u_0(x) = 0$ otherwise. The numerical solution at $T = 0.5$ is given in Figure 1.6a, where a number of overshoots and undershoots are present near the discontinuities. For linear fluxes, these oscillations may or may not be acceptable in the final solution. For nonlinear fluxes however, these nonphysical oscillations may lead to numerical instability. In this case, a stabilization procedure, such as slope limiting, must be used to suppress oscillations in the solution in case of wave steepening and shock formation. Using a slope limiter, e.g., the superbee limiter, an oscillation-free numerical solution can be obtained (Figure 1.6b). Typically, a slope limiting procedure modifies the numerical slope on each element such that it lies in a locally defined range. Then, a global stability property of the numerical solution is deduced. For one-dimensional problems, slope limiters that enforce a total variation diminishing (TVD) property have been successful [32]. However, an extension of this approach to two-dimensions was shown to yield at most first order accurate schemes [33]. For two-dimensional problems, slope limiters that maintain a local maximum principle can also preserve second order accuracy [34].

Limiting in multidimensional space is more difficult than in one dimension. In one dimension, there is only one limiting direction and a clear forward and backward neighboring element. In multiple dimensions, it is not obvious in which direction the slope must be limited, or how to define a neighboring element. Most work in multidimensional

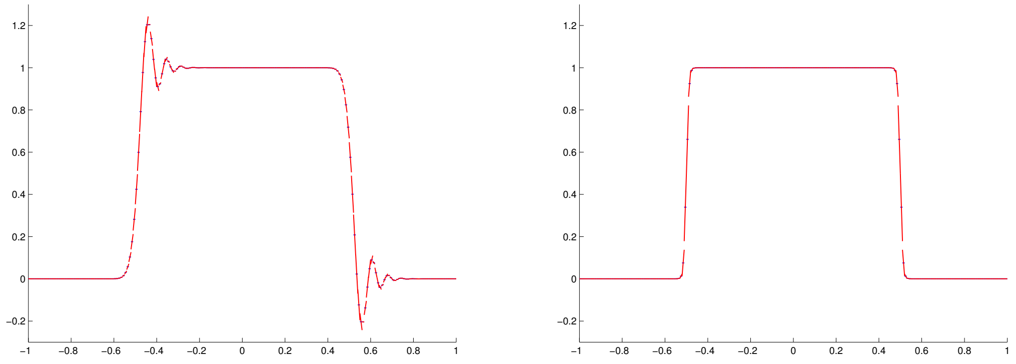

(a) Slope reconstruction without limiting. (b) Superbee slope reconstruction.

Figure 1.6: Linear advection example illustrating Gibbs' phenomenon.

limiting has concerned two-dimensional numerical schemes. For example, there are directional derivative limiters and moment limiters on Cartesian grids. Directional derivative limiters reduce the x and y partial derivatives by constant factors such that the value of the numerical solution at the surface quadrature points is contained in a local interval defined by neighboring elements. Barth-Jespersen type limiters multiply both x and y partial derivatives by the same constant factor between 0 and 1 [2, 35]. A less restrictive type of limiter solves a small linear program on each element such that x and y partial derivatives are multiplied by different factors [36]. One way of determining which elements are neighboring is finding all elements that share a geometrical vertex with the limited element. The difficulty with using this type of limiter is that the number of vertex neighbors increases quickly with dimension. In one dimension, there are only two vertex neighbors. On good quality two dimensional unstructured meshes of triangles, we have observed that each element can have up to 20 vertex neighbors. On good quality three dimensional unstructured meshes of tetrahedra, this number can reach 120, which leads to an exorbitant amount of computational work. The influence of using a subset of the vertex neighborhood for two-dimensional limiting was examined in [2]. The number of vertex neighbors that compose the limiting stencil varies from element-to-element, unless the mesh is structured. This can lead to inefficiencies for codes implemented on parallel computing architectures such as GPUs [2]. Specifically to the Euler equations of gas dynamics, positivity preserving limiters for density and pressure as well as entropy bounding limiters have been examined in [37–40] for high order DG solutions.

1.6 Outline of thesis

This thesis is structured as follows. In Chapter 2, we describe our GPU parallelized h-adaptive implementation of the DG method. Using these GPU algorithms, we solve a number of popular benchmarks in gas dynamics and two less common shock reflection problems that are intractable without some form of mesh adaptivity. In Chapter 3, we analyze the stability and accuracy of second-order slope limiters for the discontinuous Galerkin method on unstructured triangles. In Chapter 4, we study the optimal CFL number of SSP time stepping methods in one dimension used with the method of lines-type discretizations of hyperbolic conservation laws. In Chapters 5 and 6, we describe our novel approach to moment limiting on unstructured meshes of triangles and tetrahedra, respectively.

Chapter 2

Adaptive mesh refinement on unstructured meshes of triangles

In this chapter, we discuss and implement code optimization techniques for high order finite element GPU codes that support runtime adaptive mesh refinement (AMR). Our implementation presents a number of novelties. First, it is entirely implemented on the GPU. Many AMR solvers in the literature are actually hybrid CPU-GPU solvers, whereby the main solver is implemented on the GPU and some algorithms that modify the adaptively refined mesh are offloaded onto the CPU. CFD codes on GPUs can easily present race conditions when multiple threads attempt to write to the same memory location, e.g., in writing the surface contribution to the right-hand-side of two elements that share an edge or face. A suboptimal solution is extensive amounts of buffer memory [7]. In anisotropic adaptivity, a race condition can also occur when the code is modifying the position of the geometrical vertices of the mesh. In both cases, a coloring algorithm for work scheduling is a suitable solution [7, 41, 42]. The edge coloring of the initial, conforming mesh is done in the preprocessing stage. Based on this initial coloring, we describe a fast runtime mapping from parent to children which extends the edge coloring to adaptively refined meshes (Section 2.1.5). The resulting edge coloring is also used in mesh smoothing subroutines that force adjacent elements to not differ by more than a prescribed difference in refinement level (Section 2.2.1). We propose an efficient stream-compaction operation that ensures that data is contiguous in memory. Finally, we apply our optimized code to a number of computationally difficult problems in gas dynamics that are intractable without mesh adaptivity.

Our h-adaptive DG-GPU algorithm is implemented in NVIDIA CUDA C and comprises a DG module and AMR module. In the DG module, we calculate the RHS of (1.9) on a static mesh and advance the solution in time. Every $\mathcal{N}$ time steps, the AMR module adapts the mesh to the solution by refining and coarsening select elements. We show the organization of our AMR and RHS evaluation subroutines in Algorithm 1.

Algorithm 1 Pseudocode for AMR solver

```
step = 1; $t = 0$;
while $t < T$ do
    Advance $\mathbf{c}^n$ to $\mathbf{c}^{n+1}$ with the DG method and RK time stepping.    ▷ DG module
    if mod(step, $\mathcal{N}$) == 0 then
        AMR module.
    end if
    step++
    $t$ += $\Delta t$
end while
```

2.1 DG module

In this section we describe how the DG module is organized and give a brief overview of the subroutines that evaluate the right-hand-side of (1.9). For a more detailed treatment of these aspects of the solver, see [3, 7].

2.1.1 Data ordering and ID numbers

Every element and edge is assigned a unique identification integer (ID). The IDs of the elements and edges of the initial conforming mesh are given sequentially based on the output of the mesh generator. We store the IDs for elements and edges of the current mesh in arrays called `elem_list` and `edge_list` (Figure 2.1). After the AMR module is executed, the position an element or edge occupies in `elem_list` or `edge_list` may no longer correspond to its ID. This is because elements will be added and removed within the list due to refinement and coarsening subroutines. Therefore, we store the positions of IDs in `elem_id2pos` to avoid searching `elem_list`. In the example presented in Figure 2.1, the element with ID 7, Ω_7, is at the fifth index of `elem_list`, i.e. `elem_list[5] = 7`. Then, `elem_id2pos[7] = 5`. The same is done in `edge_id2pos` for edges in `edge_list`.

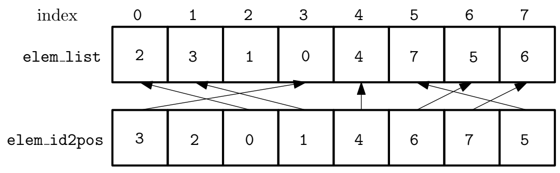


Figure 2.1: The value of `elem_id2pos[i]` indicates the location of Ω_i's ID in `elem_list`.

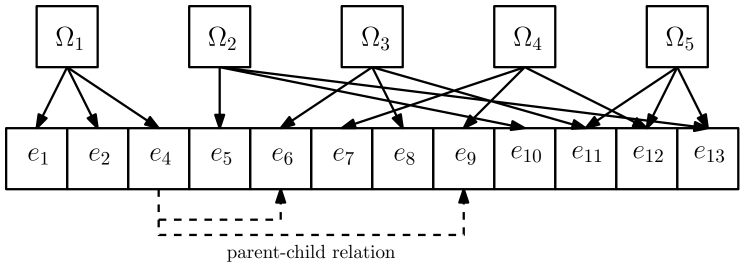
2.1.2 Element information

The DOFs are organized into arrays named `c0`, `c1`, ..., each of length N, which is the number of elements in the mesh. There is one array for every basis function and equation, i.e. the number of arrays is $N_p \times M$, where N_p is the number of basis functions and M is the number of equations in (1.1). The right-hand-side of (1.9) is stored in a similar manner in arrays named `rhs0`, `rhs1`, ... This guarantees coalesced reads and writes in computing the volume integral [7]. The DOFs are organized in the same order as the element IDs in `elem_list`. For example, $c_{7,0}$ is at index 5 of `c0` (Figure 2.1).

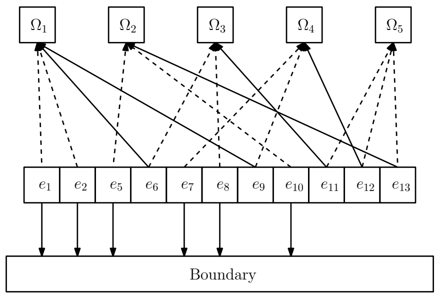
Additional data required for the computation of (1.9) such as element vertices, coordinate transformation Jacobians, and precomputed basis function values are also stored. Element connectivity is stored as the ID numbers of a triangle's three edges (element-to-edge connectivity data, Figure 2.2a). A triangle may have more than three edges if one or more of its neighbors have been refined, e.g. Ω_1 has four edges in Figure 1.1. For such nonconforming triangles, the parent ID of the refined edges is stored instead, e.g., edge ID 4 is stored rather than 6 and 9. The IDs 6 and 9 are found using the edge tree structure (Section 2.2.2).

2.1.3 Edge information

Edge normals and lengths are required for the computation of (1.9). Refining or coarsening an edge does not introduce new edge normals for straight-sided triangles. Therefore, we only store the normals and lengths of edges in the original mesh. Edges in the current mesh store their refinement level. To find the current edge length, simple arithmetic is done when evaluating the surface contribution term by dividing the original edge length by 2^r, where r is the edge refinement level. An edge points to its left and right element, i.e. the two elements that share it (Figure 2.2b); the ID of an edge's left and right elements are stored as integers in the arrays `left_elem` and `right_elem`.


Ω_1
Ω_2
Ω_3
Ω_4
Ω_5
e_1 e_2 e_4 e_5 e_6 e_7 e_8 e_9 e_{10} e_{11} e_{12} e_{13}
parent-child relation

(a) Element-to-edge connectivity.


Ω_1
Ω_2
Ω_3
Ω_4
Ω_5
e_1 e_2 e_5 e_6 e_7 e_8 e_9 e_{10} e_{11} e_{12} e_{13}
Boundary

(b) Edge-to-element connectivity. Dashed and solid lines connect an edge to its left and right element, respectively.

Figure 2.2: Connectivity information stored for the refined mesh in Figure 1.1.

2.1.4 Right-hand-side evaluation kernels

A standard RK time integrator requires the evaluation of the RHS of (1.9). This time stepping module consists of two kernels that evaluate the volume terms

$$\frac{1}{\det J_i}\int_{\Omega_c}\mathbf{F}(\mathbf{U}_i)\cdot{}^{\top}(J_i^{-1})\nabla\varphi_k\,\det J_i d\Omega_c \tag{2.1}$$

and surface terms

$$-\frac{1}{\det J_i}\sum_{j\in N_i^e, j\neq i}\int_{\partial\Omega_{i,j}}\varphi_k\mathbf{F}(\mathbf{U}_i,\mathbf{U}_j)\cdot\mathbf{n}_{i,j}\,dl. \tag{2.2}$$

One thread per element is launched for the first kernel `eval_volume`. Thread t_i computes the volume integral terms for Ω_i in (2.1). The thread then stores the volume contribution in `rhs0`, `rhs1`, ... The data for this kernel is accessed in a coalesced fashion. We illustrate this in Figure 2.3 where thread s accesses the sth positions of arrays `c0`, `c1`, ...

Similarly, one thread t_s per edge $\Omega_{i,j}$, i.e. e_s, is launched for the second kernel, `eval_surface`. Thread t_s loads the solution coefficients of its edge's left and right elements, then it computes the surface integral terms for edge e_s in (2.2). The thread then adds the surface contribution to the right-hand-side of its edge's left and right element in `rhs0`, `rhs1`, ... In this kernel, memory is not guaranteed to be accessed in a coalesced fashion. This is because consecutive edges may not necessarily have consecutive left and right elements due to the unstructured nature of the mesh.

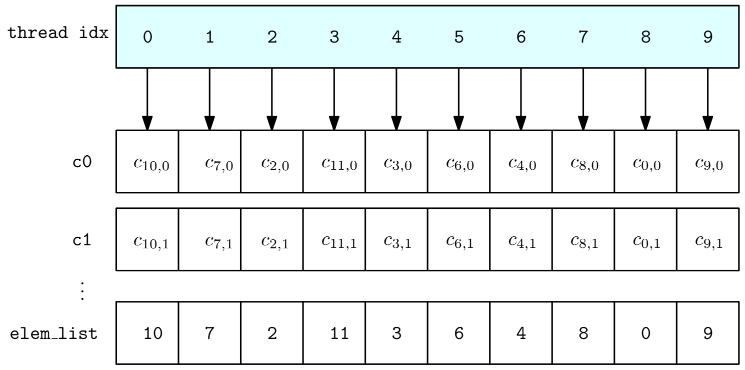


Figure 2.3: `eval_volume` coalesced read access pattern.

2.1.5 Coloring

Thread t_s in the `eval_surface` kernel evaluates the surface integral along the sth edge in `edge_list`. Then, t_s adds its surface contribution to the right-hand-side of the edge's left and right element. The race condition can arise if two threads simultaneously attempt to write their surface contribution to the same element (Figure 2.4). An edge coloring algorithm that partitions the edges of a conforming mesh was proposed in [7]. The race condition can be avoided by executing `eval_surface` over all edges of the same color in separate kernel launches (Figure 2.5). In [7], a simple mapping of the colors between coarse and fine elements is proposed as it is impractical to recolor the entire mesh when it is adaptively refined (Figure 2.6). The first child edge retains the color of its parent and the second child takes the parent's color incremented by three. If the parent's color is c, then the children's colors are c and $\mathrm{mod}(c + 2, 6) + 1$, e.g., if the parent's edge color is 1, then its children's colors are 1 and 4. Each new interior edge takes the color of the edge on the parent element to which it is parallel. This process is easily reversible during the coarsening operation. These algorithms result in the minimum number of colors used, i.e., 3 and 6 colors on conforming and nonconforming meshes of triangles, respectively.

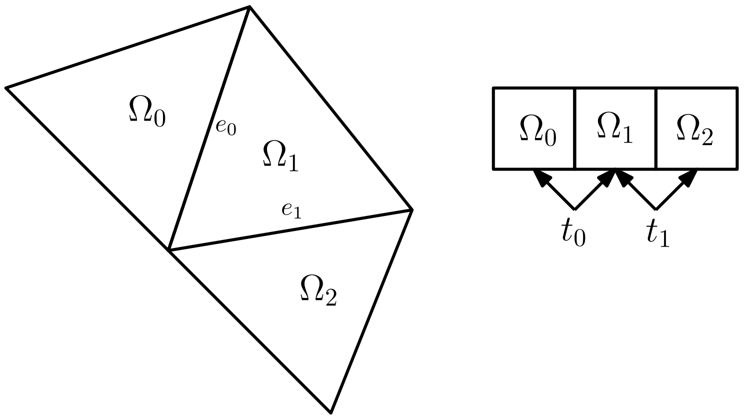
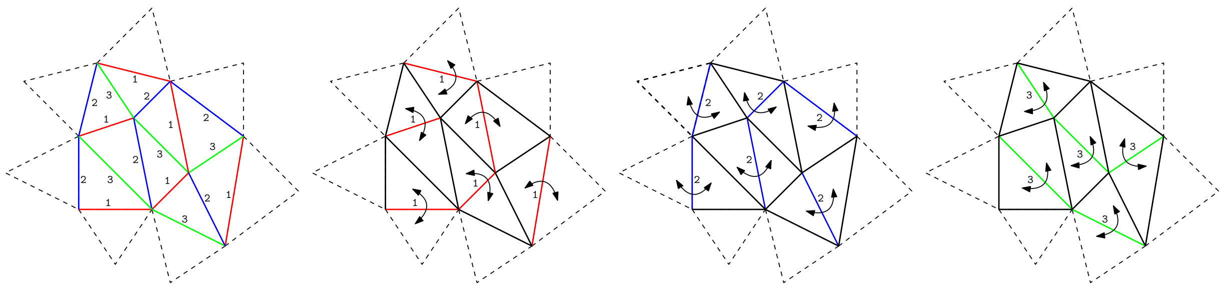


Figure 2.4: Race condition arises when threads t_0 and t_1 write simultaneously to the memory location of Ω_1.



(a) Edge coloring with colors 1, 2, and 3. (b) Kernel launch over edges of color 1. (c) Kernel launch over edges of color 2. (d) Kernel launch over edges of color 3.

Figure 2.5: Avoiding the race condition with coloring. The arrows indicate the elements to which each thread writes its surface contribution.

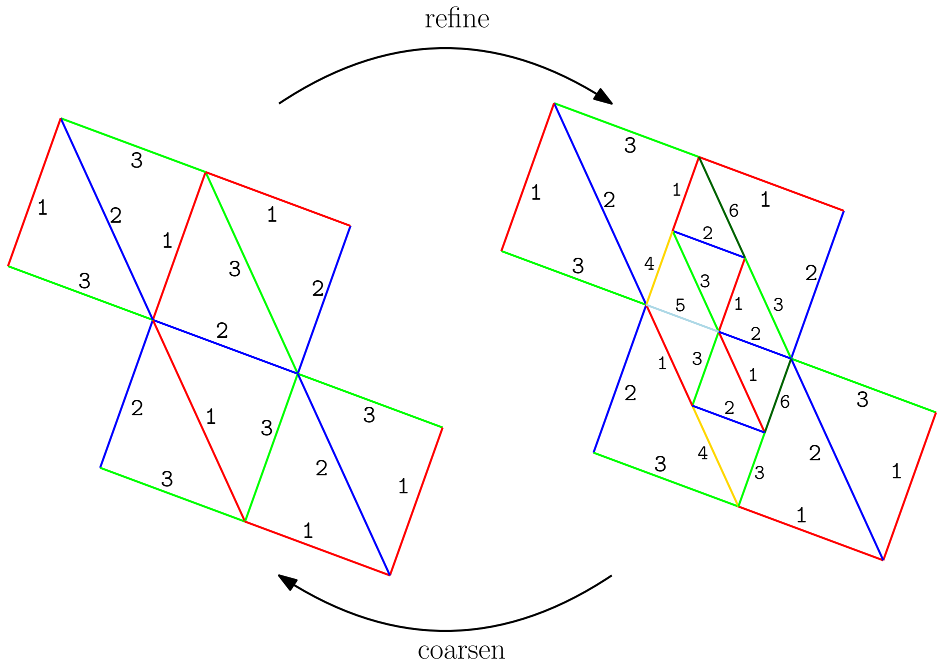


Figure 2.6: Mapping the initial coloring to refined triangles and back.

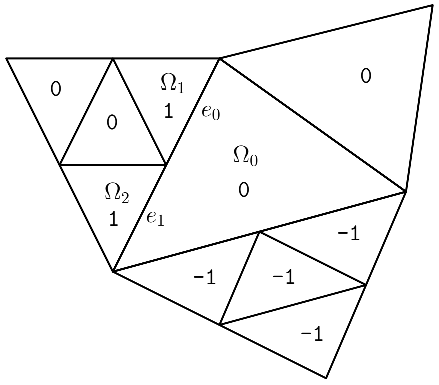
2.2 Adaptive mesh refinement

We discuss in this section the implementation details of the adaptive mesh refinement module of the code. First, we compute an indicator on each cell from which we determine which elements to flag for refinement or coarsening. During one execution of the AMR subroutines, we allow the refinement level of a cell to be adjusted by at most one. The exception is the initial condition where the AMR module is executed a number of times until a predefined maximum refinement level is reached.

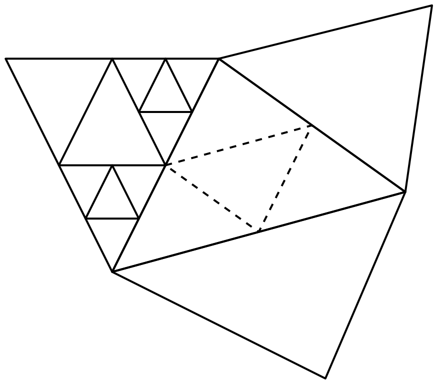
2.2.1 Mesh smoothing

After elements are flagged for refinement or coarsening (Section 2.2.8), we perform mesh smoothing to avoid creating a mesh where the refinement levels of neighboring elements differ by more than one (Figure 1.1, right), i.e., we require that

$$|r_i - r_j| \leq 1, \tag{2.3}$$



(a) '1' indicates refinement, '-1' indicates coarsening, and '0' indicates that the element is neither coarsened nor refined.



(b) Mesh after refinement, coarsening, and smoothing operations. The dashed line shows the elements added due to smoothing.

Figure 2.7: Mesh smoothing operation.

where r_i and r_j are the refinement levels of adjacent elements Ω_i and Ω_j, respectively. For example, in Figure 2.7a we display a mesh with AMR flags '`-1`', '`1`', or '`0`' on each element, indicating whether it should be coarsened, refined, or neither. These AMR flags are stored in the array `edr`, '**e**lement **d**ifference in **r**efinement'. When Ω_1 and Ω_2 are refined, the refinement level of their children and Ω_0 will differ by two, which violates the nonconformity constraint (2.3). We therefore must smooth the mesh to reduce this jump.

We implement this operation by launching one thread per edge in a kernel called `smooth` (Algorithm 2). Each thread loads the left and right element of its designated edge. Using the elements' current refinement levels and `edr` value, the element refinement levels after the AMR operation are computed. If their updated refinement levels do not satisfy the nonconformity constraint (2.3), then the AMR flag in `edr` of either the left or right element is modified such that a more refined mesh is obtained. For example, on edge e_0 in Figure 2.7a, the refinement levels after the AMR operation will be 2 on Ω_1's children and 0 on Ω_0. Consequently, we refine Ω_0 once (Figure 2.7b).

We execute `smooth` on all edges of the same color in separate kernel launches. This is done in order to avoid memory contention when modifying the AMR flag array `edr`. Without coloring, threads operating on edges e_0 and e_1 in Figure 2.7a could write their result simultaneously to the position in memory corresponding to Ω_0 in `edr`. `smooth` is launched multiple times until the nonconformity condition (2.3) is satisfied for all elements.

After mesh smoothing, it is finalized which elements are to be refined or coarsened. Thus, we may proceed to executing the refinement and coarsening subroutines.

Algorithm 2 Mesh smoothing operation

```
procedure SMOOTH(edr)
    le, re ← positions of left and right element.
    left_after, right_after ← level after refinement of left and right elements
    if left_after-right_after $> 1$ then
        edr[re] $= 1$
    else if right_after-left_after $> 1$ then
        edr[le] $= 1$
    end if
end procedure
```

2.2.2 Tree structures

The parent-children relations of elements and edges are stored in tree data structures, where parents and children point to one another. The tree structure for the elements in the refined mesh of Figure 1.1 are shown in Figure 2.8. The elements and edges that are active in the refined mesh are highlighted in blue and do not have children.

2.2.3 Connectivity

In order to evaluate the surface contributions in (2.2), each thread of the kernel `eval_surface` needs the IDs of the left and right elements sharing its assigned edge, i.e., edge-to-element connectivity data (Figure 2.2b). After the mesh is refined and coarsened, it is necessary to update mesh connectivity to reflect the addition and removal of elements, e.g., find the new left and right elements sharing an edge. We do this by using the tree structure described above.

First, we refine the edge tree by splitting edges flagged for refinement, assign new IDs to child edges, and update the IDs in `edge_list`. Similarly, we refine the element tree and update the IDs in `elem_list`. From the edge and element trees, we can compute the connectivity between new elements and edges.

As an example, consider the refined mesh in Figure 1.1. First the edges of Ω_0 are refined and the new IDs for the child edges of e_0, e_3, and e_4 are assigned (Figure 2.8,

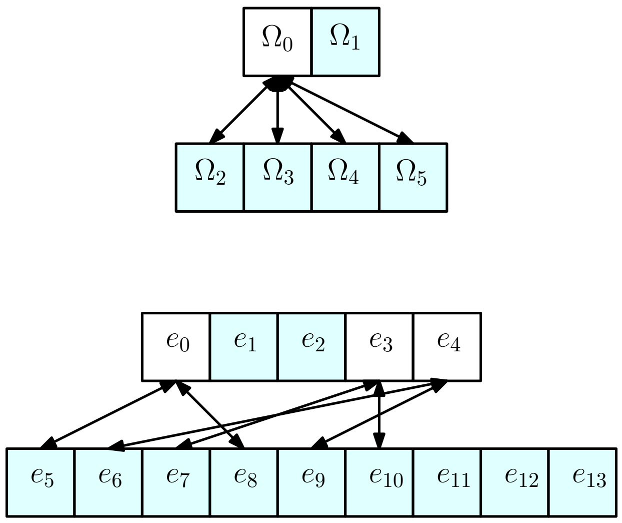


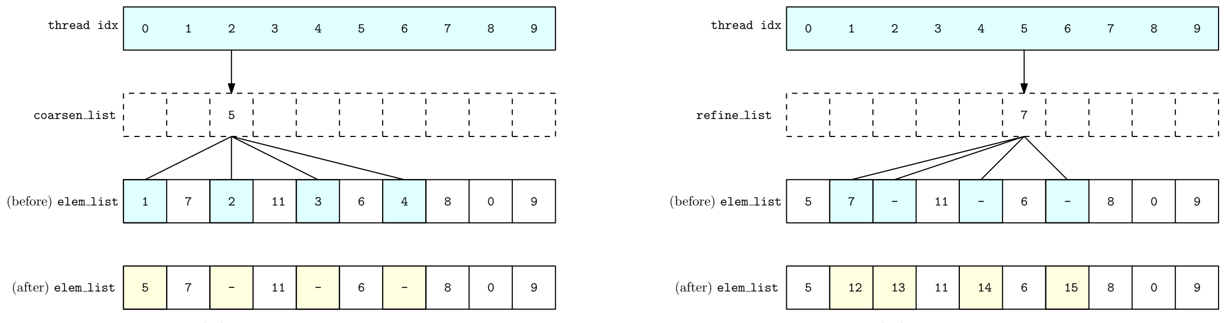
Figure 2.8: Tree for the refined mesh in Figure 1.1. Elements and edges shaded in blue are active in the current mesh.

bottom). Next, Ω_0 is refined and the new IDs for the children are assigned. The IDs of the child elements' outer edges can be found from the already updated edge tree, e.g. e_7 and e_{10}. Then, the edge IDs of the interior child triangle Ω_5 must be created, e_{11}, e_{12}, and e_{13}. Now that the new elements know the IDs of their edges, i.e. we have the updated element-to-edge connectivity (Figure 2.2a), the left and right elements of the new edges in `edge_list` can be updated.

By our convention, a triangle points to three edges that have the same refinement level. For elements that have more than three edges, e.g. Ω_1 in Figure 1.1, the ID of the refined edges' parent is stored instead. In our example, Ω_1 points to e_1, e_2, and e_4, which are all of refinement level 0. From the edge tree, we can find that e_4 points to e_6 and e_9 (Figure 2.2a).

2.2.4 Coarsening

Coarsening is done 'in-place', i.e., this modification to the mesh does not require a buffer and avoids large amounts of memory transfers (Figure 2.9a). `coarsen_list` contains a list of parent element IDs that are to replace their children. We place the parent ID in the first child's position in `elem_list` and flag the other children's memory locations as unused. We illustrate in Figure 2.9a how `elem_list` is updated to reflect the coarsening



(a) Coarsening. (b) Refining.

Figure 2.9: Access pattern for refinement and coarsening kernels.

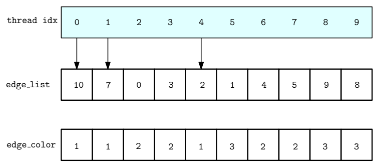
of elements Ω_1, Ω_2, Ω_3, and Ω_4. Thread 2 places the parent element ID of the cluster, i.e. Ω_5, in the list position of its first child element ID, i.e. in the position of element Ω_1. The freed memory spaces of the three other children are indicated by dashes. All the data associated with elements are dealt with in a similar fashion, e.g., the DOFs of the parent element Ω_5 are placed in the old memory location of the first child, Ω_1.

Coarsening four children leads to the removal of a section of the element and edge trees. We keep track of the memory and ID numbers freed during the coarsening operation in order for them to be reused during a refinement operation. For this reason, coarsening, if required, is always executed before refinement.

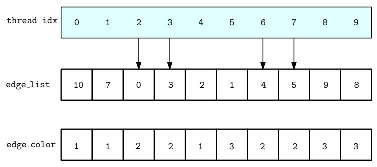
The solution coefficients on a parent element are obtained using an L_2 projection on the four child elements. The projection is implemented as a dot product of the solution coefficients on the children and weights that have been precomputed for fast execution.

2.2.5 Refinement

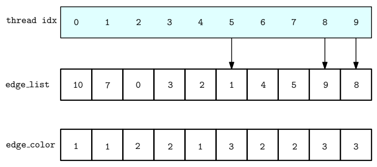
Refinement is also done 'in-place' (Figure 2.9b). `refine_list` contains a list of element IDs that are to be refined. The position of the parent element in `elem_list` is taken by its first child. The IDs for the three remaining children are placed in free memory locations, if any were made available during coarsening. If there are no free locations within the list, then the additional IDs are concatenated to the end of the array. All the data associated with elements are dealt with in a similar fashion. For example, in Figure 2.9b, Ω_7 in `refine_list` is refined by thread 5. The first child ID Ω_{12} overwrites Ω_7, the rest (Ω_{13}, Ω_{14}, and Ω_{15}) are placed in the remaining available memory locations of `elem_list` indicated by dashes. The DOFs of the four child elements are obtained with an L_2 projection, which is equivalent



(a) Launch over edges of color 1.

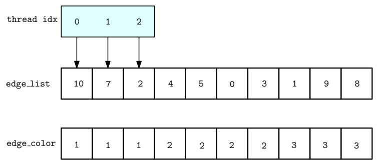


(b) Launch over edges of color 2.

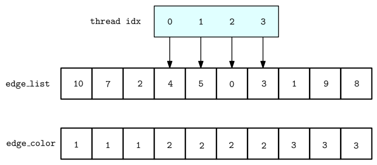


(c) Launch over edges of color 3.

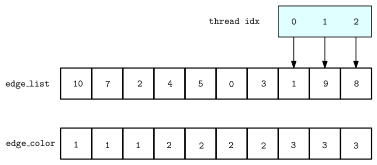
Figure 2.10: Without ordering of the edge IDs, `eval_surface` will have inactive threads, reducing parallelism.



(a) Launch over edges of color 1.



(b) Launch over edges of color 2.



(c) Launch over edges of color 3.

Figure 2.11: With ordering of the edge IDs, all threads of `eval_surface` will be active.

to a dot product of the parent DOFs and weights, which have been precomputed for fast execution.

2.2.6 Edge reordering

The surface integral kernel, `eval_surface`, is executed on the edges of a particular color in separate launches. If the edges in `edge_list` are not ordered by color, then the parallelism of `eval_surface` will be reduced. This is because some threads will be inactive during the kernel execution (Figure 2.10). The edges in `edge_list` are not guaranteed to be ordered by color after the mesh is refined and coarsened. Therefore, after the AMR subroutines complete, the edges in `edge_list` must be reordered by color. This will guarantee all threads in the launches of `eval_surface` are active (Figure 2.11).

2.2.7 Memory management

In Sections 2.2.4 and 2.2.5, we described how the coarsening and refinement operations are done in-place. This is to avoid using buffers and unnecessary memory transfers. However, if the total number of elements in the mesh is reduced by the refinement and coarsening operations, there will be 'holes' in the the data arrays. This will reduce the efficiency of memory accesses by making some reads and writes uncoalesced. Therefore, these 'holes' must be filled to make the data contiguous in memory.

Algorithm 3 Compaction

```
procedure COMPACTION(out, in, out_flag, in_flag)
    idx ← thread index
    if in_flag[idx] then
        out[out_flag[idx]] ← in[idx]
    end if
end procedure
```

This operation on GPUs is called a 'stream compaction' and it does not have a simple solution. Typically, a compaction is done using an operation called a prefix sum. A prefix sum takes as input an array of integers `in_flag` and outputs another array `out_flag`. The element at the nth index of the output array is given by the formula

$$\texttt{out\_flag}[n] = \sum_{i=0}^{n-1} \texttt{in\_flag}[i] \text{ for } n > 0, \tag{2.4}$$

and `out_flag`[0] is set to 0 (Figure 2.12). Now, assume that we wish to copy selected data from the array `in` into the array `out`. `in_flag` is an integer array of `0`'s and `1`'s, which indicates whether the data at the corresponding position in `in` must be preserved ('`1`') or removed ('`0`'). After computing the prefix sum for `in_flag`, a compaction kernel (Algorithm 3) is launched that creates a new array `out` without the unwanted data. This procedure is illustrated in Figure 2.12. There are application programming interfaces (APIs) that provide an implementation of the prefix sum and compaction operations. Unfortunately, the standard implementations of the stream compaction operation are not suitable for our purposes. This is because we may have GBs of data where only a small fraction of the elements in those arrays require removal. Executing the compaction kernel will always result in all the data of the compacted array being moved to a new location, regardless of the number of elements being removed. A more efficient solution is to implement an in-place compaction, which we will now describe.

Suppose we wish to remove the elements from the array of integers in Figure 2.13 using `in_flag`. According to `in_flag`, the compacted array will have a length of 6. Let us refer to the position between indices 5 and 6 as the 'pivot'. The idea is to fill the holes to the left of the pivot with elements from the right of the pivot. First, we find the elements to the

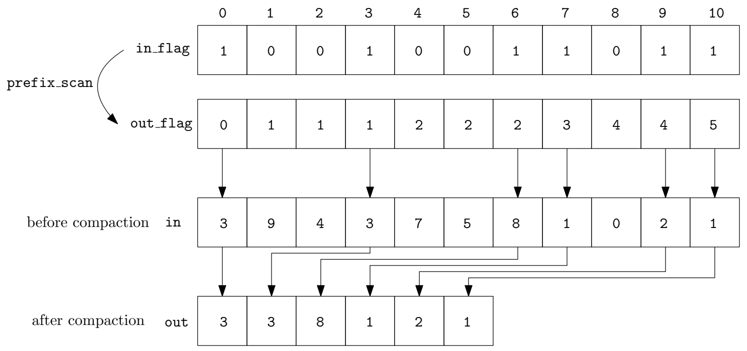


Figure 2.12: Example of a standard compaction algorithm.

right of the pivot for which `in_flag` is '`1`' and store their position in the array `from`. Next, we find the elements located to the left of the pivot for which `in_flag` is '`0`' and store their positions in `to`. Populating `to` and `from` is done using the prefix-scan from CUB [43]. The final step is to launch a kernel which completes the data transfer in-place (Algorithm 4). This operation does not preserve the initial ordering of the data, but this is not important in our application.

Algorithm 4 In-place compaction

```
procedure IN_PLACE_COMPACTION(in, to, from)
    idx ← thread index
    in[to[idx]] ← in[from[idx]]
end procedure
```

2.2.8 Refinement strategy

In this work, we are not concerned with the optimal way to flag an element for coarsening or refinement. Therefore, we adopt the following simple strategy reported in the literature [44]. We compute on Ω_i the refinement indicator ϵ_i. Then, we compare ϵ_i to reference values $\epsilon_r = \epsilon\delta$ and $\epsilon_c = \epsilon/\delta$, where ϵ and δ are prescribed constants. Ω_i is refined if $\epsilon_i > \epsilon_r$. A cluster of four elements, Ω_i, Ω_j, Ω_k, Ω_l, with the same parent element, is coarsened if $\epsilon_i, \epsilon_j, \epsilon_k, \epsilon_l < \epsilon_c$. We choose the simple refinement indicator

$$\epsilon_i = h_i||\nabla U_i||_2, \tag{2.5}$$

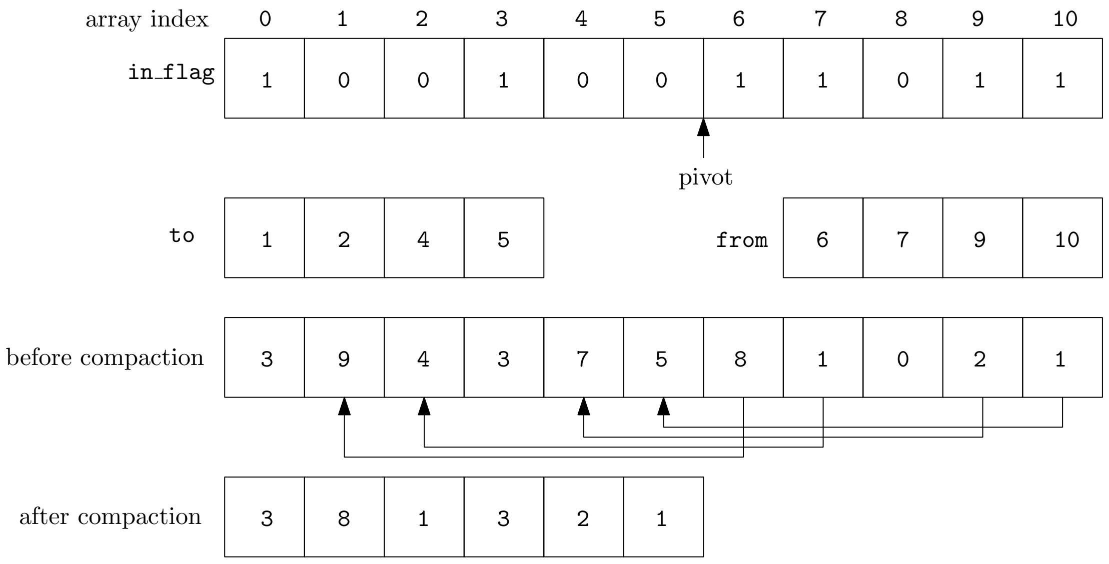


Figure 2.13: In-place compaction implementation.

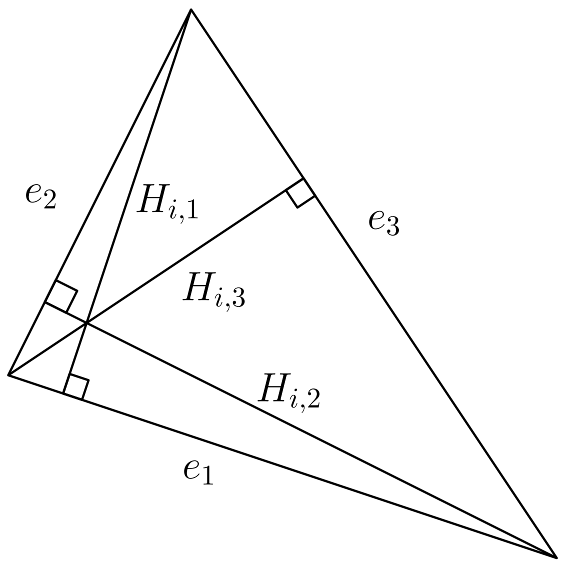


Figure 2.14: Minimum cell height $h_i = \min(H_{i,1}, H_{i,2}, H_{i,3})$ [2].

where ∇U_i is the gradient evaluated at the cell centroid and h_i is the minimum cell height (Figure 2.14, [2]).

2.3 Computed examples

2.3.1 Initial-boundary value problems

We solve the two-dimensional Euler equations, which can be written in form (1.1) with the fluxes

$$\mathbf{F}_1(\mathbf{U}) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ (E+p)u \end{pmatrix} \quad \text{and} \quad \mathbf{F}_2(\mathbf{U}) = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ (E+p)v \end{pmatrix}, \tag{2.6}$$

where $\mathbf{U} = [\rho, \rho u, \rho v, E]$. The system is closed with the equation of state

$$p = (\gamma - 1)\left(E - \frac{\rho}{2}(u^2 + v^2)\right),$$

where $\gamma = 1.4$ is the adiabatic constant for air, ρ is the density, u and v are components of the velocity vector, and E is the energy.

2.3.1.1 Smooth isentropic vortex

We use this example to illustrate the runtime performance of our AMR algorithm. This example was solved on an NVIDIA Titan X Pascal. The problem with initial conditions stated in Table 2.1 has the exact solution $\mathbf{U}(x, y, t) = \mathbf{U}_0(x, y - t)$, i.e. the initial vortex is advected in the y direction with speed 1 [45]. It is solved until the final time $T = 2$ on the domain $[-10, 10]^2$ with the exact solution used as boundary conditions. The initial mesh is coarse and composed of 180 unstructured triangles.

First, we perform an initial mesh adaptation to accurately capture the initial conditions. Then, we run the mesh adaptation subroutines every time step. In this example, we allow at most six levels of refinement. The size of the mesh for all orders of approximation was about 30,000 elements and did not vary substantially during the simulation. This is expected since the solution is a translation of the initial conditions.

The breakdown of compute time in seconds for solutions with $p = 1...4$ is reported in Table 2.2. The same data, but as a percentage of total AMR time, are shown in Figure 2.15. We report the timings in terms of seven subroutines: `determineParents`, `smooth`, `coarsen`, `refine`, `compaction`, `determineSideOrder`, `reorderSides`, and `other`. The procedures grouped in `other` include computing the refinement indicator (2.5), updating mesh connectivity and tree structures. `determineParents` looks up the parent IDs

ρ	$\left(1-(\gamma-1)\frac{(SM)^2}{8\pi^2}e^r\right)^{\frac{1}{\gamma-1}}$
u	$\frac{Sy}{2\pi R}e^{\frac{1}{2}r}$
v	$1-\frac{Sx}{2\pi R}e^{\frac{1}{2}r}$
p	$\frac{1}{\gamma M^2}\rho^\gamma$

Table 2.1: Initial conditions for the smooth isentropic vortex problem (Example 2.3.1.1), where $r = \frac{1}{R^2}(1 - x^2 - y^2)$, $S = 13.5$, $M = 0.4$, and $R = 1.5$.

of the elements flagged for coarsening and stores them in `coarsen_list` (Figure 2.9a). `determineSideOrder` determines the new order of the edge IDs based on their color and `reorderSides` reorders them and edge data to avoid race conditions in `eval_surface`.

In Table 2.2, we notice that the time spent in the AMR subroutines increases with the order of approximation as the number of calls to the time stepping and mesh adaptation modules increases with p. This is because the time step size scales inversely with the order of the method for the DG method of order approximation p paired with an RK scheme of order $p + 1$. In Table 2.3, we list the data from Table 2.2 normalized by the number of time steps. For subroutines that only depend on the number of elements in the mesh, we observe that the normalized timings are similar for all polynomial orders.

We also note from Table 2.2 that the fraction of the total runtime spent in the AMR module of the code decreases with p. This is because the number of RK stages and the cost of computing the RHS of (1.9) increases with p. For orders $p = 1$ to 3, the `smooth` subroutine is the most time consuming operation because it is difficult to parallelize efficiently on the GPU (Section 2.2.1).

Note that refinement is usually performed less frequently than after every time step as is done for this example, e.g., we may refine when the solution moves two cell widths. For the RK-DG method where $p = 1, 2, 3, 4$, this means every 8, 12, 16, 20 time steps, respectively [46]. Therefore, for practical applications, the overhead associated with the AMR subroutines is small compared to the total runtime of the solver, especially for high order simulations.

2.3.1.2 Kelvin-Helmholtz instability

Next, we solve the Kelvin-Helmholtz instability problem on the domain $[-1, 1]^2$, with the initial conditions given in Table 2.4 and illustrated in Figure 2.16 [1]. The initial conditions

Subroutine	1	2	3	4
`determineParents`	0.44	0.64	0.84	1.03
`smooth`	1.21	1.75	2.29	2.81
`coarsen`	0.49	0.91	1.71	2.90
`refine`	0.76	1.24	2.03	3.39
`compaction`	0.49	0.78	1.16	1.67
`determineSideOrder`	0.70	1.00	1.32	1.62
`reorderSides`	0.18	0.25	0.33	0.41
`other`	0.49	0.73	0.96	1.21
AMR time	4.76 (0.45)	7.30 (0.20)	10.63 (0.09)	15.05 (0.04)
Total solver runtime	10.54	36.83	121.33	380.67

Table 2.2: Break-down of time spent in each AMR subroutine in seconds for Example 2.3.1.1. The number in parentheses is the fraction of the total runtime spent in the AMR module of the code, when refining every time step.

Subroutine	1	2	3	4
`determineParents`	195.93	196.18	198.41	198.27
`smooth`	539.08	537.41	542.00	541.03
`coarsen`	216.95	279.58	404.79	558.14
`refine`	335.65	381.98	480.29	652.52
`compaction`	216.66	239.40	275.16	321.12
`determineSideOrder`	310.73	309.06	312.01	311.20
`reorderSides`	78.13	77.82	78.42	79.02
`other`	218.06	224.95	226.57	233.06
AMR time	2111.19	2246.38	2517.65	2894.35

Table 2.3: Timings in Table 2.2 divided by the number of time steps. Values are reported in microseconds per time step. The grayed rows correspond to operations with runtimes that are independent of the order of approximation.

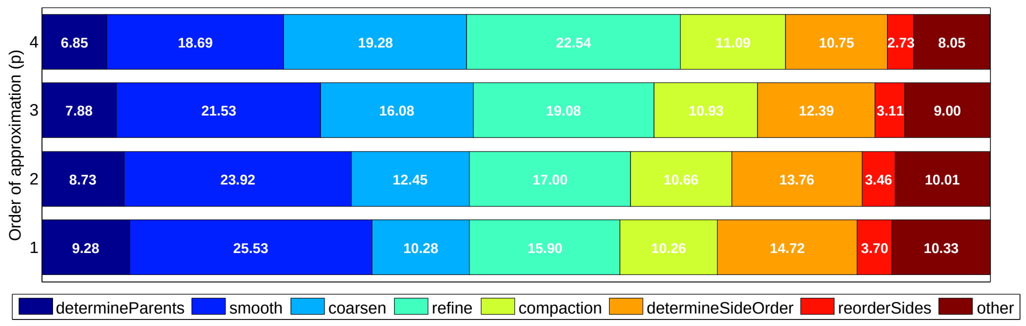


Figure 2.15: Percentage of the AMR runtime spent in each subroutine.

	$\lvert y\rvert \leq 0.5$	elsewhere
ρ	2	1
u	0.5	-0.5
v	$w \sin(4\pi x)$	$\left[e^{-\frac{1}{2s^2}(y+0.5)^2} + e^{-\frac{1}{2s^2}(y-0.5)^2)}\right]$
p	2.5	

Table 2.4: Density, velocity, and pressure of the three layers of fluid where $w = 0.1$ and $s = 0.05/\sqrt{2}$ [1] (Example 2.3.1.2).

describe three fluid layers. The outer layers are dense, rightward moving fluids that are sandwiching a less dense leftward moving fluid. In the neighborhood of the interfaces, the fluids are perturbed with an oscillatory vertical velocity. These interfaces are sliplines, which will lead to a rich production of vortices in the numerical solution. We prescribe the initial state at the horizontal boundaries of the domain and impose periodic boundary conditions at the vertical boundaries. The initial mesh is composed of two triangles.

The density along with the adaptively refined meshes at $T = 2$ with 6, 9, and 12 levels of refinement are plotted in Figures 2.17 and 2.18. The number of elements in the final meshes is approximately 4,000, 120,000, and 4,000,000. The AMR algorithms act predominantly in the neighborhood of the sliplines. As we increase the number of refinement levels, the vortices become more pronounced due to the reduction in numerical viscosity.

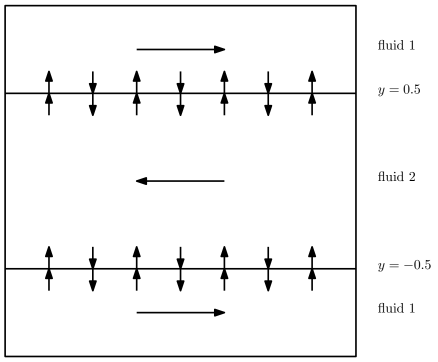


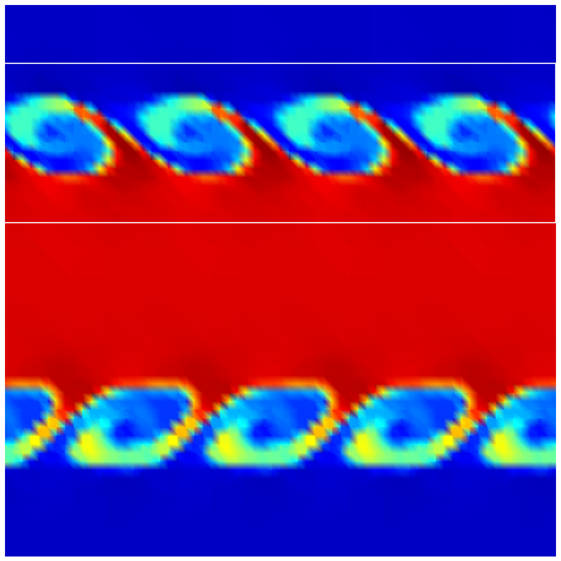
Figure 2.16: Setup of the Kelvin-Helmholtz test problem (Example 2.3.1.2). The arrows indicate the direction of fluid flow.

	$\mathbf{U}_I$	$\mathbf{U}_Q$
ρ	8	1.4
s	8.25	0
p	116.5	1

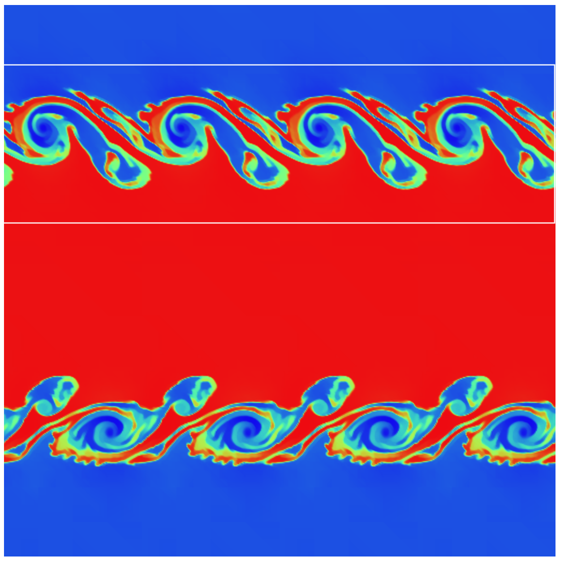
Table 2.5: Density, normal speed (s), and pressure of the incident $\mathbf{U}_I$ and quiescent $\mathbf{U}_q$ states in Example 2.3.1.3.

2.3.1.3 Double Mach reflection

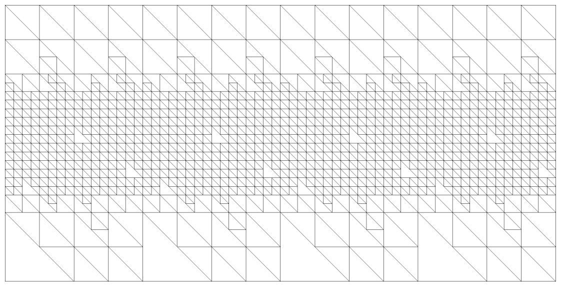
We use this example to demonstrate the algorithm's performance on a transient shock reflection problem. We solve double Mach reflection problem on the domain $[0, 3.5] \times [0, 1]$ with the initial condition of a rightward-moving shock that propagates into a quiescent gas [47]. The incident Mach 10 shock forms a $60°$ angle with a reflecting boundary, which results in the reflection pattern shown in Figure 1.5c. We provide the setup in Figure 2.19 along with the incident $\mathbf{U}_I$ and quiescent $\mathbf{U}_Q$ states in Table 2.5. We solve the problem until the final time of $T = 0.2$, allowing 3, 6, and 9 levels of refinement from an initial unstructured mesh composed of 1,776 triangles. The adaptively refined meshes at the final time are shown in Figure 2.20 along with the density isolines on the finest mesh in Figure 2.21.



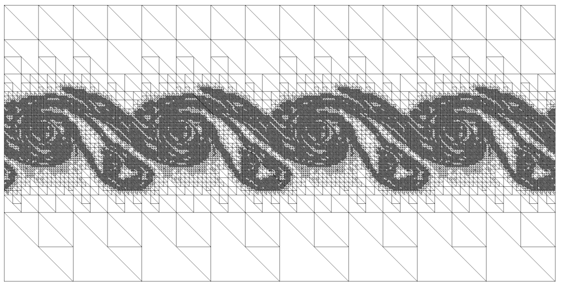
(a) 6 levels of refinement.



(b) 9 levels of refinement.

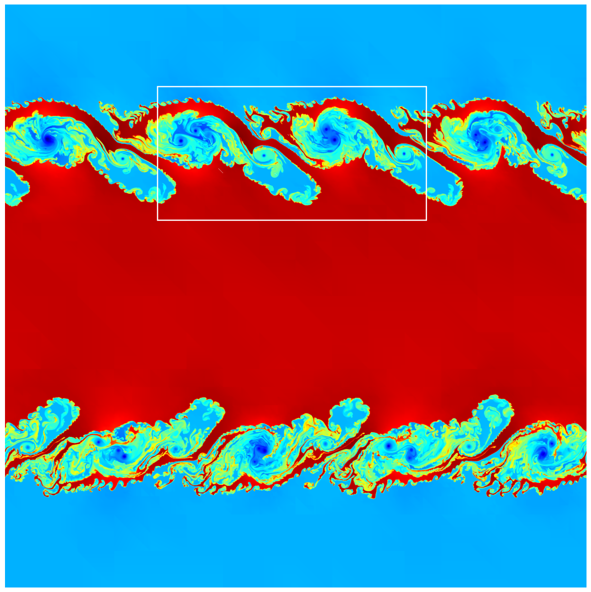


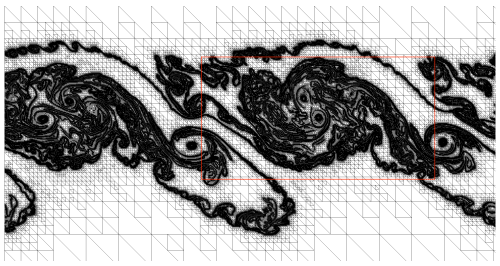
(c) Mesh on white rectangle in Figure 2.17a.



(d) Mesh on white rectangle in Figure 2.17b

Figure 2.17: Final solutions and adaptively refined meshes of the Kelvin-Helmholtz test problem allowing 6 and 9 levels of refinement.


(a) 12 levels of refinement.


(b) Mesh on white square in Figure 2.18a.

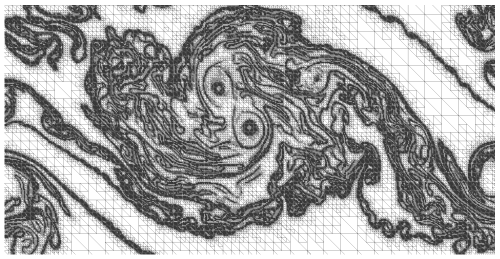

(c) Zoom on red rectangle in Figure 2.18b.

Figure 2.18: Final solution and adaptively refined mesh of the Kelvin-Helmholtz test problem allowing 12 levels of refinement.

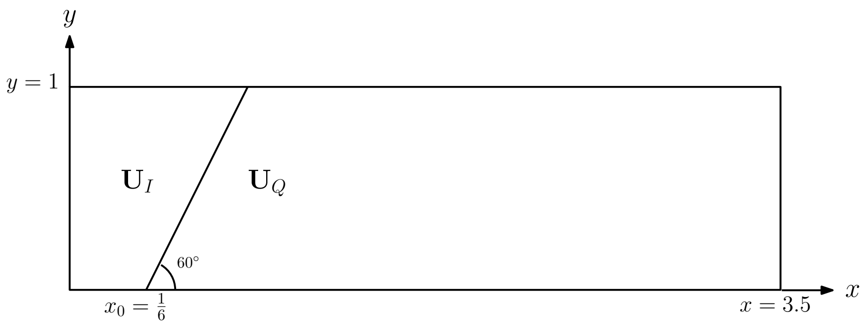
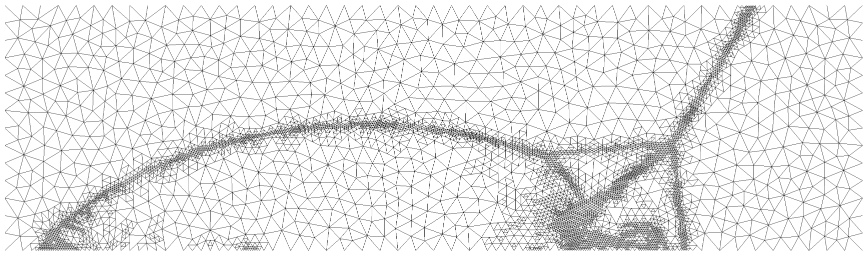
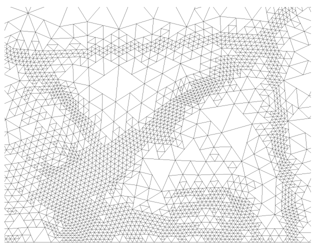
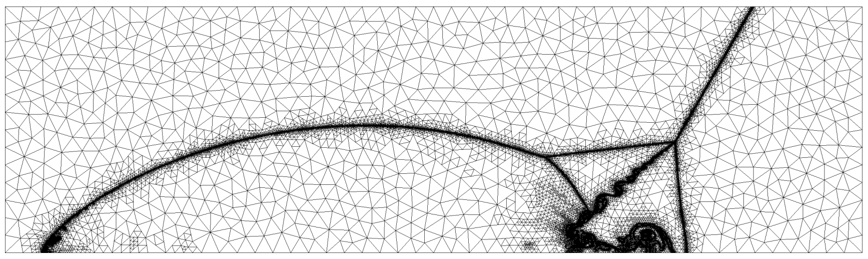
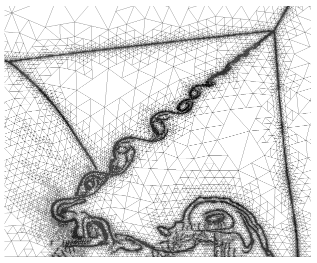


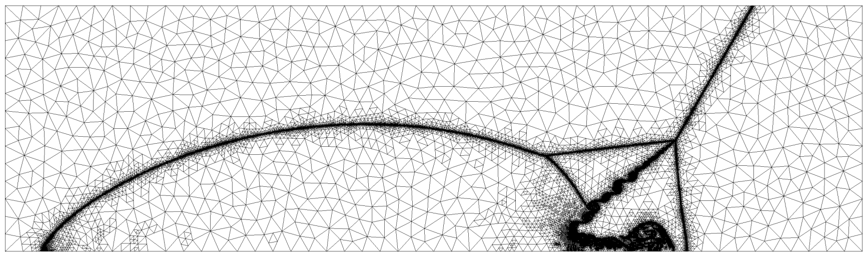
Figure 2.19: Double Mach reflection setup (Example 2.3.1.3).


(a) 3 levels of refinement.


(b) Zoom on slipline region.


(c) 6 levels of refinement.


(d) Zoom on slipline region.


(e) 9 levels of refinement.

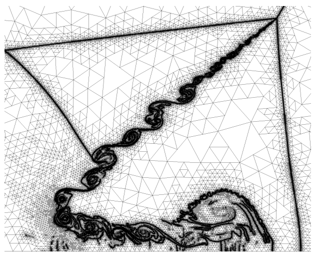

(f) Zoom on slipline region.

Figure 2.20: Final meshes of the double Mach reflection problem allowing 3, 6, and 9 levels of refinement.

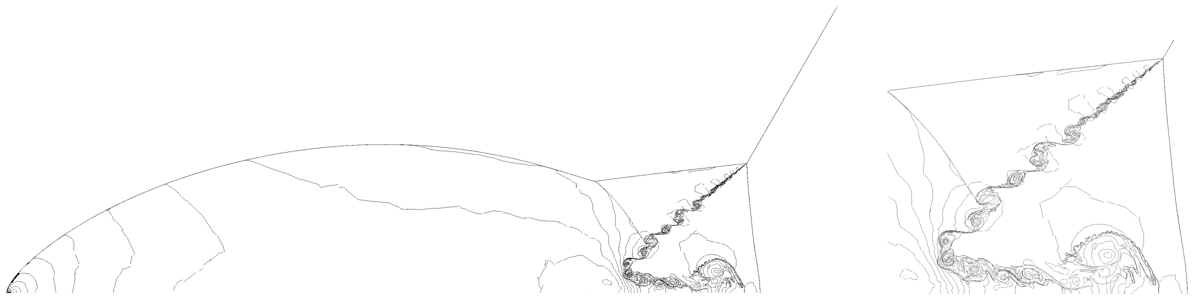

(a) Density isolines. (b) Zoom on slipline region.

Figure 2.21: Density isolines at the final time on the mesh with 9 levels of refinement in Figures 2.20e and 2.20f.

2.3.2 Boundary value problems

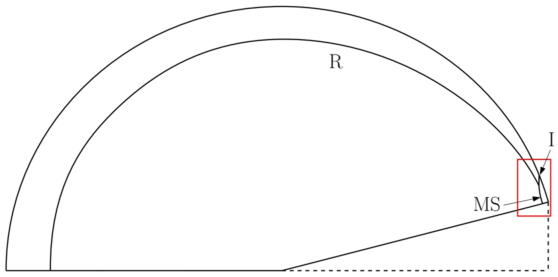
The initial-boundary value problem (1.1) can be restated as a boundary value problem in the self-similar coordinates $\xi = \frac{x}{t}$, $\eta = \frac{y}{t}$, and $\tau = \ln t$, as follows

$$\frac{\partial}{\partial \tau}\mathbf{U} + \frac{\partial}{\partial \xi}(\mathbf{F}_1(\mathbf{U}) - \xi\mathbf{U}) + \frac{\partial}{\partial \eta}(\mathbf{F}_2(\mathbf{U}) - \eta\mathbf{U}) + 2\mathbf{U} = 0. \tag{2.7}$$

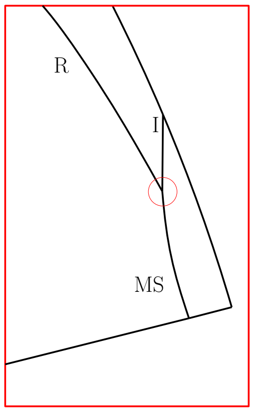
A steady state solution of (2.7) corresponds to a self-similar solution of (1.1) in (ξ, η) coordinates. Self-similar form (2.7) is useful because adapting the mesh for steady state solutions is simpler than for transient ones. This is because the relevant features in the solution do not move after the initial transient phase passes. In this section, we solve (2.7) where $\mathbf{F}_1(\mathbf{U})$ and $\mathbf{F}_2(\mathbf{U})$ are the fluxes in (2.6).

2.3.2.1 Von Neumann triple point paradox

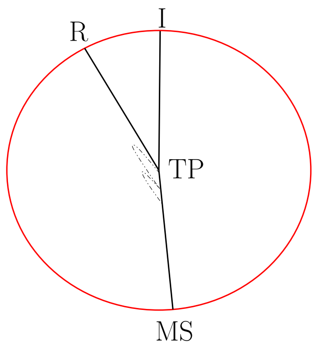
In this example, we present numerical evidence that supports Guderley's solution to the von Neumann triple point paradox. The problem setup consists of a weak, rightward moving incident shock impinging obliquely on a wedge on a computational domain in the shape of a circular sector (Figure 2.22a). The incident (I) and reflected (R) shocks detach from the wedge and meet at a triple point (TP). The triple point is connected to the wedge via a Mach stem (MS). We are interested in a very small portion of the reflection interaction shown in Figures 2.22b and 2.22c. The supersonic patches predicted by the Guderley Mach reflection are illustrated with dash-dotted lines in Figure 2.22c.



(a) Shock reflection pattern on the domain.



(b) Zoom on the red rectangle in Figure 2.22a.



(c) Zoom on the red circle in Figure 2.22b. The dashed-dotted line behind the triple point is the sonic line.

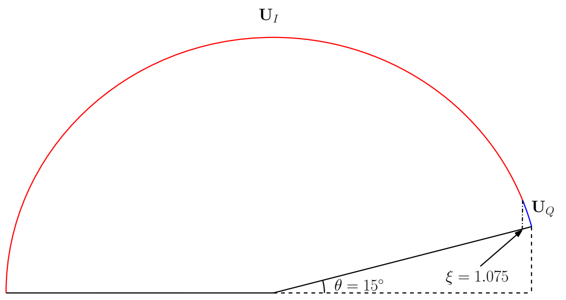
Figure 2.22: Incident (I) and reflected (R) shocks, with the Mach stem (MS) on the solution domain, with zooms on the neighborhood of the triple point (TP).

A number of numerical investigations of this problem have been executed on block adaptively refined grids [48] or distorted conforming grids [29, 49]. Here we present results on an unstructured mesh of triangles. The initial conforming mesh of the circular sector-shaped domain in Figure 2.23a is shown in Figure 2.23b. The initial mesh is constructed such that the elements are aligned with the incident shock.

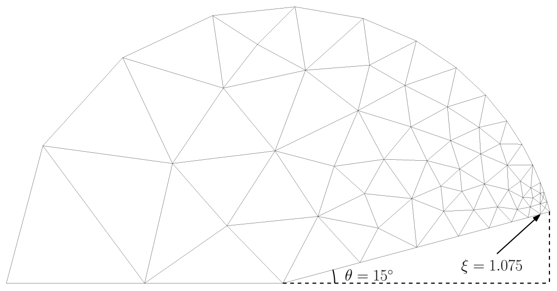
The boundary conditions are given by a weak incident shock traveling at Mach 1.075 into a quiescent gas [29]. The incident $\mathbf{U}_I$ and quiescent $\mathbf{U}_Q$ states are reported in Table 2.6, the boundaries on which they are imposed are shown in Figure 2.23a. With the goal of determining an accurate position of the triple point, we resolve the full length of the incident and Mach stem as opposed to only in a neighborhood of the triple point. Additionally, we explicitly prescribe that elements lying on a half disk centered on the triple point are refined to the maximum level (Figure 2.25c). This is because the solution varies little in that region and the refinement indicator we use has difficulty detecting the complex reflection pattern. This small refined region moves with increasing pseudotime τ, following the triple point along the vertical line $\xi = 1.075$ to its final position in self-similar coordinates $(1.075, 0.4111)$. We plot in Figure 2.24 the η coordinate as a function of pseudotime τ.

	$\mathbf{U}_I$	$\mathbf{U}_Q$
ρ	1.57697	1.4
u	0.12064	0
v	0	0
p	1.18156	1

Table 2.6: The incident $\mathbf{U}_I$ and quiescent $\mathbf{U}_Q$ states imposed as boundary conditions in Example 2.3.2.1.



(a) Initial domain. The incident $\mathbf{U}_I$ and quiescent $\mathbf{U}_Q$ states are imposed on the red and blue boundaries, respectively. The bottom boundary is reflecting.



(b) Initial, conforming mesh of domain in Figure 2.23a.

Figure 2.23: Initial domain and mesh for the von Neumann triple point paradox problem in Example 2.3.2.1.

In Figure 2.25, we provide an adapted mesh, composed of $\sim$ 800,000 elements, with $\sim$ 120,000 elements of minimum cell width $h \approx 4.6 \cdot 10^{-6}$ in the neighborhood of the triple point, where h is defined in Figure 2.14. We compute the self-similar Mach number

$$\tilde{M} = \frac{\sqrt{(u-\xi)^2 + (v-\eta)^2}}{c},$$

where c is the speed of sound, and plot the isolines of $\tilde{M}$ in Figure 2.26. The solution in Figure 2.26b was obtained by further refining the elements in the neighborhood of the triple point in Figure 2.26a by a factor of 8. On the coarser mesh, only one supersonic patch is discernible. However, two patches become visible with additional resolution. The finer mesh is composed of $\sim$ 6,200,000 elements, with $\sim$ 5,000,000 elements of minimum cell width $h \approx 5.8 \cdot 10^{-7}$ in the neighborhood of the triple point.

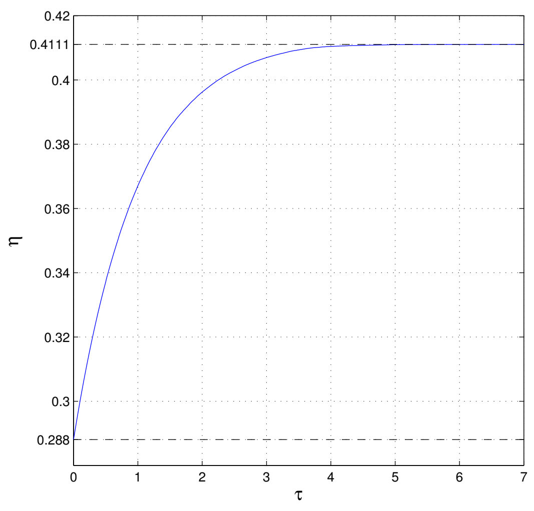
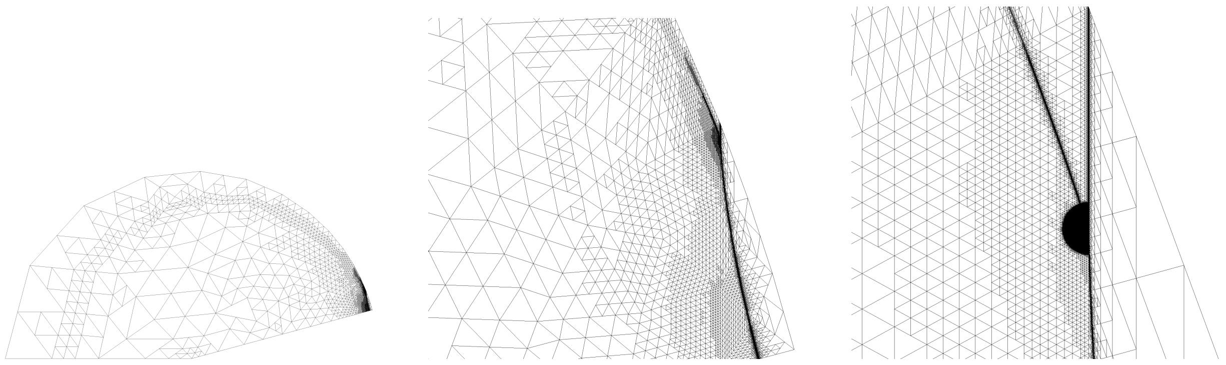


Figure 2.24: The vertical coordinate of the triple point vs. pseudotime.

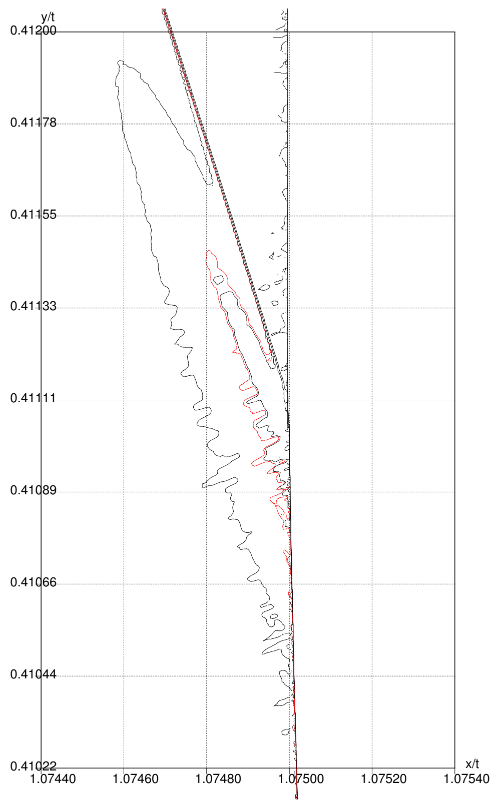


(a) Adaptively refined mesh for the solution in Figure 2.26a.

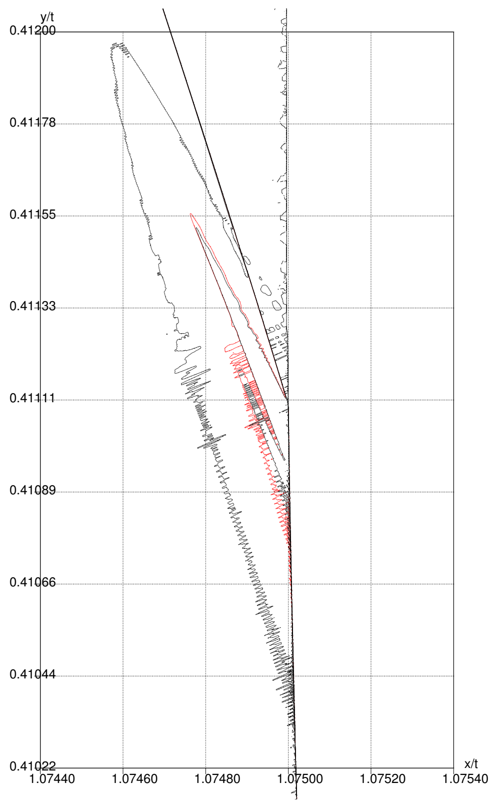
(b) First zoom of Mach stem and triple point region in Figure 2.25a.

(c) Second zoom of Mach stem and triple point region in Figure 2.25b.

Figure 2.25: The adaptively refined mesh for the solution in Figure 2.26a.



(a) $h \approx 4.3 \cdot 10^{-6}$



(b) $h \approx 5.8 \cdot 10^{-7}$

Figure 2.26: 6 isolines for self-similar Mach numbers $\tilde{M}$ in the range 0.996 to 1.02. The red isoline corresponds to the sonic line, i.e., $\tilde{M} = 1$. The h is the approximate minimum cell width in the neighborhood of the supersonic patches.

2.3.2.2 Shock diffraction around a thin film

We now consider the problem of a shock interacting with a thin, reflecting film on a square domain $[-1.125, 1.125]^2$. The thin film is located on the line $\xi = 0$ between $\eta = -1.125$ and $\eta = 0$. The incident, horizontally oriented shock propagates downward to the thin film. Once it hits the film, it reflects and diffracts around the obstacle. The incident shock is weak and propagates at Mach 1.075. As the shock diffracts, it transforms into an expansion wave in a self-similar fashion. Another characteristic of the flow is the development of a vortex at the corner of the thin film. The incident (I) and reflected (R) shocks as well as the point (P) where the shock disappears are illustrated in Figure 2.27a, after the incident shock has passed the thin film. Since this interaction is self-similar, this figure also illustrates the boundary conditions applied in (ξ, η) coordinates, which are given by three constant states $\mathbf{U}_R$, $\mathbf{U}_I$, $\mathbf{U}_Q$, i.e., the reflected, incident, and quiescent states (Table 2.7). The thin film is modeled by a reflecting internal boundary condition, indicated on the initial mesh in Figure 2.27b by a bold line. The reflected shock state $\mathbf{U}_R$ is computed from the incident shock state $\mathbf{U}_I$ by solving a one-dimensional Riemann problem about the thin film (reflecting boundary).

We compute the sonic function

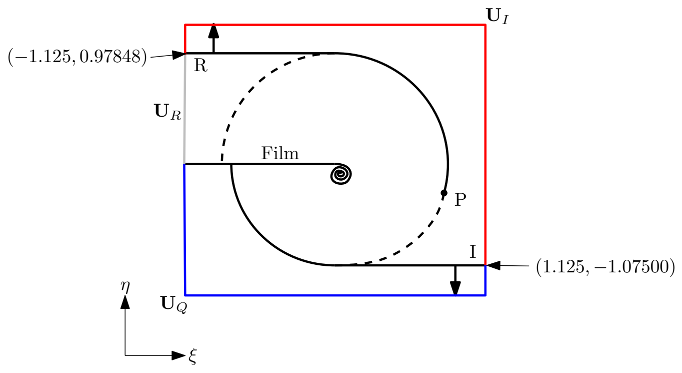
$$S = \sqrt{(u-\xi)^2 + (v-\eta)^2} - c.$$

The flow is subsonic when $S < 0$, supersonic when $S > 0$, and the sonic line is located where $S = 0$. For this problem, refinement is driven by proximity to the sonic line. The final mesh is comprised of $\sim$ 2,800,000 elements where the smallest resolution is $h \approx 4.8 \cdot 10^{-5}$ in the neighborhood of the sonic line.

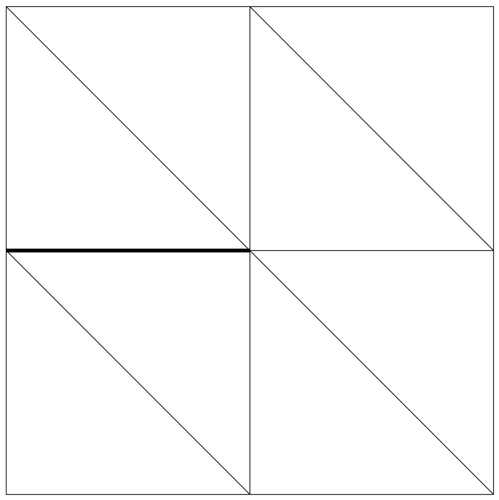
The sonic function of the solution is plotted in Figure 2.28a. In Figures 2.28b and 2.28c, cross sections of S are provided in the neighborhood of the shock disappearance point on $\eta = 0, -0.05, ..., -0.25$. On these cross sections, the shock is visible at the coordinates (ξ, η): $(0, 1.0173)$, $(-0.05, 1.0219)$, and $(-0.1, 1.0240)$. The location of shock disappearance seems to be about $\sim (-0.15, 1.0237)$, a more precise location is difficult to determine. Finally, shock disappearance seems to occur on the sonic line as indicated in [50] for the UTSDE.

	$\mathbf{U}_I$	$\mathbf{U}_R$	$\mathbf{U}_Q$
ρ	1.57697	1.77139	1.4
u	0	0	0
v	-0.12064	0	0
p	1.18156	1.39066	1

Table 2.7: The incident $\mathbf{U}_I$, reflected $\mathbf{U}_R$, and quiescent $\mathbf{U}_Q$ states imposed as boundary conditions in Example 2.3.2.2.

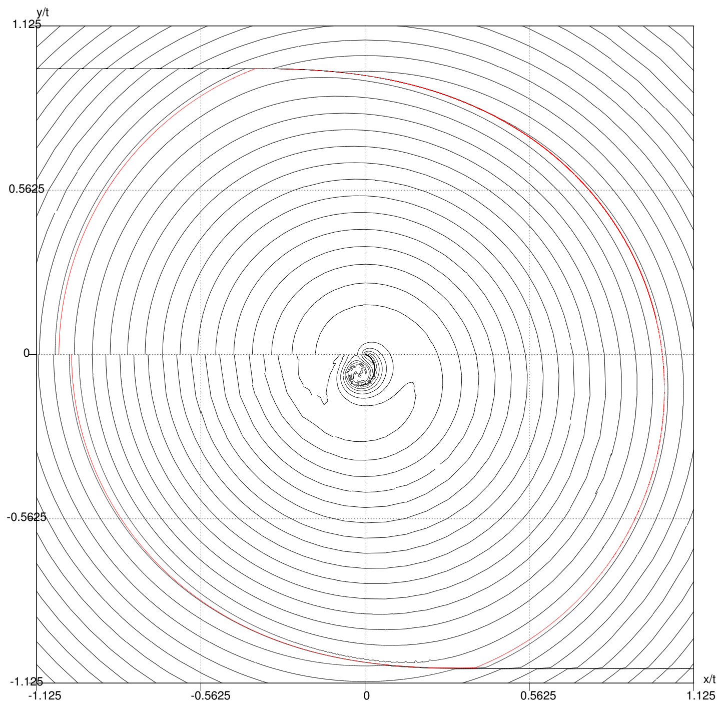


(a) The incident (I) shock propagates into quiescent (Q) gas, diffracts around, and reflects (R) off the thin film. The shock disappears at P. The incident $\mathbf{U}_I$, reflected $\mathbf{U}_R$, and quiescent $\mathbf{U}_Q$ states are imposed as boundary conditions on the red, gray, and blue boundaries, respectively.

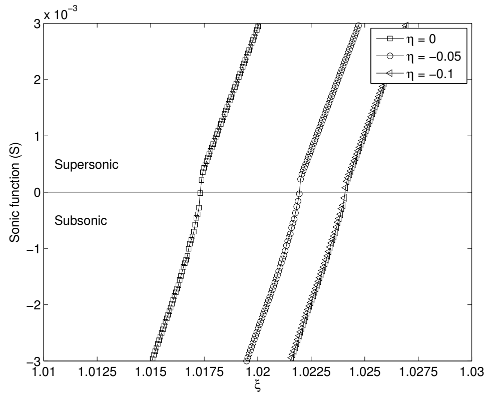


(b) Initial mesh for the shock diffraction problem. The bolded line corresponds to the thin film.

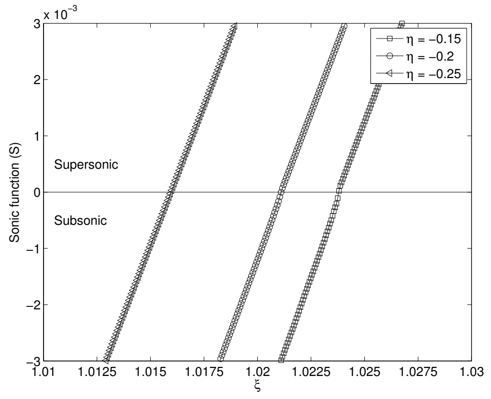
Figure 2.27: Initial setup and mesh for the shock diffraction problem in Example 2.3.2.2.



(a) Isolines of sonic function. The red isoline corresponds to the sonic line, i.e., where $S = 0$.



(b) Cross sections of S on $\eta = 0, -0.05, -0.1$.



(c) Cross sections of S on $\eta = -0.15, -0.2, -0.25$.

Figure 2.28: Sonic function (S) and its cross sections for the shock diffraction problem in Example 2.3.2.2.

2.4 Summary

We have outlined a GPU-parallelized h-adaptive implementation of the DG method for hyperbolic conservation laws on unstructured meshes. The highlights of this implementation are memory management and the use of a coloring algorithm to eliminate race conditions. Our memory management techniques allow for quickly resizing the data arrays resulting in the smallest number of necessary memory transfers. This is done by using a modified stream-compaction operation. The coloring algorithm prevents race conditions in the evaluation of an integral over cell edges. It is also used in the smoothing subroutines to ensure proper nonconformity between elements. In fact, the smoothing module is the most expensive part of the AMR algorithm. This procedure can easily be done recursively on CPUs, but it is difficult to implement efficiently on GPUs. This is because the smoothing on one element may trigger smoothing of its neighbors. Using coloring yields a lightweight implementation relative to an element-wise operation.

We have presented a number of computed examples in gas dynamics demonstrating the performance of the AMR algorithm. In particular, we computed two problems that are intractable without AMR due to the extremely high resolution required to resolve the solution features. We present numerical evidence that further supports Guderley's solution to the von Neumann triple point paradox. We also include numerical experiments that suggest shock disappearance occurs on the sonic line in the self-similar diffraction of a shock around a thin film. To our knowledge, these are the first results to the shock diffraction problem on the full Euler equations.

Chapter 3

Slope limiters in two dimensions

Weak solutions of hyperbolic PDEs admit discontinuities, which can lead to nonphysical oscillations when high order numerical methods are employed. A popular technique to stabilize the growth of these oscillations for methods that are formally second order accurate is slope limiting. The gradient is computed directly by differentiating a polynomial solution, e.g. in Galerkin methods, or reconstructed using neighboring solution means, e.g. finite volume (FV) methods. A limiting algorithm will modify, or limit, this gradient so that the solution at suitable points belongs to a specified, local range.

For one-dimensional problems, slope limiters that ensure a total variation diminishing (TVD) property are frequently used [32,51–53]. In two dimensions, enforcing a TVD property can lead to at most first order schemes [54]. A weaker requirement on the numerical solution is enforcement of a local maximum principle, studied in [55] on two-dimensional structured grids for steady state computations. This idea is used in [35] to reconstruct non-oscillatory gradients on unstructured meshes of triangles. This limiter is quite popular due to its ease of implementation and computational simplicity. It consists in writing the numerical solution as a sum of the cell mean and slope. The slope is then reduced by a scalar between 0 and 1 such that the numerical solution at predetermined points lies in a locally defined interval. Some limiters modify the x and y components of the gradient separately, e.g. [36], by solving a small linear program on each element. Slope limiters can operate on solution values at the edge midpoints [56], at the neighboring cell centroids [36], or cell vertices as in [57–59]. In contrast to the above methods, classified as monoslope methods, multislope methods have also been studied whereby the solution is reconstructed and limited independently at each face of the element [60, 61].

Much literature on limiters has been devoted to finite volume methods. When transitioning to the discontinuous Galerkin (DG) method, often the same limiters are applied. However, second-order limiters for the discontinuous Galerkin method have been presented in, e.g. [62] for one-dimensional problems and [9] for multidimensional problems. The limiter proposed in [63] requires precomputation of several mesh-dependent geometric parameters on each cell, which increases computational complexity. This explains the popularity of coupling the DG method with the so-called Barth-Jespersen limiter [35]: no geometric data needs to be precomputed, and the limiter does not require a stencil larger than that of the DG method. Another second-order limiter for the DG method on triangles was presented in [8] and requires solving an optimization problem.

Classical limiters operate only on the linear approximations to the solution. Limiters that work on higher than second order accurate approximations are needed and a significant effort has been placed into finding such limiters. In [10, 11], the idea of moment limiters was proposed, whereby the numerical solution's dth derivative is limited using the $(d-1)$th derivatives on neighboring cells. Generalizations of the moment limiter to unstructured meshes were studied in [34, 64]. Different approaches to high order limiting were described in [65, 66].

In this chapter, we analyze the Barth-Jespersen limiter [35] on two-dimensional unstructured meshes of triangles, applied to linear and nonlinear problems using the DG method. This limiter has been addressed in [67] for finite volume methods, but not for the DG method. Despite its popularity, we argue that in its simplest form, it is not a well performing limiter for the DG method.

The simplest implementation of the Barth-Jespersen limiter uses the edge neighborhood and edge midpoints as limiting points. With these choices, we show that unstructured meshes are unlikely to yield second-order accurate numerical solutions, defeating the purpose of high-resolution numerical methods. For these meshes, we show that the way a refinement study is conducted will influence the observed rate of convergence of the solution. For example, refinement obtained by tiling the initial mesh or remeshing the domain at a reduced cell size can yield first order convergence. One may observe second-order accuracy in the L_1 norm with nested refinement, but first-order accuracy is still observed in the L_∞ norm. We prove that a remedy of this problem is to choose an alternative limiting neighborhood, such that the limiting points lie in the admissibility region that we define.

In our analysis of second order limiters applied to DG, we address two issues: stability and accuracy. For stability, we have proven that the numerical solution for scalar equations will satisfy a local maximum principle that ensures L_∞ stability, provided a suitable time step restriction is enforced. This new time step restriction follows from the stability analysis, and uses a new measure of cell size, which is the cell width in the direction of flow. We show with numerical experiments that this time step restriction is tight. The new measure is approximately double the radius of the inscribed circle, typically used with maximum principle limiters and the DG method. As a result, the maximum allowable time step doubles and the amount of computational work halves.

From our analysis, we find the range to which the cell means of the solution at the next time step will belong, provided the above time step restriction is enforced. This range is determined by the solution averages on nearby elements, i.e. on the neighbors used in the limiting procedure. There is freedom in defining this neighborhood, e.g. we can choose the elements that share edges or we can choose elements that share vertices with the element being limited [68]. These neighborhoods are the most natural ones, though others are possible, e.g. the entire mesh. We find that smaller neighborhoods introduce too much numerical diffusion. In particular, limiting with the edge neighborhood is too diffusive. On the other hand, if the neighborhood has a large and variable size, e.g. vertex neighborhood on an unstructured grid, this can yield almost a threefold increase in the time spent executing the limiting subroutines. This has implications for limiters that use vertex-type neighborhoods [57, 59, 65, 69].

The other aspect of the limiting algorithm is the choice of points at which the numerical solution is checked for overshoots, i.e. the algorithm's limiting points. In this chapter, we study the one- and two-point Gauss-Legendre quadrature nodes as limiting points. Checking for oscillations at quadrature points comes naturally in the DG implementation. This is because the basis functions at these points are often precomputed, therefore solution values can be obtained efficiently. Other choices are theoretically possible though seldom done in practice. We have proven that one- and two-point limiting are sufficient for the stability of linear and nonlinear problems, respectively. Two-point limiting may lead to first order accuracy and catastrophically diffusive solutions on edge neighborhoods. Numerical experiments verify that two-point limiting is more diffusive than one-point limiting on all neighborhoods, though the difference is small for the vertex neighborhood. While the one-point limiter with nonlinear fluxes will not guarantee that the minimum and maximum of the solution are maintained, for all problems that we considered, the growth in the means was small. Finally, we find that the number of limiting points does not affect code run time as drastically as the size of the neighborhood. In the numerical experiments section (Section 3.6), we discuss which combination of limiting points and neighborhoods should be used.

3.1 Limiting algorithm

With the following limiting algorithm, we seek to enforce the local maximum principle. The numerical solution satisfies the local maximum principle if

$$\min_{j\in\mathcal{N}_i}\overline{U}_j^n \leq \overline{U}_i^{n+1} \leq \max_{j\in\mathcal{N}_i}\overline{U}_j^n, \tag{3.1}$$

where $\mathcal{N}_i$ is a set containing the index of Ω_i and the indices of elements in the neighborhood of Ω_i, and $\overline{U}_i^n$ is the cell average.

We previously defined the numerical solution in terms of basis functions and degrees of freedom. Here, we rewrite it in terms of the cell average and slope at time step n:

$$U_i^n(\mathbf{x}) = \overline{U}_i^n + \nabla U_i^n \cdot (\mathbf{x} - \mathbf{x}_i), \tag{3.2}$$

where $\mathbf{x}_i$ is the centroid of Ω_i, $\overline{U}_i^n$ is the cell average, and ∇U_i^n is the solution gradient. The limiting procedure applied to the numerical solution on Ω_i multiplies the gradient by a coefficient α_i, with the aim to enforce the maximum principle (3.1) on the means at the next time step. The limited solution $\tilde{U}_i^n(\mathbf{x})$ is of the form

$$\tilde{U}_i^n(\mathbf{x}) = \overline{U}_i^n + \alpha_i \nabla U_i^n \cdot (\mathbf{x} - \mathbf{x}_i). \tag{3.3}$$

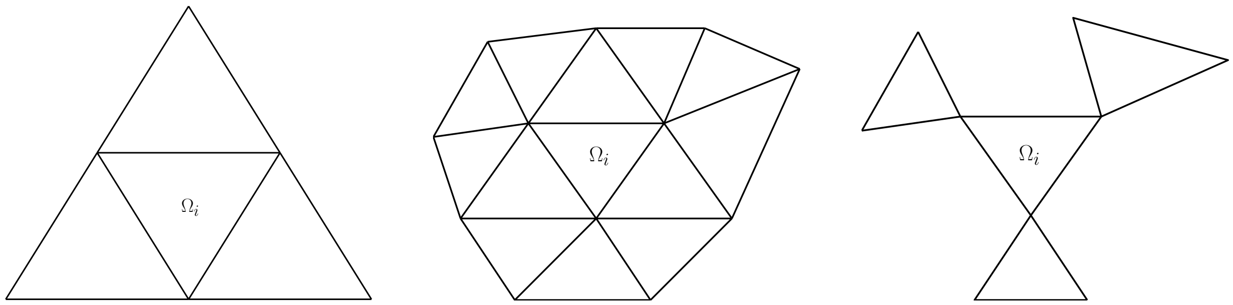
Limiting is done by comparing the values of $U_i(\mathbf{x})$ to the solution averages on neighboring elements, where the points $\mathbf{x}$ can be quadrature points, element vertices, edge midpoints, or other. We refer to these points as limiting points. If the solution at the limiting points falls outside of the range defined by its neighbors, its slope is reduced by α_i.

We collect the indices of the elements used in limiting the slope on Ω_i in a set. As with limiting points, there is freedom in choosing a suitable neighborhood of Ω_i. For example, the edge neighborhood is comprised of Ω_i itself and all the elements that share an edge with it, we refer to the set of these indices as N_i^e. The vertex neighborhood is comprised of Ω_i itself and all elements that share a vertex with it, we refer to the set of these indices as N_i^v. We can also choose a reduced subset of N_i^v, which we refer to as N_i^r. These neighborhoods are illustrated in Figure 3.1.

We execute the following algorithm to compute α_i:

1. Compute the minimum and maximum cell means on the elements in N_i:

$$m_i^n = \min_{j\in N_i}\overline{U}_j^n \quad \text{and} \quad M_i^n = \max_{j\in N_i}\overline{U}_j^n. \tag{3.4}$$



(a) Edge-neighborhood N_i^e: Ω_i and all elements that share an edge with Ω_i.

(b) Vertex-neighborhood N_i^v: Ω_i and all elements that share a vertex with Ω_i.

(c) Reduced neighborhood N_i^r: Ω_i and three vertex neighbors.

Figure 3.1: Edge, vertex, and reduced neighborhoods of Ω_i.

2. Compute the coefficient $y_i(\mathbf{x}_l)$ at each limiting point $\mathbf{x}_l$

$$y_i(\mathbf{x}_l) = \begin{cases} \frac{M_i^n - \overline{U}_i^n}{U_i^n(\mathbf{x}_l) - \overline{U}_i^n}, & \text{if } U_i^n(\mathbf{x}_l) - \overline{U}_i^n > 0, \\ \frac{m_i^n - \overline{U}_i^n}{U_i^n(\mathbf{x}_l) - \overline{U}_i^n}, & \text{if } U_i^n(\mathbf{x}_l) - \overline{U}_i^n < 0, \\ 1, & \text{otherwise.} \end{cases}$$

3. Find the smallest $y_i(\mathbf{x}_l)$ on Ω_i

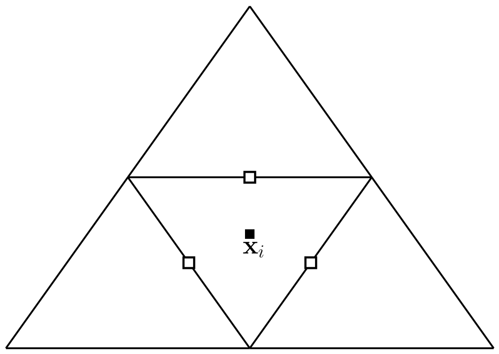
$$y_i = \min_l y_i(\mathbf{x}_l).$$

4. If $y_i \in (0,1)$, then the solution is outside the locally defined range, $[m_i^n, M_i^n]$, for at least one limiting point. Scaling the gradient by $\alpha_i = y_i$ brings that value into the prescribed range. If $y_i > 1$, then the solution at the limiting points lies in the range, i.e. the current slope is acceptable and should not be modified, so $\alpha_i = 1$. Combining the above into one formula, we have

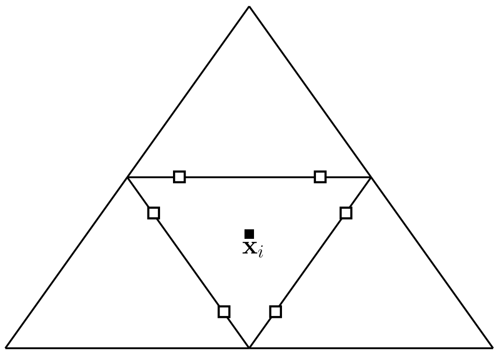
$$\alpha_i = \min(y_i, 1).$$

5. The limited numerical solution $\tilde{U}_i^n$ is now given by (3.3).

After limiting, the polynomial $\tilde{U}_i^n(\mathbf{x})$ is rewritten in terms of the basis functions and the simulation continues.



(a) One-point Gauss-Legendre quadrature rule.



(b) Two-point Gauss-Legendre quadrature rule.

Figure 3.2: Nodes of quadrature rules for edge integrals.

3.2 Time integration

We propagate (1.9) in time using an explicit two-stage second order Runge-Kutta (RK) method, known as Heun's method. For a system of ODEs of the form

$$\frac{d}{dt}\mathbf{c} = L(\mathbf{c}),$$

the time stepping scheme, with a limiter, is given by Algorithm 5.

Algorithm 5 SSP-RK2 algorithm.

```
$\mathbf{c}^{(1)} = \mathbf{c}^n + \Delta t L(\mathbf{c}^n)$
Limit $\mathbf{c}^{(1)}$
$\mathbf{c}^{(2)} = \mathbf{c}^{(1)} + \Delta t L(\mathbf{c}^{(1)})$
$\mathbf{c}^{n+1} = \frac{1}{2}\mathbf{c}^n + \frac{1}{2}\mathbf{c}^{(2)}$
Limit $\mathbf{c}^{n+1}$
```

In the algorithm above, we limit the intermediate RK stage and the solution at level t^{n+1}. The stability results we prove in the next section concern one forward Euler time step, which is only first order accurate in time. The presented analysis extends to a special subset of RK methods, called Strong Stability Preserving (SSP) schemes. This is because such methods can be written as a convex combination of forward Euler steps [70]. Since each forward Euler step does not introduce new extrema, a convex combination of them will not either. Note that Heun's method is SSP.

3.3 Stability

We now prove stability of the DG scheme coupled with the limiter (3.3) under a suitable time step constraint for linear and nonlinear equations. From (1.9), with the test function $\phi_{i,0} = |\Omega_i|^{-\frac{1}{2}}$, where $|\Omega_i|$ is the area of the cell, we obtain the ordinary differential equation for propagation of the mode corresponding to the constant basis function, $c_{i,0}$,

$$\frac{d}{dt}c_{i,0} = -\frac{1}{\sqrt{|\Omega_i|}} \sum_{j\in N_i^e, j\neq i} \int_{\partial\Omega_{i,j}} \mathbf{F}^*(U_i(\mathbf{x}), U_j(\mathbf{x})) \cdot \mathbf{n}\; dl.$$

Multiplying both sides of the equation by $\phi_{i,0}$, recalling the orthonormal property of the basis, and using $\overline{U}_i = c_{i,0}\phi_{i,0}$, we have

$$\frac{d}{dt}\overline{U}_i = -\frac{1}{|\Omega_i|} \sum_{j\in N_i^e, j\neq i} \int_{\partial\Omega_{i,j}} \mathbf{F}^*(U_i(\mathbf{x}), U_j(\mathbf{x})) \cdot \mathbf{n}\; dl.$$

We apply one forward Euler time step to the equation above, and the scheme for the cell average on Ω_i becomes

$$\overline{U}_i^{n+1} = \overline{U}_i^n - \frac{\Delta t}{|\Omega_i|} \sum_{j\in N_i^e, j\neq i} \int_{\partial\Omega_{i,j}} \mathbf{F}^*(U_i^n(\mathbf{x}), U_j^n(\mathbf{x})) \cdot \mathbf{n}\; dl. \tag{3.5}$$

In the case of nonlinear fluxes $\mathbf{F}(u)$, the DG method needs to integrate the boundary integral with third order accuracy [9]. An efficient choice of approximation is the two-point Gauss-Legendre quadrature rule, with $\mathbf{x}_{i,j,q}$ being the qth quadrature point on $\partial\Omega_{i,j}$. Replacing the boundary integral in (3.5) with the quadrature rule gives

$$\overline{U}_i^{n+1} = \overline{U}_i^n - \Delta t \sum_{j\in N_i^e, j\neq i} \frac{1}{2}\frac{|\partial\Omega_{i,j}|}{|\Omega_i|} \sum_{q=1,2} \mathbf{F}^*(U_i^n(\mathbf{x}_{i,j,q}), U_j^n(\mathbf{x}_{i,j,q})) \cdot \mathbf{n}_{i,j}, \tag{3.6}$$

where $|\partial\Omega_{i,j}|$ is the length of $\partial\Omega_{i,j}$. For a linear flux, this becomes

$$\overline{U}_i^{n+1} = \overline{U}_i^n - \Delta t \sum_{j\in N_i^e, j\neq i} \frac{|\partial\Omega_{i,j}|}{|\Omega_i|} \mathbf{F}^*(U_i^n(\mathbf{x}_{i,j}), U_j^n(\mathbf{x}_{i,j})) \cdot \mathbf{n}_{i,j}, \tag{3.7}$$

where $\mathbf{x}_{i,j}$ is the midpoint of the edge shared by Ω_i and Ω_j. Before presenting the main result, we state the following proposition, the proof of which is provided in A.

Proposition 1. *For a quadrature point $\mathbf{x}$, there exists a multiplier $0 \le r \le 2$ and another quadrature point $\mathbf{x}'$ on a different edge, such that*

$$U_i(\mathbf{x}) - \overline{U}_i = r(\overline{U}_i - U_i(\mathbf{x}')).$$

For schemes (3.6) and (3.7), we have the following maximum principle result.

Theorem 1. *Let $m_i' = \min_{k\in N_i^e} m_k^n$ and $M_i' = \max_{k\in N_i^e} M_k^n$, where m_k^n and M_k^n are given by (3.4). If $m_i^n \le U_i^n(\mathbf{x}) \le M_i^n$ for all quadrature points $\mathbf{x}_{i,j}$ in (3.7) or $\mathbf{x}_{i,j,q}$ in (3.6), and Δt is subject to the CFL constraint*

$$\Delta t \le \frac{1}{6} \min_i \frac{h_{c,i}}{\lambda_i}, \tag{3.8}$$

where $h_{c,i}$ is the radius of the inscribed circle of Ω_i, and λ_i is the magnitude of the wave speed on Ω_i, then

$$\overline{U}_i^{n+1} \in [m_i', M_i']. \tag{3.9}$$

That is, the schemes (3.6) and (3.7) satisfy the local maximum principle (3.1).

Proof. The proof consists of three steps. First, we write the solution mean at t^{n+1}, $\overline{U}_i^{n+1}$, in the following form

$$\overline{U}_i^{n+1} = d_i\overline{U}_i^n + \sum d_j U_j^n(\mathbf{x}), \tag{3.10}$$

where the sum is over all edge quadrature points $\mathbf{x}$, and $U_j^n(\mathbf{x})$ are understood to be the solution values from either inside or outside the element Ω_i. Next, we show that under the CFL constraint (3.8), the coefficients d_j are non-negative, and their sum is equal to 1, i.e. they have

1. sum property:

$$d_i + \sum d_j = 1, \tag{3.11}$$

2. non-negativity property:

$$d_j \geq 0. \tag{3.12}$$

This means that $\overline{U}_i^{n+1}$ in (3.10) is a convex combination of solution values at t^n. Upon application of the limiter (3.3), these values will be bounded, i.e. we have

3. limiting property:

$$U_j^n(\mathbf{x}) \in \left[\min_{k\in N_j} \overline{U}_k^n, \max_{k\in N_j} \overline{U}_k^n\right] = [m_j^n, M_j^n],$$

where $\mathbf{x}$ is understood to be an edge quadrature point. Finally, if the conditions in properties 1, 2, and 3 are satisfied, then the bounds (3.9) on $\overline{U}_i^{n+1}$ directly follow. We now prove the theorem for linear problems, i.e. (1.1) with linear fluxes.

Linear problems. For linear problems we use the upwind numerical flux, which is given by

$$\mathbf{F}^*(U_i^n(\mathbf{x}_{i,j}), U_j^n(\mathbf{x}_{i,j})) \cdot \mathbf{n}_{i,j} = \begin{cases} (\mathbf{a}\cdot\mathbf{n}_{i,j})U_j^n(\mathbf{x}_{i,j}) \text{ if } j \in N_i^{e,-}, \\ (\mathbf{a}\cdot\mathbf{n}_{i,j})U_i^n(\mathbf{x}_{i,j}) \text{ if } j \in N_i^{e,+}, \end{cases}$$

where $N_i^{e,-}$ and $N_i^{e,+}$ are the sets of inflow and outflow neighbors, respectively, i.e. $N_i^{e,\pm} = \{j : j \in N_i^e, j \neq i, \text{ such that } \pm\mathbf{a}\cdot\mathbf{n}_{i,j} > 0\}$. Therefore, scheme (3.7) becomes

$$\overline{U}_i^{n+1} = \overline{U}_i^n + \Delta t \sum_{j\in N_i^{e,-}} |\mathbf{a}\cdot\mathbf{n}_{i,j}| \frac{|\partial\Omega_{i,j}|}{|\Omega_i|} U_j^n(\mathbf{x}_{i,j}) - \Delta t \sum_{j\in N_i^{e,+}} |\mathbf{a}\cdot\mathbf{n}_{i,j}| \frac{|\partial\Omega_{i,j}|}{|\Omega_i|} U_i^n(\mathbf{x}_{i,j}). \tag{3.13}$$

By the divergence theorem, we have the following relation

$$\sum_{j\in N_i^e, j\neq i} |\partial\Omega_{i,j}| \mathbf{a}\cdot\mathbf{n}_{i,j} = 0. \tag{3.14}$$

Using (3.14) in (3.13), we have

$$\begin{aligned} \overline{U}_i^{n+1} = \overline{U}_i^n &+ \Delta t \sum_{j\in N_i^{e,-}} |\mathbf{a}\cdot\mathbf{n}_{i,j}| \frac{|\partial\Omega_{i,j}|}{|\Omega_i|} (U_j^n(\mathbf{x}_{i,j}) - \overline{U}_i^n) \\ &- \Delta t \sum_{j\in N_i^{e,+}} |\mathbf{a}\cdot\mathbf{n}_{i,j}| \frac{|\partial\Omega_{i,j}|}{|\Omega_i|} (U_i^n(\mathbf{x}_{i,j}) - \overline{U}_i^n). \end{aligned}$$

Applying Proposition 1 to the outflow terms in the previous equation, we obtain

$$\overline{U}_i^{n+1} = \overline{U}_i^n + \Delta t \sum_{j \in N_i^{e,-}} |\mathbf{a} \cdot \mathbf{n}_{i,j}| \frac{|\partial \Omega_{i,j}|}{|\Omega_i|} (U_j^n(\mathbf{x}_{i,j}) - \overline{U}_i^n)$$
$$- \Delta t \sum_{j \in N_i^{e,+}} |\mathbf{a} \cdot \mathbf{n}_{i,j}| \frac{|\partial \Omega_{i,j}|}{|\Omega_i|} r_{i,j} (\overline{U}_i^n - U_i^n(\mathbf{x}_{i,j'})),$$

where $r_{i,j}$ is the scaling coefficient r on edge $\partial\Omega_{i,j}$. Grouping terms allows us to write the above equation in the form (3.10):

$$\overline{U}_i^{n+1} = \left(1 - \Delta t \sum_{j \in N_i^{e,-}} |\mathbf{a} \cdot \mathbf{n}_{i,j}| \frac{|\partial \Omega_{i,j}|}{|\Omega_i|} - \Delta t \sum_{j \in N_i^{e,+}} |\mathbf{a} \cdot \mathbf{n}_{i,j}| \frac{|\partial \Omega_{i,j}|}{|\Omega_i|} r_{i,j}\right) \overline{U}_i^n$$
$$+ \Delta t \sum_{j \in N_i^{e,-}} |\mathbf{a} \cdot \mathbf{n}_{i,j}| \frac{|\partial \Omega_{i,j}|}{|\Omega_i|} U_j^n(\mathbf{x}_{i,j}) + \Delta t \sum_{j \in N_i^{e,+}} |\mathbf{a} \cdot \mathbf{n}_{i,j}| \frac{|\partial \Omega_{i,j}|}{|\Omega_i|} r_{i,j} U_i^n(\mathbf{x}_{i,j'}). \quad (3.15)$$

We will now prove Properties 1 and 2.

Sum. The sum constraint is automatically satisfied because

$$\begin{aligned} d_i + \sum d_j &= \left(1 - \Delta t \sum_{j \in N_i^{e,+}} r_{i,j} |\mathbf{a} \cdot \mathbf{n}_{i,j}| \frac{|\partial \Omega_{i,j}|}{|\Omega_i|} - \Delta t \sum_{j \in N_i^{e,-}} |\mathbf{a} \cdot \mathbf{n}_{i,j}| \frac{|\partial \Omega_{i,j}|}{|\Omega_i|}\right) \\ &\quad + \Delta t \sum_{j \in N_i^{e,+}} r_{i,j} |\mathbf{a} \cdot \mathbf{n}_{i,j}| \frac{|\partial \Omega_{i,j}|}{|\Omega_i|} + \Delta t \sum_{j \in N_i^{e,-}} |\mathbf{a} \cdot \mathbf{n}_{i,j}| \frac{|\partial \Omega_{i,j}|}{|\Omega_i|} \\ &= 1. \end{aligned}$$

Non-negativity. First, note that the coefficients in (3.15)

$$\Delta t |\mathbf{a} \cdot \mathbf{n}_{i,j}| \frac{|\partial \Omega_{i,j}|}{|\Omega_i|} \quad \text{and} \quad \Delta t |\mathbf{a} \cdot \mathbf{n}_{i,j}| \frac{|\partial \Omega_{i,j}|}{|\Omega_i|} r_{i,j}$$

corresponding to d_j in (3.10) are always non-negative. Next, we will choose a local stable time step Δt_i such that d_i is non-negative as well, i.e.

$$d_i = 1 - \Delta t_i \sum_{j \in N_i^{e,-}} |\mathbf{a} \cdot \mathbf{n}_{i,j}| \frac{|\partial \Omega_{i,j}|}{|\Omega_i|} - \Delta t_i \sum_{j \in N_i^{e,+}} |\mathbf{a} \cdot \mathbf{n}_{i,j}| \frac{|\partial \Omega_{i,j}|}{|\Omega_i|} r_{i,j} \geq 0. \tag{3.16}$$

Observing that $|\mathbf{a} \cdot \mathbf{n}_{i,j}| \leq ||\mathbf{a}||$, $r_{i,j} \leq 2$, extending the sums from $N_i^{e,\pm}$ to N_i^e, and rearranging the terms, we obtain the following upper bound on the sum terms in (3.16)

$$\Delta t_i \sum_{j \in N_i^{e,-}} |\mathbf{a} \cdot \mathbf{n}_{i,j}| \frac{|\partial \Omega_{i,j}|}{|\Omega_i|} + \Delta t_i \sum_{j \in N_i^{e,+}} |\mathbf{a} \cdot \mathbf{n}_{i,j}| \frac{|\partial \Omega_{i,j}|}{|\Omega_i|} r_{i,j} \leq 3 \Delta t_i ||\mathbf{a}|| \frac{\sum_{j \in N_i^e, j \neq i} |\partial \Omega_{i,j}|}{|\Omega_i|}.$$

Coefficient d_i will be non-negative if

$$3 \Delta t_i ||\mathbf{a}|| \frac{\sum_{j \in N_i^e, j \neq i} |\partial \Omega_{i,j}|}{|\Omega_i|} \leq 1.$$

Solving for Δt_i yields the sufficient condition for the non-negativity property (3.12)

$$\Delta t_i \leq \frac{1}{6} \frac{h_{c,i}}{||\mathbf{a}||},$$

where

$$h_{c,i} = 2 \frac{|\Omega_i|}{|\partial \Omega_i|},$$

$|\partial \Omega_i|$ is the perimeter of Ω_i, and $h_{c,i}$ is the radius of the circle inscribed in Ω_i. Then the non-negativity constraint on the entire mesh is

$$\Delta t \leq \frac{1}{6} \min_i \frac{h_{c,i}}{||\mathbf{a}||}. \tag{3.17}$$

Finally, property 3 is guaranteed by limiter (3.3). Thus (3.9) is true, and the linear scheme (3.7) is L_∞ non-increasing with time in the means.

Nonlinear problems. We consider the scheme (3.6), and use the notation $F_{i,j}(U_1, U_2) = \mathbf{F}^*(U_1, U_2) \cdot \mathbf{n}_{i,j}$. Similar to the linear case, we use the divergence theorem to obtain

$$\sum_{j \in N_i^e, j \neq i} |\partial \Omega_{i,j}| F_{i,j}(\overline{U}_i^n, \overline{U}_i^n) = 0. \tag{3.18}$$

Using (3.18) in (3.6), we obtain

$$\overline{U}_i^{n+1} = \overline{U}_i^n - \Delta t \sum_{j\in N_i^e, j\neq i} \frac{1}{2}\frac{|\partial\Omega_{i,j}|}{|\Omega_i|} \sum_{q=1,2} \left\{ F_{i,j}(U_i^n(\mathbf{x}_{i,j,q}), U_j^n(\mathbf{x}_{i,j,q})) - F_{i,j}(\overline{U}_i^n, \overline{U}_i^n) \right\}.$$

Adding and subtracting $F_{i,j}(\overline{U}_i^n, U_j^n(\mathbf{x}_{i,j,q}))$ in the inner sum, we have

$$\begin{aligned}\overline{U}_i^{n+1} = \overline{U}_i^n - \Delta t \sum_{j\in N_i^e, j\neq i} \frac{1}{2}\frac{|\partial\Omega_{i,j}|}{|\Omega_i|} \sum_{q=1,2} &\{F_{i,j}(U_i^n(\mathbf{x}_{i,j,q}), U_j^n(\mathbf{x}_{i,j,q})) - F_{i,j}(\overline{U}_i^n, U_j^n(\mathbf{x}_{i,j,q}))\} \\ &+\{F_{i,j}(\overline{U}_i^n, U_j^n(\mathbf{x}_{i,j,q})) - F_{i,j}(\overline{U}_i^n, \overline{U}_i^n)\}.\end{aligned}$$

Using the mean value theorem, we obtain

$$\begin{aligned}\overline{U}_i^{n+1} = \overline{U}_i^n - \Delta t \sum_{j\in N_i^e, j\neq i} \frac{1}{2}\frac{|\partial\Omega_{i,j}|}{|\Omega_i|} \sum_{q=1,2} &\frac{\partial F_{i,j}}{\partial U_1}(\xi_1, U_j^n(\mathbf{x}_{i,j,q}))(U_i^n(\mathbf{x}_{i,j,q}) - \overline{U}_i^n) \\ &+ \frac{\partial F_{i,j}}{\partial U_2}(\overline{U}_i^n, \xi_2)(U_j^n(\mathbf{x}_{i,j,q}) - \overline{U}_i^n)\end{aligned}$$

with $\frac{\partial F_{i,j}}{\partial U_1}$ and $\frac{\partial F_{i,j}}{\partial U_2}$ as the partial derivatives with respect to the first and second arguments of $F_{i,j}$, respectively, ξ_1 between $\overline{U}_i^n$ and $U_i^n(\mathbf{x}_{i,j,q})$ and ξ_2 between $\overline{U}_i^n$ and $U_j^n(\mathbf{x}_{i,j,q})$. Introducing $v_{i,j,q}^1 = \Delta t_i \frac{|\partial\Omega_{i,j}|}{|\Omega_i|}\frac{\partial F_{i,j}}{\partial U_1}(\xi_1, U_j^n(\mathbf{x}_{i,j,q}))$, and $v_{i,j,q}^2 = \Delta t_i \frac{|\partial\Omega_{i,j}|}{|\Omega_i|}\frac{\partial F_{i,j}}{\partial U_2}(\overline{U}_i^n, \xi_2)$, we have

$$\overline{U}_i^{n+1} = \overline{U}_i^n + \sum_{j\in N_i^e, j\neq i} \frac{1}{2} \sum_{q=1,2} v_{i,j,q}^1(\overline{U}_i^n - U_i^n(\mathbf{x}_{i,j,q})) - v_{i,j,q}^2(U_j^n(\mathbf{x}_{i,j,q}) - \overline{U}_i^n).$$

By the monotonicity property of the numerical flux

$$v_{i,j,q}^1 \geq 0 \text{ and } -v_{i,j,q}^2 \geq 0. \tag{3.19}$$

As in the case of a linear flux, we apply Proposition 1 to the $\overline{U}_i^n - U_i^n(\mathbf{x}_{i,j,q})$ term, i.e.

$$\overline{U}_i^{n+1} = \overline{U}_i^n + \sum_{j\in N_i^e, j\neq i} \frac{1}{2} \sum_{q=1,2} v_{i,j,q}^1 r_{i,j,q}(U_i^n(\mathbf{x}_{i,j',q}) - \overline{U}_i^n) - v_{i,j,q}^2(U_j^n(\mathbf{x}_{i,j,q}) - \overline{U}_i^n),$$

where $r_{i,j,q}$ is the scaling coefficient r on edge $\partial\Omega_{i,j}$ at the qth quadrature point. Grouping terms yields

$$\begin{aligned}\overline{U}_i^{n+1} &= \left(1 - \sum_{j\in N_i^e, j\neq i} \sum_{q=1,2} \frac{1}{2}(v_{i,j,q}^1 r_{i,j,q} - v_{i,j,q}^2)\right)\overline{U}_i^n \\ &\qquad + \sum_{j\in N_i^e, j\neq i} \frac{1}{2} \sum_{q=1,2} \left[v_{i,j,q}^1 r_{i,j,q} U_i^n(\mathbf{x}_{i,j',q}) - v_{i,j,q}^2 U_j^n(\mathbf{x}_{i,j,q})\right]. \quad (3.20)\end{aligned}$$

This is of the form (3.10). We will now prove properties 1 and 2.

Sum. The sum constraint is automatically satisfied because

$$\begin{aligned}d_i + \sum d_j &= 1 - \sum_{j\in N_i^e, j\neq i} \sum_{q=1,2} \frac{1}{2}(v_{i,j,q}^1 r_{i,j,q} - v_{i,j,q}^2) \\ &\quad + \sum_{j\in N_i^e, j\neq i} \{\frac{1}{2}\sum_{q=1,2} v_{i,j,q}^1 r_{i,j,q} - \frac{1}{2}\sum_{q=1,2} v_{i,j,q}^2\} \\ &= 1.\end{aligned}$$

Non-negativity. First, note that the coefficients in (3.20) corresponding to the d_j coefficients in (3.10) are always non-negative by (3.19). Next, we will choose a Δt_i such that d_i is non-negative as well, i.e.

$$d_i = 1 - \sum_{j\in N_i^e, j\neq i} \sum_{q=1,2} \frac{1}{2}(v_{i,j,q}^1 r_{i,j,q} - v_{i,j,q}^2) \geq 0.$$

Due to the differentiability of the numerical flux, there exists a λ_i such that $\frac{\partial F_{i,j}}{\partial U_1}(\xi_1, U_j^n(\mathbf{x}_{i,j,q})) \leq \lambda_i$ and $-\frac{\partial F_{i,j}}{\partial U_2}(\overline{U}_i^n, \xi_2) \leq \lambda_i$ hold. Similar to the linear case, a sufficient condition on the local time step Δt_i is

$$6\Delta t_i \frac{\lambda_i}{h_{c,i}} \leq 1.$$

A time step suitable for all elements is determined by minimizing the ratio $\frac{h_{c,i}}{\lambda_i}$, i.e.

$$\Delta t \leq \frac{1}{6}\min_i \frac{h_{c,i}}{\lambda_i}.$$

Finally, Property 3 is enforced with limiter (3.3). Thus (3.9) is true and the nonlinear scheme (3.6) is L_∞ non-increasing. $\square$

Remark.

Here we derive a less restrictive CFL condition for linear problems. Consider the coefficient

$$d_i = 1 - \Delta t \sum_{j \in N_i^{e,-}} |\mathbf{a} \cdot \mathbf{n}_{i,j}| \frac{|\partial \Omega_{i,j}|}{|\Omega_i|} - \Delta t \sum_{j \in N_i^{e,+}} |\mathbf{a} \cdot \mathbf{n}_{i,j}| \frac{|\partial \Omega_{i,j}|}{|\Omega_i|} r_{i,j}. \tag{3.21}$$

Because $0 \leq r_{i,j} \leq 2$, it follows that d_i is bounded below by

$$1 - \Delta t \sum_{j \in N_i^{e,-}} |\mathbf{a} \cdot \mathbf{n}_{i,j}| \frac{|\partial \Omega_{i,j}|}{|\Omega_i|} - 2\Delta t \sum_{j \in N_i^{e,+}} |\mathbf{a} \cdot \mathbf{n}_{i,j}| \frac{|\partial \Omega_{i,j}|}{|\Omega_i|} \leq d_i.$$

The non-negativity of d_i is guaranteed if

$$0 \leq 1 - \Delta t \sum_{j \in N_i^{e,-}} |\mathbf{a} \cdot \mathbf{n}_{i,j}| \frac{|\partial \Omega_{i,j}|}{|\Omega_i|} - 2\Delta t \sum_{j \in N_i^{e,+}} |\mathbf{a} \cdot \mathbf{n}_{i,j}| \frac{|\partial \Omega_{i,j}|}{|\Omega_i|} \leq d_i. \tag{3.22}$$

From (3.14), we have the identity

$$- \sum_{j \in N_i^{e,-}} |\partial \Omega_{i,j}| \mathbf{a} \cdot \mathbf{n}_{i,j} = \sum_{j \in N_i^{e,+}} |\partial \Omega_{i,j}| \mathbf{a} \cdot \mathbf{n}_{i,j}.$$

Because $\mathbf{a} \cdot \mathbf{n}_{i,j} < 0$ for $j \in N_i^{e,-}$ and $\mathbf{a} \cdot \mathbf{n}_{i,j} > 0$ for $j \in N_i^{e,+}$, this becomes

$$\sum_{j \in N_i^{e,-}} |\partial \Omega_{i,j}| |\mathbf{a} \cdot \mathbf{n}_{i,j}| = \sum_{j \in N_i^{e,+}} |\partial \Omega_{i,j}| |\mathbf{a} \cdot \mathbf{n}_{i,j}|. \tag{3.23}$$

For linear problems, three situations are possible. There can be two inflow edges and one outflow edge, or one inflow edge and two outflow edges. In these two situations, there is a single inflow or a single outflow edge. We refer to that edge as $\partial \Omega_{i,J}$. Finally, when the direction of flow is parallel to an edge, i.e. is one inflow and one outflow edge. In this case, $\partial \Omega_{i,J}$ can refer to either the inflow or outflow edge. In terms of $\partial \Omega_{i,J}$, identity (3.23) now becomes

$$|\partial\Omega_{i,J}||\mathbf{a}\cdot\mathbf{n}_{i,J}| = \sum_{j\in N_i^{e,-}} |\partial\Omega_{i,j}||\mathbf{a}\cdot\mathbf{n}_{i,j}| = \sum_{j\in N_i^{e,+}} |\partial\Omega_{i,j}||\mathbf{a}\cdot\mathbf{n}_{i,j}|.$$

Using the above in (3.22), we obtain

$$0 \le 1 - 3\Delta t|\mathbf{a}\cdot\mathbf{n}_{i,J}|\frac{|\partial\Omega_{i,J}|}{|\Omega_i|}. \tag{3.24}$$

The area of the cell Ω_i is $\frac{1}{2}|\partial\Omega_{i,J}|H_{i,J}$, where $H_{i,J}$ is the height of the cell measured from the edge $\partial\Omega_{i,J}$ as shown in Figure 3.3a. Further, a simple geometric consideration reveals that $||\mathbf{a}||H_{i,J} = h_{d,i}|\mathbf{a}\cdot\mathbf{n}_J|$, where $h_{d,i}$ is the width of the cell in the direction of $\mathbf{a}$ as in Figure 3.3a. The non-negativity constraint on the entire mesh is then given by

$$\Delta t \le \frac{1}{6}\min_i \frac{h_{d,i}}{||\mathbf{a}||}. \tag{3.25}$$

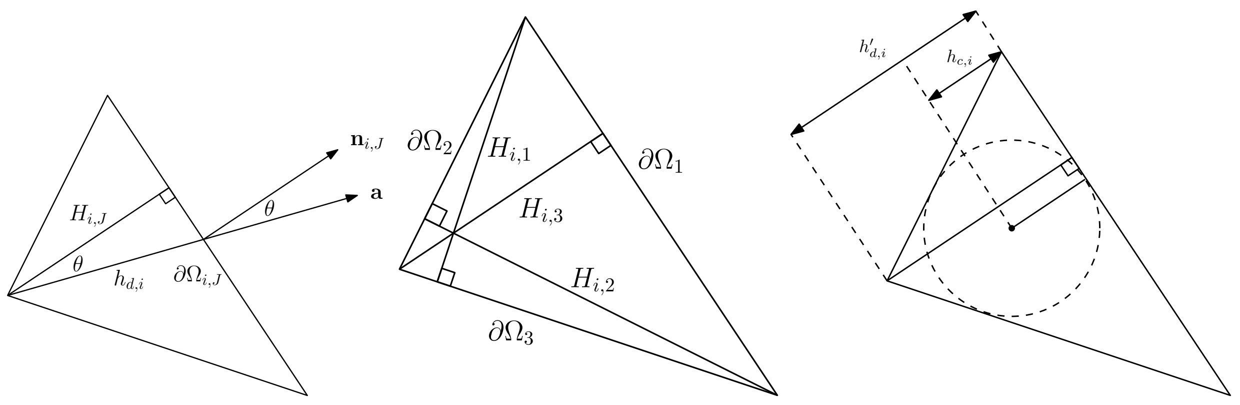
By geometrical considerations, we note that this $h_{d,i}$ is larger than the radius of the inscribed circle $h_{c,i}$ (Figure 3.3c). Therefore, this CFL condition (3.25) is less restrictive.

For systems of equations, in general, there is not a single direction along which information is propagated. For simplicity, we propose to take the minimum possible cell width, i.e.,

$$h'_{d,i} = \min(H_{i,1}, H_{i,2}, H_{i,3}), \tag{3.26}$$

where $H_{i,1}$, $H_{i,2}$, and $H_{i,3}$ are the cell widths perpendicular to the three edges of the element, $\partial\Omega_1$, $\partial\Omega_2$, and $\partial\Omega_3$, as shown in Figure 3.3b.

We have shown that at the next time step the solution means will satisfy a local maximum principle that depends on the chosen limiting neighborhood. That is, $\overline{U}_i^{n+1}$ will lie in the interval $[m', M']$, where m' and M' depend on the elements involved in limiting by (3.4). In the next section, we discuss how the choice of limiting points and neighborhoods affects the accuracy of the numerical solution.



(a) Cell size in the direction of flow $h_{d,i}$.

(b) Cell size for systems $h'_{d,i} = \min(H_{i,1}, H_{i,2}, H_{i,3})$.

(c) Comparison of minimum cell width $h'_{d,i}$ and radius of the inscribed circle $h_{c,i}$.

Figure 3.3: Measures of cell size for time step restriction (3.25).

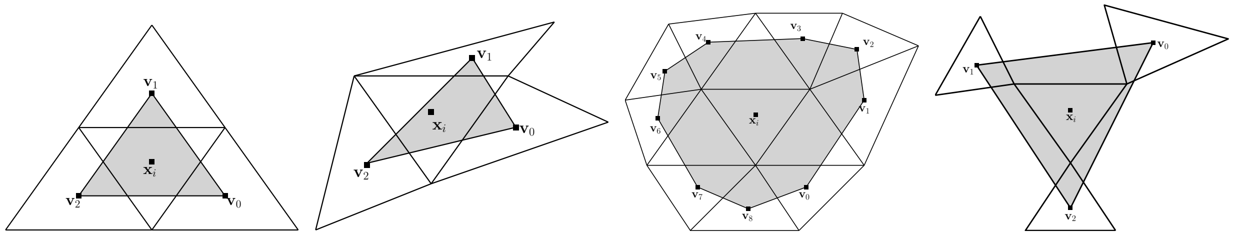
3.4 Solution accuracy and admissibility region

In order to preserve second order accuracy, the limiting algorithm (3.3) must not modify linear data. We call the set from which one can choose limiting points such that this condition is not violated the *admissibility region*. First, we give a definition of this region, and then prove in Theorem 2 that points from this region satisfy the desired property.

Definition 1. *The limiter's admissibility region is defined as the convex hull of the centroids of the elements in N_i, where N_i is the neighborhood of Ω_i involved in the limiter* (3.3)*. Geometrically, this region is a convex polygon whose vertices are labeled $\mathbf{v}_k$ and ordered counterclockwise about their barycenter (Figure 3.4). Any point $\mathbf{x}$ in the region can be written as*

$$\mathbf{x} = \sum_k \gamma_k \mathbf{v}_k,$$

$$\sum_k \gamma_k = 1 \text{ and } \gamma_k \geq 0.$$



(a) Edge-neighborhood, equilateral triangles. (b) Edge-neighborhood, deformed triangles. (c) Vertex-neighborhood. (d) Reduced-neighborhood.

Figure 3.4: Admissibility regions for Ω_i and various limiting neighborhoods.

By definition, the points $\mathbf{x}$ in the convex hull satisfy the following conditions

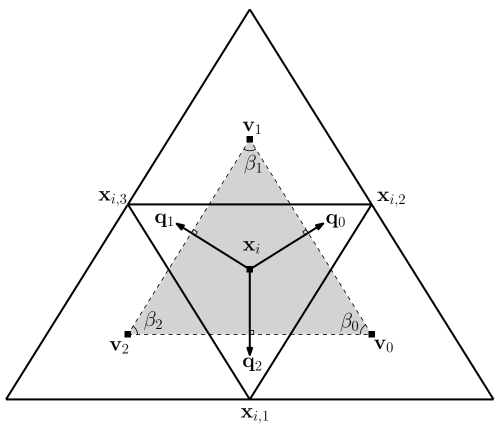
$$(\mathbf{x} - \mathbf{x}_i) \cdot \mathbf{q}_k \leq (\mathbf{v}_k - \mathbf{x}_i) \cdot \mathbf{q}_k, \tag{3.27}$$
$$(\mathbf{x} - \mathbf{x}_i) \cdot \mathbf{q}_k \leq (\mathbf{v}_{k+1} - \mathbf{x}_i) \cdot \mathbf{q}_k, \tag{3.28}$$

for all indices k, where $\mathbf{q}_k$ are outward pointing unit vectors such that $\mathbf{q}_k \cdot (\mathbf{v}_{k+1} - \mathbf{v}_k) = 0$, i.e. they are vectors perpendicular to the boundaries of the convex hull. Additionally, the pairs $\mathbf{q}_k$ and $\mathbf{q}_{k+1}$ are linearly independent.

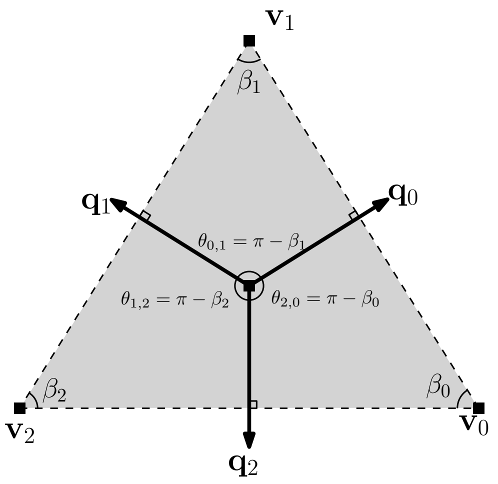
We display examples of admissibility regions in Figure 3.4. These regions depend on the neighborhood involved in computing the local minimum and maximum $[m_i^n, M_i^n]$ in the limiting procedure in Section 3.1. For the edge neighborhood the region is simply the triangle formed by connecting the centroids of the elements that share an edge with Ω_i, as shown in Figures 3.4a and 3.4b. For the vertex neighborhood, the shape is more complex, as shown in Figure 3.4c.

The admissibility region of the vertex neighborhood usually contains all the limiting points. An exception would be neighborhoods of elements that are located on the boundary of the computational domain. The number of elements in the vertex neighborhood is variable in unstructured meshes. This leads to memory inefficiencies in numerical codes as the stencil for the limiter varies from element to element. This motivates defining the reduced neighborhood. We iterate through all possible combinations of three elements in the vertex neighborhood and choose the first subset such that all limiting points are contained in its convex hull (Figure 3.4d).

Theorem 2. *(i) If the limiting points lie in the admissibility region then linear data will not be modified by the limiter* (3.3)*. (ii) If a limiting point lies outside the admissibility region, then there exists a gradient that will be modified by the limiter* (3.3)*.*



(a) Edge neighborhood, with vectors $\mathbf{q}_k$.



(b) Admissibility region of Ω_i in Figure 3.5a.

Figure 3.5: Illustration for Theorem 2, where $\Omega_i = (\mathbf{x}_{i,1}, \mathbf{x}_{i,2}, \mathbf{x}_{i,3})$, and the admissibility region is shaded.

Proof. We first prove part (i) of the theorem. Let us consider an admissibility region, which is a polygon with vertices $\mathbf{v}_k$ (Figure 3.5a). We denote by β_k the angle formed by the edges of the polygon at vertex $\mathbf{v}_k$. Since the region is convex, we have that $0 < \beta_k < \pi$. We denote by $\theta_{k,k+1}$ the angle between $\mathbf{q}_k$ and $\mathbf{q}_{k+1}$. A simple geometric consideration reveals that $\theta_{k,k+1} = \pi - \beta_k$ (Figure 3.5b) and, consequently, $0 < \theta_{k,k+1} < \pi$.

We consider a vector $\mathbf{g} = \nabla U_i$. There exists an index K such that $\mathbf{g}$ lies between $\mathbf{q}_K$ and $\mathbf{q}_{K+1}$ (Figure 3.5b). Since $0 < \theta_{K,K+1} < \pi$, we can express $\mathbf{g} = c_1\mathbf{q}_K + c_2\mathbf{q}_{K+1}$ such that $c_1, c_2 \geq 0$. Assume the limiting point $\mathbf{x} \in \Omega_i$ is in the admissibility region. Therefore, it satisfies (3.27) and (3.28), with index $k = K + 1$ and $k = K$, respectively, i.e.

$$(\mathbf{x} - \mathbf{x}_i) \cdot \mathbf{q}_K \leq (\mathbf{v}_{K+1} - \mathbf{x}_i) \cdot \mathbf{q}_K,$$
$$(\mathbf{x} - \mathbf{x}_i) \cdot \mathbf{q}_{K+1} \leq (\mathbf{v}_{K+1} - \mathbf{x}_i) \cdot \mathbf{q}_{K+1}.$$

Multiplying the first inequality by c_1, and the second by c_2, then summing, we have

$$(\mathbf{x} - \mathbf{x}_i) \cdot \mathbf{g} \leq (\mathbf{v}_{K+1} - \mathbf{x}_i) \cdot \mathbf{g}.$$

Adding the cell average $\overline{U}_i$, we have by (3.2),

$$U_i(\mathbf{x}) \leq U_i(\mathbf{v}_{K+1}).$$

Therefore,

$$U_i(\mathbf{x}) \leq M_i,$$

where M_i is given in (3.4). The same reasoning can be applied to $\mathbf{g} = -\nabla U_i$, which gives

$$-U_i(\mathbf{x}) \leq -m_i.$$

Therefore,

$$m_i \leq U_i(\mathbf{x}),$$

where m_i is given by (3.4). Therefore $m_i \leq U_i(\mathbf{x}) \leq M_i$, $\mathbf{x} \in \Omega_i$, and by the limiter algorithm (3.3), the slope is not limited.

We now prove part (ii) of the theorem by constructing a solution that will be limited by algorithm (3.3) if a limiting point $\mathbf{x} \in \Omega_i$ lies outside of the admissibility region. In this case, at least one inequality (3.27) or (3.28), e.g. (3.27) with index K, will not hold:

$$(\mathbf{v}_K - \mathbf{x}_i) \cdot \mathbf{q}_K < (\mathbf{x} - \mathbf{x}_i) \cdot \mathbf{q}_K.$$

However, each of the vertices $\mathbf{v}_k$ of the admissibility region belongs to the region itself, therefore by (3.27) and above, we have

$$(\mathbf{v}_k - \mathbf{x}_i) \cdot \mathbf{q}_K \leq (\mathbf{v}_K - \mathbf{x}_i) \cdot \mathbf{q}_K < (\mathbf{x} - \mathbf{x}_i) \cdot \mathbf{q}_K. \tag{3.29}$$

Let us assume that the global solution is a plane, whose gradient is $\mathbf{q}_K$. We add to (3.29) the solution average $\overline{U}_i$ on Ω_i. Using (3.2), we obtain

$$U_i(\mathbf{v}_k) < U_i(\mathbf{x}), \quad \forall k. \tag{3.30}$$

Additionally, because the vertices of the admissibility region correspond to neighboring element centroids, we have $U_i(\mathbf{v}_k) = \overline{U}_k$. Taking the maximum over k in (3.30) gives

$$M_i < U_i(\mathbf{x}).$$

Since U_i evaluated at $\mathbf{x}$ exceeds its allowed range, the slope of U_i will be limited by the limiting algorithm (3.3). □

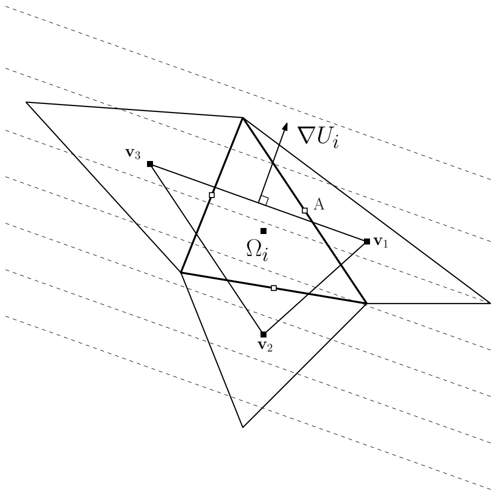


Figure 3.6: Limiting point A is outside the admissibility region of Ω_i, described by $\mathbf{v}_1$, $\mathbf{v}_2$, and $\mathbf{v}_3$.

Remark: Part (ii) of this theorem has a simple geometric interpretation that is illustrated in Figure 3.6. U_i is a linear function whose isolines are parallel lines. Since the isoline passing through the limiting point, A, lies higher than the centroids on the neighboring elements, the value of U_i at A exceeds the value at the neighboring centroids and the numerical solution on Ω_i will be limited.

3.5 Refinement studies

In the following discussion, we argue that limiting with the edge neighborhood should not be used with the discontinuous Galerkin method. With this limiter, we are not guaranteed that an element's limiting points will all lie in the admissibility region. This will lead to a reduced rate of convergence on smooth solutions. The observed rate of convergence under mesh refinement will depend on a number of factors: the quality of the initial mesh, the particular numerical solution, and the method of refinement. To illustrate this point, we construct two sequences of meshes and conduct numerical simulations that demonstrate different convergence behaviors. Here we analyze only the midpoint limiter, the two-point limiter will perform even worse.

All numerical simulations were done using the DG code described in [3] written for NVIDIA GPUs, using the code optimizations described in [7].

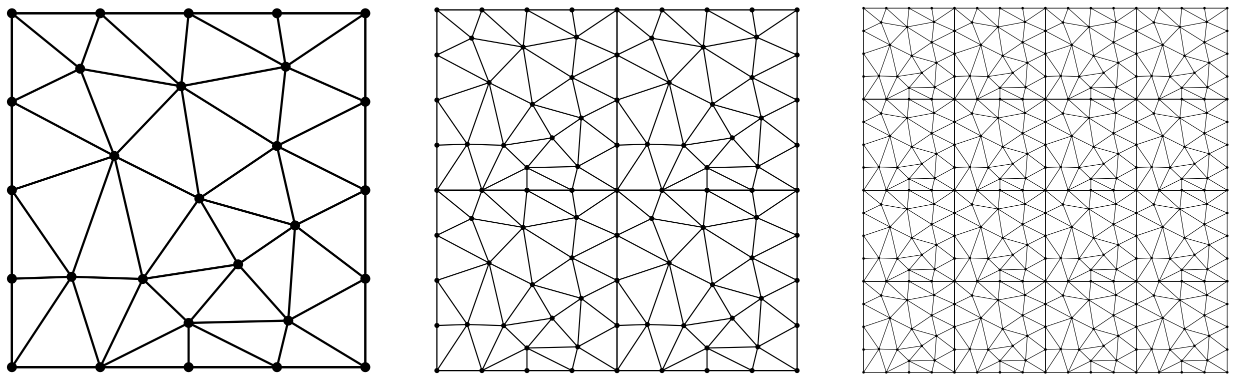

(a) Ω^0. (b) Ω^1. (c) Ω^2.

Figure 3.7: Mesh sequence obtained through tiling.

3.5.1 Tiled refinement

We start with the initial mesh Ω_0 of a square domain Ω. Then, Ω^0 is scaled by a factor of $\frac{1}{2}$, and tiled over Ω to obtain the next mesh in the sequence Ω_1; that is, Ω^1 is composed of four scaled copies of Ω^0. We continue in a similar fashion, i.e. Ω_2 contains 16 scaled copies of Ω_0. We show a sample initial mesh, and two subsequent meshes obtained through tiling in Figure 3.7. The initial mesh is arbitrary with the only restriction that vertex placement on opposing boundaries is identical. This is needed in order to avoid nonconforming elements on the boundary of adjacent tiles. To simplify this discussion, we assume that elements on tile boundaries are not limited.

To demonstrate a loss of accuracy under limiting, we examine the limiting operation applied to a linear function $u(x, y)$. On Ω_i, this function can be written as

$$u_i(x, y) = a(x - x_i) + b(y - y_i) + \overline{u}_i,$$

where $\overline{u}_i$ is the average over Ω_i, x_i, y_i are the coordinates of the cell centroid. After applying the limiter to $u_i(x, y)$, we obtain the limited function

$$\tilde{U}_i(x, y) = \alpha_i a(x - x_i) + \alpha_i b(y - y_i) + \overline{u}_i,$$

where $\alpha_i \in [0, 1]$ is the limiting coefficient. This coefficient will not change upon translation or scaling of the mesh, provided the numerical solution is linear.

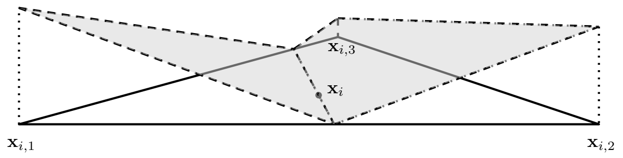


Figure 3.8: Shaded surfaces are described by the integrand $|a(x-x_i)+b(y-y_i)|$ in (3.31).

The L_1 norm of the error introduced due to limiting is

$$\begin{aligned} E_1(\Omega^0) &= \sum_i \int_{\Omega_i} |u_i(x,y) - \tilde{U}_i(x,y)| dxdy \\ &= \sum_i (1-\alpha_i) \int_{\Omega_i} |a(x-x_i) + b(y-y_i)| dxdy. \end{aligned} \tag{3.31}$$

Each integral in the sum has a geometrical interpretation of the volume of two polyhedra since $a(x-x_i)+b(y-y_i)$ is zero along a line passing through the centroid of Ω_i. This is illustrated in Figure 3.8, where the shaded planes are the surfaces described by the integrand.

Shrinking the mesh by a factor of two, $\mathbf{x}' = \frac{1}{2}\mathbf{x}$, shrinks the volume of the polyhedra and, therefore, the error by a factor of eight. We have on the scaled mesh, Ω', the L_1 error

$$E_1(\Omega') = \frac{1}{8} E_1(\Omega^0).$$

Additionally, translating the mesh, $\mathbf{x}' = \mathbf{x}+\mathbf{d}$, does not change the error. On the translated mesh, Ω', the L_1 error is

$$\begin{aligned} E_1(\Omega') &= \sum_i (1-\alpha_i) \int_{\Omega_i} |a((x+d_x)-(x_i+d_x)) + b((y+d_y)-(y_i+d_y))| dxdy \\ &= \sum_i (1-\alpha_i) \int_{\Omega_i} |a(x-x_i)+b(y-y_i)| dxdy \\ &= E_1(\Omega^0). \end{aligned}$$

To summarize, scaling the Ω^0 by a factor of two reduces the error by a factor of eight and translation does not affect the error. Thus, the L_1 error on Ω^1, which consists of four scaled and translated copies of Ω_0 (Figure 3.7) is

$$E_1(\Omega^1) = \frac{4}{8}E_1(\Omega^0) = \frac{1}{2}E_1(\Omega^0).$$

This implies that the nth mesh in the tiled sequence has the error

$$E_1(\Omega^n) = \left(\frac{1}{2}\right)^n E_1(\Omega^0),$$

which indicates at most first order convergence in the general case.

3.5.2 Nested refinement

We now define a mesh sequence for which the same limiter, i.e. edge midpoints as limiting points coupled with the edge neighborhood, will yield a second order approximation of the initial data. We start by considering a mesh consisting of one element, Ω_i (Figure 3.9a). It is refined by splitting into four children, Ω_j, Ω_k, Ω_l, and Ω_m (Figure 3.9b). We can show that on the center child element of Ω_i, in this case Ω_j, linear data will not be limited. To show this, note that the limiting points of Ω_j, $\boldsymbol{\xi}$, are the midpoints of its edges:

$$\begin{aligned}
\boldsymbol{\xi}_{j,1} &= \frac{1}{2}(\mathbf{x}_{j,3} + \mathbf{x}_{j,1}),\\
\boldsymbol{\xi}_{j,2} &= \frac{1}{2}(\mathbf{x}_{j,1} + \mathbf{x}_{j,2}),\\
\boldsymbol{\xi}_{j,3} &= \frac{1}{2}(\mathbf{x}_{j,2} + \mathbf{x}_{j,3}),
\end{aligned} \tag{3.32}$$

where $\mathbf{x}_{j,1}$, $\mathbf{x}_{j,2}$, and $\mathbf{x}_{j,3}$ are the vertices of Ω_j. Further, the vertices of Ω_j are the midpoints of Ω_i's edges:

$$\begin{aligned}
\mathbf{x}_{j,1} &= \frac{1}{2}(\mathbf{x}_{i,1} + \mathbf{x}_{i,2}),\\
\mathbf{x}_{j,2} &= \frac{1}{2}(\mathbf{x}_{i,2} + \mathbf{x}_{i,3}),\\
\mathbf{x}_{j,3} &= \frac{1}{2}(\mathbf{x}_{i,3} + \mathbf{x}_{i,1}),
\end{aligned} \tag{3.33}$$

where $\mathbf{x}_{i,1}$, $\mathbf{x}_{i,2}$, and $\mathbf{x}_{i,3}$ are the vertices of Ω_i. Combining (3.32) and (3.33), we have

$$\begin{pmatrix} \boldsymbol{\xi}_{j,1} \\ \boldsymbol{\xi}_{j,2} \\ \boldsymbol{\xi}_{j,3} \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{2} \end{pmatrix} \begin{pmatrix} \mathbf{x}_{i,1} \\ \mathbf{x}_{i,2} \\ \mathbf{x}_{i,3} \end{pmatrix}. \tag{3.34}$$

We also write the centroids $\mathbf{x}_m$, $\mathbf{x}_k$, and $\mathbf{x}_l$ in terms of $\mathbf{x}_{i,1}$, $\mathbf{x}_{i,2}$, and $\mathbf{x}_{i,3}$:

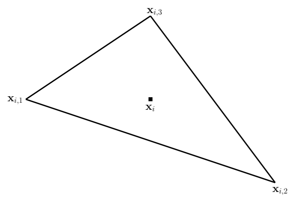
$$\begin{pmatrix} \mathbf{x}_m \\ \mathbf{x}_k \\ \mathbf{x}_l \end{pmatrix} = \begin{pmatrix} \frac{2}{3} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} & \frac{2}{3} \end{pmatrix} \begin{pmatrix} \mathbf{x}_{i,1} \\ \mathbf{x}_{i,2} \\ \mathbf{x}_{i,3} \end{pmatrix}. \tag{3.35}$$

Combining (3.34) and (3.35), we obtain

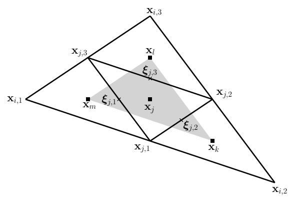
$$\begin{pmatrix} \boldsymbol{\xi}_{j,1} \\ \boldsymbol{\xi}_{j,2} \\ \boldsymbol{\xi}_{j,3} \end{pmatrix} = \begin{pmatrix} \frac{2}{3} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} & \frac{2}{3} \end{pmatrix} \begin{pmatrix} \mathbf{x}_m \\ \mathbf{x}_k \\ \mathbf{x}_l \end{pmatrix}.$$

Therefore, by Definition 1 the limiting points of Ω_j belong to its admissibility region. Thus linear data on Ω_j will not be limited by (3.3). An example is given in Figure 3.9b, where the limiting points lie inside the shaded admissibility region. In Figure 3.9c, the third mesh in the sequence, Ω^2, is shown. A simple geometric consideration reveals that elements that do not share an edge with the boundary of the original element will not be limited. This is because they are the center element of Ω^1, scaled by a factor of $\frac{1}{2}$, translated and rotated. Therefore, elements on which linear data is limited can only appear on the boundaries of the initial mesh elements in Ω^0 (Figure 3.9d). In the nth mesh of this sequence, there are $3(2^n - 1)$, $n > 0$, elements on the boundary.

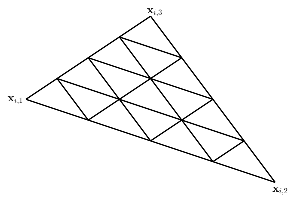
We now construct a nested refinement sequence starting with an arbitrary initial triangulation, Ω^0, with N_0 triangles, of a square domain Ω, e.g. the initial mesh in Section 3.5.1. To obtain the first mesh in the sequence, Ω^1, we refine each cell as described above. Completing this procedure n times gives the nth mesh in the sequence, Ω^n, which has $4^n N_0$ elements. The number of elements that can possibly be limited in Ω^n is $N_0 \cdot 3(2^n - 1)$, for $n > 0$. The upper bound on the number of elements on which the function will be approximated to first order grows linearly with h decreasing while the number of elements in the mesh grows quadratically, where h is a measure of cell size. This yields an effective second order rate of convergence in an integral norm of an approximation of the initial data, because the limited elements form a smaller and smaller proportion of the total number of elements in the mesh. This is easy to see by noting that the error on limited elements is $\mathcal{O}(h)$, the



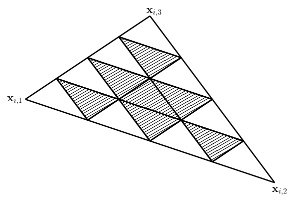
(a) Ω^0, one element Ω_i.



(b) Ω^1, four child elements Ω_j, Ω_k, Ω_l, and Ω_m. Grey zone is the admissibility region for Ω_j.



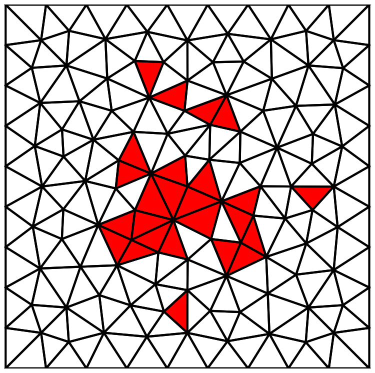
(c) Ω^2, sixteen child elements.

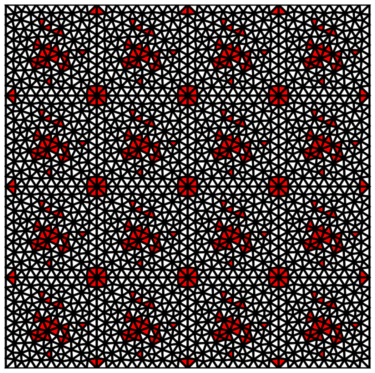


(d) Linear data on hatched elements in Ω^2 will not be limited by (3.3).

Figure 3.9: Mesh sequence obtained through nested refinement, with Ω^0 in (a) as the starting mesh.

area of an element is $\mathcal{O}(h^2)$, and the number of limited elements is $\mathcal{O}(h^{-1})$. The product of these estimates gives $\mathcal{O}(h^2)$.


(a) Initial mesh.


(b) Third mesh in tiling sequence.

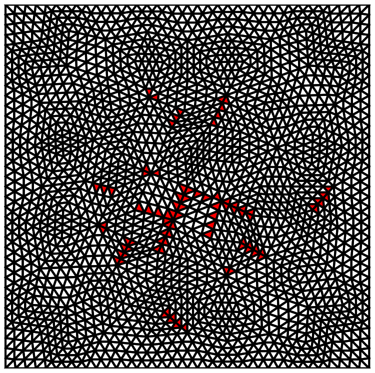

(c) Third mesh in nested refinement sequence.

Figure 3.10: Sample meshes from tiling and nested refinement sequences. The elements on which linear data is limited are shaded.

3.6 Numerical experiments

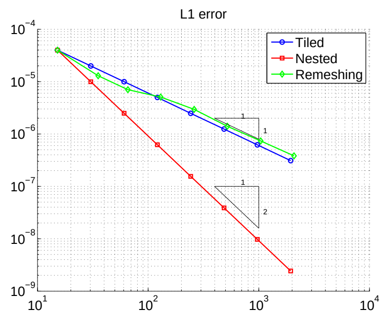
Unless otherwise specified, the problem is solved on the domain $[-1, 1]^2$, using the upwind or Lax-Friedrichs numerical flux on linear and nonlinear problems, respectively. We use the Heun's method to integrate in time unless otherwise stated and specify the particular time step restriction used in each example.

3.6.1 Accuracy verification - linear exact solution

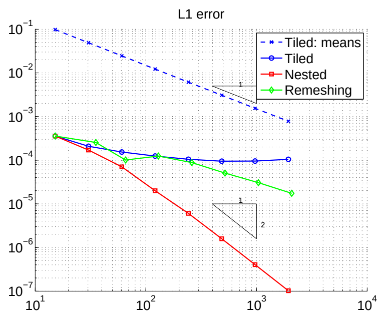
We solve (1) with the flux $\mathbf{F}(u) = [u, u]$, on the tiled and nested mesh sequences (Figure 3.10) described in Section 3.5 with the initial and boundary condition chosen such that the exact solution is

$$u(x, y, t) = t - \frac{1}{2}x - \frac{1}{2}y. \tag{3.36}$$

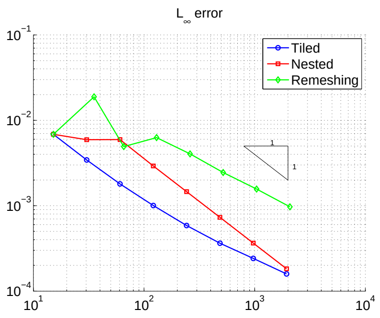
We use the limiter based on the edge midpoints coupled with the edge neighborhood. First we project and limit the initial condition on the mesh sequences and report the error resulting from the application of the limiter in Figure 3.11a. Since there is no error due to projection of the initial condition into the finite element space, the observed error is entirely due to one application of the limiter. We observe the first and second order rate of convergence as discussed in Sections 3.5.1 and 3.5.2 for the tiled and nested mesh sequences, respectively. We also construct another sequence by remeshing the domain, whereby each



(a) L_1 error after application of the limiter on the initial projection.



(b) L_1 error at the final time $T = 1$.

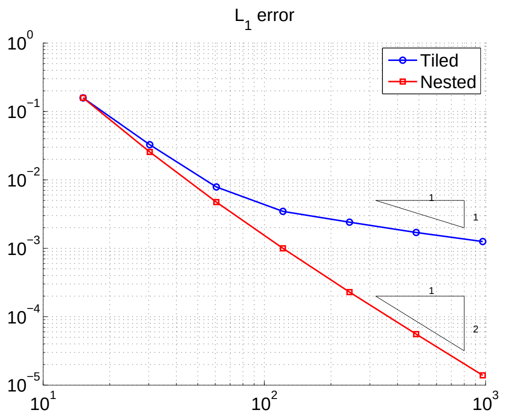


(c) L_∞ error at the final time $T = 1$.

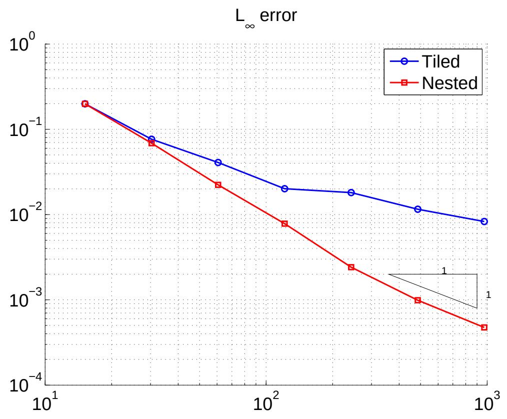
Figure 3.11: Convergence history of initial condition (3.36) after projection and limiting, and final solution at $T = 1$ of (1.1) in Section 3.6.1. We plot the error versus (number of elements)$^{\frac{1}{2}}$.

mesh in the sequence has a comparable number of elements to nested and tiled refinement. The L_1 error on the remeshed sequence behaves similarly to tiled refinement, i.e. with first order accuracy.

Next, we examine the global error due to the cumulative effect of the interaction between the limiter and the DG method at the final time $T = 1$. In these experiments, we use the time step restriction based on the radius of the inscribed circle $h_{c,i}$ in (3.17) (Figures 3.11b and 3.11c). The limiter severely affects the solutions on the tiled sequence. The maximum error initially decreases with a rate of one, and then the rate tapers off to approximately 0.6. However, in the L_1 norm, convergence appears to stall. The error even increases at the last two solutions in the sequence. Note that the solution means are still converging with first order accuracy (Figure 3.11b); in the plot, we refer to this error with 'Tiled: means'. For nested refinement, we observe quadratic convergence in the L_1 norm and linear convergence in the L_∞ norm. The reason for the first order convergence in the L_∞ norm is that some elements are limited. On those elements, the accuracy is only first order. However, we observe second order convergence in the L_1 norm because the number of limited elements is small relative to the total number of elements in the mesh (for example Figure 3.10c). Even though the analysis in Section 3.5.2 is only valid for the first application of the limiter, its conclusion seems to hold for multiple applications during a simulation.



(a) L_1 error at the final time $T = 1$.



(b) L_∞ error at the final time $T = 1$.

Figure 3.12: Convergence history of the final solution of (1.1) in Section 3.6.2. We plot the error versus (number of elements)$^{\frac{1}{2}}$.

3.6.2 Accuracy verification - nonlinear exact solution

We consider (1.1) with the flux $\mathbf{F}(u) = [u, u]$, and the following initial condition

$$u(x, y, 0) = \sin(\pi x) \sin(\pi y), \tag{3.37}$$

along with periodic boundary conditions in the x and y directions. We use a time step restriction based on the radius of the inscribed circle, $h_{c,i}$, in (3.17). We report the global error in the numerical solution at $T = 1$ on the nested and tiled mesh sequences (Figure 3.12). We note that convergence behavior is similar to that in Section 3.6.1 (Figure 3.11).

3.6.3 Validation of CFL number for linear fluxes

These experiments verify that the less restrictive CFL condition (3.25) based on cell width in the direction of flow $h_{d,i}$ is tight. We solve (1.1) with flux $\mathbf{F}(u) = [u, u]$ and a square pulse as the initial condition

$$u(x, y, 0) = \begin{cases} 1 \text{ if } \max(|x|, |y|) \le \frac{1}{4} \\ 0 \text{ elsewhere.} \end{cases}$$

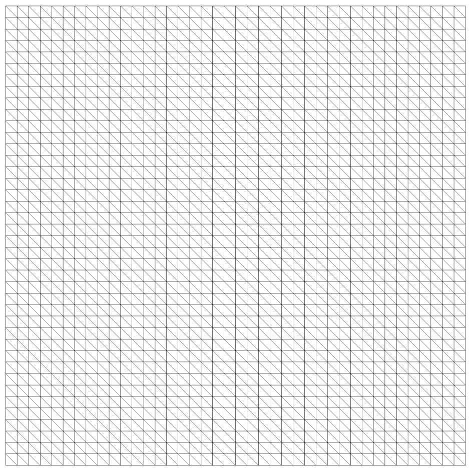


Figure 3.13: Structured mesh on which the linear CFL (3.25) based on $h_{d,i}$ is verified to be tight.

We limit at the edge midpoints and use the vertex neighborhood. The domain is divided into a 40 by 40 grid of squares. Then the squares are split into triangles by connecting the upper left and lower right corners of each square (Figure 3.13). The width of the cells in the direction of the flow is $h_{d,i} = \frac{1}{40}\sqrt{2} \approx 3.535 \cdot 10^{-2}$ and the wave speed is $||\mathbf{a}|| = \sqrt{2}$. We solve the problem until the final time $T = 0.1$ with forward Euler and RK2 time stepping. The smallest and largest of the cell-wise solution averages at the final time are reported in Table 3.1 for various values of the CFL number.

We observe that the time step restriction (3.25) is valid and tight for the forward Euler method. For RK2, we notice in Table 3.1b that the CFL number can be increased without the solution violating the local maximum principle. A possible reason is the larger absolute stability region of the RK2 family of time integrators.

3.6.4 CFL experiments for a nonlinear flux

This experiment demonstrates that both measures of cell size, radius of the inscribed circle $h_{c,i}$ in (3.17) and minimum cell height $h'_{d,i}$ in (3.26), yield solutions that appear to satisfy the maximum principle for nonlinear problems. We consider problem (1.1) with flux $\mathbf{F}(u) = \left[\frac{1}{2}u^2, \frac{1}{2}u^2\right]$ (the two-dimensional Burgers' equation). The initial condition is a square pulse of side length $\frac{1}{2}$, centered at the origin and rotated by $\frac{\pi}{4}$ (Figure 3.14a). The

1/CFL	Minimum	Maximum
3	-1.84e-01	1.23
4	-3.11-03	1.00085
5	-1.15e-05	1
5.95	-2.69e-07	1
6	**-6.56e-18**	**1**

(a) Forward Euler.

1/CFL	Minimum	Maximum
3	-9.50e-18	1.000336
4	-6.15e-18	1
5	-3.93e-18	1
5.95	-3.72e-18	1
6	**-3.62e-18**	**1**

(b) RK2-SSP.

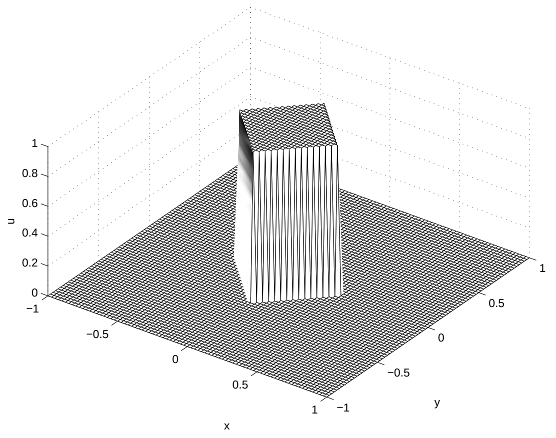
Table 3.1: Minimum and maximum cell average for Example 3.6.3 using time step restriction (3.25) based on the width of the cell in the direction of flow, $h_{d,i}$, for various CFL numbers.

exact solution at $T = \frac{\sqrt{2}}{2}$ is given by

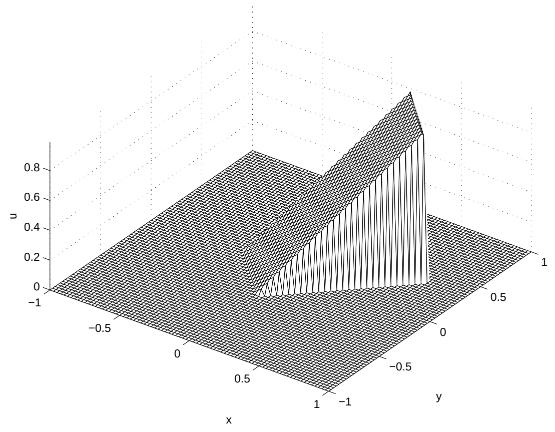
$$u\left(x', y', \frac{\sqrt{2}}{2}\right) = \begin{cases} x' + \frac{1}{4}, \text{ if } -\frac{1}{4} \leq x' \leq \frac{3}{4} \text{ and } -\frac{1}{4} \leq y' \leq \frac{1}{4}, \\ 0, \text{ otherwise,} \end{cases}$$

where $x' = \frac{\sqrt{2}}{2}x + \frac{\sqrt{2}}{2}y$ and $y' = -\frac{\sqrt{2}}{2}x + \frac{\sqrt{2}}{2}y$ (Figure 3.14b). We use the vertex neighborhood and the nodes of the two-point Gauss-Legendre quadrature rule as limiting points. The first mesh is generated by dividing the domain into a 10 by 10 grid of squares. Then the square elements are split into triangles by connecting the upper left and lower right corners of each square. Subsequent meshes we test on are obtained through nested refinement. We report the minimum and maximum cell means for both measures of cell size in Table 3.2. It appears that both time step restrictions result in solutions that are L_∞ non-increasing. However, the minimum cell height is substantially larger than the inscribed radius, which reduces the number of time steps by more than half.

We also solve this problem using the vertex neighborhood and one-point limiting. According to the analysis in Theorem 1, the solution is not guaranteed to stay within the local bounds. We find this to be the case, however the growth in the means is on the order of approximately 10^{-6}. This indicates that while the proof is correct, practically the violation of the local maximum principle is small.



(a) Exact initial condition.



(b) Exact solution at $T = \frac{\sqrt{2}}{2}$.

Figure 3.14: Exact solution at initial and final time for Example 3.6.4.

Number of elements	Time steps	Minimum	Maximum
200	85	5.22e-11	0.693
800	199	2.59e-18	0.873
3200	406	9.1e-36	0.934
12800	817	1.05e-69	0.962
51200	1637	4.14e-137	0.980

(a) Radius of the inscribed circle (3.17).

Number of elements	Time steps	Minimum	Maximum
200	35	5.53e-11	0.691
800	82	2.55e-18	0.871
3200	168	1.01e-35	0.932
12800	338	9.51e-70	0.962
51200	678	3.31e-137	0.980

(b) Minimum cell height (3.26).

Table 3.2: Verification of the time step restriction based on and radius of the inscribed circle $h_{c,i}$ in (3.17) and minimum cell height $h'_{d,i}$ in (3.26).

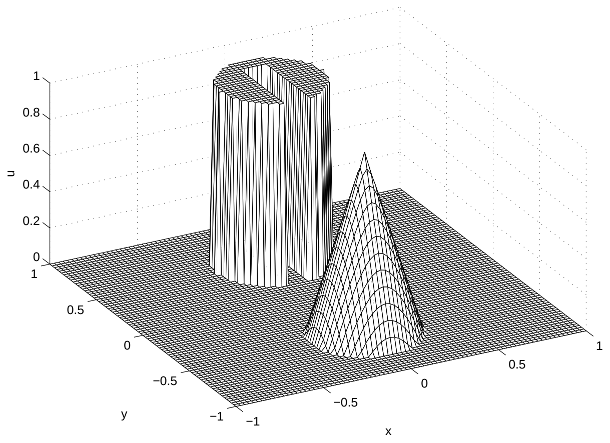


Figure 3.15: Initial condition for rotating objects test problem in Example 3.6.5.

3.6.5 Rotating objects

This example demonstrates the comparative performance of the proposed limiters, i.e., one- or two-point limiting points with edge, vertex or reduced neighborhoods. We solve (1.1) with the flux $\mathbf{F}(u) = [-2\pi yu, 2\pi xu]$ on the square domain $[-1,1]^2$. The exact solution is a rotation of the initial data about the origin. The solution comprises a slotted cylinder and a cone (Figure 3.15). Each object is defined on a disc of radius $r_0 = 0.3$, the center of which is (x_0, y_0). The height of the objects are written in terms of $r(x,y) = \frac{1}{r_0}\sqrt{(x-x_0)^2 + (y-y_0)^2}$. Outside of the discs, the initial solution values are zero. The center of the slotted cylinder is $(x_0, y_0) = (0, 0.5)$ and its height is defined as

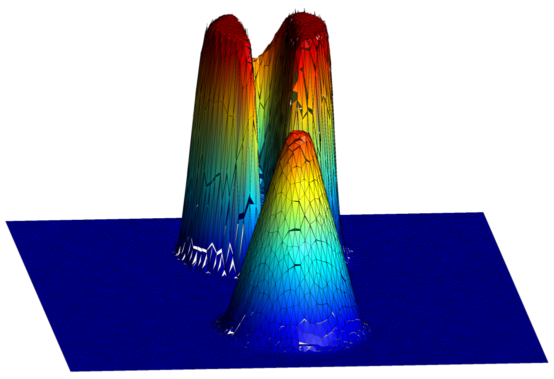
$$h(x,y) = \begin{cases} 1 \text{ if } |x - x_0| \geq 0.05 \text{ or } y \geq 0.7 \\ 0 \text{ otherwise,} \end{cases} \qquad \text{for } r(x,y) \leq 1.$$

The center of the cone is $(x_0, y_0) = (0, -0.5)$ and its height is defined as

$$h(x,y) = 1 - r(x,y) \text{ for } r(x,y) \leq 1.$$

We use the time step restriction based on the minimum cell height $h'_{d,i}$ (3.26), and an unstructured mesh of 16,870 triangles. The surface integral in (1.9) is evaluated using the two-point quadrature rule. As a result, one-point limiting does not guarantee a solution that is L_∞ non-increasing. The solutions at $T = 1$ are plotted in Figures 3.16 - 3.19. The two-point, edge neighborhood limiter is clearly the most diffusive and the one-point, vertex neighborhood limiter is the least. The two-point, edge neighborhood limiter provides a noticeably worse solution than the other limiters, Figure 3.19. The one- and two-point vertex limiters yield the least diffusive results, and perform similarly.

In conclusion, we observe that neighborhoods with fewer elements are more diffusive, e.g. edge and reduced neighborhoods and larger neighborhoods are less diffusive, e.g. vertex neighborhood. Larger neighborhoods result in larger intervals to which the numerical solution at the limiting points is constrained. Finally as expected, two-point limiting is more diffusive than one-point limiting.


(a) One-point limiting, edge neighborhood.

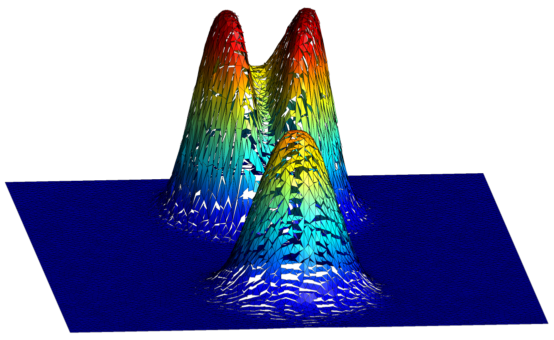
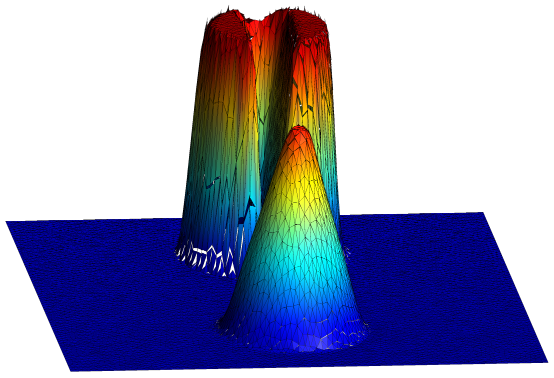
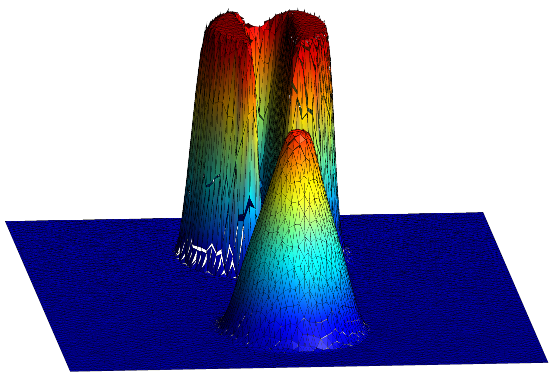

(b) Two-point limiting, edge neighborhood.

Figure 3.16: Raised solution in Example 3.6.5 for the edge neighborhoods.

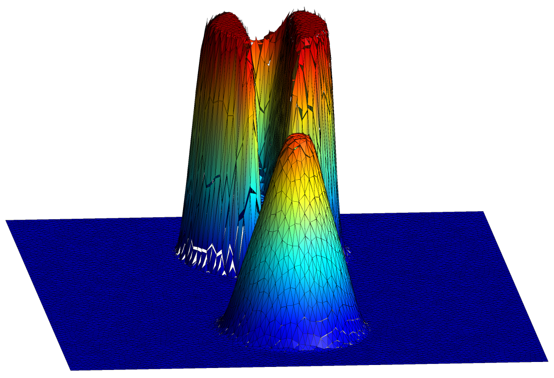


(a) One-point limiting, vertex neighborhood.

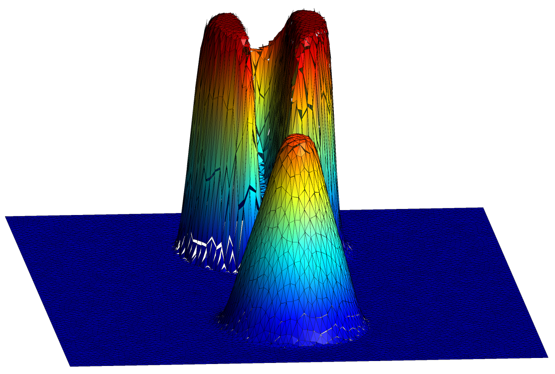


(b) Two-point limiting, vertex neighborhood.

Figure 3.17: Raised solution in Example 3.6.5 for the vertex neighborhoods.

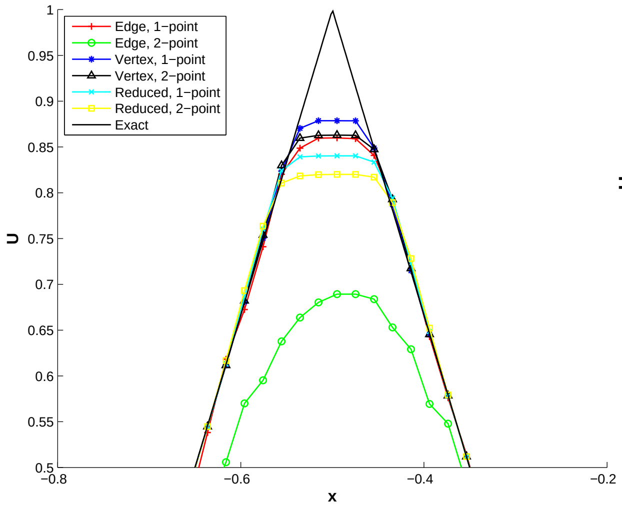


(a) One-point limiting, reduced neighborhood.

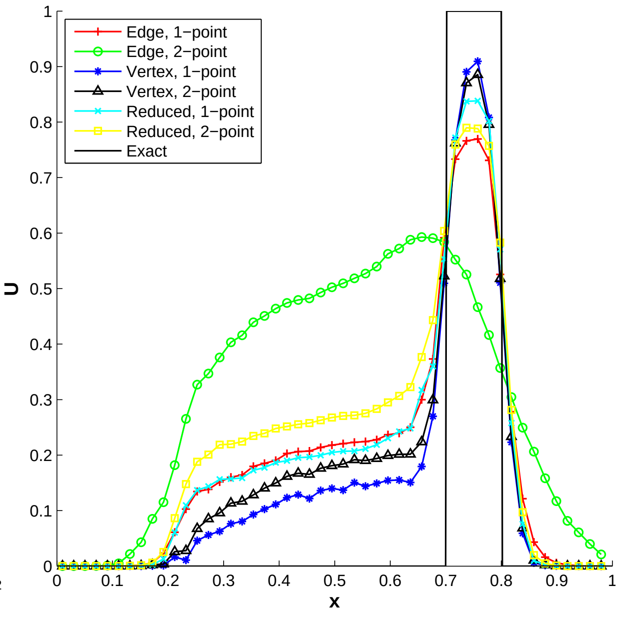


(b) Two-point limiting, reduced neighborhood.

Figure 3.18: Raised solution in Example 3.6.5 for the reduced neighborhoods.



(a) Profile at the cone.



(b) Profile at the slotted cylinder.

Figure 3.19: Cross sections of solution for Example 3.6.5 along $x = 0$ at $T = 1$.

3.6.6 Transient shock - double Mach reflection

We solve the double Mach reflection problem on an unstructured mesh of 271,458 triangles. The set-up is described in Section 2.3.1.3. We extend the limiter to systems of equations by limiting each component separately, i.e., we limit the conserved variables.

The contour plots for density using three different limiters are shown in Figures 3.20, 3.21, and 3.22. Although computationally simple, limiting using the edge neighborhood does not yield a solution of good quality. One-point limiting smears the slipline (contact discontinuity) emanating from the primary triple point in Figure 3.20a. Two-point limiting in Figure 3.20c is clearly too diffusive: the contact and reflected shock are smeared significantly and the rightward moving jet is not resolved at all.

One- and two-point limiting with the reduced neighborhood in Figures 3.21a and 3.21c performs just as poorly as one-point limiting with the edge neighborhood. Enlarging the limiting stencil to the vertex neighborhood significantly improves the solution in Figures in Figures 3.22a and 3.22c. The shocks and slipline are tighter and the jet is better resolved.

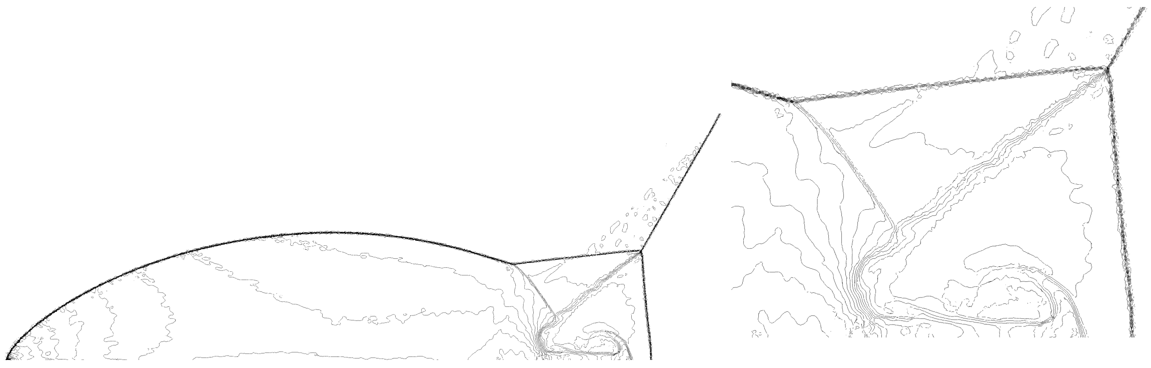
Neighborhood	limiting points	Run time (s)	Time steps	Time (ms) /step
vertex	1	51.4	8,624	5.9 (-)
reduced	1	18.7	8,409	2.2 (2.68x)
vertex	2	55.5	8,371	6.6 (-)
reduced	2	27.7	8,464	3.2 (2.06x)

Table 3.3: Comparison of run time for limiters. The number in brackets is the speed up factor of the limiters using the reduced neighborhood relative to those using the vertex neighborhood, with the same number of limiting points.

We observe more vortices due to Rayleigh-Taylor instabilities. Both one- and two-point limiting appear to be numerically stable, though two-point limiting is more diffusive.

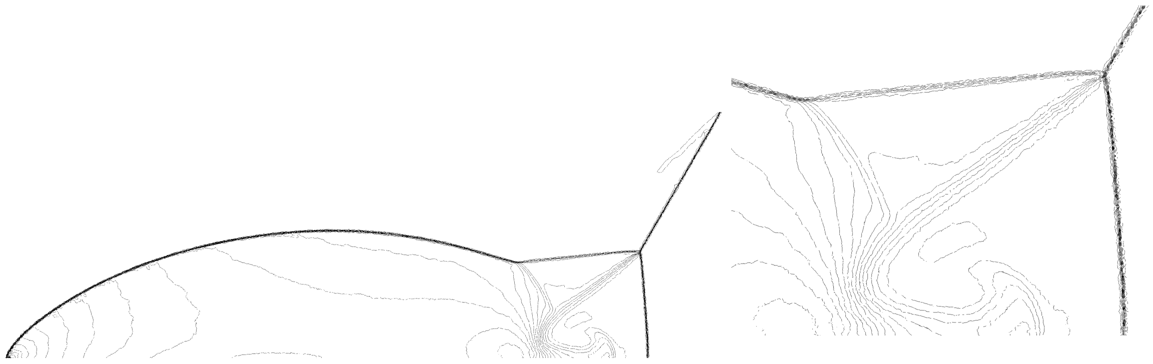
In Table 3.3, we report the time spent executing subroutines for limiters using the vertex and reduced neighborhoods in the DG-GPU code [3, 7] on an NVIDIA Titan X Pascal. The number of quadrature points does not seem to affect run time of the limiter subroutine as much as stencil size. We note that the run time is gravely affected when using a variable stencil size. For this test problem, the run time of the one-point, vertex neighborhood limiter subroutines took 5.9 ms, and the one-point, reduced neighborhood limiter subroutines was 2.2 ms; this is a 2.68x reduction in run time. On the mesh in this example, the vertex neighborhood size varies from 7 to 17, whereas the size of the reduced neighborhood is simply 3. The substantial increase in runtime of the limiter algorithm can be explained by the following. First, the amount of memory loads required to execute any vertex neighborhood based limiters is at least double that required for the reduced neighborhood. Further, due to size variability of the vertex neighborhood, thread divergence in the GPU code will limit the attainable parallelism at runtime. For a discussion of thread divergence in unstructured CFD codes on GPUs, see [7].

The conclusion to draw from this example is that the quality of the limited solution is a trade-off between computational work and numerical diffusion. For more computational work, one can reduce the amount of numerical diffusion introduced by the limiter by using the vertex neighborhood. The least computationally intensive limiter using the edge neighborhood can excessively smooth the solution. For this problem, we consider the two-point, vertex limiter as the best trade-off between solution quality and run time. Two-point limiting is preferred because the basis functions are already precomputed at these quadrature points. Using two-point limiting is not substantially slower than using one-point limiting, though this may depend on the implementation.



(a) One-point limiting, edge neighborhood.

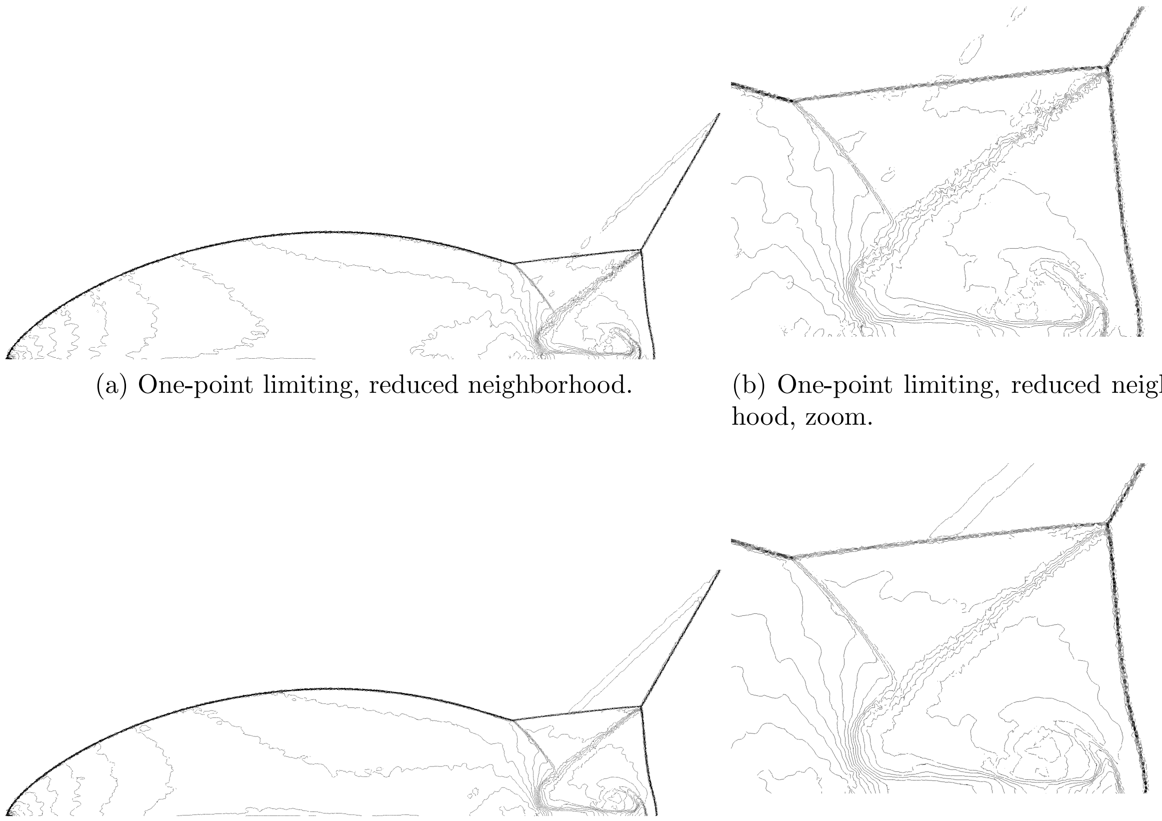
(b) One-point limiting, edge neighborhood, zoom.



(c) Two-point limiting, edge neighborhood.

(d) Two-point limiting, edge neighborhood, zoom.

Figure 3.20: Double Mach reflection problem using the edge neighborhood.

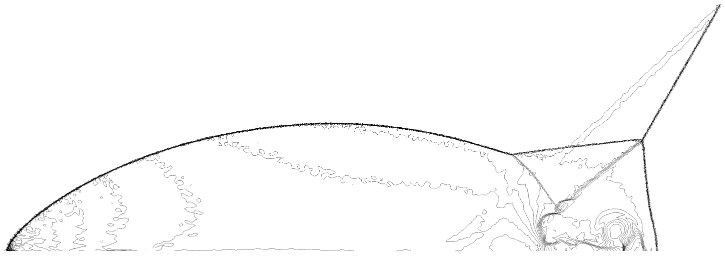

(a) One-point limiting, reduced neighborhood.

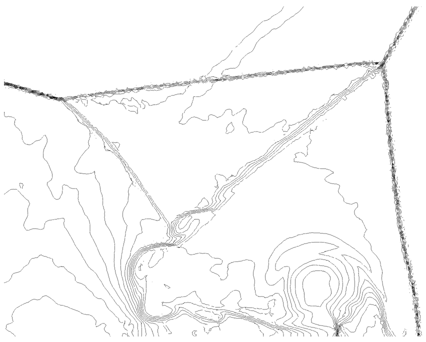
(b) One-point limiting, reduced neighborhood, zoom.

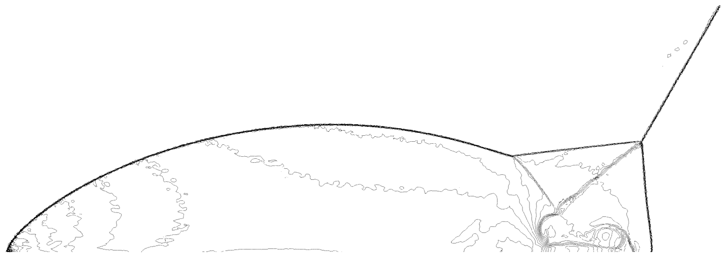
(c) Two-point limiting, reduced neighborhood.

(d) Two-point limiting, reduced neighborhood, zoom.

Figure 3.21: Double Mach reflection problem using the reduced neighborhood.


(a) One-point limiting, vertex neighborhood.


(b) One-point limiting, vertex neighborhood, zoom.


(c) Two-point limiting, vertex neighborhood.

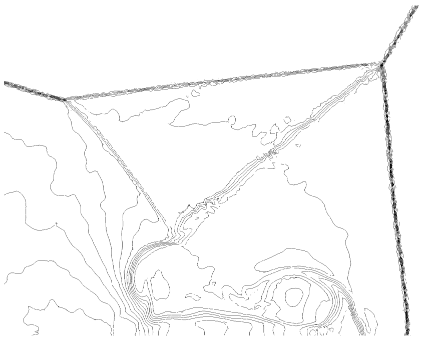

(d) Two-point limiting, vertex neighborhood, zoom.

Figure 3.22: Double Mach reflection problem using the vertex neighborhood.

3.7 Summary

We have studied aspects of second order limiters on unstructured meshes of triangles. These limiters were first introduced and analyzed for finite volume methods but are often applied to DG methods. The important difference between finite volume and DG methods is the manner by which the surface integral is evaluated. FV methods use midpoint quadrature for nonlinear problems, while DG uses two-point quadrature, thus the FV analysis is not directly transferable to the DG method. Since the the surface integral advances the solution means in time, the values at the quadrature points need to be controlled for overshoots if we are to enforce the local maximum principle on the means. Numerical experiments indicate that limiting at the midpoint for nonlinear equations leads to the violation of the maximum principle by a small amount. For example, the solution in Section 3.6.4 grew only by 10^{-6}. While solutions with one-point limiting might look noisy, they do not have unbounded growth, even for long time integration. This indicates that strong stability may be more of theoretical interest rather than of practical purpose, unless an application cannot tolerate any deviation from the initial means.

We find that maintaining accuracy is more difficult than preventing uncontrolled growth of the solution. Our findings indicate that despite ease of implementation and convenience, the edge neighborhood, i.e. elements that share an edge with the element being limited, should not be used for limiting. This is because the limiter is only first order accurate, even on meshes of reasonably good quality, e.g. Delaunay discretization of a square domain. While the edge neighborhood comes naturally as it is the stencil of the DG method, larger neighborhoods, e.g. the vertex neighborhood, should be employed for the solution to stay second order accurate. Unfortunately, this destroys the locality of the DG method, which is one of its advantageous aspects. Although a limiter that uses the DG stencil and preserves locality exists [63], it has its own drawbacks. It requires precomputing and storing geometrical coefficients, which is costly and it needs a user-defined parameter. A possible extension of the present work would be to analyze the admissible range of this parameter and its optimal value.

To summarize, the best limiter is the one based on the vertex neighborhood, though it is almost three times as expensive as one based on smaller neighborhoods. The edge neighborhood provides visibly worse solutions both at shocks and smooth regions. Although the maximum principle can only be enforced for scalar equations, the performance of the limiter on scalar equations is a predictor of performance on systems of equations. Numerical experiments with the Euler equations confirm this.

We show that the local maximum principle is satisfied under a suitable time step restriction. The analysis is valid on one forward Euler time step and the bound is shown to be tight. This restriction involves a new measure of cell size, which is the width of the cell in the direction of flow. This is larger than the commonly used radius of the inscribed circle. A convex combination of forward Euler time steps can extend this stability restriction to high order time integration schemes, e.g. SSP-RK methods. Experimentally, we find that a larger time step can be taken for the SSP-RK2 method without violating the local maximum principle. Finding the analytical CFL number in this case is subject of the following chapter.

Chapter 4

On the optimal CFL number of SSP methods for hyperbolic problems

In one dimension, hyperbolic conservation laws are of the form

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x} f(u) = 0, \tag{4.1}$$

where $u(x,t)$ is the solution and $f(u)$ is the flux function. A popular approach to solving these partial differential equations (PDEs) is the method of lines. This entails first discretizing the spatial derivative, e.g., with the finite volume (FV) method. The result is said to be in a semidiscrete form and is a system of ordinary differential equations (ODEs) for the degrees of freedom (DOFs) of the spatial discretization

$$\frac{d}{dt} U = L(U), \tag{4.2}$$

where U is the numerical solution which approximates u and the operator L approximates $-\frac{\partial}{\partial x} f(u)$. This system is then advanced in time using a time stepping scheme, e.g., an explicit Runge-Kutta (RK) method. Typically, one chooses a time integrator of the same order as the spatial order of accuracy. If $L(U)$ is linear, then stability of the fully discrete scheme under a suitable time step restriction can be shown using the absolute stability region of the time stepper and the eigenvalues of the spatial operator L [71, 72]. This approach cannot be directly applied if $L(U)$ is nonlinear, e.g., if $f(u)$ is nonlinear or if $f(u)$ is linear but a limiter is applied. In this case, we can first show stability of a forward Euler time step applied to (4.2), i.e.,

$$||U + \Delta t L(U)|| \leq ||U||, \quad \forall U, \tag{4.3}$$

where $\Delta t \leq \Delta t_{\mathrm{FE}}$, Δt_{FE} is the maximum stable forward Euler time step specific to the chosen spatial discretization, and $||\cdot||$ is a convex functional [73]. This result can be extended to a higher order RK method if the method can be written as a convex combination of forward Euler time steps. If each forward Euler step does not violate the stability property, then a convex combination of them will not either. This is the idea behind strong stability preserving (SSP) methods [74]. The need for these time discretizations was demonstrated in [75–77]. If the high order time stepper is not SSP, then an oscillation-free numerical solution is not guaranteed even if the spatial reconstruction is total variation diminishing.

The time step Δt of high order SSP methods is related to Δt_{FE} by the SSP coefficient c [77], i.e.,

$$\Delta t \leq c \Delta t_{\mathrm{FE}}. \tag{4.4}$$

The optimal SSP coefficient for the second and third order RK methods, SSP-RK2 and SSP-RK3, is $c = 1$, meaning that forward Euler, RK2, and RK3 time integrators all have the same severe time step restriction. For example, second order finite volume methods with linear slope reconstruction (Section 4.2) coupled with the forward Euler method have a CFL number of $\frac{1}{2}$. Such schemes applied to the scalar advection equation yield a maximum allowed forward Euler time step $\Delta t_{FE} = \frac{1}{2}\frac{h}{a}$, where h is the grid spacing and a is the advection speed. Thus, by the standard SSP theory, this spatial discretization coupled with the SSP-RK2 and SSP-RK3 methods also has a CFL number of $\frac{1}{2}$. In two dimensions, this maximum allowable CFL number becomes $\frac{1}{6}$ for the discontinuous Galerkin (DG) method coupled with the vertex-based limiter in [2], i.e., $\Delta t_{FE} = \frac{1}{6}\frac{h}{||\mathbf{a}||}$, where h is a measure of cell size and $\mathbf{a}$ is the speed vector. This is unlike what is known about linear stability of RK methods, where increasing the number of stages in the RK time stepper can increase the area of its absolute stability region and possibly increase the maximum stable time step.

The advantage of using the standard SSP analysis is that from the stability of the spatial discretization coupled with the forward Euler method, stability with high order SSP time integrators is guaranteed under the suitable time step restriction (4.4). This time step restriction may or may not be tight. We show that by analyzing the fully discrete FV and RK2 schemes for the scalar advection equation with periodic boundary conditions, the SSP coefficient in (4.4) can be increased while still guaranteeing stability of the numerical solution in the maximum norm. We note that our analysis does not preclude the existence of spatial discretizations for which (4.4) is tight.

It has been shown in other contexts that the SSP restriction (4.4) can be relaxed without sacrificing positivity of the solution, in e.g. [78, 79] for well resolved and smooth problems. Additionally, if monotonicity in an inner product norm rather than in a convex functional is desired then a more relaxed time step restriction than (4.4) is possible [80]. The work presented here makes no assumptions on the solution and shows that the maximum stable time step of a second order finite volume scheme can be increased by analyzing the fully discrete spatial and temporal discretization.

4.1 The FV method

We consider a second order finite volume method with slope reconstruction. The periodic computational domain is divided uniformly into elements Ω_i with left, $x_{i-\frac{1}{2}}$, and right, $x_{i+\frac{1}{2}}$, end points, where $h = x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}$ is the grid spacing. A semi-discrete finite volume scheme for (4.1) is given by

$$\frac{d}{dt}U_i = \frac{1}{h}\left[f^*(Q_{i-1}(x_{i-\frac{1}{2}}), Q_i(x_{i-\frac{1}{2}})) - f^*(Q_i(x_{i+\frac{1}{2}}), Q_{i+1}(x_{i+\frac{1}{2}}))\right], \tag{4.5}$$

where U_i is an approximation to the cell average of the exact solution on Ω_i, $Q_i(x)$ is a linearly reconstructed solution on Ω_i, and f^* is the numerical flux [81]. The linearly reconstructed numerical solution at time t^n on cell Ω_i is

$$Q_i^n(x) = U_i^n + \sigma_i^n(x - x_i) \text{ for } x \in [x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}). \tag{4.6}$$

The slope σ_i^n is reconstructed using a second order TVD limiter [32, 82]. Using (4.6), we define the correction term

$$\Delta_i^n = U_{i,r}^n - U_i^n = U_i^n - U_{i,l}^n, \tag{4.7}$$

where $U_{i,r}^n = Q_i^n(x_{i+\frac{1}{2}})$ and $U_{i,l}^n = Q_i^n(x_{i-\frac{1}{2}})$. With a TVD limiter, we have

$$\Delta_{i-1}^n = \gamma_{i-\frac{1}{2},l}(U_i^n - U_{i-1}^n) \text{ and } \Delta_i^n = \gamma_{i-\frac{1}{2},r}(U_i^n - U_{i-1}^n), \tag{4.8}$$

for $0 \leq \gamma_{i-\frac{1}{2},l}, \gamma_{i-\frac{1}{2},r} \leq 1$ (Figure 4.1). Assuming a linear flux $f(u) = au$ where $a > 0$, and the upwind numerical flux in (4.5), we obtain the scheme

$$\frac{d}{dt}U_i = \frac{1}{h}(aU_{i-1,r} - aU_{i,r}). \tag{4.9}$$

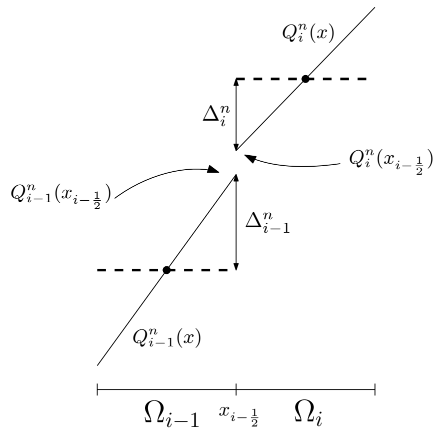


Figure 4.1: Cell averages (dashed lines) and reconstructed slopes (solid lines).

Adding and subtracting aU_i inside the parentheses on the right-hand-side of (4.9), we have

$$\frac{d}{dt}U_i = \frac{a}{h}(U_{i-1,r} - U_i) - \frac{a}{h}(U_{i,r} - U_i). \tag{4.10}$$

Substituting the second identity of (4.7) into (4.10), we obtain

$$\frac{d}{dt}U_i = \frac{a}{h}(U_{i-1,r} - U_i) - \frac{a}{h}(U_i - U_{i,l}). \tag{4.11}$$

4.2 First order forward Euler time stepping

We discretize (4.11) in time using the forward Euler method to obtain a second order in space finite volume scheme, which has been examined in [77, 81, 83]. This gives

$$U_i^{n+1} = \left(1 - 2\Delta t\frac{a}{h}\right)U_i^n + \Delta t\frac{a}{h}U_{i-1,r}^n + \Delta t\frac{a}{h}U_{i,l}^n. \tag{4.12}$$

Letting $\alpha = 2\Delta t\frac{a}{h}$, the cell average at t^{n+1} becomes

$$U_i^{n+1} = (1-\alpha)\,U_i^n + \frac{\alpha}{2}U_{i-1,r}^n + \frac{\alpha}{2}U_{i,l}^n. \tag{4.13}$$

Using periodicity and a reconstruction (4.6) with a TVD slope limiter, $U_{i-1,r}^n, U_{i,l}^n$ will lie within the interval defined by the cell averages of Ω_i and Ω_{i-1} (Figure 4.1). If $\alpha \leq 1$, then

U_i^{n+1} can be expressed as a convex combination of solution values at t^n, and the scheme (4.12) will be stable in the maximum norm. Thus, the forward Euler time step restriction is

$$\Delta t_{FE} \leq \frac{1}{2}\frac{h}{a}.$$

4.3 Second order Runge-Kutta time stepping

The first and second intermediate solution values of the RK2 time stepping algorithm can be written as

$$U_i^{(1)} = (1-\alpha)\,U_i^n + \frac{\alpha}{2}U_{i-1,r}^n + \frac{\alpha}{2}U_{i,l}^n, \tag{4.14}$$

$$U_i^{(2)} = (1-\alpha)\,U_i^{(1)} + \frac{\alpha}{2}U_{i-1,r}^{(1)} + \frac{\alpha}{2}U_{i,l}^{(1)}. \tag{4.15}$$

On each cell, the left $U_{i,l}^{(1)}$ and right $U_{i,r}^{(1)}$ intermediate values can be written in terms of the average $U_i^{(1)}$ and the correction term $\Delta_i^{(1)}$

$$U_{i,l}^{(1)} = U_i^{(1)} - \Delta_i^{(1)} \text{ and } U_{i,r}^{(1)} = U_i^{(1)} + \Delta_i^{(1)}. \tag{4.16}$$

Substituting $U_i^n = (U_{i,l}^n + U_{i,r}^n)/2$ into (4.14), and (4.16) into (4.15), the RK2 algorithm becomes

$$U_i^{(1)} = \frac{1-\alpha}{2}\left(U_{i,l}^n + U_{i,r}^n\right) + \frac{\alpha}{2}U_{i-1,r}^n + \frac{\alpha}{2}U_{i,l}^n, \tag{4.17}$$

$$U_i^{(2)} = (1-\alpha)\,U_i^{(1)} + \frac{\alpha}{2}\left(U_{i-1}^{(1)} + \Delta_{i-1}^{(1)}\right) + \frac{\alpha}{2}\left(U_i^{(1)} - \Delta_i^{(1)}\right), \tag{4.18}$$

$$U_i^{n+1} = \frac{U_i^{(2)} + U_i^n}{2}. \tag{4.19}$$

Substituting (4.17) and (4.18) into (4.19) yields

$$U_i^{n+1} = \left(-\frac{1}{8}\alpha + \frac{1}{2}\right)U_{i,l}^n + \frac{1}{8}\alpha U_{i-1,l}^n + \left(-\frac{1}{4}\alpha^2 + \frac{3}{8}\alpha\right)U_{i-1,r}^n + \frac{1}{8}\alpha^2 U_{i-2,r}^n + \left(\frac{1}{8}\alpha^2 - \frac{3}{8}\alpha + \frac{1}{2}\right)U_{i,r}^n + \frac{1}{4}\alpha(\Delta_{i-1}^{(1)} -$$

U_i^{n+1} is now in terms of the solution values at t^n and the correction terms $\Delta_i^{(1)}$ and $\Delta_{i-1}^{(1)}$. From (4.8), the values of $\Delta_i^{(1)}$ and $\Delta_{i-1}^{(1)}$ take on the following four extreme cases:

1. $\Delta_i^{(1)} = 0$ and $\Delta_{i-1}^{(1)} = 0$,
2. $\Delta_i^{(1)} = U_i^{(1)} - U_{i-1}^{(1)}$ and $\Delta_{i-1}^{(1)} = U_i^{(1)} - U_{i-1}^{(1)}$,
3. $\Delta_i^{(1)} = 0$ and $\Delta_{i-1}^{(1)} = U_i^{(1)} - U_{i-1}^{(1)}$,
4. $\Delta_i^{(1)} = U_i^{(1)} - U_{i-1}^{(1)}$ and $\Delta_{i-1}^{(1)} = 0$.

For each of the above cases, we will show that U_i^{n+1} can be written as a convex combination of solution values at t^n, i.e.,

$$U_i^{n+1} = \sum_j d_j U_j, \tag{4.20}$$

where U_j are understood to be solution averages at time t^n or values at the left and right endpoints of the elements. The multipliers d_j, which are functions of α, must satisfy the following conditions

1. Sum condition
$$\sum_j d_j = 1, \tag{4.21}$$
2. Non-negativity condition
$$d_j \geq 0 \qquad \forall j, \tag{4.22}$$

in order for the scheme to preserve the local and global bounds on the solution. In each of the Cases 1-4, the sum condition (4.21) is satisfied. We will now comment on the values of α for which the multipliers d_j are non-negative.

Cases 1. and 2.

$$U_i^{n+1} = \left(-\frac{1}{8}\alpha + \frac{1}{2}\right) U_{i,l}^n + \frac{1}{8}\alpha U_{i-1,l}^n + \left(-\frac{1}{4}\alpha^2 + \frac{3}{8}\alpha\right) U_{i-1,r}^n + \frac{1}{8}\alpha^2 U_{i-2,r}^n + \left(\frac{1}{8}\alpha^2 - \frac{3}{8}\alpha + \frac{1}{2}\right) U_{i,r}^n.$$

The multipliers are non-negative for $0 \leq \alpha \leq \frac{3}{2}$.

Case 3.

$$U_i^{n+1} = \frac{1}{2} U_{i,l}^n + \frac{1}{4}\alpha U_{i-1,r}^n + \left(-\frac{1}{4}\alpha + \frac{1}{2}\right) U_{i,r}^n.$$

The multipliers are non-negative for $0 \leq \alpha \leq 2$.

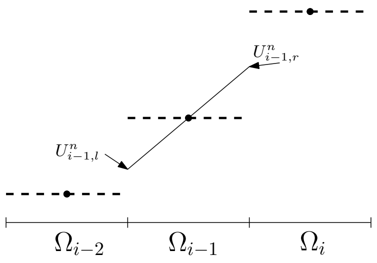


Figure 4.2: The limiter ensures that $U_{i-1,l} - U_{i-1,r} = \beta(U_{i-2} - U_i)$, where $0 \leq \beta \leq 1$.

Case 4.

$$U_i^{n+1} = \frac{1}{2}\left(1-\frac{1}{2}\alpha\right)U^n_{i,l} + \frac{1}{2}\left(1-\alpha+\frac{1}{2}\alpha^2\right)U^n_{i,r} + \frac{1}{4}\alpha U^n_{i-1,l} + \frac{1}{2}\left(\alpha-\alpha^2\right)U^n_{i-1,r} + \frac{1}{4}\alpha^2 U^n_{i-2,r}. \tag{4.23}$$

The multipliers in the above expression are non-negative for $0 \leq \alpha \leq 1$. We can obtain a larger interval for α by rearranging terms. Introducing the difference $\left(U^n_{i-1,l} - U^n_{i-1,r}\right)$ and using $U_i^n = (U^n_{i,l} + U^n_{i,r})/2$, we obtain

$$U_i^{n+1} = \left(1-\frac{1}{2}\alpha\right)U_i^n + \frac{1}{4}\left(\alpha^2-\alpha\right)U^n_{i,r} + \left(-\frac{1}{2}\alpha^2+\frac{3}{4}\alpha\right)U^n_{i-1,l} + \frac{1}{2}\left(\alpha^2-\alpha\right)\left(U^n_{i-1,l} - U^n_{i-1,r}\right) + \frac{1}{4}\alpha^2 U^n_{i-2,r}.$$

With a limiter, the solution satisfies

$$U^n_{i-1,l} - U^n_{i-1,r} = \beta\left(U^n_{i-2} - U^n_i\right)$$

for some $0 \leq \beta \leq 1$ (Figure 4.2). Then, the scheme can be written as

$$U_i^{n+1} = \left(-\frac{1}{2}\beta\alpha^2 + \frac{1}{2}(\beta-1)\alpha + 1\right)U_i^n + \frac{1}{2}\beta\left(\alpha^2-\alpha\right)U^n_{i-2} + \left(\frac{3}{4}\alpha - \frac{1}{2}\alpha^2\right)U^n_{i-1,l} + \frac{1}{4}\alpha^2 U^n_{i-2,r} + \frac{1}{4}(\alpha^2-\alpha)U^n_{i,} \tag{4.24}$$

The multipliers are non-negative for

$$1 \leq \alpha \leq \sqrt{2} = \min_{0<\beta\leq 1}\left(\frac{3}{2}, \frac{\beta - 1 + \sqrt{\beta^2+6\beta+1}}{2\beta}\right).$$

Combining this interval with $0 \leq \alpha \leq 1$, we obtain with this expression for U_i^{n+1} that it can be written as a convex combination of solution values at t^n with $0 \leq \alpha \leq \sqrt{2}$, though we need to use two different expressions.

A slightly larger bound on α can be obtained if we specify which TVD slope limiter is used in the scheme. For example, consider the monotonized central-difference (MC) slope limiter [84]

$$\sigma_i^n = \frac{1}{h}\text{minmod}\left(2(U_i^n - U_{i-1}^n), \frac{U_{i+1}^n - U_{i-1}^n}{2}, 2(U_{i+1}^n - U_i^n)\right). \tag{4.25}$$

With this limiter, we now show that $0 \leq \beta \leq \frac{1}{2}$. First, assume that the forward, central, and backward differences are all of the same sign and nonzero. Then, multiplying both sides of (4.25) by $h/(U_{i+1}^n - U_{i-1}^n)$, recognizing that $U_{i,r}^n - U_{i,l}^n = h\sigma_i^n$, and substituting $\beta = (U_{i,r}^n - U_{i,l}^n)/(U_{i+1}^n - U_{i-1}^n)$, we have

$$\beta = \min\left(2\frac{U_i^n - U_{i-1}^n}{U_{i+1}^n - U_{i-1}^n}, \frac{1}{2}, 2\frac{U_{i+1}^n - U_i^n}{U_{i+1}^n - U_{i-1}^n}\right).$$

From the above, it is clear that β is bounded above by $\frac{1}{2}$. If the forward, central, and backward differences do not have the same sign or at least one is zero, then $\beta = 0$. This gives that $0 \leq \beta \leq \frac{1}{2}$. From this smaller interval for the β coefficient, we have that the multipliers of (4.24) are non-negative for

$$1 \leq \alpha \leq \frac{3}{2} = \min_{0<\beta\leq\frac{1}{2}}\left(\frac{3}{2}, \frac{\beta - 1 + \sqrt{\beta^2 + 6\beta + 1}}{2\beta}\right).$$

Depending on the value of α, we have different expressions of U^{n+1}: if $0 \leq \alpha \leq 1$, then U^{n+1} from (4.23) can be used, if $1 \leq \alpha \leq \frac{3}{2}$, then (4.24) can be used. Overall, U^{n+1} can be written as a convex combination of solution values at t^n for $0 \leq \alpha \leq \frac{3}{2}$.

Putting it all together.

Combining the above with the results from Section 4.2, we find that the scheme satisfies the local maximum principle for $0 \leq \alpha \leq \sqrt{2}$. All above cases are convex combinations of solution values at time t^n, thus a larger time step than in (4.4) is possible. The time step restriction is therefore

$$\Delta t \leq \frac{\sqrt{2}}{2}\frac{h}{a}. \tag{4.26}$$

Taking into account the chosen TVD limiter, which in our case is the MC limiter, this time step restriction can be increased to

$$\Delta t \leq \frac{3}{4}\frac{h}{a}. \tag{4.27}$$

Remark

These results are based on the assumption of a piecewise linear numerical solution and a limiter that forces solution values to belong to a local interval defined by its immediate neighbors. As such, this larger CFL number immediately extends to other spatial discretizations, e.g. the DG method, where we have stability of the solution means in the maximum norm.

4.4 Numerical examples

In this section, we demonstrate that numerical solutions obtained with the time step restriction (4.27) and the MC limiter are accurate and stable. Unless otherwise stated, in all one-dimensional examples we use periodic boundary conditions on the domain $[-1, 1]$ and integrate until the final time $T = 1$.

4.4.1 Advecting sine wave

We solve (4.1) with the flux $f(u) = u$ and the initial condition $u_0(x) = \cos(2\pi x)$. We provide the L_1 errors, convergence rates, and lower and upper bounds attained by the solution means in Table 4.1. We observe that the scheme is second order accurate and preserves the global minimum and maximum of the solution.

4.4.2 Advecting discontinuities

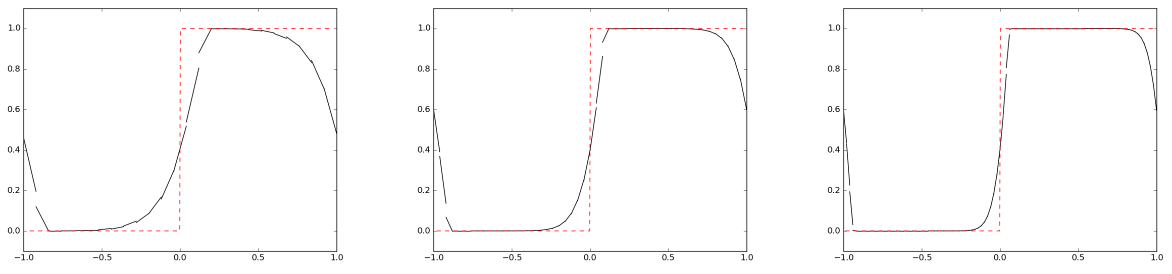
We solve (4.1) with the flux $f(u) = u$ and the initial condition $u_0(x) = 1$ if $x < 0,$ and 0 elsewhere. The exact and numerical solutions at the final time are plotted in Figure 4.3. The global minimum and maximum of the cell averages are maintained. We tabulate them at the final time in Table 4.2.

N	Error	Minimum	Maximum
25	2.814176e-01 (-)	-8.019780e-01	8.042554e-01
50	1.072674e-01 (1.39)	-9.306829e-01	9.283151e-01
100	3.476506e-02 (1.62)	-9.748830e-01	9.748830e-01
200	9.814755e-03 (1.82)	-9.906997e-01	9.906997e-01
400	2.629868e-03 (1.89)	-9.965111e-01	9.965111e-01
800	6.910883e-04 (1.92)	-9.986542e-01	9.986542e-01

Table 4.1: L_1 errors, rates of convergence (in parentheses), and global minimum and maximum of cell averages with the number of cells N for Example 4.4.1.

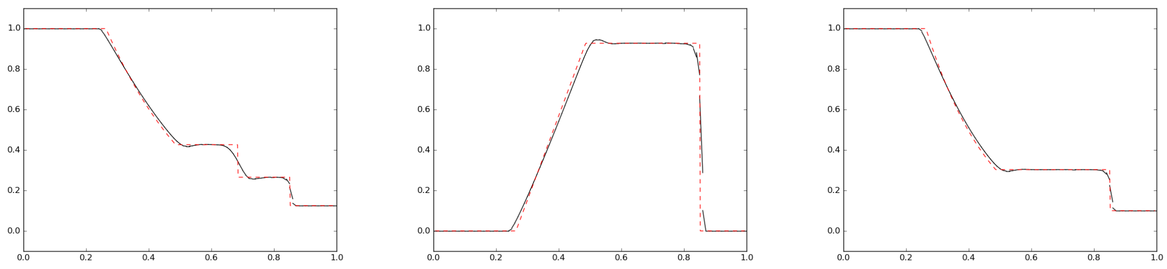
N	Minimum	Maximum
25	1.912839e-04	9.994733e-01
50	8.725729e-09	1
100	0	1
200	0	1
400	0	1
800	0	1

Table 4.2: Global minimum and maximum of cell averages in terms of the number of elements in Example 4.4.2.



(a) $N = 25$ (b) $N = 50$ (c) $N = 100$

Figure 4.3: Exact (dashed line) and numerical (solid line) solutions (Example 4.4.2).



(a) Density (b) Velocity (c) Pressure

Figure 4.4: Exact (dashed) and numerical (solid) density, velocity, and pressure for the Sod tube problem with $N = 100$ (Example 4.4.3).

4.4.3 Euler equations

We solve the Sod tube problem on the domain $[0, 1]$ with the initial states $(\rho_l, u_l, p_l) = (1, 0, 1)$ and $(\rho_r, u_r, p_r) = (0.125, 0, 0.1)$ to the left and right of $x = 0.5$, respectively. The exact and numerical solutions at the final time $T = 0.2$ are plotted in Figure 4.4; slight over- and undershoots are observed in the numerical solution. These oscillations are due to the reconstruction in conserved variables [85] and are present when the CFL number is both $\frac{3}{4}$ and $\frac{1}{2}$. These overshoots and undershoots can occur even in numerical solutions obtained with first order schemes [86].

4.4.4 Two-dimensional advection equation

In this example, we demonstrate that a larger time step is possible in two dimensions, as well as with a spatial discretization different from the FV method. We solve $u_t+u_x+u_y = 0$ on $[-1, 1]^2$ using the DG spatial discretization with a linear basis, coupled with the limiter based on the vertex neighborhood in [2]. The mesh was obtained by discretizing the domain into a 40×40 grid of squares, then splitting each square along its diagonal from the top left to bottom right, into two triangles. It was shown in [2] that solution means do not grow in the maximum norm when

$$\Delta t \le \text{CFL}\frac{h}{||\mathbf{a}||}, \tag{4.28}$$

1/CFL	Minimum	Maximum
3	-9.50e-18	1.000336
4	-6.15e-18	1
5	-3.93e-18	1
6	**-3.62e-18**	**1**

Table 4.3: Minimum and maximum cell averages for Example 4.4.4 using time step restriction (4.28) for various CFL numbers.

with CFL $\leq \frac{1}{6}$, $\mathbf{a}$ being the speed vector, and h being the cell width in the direction of $\mathbf{a}$. Here, $h = \frac{1}{40}\sqrt{2}$. The problem is solved until a final time $T = 0.1$ with the initial condition $u_0(x, y) = 1$ if $\max(|x|, |y|) \leq \frac{1}{4}$, and 0 elsewhere.

In Table 4.3, we show the global maximum and minimum cell averages over the entire mesh for various CFL numbers. Extrapolating from the one-dimensional analysis, we see that the numerical solution in two dimensions preserves the global bounds on the solution for a time step $\Delta t \leq \frac{3}{2}\Delta t_{\text{FE}} = \frac{1}{4}\frac{h}{||\mathbf{a}||}$.

4.5 Conclusion

We have demonstrated analytically and numerically for one-dimensional finite volume methods that the time step restriction for the stability in the maximum norm with RK2 time stepping is larger than the SSP theory's prediction. We provide numerical evidence that this conclusion extends to two dimensions and other spatial discretizations, e.g., the DG method. The main conclusion here is that the stability of the fully discrete numerical method depends on both the temporal and spatial discretizations. We believe that the result can be extended to other SSP methods and spatial discretization schemes. The analysis in multi-dimensions and for time integrators using a larger number of stages will be significantly more involved algebraically.

Chapter 5

Moment limiters in two dimensions

The limiter that we present here can be viewed as a first step in the generalization of the moment limiter in [10, 11] to unstructured meshes, or as a standalone second order limiter with proven stability and accuracy properties. We start by noting that the second order DG solution is written in terms of an orthonormal basis that contains a constant function and two linear functions. We find two directions in which the directional derivative of the solution is proportional to either of the two solution coefficients corresponding to the non-constant basis functions. Each separated coefficient can be limited independently from the other by comparing it to a suitably reconstructed approximation to this directional derivative, as opposed to scaling them both by a constant multiplier [67]. We prove for linear and nonlinear scalar hyperbolic equations that the solution with this limiter satisfies the local maximum principle in the means.

The result of this analysis is a limiter on two-dimensional unstructured meshes that is composed of two independent one-dimensional limiters. The implementation of the limiter is straightforward as it uses the minmod function to compare the solution coefficients to suitable forward and backward differences. The mesh preprocessing stage determines the directional derivatives to be limited and the neighboring elements involved in the reconstruction. The stability analysis provides a set of constraints on the solution coefficients, i.e., a set of inequalities. Finding the optimal limited solution satisfying these constraints will result in the least diffusive limiter, but would be computationally costly. Instead, we derive a simplified region in the space of limiting coefficients that ensures that the numerical solution is second order accurate and that it satisfies the local maximum principle, similar to the Sweby's second order TVD region [32]. From this region, we choose a limiter that is easy to code and compute, as opposed to finding the least diffusive limiter.

We derive a local time step restriction for which application of the limiter guarantees that the cell averages remain within a locally defined interval for one forward Euler time step. This can be extended to high order time stepping, i.e., strong stability preserving (SSP) Runge Kutta methods. The derived CFL number is larger than the one needed for linear stability [87] and we show that using the time step restriction derived from the LMP can lead to stable but inaccurate solutions.

5.1 Ordinary differential equation for the solution averages

Let us consider the case where (1.1) is a scalar conservation law. With $k = 0$, (1.9) becomes

$$\frac{d}{dt}c_{i,0} = -\frac{1}{\det J_i} \sum_{j \in N_i^e, j \neq i} \int_{\partial\Omega_{i,j}} \varphi_0 \mathbf{F}^*(U_i, U_j) \cdot \mathbf{n}_{i,j} \ dl. \tag{5.1}$$

Multiplying the above by $\varphi_0 = \sqrt{2}$ and recognizing that the cell average of U_i is $\overline{U}_i = c_{i,0}\varphi_0$, we obtain

$$\frac{d}{dt}\overline{U}_i = -\frac{1}{|\Omega_i|} \sum_{j \in N_i^e, j \neq i} \int_{\partial\Omega_{i,j}} \mathbf{F}^*(U_i, U_j) \cdot \mathbf{n}_{i,j} \ dl, \tag{5.2}$$

where $|\Omega_i|$ is the area of the cell and $\det J_i = 2|\Omega_i|$. This is an equation for the propagation of the solution average on Ω_i in time. We apply one forward Euler time step to (5.2) to obtain

$$\overline{U}_i^{n+1} = \overline{U}_i^n - \frac{\Delta t}{|\Omega_i|} \sum_{j \in N_i^e, j \neq i} \int_{\partial\Omega_{i,j}} \mathbf{F}^*(U_i^n, U_j^n) \cdot \mathbf{n}_{i,j} \ dl. \tag{5.3}$$

For nonlinear fluxes, the DG method needs to integrate the boundary integral with third order accuracy. An efficient choice is the two-point Gauss-Legendre quadrature rule [62], with $\mathbf{x}_{i,j,q}$ being the qth quadrature point on the edge shared by Ω_i and Ω_j. Replacing the boundary integral in (5.3) with the quadrature rule gives

$$\overline{U}_i^{n+1} = \overline{U}_i^n - \sum_{j \in N_i^e, j \neq i} \frac{1}{2}\Delta t \frac{|\partial\Omega_{i,j}|}{|\Omega_i|} \sum_{q=1,2} \mathbf{F}^*(U_i^n(\mathbf{x}_{i,j,q}), U_j^n(\mathbf{x}_{i,j,q})) \cdot \mathbf{n}_{i,j}. \tag{5.4}$$

For a linear flux, this becomes

$$\overline{U}_i^{n+1} = \overline{U}_i^n - \sum_{j\in N_i^e, j\neq i} \Delta t \frac{|\partial\Omega_{i,j}|}{|\Omega_i|} \mathbf{F}^*(U_i^n(\mathbf{x}_{i,j}), U_j^n(\mathbf{x}_{i,j})) \cdot \mathbf{n}_{i,j}, \tag{5.5}$$

where $\mathbf{x}_{i,j}$ is the midpoint of the edge shared by Ω_i and Ω_j.

5.2 Limiting algorithm

The numerical solution satisfies the local maximum principle in the means if

$$\min_{j\in\mathcal{N}_i} \overline{U}_j^n \leq \overline{U}_i^{n+1} \leq \max_{j\in\mathcal{N}_i} \overline{U}_j^n, \tag{5.6}$$

where $\mathcal{N}_i$ is a set containing the index of Ω_i and the indices of elements neighboring Ω_i, and $\overline{U}_i^n$, $\overline{U}_j^n$ are cell averages. In order to enforce the local maximum principle (5.6), we apply a limiter to the solution coefficients $c_{i,1}^n$ and $c_{i,2}^n$. We consider the directional derivative of $U_i^n(\mathbf{r})$ in the direction of the unit vector $\mathbf{w}$, in the canonical coordinate system $\mathbf{r} = (r, s)$

$$D_{\mathbf{w}} U_i^n(\mathbf{r}) = \nabla_{rs} U_i^n \cdot \mathbf{w} = \left(c_{i,1}^n \nabla_{rs}\varphi_1 + c_{i,2}^n \nabla_{rs}\varphi_2\right) \cdot \mathbf{w}.$$

Computing the gradient of the basis functions φ_1 and φ_2 in (1.7), yields

$$D_{\mathbf{w}} U_i^n(\mathbf{r}) = \left(c_{i,1}^n (6\ ,\ 0) + c_{i,2}^n \left(2\sqrt{3}\ ,\ 4\sqrt{3}\right)\right) \cdot \mathbf{w}. \tag{5.7}$$

In the directions $\mathbf{w}_1 = \frac{2}{\sqrt{5}}\left(1, -\frac{1}{2}\right)$ and $\mathbf{w}_2 = (0, 1)$, the directional derivatives are

$$D_{\mathbf{w}_1} U_i^n = 6\left(\frac{2}{\sqrt{5}}\right) c_{i,1}^n \quad \text{and} \quad D_{\mathbf{w}_2} U_i^n = 4\sqrt{3} c_{i,2}^n.$$

They depend on either $c_{i,1}^n$, or $c_{i,2}^n$, i.e. we have found the directions in which the DOFs are uncoupled. This will allow us to limit each solution coefficient separately by comparing them to forward and backward approximations of the derivatives in the directions $\mathbf{w}_1$ and $\mathbf{w}_2$.

Using (1.3), we map $\mathbf{w}$ into the physical space and normalize to obtain on Ω_i

$$\mathbf{v}_i = \frac{J_i \mathbf{w}}{||J_i \mathbf{w}||}.$$

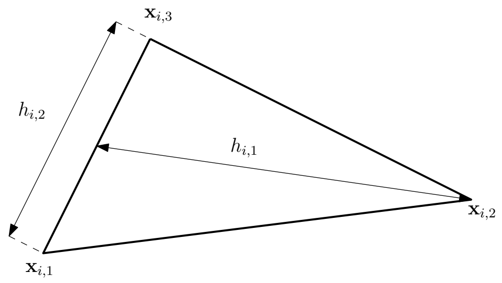


Figure 5.1: $h_{i,1}$ is the length of the segment connecting $\mathbf{x}_{i,2}$ and the midpoint of the edge defined by $\mathbf{x}_{i,1}$ and $\mathbf{x}_{i,3}$. $h_{i,2}$ is the length of the edge defined by $\mathbf{x}_{i,1}$ and $\mathbf{x}_{i,3}$.

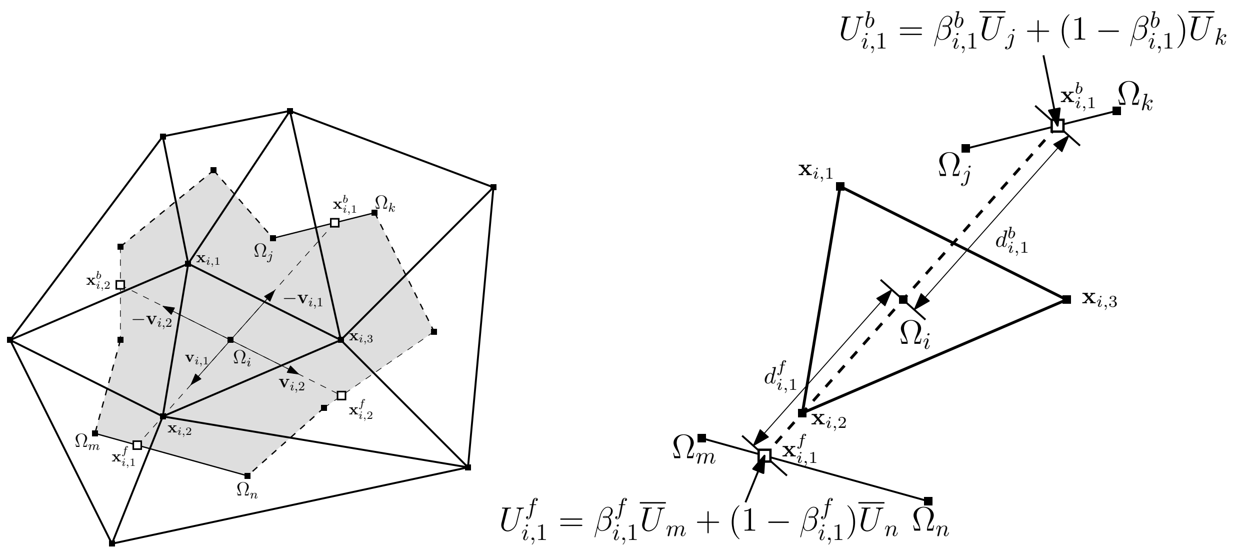
For $\mathbf{w}_1$ and $\mathbf{w}_2$ in canonical coordinates, the corresponding vectors in the physical space are $\mathbf{v}_{i,1}$ and $\mathbf{v}_{i,2}$, and the directional derivatives are

$$D_{\mathbf{v}_{i,1}} U_i^n = c_{i,1}^n \frac{6}{\frac{\sqrt{5}}{2}||J_i \mathbf{w}_1||} \quad \text{and} \quad D_{\mathbf{v}_{i,2}} U_i^n = c_{i,2}^n \frac{4\sqrt{3}}{||J_i \mathbf{w}_2||}.$$

Let $h_{i,1} = \frac{\sqrt{5}}{2}||J_i \mathbf{w}_1||$ and $h_{i,2} = ||J_i \mathbf{w}_2||$. From (1.4), it follows that $h_{i,1}$ is the distance between $\mathbf{x}_{i,2}$ and the midpoint of the opposite edge, and $h_{i,2}$ is the distance between $\mathbf{x}_{i,3}$ and $\mathbf{x}_{i,1}$ (Figure 5.1). In terms of derivatives $D_{\mathbf{v}_{i,1}} U_i^n$ and $D_{\mathbf{v}_{i,2}} U_i^n$, the solution coefficients $c_{i,1}^n$ and $c_{i,2}^n$ are written as

$$c_{i,1}^n = \frac{h_{i,1}}{6} D_{\mathbf{v}_{i,1}} U_i^n \quad \text{and} \quad c_{i,2}^n = \frac{h_{i,2}}{4\sqrt{3}} D_{\mathbf{v}_{i,2}} U_i^n. \tag{5.8}$$

We reconstruct the slopes of the numerical solution in these two directions using solution averages on neighboring elements. We start by compiling a list of all elements that share a vertex with Ω_i. We connect the centroids of the elements with linear segments to form a polygon, see Figure 5.2a, shaded region. We find the four points where this polygon is crossed by the lines with directions $\mathbf{v}_{i,1}$ and $\mathbf{v}_{i,2}$ that pass through the centroid of Ω_i. We name them $\mathbf{x}_{i,1}^b$, $\mathbf{x}_{i,2}^b$, and $\mathbf{x}_{i,1}^f$, $\mathbf{x}_{i,2}^f$, respectively (Figure 5.2a). Next, using linear interpolation, we reconstruct the values of the numerical solution at the forward and backward points of intersection. The reconstructed numerical solution in the forward and backward direction of $\mathbf{v}_{i,1}$ are $U_{i,1}^f$ and $U_{i,1}^b$. Likewise, in the forward and backward direction of $\mathbf{v}_{i,2}$, they are $U_{i,2}^f$ and $U_{i,2}^b$, respectively. For example, in Figure 5.2b the forward interpolated



(a) Reconstruction neighborhood. (b) Forward and backward reconstruction in the direction of $\mathbf{v}_{i,1}$.

Figure 5.2: Approximation of directional derivatives on $\Omega_i = (\mathbf{x}_{i,1}, \mathbf{x}_{i,2}, \mathbf{x}_{i,3})$.

solution value $U^f_{i,1}$ is given by

$$U^f_{i,1} = \beta^f_{i,1}\overline{U}^n_m + (1-\beta^f_{i,1})\overline{U}_n \quad \text{with } 0 \le \beta^f_{i,1} \le 1.$$

The reconstructed forward differences $\Delta^f_{i,1}$ and $\Delta^f_{i,2}$ are defined as

$$\Delta^f_{i,1} = \frac{1}{d^f_{i,1}}(U^f_{i,1} - \overline{U}_i) \text{ and } \Delta^f_{i,2} = \frac{1}{d^f_{i,2}}(U^f_{i,2} - \overline{U}_i), \tag{5.9}$$

where $d^f_{i,1}$, $d^f_{i,2}$ are the distances from $\mathbf{x}^f_{i,1}$ and $\mathbf{x}^f_{i,2}$ to the cell centroid $\mathbf{x}_i$, respectively. Similarly, the backward differences $\Delta^b_{i,1}$ and $\Delta^b_{i,2}$ are

$$\Delta^b_{i,1} = \frac{1}{d^b_{i,1}}(\overline{U}_i - U^b_{i,1}) \text{ and } \Delta^b_{i,2} = \frac{1}{d^b_{i,2}}(\overline{U}_i - U^b_{i,2}), \tag{5.10}$$

where $d^b_{i,1}$, $d^b_{i,2}$ are the distances measured from $\mathbf{x}^b_{i,1}$ and $\mathbf{x}^b_{i,2}$ to the cell centroid $\mathbf{x}_i$, respectively. We limit $c^n_{i,1}$ by comparing $D_{\mathbf{v}_{i,1}}U^n_i$ to the reconstructed forward and backward differences. The same is done for $c^n_{i,2}$ and $D_{\mathbf{v}_{i,2}}U^n_i$. The limited degrees of freedom can be written in terms of the forward or backward differences

$$\tilde{c}_{i,1}^n = l_{i,1}^f \frac{h_{i,1}}{6}\Delta_{i,1}^f \quad \text{or} \quad \tilde{c}_{i,1}^n = l_{i,1}^b \frac{h_{i,1}}{6}\Delta_{i,1}^b,$$
$$\tilde{c}_{i,2}^n = l_{i,2}^f \frac{h_{i,2}}{4\sqrt{3}}\Delta_{i,2}^f \quad \text{or} \quad \tilde{c}_{i,2}^n = l_{i,2}^b \frac{h_{i,2}}{4\sqrt{3}}\Delta_{i,1}^b,$$

where $l_{i,k}^f$ and $l_{i,k}^b$ for $k = 1, 2$ are non-negative limiting coefficients for forward and backward differences that we will derive. Introducing $r_{i,1}$ and $r_{i,2}$, the ratios of the backward and forward differences,

$$r_{i,1} = \frac{\Delta_{i,1}^b}{\Delta_{i,1}^f} = \left(\frac{d_{i,1}^f}{d_{i,1}^b}\right)\frac{\overline{U}_i - U_{i,1}^b}{U_{i,1}^f - \overline{U}_i} \quad r_{i,2} = \frac{\Delta_{i,2}^b}{\Delta_{i,2}^f} = \left(\frac{d_{i,2}^f}{d_{i,2}^b}\right)\frac{\overline{U}_i - U_{i,2}^b}{U_{i,2}^f - \overline{U}_i}, \tag{5.11}$$

we express the limited degrees of freedom in terms of the forward differences

$$\begin{aligned} \tilde{c}_{i,1}^n &= l_{i,1}\frac{h_{i,1}}{6}\Delta_{i,1}^f, \\ \tilde{c}_{i,2}^n &= l_{i,2}\frac{h_{i,2}}{4\sqrt{3}}\Delta_{i,2}^f, \end{aligned} \tag{5.12}$$

with non-negative limiting coefficients $l_{i,1}$ and $l_{i,2}$.

We want to derive upper bounds on the limiting coefficients such that the maximum principle (5.6) is satisfied under a suitable time step restriction. To do so, we write the solution mean at t^{n+1}, $\overline{U}_i^{n+1}$, in the following form

$$\overline{U}_i^{n+1} = \overline{U}_i^n + \sum_j d_j(U_j - \overline{U}_i^n), \tag{5.13}$$

where U_j are understood to be solution means in the neighborhood of Ω_i or reconstructed solution values, e.g. $U_{i,1}^f$, $U_{i,1}^b$, $U_{i,2}^f$, $U_{i,2}^b$. Next, we show that under the limiter (5.12) and a time step restriction, we ensure that the coefficients d_j are non-negative and that the sum of the coefficients is less than or equal to 1. In summary, we need to prove

1. sum property:

$$\sum_j d_j \leq 1, \tag{5.14}$$

2. non-negativity property:

$$d_j \geq 0. \tag{5.15}$$

Properties (5.14) and (5.15) mean that $\overline{U}_i^{n+1}$ in (5.13) is a convex combination of solution values at time t^n and the local maximum principle (5.6) holds for one forward Euler time step. The result can be extended to higher order Runge Kutta methods if they are strong stability preserving (SSP). SSP-RK methods are convex combinations of stages of forward Euler time steps. Since each forward Euler step produces an intermediate solution with cell averages that do not violate those of the initial condition, then a convex combination of forward Euler steps will not either [73].

We present analysis for when (1.1) is a linear equation. The nonlinear case closely follows the linear one and is presented in Appendix B.

5.3 Linear advection equation

We use the upwind numerical flux, which is given by

$$\mathbf{F}^*(U_i^n(\mathbf{x}_{i,j}), U_j^n(\mathbf{x}_{i,j})) \cdot \mathbf{n}_{i,j} = \begin{cases} (\mathbf{a} \cdot \mathbf{n}_{i,j}) U_j^n(\mathbf{x}_{i,j}) \text{ if } j \in N_i^-, \\ (\mathbf{a} \cdot \mathbf{n}_{i,j}) U_i^n(\mathbf{x}_{i,j}) \text{ if } j \in N_i^+, \end{cases}$$

where $\mathbf{a}$ is the flow direction, N_i^- and N_i^+ are the sets of inflow and outflow neighbors that share an edge with Ω_i, respectively, i.e. $N_i^\pm = \{j : j \in N_i^e, j \neq i \text{ such that } \pm \mathbf{a} \cdot \mathbf{n}_{i,j} > 0\}$. For a neighboring element Ω_j with $\mathbf{a} \cdot \mathbf{n}_{i,j} = 0$, the flux term does not contribute to the right hand side of (5.2), and this element can be omitted from both $N_i^\pm$. Therefore, the scheme (5.5) becomes

$$\overline{U}_i^{n+1} = \overline{U}_i^n + \Delta t \sum_{j \in N_i^-} |\mathbf{a} \cdot \mathbf{n}_{i,j}| \frac{|\partial \Omega_{i,j}|}{|\Omega_i|} U_j^n(\mathbf{x}_{i,j}) - \Delta t \sum_{j \in N_i^+} |\mathbf{a} \cdot \mathbf{n}_{i,j}| \frac{|\partial \Omega_{i,j}|}{|\Omega_i|} U_i^n(\mathbf{x}_{i,j}). \tag{5.16}$$

By the divergence theorem, we have the following relation

$$\sum_{j \in N_i^e, j \neq i} |\partial \Omega_{i,j}| \mathbf{a} \cdot \mathbf{n}_{i,j} = 0. \tag{5.17}$$

Using (5.17) in (5.16), we have

$$\overline{U}_i^{n+1} = \overline{U}_i^n + \Delta t \sum_{j\in N_i^-} |\mathbf{a}\cdot\mathbf{n}_{i,j}|\frac{|\partial\Omega_{i,j}|}{|\Omega_i|}(U_j^n(\mathbf{x}_{i,j}) - \overline{U}_i^n) - \Delta t \sum_{j\in N_i^+} |\mathbf{a}\cdot\mathbf{n}_{i,j}|\frac{|\partial\Omega_{i,j}|}{|\Omega_i|}(U_i^n(\mathbf{x}_{i,j}) - \overline{U}_i^n).$$

Introducing the coefficients

$$v_{j,i}^- = -\Delta t\mathbf{a}\cdot\mathbf{n}_{i,j}\frac{|\partial\Omega_{i,j}|}{|\Omega_i|} \quad \text{and} \quad v_{i,j}^+ = \Delta t\mathbf{a}\cdot\mathbf{n}_{i,j}\frac{|\partial\Omega_{i,j}|}{|\Omega_i|}, \tag{5.18}$$

we write the linear scheme as

$$\overline{U}_i^{n+1} = \overline{U}_i^n + \sum_{j\in N_i^-} v_{j,i}^-(U_j^n(\mathbf{x}_{i,j}) - \overline{U}_i^n) - \sum_{j\in N_i^+} v_{i,j}^+(U_i^n(\mathbf{x}_{i,j}) - \overline{U}_i^n). \tag{5.19}$$

Limiting the numerical solution at the time t^n gives

$$\overline{U}_i^{n+1} = \overline{U}_i^n + \sum_{j\in N_i^-} v_{j,i}^-(\tilde{U}_j^n(\mathbf{x}_{i,j}) - \overline{U}_i^n) - \sum_{j\in N_i^+} v_{i,j}^+(\tilde{U}_i^n(\mathbf{x}_{i,j}) - \overline{U}_i^n), \tag{5.20}$$

where $\tilde{U}_i^n$ and $\tilde{U}_j^n$ are the limited numerical solutions on Ω_i and Ω_j, respectively. We aim to rewrite the inflow terms in (5.20) for each $j \in N_i^-$ in the form

$$\tilde{U}_j^n(\mathbf{x}_{i,j}) - \overline{U}_i^n = f_{j,i}(\overline{U}_j^n - \overline{U}_i^n) + f_{j,i,1}(U_{j,i,1}^- - \overline{U}_i^n) + f_{j,i,2}(U_{j,i,2}^- - \overline{U}_i^n), \tag{5.21}$$

where $U_{j,i,1}^-$ and $U_{j,i,2}^-$ are reconstructed solution values in the forward or backward directions $\pm\mathbf{v}_{i,1}$ and $\pm\mathbf{v}_{i,2}$, respectively, and $f_{j,i}$, $f_{j,i,1}$, and $f_{j,i,2}$ are non-negative constants. Likewise, we aim to rewrite the outflow terms for each $j \in N_i^+$ in (5.20) in the form

$$\tilde{U}_i^n(\mathbf{x}_{i,j}) - \overline{U}_i^n = -\left[g_{i,j,1}(U_{i,j,1}^+ - \overline{U}_i^n) + g_{i,j,2}(U_{i,j,2}^+ - \overline{U}_i^n)\right], \tag{5.22}$$

where $U_{j,i,1}^+$ and $U_{j,i,2}^+$ are reconstructed solution values in the forward or backward directions $\pm\mathbf{v}_{i,1}$ and $\pm\mathbf{v}_{i,2}$, respectively, and $g_{i,j,1}$ and $g_{i,j,2}$ are non-negative constants.

5.3.1 Inflow term

We start with the inflow term in (5.20). Consider the limited numerical solution on cell Ω_j, $j \in N_i^-$, an inflow neighbor of Ω_i, at $\mathbf{x}_{i,j} = \mathbf{x}(\mathbf{r}_{j,i})$, where $\mathbf{r}_{j,i}$ is the quadrature point on Ω_0

$$\tilde{U}_j^n(\mathbf{x}(\mathbf{r}_{j,i})) = \overline{U}_j^n + \tilde{c}_{j,1}^n\varphi_1(\mathbf{r}_{j,i}) + \tilde{c}_{j,2}^n\varphi_2(\mathbf{r}_{j,i}).$$

Note that the physical point $\mathbf{x}_{i,j}$ might be mapped using (1.3) to different edges of the canonical triangle Ω_0 by Ω_i and Ω_j. On Ω_j, $\mathbf{x}_{i,j}$ is mapped to $\mathbf{r}_{j,i} \in \Omega_0$, while the same point on Ω_i is mapped to $\mathbf{r}_{i,j} \in \Omega_0$. Using (5.9) and (5.12), the limited solution can be rewritten in terms of the forward differences

$$\tilde{U}_j^n(\mathbf{x}(\mathbf{r}_{j,i})) = \overline{U}_j^n + l_{j,1}\varphi_1(\mathbf{r}_{j,i})\frac{h_{j,1}}{6}\frac{1}{d_{j,1}^f}(U_{j,1}^f - \overline{U}_j^n) + l_{j,2}\varphi_2(\mathbf{r}_{j,i})\frac{h_{j,2}}{4\sqrt{3}}\frac{1}{d_{j,2}^f}(U_{j,2}^f - \overline{U}_j^n). \quad (5.23)$$

Consider the second term in the right hand side of (5.23). To satisfy the non-negativity property (5.15), we require that the multiplier of the difference $U_{j,1}^f - \overline{U}_j^n$ be non-negative. If $\varphi_1(\mathbf{r}_{j,i}) \geq 0$, then this requirement is satisfied. Otherwise, using (5.11), we replace the forward difference with the backward difference to obtain

$$\varphi_1(\mathbf{r}_{j,i})\frac{1}{d_{j,1}^f}(U_{j,1}^f - \overline{U}_j^n) = \varphi_1(\mathbf{r}_{j,i})\frac{1}{r_{j,1}d_{j,1}^b}(\overline{U}_j^n - U_{j,1}^b) = |\varphi_1(\mathbf{r}_{j,i})|\frac{1}{r_{j,1}d_{j,1}^b}(U_{j,1}^b - \overline{U}_j^n).$$

This results in a non-negative multiplier in front of $U_{j,1}^b - \overline{U}_j^n$, if the forward and backward differences are of the same sign. If the differences are of opposite sign, then $l_{j,1}$ is zero and, consequently, $f_{j,i,1}$ is equal to zero. We introduce the following notation for convenience

$$\alpha_{j,i,1}^- = \begin{cases} \frac{h_{j,1}}{6}\frac{\varphi_1(\mathbf{r}_{j,i})}{d_{j,1}^f} & \text{if } \varphi_1(\mathbf{r}_{j,i}) \geq 0, \\ \frac{h_{j,1}}{6}\frac{|\varphi_1(\mathbf{r}_{j,i})|}{r_{j,1}d_{j,1}^b} & \text{otherwise,} \end{cases} \quad \text{and} \quad U_{j,i,1}^- = \begin{cases} U_{j,1}^f & \text{if } \varphi_1(\mathbf{r}_{j,i}) \geq 0, \\ U_{j,1}^b & \text{otherwise.} \end{cases} \quad (5.24)$$

Similarly, for the last term on the right hand side of (5.23) we introduce

$$\alpha_{j,i,2}^- = \begin{cases} \frac{h_{j,2}}{4\sqrt{3}}\frac{\varphi_2(\mathbf{r}_{j,i})}{d_{j,2}^f} & \text{if } \varphi_2(\mathbf{r}_{j,i}) \geq 0, \\ \frac{h_{j,2}}{4\sqrt{3}}\frac{|\varphi_2(\mathbf{r}_{j,i})|}{r_{j,2}d_{j,2}^b} & \text{otherwise,} \end{cases} \quad \text{and} \quad U_{j,i,2}^- = \begin{cases} U_{j,2}^f & \text{if } \varphi_2(\mathbf{r}_{j,i}) \geq 0, \\ U_{j,2}^b & \text{otherwise.} \end{cases} \quad (5.25)$$

Using (5.24) and (5.25) in (5.23), we obtain

$$\tilde{U}_j^n(\mathbf{x}_{i,j}) = \overline{U}_j^n + l_{j,1}\alpha_{j,i,1}^-(U_{j,i,1}^- - \overline{U}_j^n) + l_{j,2}\alpha_{j,i,2}^-(U_{j,i,2}^- - \overline{U}_j^n). \quad (5.26)$$

Subtracting $\overline{U}_i^n$ from both sides of (5.26), then adding and subtracting $\overline{U}_i^n$ in the last two terms on the right, we have

$$\tilde{U}_j^n(\mathbf{x}_{i,j}) - \overline{U}_i^n = \overline{U}_j^n - \overline{U}_i^n + l_{j,1}\alpha_{j,i,1}^-(U_{j,i,1}^- - \overline{U}_i^n + \overline{U}_i^n - \overline{U}_j^n) + l_{j,2}\alpha_{j,i,2}^-(U_{j,i,2}^- - \overline{U}_i^n + \overline{U}_i^n - \overline{U}_j^n),$$

which gives

$$\begin{aligned}\tilde{U}_j^n(\mathbf{x}_{i,j}) - \overline{U}_i^n &= (1 - l_{j,1}\alpha_{j,i,1}^- - l_{j,2}\alpha_{j,i,2}^-)(\overline{U}_j^n - \overline{U}_i^n) \\ &\qquad + l_{j,1}\alpha_{j,i,1}^-(U_{j,i,1}^- - \overline{U}_i^n) + l_{j,2}\alpha_{j,i,2}^-(U_{j,i,2}^- - \overline{U}_i^n). \quad (5.27)\end{aligned}$$

Thus, (5.27) is in the form of (5.21), with

$$\begin{aligned}f_{j,i} &= 1 - l_{j,1}\alpha_{j,i,1}^- - l_{j,2}\alpha_{j,i,2}^-, \\ f_{j,i,1} &= l_{j,1}\alpha_{j,i,1}^-, \\ f_{j,i,2} &= l_{j,2}\alpha_{j,i,2}^-.\end{aligned}$$

Sum and non-negativity

The coefficients $f_{j,i,1}$ and $f_{j,i,2}$ are non-negative by (5.24) and (5.25). Requiring the coefficient $f_{j,i}$ to be non-negative gives the following condition

$$1 - l_{j,1}\alpha_{j,i,1}^- - l_{j,2}\alpha_{j,i,2}^- \geq 0 \quad \forall j \in N_i^-. \quad (5.28)$$

Note that (5.28) imposes a restriction on the neighboring inflow element Ω_j, rather than on Ω_i itself. The sum of the coefficients over the inflow edge is

$$f_{j,i} + f_{j,i,1} + f_{j,i,2} = l_{j,1}\alpha_{j,i,1}^- + l_{j,2}\alpha_{j,i,2}^- + (1 - l_{j,1}\alpha_{j,i,1}^- - l_{j,2}\alpha_{j,i,2}^-) = 1. \quad (5.29)$$

5.3.2 Outflow term

We now deal with the outflow term in (5.20). Consider the limited numerical solution on cell Ω_i at the quadrature point $\mathbf{x}_{i,j} = \mathbf{x}(\mathbf{r}_{i,j})$, $j \in N_i^+$,

$$\tilde{U}_i^n(\mathbf{x}(\mathbf{r}_{i,j})) = \overline{U}_i^n + \tilde{c}_{i,1}^n\varphi_1(\mathbf{r}_{i,j}) + \tilde{c}_{i,2}^n\varphi_2(\mathbf{r}_{i,j}).$$

Using (5.9) and (5.12), the limited solution can be rewritten in terms of the forward differences

$$\tilde{U}_i^n(\mathbf{x}(\mathbf{r}_{i,j})) - \overline{U}_i^n = l_{i,1}\varphi_{i,1}(\mathbf{r}_{i,j})\frac{h_{i,1}}{6}\frac{1}{d_{i,1}^f}(U_{i,1}^f - \overline{U}_i^n) + l_{i,2}\varphi_{i,2}(\mathbf{r}_{i,j})\frac{h_{i,2}}{4\sqrt{3}}\frac{1}{d_{i,2}^f}(U_{i,2}^f - \overline{U}_i^n). \quad (5.30)$$

As for the inflow term, we introduce the following notation for the first term in the right hand side of (5.30)

$$\alpha_{i,1}^+ = \begin{cases} \frac{h_{i,1}}{6}\frac{|\varphi_1(\mathbf{r}_{i,j})|}{d_{i,1}^f} & \text{if } \varphi_1(\mathbf{r}_{i,j}) \le 0, \\ \frac{h_{i,1}}{6}\frac{\varphi_1(\mathbf{r}_{i,j})}{r_{i,1}d_{i,1}^b} & \text{otherwise,} \end{cases} \qquad \text{and } U_{i,j,1}^+ = \begin{cases} U_{i,1}^f & \text{if } \varphi_1(\mathbf{r}_{i,j}) \le 0 \\ U_{i,1}^b & \text{otherwise,} \end{cases} \quad (5.31)$$

and for the second term in the right hand side of (5.30)

$$\alpha_{i,2}^+ = \begin{cases} \frac{h_{i,2}}{4\sqrt{3}}\frac{|\varphi_2(\mathbf{r}_{i,j})|}{d_{i,2}^f} & \text{if } \varphi_2(\mathbf{r}_{i,j}) \le 0 \\ \frac{h_{i,2}}{4\sqrt{3}}\frac{\varphi_2(\mathbf{r}_{i,j})}{r_{i,2}d_{i,2}^b} & \text{otherwise.} \end{cases} \qquad \text{and } \; U_{i,j,2}^+ = \begin{cases} U_{i,2}^f & \text{if } \varphi_2(\mathbf{r}_{i,j}) \le 0 \\ U_{i,2}^b & \text{otherwise.} \end{cases} \quad (5.32)$$

Therefore, (5.30) becomes

$$\tilde{U}_i^n(\mathbf{x}_{i,j}) - \overline{U}_i^n = -\left[l_{i,1}\alpha_{i,j,1}^+(U_{i,j,1}^+ - \overline{U}_i^n) + l_{i,2}\alpha_{i,j,2}^+(U_{i,j,2}^+ - \overline{U}_i^n)\right]. \quad (5.33)$$

This is of the form (5.22) with $g_{i,j,1} = l_{i,1}\alpha_{i,j,1}^+$ and $g_{i,j,2} = l_{i,2}\alpha_{i,j,2}^+$. As in the inflow case, $l_{i,1}$ or $l_{i,2}$ are zero when backward and forward differences are of opposite sign.

Sum and non-negativity

The multipliers $g_{i,j,1}$ and $g_{i,j,2}$ are non-negative by (5.31) and (5.32). The sum of the coefficients is given by

$$g_{i,j,1} + g_{i,j,2} = l_{i,1}\alpha_{i,j,1}^+ + l_{i,2}\alpha_{i,j,2}^+. \quad (5.34)$$

5.3.3 Putting it all together

The inflow and outflow terms have been expanded into sums of the form (5.21) and (5.22) in (5.27) and (5.33), respectively. Substituting these sums into (5.20) gives

$$\overline{U}_i^{n+1} = \overline{U}_i^n + \sum_{j \in N_i^-} v_{j,i}^- \left[f_{j,i}(\overline{U}_j^n - \overline{U}_i^n) + f_{j,i,1}(U_{j,i,1}^- - \overline{U}_i^n) + f_{j,i,2}(U_{j,i,2}^- - \overline{U}_i^n) \right] \\ + \sum_{j \in N_i^+} v_{i,j}^+ \left[g_{i,j,1}(U_{i,j,1}^+ - \overline{U}_i^n) + g_{i,j,2}(U_{i,j,2}^+ - \overline{U}_i^n) \right].$$

This is of the form (5.13). We require that the sum of coefficients in front of the differences above is less than or equal to one. Using (5.29) and (5.34), we write this requirement as

$$\sum_{j \in N_i^-} v_{j,i}^- \left[f_{j,i} + f_{j,i,1} + f_{j,i,2} \right] + \sum_{j \in N_i^+} v_{i,j}^+ \left[g_{i,j,1} + g_{i,j,2} \right] = \\ \sum_{j \in N_i^-} v_{j,i}^- + \sum_{j \in N_i^+} v_{i,j}^+ \left(l_{i,1} \alpha_{i,j,1}^+ + l_{i,2} \alpha_{i,j,2}^+ \right) \leq 1. \tag{5.35}$$

For (5.35) to be satisfied, we enforce on each outflow edge of Ω_i

$$l_{i,1} \alpha_{i,j,1}^+ + l_{i,2} \alpha_{i,j,2}^+ \leq 1, \quad \forall j \in N_i^+, \tag{5.36}$$

and on all elements

$$\sum_{j \in N_i^-} v_{j,i}^- + \sum_{j \in N_i^+} v_{i,j}^+ \leq 1, \quad \forall \Omega_i. \tag{5.37}$$

For non-negativity of the expansion coefficient $f_{j,i}$ we must enforce the constraint (5.28) on the inflows edges of Ω_i, i.e. on the outflow edges of elements sharing an edge with Ω_i,

$$l_{j,1} \alpha_{j,i,1}^- + l_{j,2} \alpha_{j,i,2}^- \leq 1, \quad \forall j \in N_i^-. \tag{5.38}$$

5.3.4 Time step restriction

Using the definition of $v_{j,i}^-$ and $v_{i,j}^+$ in (5.18), (5.37) becomes

$$\Delta t \left(\sum_{j \in N_i^-} |\mathbf{a} \cdot \mathbf{n}_{i,j}| \frac{|\partial\Omega_{i,j}|}{|\Omega_i|} + \sum_{j \in N_i^+} |\mathbf{a} \cdot \mathbf{n}_{i,j}| \frac{|\partial\Omega_{i,j}|}{|\Omega_i|} \right) \leq 1, \tag{5.39}$$

since $\mathbf{a} \cdot \mathbf{n}_{i,j} < 0$ for $j \in N_i^-$ and $\mathbf{a} \cdot \mathbf{n}_{i,j} > 0$ for $j \in N_i^+$. From (5.17), we obtain

$$-\sum_{j \in N_i^-} |\partial\Omega_{i,j}| \mathbf{a} \cdot \mathbf{n}_{i,j} = \sum_{j \in N_i^+} |\partial\Omega_{i,j}| \mathbf{a} \cdot \mathbf{n}_{i,j}.$$

Because $\mathbf{a} \cdot \mathbf{n}_{i,j} < 0$ for $j \in N_i^-$ and $\mathbf{a} \cdot \mathbf{n}_{i,j} > 0$ for $j \in N_i^+$, this becomes

$$\sum_{j \in N_i^-} |\partial\Omega_{i,j}| |\mathbf{a} \cdot \mathbf{n}_{i,j}| = \sum_{j \in N_i^+} |\partial\Omega_{i,j}| |\mathbf{a} \cdot \mathbf{n}_{i,j}|. \tag{5.40}$$

If $\mathbf{a}$ is not parallel to one of the edges, Ω_i has either a single inflow edge or a single outflow edge, which we will refer to as $\partial\Omega_{i,j}$. Otherwise, either the inflow or outflow edge can be chosen to be $\partial\Omega_{i,J}$. In terms of $\partial\Omega_{i,J}$, identity (5.40) now becomes

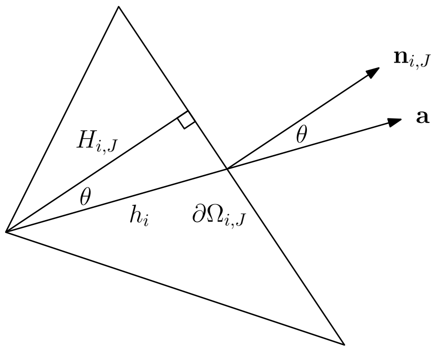
$$|\partial\Omega_{i,J}| |\mathbf{a} \cdot \mathbf{n}_{i,J}| = \sum_{j \in N_i^-} |\partial\Omega_{i,j}| |\mathbf{a} \cdot \mathbf{n}_{i,j}| = \sum_{j \in N_i^+} |\partial\Omega_{i,j}| |\mathbf{a} \cdot \mathbf{n}_{i,j}|.$$

Using the above in (5.39), we obtain

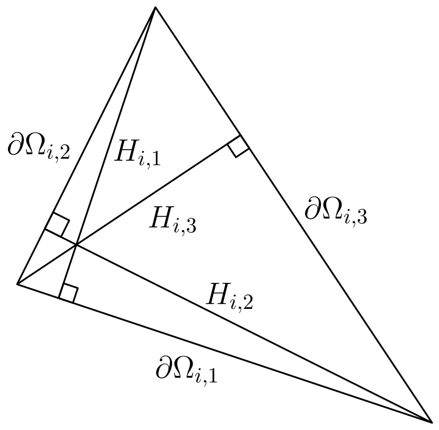
$$2\Delta t \frac{|\partial\Omega_{i,J}| |\mathbf{a} \cdot \mathbf{n}_{i,J}|}{|\Omega_i|} \leq 1. \tag{5.41}$$

The area of the cell Ω_i is $\frac{1}{2}|\partial\Omega_{i,j}|H_{i,J}$, where $H_{i,J}$ is the height of the cell measured from the edge $\partial\Omega_{i,j}$ (Figure 5.3a). Further, a simple geometric consideration reveals that $||\mathbf{a}||H_{i,J} = h_i|\mathbf{a} \cdot \mathbf{n}_J|$, where h_i is the width of the cell in the direction of $\mathbf{a}$. Using this, (5.41) simplifies to

$$2\Delta t \frac{|\partial\Omega_{i,J}| |\mathbf{a} \cdot \mathbf{n}_{i,J}|}{|\Omega_i|} = 4\Delta t \frac{||\mathbf{a}||}{h_i} \leq 1. \tag{5.42}$$



(a) Cell size in the direction of flow h_i.



(b) Cell size for systems $h_i = \min(H_{i,1}, H_{i,2}, H_{i,3})$.

Figure 5.3: Measures of cell size for time step restriction (5.43).

The time step restriction on the entire mesh is then given by

$$\Delta t \leq \frac{1}{4} \min_i \frac{h_i}{||\mathbf{a}||}. \tag{5.43}$$

For systems of equations, in general, information can be propagated along multiple directions. Therefore, we take the smallest cell width, i.e.,

$$h_i = \min(H_{i,1}, H_{i,2}, H_{i,3}), \tag{5.44}$$

where $H_{i,1}$, $H_{i,2}$, and $H_{i,3}$ are the cell widths perpendicular to the three edges of the element, e_1, e_2, and e_3 (Figure 5.3b).

5.3.5 Moment limiter

By the analysis in Sections 5.3.1-5.3.3, there are two constraints on the slope of $U_i^n(\mathbf{x})$ for each outflow edge of Ω_i and no constraints on the inflow edges. One constraint is given by (5.36) and the other is imposed by Ω_i's neighbor via (5.38). Constraints (5.36) and (5.38) involve geometric constants and the values of the basis functions at the outflow edges' midpoints, which are listed in Table 5.1. Consequently, the constraints depend on how the physical triangles are mapped to the canonical triangle. This depends on how the vertices

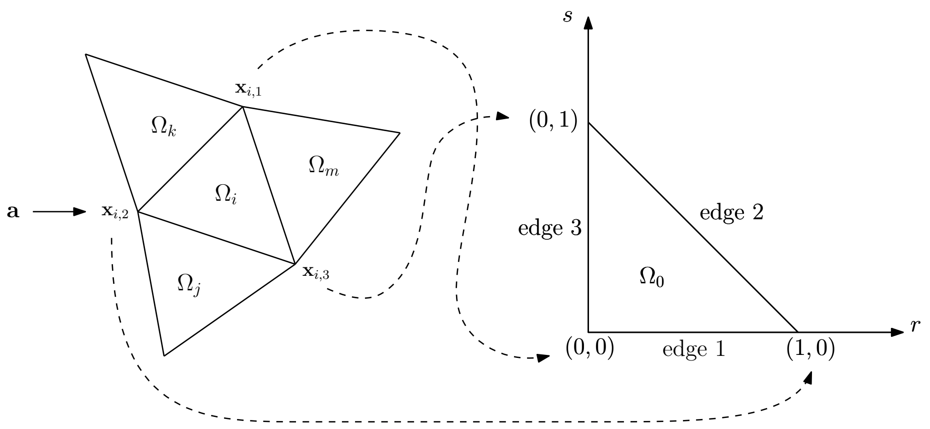


Figure 5.4: Mapping of Ω_i to the canonical triangle Ω_0 by (1.3).

are labeled in the physical triangle. By (1.3), vertex $\mathbf{x}_{i,1}$ is always mapped to $(0,0)$ (Figure 5.4). However, which vertex of the triangle is listed as $\mathbf{x}_{i,1}$ depends on the mesh generator. For the edge number s, if it is an outflow edge of Ω_i, the constraints (5.36) and (5.38) are

$$\frac{l_{i,1}}{6\gamma^f_{i,1}} + \frac{l_{i,2}}{4\gamma^b_{i,2} r_{i,2}} \le 1 \quad \text{and} \quad \frac{l_{i,1}}{6\gamma^b_{i,1} r_{i,1}} + \frac{l_{i,2}}{4\gamma^f_{i,2}} \le 1 \quad \text{if} \quad s = 1, \tag{5.45}$$

$$\frac{l_{i,1}}{6\gamma^f_{i,1}} + \frac{l_{i,2}}{4\gamma^f_{i,2}} \le 1 \quad \text{and} \quad \frac{l_{i,1}}{6\gamma^b_{i,1} r_{i,1}} + \frac{l_{i,2}}{4\gamma^b_{i,1} r_{i,2}} \le 1 \quad \text{if} \quad s = 2, \tag{5.46}$$

$$\frac{l_{i,1}}{3\gamma^b_{i,1} r_{i,1}} \le 1 \quad \text{and} \quad \frac{l_{i,1}}{3\gamma^f_{i,1}} \le 1 \quad \text{if} \quad s = 3, \tag{5.47}$$

where $\gamma^f_{i,k} = \frac{d^f_{i,k}}{h_{i,k}}$, $\gamma^b_{i,k} = \frac{d^b_{i,k}}{h_{i,k}}$, and $r_{i,k}$ are given by (5.11), for $k = 1, 2$.

To illustrate, let us consider an example in Figure 5.4, with the flow direction $\mathbf{a}$. In this case, Ω_i has one outflow edge $\partial\Omega_{i,m}$. This edge is mapped to edge 3 on the canonical triangle and, consequently, the only constraints on the slope of $U^n_i(\mathbf{x})$ are (5.47). Next, consider the opposite flow direction $-\mathbf{a}$. The outflow edges of Ω_i are now $\partial\Omega_{i,k}$ and $\partial\Omega_{i,j}$. These edges are mapped to edges 1 and 2 on the canonical triangle, respectively. Therefore, the constraints on the slope of $U^n_i(\mathbf{x})$ are (5.45) and (5.46).

Thus, we must check each edge for being an outflow edge and determine which edge of the canonical triangle it maps to. This involves extra coding effort and slows computations. Alternatively, we might choose enforcing (5.36) and (5.38) on all three edges of Ω_i, i.e. enforce (5.45) - (5.47). This will result in a more restrictive limiter, however this limiter will be easier to implement. Inequalities (5.45) - (5.47) are satisfied if

$$\begin{cases} \frac{l_{i,1}}{6\min(\gamma^f_{i,1},\gamma^b_{i,1}r_{i,1})} + \frac{l_{i,2}}{4\min(\gamma^f_{i,2},\gamma^b_{i,2}r_{i,2})} & \leq 1, \\ \frac{l_{i,1}}{3\min(\gamma^f_{i,1},\gamma^b_{i,1}r_{i,1})} & \leq 1. \end{cases}$$

Introducing the modified limiting coefficients

$$\tilde{l}_{i,1} = \frac{l_{i,1}}{\min(\gamma^f_{i,1},\gamma^b_{i,1}r_{i,1})} \text{ and } \tilde{l}_{i,2} = \frac{l_{i,2}}{\min(\gamma^f_{i,2},\gamma^b_{i,2}r_{i,2})},$$

the constraints become

$$\begin{cases} \frac{1}{6}\tilde{l}_{i,1} + \frac{1}{4}\tilde{l}_{i,2} & \leq 1, \\ \frac{1}{3}\tilde{l}_{i,1} & \leq 1. \end{cases}$$

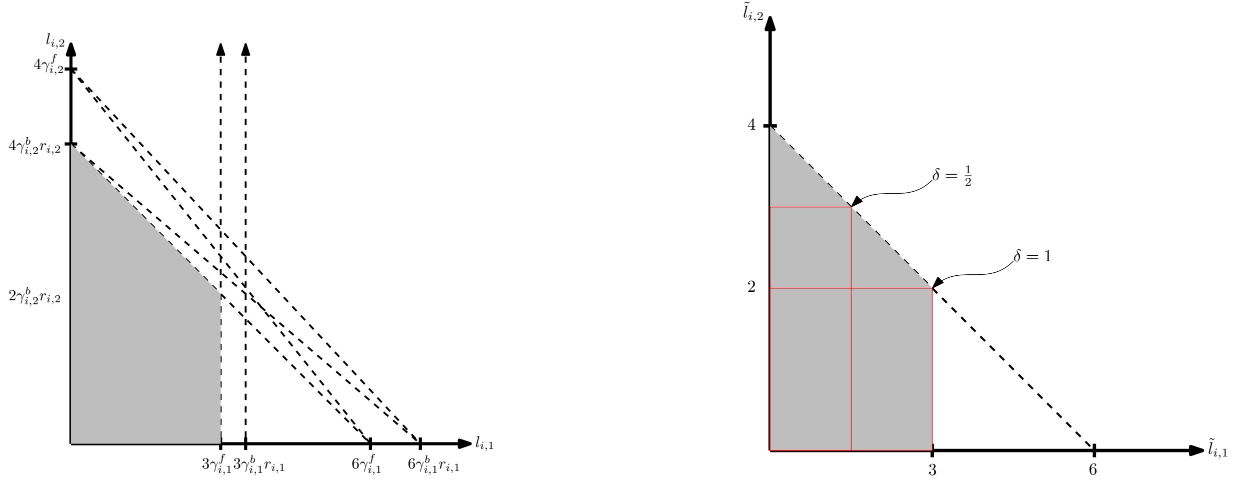
We give an illustration of the inequalities (5.45) - (5.47) in Figure 5.5a. For plotting, we chose a particular relation between the geometric constants γ and ratios r, i.e., $\gamma^f_{i,1} \leq \gamma^b_{i,1}r_{i,1}$ and $\gamma^b_{i,2}r_{i,2} \leq \gamma^f_{i,2}$, though in general this is not always the case. The boundaries of the inequalities are plotted as dashed lines. Values satisfying the inequalities will be to the left and below those lines. The same is done for the simplified set of inequalities in Figure 5.5b. Recall that $l_{i,1}$ or $l_{i,2}$ are non-negative by construction. The case where $l_{i,1}$ or $l_{i,2}$ is equal to zero corresponds to when the forward and backward differences are of opposite sign. This gives the left and bottom boundaries of the stability region. Then, the region from which suitable limiting coefficients can be taken is shown in gray. A simple sufficient condition that satisfies all these constraints is

$$\tilde{l}_{i,1} \leq 3\delta \quad \text{and} \quad \tilde{l}_{i,2} \leq 4 - 2\delta,$$

for any $\delta \in (0,1]$. That is, with $r_{i,1} > 0$ and $r_{i,2} > 0$,

$$l_{i,1} \leq 3\delta\min(\gamma^f_{i,1},\gamma^b_{i,1}r_{i,1}) \quad \text{and} \quad l_{i,2} \leq (4-2\delta)\min(\gamma^f_{i,2},\gamma^b_{i,2}r_{i,2}). \tag{5.48}$$

This is the region bounded by the rectangles in Figure 5.5b. Multiplying the first inequality by $\frac{h_{i,1}}{6}|\Delta^f_{i,1}|$ and the second by $\frac{h_{i,2}}{4\sqrt{3}}|\Delta^f_{i,2}|$, by (5.12) the limited coefficients $\tilde{c}^n_{i,1}$ and $\tilde{c}^n_{i,2}$ must



(a) The full set of inequalities. (b) Simplified set of inequalities.

Figure 5.5: The gray region is the admissibility region which satisfies all constraints. The region bounded by the rectangle is the one used by the limiter for the chosen δ.

s	φ_1	φ_2
1	1	$-\sqrt{3}$
2	1	$\sqrt{3}$
3	-2	0

Table 5.1: Values of the basis functions at the midpoints of the canonical edges with edge number s.

satisfy

$$|\tilde{c}^n_{i,1}| \leq \frac{h_{i,1}}{2}\delta|\Delta^f_{i,1}|\min\left(\frac{d^f_{i,1}}{h_{i,1}}, \frac{d^b_{i,1}}{h_{i,1}}r_{i,1}\right),$$
$$|\tilde{c}^n_{i,2}| \leq \frac{h_{i,2}}{4\sqrt{3}}(4-2\delta)|\Delta^f_{i,2}|\min\left(\frac{d^f_{i,2}}{h_{i,2}}, \frac{d^b_{i,2}}{h_{i,2}}r_{i,2}\right).$$

We choose between the current solution coefficient $c^n_{i,k}$ and the largest one allowed, i.e., the upper bounds in the above inequalities. Simplifying, this becomes

$$\tilde{c}^n_{i,1} = \text{minmod}\left(\delta\frac{U^f_{i,1} - \overline{U}^n_i}{2}, c^n_{i,1}, \delta\frac{\overline{U}^n_i - U^b_{i,1}}{2}\right), \tag{5.49}$$
$$\tilde{c}^n_{i,2} = \text{minmod}\left((4-2\delta)\frac{U^f_{i,2} - \overline{U}^n_i}{4\sqrt{3}}, c^n_{i,2}, (4-2\delta)\frac{\overline{U}^n_i - U^b_{i,2}}{4\sqrt{3}}\right). \tag{5.50}$$

5.3.6 Geometric requirements

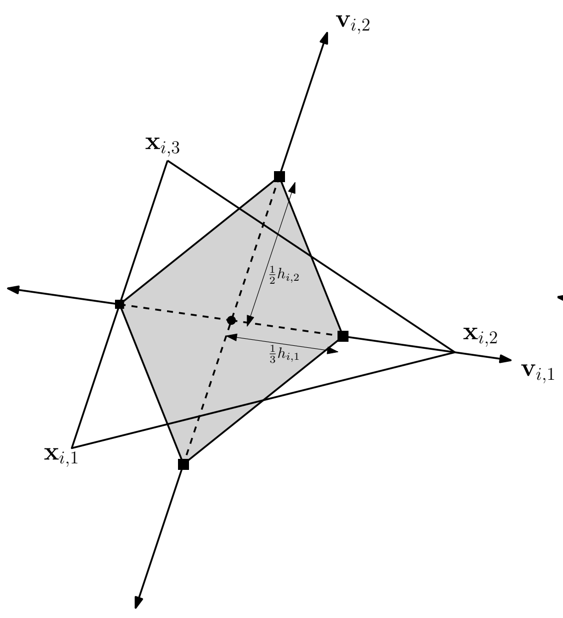
By (5.11), a solution that is linear in x and y will have that $r_{i,1} = r_{i,2} = 1$. In this case, the reconstructed slope must not be reduced for second order spatial accuracy to be preserved, i.e. $1 \leq l_{i,1}, l_{i,2}$. Then, by (5.48) we have

$$1 \leq 3\delta\min(\gamma^f_{i,1}, \gamma^b_{i,1}), \tag{5.51}$$
$$1 \leq (4-2\delta)\min(\gamma^f_{i,2}, \gamma^b_{i,2}). \tag{5.52}$$

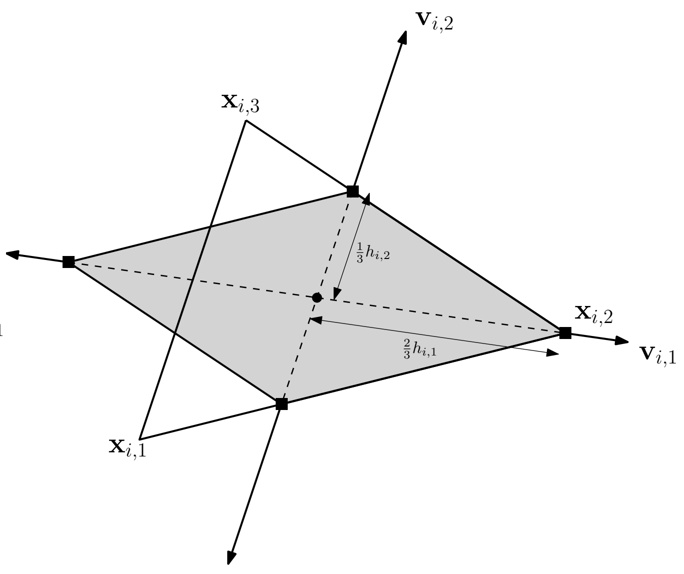
By the definition of the coefficients $\gamma^b_{i,1}$, $\gamma^f_{i,1}$, $\gamma^b_{i,2}$, $\gamma^f_{i,2}$, the above becomes

$$\min(d^f_{i,1}, d^b_{i,1}) \geq \frac{h_{i,1}}{3\delta}, \tag{5.53}$$
$$\min(d^f_{i,2}, d^b_{i,2}) \geq \frac{h_{i,2}}{4-2\delta}. \tag{5.54}$$

This means that the interpolation points must lie a certain minimum distance from the cell centroid that depends on δ (Figure 5.6).



(a) $\delta = 1$



(b) $\delta = \frac{1}{2}$

Figure 5.6: The forward and backward interpolation points $x^f_{i,k}$, $x^b_{i,k}$ must lie a distance from the centroid of the cell in the direction of $\mathbf{v}_{i,k}$ for $k = 1, 2$ greater than the one indicated, i.e. outside the shaded diamond.

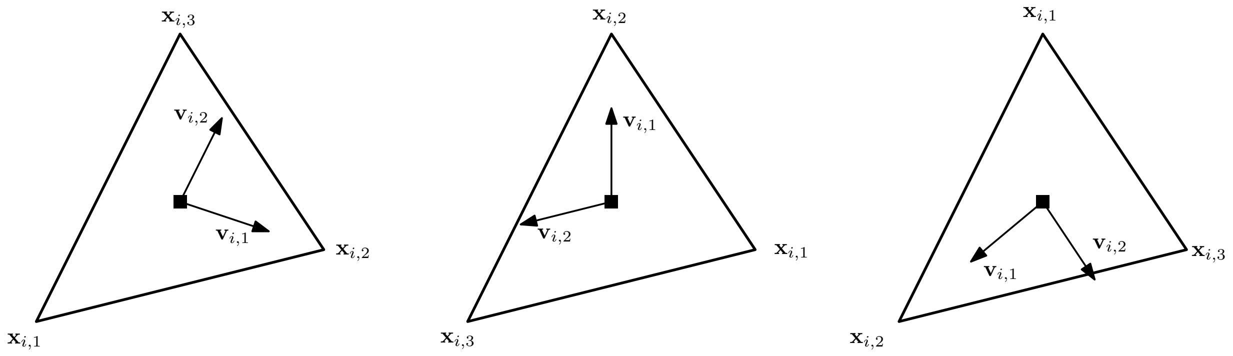


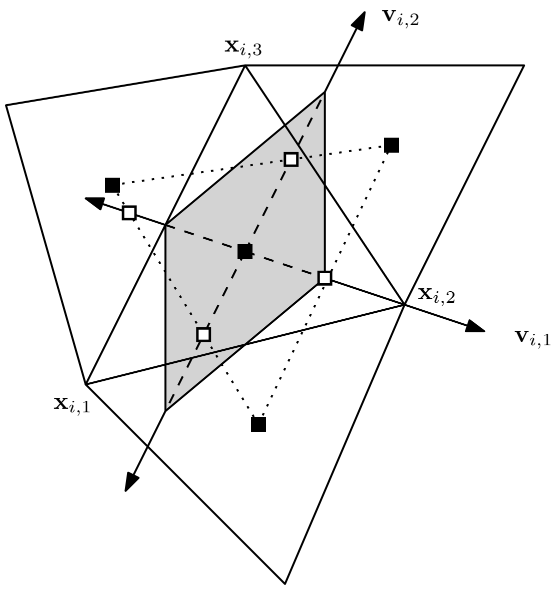
Figure 5.7: Limiting directions of the triangle Ω_i for three vertex numberings.

5.3.7 Preprocessing

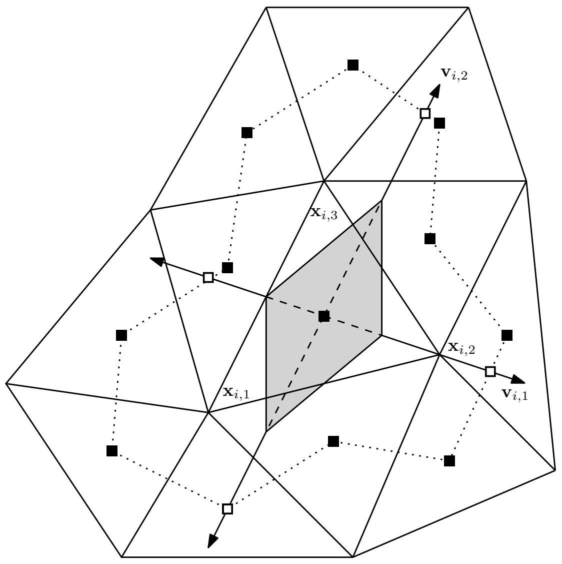
The vertices of the triangle Ω_i, $(\mathbf{x}_{i,1}, \mathbf{x}_{i,2}, \mathbf{x}_{i,3})$, are mapped by (1.3) to the vertices $(0,0)$, $(1,0)$, and $(0,1)$ of the canonical triangle, respectively. In order for the determinant of the Jacobian of this mapping given in (1.4) to be positive, the vertices must be ordered counterclockwise. There are three mappings and, consequently, three pairs of limiting directions, that depend on which vertex of the triangle is listed as $\mathbf{x}_{i,1}$ (Figure 5.7).

The local maximum principle (5.6) restricts the cell average on Ω_i at time t^{n+1} to a locally defined interval that depends on how the reconstruction neighborhood N_i is chosen. It would be computationally advantageous for N_i to consist of elements that share an edge with the cell Ω_i, as this is the stencil for the DG method. Unfortunately, this may be incompatible with the geometrical requirements (5.51) and (5.52) given in Section 5.3.6. This is illustrated in Figure 5.8a where we determine the interpolation points for the limiter (5.49), (5.50) with $\delta = 1$ using the procedure described in Section 5.2. The forward and backwards interpolation points in the direction of $\mathbf{v}_{i,2}$ do not lie far enough away from the centroid of Ω_i because they lie inside the shaded diamond. However, if we enlarge N_i to contain all elements that share a vertex with Ω_i, we obtain interpolation points that satisfy (5.51) and (5.52) (Figure 5.8b). Note that other approaches to find interpolation points are possible, however the present one provides a systematic way to find such points and is simple to code.

These interpolation points are computed during preprocessing of the mesh. We store the pointers to the eight elements involved in limiting and four interpolation coefficients $\beta^f_{i,1}$, $\beta^b_{i,1}$, $\beta^f_{i,2}$, $\beta^b_{i,2}$, (Figure 5.2). The coordinates of the interpolation points are not needed so they are not stored.



(a) Edge neighborhood. The forward and backwards interpolation points in the direction of $\mathbf{v}_{i,2}$ do not satisfy the geometric requirement (5.52).



(b) Vertex neighborhood. All interpolation points satisfy the geometric requirements (5.51) and (5.52).

Figure 5.8: Interpolation points for $\Omega_i = (\mathbf{x}_{i,1}, \mathbf{x}_{i,2}, \mathbf{x}_{i,3})$, denoted by hollow square ticks, with the edge (a) and vertex (b) neighborhoods. Solid squares denote cell centroids. The interpolation points must lie outside the shaded region to allow for second order accuracy.

5.4 Numerical examples

In all examples, we choose $\delta = 1$. The limiter (5.49), (5.50) becomes

$$\tilde{c}^n_{i,1} = \text{minmod}\left(\frac{U^f_{i,1} - \overline{U}^n_i}{2}, c^n_{i,1}, \frac{\overline{U}^n_i - U^b_{i,1}}{2}\right), \tag{5.55}$$

$$\tilde{c}^n_{i,2} = \text{minmod}\left(\frac{U^f_{i,2} - \overline{U}^n_i}{2\sqrt{3}}, c^n_{i,2}, \frac{\overline{U}^n_i - U^b_{i,2}}{2\sqrt{3}}\right). \tag{5.56}$$

Unless otherwise stated, we solve (1.1) on the square domain $[-1, 1]^2$ with RK2 time stepping, and limiter (5.55), (5.56). The moment limiter is implemented in CUDA C and executed on NVIDIA GPUs with the DG implementation [3] using the optimizations described in [7]. This limiting algorithm is very suitable for GPU acceleration due to the stencil of constant size.

5.4.1 Verification of CFL number and global bounds on the solution

In this example, we verify the time step restriction (5.43). We use the flux $\mathbf{F}(u) = [u, u]$, along with the the initial condition of a square pulse

$$u_0(x, y) = \begin{cases} 1 \text{ if } \max(|x|, |y|) \leq \frac{1}{4} \\ 0 \text{ elsewhere.} \end{cases}$$

The problem is solved until the final time $T = 0.1$ on a structured mesh of 11,522 triangles. The mesh is obtained by first tiling the domain with 76 by 76 squares. Then, we split the squares along their diagonals, connecting the square's upper left and lower right corners, to create triangles. The time step restriction is obtained from the wave speed $||\mathbf{a}|| = \sqrt{2}$ and cell width in the direction of flow, which is $\frac{1}{76}\sqrt{2} \approx 1.860 \cdot 10^{-2}$. The minimum and maximum solution cell averages of the final solution are tabulated in Table 5.2 for forward Euler and RK2 time steppers using different values of the CFL number. These results verify that time step restriction (5.43) is tight for the forward Euler method. Additionally, we find that the CFL number can be increased without the the solution averages at the final time exceeding the global bounds of the initial condition. However, the quality of the solution is adversely affected when we exceed the CFL number required for linear stability [87]. We will demonstrate this in the following example.

1/CFL	Minimum	Maximum
2	-3.97e-01	1.14
3	-4.83-02	1.00062
3.5	-4.31e-03	1
4	**-3.39e-19**	**1**

(a) Forward Euler.

1/CFL	Minimum	Maximum
2	-1.19e-19	1
3	-3.31-22	1
3.5	-1.91e-21	1
4	**-4.10e-21**	**1**

(b) RK2.

Table 5.2: Global bounds on the cell averages of the final solution in Example 5.4.1 using time step restriction (5.43).

5.4.2 Verification of accuracy

The CFL number from a linear stability analysis of the scheme (1.9) is $\frac{3}{13} = \frac{1}{4.333...}$ [87]. We note that this is smaller than the one required by the LMP in (5.43). With the flux $\mathbf{F}(u) = [u, 0]$, we solve an advecting pulse problem with the initial condition

$$u_0(x, y) = \begin{cases} \cos^2(2\pi r), & \text{if } r \leq \frac{1}{4}, \\ 0, & \text{otherwise}, \end{cases} \tag{5.57}$$

where $r = \sqrt{(x + \frac{1}{4})^2 + y^2}$. In Table 5.3, we present the global solution bounds and L_1 errors at $T = 0.5$. With a CFL number equal to $\frac{1}{4}$, unlimited solutions are unstable (Table 5.3c). The limiter suppresses the growth of the global bounds of the solution but the rate of convergence of the L_1 error drops to less than 0.6 (compare the L_1 errors in Tables 5.3a and 5.3b). The limiter does not introduce a substantial error on this structured grid (compare the L_1 errors in Tables 5.3b and 5.3d). Thus, in the following examples we use the CFL number given by the linear stability analysis [87].

5.4.3 Limiter (5.55)

The limiter presented in Section 5.3 depends on the numbering of the element vertices, i.e. is mapping dependent. Here we construct an example where all elements in the mesh have only one outflow edge, and that edge is always mapped to the third edge ($k = 3$) of Ω_0 (Figure 5.4). In this case, instead of the general limiter (5.55)-(5.56), we can use only limiter (5.55). Then, only the $c_{i,1}^n$ coefficient is limited and $c_{i,2}^n$ is left untouched.

Number of elements	Minimum	Maximum	L_1 error
200	-1.004307e-37	2.932214e-01	5.1693e-02 (-)
800	-3.317668e-39	6.271256e-01	2.1214e-02 (1.28)
3,200	-1.500449e-62	8.490757e-01	5.5147e-03 (1.94)
12,800	-3.515808e-42	9.444271e-01	1.3530e-03 (2.02)
51,200	**-3.410031e-45**	**9.803368e-01**	**4.5571e-04 (1.56)**
204,800	**-7.484794e-45**	**9.932226e-01**	**3.0248e-04 (0.59)**

(a) Accuracy for 1/CFL = 4 with moment limiter.

Number of elements	Minimum	Maximum	L_1 error
200	-4.482126e-38	2.930021e-01	5.1733e-02 (-)
800	-1.974258e-36	6.264496e-01	2.1039e-02 (1.29)
3,200	-3.639373e-73	8.495680e-01	5.4395e-03 (1.95)
12,800	-1.525488e-60	9.446573e-01	1.3091e-03 (2.05)
51,200	-2.857630e-44	9.804285e-01	3.0646e-04 (2.09)
204,800	-4.616411e-42	9.932461e-01	7.2674e-05 (2.07)

(b) Accuracy for $1/\text{CFL} = \frac{13}{3} = 4.333\ldots$ with moment limiter.

Number of elements	Minimum	Maximum	L_1 error
200	-1.069623e-01	5.838654e-01	4.0877e-02 (-)
800	-4.193137e-02	8.903924e-01	1.5198e-02 (1.42)
3,200	-4.965123e-02	9.683763e-01	1.5860e-02 (-)
12,800	-3.270390e+00	3.202645e+00	7.7678e-01 (-)
51,200	-1.001783e+05	1.069657e+05	1.6659e+04 (-)
204,800	-5.694670e+14	5.489725e+14	6.2178e+13 (-)

(c) Accuracy for 1/CFL = 4, unlimited.

Number of elements	Minimum	Maximum	L_1 error
200	-7.867535e-02	5.795796e-01	3.1425e-02 (-)
800	-5.092045e-02	8.823980e-01	9.9184e-03 (1.66)
3,200	-2.248452e-02	9.678742e-01	2.7487e-03 (1.85)
12,800	-9.359466e-03	9.920149e-01	7.3307e-04 (1.90)
51,200	-3.713583e-03	9.980151e-01	1.9241e-04 (1.92)
204,800	-1.459199e-03	9.995055e-01	4.9797e-05 (1.95)

(d) Accuracy for $1/\text{CFL} = \frac{13}{3} = 4.333\ldots$, unlimited.

Table 5.3: L_1 errors, convergence rates (in parentheses), and global bounds on the final solution for Example 5.4.2.

With the flux $\mathbf{F}(u) = [u, 0]$, we solve an advecting square pulse problem with the initial condition

$$u_0(x, y) = \begin{cases} 1, & \text{if } \max(\left|x + \frac{1}{4}\right|, |y|) \leq \frac{1}{4}, \\ 0, & \text{elsewhere,} \end{cases} \tag{5.58}$$

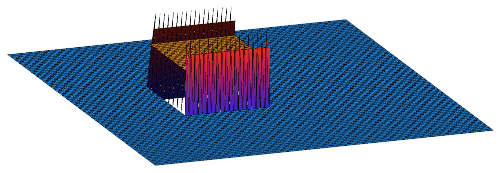
on the same mesh as in Section 5.4.1 until $T = 0.5$. We preprocess the mesh so that the outflow edges of all elements are mapped to the third edge of Ω_0. In both the initial and final solutions (Figure 5.9), we observe oscillations in the y-direction. The amplitude of these oscillations diminishes with time (Figures 5.9a and 5.9b). In the x direction, the solutions are smoother, with discontinuities spread over two cell widths. Although large overshoots are present in the solution, the means are still confined to the initial range $[0, 1]$. This example demonstrates that the local maximum principle (5.6) does not guarantee an oscillation-free solution, even for scalar equations. For comparison, we plot the solution with the full limiter (5.55)-(5.56) in Figures 5.9e, 5.9f, 5.10c, and 5.10d. We observe that the oscillations in the y-direction are suppressed, although the discontinuities are more diffused and there is a slight smearing effect at the corners of the pulse.

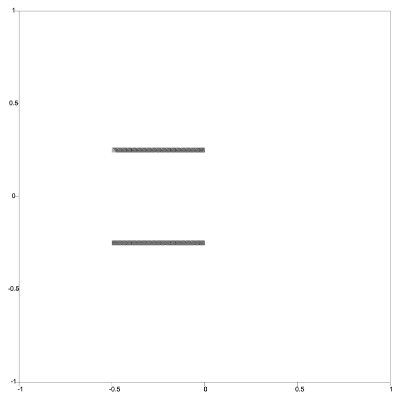
We note however that this is an artificial example as we have a discontinuity that is perpendicular to the edges of the elements, on a mesh of elements with a special mapping to the canonical element. Rotating the initial square pulse by 45 degrees results in a solution with suppressed overshoots, even with limiter (5.55) (Figures 5.10a, 5.10b).

Finally, we solve the problem with a smooth initial condition given by (5.57). The L_1 error at $T = 0.5$ is 1.4982e-03 for the limiter (5.55)-(5.56), and 1.1346e-03 for the limiter (5.55). This demonstrates that the limiter (5.55)-(5.56) does not catastrophically degrade solution quality.

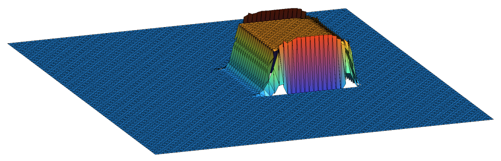
5.4.4 Advecting hill

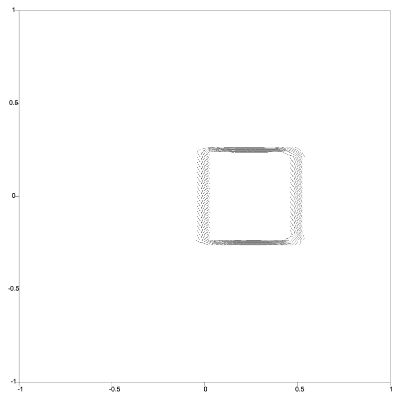
With the flux $\mathbf{F}(u) = [u, u]$, we solve an advecting hill problem with the initial condition (5.57), where $r = \sqrt{(x + \frac{1}{4})^2 + (y + \frac{1}{4})^2}$, on a sequence of unstructured meshes A-D. Refined meshes were obtained by remeshing the domain with an increased target number of elements. The final solutions at $T = 0.5$ on meshes A and D with the moment limiter are plotted in Figure 5.11. The L_1 errors on meshes A-D are reported in Table 5.4. As expected, the limiter reduces solution accuracy. Nevertheless, we observe the second order rate of convergence in the L_1 norm and linear convergence in the L_∞ norm in approximation of the local maximum (Figure 5.11c). For the resolved solutions, the moment limiter


(a) The initial condition using limiter (5.55).

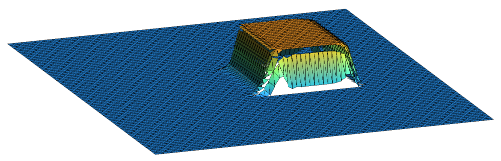


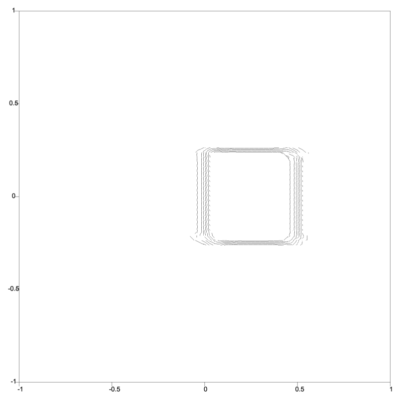
(b) Isolines of the initial condition using limiter (5.55).


(c) The final solution at $T = 0.5$ using (5.55).



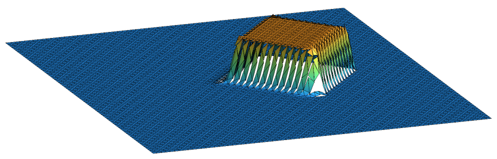
(d) Isolines of the final solution at $T = 0.5$ using limiter (5.55).


(e) The final solution at $T = 0.5$ using limiter (5.55)-(5.56).

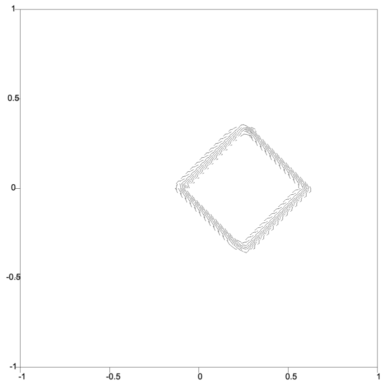


(f) Isolines of the final solution at $T = 0.5$ using limiter (5.55)-(5.56).

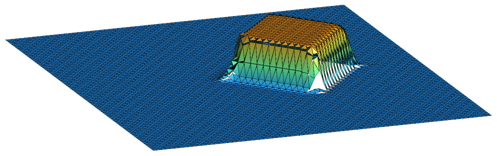
Figure 5.9: Advecting square pulse (5.58) in Example 5.4.3.



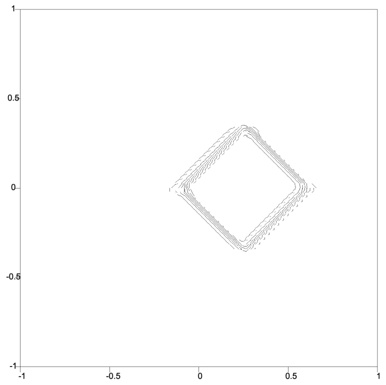
(a) The final solution at $T = 0.5$ using limiter (5.55).



(b) The final solution at $T = 0.5$ using limiter (5.55).



(c) The final solution at $T = 0.5$ using limiter (5.55)-(5.56).



(d) Isolines of the final solution at $T = 0.5$ using limiter (5.55)-(5.56).

Figure 5.10: Advecting rotated square pulse in Example 5.4.3.

Mesh	Elements	Moment	Unlimited
A	1,250	1.9615e-02 (-)	8.3553e-03 (-)
B	5,190	3.7825e-03 (2.37)	1.6512e-03 (2.33)
C	20,552	7.4542e-04 (2.34)	3.7238e-04 (2.14)
D	81,878	1.4196e-04 (2.39)	8.5916e-05 (2.11)

Table 5.4: L_1 errors and convergence rates (in parentheses) for the advecting pulse problem (5.57) in Example 5.4.4.

Mesh	Minimized γ	Mesh generator γ	Maximized γ
A	1.9042e-02	1.9615e-02	1.9175e-02
B	3.7692e-03	3.7825e-03	3.9160e-03
C	7.3582e-04	7.4542e-04	7.6800e-04
D	1.3915e-04	1.4196e-04	1.4369e-04

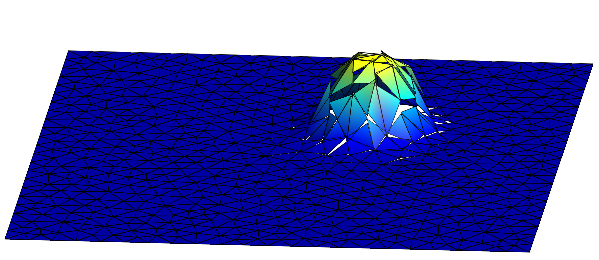
Table 5.5: L_1 errors for different choices of parameters γ in Example 5.4.4.

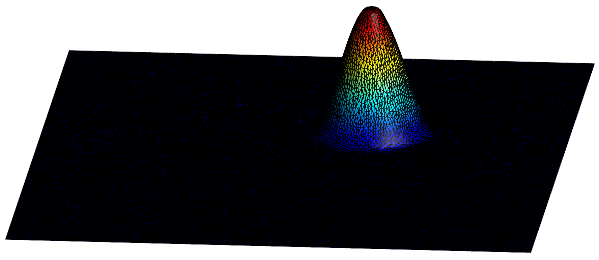
increases the error by a factor of approximately two (the solution on mesh A is badly resolved due to an insufficient number of elements).

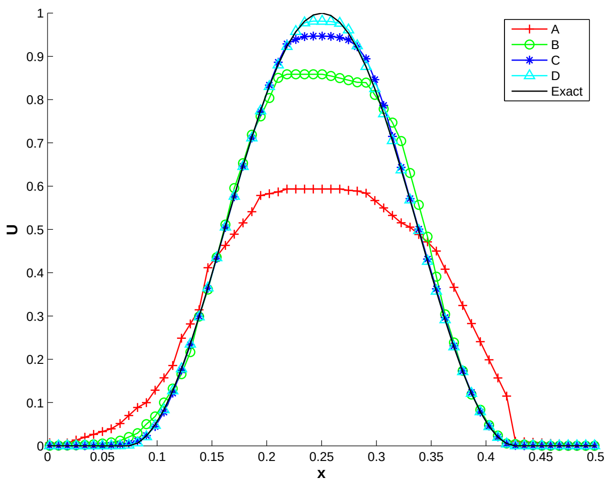
Next, we study how the limiting directions $\mathbf{v}_{i,1}$ and $\mathbf{v}_{i,2}$ (Figure 5.7) influence the performance of the limiter (5.55)-(5.56). Three pairs of $\mathbf{v}_{i,1}$, $\mathbf{v}_{i,2}$ are possible on each Ω_i. This results in three possible limiting stencils and sets of parameter γ. For each pair of limiting directions in Figure 5.7, we computed the minimum and maximum γ out of the four involved in the limiting process, named γ_{min} and γ_{max}, respectively. By maximizing γ, we mean that the configuration with the largest γ_{min} was chosen. Conversely, by minimizing γ, we mean that the configuration with the smallest γ_{max} was chosen. The errors in solutions obtained with different limiting directions are given in Table 5.5. It appears that choosing reconstruction points closest to the cell centroid is beneficial, but not substantially so. Thus, the vertex ordering given by the mesh generator can be used directly, without additional preprocessing of the mesh.

5.4.5 Rotating shapes

With the flux $\mathbf{F}(u) = [-2\pi yu, 2\pi xu]$, we solve a rotating shapes problem. The exact solution is a rotation of the initial data about the origin. The initial condition comprises


(a) Raised solution A.


(b) Raised solution D.



(c) Profile of solution along the line $y = x$.

Figure 5.11: Advecting hill (5.57) in Example 5.4.4 at $T = 0.5$ with the moment limiter.

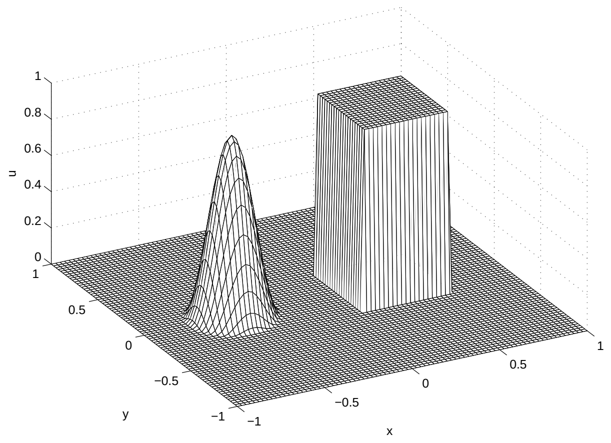


Figure 5.12: Initial condition in Example 5.4.5.

a hill and a square pulse (Figure 5.12) defined by

$$u_0(x,y) = \begin{cases} \cos^2(2\pi r) \text{ if } r \leq 0.25, \\ 1 \text{ if } \max(|x-0.35|,|y|) \leq 0.25, \\ 0 \text{ elsewhere,} \end{cases}$$

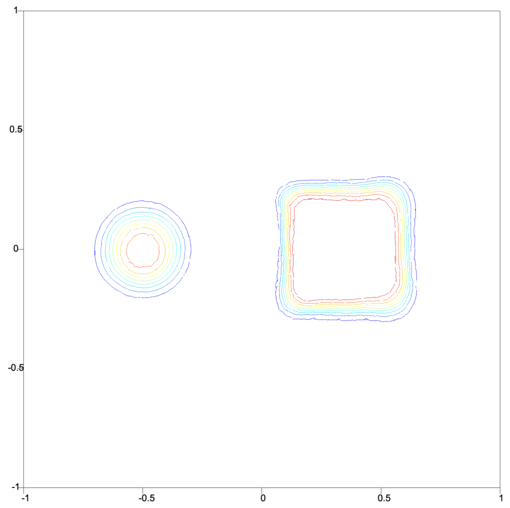
where $r = \sqrt{(x+0.5)^2+y^2}$. This problem is solved on an unstructured mesh of 12,792 triangles using the minimum cell width (5.44) as the measure of cell size. The isolines and profile along $y=0$ of the final solution are displayed in Figure 5.13. This solution is of comparable quality to the moment limiter described in [11], solved on a structured mesh of $80 \times 80 = 6{,}400$ quadrilaterals.

5.4.6 Double Mach reflection problem

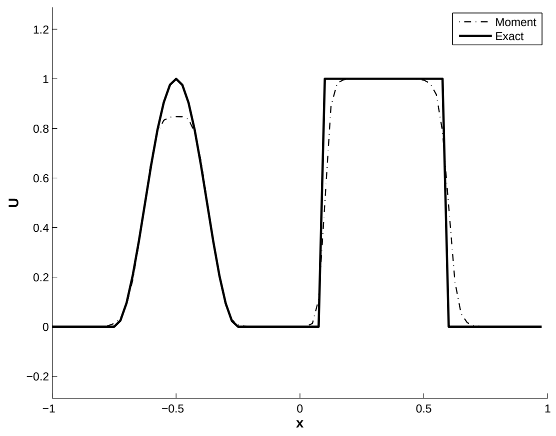
We solve the double Mach reflection problem on an unstructured mesh of 271,458 triangles. The set-up is described in Chapter 5. We extend the limiter to systems of equations by limiting each component separately, i.e., we limit the conserved variables.

We observe that the shocks are well resolved, and the slipline emanating from the primary triple point is tight. These are results are again comparable to those obtained with a moment limiter on Cartesian grids [11].

We compare the GPU runtime performance of the moment limiter to the so-called Barth-Jespersen limiter. The Barth-Jespersen limiter uses either the edge neighborhood or



(a) Isolines with moment limiter.



(b) Profile of solution on the line $y = 0$.

Figure 5.13: Rotating shapes at $T = 1$.

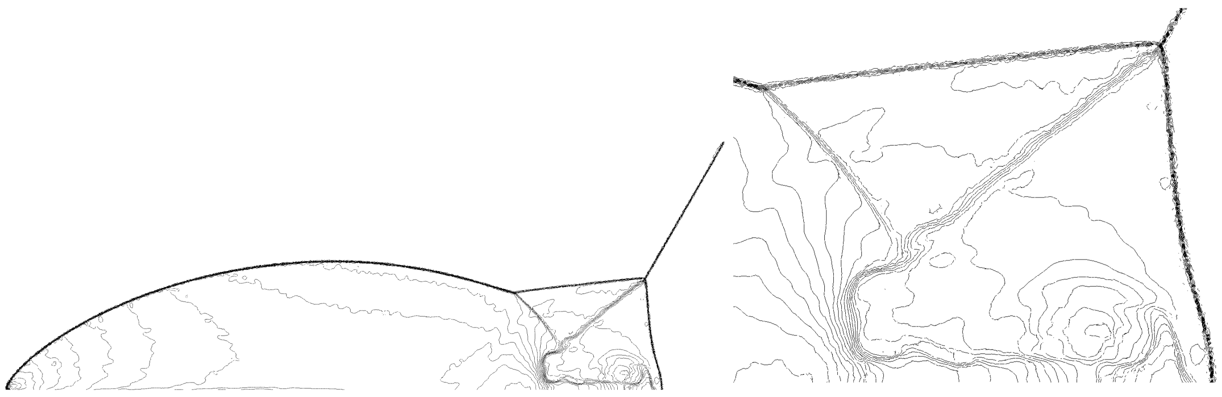
vertex neighborhood to determine a local interval by which to bound the numerical solution at the edge midpoints. It was shown in [2] that the limiter using the edge neighborhood is first order accurate but fast and the limiter using the vertex neighborhood is second order accurate but slow. The total time spent executing the three limiting algorithms on an NVIDIA Titan X Pascal is provided in Table 5.7. We observe that the moment limiter is the fastest of the three limiters and takes approximately 8.6 percent of the total DG-GPU solver run time. The Barth-Jespersen limiter using the edge neighborhood is slightly slower than the moment limiter, but it is only first order accurate. The Barth-Jespersen limiter using the vertex neighborhood is second order accurate, but it executes three times slower per time step than the moment limiter and takes 22 percent of the total solver run time, which is non-negligible. We also note that both Barth-Jespersen limiters have a more restrictive CFL number of $\frac{1}{6}$ in comparison to the moment limiter's CFL number of $\frac{3}{13}$, which explains the increased number of time steps for the Barth-Jespersen limiters in Table 5.7.

	ρ	s	p
$\mathbf{U}_l$	8	8.25	116.5
$\mathbf{U}_r$	1.4	0	1

Table 5.6: Density, normal speed, and pressure to the left and right of the shock in Example 5.4.6.

Limiter	Total solver run time (s)	Limiter run time (s)	Time steps	Time (ms)/ step
Moment	159	13.8 (8.6 %)	6,941	1.9 (-)
Barth-Jespersen (Edge)	191	18.2 (9.5 %)	8,305	2.1 (1.1x)
Barth-Jespersen (Vertex)	231	51.4 (22 %)	8,624	5.9 (3.1x)

Table 5.7: Run time comparisons for moment and Barth-Jespersen limiters. The number in brackets in the 'Limiter run time (s)' column is the percentage of the total solver run time that the limiting algorithm takes to execute. The number in brackets in the 'Time (ms)/ step' column is the speed up factor of the moment limiter compared to the Barth-Jespersen limiters per time step.


(a) Isovalues of the density.

(b) Zoom of the isovalues in neighborhood of the slipline.

Figure 5.14: Double Mach reflection problem.

5.5 Summary

We have proposed a new second order limiter that operates directly on solution coefficients by finding suitable directions in which the one-dimensional minmod operation can be employed. This is in contrast to existing limiters that sample solution values on the edges of triangles and require that they remain in some locally defined range. We derive a family of possible limiters and pick one that is easy to implement. We show on a number of numerical examples that the limiter produces second order accurate, stable numerical solutions, and performs comparably to a moment limiter on Cartesian grids.

This limiter is appealing because it has a stencil of constant size, it is easy to implement, much of its overhead is moved to the preprocessing stage, e.g., determination of the limiter stencil and interpolation coefficients, and it does not require computing the solution values at quadrature points.

Analysis for nonlinear fluxes is provided in the Appendix, though the resulting stability constraints are more involved. Traditionally, limiters developed for linear problems are applied to nonlinear systems. Thus, we do the same here.

Extending this limiter to nonconforming meshes for use in adaptive simulations appears to be possible. An element's limiting directions will stay the same when its neighbors are refined or coarsened, though the elements used in the reconstruction of forward and backward differences may have to be updated. This will not require projecting the neighboring solutions onto coarser or finer elements as is sometimes done. Another possibility for future work is the extension of this limiter to higher order bases, though the analysis is more involved as there are a higher number of solution coefficients that must be limited. An extension to three-dimensional computations would also be of practical interest. The advantages of a stencil of constant size become all the more important as the number of neighbors will grow in three dimensions. Finally, we plan on investigating the applicability of this limiter to computations involved in cut cell element geometries [36, 88].

Chapter 6

Moment limiters in three dimensions

The limiter that we present in this chapter is an extension of the limiter described in Chapter 5. The extension from triangles to tetrahedra is not straightforward as defining the forward and backward differences is more involved, as is the derivation of the time step restriction. An additional difficulty common to all limiters is how to appropriately modify the slope on elements adjacent reflecting boundaries of the domain. One possible approach that simplifies the implementation of the limiter is to not modify the slopes on those elements at all [36]. We propose an alternative, straightforward approach to limiting across reflecting boundaries that does not significantly modify the implementation of the limiter in the code, with the majority of modifications to the code occurring in the preprocessor. Reflecting boundaries model the influence of solid walls in the flow, i.e. the flow normal the wall is zero. For planar boundaries, the effect of the solid wall can be modeled by reflecting the entire computational domain about the wall and solving the enlarged symmetrical problem. Instead of doubling the amount of computational work, we propose to locally reflect elements in the neighborhood of the boundary about the plane of symmetry, which is a simple and effective solution to the problem.

Similar to the analysis for two-dimensional conservation laws in Chapter 5, we assume a scalar conservation law with linear flux $\mathbf{F}(u) = u\mathbf{a}$, where $\mathbf{a}$ is the flow direction. Using the upwind numerical flux, the scheme for the solution averages can be written as

$$\overline{U}_i^{n+1} = \overline{U}_i^n + \sum_{j\in N_i^-} v_{j,i}^-(U_j^n(\mathbf{x}_{i,j}) - \overline{U}_i^n) - \sum_{j\in N_i^+} v_{i,j}^+(U_i^n(\mathbf{x}_{i,j}) - \overline{U}_i^n), \tag{6.1}$$

where $\mathbf{a}$ is the flow direction, N_i^- and N_i^+ are the sets of inflow and outflow neighbors that share an edge with Ω_i, respectively, i.e. $N_i^{\pm} = \{j : j \in N_i^e, j \neq i \text{ such that } \pm\mathbf{a}\cdot\mathbf{n}_{i,j} > 0\}$, and

$$v_{j,i}^{-} = -\Delta t \mathbf{a} \cdot \mathbf{n}_{i,j} \frac{|\partial\Omega_{i,j}|}{|\Omega_i|} \quad \text{and} \quad v_{i,j}^{+} = \Delta t \mathbf{a} \cdot \mathbf{n}_{i,j} \frac{|\partial\Omega_{i,j}|}{|\Omega_i|}. \tag{6.2}$$

Limiting the numerical solution at the time t^n gives

$$\overline{U}_i^{n+1} = \overline{U}_i^n + \sum_{j\in N_i^-} v_{j,i}^{-}(\tilde{U}_j^n(\mathbf{x}_{i,j}) - \overline{U}_i^n) - \sum_{j\in N_i^+} v_{i,j}^{+}(\tilde{U}_i^n(\mathbf{x}_{i,j}) - \overline{U}_i^n), \tag{6.3}$$

where $\tilde{U}_i^n$ and $\tilde{U}_j^n$ are the limited numerical solutions on Ω_i and Ω_j, respectively.

6.1 The limiting algorithm

Our goal is to limit the moments of the DG solution corresponding to the three linear orthonormal basis functions such that we guarantee that the solution average at time t^{n+1} satisfies the local maximum principle (LMP)

$$\min_{j\in\mathcal{N}_i} \overline{U}_j^n \leq \overline{U}_i^{n+1} \leq \max_{j\in\mathcal{N}_i} \overline{U}_j^n, \tag{6.4}$$

without affecting the second order convergence rate of the numerical method. $\mathcal{N}_i$ is a set of indices of elements in some neighborhood of Ω_i along with the index i itself. A scheme for the solution averages of the form

$$\overline{U}_i^{n+1} = \overline{U}_i^n + \sum_j d_j(U_j - \overline{U}_i^n), \tag{6.5}$$

will satisfy the LMP (6.4) if the following two conditions on the d_j are satisfied

1. sum property:
$$\sum_j d_j \leq 1, \tag{6.6}$$

2. non-negativity property:
$$d_j \geq 0, \tag{6.7}$$

where the U_j are understood to be reconstructed solution values or neighboring solution means in a locally defined interval and d_j are scalar multipliers. Therefore, we attempt to write the DG scheme for the solution averages (6.3) in the form (6.5) and derive using the sum and non-negativity properties (6.6) and (6.7) the conditions under which the LMP (6.4) is satisfied.

The first step is to relate each solution moment to a directional derivative. Differentiating the solution on element i in a direction $\mathbf{w}$ gives

$$D_{\mathbf{w}} U_i^n(\mathbf{r}) = c_{i,1}^n \left(4\sqrt{10}\ ,\ 0,\ 0\right)\cdot\mathbf{w} + c_{i,2}^n \left(2\sqrt{5}\ ,\ 6\sqrt{5}\ ,\ 0\right)\cdot\mathbf{w} + c_{i,3}^n \left(2\sqrt{15}\ ,\ 2\sqrt{15}\ ,\ 4\sqrt{15}\right)\cdot\mathbf{w}. \tag{6.8}$$

In the directions $\mathbf{w}_1 = \frac{3}{\sqrt{11}}\left(1, -\frac{1}{3}, -\frac{1}{3}\right)$, $\mathbf{w}_2 = \frac{2}{\sqrt{5}}\left(0, 1, -\frac{1}{2}\right)$, and $\mathbf{w}_3 = (0,0,1)$, the directional derivatives are

$$D_{\mathbf{w}_1} U_i^n = \frac{12}{11}\sqrt{10}\sqrt{11} c_{i,1}^n \quad , \quad D_{\mathbf{w}_2} U_i^n = 12 c_{i,2}^n, \quad \text{and } D_{\mathbf{w}_3} U_i^n = 4\sqrt{15} c_{i,3}^n.$$

We map $\mathbf{w}$ into the physical space and normalize to obtain on Ω_i

$$\mathbf{v}_i = \frac{J_i \mathbf{w}}{||J_i \mathbf{w}||}.$$

For $\mathbf{w}_1$, $\mathbf{w}_2$, and $\mathbf{w}_3$ in canonical coordinates and the corresponding vectors in the physical space $\mathbf{v}_{i,1}$, $\mathbf{v}_{i,2}$, and $\mathbf{v}_{i,3}$, the directional derivatives are

$$D_{\mathbf{v}_{i,1}} U_i^n = c_{i,1}^n \frac{4\sqrt{10}}{\frac{\sqrt{11}}{3}||J_i \mathbf{w}_1||}, \quad D_{\mathbf{v}_{i,2}} U_i^n = c_{i,2}^n \frac{6\sqrt{5}}{\frac{\sqrt{5}}{2}||J_i \mathbf{w}_2||}, \quad \text{and} \quad D_{\mathbf{v}_{i,3}} U_i^n = c_{i,3}^n \frac{4\sqrt{15}}{||J_i \mathbf{w}_3||}.$$

Let $h_{i,1} = \frac{\sqrt{11}}{3}||J_i \mathbf{w}_1||$, $h_{i,2} = \frac{\sqrt{5}}{2}||J_i \mathbf{w}_2||$, and $h_{i,3} = ||J_i \mathbf{w}_3||$. The constants $h_{i,1}$, $h_{i,2}$, and $h_{i,3}$ have the geometrical interpretation illustrated in Figure 6.1, which follows from the Jacobian J_i (1.4) and geometrical considerations of the tetrahedron. The solution coefficients $c_{i,1}^n$, $c_{i,2}^n$, and $c_{i,3}^n$ written in terms of the derivatives $D_{\mathbf{v}_{i,1}} U_i^n$, $D_{\mathbf{v}_{i,2}} U_i^n$, and $D_{\mathbf{v}_{i,3}} U_i^n$ are

$$c_{i,1}^n = \frac{h_{i,1}}{4\sqrt{10}} D_{\mathbf{v}_{i,1}} U_i^n, \quad c_{i,2}^n = \frac{h_{i,2}}{6\sqrt{5}} D_{\mathbf{v}_{i,2}} U_i^n, \quad \text{and} \quad c_{i,3}^n = \frac{h_{i,3}}{4\sqrt{15}} D_{\mathbf{v}_{i,3}} U_i^n. \tag{6.9}$$

Next, we will reconstruct the directional derivatives of the numerical solution from the solution averages in these directions.

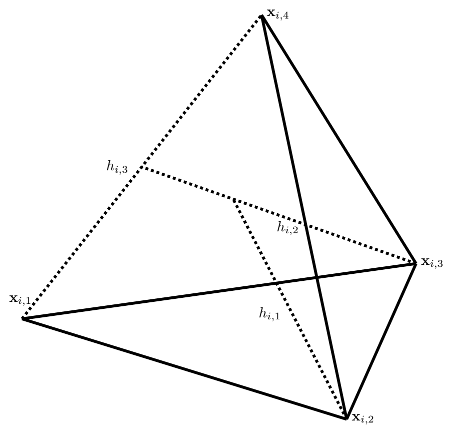


Figure 6.1: $h_{i,1}$ is the length of the segment connecting $\mathbf{x}_{i,2}$ and the centroid of the face defined by $\mathbf{x}_{i,1}$, $\mathbf{x}_{i,3}$, and $\mathbf{x}_{i,4}$. $h_{i,2}$ is the length of the segment connecting $\mathbf{x}_{i,3}$ and the midpoint of the edge defined by $\mathbf{x}_{i,1}$ and $\mathbf{x}_{i,4}$. $h_{i,3}$ is the length of the edge defined by $\mathbf{x}_{i,1}$ and $\mathbf{x}_{i,4}$.

6.1.1 Forward and backward differences

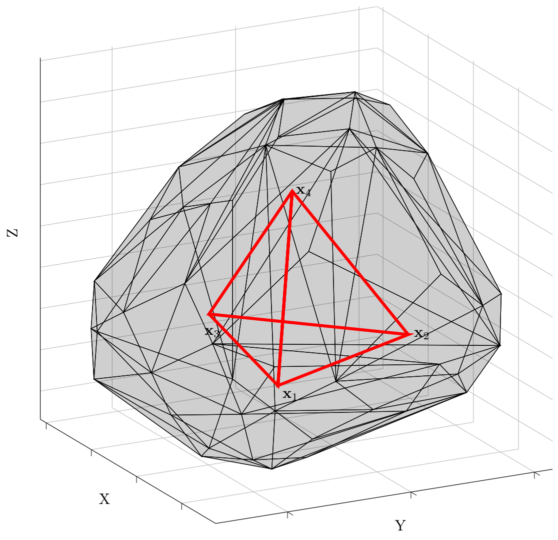
We aim to reconstruct the directional derivatives in these three directions using solution averages on neighboring elements. $U^f_{i,k}$ and $U^b_{i,k}$ are the forward and backward interpolated solution values at the interpolation points $\mathbf{x}^f_{i,k}$ and $\mathbf{x}^b_{i,k}$ in the directions $\mathbf{v}_{i,k}$ and $-\mathbf{v}_{i,k}$, respectively. The interpolation points lie at distances $d^f_{i,k}$ and $d^b_{i,k}$ from the the centroid $\mathbf{x}_i$, i.e. $\mathbf{x}^f_{i,k} - \mathbf{x}_i = d^f_{i,k}\mathbf{v}_{i,k}$ and $\mathbf{x}_i - \mathbf{x}^b_{i,k} = d^b_{i,k}\mathbf{v}_{i,k}$ (Figure 6.2). The reconstruction of the directional derivatives is done using the forward differences

$$\Delta^f_{i,1} = \frac{1}{d^f_{i,1}}(U^f_{i,1} - \overline{U}_i), \quad \Delta^f_{i,2} = \frac{1}{d^f_{i,2}}(U^f_{i,2} - \overline{U}_i), \quad \text{and} \quad \Delta^f_{i,3} = \frac{1}{d^f_{i,3}}(U^f_{i,3} - \overline{U}_i), \tag{6.10}$$

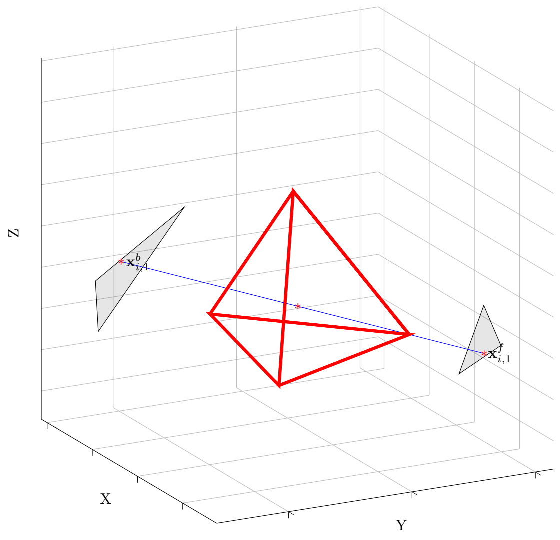
and the backward differences

$$\Delta^b_{i,1} = \frac{1}{d^b_{i,1}}(\overline{U}_i - U^b_{i,1}), \quad \Delta^b_{i,2} = \frac{1}{d^b_{i,2}}(\overline{U}_i - U^b_{i,2}), \quad \text{and} \quad \Delta^b_{i,3} = \frac{1}{d^b_{i,3}}(\overline{U}_i - U^b_{i,3}). \tag{6.11}$$

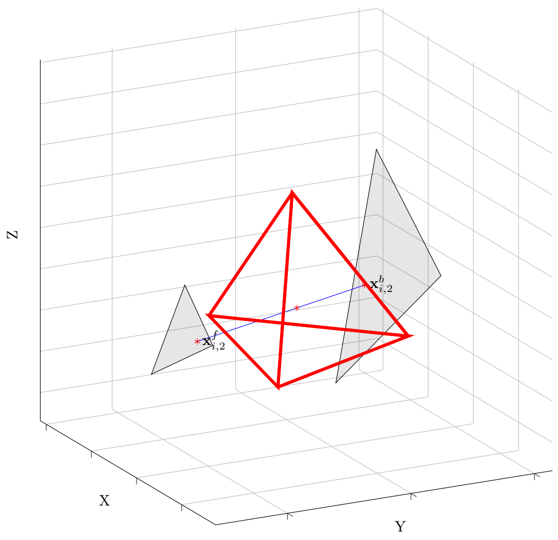
The reconstructed solution values $U^f_{i,k}$ and $U^b_{i,k}$ are determined from linear interpolation of neighboring solution averages. The determination of which elements to include in the linear



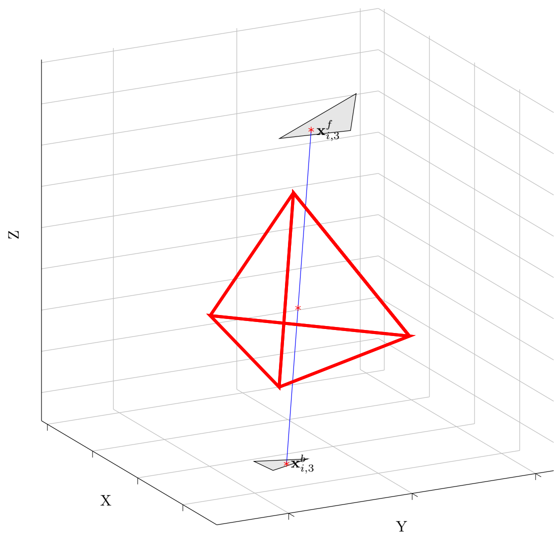
(a) Triangulation of the convex hull of neighboring centroids.



(b) Interpolation planes for $\mathbf{v}_{i,1}$.



(c) Interpolation planes for $\mathbf{v}_{i,2}$.



(d) Interpolation planes for $\mathbf{v}_{i,3}$.

Figure 6.2: Determining the forward and backward interpolation planes in the three limiting directions.

interpolation is completed once for a particular mesh as a preprocessing stage composed of the following three steps for each element Ω_i:

1. Compile the list of vertex neighbors of Ω_i. Two elements are considered vertex neighbors if they share a geometrical vertex.
2. Find the triangular faces of the convex hull defined by the centroids of the vertex neighbors (Figure 6.2a).
3. Determine the forward and backward interpolation elements for each limiting direction $\mathbf{v}_{i,k}$, $k = 1, 2, 3$.

After completion of Step 1 for all elements, the quickhull algorithm provided in the Qhull package [89] is used for Step 2. The algorithm returns the surface triangulation as a list of triangular faces. We iterate through the list until the forward and backwards interpolation points for the reconstruction direction $\mathbf{v}_{i,k}$ are found, $k = 1, 2, 3$. The interpolation points are the points of intersection between the hull and the line defined by $\mathbf{v}_{i,k}$ and passing through the centroid $\mathbf{x}_i$. An interpolation point (forward or backward) $\mathbf{p}$ in the direction $\mathbf{v}_{i,k}$ belongs to a face of the hull, where it satisfies

$$\mathbf{n} \cdot (\mathbf{p} - \mathbf{f}) = 0, \tag{6.12}$$

where $\mathbf{n}$ is the normal of the face and $\mathbf{f}$ is a vertex of the face. This point can be written in vector form

$$\mathbf{p} = \mathbf{x}_i + s\mathbf{v}_{i,k}, \tag{6.13}$$

where s is the Euclidean distance between the centroid of Ω_i and the interpolation point. Subtracting $\mathbf{f}$ from (6.13) and dotting the difference with $\mathbf{n}$, we can solve for s

$$s = \frac{(\mathbf{f} - \mathbf{x}_i) \cdot \mathbf{n}}{\mathbf{v}_{i,k} \cdot \mathbf{n}}. \tag{6.14}$$

If $s > 0$, then $\mathbf{x}_i + s\mathbf{v}_{i,k}$ is the forward interpolation point $\mathbf{x}^f_{i,k}$ and $d^f_{i,k} = s$. Alternatively, if $s < 0$ then it corresponds to a backward interpolation point $\mathbf{x}^b_{i,k}$ and $d^b_{i,k} = -s$. $\mathbf{p}$ can be written as a convex combination of the three vertices of the face to which it belongs using the barycentric coordinates. For example, consider the three forward interpolation elements for $\mathbf{v}_{i,1}$, the centroids of which we refer to as $\mathbf{x}_j$, $\mathbf{x}_q$, and $\mathbf{x}_m$. The area of the triangle formed by these centroids is A, and A_j, A_q, and A_m are the areas of the smaller triangles in Figure 6.3. The forward reconstructed solution value is

$$U^f_{i,1} = w^f_{i,j}\overline{U}_j + w^f_{i,q}\overline{U}_q + w^f_{i,m}\overline{U}_m,$$

where the interpolation weights are given by

$$w^f_{i,j} = \frac{A_j}{A}, \quad w^f_{i,q} = \frac{A_q}{A}, \quad \text{and} \quad w^f_{i,m} = \frac{A_m}{A}.$$

6.1.2 The limited numerical solution

We aim to limit the solution moments by comparing them to the appropriate forward and backward differences. From (6.9), the limited solution moments can be written in terms of the forward differences

$$\tilde{c}^n_{i,1} = l_{i,1}\frac{h_{i,1}}{4\sqrt{10}}\Delta^f_{i,1}, \quad \tilde{c}^n_{i,2} = l_{i,2}\frac{h_{i,2}}{6\sqrt{5}}\Delta^f_{i,2}, \quad \text{and} \quad \tilde{c}^n_{i,3} = l_{i,3}\frac{h_{i,3}}{4\sqrt{15}}\Delta^f_{i,3}. \tag{6.15}$$

where $l_{i,1}$, $l_{i,2}$, and $l_{i,3}$ are the non-negative limiting coefficients that will be specified later.

We aim to express the DG scheme with the limited numerical solution (6.3) in the form (6.5) by writing the inflow terms of (6.3) as

$$\tilde{U}^n_j(\mathbf{x}_{i,j}) - \overline{U}^n_i = f_{j,i}(\overline{U}^n_j - \overline{U}^n_i) + f_{j,i,1}(U^-_{j,i,1} - \overline{U}^n_i) + f_{j,i,2}(U^-_{j,i,2} - \overline{U}^n_i) + f_{j,i,3}(U^-_{j,i,3} - \overline{U}^n_i) \text{ for } j \in N^-_i, \tag{6.16}$$

where $f_{j,i}$, $f_{j,i,1}$, $f_{j,i,2}$, and $f_{j,i,3}$ are non-negative constants and $U^-_{j,i,1}$, $U^-_{j,i,2}$, and $U^-_{j,i,3}$ are reconstructed solution values in the forward or backward directions $\pm\mathbf{v}_{i,1}$, $\pm\mathbf{v}_{i,2}$, and $\pm\mathbf{v}_{i,3}$, respectively. Similarly, we will write the outflow terms of (6.3) as

$$\tilde{U}^n_i(\mathbf{x}_{i,j}) - \overline{U}^n_i = -\left[g_{i,j,1}(U^+_{i,j,1} - \overline{U}^n_i) + g_{i,j,2}(U^+_{i,j,2} - \overline{U}^n_i) + g_{i,j,3}(U^+_{i,j,3} - \overline{U}^n_i)\right] \text{ for } j \in N^+_i, \tag{6.17}$$

where $g_{i,j,1}$, $g_{i,j,2}$, and $g_{i,j,3}$ are non-negative constants and $U^+_{i,j,1}$, $U^+_{i,j,2}$, and $U^+_{i,j,3}$ are reconstructed solution values in the forward or backward directions $\pm\mathbf{v}_{i,1}$, $\pm\mathbf{v}_{i,2}$, and $\pm\mathbf{v}_{i,3}$, respectively.

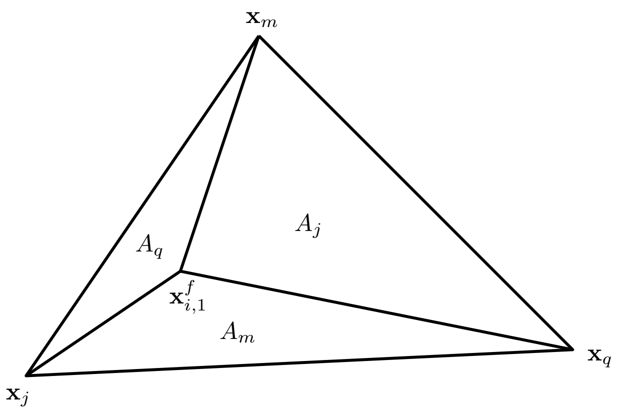


Figure 6.3: Areas A_j, A_q, and A_m used for determining the forward interpolation weights for limiting direction $\mathbf{v}_{i,1}$. The vertices $\mathbf{x}_j$, $\mathbf{x}_q$, $\mathbf{x}_m$, and $\mathbf{x}_{i,1}^f$ lie on the same plane.

6.1.3 Inflow term

We start with the inflow term of (6.3). The limited numerical solution on an inflow neighbor Ω_j at $\mathbf{x}(\mathbf{r}_{j,i})$ can be written using (6.10), (6.11), and (6.15) as

$$\tilde{U}_j^n(\mathbf{x}(\mathbf{r}_{j,i})) = \sum_{k=0}^{3} \tilde{c}_{j,k}\varphi_k(\mathbf{x}(\mathbf{r}_{j,i})) = \overline{U}_j^n + l_{j,1}\varphi_1(\mathbf{r}_{j,i})\frac{h_{j,1}}{4\sqrt{10}}\frac{1}{d_{j,1}^f}(U_{j,1}^f - \overline{U}_j^n) + l_{j,2}\varphi_2(\mathbf{r}_{j,i})\frac{h_{j,2}}{6\sqrt{5}}\frac{1}{d_{j,2}^f}(U_{j,2}^f - \overline{U}_j^n) + l_{j,3}\varphi_3(\mathbf{r}_{j,i})\frac{h_{j,3}}{4\sqrt{15}}\frac{1}{d_{j,3}^f}(U_{j,3}^f - \overline{U}_j^n). \quad (6.18)$$

We relate the forward and backward differences with

$$r_{i,k} = \frac{\Delta_{i,k}^b}{\Delta_{i,k}^f} \text{ for } k = 1, 2, 3, \quad (6.19)$$

and introduce the following notation for convenience

$$\alpha_{j,i,1}^- = \begin{cases} \frac{h_{j,1}}{4\sqrt{10}}\frac{\varphi_1(\mathbf{r}_{j,i})}{d_{j,1}^f} \text{ if } \varphi_1(\mathbf{r}_{j,i}) \geq 0, \\ \frac{h_{j,1}}{4\sqrt{10}}\frac{|\varphi_1(\mathbf{r}_{j,i})|}{r_{j,1}d_{j,1}^b} \text{ otherwise,} \end{cases} \quad \text{and} \quad U_{j,i,1}^- = \begin{cases} U_{j,1}^f \text{ if } \varphi_1(\mathbf{r}_{j,i}) \geq 0, \\ U_{j,1}^b \text{ otherwise.} \end{cases} \quad (6.20)$$

$$\alpha^-_{j,i,2} = \begin{cases} \frac{h_{j,2}}{6\sqrt{5}} \frac{\varphi_2(\mathbf{r}_{j,i})}{d^f_{j,2}} \text{ if } \varphi_2(\mathbf{r}_{j,i}) \geq 0, \\ \frac{h_{j,2}}{6\sqrt{5}} \frac{|\varphi_2(\mathbf{r}_{j,i})|}{r_{j,2} d^b_{j,2}} \text{ otherwise,} \end{cases} \quad \text{and} \quad U^-_{j,i,2} = \begin{cases} U^f_{j,2} \text{ if } \varphi_2(\mathbf{r}_{j,i}) \geq 0, \\ U^b_{j,2} \text{ otherwise.} \end{cases} \tag{6.21}$$

$$\alpha^-_{j,i,3} = \begin{cases} \frac{h_{j,3}}{4\sqrt{15}} \frac{\varphi_3(\mathbf{r}_{j,i})}{d^f_{j,3}} \text{ if } \varphi_3(\mathbf{r}_{j,i}) \geq 0, \\ \frac{h_{j,3}}{4\sqrt{15}} \frac{|\varphi_3(\mathbf{r}_{j,i})|}{r_{j,3} d^b_{j,3}} \text{ otherwise,} \end{cases} \quad \text{and} \quad U^-_{j,i,3} = \begin{cases} U^f_{j,3} \text{ if } \varphi_3(\mathbf{r}_{j,i}) \geq 0, \\ U^b_{j,3} \text{ otherwise.} \end{cases} \tag{6.22}$$

Using (6.20)-(6.22) in (6.18), we obtain

$$\tilde{U}^n_j(\mathbf{x}_{i,j}) = \overline{U}^n_j + l_{j,1}\alpha^-_{j,i,1}(U^-_{j,i,1} - \overline{U}^n_j) + l_{j,2}\alpha^-_{j,i,2}(U^-_{j,i,2} - \overline{U}^n_j) + l_{j,3}\alpha^-_{j,i,3}(U^-_{j,i,3} - \overline{U}^n_j). \tag{6.23}$$

Subtracting $\overline{U}^n_i$ from both sides of (6.23), then adding and subtracting $\overline{U}^n_i$ in the last three terms on the right, we have

$$\begin{aligned} \tilde{U}^n_j(\mathbf{x}_{i,j}) - \overline{U}^n_i = \overline{U}^n_j - \overline{U}^n_i + l_{j,1}\alpha^-_{j,i,1}(U^-_{j,i,1} - \overline{U}^n_i + \overline{U}^n_i - \overline{U}^n_j) + l_{j,2}\alpha^-_{j,i,2}(U^-_{j,i,2} - \overline{U}^n_i + \overline{U}^n_i - \overline{U}^n_j) \\ + l_{j,3}\alpha^-_{j,i,3}(U^-_{j,i,3} - \overline{U}^n_i + \overline{U}^n_i - \overline{U}^n_j), \end{aligned}$$

which gives

$$\begin{aligned} \tilde{U}^n_j(\mathbf{x}_{i,j}) - \overline{U}^n_i &= (1 - l_{j,1}\alpha^-_{j,i,1} - l_{j,2}\alpha^-_{j,i,2} - l_{j,3}\alpha^-_{j,i,3})(\overline{U}^n_j - \overline{U}^n_i) \\ &+ l_{j,1}\alpha^-_{j,i,1}(U^-_{j,i,1} - \overline{U}^n_i) + l_{j,2}\alpha^-_{j,i,2}(U^-_{j,i,2} - \overline{U}^n_i) + l_{j,3}\alpha^-_{j,i,3}(U^-_{j,i,3} - \overline{U}^n_i). \end{aligned} \tag{6.24}$$

With the introduction of coefficients

$$\begin{aligned} f_{j,i} &= 1 - l_{j,1}\alpha^-_{j,i,1} - l_{j,2}\alpha^-_{j,i,2} - l_{j,3}\alpha^-_{j,i,3}, \\ f_{j,i,1} &= l_{j,1}\alpha^-_{j,i,1}, \\ f_{j,i,2} &= l_{j,2}\alpha^-_{j,i,2}, \\ f_{j,i,3} &= l_{j,2}\alpha^-_{j,i,3}, \end{aligned} \tag{6.25}$$

(6.24) is now in the form (6.16).

6.1.4 Outflow term

Now consider the outflow term of (6.3). The limited numerical solution on Ω_i at $\mathbf{x}(\mathbf{r}_{i,j})$ can be written using (6.10), (6.11), and (6.15) as

$$\begin{aligned}\tilde{U}_i^n(\mathbf{x}(\mathbf{r}_{i,j})) - \overline{U}_i^n = l_{i,1}\varphi_1(\mathbf{r}_{i,j})\frac{h_{i,1}}{4\sqrt{10}}\frac{1}{d_{i,1}^f}(U_{i,1}^f - \overline{U}_i^n) + l_{i,2}\varphi_2(\mathbf{r}_{i,j})\frac{h_{i,2}}{6\sqrt{5}}\frac{1}{d_{i,2}^f}(U_{i,2}^f - \overline{U}_i^n) \\ + l_{i,3}\varphi_3(\mathbf{r}_{i,j})\frac{h_{i,3}}{4\sqrt{15}}\frac{1}{d_{i,3}^f}(U_{i,3}^f - \overline{U}_i^n).\end{aligned} \tag{6.26}$$

As for the inflow term, we introduce the following notation for convenience

$$\alpha_{i,1}^+ = \begin{cases} \frac{h_{i,1}}{4\sqrt{10}}\frac{|\varphi_1(\mathbf{r}_{i,j})|}{d_{i,1}^f} \text{ if } \varphi_1(\mathbf{r}_{i,j}) \le 0, \\ \frac{h_{i,1}}{4\sqrt{10}}\frac{\varphi_1(\mathbf{r}_{i,j})}{r_{i,1}d_{i,1}^b} \text{ otherwise,} \end{cases} \quad \text{and } U_{i,j,1}^+ = \begin{cases} U_{i,1}^f \text{ if } \varphi_1(\mathbf{r}_{i,j}) \le 0 \\ U_{i,1}^b \text{ otherwise,} \end{cases} \tag{6.27}$$

$$\alpha_{i,2}^+ = \begin{cases} \frac{h_{i,2}}{6\sqrt{5}}\frac{|\varphi_2(\mathbf{r}_{i,j})|}{d_{i,2}^f} \text{ if } \varphi_2(\mathbf{r}_{i,j}) \le 0 \\ \frac{h_{i,2}}{6\sqrt{5}}\frac{\varphi_2(\mathbf{r}_{i,j})}{r_{i,2}d_{i,2}^b} \text{ otherwise.} \end{cases} \quad \text{and } \; U_{i,j,2}^+ = \begin{cases} U_{i,2}^f \text{ if } \varphi_2(\mathbf{r}_{i,j}) \le 0 \\ U_{i,2}^b \text{ otherwise.} \end{cases} \tag{6.28}$$

$$\alpha_{i,3}^+ = \begin{cases} \frac{h_{i,3}}{4\sqrt{15}}\frac{|\varphi_3(\mathbf{r}_{i,j})|}{d_{i,3}^f} \text{ if } \varphi_3(\mathbf{r}_{i,j}) \le 0 \\ \frac{h_{i,3}}{4\sqrt{15}}\frac{\varphi_3(\mathbf{r}_{i,j})}{r_{i,3}d_{i,3}^b} \text{ otherwise.} \end{cases} \quad \text{and } \; U_{i,j,3}^+ = \begin{cases} U_{i,3}^f \text{ if } \varphi_3(\mathbf{r}_{i,j}) \le 0 \\ U_{i,3}^b \text{ otherwise.} \end{cases} \tag{6.29}$$

Therefore, (6.26) becomes

$$\tilde{U}_i^n(\mathbf{x}_{i,j}) - \overline{U}_i^n = -\left[l_{i,1}\alpha_{i,j,1}^+(U_{i,j,1}^+ - \overline{U}_i^n) + l_{i,2}\alpha_{i,j,2}^+(U_{i,j,2}^+ - \overline{U}_i^n) + l_{i,3}\alpha_{i,j,3}^+(U_{i,j,3}^+ - \overline{U}_i^n)\right], \tag{6.30}$$

with

$$\begin{aligned} g_{i,j,1} &= l_{i,1}\alpha_{i,j,1}^+ \\ g_{i,j,2} &= l_{i,2}\alpha_{i,j,2}^+ \\ g_{i,j,3} &= l_{i,2}\alpha_{i,j,3}^+. \end{aligned} \tag{6.31}$$

(6.26) is now in the form (6.17).

6.1.5 Putting it all together

Combining (6.24) and (6.30), we obtain

$$\overline{U}_i^{n+1} = \overline{U}_i^n + \sum_{j\in N_i^-} v_{j,i}^- \Big[f_{j,i}(\overline{U}_j^n - \overline{U}_i^n) + f_{j,i,1}(U_{j,i,1}^- - \overline{U}_i^n) + f_{j,i,2}(U_{j,i,2}^- - \overline{U}_i^n) + f_{j,i,3}(U_{j,i,3}^- - \overline{U}_i^n) \Big]$$

$$+ \sum_{j\in N_i^+} v_{i,j}^+ \left[g_{i,j,1}(U_{i,j,1}^+ - \overline{U}_i^n) + g_{i,j,2}(U_{i,j,2}^+ - \overline{U}_i^n) + g_{i,j,3}(U_{i,j,3}^+ - \overline{U}_i^n) \right]. \tag{6.32}$$

This is now in the form (6.5). In order for $\overline{U}_i^{n+1}$ to satisfy the LMP (6.4), the sum condition (6.6) must be satisfied, i.e.,

$$\sum_{j\in N_i^-} v_{j,i}^- \left[f_{j,i} + f_{j,i,1} + f_{j,i,2} + f_{j,i,3} \right] + \sum_{j\in N_i^+} v_{i,j}^+ \left[g_{i,j,1} + g_{i,j,2} + g_{i,j,3} \right] \le 1. \tag{6.33}$$

Using (6.25) and (6.31) in the above yields

$$\sum_{j\in N_i^-} v_{j,i}^- + \sum_{j\in N_i^+} v_{i,j}^+ \left(l_{i,1}\alpha_{i,j,1}^+ + l_{i,2}\alpha_{i,j,2}^+ + l_{i,3}\alpha_{i,j,3}^+ \right) \le 1. \tag{6.34}$$

For (6.34) to be satisfied, we enforce on each outflow face of Ω_i

$$l_{i,1}\alpha_{i,j,1}^+ + l_{i,2}\alpha_{i,j,2}^+ + l_{i,3}\alpha_{i,j,3}^+ \le 1, \quad \forall j \in N_i^+, \tag{6.35}$$

and on all elements

$$\sum_{j\in N_i^-} v_{j,i}^- + \sum_{j\in N_i^+} v_{i,j}^+ \le 1. \tag{6.36}$$

The non-negativity condition (6.7) is satisfied if $f_{j,i} \ge 0$ by (6.25), i.e.

$$l_{j,1}\alpha_{j,i,1}^- + l_{j,2}\alpha_{j,i,2}^- + l_{j,3}\alpha_{j,i,3}^- \le 1, \quad \forall j \in N_i^-. \tag{6.37}$$

The other coefficients $f_{j,i,1}$, $f_{j,i,2}$, $f_{j,i,3}$ and $g_{i,j,1}$, $g_{i,j,2}$, $g_{i,j,3}$ are non-negative by construction.

6.1.6 Time step restriction

Using the definitions of $v_{j,i}^-$ and $v_{i,j}^+$ in (6.2), we have from (6.36)

$$\Delta t \left(\sum_{j \in N_i^-} |\mathbf{a} \cdot \mathbf{n}_{i,j}| \frac{|\partial\Omega_{i,j}|}{|\Omega_i|} + \sum_{j \in N_i^+} |\mathbf{a} \cdot \mathbf{n}_{i,j}| \frac{|\partial\Omega_{i,j}|}{|\Omega_i|} \right) \leq 1. \tag{6.38}$$

since $\mathbf{a} \cdot \mathbf{n}_{i,j} < 0$ for $j \in N_i^-$ and $\mathbf{a} \cdot \mathbf{n}_{i,j} > 0$ for $j \in N_i^+$. From (3.14), we obtain

$$-\sum_{j \in N_i^-} |\partial\Omega_{i,j}| \mathbf{a} \cdot \mathbf{n}_{i,j} = \sum_{j \in N_i^+} |\partial\Omega_{i,j}| \mathbf{a} \cdot \mathbf{n}_{i,j}.$$

Because $\mathbf{a} \cdot \mathbf{n}_{i,j} < 0$ for $j \in N_i^-$ and $\mathbf{a} \cdot \mathbf{n}_{i,j} > 0$ for $j \in N_i^+$, this becomes

$$\sum_{j \in N_i^-} |\partial\Omega_{i,j}| |\mathbf{a} \cdot \mathbf{n}_{i,j}| = \sum_{j \in N_i^+} |\partial\Omega_{i,j}| |\mathbf{a} \cdot \mathbf{n}_{i,j}|. \tag{6.39}$$

Using the above, we can determine a restriction on the time step. First, consider the case of one outflow face and three inflow faces, as illustrated in Figure 6.4a. From (6.39), we have

$$\sum_{j \in N_i^-} |\partial\Omega_{i,j}| |\mathbf{a} \cdot \mathbf{n}_{i,j}| = |\partial\Omega_{i,J}| |\mathbf{a} \cdot \mathbf{n}_{i,J}|, \tag{6.40}$$

where $\partial\Omega_{i,J}$ is the single outflow face (the face opposite vertex $\mathbf{x}_{i,1}$ in Figure 6.4a). Using (6.40) in (6.38), we obtain

$$2\Delta t \frac{|\partial\Omega_{i,J}| |\mathbf{a} \cdot \mathbf{n}_{i,J}|}{|\Omega_i|} \leq 1. \tag{6.41}$$

From Figure 6.4a, we have

$$\frac{|\mathbf{a} \cdot \mathbf{n}_{i,J}|}{||\mathbf{a}||} = \cos\theta = \frac{H_i}{h_i}, \tag{6.42}$$

where $\mathbf{n}_{i,J}$ is the unit outward normal to the outflow face, θ is the angle between $\mathbf{a}$ and $\mathbf{n}_{i,J}$, h_i is the cell width in the direction of $\mathbf{a}$ shown in Figure 6.4a, and H_i is the cell width in the direction of $\mathbf{n}_{i,J}$. Finally, the volume of the tetrahedron is

$$|\Omega_i| = \frac{1}{3} |\partial\Omega_{i,J}| H_i. \tag{6.43}$$

Using (6.42) and (6.43) in (6.41) and solving for Δt, the time step restriction becomes

$$\Delta t \leq \frac{1}{6}\frac{h_i}{||\mathbf{a}||}. \tag{6.44}$$

This result is also valid in the case of three inflow faces and one outflow face.

Consider now the case of two inflow and two outflow faces. We subdivide Ω_i into two smaller tetrahedra as illustrated in Figure 6.4b. Summing the volume of these two smaller tetrahedra gives the volume of the larger tetrahedron

$$|\Omega_i| = \frac{1}{3}|\partial\Omega_{i,K}|H_K + \frac{1}{3}|\partial\Omega_{i,L}|H_L, \tag{6.45}$$

where H_K and H_L are the heights of the two smaller tetrahedra measured from the two inflow faces given by vertices ($\mathbf{x}_{i,1}$, $\mathbf{x}_{i,2}$, $\mathbf{x}_{i,4}$) and ($\mathbf{x}_{i,2}$, $\mathbf{x}_{i,3}$, $\mathbf{x}_{i,4}$), respectively. Multiplying and dividing (6.45) by h_i, which is now the cell width illustrated in Figure 6.4b, and using (6.42), we have

$$|\Omega_i| = \frac{1}{3}\frac{h_i}{||\mathbf{a}||}\left(|\partial\Omega_{i,K}||\mathbf{a}\cdot\mathbf{n}_{i,K}| + |\partial\Omega_{i,L}||\mathbf{a}\cdot\mathbf{n}_{i,L}|\right). \tag{6.46}$$

In this case, h_i is the line segment with direction $\mathbf{a}$ that connects the edge shared by the two inflow faces ($\mathbf{x}_{i,2}$ and $\mathbf{x}_{i,4}$) with the edge shared by the two outflow faces ($\mathbf{x}_{i,1}$ and $\mathbf{x}_{i,3}$). Expressing the time step restriction for this case in terms of the two inflow faces using (6.39) in (6.38), we obtain

$$2\Delta t\frac{|\partial\Omega_{i,K}||\mathbf{a}\cdot\mathbf{n}_{i,K}| + |\partial\Omega_{i,L}||\mathbf{a}\cdot\mathbf{n}_{i,L}|}{|\Omega_i|} \leq 1. \tag{6.47}$$

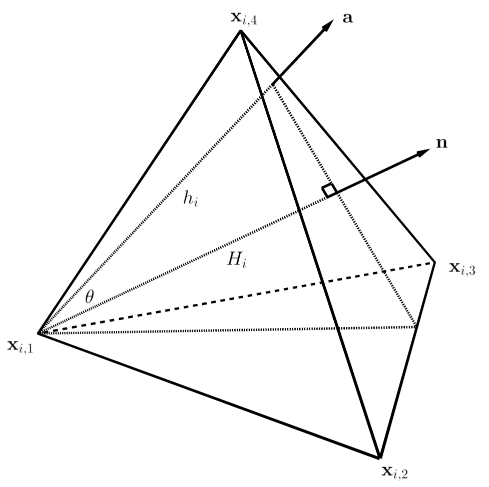
Substituting (6.46) into (6.47), we then obtain the same restriction on the time step as in (6.44) although with the h_i defined in Figure 6.4b.

In the case where $\mathbf{a}$ is parallel to one or two of the faces, the analysis is similar to the case of one inflow and three outflow faces and h_i is the one depicted in Figure 6.4a.

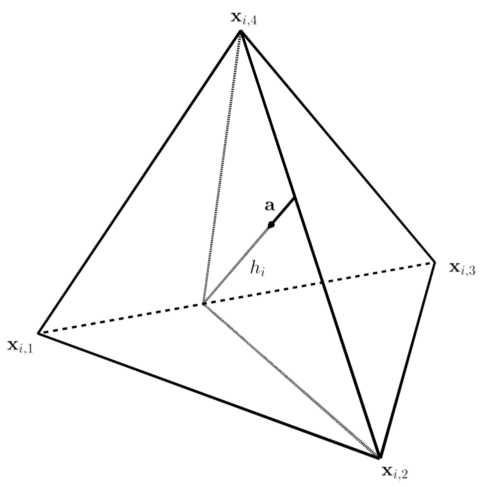
For systems of equations, generally, there is not a single direction of wave propagation. A simple measure of cell size independent of the direction of propagation is

$$h_i = 3\frac{|\Omega_i|}{|\partial\Omega_{i,j_1}| + |\partial\Omega_{i,j_2}|}. \tag{6.48}$$

where $\partial\Omega_{i,j_1}$ and $\partial\Omega_{i,j_2}$ are the faces of Ω_i with the largest areas.



(a) Three inflow faces and one outflow face.



(b) Two inflow faces and two outflow faces.

Figure 6.4: Measure of cell width h_i in the direction of flow.

6.2 Moment limiter

Analysis in Sections 6.1.3 - 6.1.6 shows that each outflow face of an element produces two constraints on that element's limiting coefficients. The first constraint on Ω_i's limiting coefficients results from (6.35) and the second results from (6.37) via Ω_i's neighbor. By (6.20)-(6.22) and (6.27)-(6.29), the constraints depend on the value of the basis function evaluated at the centroid of the outflow face (Table 6.1). We will have four sets of possible conditions since a tetrahedron has four faces. The constraints on the limiting coefficients based on the face number $s = 1 \dots 4$ are

$$\frac{l_{i,1}}{4\gamma^b_{i,1} r_{i,1}} \le 1 \quad \text{and} \quad \frac{l_{i,1}}{4\gamma^f_{i,1}} \le 1 \quad \text{if} \quad s = 1, \tag{6.49}$$

$$\frac{l_{i,1}}{12\gamma^f_{i,1}} + \frac{l_{i,2}}{\frac{9}{2}\gamma^b_{i,2} r_{i,2}} \le 1 \quad \text{and} \quad \frac{l_{i,1}}{12\gamma^b_{i,1} r_{i,1}} + \frac{l_{i,2}}{\frac{9}{2}\gamma^f_{i,2}} \le 1 \quad \text{if} \quad s = 2, \tag{6.50}$$

$$\frac{l_{i,1}}{12\gamma^f_{i,1}} + \frac{l_{i,2}}{9\gamma^f_{i,2}} + \frac{l_{i,3}}{6\gamma^b_{i,3} r_{i,3}} \le 1 \quad \text{and} \quad \frac{l_{i,1}}{12\gamma^b_{i,1} r_{i,1}} + \frac{l_{i,2}}{9\gamma^b_{i,2} r_{i,2}} + \frac{l_{i,3}}{6\gamma^f_{i,3}} \le 1 \quad \text{if} \quad s = 3 \tag{6.51}$$

$$\frac{l_{i,1}}{12\gamma^f_{i,1}} + \frac{l_{i,2}}{9\gamma^f_{i,2}} + \frac{l_{i,3}}{6\gamma^f_{i,3}} \le 1 \quad \text{and} \quad \frac{l_{i,1}}{12\gamma^b_{i,1} r_{i,1}} + \frac{l_{i,2}}{9\gamma^b_{i,2} r_{i,2}} + \frac{l_{i,3}}{6\gamma^b_{i,3} r_{i,3}} \le 1 \quad \text{if} \quad s = 4, \tag{6.52}$$

where $\gamma^f_{i,k} = \frac{d^f_{i,k}}{h_{i,k}}$, $\gamma^b_{i,k} = \frac{d^b_{i,k}}{h_{i,k}}$. The above inequalities are satisfied if

$$l_{i,1} \le 4\min(\gamma^f_{i,1}, \gamma^b_{i,1} r_{i,1}) \text{ and } l_{i,2} \le 3\min(\gamma^f_{i,2}, \gamma^b_{i,2} r_{i,2}) \text{ and } l_{i,3} \le 2\min(\gamma^f_{i,3}, \gamma^b_{i,3} r_{i,3}). \tag{6.53}$$

We now write the inequalities in terms of the solution moments assuming a non-negative forward difference. Note that a similar analysis can be used for negative forward differences. Multiplying the inequalities in (6.53) by $\Delta^f_{i,1}\frac{h_{i,1}}{4\sqrt{10}}$, $\Delta^f_{i,2}\frac{h_{i,2}}{6\sqrt{5}}$, and $\Delta^f_{i,3}\frac{h_{i,3}}{4\sqrt{15}}$, respectively, gives

$$l_{i,1}\Delta^f_{i,1}\frac{h_{i,1}}{4\sqrt{10}} \le \frac{h_{i,1}}{\sqrt{10}}\Delta^f_{i,1}\min(\gamma^f_{i,1}, \gamma^b_{i,1} r_{i,1}), \tag{6.54}$$

$$l_{i,2}\Delta^f_{i,2}\frac{h_{i,2}}{6\sqrt{5}} \le \frac{h_{i,2}}{2\sqrt{5}}\Delta^f_{i,2}\min(\gamma^f_{i,2}, \gamma^b_{i,2} r_{i,2}), \tag{6.55}$$

$$l_{i,3}\Delta^f_{i,3}\frac{h_{i,3}}{4\sqrt{15}} \le \frac{h_{i,3}}{2\sqrt{15}}\Delta^f_{i,3}\min(\gamma^f_{i,3}, \gamma^b_{i,3} r_{i,3}). \tag{6.56}$$

Substituting the expressions of the forward differences, $\gamma_{i,k}$, and $r_{i,k}$ for $k = 1, 2, 3$ and replacing the left hand side of the above inequalities with the limited solution moments, we have

$$\tilde{c}^n_{i,1} \leq \min\left(\frac{U^f_{i,1} - \overline{U}^n_i}{\sqrt{10}}, \frac{\overline{U}^n_i - U^b_{i,1}}{\sqrt{10}}\right), \tag{6.57}$$

$$\tilde{c}^n_{i,2} \leq \min\left(\frac{U^f_{i,2} - \overline{U}^n_i}{2\sqrt{5}}, \frac{\overline{U}^n_i - U^b_{i,2}}{2\sqrt{5}}\right), \tag{6.58}$$

$$\tilde{c}^n_{i,3} \leq \min\left(\frac{U^f_{i,3} - \overline{U}^n_i}{2\sqrt{15}}, \frac{\overline{U}^n_i - U^b_{i,3}}{2\sqrt{15}}\right). \tag{6.59}$$

The limiter must restrict the solution coefficients to the interval defined by these upper bounds defined in (6.57)-(6.59), i.e. the limiter becomes

$$\tilde{c}^n_{i,1} = \operatorname{minmod}\left(\frac{U^f_{i,1} - \overline{U}^n_i}{\sqrt{10}}, c^n_{i,1}, \frac{\overline{U}^n_i - U^b_{i,1}}{\sqrt{10}}\right), \tag{6.60}$$

$$\tilde{c}^n_{i,2} = \operatorname{minmod}\left(\frac{U^f_{i,2} - \overline{U}^n_i}{2\sqrt{5}}, c^n_{i,2}, \frac{\overline{U}^n_i - U^b_{i,2}}{2\sqrt{5}}\right), \tag{6.61}$$

$$\tilde{c}^n_{i,3} = \operatorname{minmod}\left(\frac{U^f_{i,3} - \overline{U}^n_i}{2\sqrt{15}}, c^n_{i,3}, \frac{\overline{U}^n_i - U^b_{i,3}}{2\sqrt{15}}\right). \tag{6.62}$$

6.2.1 Geometrical requirements

Second order accuracy requires that linear data is not limited. In this case, we have that the ratio of forward and backward differences is 1, i.e., $r_{i,k} = 1$ for $k = 1, 2, 3$, $\forall i$. In order for the slope to not be limited, we must have $l_{i,k} = 1$ for $k = 1, 2, 3$, $\forall i$ by (6.15). Using (6.53), this results in the following geometrical requirements on the limiting coefficients

$$1 \leq 4\min(\gamma^f_{i,1}, \gamma^b_{i,1} r_{i,1}) \text{ and } 1 \leq 3\min(\gamma^f_{i,2}, \gamma^b_{i,2} r_{i,2}) \text{ and } 1 \leq 2\min(\gamma^f_{i,3}, \gamma^b_{i,3} r_{i,3}).$$

In terms of the forward and backward interpolation distances, we have

$$h_{i,1} \leq 4\min(d^f_{i,1}, d^b_{i,1}) \text{ and } h_{i,2} \leq 3\min(d^f_{i,2}, d^b_{i,2}) \text{ and } h_{i,3} \leq 2\min(d^f_{i,3}, d^b_{i,3}).$$

We check that the above requirements are satisfied after determining the forward and backward interpolation points.

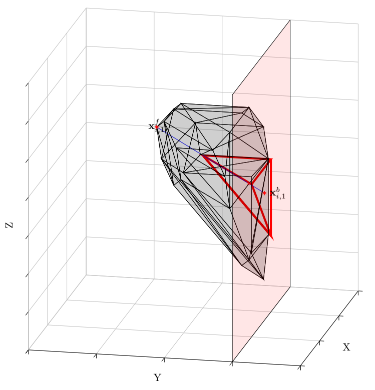
s	φ_1	φ_2	φ_3
1	$-\sqrt{10}$	0	0
2	$\frac{1}{3}\sqrt{10}$	$-\frac{4}{3}\sqrt{5}$	0
3	$\frac{1}{3}\sqrt{10}$	$\frac{2}{3}\sqrt{5}$	$-\frac{2}{3}\sqrt{15}$
4	$\frac{1}{3}\sqrt{10}$	$\frac{2}{3}\sqrt{5}$	$\frac{2}{3}\sqrt{15}$

Table 6.1: Values of the basis functions at the centroids of the canonical faces with face number s.

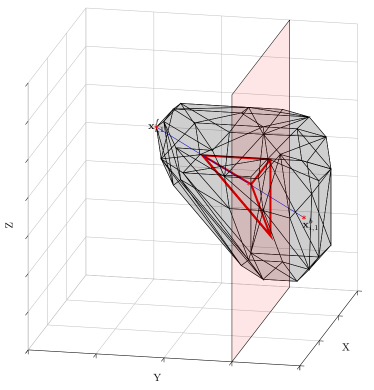
6.3 Limiting on reflecting boundaries

Limiting elements on the domain's boundary in unstructured meshes is nontrivial and rarely mentioned in the literature. We present our approach to limiting across planar reflecting boundaries, though we expect that our methodology extends to curved geometries. A reflecting boundary condition (BC) can model the influence of a solid wall in a flow and is implemented using ghost elements. The ghost elements take on a reflected state where the ghost density, energy, and speeds in the two directions tangential to the plane are taken to be the same as the interior value $\rho_g = \rho_i$, $E_g = E_i$, $\mathbf{u}_g \cdot \mathbf{t}_1 = \mathbf{u}_i \cdot \mathbf{t}_1$, and $\mathbf{u}_g \cdot \mathbf{t}_2 = \mathbf{u}_i \cdot \mathbf{t}_2$. The ghost speed in the direction normal to the plane of symmetry is negative that of the interior value, $\mathbf{u}_g \cdot \mathbf{n} = -\mathbf{u}_i \cdot \mathbf{n}$.

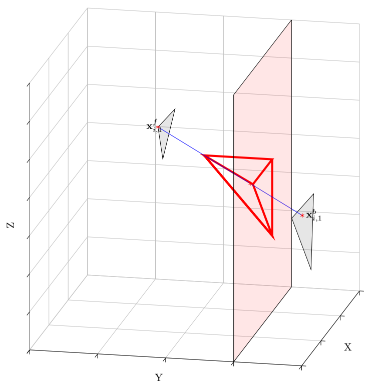
The difficulty in limiting in the neighborhood of these boundaries arises in Step 3 of the reconstruction procedure in Section 6.1.1. If an element has a vertex on a reflecting boundary, then it may not be possible to satisfy the second order geometrical requirements in Section 6.2.1. Our solution to this problem is expanding the vertex neighborhood by including neighbors reflected about the plane of symmetry located on the boundary (Figure 6.5). For neighbors located on the corner of two reflecting planes, the neighbors are reflected about both planes of symmetry (Figures 6.5d and 6.5e).



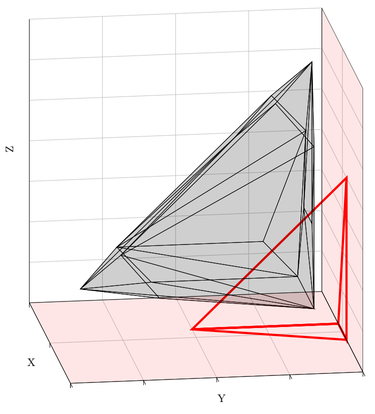
(a) A boundary element's convex hull.



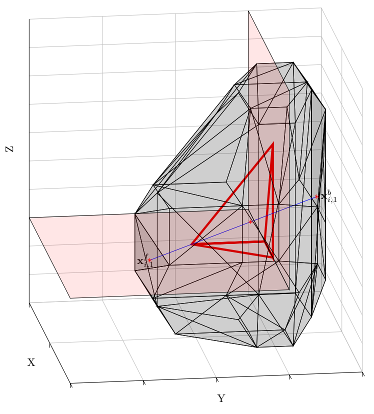
(b) The convex hull after reflection of elements about the plane of symmetry.



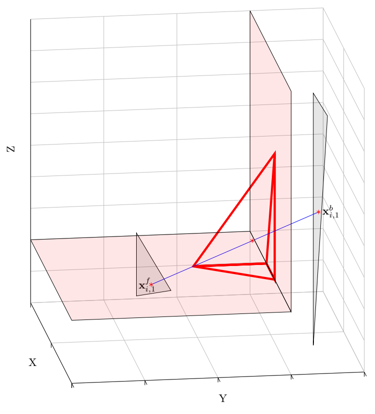
(c) The forward and backward interpolation planes after reflection.



(d) Boundary element located in the corner of two reflecting planes.



(e) The convex hull after reflection of the elements about the two planes of symmetry.



(f) The forward and backward interpolation planes after reflection.

Figure 6.5: Reflecting neighboring centroids about the planes of symmetry (shaded red).

6.4 Computed examples

6.4.1 Verification of the CFL

In this example, we demonstrate numerically that the time step restriction (6.44) is tight. We construct two meshes by first subdividing a unit cube into a mesh of six tetrahedra using two decompositions (Figures 6.6, 6.7). Next, the dimensions of the tetrahedra are scaled by a factor of $\frac{1}{30}$. Finally, the cube of scaled tetrahedra is tiled on the domain $[-1,1]^3$, resulting in a regular mesh. In the direction $[1,1,0]$, the minimum cell width on every tetrahedron in the unit cube is $\sqrt{2}$ for the first decomposition. After scaling, it becomes $h_i = \frac{1}{30}\sqrt{2}$. In the direction $[-1,1,1]$, the minimum cell width on every tetrahedron in the unit cube is $\sqrt{3}$ for the second decomposition. After scaling, it becomes $h_i = \frac{1}{30}\sqrt{3}$. The resulting tetrahedral meshes are composed of 1,296,000 elements. We solve (1.1) until the final time $T = 0.25$ with the linear flux $\mathbf{F}(u) = [u, u, 0]$, corresponding to the flow direction $[1,1,0]$ for the first mesh. On the second mesh, we use the flux $\mathbf{F}(u) = [-u, u, u]$ corresponding to the flow direction $[-1,1,1]$. The initial condition is given by

$$u_0(x,y,z) = \begin{cases} 1 \text{ if } \sqrt{x^2+y^2+z^2} \leq \frac{1}{4}, \\ 0 \text{ otherwise.} \end{cases}$$

In Tables 6.2 and 6.3, we report the maximum and minimum cell averages at the end of the simulation on the two meshes for various CFL numbers in (6.44). As predicted by the analysis in Section 6.1.6 with forward Euler time stepping, the scheme is local maximum principle satisfying for a CFL of $\frac{1}{6}$. For RK2 time stepping, we observe that it is possible to take a larger CFL and still satisfy the local maximum principle (Chapter 4, [90]).

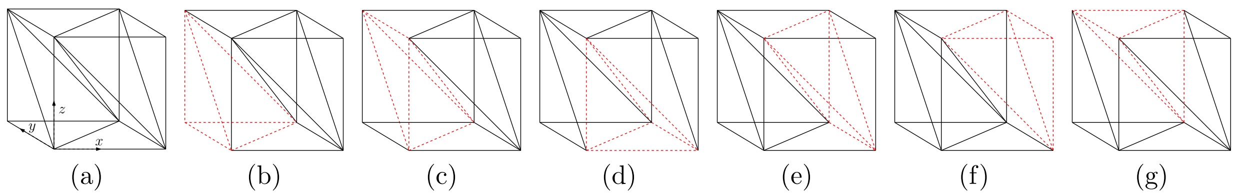


Figure 6.6: First decomposition of the unit cube into six tetrahedra indicated by dashed red lines.

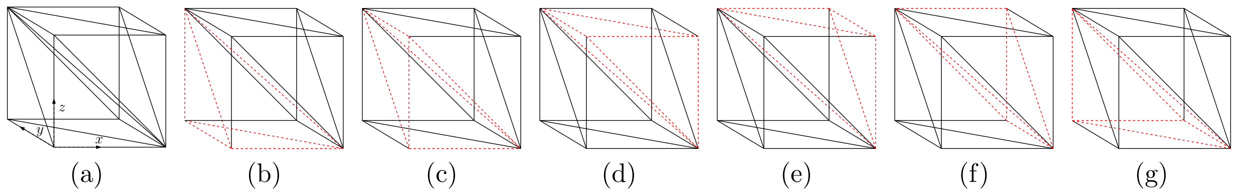


Figure 6.7: Second decomposition of the unit cube into six tetrahedra indicated by dashed red lines.

1/CFL	Minimum	Maximum
3	-0.8727	2.0377
4	-0.3134	1.3169
5.5	-0.001642	1.0004169
6	**0**	**1**

(a) Forward Euler.

1/CFL	Minimum	Maximum
3	-0.01077	1.0122
4	-8.271e-7	1
5.5	0	0.9999
6	**0**	**0.9999**

(b) RK2.

Table 6.2: Minimum and maximum cell averages for various CFL numbers for the first decomposition (Figure 6.6).

1/CFL	Minimum	Maximum
3	-0.9426	1.7499
4	-0.3151	1.3559
5.5	-0.02076	1.02133
6	**0**	**1**

(a) Forward Euler.

1/CFL	Minimum	Maximum
3	-0.25936	1.06160
4	-1.3463e-4	0.9999
5.5	0	0.9999
6	**0**	**0.9999**

(b) RK2.

Table 6.3: Minimum and maximum cell averages for various CFL numbers for the second decomposition (Figure 6.7).

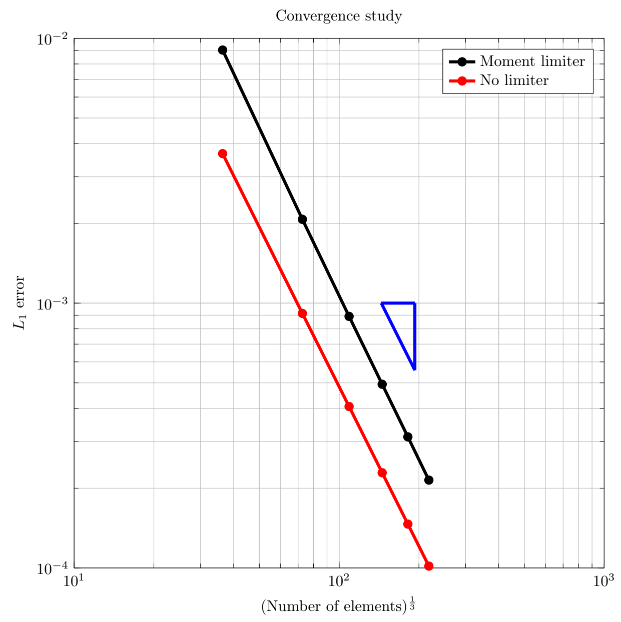


Figure 6.8: Convergence study in Example 6.4.2. The slope of the blue triangle's hypotenuse indicates second order accuracy.

6.4.2 Second order accuracy

In this example, we demonstrate the accuracy of our limiting procedure. We solve (1.1) with the linear flux $\mathbf{F}(u) = [u, u, u]$, corresponding to the advection direction $[1, 1, 1]$, with a bump as the initial condition given by

$$u_0(x, y, z) = \begin{cases} \cos^2\left(\frac{\pi}{2}r\right) & \text{if } r \leq 1, \\ 0 \text{ elsewhere,} \end{cases}$$

where $r = 2\sqrt{x^2 + y^2 + z^2}$ until the final time $T = 0.25$. The errors of the limited and unlimited solutions on a sequence of structured meshes using the first decomposition in Figure 6.6 (Section 6.4.1) is given in Figure 6.8. Both the limited and unlimited solutions converge with second order accuracy.

	$\mathbf{U}_Q$	$\mathbf{U}_B$	$\mathbf{U}_I$
ρ	1.4	0.5	8
u	0	0	-8.25
v	0	0	0
w	0	0	0
p	1	1	116.5

Table 6.4: Initial states for the quiescent, bubble, and incident shock states.

6.4.3 Shock-bubble interaction

In this example, we model the interaction between a Mach 10 shock and a bubble with a spherical geometry. The domain is given by $[-1, 1]^3$, where the spherical bubble of radius $R = 0.55$ is centered at the point $(-0.4, 0, 0)$ and incident shock is located on the plane $x = -0.95$. Due to the symmetry of the problem, we only need to model a quarter of the original domain, i.e. $[-1, 1] \times [0, 1]^2$. The mesh is composed of 12,000,000 tetrahedra by subdividing the quarter domain into a $100 \times 100 \times 200$ grid of cubes. Then, each cube is subdivided into six tetrahedra as shown in Figure 6.6. The initial condition is given by the incident, quiescent, and bubble states tabulated in Table 6.4 and are shown in Figure 6.9. The states inside and outside the bubble are given by $\mathbf{U}_B$ and $\mathbf{U}_Q$, respectively. The state of the gas to the right of the leftward moving Mach 10 incident shock is given by $\mathbf{U}_I$. On the boundaries defined by the planes $y = 0$ and $z = 0$, we use reflecting boundary conditions. Since the bubble does not interact with the other boundaries, we apply the exact solution of the propagating Mach 10 shock. The solution at different times between $t = 0$ and the final time $T = 0.125$ are shown in Figure 6.10, revealing a complex interaction that compresses the initial bubble.

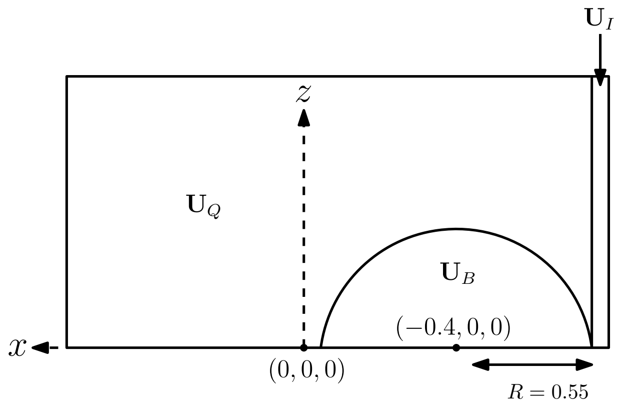
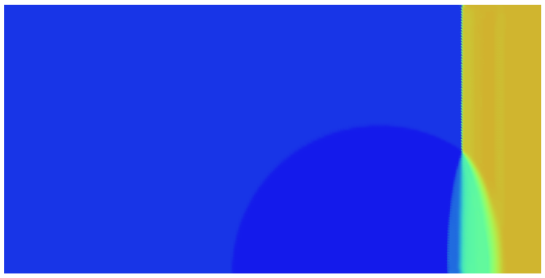
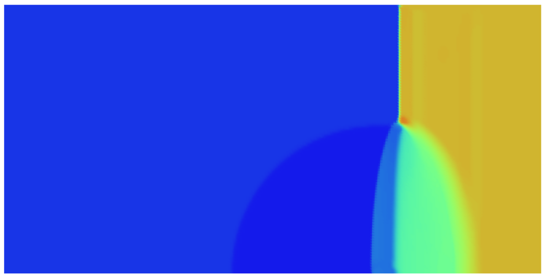
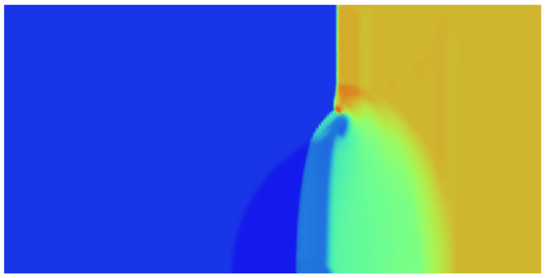
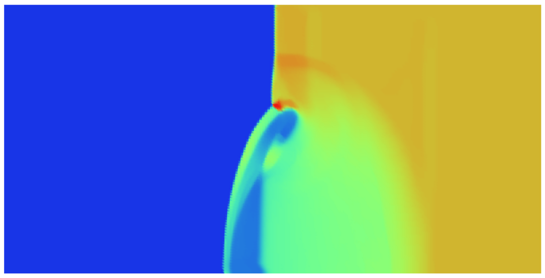


Figure 6.9: Initial set-up of the shock-bubble interaction problem. The incident shock is planar and moves towards the spherical bubble.


(a) $T = 0.0241$.


(b) $T = 0.0475$.


(c) $T = 0.0707$.


(d) $T = 0.0938$.

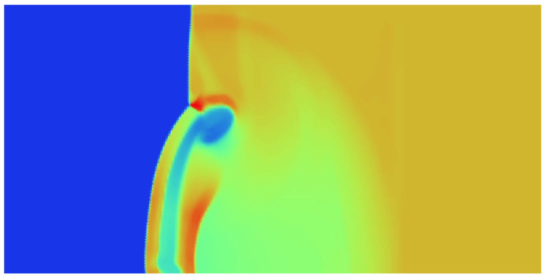

(e) $T = 0.125$.

Figure 6.10: Shock-bubble interaction for high Mach number ($M = 10$).

6.5 Summary

We have presented an extension of the moment limiter on unstructured triangles [34] to unstructured tetrahedra. Our analysis has revealed three directions in which the moments of the numerical solution can be reconstructed. We have determined constraints on the moments and on the time step such that a local maximum principle is satisfied. This limiting procedure is also extended to limiting across planar reflecting boundaries.

Similar to the moment limiter [34], this limiter has a stencil of constant size and does not require evaluating the numerical solution at the quadrature points. Additionally, it's implementation is lightweight since much of the overhead is moved to the mesh preprocessing stage.

Future work includes extending the moment limiter to support higher order basis functions, while keeping the same limiting stencil.

Chapter 7

Conclusion

In this thesis, we have presented efficient GPU-accelerated algorithms for adaptive mesh refinement and new limiters for the discontinuous Galerkin method on unstructured meshes. In Chapter 2, we presented an adaptive mesh refinement algorithm that was entirely implemented on the GPU. We described the novelties of the implementation, along with the computationally difficult problems in gas dynamics that we tackled. First, a coloring algorithm was used for work scheduling to avoid race conditions in the evaluation of an integral on the edges of the computational mesh. It also allowed for the parallelization of smoothing subroutines, which are not easily implemented on the GPU. Second, efficient compaction operations were also proposed that reduced the amount of required memory transfers to ensure that data was contiguous in memory. Third, we solved a number of benchmarks in gas dynamics to demonstrate the efficacy of the implementation. We also solved two less common problems concerning shock reflection that are intractable without some form of mesh adaptivity. The first problem concerned the von Neumann triple point paradox, which presents a reflection pattern that is on the order of 10^{-4} in the neighborhood of a triple point. In order to resolve the relevant flow features, we required elements with cell width on the order of 10^{-6}. Finally, we solved a problem regarding shock diffraction around a thin film and the location of shock disappearance. To our knowledge, our results are the first obtained on the Euler equations.

In Chapter 3, we studied slope limiters on unstructured meshes of triangles. The purpose of this work was to understand the proper usage of the popular Barth-Jespersen type limiters on triangles for the discontinuous Galerkin method. These limiters constrain the numerical solution at predefined points to lie in a locally defined range. We have studied the influence of the choice of limiting points and limiting neighborhoods on the stability and accuracy of the numerical solution. Our analysis also revealed a new measure of cell size that yields stable solutions and is approximately twice as large as the radius of the inscribed circle.

In Chapter 4, we showed numerically and analytically on a one-dimensional finite volume method with SSP-RK2 time stepping that the time step restriction for stability in the maximum norm can be increased from the SSP theory's prediction. We also have given numerical evidence that this result extends to two dimensions with the DG method.

In Chapters 5 and 6, we proposed and analyzed second order moment limiters for the DG method on unstructured triangles and tetrahedra, respectively. The limiting algorithms are simple to implement and computationally efficient, composed of one-dimensional minmod operations. These limiters have a number of attractive qualities. First, we have shown that the limited numerical solution satisfies a local maximum principle under a suitable time step restriction. Second, as opposed to other limiters on unstructured meshes, the limiters have a stencil of constant size. Third, they are easy to implement as the majority of the computational work is moved to the mesh preprocessing stage. Finally, they do not require the evaluation of the numerical solution at the quadrature points. We also describe a simple procedure to limit across reflecting boundary conditions. This is especially useful for three dimensional simulations, where reflecting boundary conditions can be used to take advantage of symmetries in the solution to reduce the amount of required computational work.

There are a number of future projects that can result from this thesis. First, extending the AMR algorithms to support multi-node, multi-GPU parallel platforms would be useful. This would allow us to better resolve Guderley Mach reflection and the location of shock disappearance in the problem of shock diffraction around thin films. Second, extending the AMR code to support unstructured tetrahedra would be of practical interest, where the multi-node, multi-GPU implementation would be all the more important. This would require extending the run-time coloring algorithm to tetrahedra, which appears to be difficult.

Extending the moment limiters to nonconforming meshes for use in adaptive simulations appears to be possible. An element's limiting directions will stay the same when its neighbors are refined or coarsened, though the elements used in the reconstruction of forward and backward differences may have to be updated. This will not require projecting the neighboring solutions onto coarser or finer elements as is sometimes done. Another possibility for future work is the extension of these limiters to higher order bases, though the analysis is more involved as there are a higher number of solution coefficients that must be limited. Finally, we plan on investigating the applicability of this limiter to computations involved in cut cell element geometries [36, 88].

References

[1] V. Springel, “E pur si muove: Galilean-invariant cosmological hydrodynamical simulations on a moving mesh,” *Monthly Notices of the Royal Astronomical Society*, vol. 401, no. 2, pp. 791–851, 2010.

[2] A. Giuliani and L. Krivodonova, “Analysis of slope limiters on unstructured triangular meshes,” *Journal of Computational Physics*, vol. 374, pp. 1 – 26, 2018.

[3] M. Fuhry, A. Giuliani, and L. Krivodonova, “Discontinuous Galerkin methods on graphics processing units for nonlinear hyperbolic conservation laws,” *International Journal for Numerical Methods in Fluids*, vol. 76, no. 12, pp. 982–1003, 2014.

[4] R. Gandham, D. Medina, and T. Warburton, “GPU accelerated discontinuous Galerkin methods for shallow water equations,” *Communications in Computational Physics*, vol. 18, no. 1, pp. 37–64, 2015.

[5] A. Karakus, T. Warburton, M. Aksel, and C. Sert, “A GPU-accelerated adaptive discontinuous Galerkin method for level set equation,” *International Journal of Computational Fluid Dynamics*, vol. 30, no. 1, pp. 56–68, 2016.

[6] J. Chan, Z. Wang, A. Modave, J.-F. Remacle, and T. Warburton, “GPU-accelerated discontinuous Galerkin methods on hybrid meshes,” *Journal of Computational Physics*, vol. 318, pp. 142–168, 2016.

[7] A. Giuliani and L. Krivodonova, “Face coloring in unstructured CFD codes,” *Parallel Computing*, vol. 63, pp. 17–37, 2017.

[8] H. Hoteit, P. Ackerer, R. Mosé, J. Erhel, and B. Philippe, “New two-dimensional slope limiters for discontinuous Galerkin methods on arbitrary meshes,” *International Journal for Numerical Methods in Engineering*, vol. 61, no. 14, pp. 2566–2593, 2004.

[9] 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,” *Mathematics of Computation*, vol. 54, no. 190, pp. 545–581, 1990.

[10] R. Biswas, K. D. Devine, and J. E. Flaherty, “Parallel, adaptive finite element methods for conservation laws,” *Applied Numerical Mathematics*, vol. 14, no. 1-3, pp. 255–283, 1994.

[11] L. Krivodonova, “Limiters for high-order discontinuous Galerkin methods,” *Journal of Computational Physics*, vol. 226, no. 1, pp. 879–896, 2007.

[12] M. Dubiner, “Spectral methods on triangles and other domains,” *Journal of Scientific Computing*, vol. 6, pp. 345–390, Dec 1991.

[13] J. S. Hesthaven and T. Warburton, *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications.* Springer Publishing Company, Incorporated, 1st ed., 2007.

[14] C. Pain, A. Umpleby, C. De Oliveira, and A. Goddard, “Tetrahedral mesh optimisation and adaptivity for steady-state and transient finite element calculations,” *Computer Methods in Applied Mechanics and Engineering*, vol. 190, no. 29-30, pp. 3771–3796, 2001.

[15] S. M. Schnepp and T. Weiland, “Efficient large scale electromagnetic simulations using dynamically adapted meshes with the discontinuous Galerkin method,” *Journal of Computational and Applied Mathematics*, vol. 236, no. 18, pp. 4909–4924, 2012.

[16] C. Eskilsson, “An hp-adaptive discontinuous Galerkin method for shallow water flows,” *International Journal for Numerical Methods in Fluids*, vol. 67, no. 11, pp. 1605–1623, 2011.

[17] P. MacNeice, K. M. Olson, C. Mobarry, R. De Fainchtein, and C. Packer, “PARAMESH: A parallel adaptive mesh refinement community toolkit,” *Computer physics communications*, vol. 126, no. 3, pp. 330–354, 2000.

[18] M. Adams, “CHOMBO software package for AMR applications–design document, Lawrence Berkeley National Laboratory technical report lbnl-6616e,” 2011.

[19] W. Bangerth, R. Hartmann, and G. Kanschat, “deal.II – a general purpose object oriented finite element library,” *ACM Trans. Math. Softw.*, vol. 33, no. 4, pp. 24/1–24/27, 2007.

[20] K. T. Mandli, A. J. Ahmadia, M. Berger, D. Calhoun, D. L. George, Y. Hadjimichael, D. I. Ketcheson, G. I. Lemoine, and R. J. LeVeque, "Clawpack: building an open source ecosystem for solving hyperbolic PDEs," *PeerJ Computer Science*, vol. 2, p. e68, 2016.

[21] M. J. Berger and J. Oliger, "Adaptive mesh refinement for hyperbolic partial differential equations," *Journal of Computational Physics*, vol. 53, no. 3, pp. 484 – 512, 1984.

[22] J. Z. X. Zheng, *Block-based adaptive mesh refinement finite-volume scheme for hybrid multi-block meshes*. PhD thesis, University of Toronto, 2012.

[23] L. Ivan, H. D. Sterck, S. A. Northrup, and C. Groth, "Multi-dimensional finite-volume scheme for hyperbolic conservation laws on three-dimensional solution-adaptive cubed-sphere grids," *Journal of Computational Physics*, vol. 255, no. 0, pp. 205 – 227, 2013.

[24] C. Burstedde, L. C. Wilcox, and O. Ghattas, "p4est: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees," *SIAM Journal on Scientific Computing*, vol. 33, no. 3, pp. 1103–1133, 2011.

[25] K. D. Devine and J. E. Flaherty, "Parallel adaptive hp-refinement techniques for conservation laws," *Applied Numerical Mathematics*, vol. 20, no. 4, pp. 367–386, 1996.

[26] T. Richter, *Parallel multigrid method for adaptive finite elements with application to 3D flow problems*. PhD thesis, University of Heidelberg, 2005.

[27] O. Zanotti and M. Dumbser, "A high order special relativistic hydrodynamic and magnetohydrodynamic code with space-time adaptive mesh refinement," *Computer Physics Communications*, vol. 188, pp. 110–127, 2015.

[28] K. G. Guderley, *Considerations on the structure of mixed subsonic-supersonic flow patterns*. Headquarters Air Materiel Command, 1947.

[29] A. M. Tesdall, R. Sanders, and B. L. Keyfitz, "Self-similar solutions for the triple point paradox in gasdynamics," *SIAM Journal on Applied Mathematics*, vol. 68, no. 5, pp. 1360–1377, 2008.

[30] E. I. Vasilev and A. N. Kraiko, "Numerical simulation of weak shock diffraction over a wedge under the von Neumann paradox conditions," *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, vol. 39, no. 8, pp. 1393–1404, 1999.

[31] B. W. Skews and J. T. Ashworth, "The physical nature of weak shock wave reflection," *Journal of Fluid Mechanics*, vol. 542, pp. 105–114, 2005.

[32] P. K. Sweby, "High resolution schemes using flux limiters for hyperbolic conservation laws," *SIAM Journal on Numerical Analysis*, vol. 21, no. 5, pp. 995–1011, 1984.

[33] J. Goodman and R. LeVeque, "On the accuracy of TVD schemes in two space dimensions," *Math. Comp*, vol. 45, pp. 15–21, 1985.

[34] A. Giuliani and L. Krivodonova, "A moment limiter for the discontinuous Galerkin method on unstructured meshes of triangles," *Draft available at http://www.math.uwaterloo.ca/~lgk/moment.pdf*, 2017.

[35] T. Barth and D. Jespersen, "The design and application of upwind schemes on unstructured meshes," *AIAA paper*, pp. 89–0366, 1989.

[36] S. May and M. Berger, "Two-dimensional slope limiters for finite volume schemes on non-coordinate-aligned meshes," *SIAM Journal on Scientific Computing*, vol. 35, no. 5, pp. A2163–A2187, 2013.

[37] X. Zhang and C.-W. Shu, "A minimum entropy principle of high order schemes for gas dynamics equations," *Numerische Mathematik*, vol. 121, no. 3, pp. 545–563, 2012.

[38] X. Zhang and C.-W. Shu, "Positivity-preserving high order discontinuous Galerkin schemes for compressible Euler equations with source terms," *Journal of Computational Physics*, vol. 230, no. 4, pp. 1238–1248, 2011.

[39] X. Zhang and C.-W. Shu, "On positivity-preserving high order discontinuous Galerkin schemes for compressible Euler equations on rectangular meshes," *Journal of Computational Physics*, vol. 229, no. 23, pp. 8918–8934, 2010.

[40] Y. Lv and M. Ihme, "Entropy-bounded discontinuous Galerkin scheme for Euler equations," *Journal of Computational Physics*, vol. 295, pp. 715–739, 2015.

[41] G. Rokos, G. Gorman, and P. H. Kelly, "Accelerating anisotropic mesh adaptivity on nVIDIAs CUDA using texture interpolation," in *European Conference on Parallel Processing*, pp. 387–398, Springer, 2011.

[42] Y. Xia, J. Lou, H. Luo, J. Edwards, and F. Mueller, "OpenACC acceleration of an unstructured CFD solver based on a reconstructed discontinuous Galerkin method for compressible flows," *International Journal for Numerical Methods in Fluids*, vol. 78, no. 3, pp. 123–139, 2015.

[43] D. Merrill and NVIDIA-Labs, "CUDA UnBound (CUB) Library."

[44] K. Schaal, A. Bauer, P. Chandrashekar, R. Pakmor, C. Klingenberg, and V. Springel, "Astrophysical hydrodynamics with a high-order discontinuous Galerkin scheme and adaptive mesh refinement," *Monthly Notices of the Royal Astronomical Society*, vol. 453, no. 4, pp. 4278–4300, 2015.

[45] P. E. Vincent, P. Castonguay, and A. Jameson, "Insights from von Neumann analysis of high-order flux reconstruction schemes," *Journal of Computational Physics*, vol. 230, no. 22, pp. 8134–8154, 2011.

[46] N. Chalmers and L. Krivodonova, "A robust CFL condition for the discontinuous Galerkin method on triangular meshes," *Draft available at http://www.math.uwaterloo.ca/~lgk/A_Characteristic_Based_CFL_Number.pdf*, 2018.

[47] P. Woodward and P. Colella, "The numerical simulation of two-dimensional fluid flow with strong shocks," *Journal of Computational Physics*, vol. 54, no. 1, pp. 115–173, 1984.

[48] A. Zakharian, M. Brio, J. Hunter, and G. Webb, "The von Neumann paradox in weak shock reflection," *Journal of Fluid Mechanics*, vol. 422, pp. 193–205, 2000.

[49] A. M. Tesdall, R. Sanders, and N. Popivanov, "Further results on Guderley Mach reflection and the triple point paradox," *Journal of Scientific Computing*, vol. 64, no. 3, pp. 721–744, 2015.

[50] A. M. Tesdall and J. K. Hunter, "Self-similar solutions for the diffraction of weak shocks," *Journal of Computational Science*, vol. 4, no. 1-2, pp. 92–100, 2013.

[51] A. Harten, "High resolution schemes for hyperbolic conservation laws," *Journal of Computational Physics*, vol. 49, no. 3, pp. 357–393, 1983.

[52] A. Harten, "On a class of high resolution total-variation-stable finite-difference schemes," *SIAM Journal on Numerical Analysis*, vol. 21, no. 1, pp. 1–23, 1984.

[53] E. Tadmor, "Convenient total variation diminishing conditions for nonlinear difference schemes," *SIAM Journal on Numerical Analysis*, vol. 25, no. 5, pp. 1002–1014, 1988.

[54] J. B. Goodman and R. J. LeVeque, "On the accuracy of stable schemes for 2D scalar conservation laws," *Mathematics of Computation*, pp. 15–21, 1985.

[55] S. Spekreijse, “Multigrid solution of monotone second-order discretizations of hyperbolic conservation laws,” *Mathematics of Computation*, vol. 49, no. 179, pp. 135–155, 1987.

[56] P. Batten, C. Lambert, and D. Causon, “Positively conservative high-resolution convection schemes for unstructured elements,” *International Journal for Numerical Methods in Engineering*, vol. 39, no. 11, pp. 1821–1838, 1996.

[57] J. S. Park, S.-H. Yoon, and C. Kim, “Multi-dimensional limiting process for hyperbolic conservation laws on unstructured grids,” *Journal of Computational Physics*, vol. 229, no. 3, pp. 788–812, 2010.

[58] J. S. Park and C. Kim, “Multi-dimensional limiting process for finite volume methods on unstructured grids,” *Computers & Fluids*, vol. 65, pp. 8–24, 2012.

[59] D. Kuzmin, “Slope limiting for discontinuous Galerkin approximations with a possibly non-orthogonal Taylor basis,” *International Journal for Numerical Methods in Fluids*, vol. 71, no. 9, pp. 1178–1190, 2013.

[60] T. Buffard and S. Clain, “Monoslope and multislope MUSCL methods for unstructured meshes,” *Journal of Computational Physics*, vol. 229, no. 10, pp. 3745–3776, 2010.

[61] C. Le Touze, A. Murrone, and H. Guillard, “Multislope MUSCL method for general unstructured meshes,” *Journal of Computational Physics*, vol. 284, pp. 389–418, 2015.

[62] B. Cockburn and C.-W. Shu, “TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws. II. General framework,” *Mathematics of Computation*, vol. 52, no. 186, pp. 411–435, 1989.

[63] B. Cockburn and C.-W. Shu, “The Runge-Kutta discontinuous Galerkin method for conservation laws V: multidimensional systems,” *Journal of Computational Physics*, vol. 141, no. 2, pp. 199–224, 1998.

[64] M. Yang and Z.-J. Wang, “A parameter-free generalized moment limiter for high-order methods on unstructured grids,” *Adv. Appl. Math. Mech*, vol. 1, no. 4, pp. 451–480, 2009.

[65] J. S. Park and C. Kim, “Higher-order multi-dimensional limiting strategy for discontinuous Galerkin methods in compressible inviscid and viscous flows,” *Computers & Fluids*, vol. 96, pp. 377–396, 2014.

[66] M. Dumbser, O. Zanotti, R. Loubère, and S. Diot, "A posteriori subcell limiting of the discontinuous Galerkin finite element method for hyperbolic conservation laws," *Journal of Computational Physics*, vol. 278, pp. 47–75, 2014.

[67] T. Barth and M. Ohlberger, "Finite volume methods: foundation and analysis," *Encyclopedia of Computational Mechanics*, 2004.

[68] B. Swartz, "Good neighborhoods for multidimensional van Leer limiting," *Journal of Computational Physics*, vol. 154, no. 1, pp. 237–241, 1999.

[69] V. Aizinger, A. Kosík, D. Kuzmin, and B. Reuter, "Anisotropic slope limiting for discontinuous Galerkin methods," *International Journal for Numerical Methods in Fluids*, 2017.

[70] S. Gottlieb, "On high order strong stability preserving Runge-Kutta and multi step time discretizations," *Journal of Scientific Computing*, vol. 25, no. 1, pp. 105–128, 2005.

[71] L. Krivodonova and R. Qin, "An analysis of the spectrum of the discontinuous Galerkin method," *Applied Numerical Mathematics*, vol. 64, no. Supplement C, pp. 1 – 18, 2013.

[72] L. Krivodonova and R. Qin, "An analysis of the spectrum of the discontinuous Galerkin method II: Nonuniform grids," *Applied Numerical Mathematics*, vol. 71, pp. 41–62, 2013.

[73] S. Gottlieb, D. Ketcheson, and C.-W. Shu, *Strong Stability Preserving Runge-Kutta and Multistep Time Discretizations*. River Edge, NJ, USA: World Scientific Publishing Co., Inc., 2011.

[74] S. Gottlieb, C.-W. Shu, and E. Tadmor, "Strong stability-preserving high-order time discretization methods," *SIAM review*, vol. 43, no. 1, pp. 89–112, 2001.

[75] C.-W. Shu, "Total-variation-diminishing time discretizations," *SIAM J. Sci. Stat. Comput.*, vol. 9, pp. 1073–1084, Nov. 1988.

[76] C.-W. Shu and S. Osher, "Efficient implementation of essentially non-oscillatory shock-capturing schemes," *Journal of Computational Physics*, vol. 77, no. 2, pp. 439 – 471, 1988.

[77] S. Gottlieb and C.-W. Shu, “Total variation diminishing Runge-Kutta schemes,” *Mathematics of computation of the American Mathematical Society*, vol. 67, no. 221, pp. 73–85, 1998.

[78] Z. Horvàth, “On the positivity step size threshold of Runge-Kutta methods,” *Applied Numerical Mathematics*, vol. 53, no. 2, pp. 341 – 356, 2005. Tenth Seminar on Numerical Solution of Differential and Differntial-Algebraic Euqations (NUMDIFF-10).

[79] Z. Horvàth, “On the positivity of matrix-vector products,” *Linear Algebra and its Applications*, vol. 393, pp. 253 – 258, 2004. Special Issue on Positivity in Linear Algebra.

[80] I. Higueras, “Monotonicity for Runge-Kutta methods: Inner product norms,” *Journal of Scientific Computing*, vol. 24, pp. 97–117, Jul 2005.

[81] R. LeVeque, *Finite volume methods for hyperbolic conservation laws.* New York: Cambridge University Press, Cambridge, 2002.

[82] M. Berger, M. Aftosmis, and S. Murman, “Analysis of slope limiters on irregular grids,” in *43rd AIAA Aerospace Sciences Meeting and Exhibit*, p. 490, 2005.

[83] B. van Leer, “Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov’s method,” *Journal of Computational Physics*, vol. 32, no. 1, pp. 101 – 136, 1979.

[84] B. Van Leer, “Towards the ultimate conservative difference scheme. IV. A new approach to numerical convection,” *Journal of Computational Physics*, vol. 23, no. 3, pp. 276–299, 1977.

[85] B. Cockburn, S.-Y. Lin, and C.-W. Shu, “TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws III: one-dimensional systems,” *Journal of Computational Physics*, vol. 84, no. 1, pp. 90–113, 1989.

[86] S. Karni and S. Čanić, “Computations of slowly moving shocks,” *Journal of Computational Physics*, vol. 136, no. 1, pp. 132–139, 1997.

[87] N. Chalmers, *Superconvergence, Superaccuracy, and Stability of the Discontinuous Galerkin Finite Element Method.* PhD thesis, University of Waterloo, 2015.

[88] R. Qin and L. Krivodonova, “A discontinuous Galerkin method for solutions of the Euler equations on Cartesian grids with embedded geometries,” *Journal of Computational Science*, vol. 4, no. 1, pp. 24–35, 2013.

[89] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The quickhull algorithm for convex hulls," *ACM Transactions on Mathematical Software (TOMS)*, vol. 22, no. 4, pp. 469–483, 1996.

[90] A. Giuliani and L. Krivodonova, "On the optimal CFL number of SSP methods for hyperbolic problems," *Applied Numerical Mathematics*, 2018.

Appendix A

Proof of proposition 1

For simplicity of discussion, we translate the element Ω_i such that its centroid is located at the origin. The vectors pointing from the origin to the three vertices of the triangle are $\mathbf{v}_{i,1}$, $\mathbf{v}_{i,2}$, and $\mathbf{v}_{i,3}$. The vectors $\boldsymbol{\xi}_1$, $\boldsymbol{\xi}_2$, and $\boldsymbol{\xi}_3$, pointing from the origin to the one- or two-point Gauss-Legendre quadrature points can be written as

$$\begin{aligned}\boldsymbol{\xi}_1 &= \epsilon\mathbf{v}_{i,2} + (1-\epsilon)\mathbf{v}_{i,3},\\ \boldsymbol{\xi}_2 &= \epsilon\mathbf{v}_{i,3} + (1-\epsilon)\mathbf{v}_{i,1},\\ \boldsymbol{\xi}_3 &= \epsilon\mathbf{v}_{i,1} + (1-\epsilon)\mathbf{v}_{i,2},\end{aligned}$$

where $\epsilon = \frac{1}{2}$ for the one-point rule, and $\epsilon = \frac{1}{2} \pm \frac{\sqrt{3}}{6}$ for the two-point rule. Let $\hat{\boldsymbol{\xi}}_{23,\perp}$ be the unit vector that is perpendicular to the vector $\boldsymbol{\xi}_2 - \boldsymbol{\xi}_3$, and that is pointing toward $\boldsymbol{\xi}_1$, i.e., $\boldsymbol{\xi}_1 \cdot \hat{\boldsymbol{\xi}}_{23,\perp} > 0$. Let $\hat{\boldsymbol{\xi}}_{1,\perp}$ be the unit vector that is perpendicular to $\boldsymbol{\xi}_1$, and that is pointing towards $\boldsymbol{\xi}_2$, i.e. $\hat{\boldsymbol{\xi}}_{1,\perp} \cdot \boldsymbol{\xi}_2 > 0$ (Figure A.1).

Because the centroid of Ω_i has been translated to the origin, we have

$$\boldsymbol{\xi}_1 + \boldsymbol{\xi}_2 + \boldsymbol{\xi}_3 = \mathbf{0}, \tag{A.1}$$

which gives

$$\boldsymbol{\xi}_2 - \boldsymbol{\xi}_3 = -\boldsymbol{\xi}_1 - 2\boldsymbol{\xi}_3 = \boldsymbol{\xi}_1 + 2\boldsymbol{\xi}_2. \tag{A.2}$$

Multiplying (A.2) by $\boldsymbol{\xi}_{23,\perp}$, we have

$$0 = -\boldsymbol{\xi}_1 \cdot \hat{\boldsymbol{\xi}}_{23,\perp} - 2\boldsymbol{\xi}_3 \cdot \hat{\boldsymbol{\xi}}_{23,\perp} = \boldsymbol{\xi}_1 \cdot \hat{\boldsymbol{\xi}}_{23,\perp} + 2\boldsymbol{\xi}_2 \cdot \hat{\boldsymbol{\xi}}_{23,\perp}. \tag{A.3}$$

Rearranging terms gives the following relations

$$\boldsymbol{\xi}_1 \cdot \hat{\boldsymbol{\xi}}_{23,\perp} = -2\boldsymbol{\xi}_2 \cdot \hat{\boldsymbol{\xi}}_{23,\perp} = -2\boldsymbol{\xi}_3 \cdot \hat{\boldsymbol{\xi}}_{23,\perp}. \tag{A.4}$$

We now use this to prove proposition 1, which we restate here.

Proposition 1. *For a quadrature point* $\mathbf{x}$*, there exists a multiplier* $0 \le r \le 2$ *and another quadrature point* $\mathbf{x}'$ *on a different edge, such that*

$$U_i(\mathbf{x}) - \overline{U}_i = r(\overline{U}_i - U_i(\mathbf{x}')).$$

Proof. Without loss of generality, let us assume that $\mathbf{x}$ in the Proposition is $\boldsymbol{\xi}_1$ in Figure A.1. We will show that for a given gradient of numerical solution U_i, $\mathbf{g}$, there is a different quadrature point $\mathbf{x}'$ with $0 \le r \le 2$.

From the definition of $\hat{\boldsymbol{\xi}}_{23,\perp}$, $\boldsymbol{\xi}_1 \cdot \hat{\boldsymbol{\xi}}_{23,\perp} > 0$ (Figure A.1) and from (A.4) we have that $\boldsymbol{\xi}_2 \cdot \hat{\boldsymbol{\xi}}_{23,\perp} < 0$ and $\boldsymbol{\xi}_3 \cdot \hat{\boldsymbol{\xi}}_{23,\perp} < 0$. Additionally, taking the dot product of (A.1) and $\hat{\boldsymbol{\xi}}_{1,\perp}$, we have $\boldsymbol{\xi}_2 \cdot \hat{\boldsymbol{\xi}}_{1,\perp} = -\boldsymbol{\xi}_3 \cdot \hat{\boldsymbol{\xi}}_{1,\perp}$. Therefore $\boldsymbol{\xi}_2 \cdot \hat{\boldsymbol{\xi}}_{1,\perp}$ and $\boldsymbol{\xi}_3 \cdot \hat{\boldsymbol{\xi}}_{1,\perp}$ are of opposite sign. By definition of $\hat{\boldsymbol{\xi}}_{1,\perp}$, we have $\boldsymbol{\xi}_2 \cdot \hat{\boldsymbol{\xi}}_{1,\perp} > 0$, which implies that $\boldsymbol{\xi}_3 \cdot \hat{\boldsymbol{\xi}}_{1,\perp} < 0$.

Now, if the vector $\mathbf{g}$ lies between $\hat{\boldsymbol{\xi}}_{23,\perp}$ and $\hat{\boldsymbol{\xi}}_{1,\perp}$, i.e. in the region that we denote by (I) in Figure A.1, then we can write $\mathbf{g}$ as

$$\mathbf{g} = c_1\hat{\boldsymbol{\xi}}_{23,\perp} + c_2\hat{\boldsymbol{\xi}}_{1,\perp} \text{ with } c_1, c_2 > 0. \tag{A.5}$$

Recalling (A.4) and the definition of $\hat{\boldsymbol{\xi}}_{1,\perp}$, we have

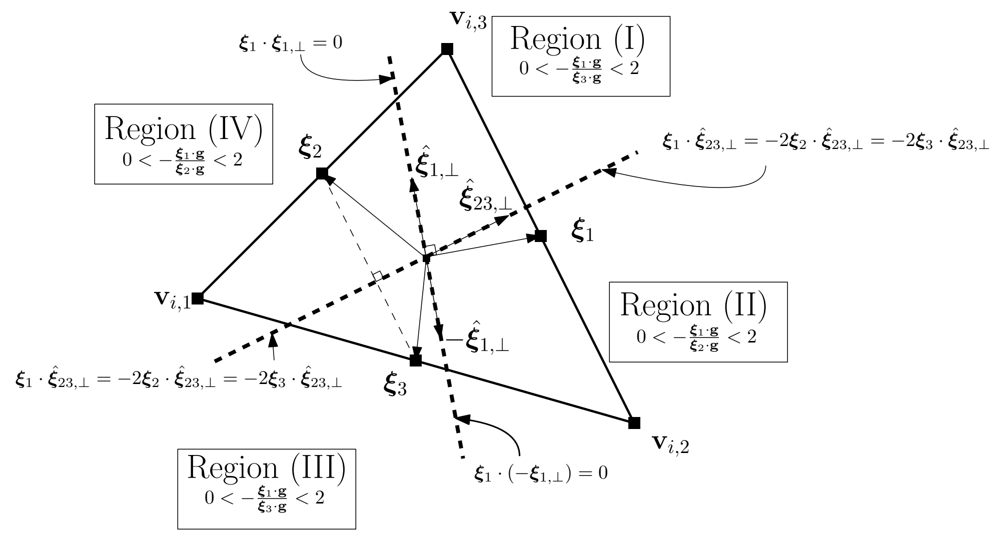
$$\begin{cases} \boldsymbol{\xi}_1 \cdot \hat{\boldsymbol{\xi}}_{23,\perp} = -2\boldsymbol{\xi}_3 \cdot \hat{\boldsymbol{\xi}}_{23,\perp}, \\ \boldsymbol{\xi}_1 \cdot \hat{\boldsymbol{\xi}}_{1,\perp} = 0. \end{cases} \tag{A.6}$$

Multiplying the first line in (A.6) by c_1 and the second by c_2, then summing, we have by (A.5)

$$\boldsymbol{\xi}_1 \cdot (c_1\hat{\boldsymbol{\xi}}_{23,\perp} + c_2\hat{\boldsymbol{\xi}}_{1,\perp}) = \boldsymbol{\xi}_1 \cdot \mathbf{g} = -2\boldsymbol{\xi}_3 \cdot (c_1\hat{\boldsymbol{\xi}}_{23,\perp}). \tag{A.7}$$

Since $\boldsymbol{\xi}_3 \cdot \hat{\boldsymbol{\xi}}_{23,\perp} < 0$, we have $\boldsymbol{\xi}_1 \cdot \mathbf{g} > 0$ by (A.7). Further, because $\boldsymbol{\xi}_3 \cdot \hat{\boldsymbol{\xi}}_{1,\perp} < 0$, the last term of (A.7) can be bounded below and above by

$$0 < \boldsymbol{\xi}_1 \cdot \mathbf{g} = -2\boldsymbol{\xi}_3 \cdot (c_1\hat{\boldsymbol{\xi}}_{23,\perp}) < -2\boldsymbol{\xi}_3 \cdot (c_1\hat{\boldsymbol{\xi}}_{23,\perp} + c_2\hat{\boldsymbol{\xi}}_{1,\perp})$$



$\mathbf{v}_{i,3}$
Region (I) $0 < -\frac{\boldsymbol{\xi}_1\cdot\mathbf{g}}{\boldsymbol{\xi}_3\cdot\mathbf{g}} < 2$
$\boldsymbol{\xi}_1 \cdot \boldsymbol{\xi}_{1,\perp} = 0$
Region (IV) $0 < -\frac{\boldsymbol{\xi}_1\cdot\mathbf{g}}{\boldsymbol{\xi}_2\cdot\mathbf{g}} < 2$
$\boldsymbol{\xi}_2$
$\hat{\boldsymbol{\xi}}_{1,\perp}$
$\hat{\boldsymbol{\xi}}_{23,\perp}$
$\boldsymbol{\xi}_1 \cdot \hat{\boldsymbol{\xi}}_{23,\perp} = -2\boldsymbol{\xi}_2 \cdot \hat{\boldsymbol{\xi}}_{23,\perp} = -2\boldsymbol{\xi}_3 \cdot \hat{\boldsymbol{\xi}}_{23,\perp}$
$\boldsymbol{\xi}_1$
$\mathbf{v}_{i,1}$
Region (II) $0 < -\frac{\boldsymbol{\xi}_1\cdot\mathbf{g}}{\boldsymbol{\xi}_2\cdot\mathbf{g}} < 2$
$-\hat{\boldsymbol{\xi}}_{1,\perp}$
$\boldsymbol{\xi}_1 \cdot \hat{\boldsymbol{\xi}}_{23,\perp} = -2\boldsymbol{\xi}_2 \cdot \hat{\boldsymbol{\xi}}_{23,\perp} = -2\boldsymbol{\xi}_3 \cdot \hat{\boldsymbol{\xi}}_{23,\perp}$
$\boldsymbol{\xi}_3$
$\mathbf{v}_{i,2}$
Region (III) $0 < -\frac{\boldsymbol{\xi}_1\cdot\mathbf{g}}{\boldsymbol{\xi}_3\cdot\mathbf{g}} < 2$
$\boldsymbol{\xi}_1 \cdot (-\boldsymbol{\xi}_{1,\perp}) = 0$

Figure A.1: Geometric set-up for proposition 1.

i.e.

$$0 < \boldsymbol{\xi}_1 \cdot \mathbf{g} < -2\boldsymbol{\xi}_3 \cdot \mathbf{g}.$$

Therefore, for $\mathbf{g}$ in region (I)

$$0 < -\frac{\boldsymbol{\xi}_1 \cdot \mathbf{g}}{\boldsymbol{\xi}_3 \cdot \mathbf{g}} < 2.$$

From this we can conclude

$$0 < \frac{U_i(\mathbf{x}_1) - \overline{U}_i}{\overline{U}_i - U_i(\mathbf{x}_3)} < 2.$$

Recognizing that $r = -\frac{\boldsymbol{\xi}_1\cdot\mathbf{g}}{\boldsymbol{\xi}_3\cdot\mathbf{g}}$, we have the bounds $0 \leq r \leq 2$ and that $\mathbf{x}' = \boldsymbol{\xi}_3$.

This also holds for vectors in Region (III), in particular, $-\mathbf{g}$. This can be shown by multiplying the numerator and denominator by -1:

$$0 < -\frac{\boldsymbol{\xi}_1 \cdot (-\mathbf{g})}{\boldsymbol{\xi}_3 \cdot (-\mathbf{g})} < 2.$$

If $\mathbf{g}$ lies in region (II) or (IV), the same arguments can be made for the ratio $r = -\frac{\boldsymbol{\xi}_1\cdot\mathbf{g}}{\boldsymbol{\xi}_2\cdot\mathbf{g}}$, i.e. $\mathbf{x}' = \mathbf{x}_2$ and $0 \leq r \leq 2$.

□

Appendix B

Nonlinear Fluxes

For nonlinear fluxes, the RK-DG method requires third order accuracy to evaluate surface integrals in (5.2). With Gauss-Legendre numerical integration, we will need two quadrature points. We will modify our analysis to account for this. Most of the derivation follows the linear case.

Using the divergence theorem with (3.6), we obtain

$$\overline{U}_i^{n+1} = \overline{U}_i^n - \Delta t \sum_{j \in N_i^e, j \neq i} \frac{1}{2} \frac{|\partial \Omega_{i,j}|}{|\Omega_i|} \sum_{q=1,2} \left[\mathbf{F}^*(U_i^n(\mathbf{x}_{i,j,q}), U_j^n(\mathbf{x}_{i,j,q})) - \mathbf{F}^*(\overline{U}_i^n, \overline{U}_i^n) \right] \cdot \mathbf{n}_{i,j}.$$

Then, adding and subtracting $\mathbf{F}^*(U_i^n(\mathbf{x}_{i,j,q}), \overline{U}_i) \cdot \mathbf{n}_{i,j}$ in the inner sum, we have

$$\begin{aligned} \overline{U}_i^{n+1} = \overline{U}_i^n - \Delta t \sum_{j \in N_i^e, j \neq i} \sum_{q=1,2} \frac{1}{2} \frac{|\partial \Omega_{i,j}|}{|\Omega_i|} \Big[& F_{i,j}(U_i^n(\mathbf{x}_{i,j,q}), \overline{U}_i^n) - F_{i,j}(\overline{U}_i^n, \overline{U}_i^n) \\ & + F_{i,j}(U_i^n(\mathbf{x}_{i,j,q}), U_j^n(\mathbf{x}_{i,j,q})) - F_{i,j}(U_i^n(\mathbf{x}_{i,j,q}), \overline{U}_i^n) \Big], \end{aligned}$$

where $F_{i,j}(U_1, U_2) = \mathbf{F}(U_1, U_2) \cdot \mathbf{n}_{i,j}$. For $c_{i,j,q}^*$ belonging to the interval defined by $\overline{U}_i^n$ and $U_i^n(\mathbf{x}_{i,j,q})$, and $c_{i,j,q}^{**}$ belonging to the interval defined by $\overline{U}_i^n$ and $U_j^n(\mathbf{x}_{i,j,q})$, we have by the mean value theorem

$$\overline{U}_i^{n+1} = \overline{U}_i^n - \Delta t \sum_{j\in N_i^e, j\neq i} \sum_{q=1,2} \frac{1}{2} \frac{|\partial\Omega_{i,j}|}{|\Omega_i|} \Bigg[\frac{\partial F_{i,j}}{\partial U_1}(c^*_{i,j,q}, \overline{U}_i^n)(U_i^n(\mathbf{x}_{i,j,q}) - \overline{U}_i^n) + \frac{\partial F_{i,j}}{\partial U_2}(U_i^n(\mathbf{x}_{i,j,q}), c^{**}_{i,j,q})(U_j^n(\mathbf{x}_{i,j,q}) - \overline{U}_i^n) \Bigg],$$

where $\frac{\partial F_{i,j}}{\partial U_1}$ and $\frac{\partial F_{i,j}}{\partial U_2}$ are the partial derivatives with respect to the first and second variables. We introduce the following coefficients

$$v^+_{i,j,q} = \Delta t \frac{|\partial\Omega_{i,j}|}{|\Omega_i|} \frac{\partial F_{i,j}}{\partial U_1}(c^*_{i,j,q}, \overline{U}_i^n) \quad \text{and} \quad v^-_{j,i,q} = -\Delta t \frac{|\partial\Omega_{i,j}|}{|\Omega_i|} \frac{\partial F_{i,j}}{\partial U_2}(U_i^n(\mathbf{x}_{i,j,q}), c^{**}_{i,j,q}), \quad (B.1)$$

that are non-negative by the assumed monotonicity of the flux. We now have

$$\overline{U}_i^{n+1} = \overline{U}_i^n + \sum_{j\in N_i^e, j\neq i} \sum_{q=1,2} \frac{1}{2} \left[v^-_{j,i,q}(U_j^n(\mathbf{x}_{i,j,q}) - \overline{U}_i^n) - v^+_{i,j,q}(U_i^n(\mathbf{x}_{i,j,q}) - \overline{U}_i^n) \right]. \quad (B.2)$$

After limiting, the solution average at time t^{n+1} is written

$$\overline{U}_i^{n+1} = \overline{U}_i^n + \sum_{j\in N_i^e, j\neq i} \sum_{q=1,2} \frac{1}{2} \left[v^-_{j,i,q}(\tilde{U}_j^n(\mathbf{x}_{i,j,q}) - \overline{U}_i^n) - v^+_{i,j,q}(\tilde{U}_i^n(\mathbf{x}_{i,j,q}) - \overline{U}_i^n) \right], \quad (B.3)$$

where the values $v^-_{j,i,q}$ and $v^+_{j,i,q}$ have been updated using (B.1), with $\tilde{U}_i^n(\mathbf{x}_{i,j,q})$ and $\tilde{U}_j^n(\mathbf{x}_{i,j,q})$. Finally, $\tilde{U}_i^n(\mathbf{x}_{i,j,q})$, and $\tilde{U}_j^n(\mathbf{x}_{i,j,q})$ are the limited numerical solutions on Ω_i and Ω_j, respectively. Following the linear case, we aim to rewrite the sums in (B.3) in the form

$$\sum_{q=1,2} \frac{1}{2} v^-_{j,i,q}(\tilde{U}_j^n(\mathbf{x}_{i,j,q}) - \overline{U}_i^n) = v^-_{j,i} \left[f_{j,i}(\overline{U}_j^n - \overline{U}_i^n) + f_{j,i,1}(U^-_{j,i,1} - \overline{U}_i^n) + f_{j,i,2}(U^-_{j,i,2} - \overline{U}_i^n) \right] \quad (B.4)$$

for the inflow term and

$$\sum_{q=1,2} \frac{1}{2} v^+_{i,j,q}(\tilde{U}_i^n(\mathbf{x}_{i,j,q}) - \overline{U}_i^n) = -v^+_{i,j} \left[g_{i,j,1}(U^+_{i,j,1} - \overline{U}_i^n) + g_{i,j,2}(U^+_{i,j,2} - \overline{U}_i^n) \right] \quad (B.5)$$

for the outflow term, with non-negative multipliers $v^-_{j,i}$, $v^+_{i,j}$, $f_{j,i}$, $f_{j,i,1}$, $f_{j,i,2}$, $g_{i,j,1}$, and $g_{i,j,2}$.

B.1 Inflow term

First we consider (B.4). Analogous to (5.23), we replace the limited solution values with the limited forward differences to obtain

$$\begin{aligned}\sum_{q=1,2}\frac{1}{2}v^{-}_{j,i,q}(\tilde{U}^n_j(\mathbf{x}_{i,j,q})-\overline{U}^n_i) = {} & \frac{v^{-}_{j,i,1}+v^{-}_{j,i,2}}{2}(\overline{U}^n_j-\overline{U}^n_i)\\ & + \frac{v^{-}_{j,i,1}\varphi_1(\mathbf{r}_{j,i,1})+v^{-}_{j,i,2}\varphi_1(\mathbf{r}_{j,i,2})}{2}l_{j,1}\frac{h_{j,1}}{6}\frac{U^f_{j,1}-\overline{U}^n_j}{d^f_{j,1}}\\ & + \frac{v^{-}_{j,i,1}\varphi_2(\mathbf{r}_{j,i,1})+v^{-}_{j,i,2}\varphi_2(\mathbf{r}_{j,i,2})}{2}l_{j,2}\frac{h_{j,2}}{4\sqrt{3}}\frac{U^f_{j,2}-\overline{U}^n_j}{d^f_{j,2}}.\end{aligned}$$

Let $v^{-}_{j,i}=\frac{v^{-}_{j,i,1}+v^{-}_{j,i,2}}{2}$, $v^{-,1}_{j,i}=\frac{v^{-}_{j,i,1}\varphi_1(\mathbf{r}_{j,i,1})+v^{-}_{j,i,2}\varphi_1(\mathbf{r}_{j,i,2})}{2}$, and $v^{-,2}_{j,i}=\frac{v^{-}_{j,i,1}\varphi_2(\mathbf{r}_{j,i,1})+v^{-}_{j,i,2}\varphi_2(\mathbf{r}_{j,i,2})}{2}$. If $v^{-}_{j,i}$ is nonzero, the above becomes

$$\sum_{q=1,2}\frac{1}{2}v^{-}_{j,i,q}(\tilde{U}^n_j(\mathbf{x}_{i,j,q})-\overline{U}^n_i) = v^{-}_{j,i}\left[(\overline{U}^n_j-\overline{U}^n_i)+l_{j,1}\frac{v^{-,1}_{j,i}}{v^{-}_{j,i}}\frac{h_{j,1}}{6}\frac{U^f_{j,1}-\overline{U}^n_j}{d^f_{j,1}}+l_{j,2}\frac{v^{-,2}_{j,i}}{v^{-}_{j,i}}\frac{h_{j,2}}{4\sqrt{3}}\frac{U^f_{j,2}-\overline{U}^n_j}{d^f_{j,2}}\right]. \tag{B.6}$$

We replace the forward difference with the backward difference if the sign of $v^{-,1}_{j,i}$ or $v^{-,2}_{j,i}$ is negative. We use the following shorthand notation

$$\alpha^{-}_{j,i,1}=\begin{cases}\frac{1}{6}\frac{h_{j,1}}{d^f_{j,1}}\frac{v^{-,1}_{j,i}}{v^{-}_{j,i}} \text{ if } v^{-,1}_{j,i}\geq 0\\ \frac{1}{6}\frac{h_{j,1}}{d^b_{j,1}r_{j,1}}\frac{|v^{-,1}_{j,i}|}{v^{-}_{j,i}} \text{ otherwise.}\end{cases} \quad \text{and} \quad U^{-}_{j,i,1}=\begin{cases}U^f_{j,1} \text{ if } v^{-,1}_{j,i}\geq 0\\ U^b_{j,1} \text{ otherwise.}\end{cases} \tag{B.7}$$

$$\alpha^{-}_{j,i,2}=\begin{cases}\frac{1}{4\sqrt{3}}\frac{h_{j,2}}{d^f_{j,2}}\frac{v^{-,2}_{j,i}}{v^{-}_{j,i}} \text{ if } v^{-,2}_{j,i}\geq 0\\ \frac{1}{4\sqrt{3}}\frac{h_{j,2}}{d^b_{j,2}r_{j,2}}\frac{|v^{-,2}_{j,i}|}{v^{-}_{j,i}} \text{ otherwise,}\end{cases} \quad \text{and} \quad U^{-}_{j,i,2}=\begin{cases}U^f_{j,2} \text{ if } v^{-,2}_{j,i}\geq 0\\ U^b_{j,2} \text{ otherwise.}\end{cases} \tag{B.8}$$

Then (B.6) becomes

$$\sum_{q=1,2}\frac{1}{2}v^{-}_{j,i,q}(\tilde{U}^n_j(\mathbf{x}_{i,j,q})-\overline{U}^n_i) = v^{-}_{j,i}\left[(\overline{U}^n_j-\overline{U}^n_i)+l_{j,1}\alpha^{-}_{j,i,1}(U^{-}_{j,i,1}-\overline{U}^n_j)+l_{j,2}\alpha^{-}_{j,i,2}(U^{-}_{j,i,2}-\overline{U}^n_j)\right].$$

Putting the above in the form (B.4), we have

$$\sum_{q=1,2}\frac{1}{2}v^-_{j,i,q}(\tilde{U}^n_j(\mathbf{x}_{i,j,q})-\overline{U}^n_i)=v^-_{j,i}\left[(1-l_{j,1}\alpha^-_{j,i,1}-l_{j,2}\alpha^-_{j,i,2})(\overline{U}^n_j-\overline{U}^n_i)\right.$$
$$\left.+l_{j,1}\alpha^-_{j,i,1}(U^-_{j,i,1}-\overline{U}^n_i)+l_{j,2}\alpha^-_{j,i,2}(U^-_{j,i,2}-\overline{U}^n_i)\right], \tag{B.9}$$

with $f_{j,i}=1-l_{j,1}\alpha^-_{j,i,1}-l_{j,2}\alpha^-_{j,i,2}$, $f_{j,i,1}=l_{j,1}\alpha^-_{j,i,1}$, and $f_{j,i,2}=l_{j,2}\alpha^-_{j,i,2}$.

Sum and non-negativity

The multipliers $f_{j,i,1}$ and $f_{j,i,2}$ are non-negative by (B.7) and (B.8). Requiring the coefficient $f_{j,i}$ to be non-negative gives the following condition

$$f_{j,i}=1-l_{j,1}\alpha^-_{j,i,1}-l_{j,2}\alpha^-_{j,i,2}\geq 0\quad\forall j\in N^e_i. \tag{B.10}$$

By definition, the sum of the coefficients is

$$f_{j,i}+f_{j,i,1}+f_{j,i,2}=1. \tag{B.11}$$

B.2 Outflow term

We now consider (B.5). Let $v^+_{i,j}=\frac{v^+_{i,j,1}+v^+_{i,j,2}}{2}$, $v^{+,1}_{i,j}=\frac{v^+_{i,j,1}\varphi_1(\mathbf{r}_{i,j,1})+v^+_{i,j,2}\varphi_1(\mathbf{r}_{i,j,2})}{2}$, and $v^{+,2}_{i,j}=\frac{v^+_{i,j,1}\varphi_2(\mathbf{r}_{i,j,1})+v^+_{i,j,2}\varphi_2(\mathbf{r}_{i,j,2})}{2}$. We replace the limited solution coefficients with the limited forward differences and if $v^+_{i,j}$ is nonzero, we obtain

$$\sum_{q=1,2}\frac{1}{2}v^+_{i,j,q}(\tilde{U}^n_i(\mathbf{x}_{i,j,q})-\overline{U}^n_i)=v^+_{i,j}\left(\frac{v^{+,1}_{i,j}}{v^+_{i,j}}l_{i,1}\frac{h_{i,1}}{6}\frac{U^f_{i,1}-\overline{U}^n_i}{d^f_{i,1}}+\frac{v^{+,2}_{i,j}}{v^+_{i,j}}l_{i,2}\frac{h_{i,2}}{4\sqrt{3}}\frac{U^f_{i,2}-\overline{U}^n_i}{d^f_{i,2}}\right).$$

We introduce the following variables for convenience

$$\alpha^+_{i,j,1}=\begin{cases}\frac{1}{6}\frac{h_{i,1}}{d^f_{i,1}}\frac{|v^{+,1}_{i,j}|}{v^+_{i,j}} & \text{if } v^{+,1}_{i,j}\leq 0\\ \frac{1}{6}\frac{h_{i,1}}{d^b_{i,1}r_{i,1}}\frac{v^{+,1}_{i,j}}{v^+_{i,j}} & \text{otherwise.}\end{cases}\quad\text{and}\quad U^+_{i,j,1}=\begin{cases}U^f_{i,1} & \text{if } v^{+,1}_{i,j}\leq 0\\ U^b_{i,1} & \text{otherwise,}\end{cases} \tag{B.12}$$

$$\alpha^+_{i,j,2} = \begin{cases} \frac{1}{4\sqrt{3}} \frac{h_{i,2}}{d^f_{i,2}} \frac{|v^{+,2}_{i,j}|}{v^+_{i,j}} \text{ if } v^{+,2}_{i,j} \leq 0 \\ \frac{1}{4\sqrt{3}} \frac{h_{i,2}}{d^b_{i,2} r_{i,2}} \frac{v^{+,2}_{i,j}}{v^+_{i,j}} \text{ otherwise.} \end{cases} \quad \text{and} \quad U^-_{i,j,2} = \begin{cases} U^f_{i,2} \text{ if } v^{+,2}_{i,j} \leq 0 \\ U^b_{i,2} \text{ otherwise.} \end{cases} \tag{B.13}$$

In the form (B.5), the above expansion is

$$\sum_{q=1,2} \frac{1}{2} v^+_{i,j,q} (\tilde{U}^n_i(\mathbf{x}_{i,j,q}) - \overline{U}^n_i) = -v^+_{i,j} \left[l_{i,1}\alpha^+_{i,j,1}(U^+_{i,j,1} - \overline{U}^n_i) + l_{i,2}\alpha^+_{i,j,2}(U^+_{i,j,2} - \overline{U}^n_i) \right]. \tag{B.14}$$

with $g_{i,j,1} = l_{i,1}\alpha^+_{i,j,1}$ and $g_{i,j,2} = l_{i,2}\alpha^+_{i,j,2}$.

Sum and non-negativity

The multipliers $g_{i,j,1}$ and $g_{i,j,2}$ are non-negative by (B.12) and (B.13). The sum of the coefficients is given by

$$g_{i,j,1} + g_{i,j,2} = l_{i,1}\alpha^+_{i,j,1} + l_{i,2}\alpha^+_{i,j,2}. \tag{B.15}$$

B.3 Putting it all together

The inflow terms and outflow terms have been expanded into sums of the form (B.4) and (B.5) in (B.9) and (B.14), respectively. Substituting these sums into the scheme (B.3) gives

$$\overline{U}^{n+1}_i = \overline{U}^n_i + \sum_{j \in N^e_i, j \neq i} v^-_{j,i} \left[f_{j,i}(\overline{U}^n_j - \overline{U}^n_i) + f_{j,i,1}(U^-_{j,i,1} - \overline{U}^n_i) + f_{j,i,2}(U^-_{j,i,2} - \overline{U}^n_i) \right]$$
$$+ v^+_{i,j} \left[g_{i,j,1}(U^+_{i,j,1} - \overline{U}^n_i) + g_{i,j,2}(U^+_{i,j,2} - \overline{U}^n_i) \right].$$

This is of the form (3.10). We require that the sum of the coefficients in front of the differences above is less than or equal to one. Using (B.11) and (B.15), we write this requirement as

$$\sum_{j \in N^e_i, j \neq i} v^-_{j,i} \left[f_{j,i} + f_{j,i,1} + f_{j,i,2} \right] + v^+_{i,j} \left[g_{i,j,1} + g_{i,j,2} \right] = \sum_{j \in N^e_i, j \neq i} v^-_{j,i} + v^+_{i,j} \left(l_{i,1}\alpha^+_{i,j,1} + l_{i,2}\alpha^+_{i,j,2} \right) \leq 1. \tag{B.16}$$

For (B.16) to be satisfied, we must satisfy on each edge of Ω_i

$$l_{i,1}\alpha^+_{i,j,1} + l_{i,2}\alpha^+_{i,j,2} \leq 1 \quad \forall j \in N^e_i, \tag{B.17}$$

and the time step constraint

$$\sum_{j \in N^e_i, j \neq i} \left(v^-_{j,i} + v^+_{i,j}\right) \leq 1, \quad \forall \Omega_i. \tag{B.18}$$

For positivity of the expansion coefficient $f_{j,i}$ we must satisfy the constraint (B.10) on all the edges of Ω_i

$$l_{j,1}\alpha^-_{j,i,1} + l_{j,2}\alpha^-_{j,i,2} \leq 1, \quad \forall j \in N^e_i. \tag{B.19}$$

B.4 Time step restriction

Using the definition of $v^-_{j,i}$ and $v^+_{i,j}$ in (B.1), (B.18) becomes

$$\Delta t \sum_{j \in N^e_i, j \neq i} \sum_{q=1,2} \frac{1}{2} \frac{|\partial\Omega_{i,j}|}{|\Omega_i|} \left(\frac{\partial F_{i,j}}{\partial U_1}(c^*_{i,j,q}, \overline{U}^n_i) - \frac{\partial F_{i,j}}{\partial U_2}(\tilde{U}^n_i(\mathbf{x}_{i,j,q}), c^{**}_{i,j,q}) \right) \leq 1. \tag{B.20}$$

Due to the differentiability of the numerical flux, there exists a λ_i such that $\frac{\partial F_{i,j}}{\partial U_1}(c^*_{i,j,q}, \overline{U}^n_i) \leq \lambda_i$ and $-\frac{\partial F_{i,j}}{\partial U_2}(\tilde{U}^n_i(\mathbf{x}_{i,j,q}), c^{**}_{i,j,q}) \leq \lambda_i$ hold. The time step restriction in (B.20) now becomes

$$2\Delta t \lambda_i \sum_{j \in N^e_i, j \neq i} \frac{|\partial\Omega_{i,j}|}{|\Omega_i|} \leq 1. \tag{B.21}$$

Note that the radius of the circle inscribed in Ω_i is

$$h_i = 2\frac{|\Omega_i|}{|\partial\Omega_i|}, \tag{B.22}$$

where $|\partial\Omega_i|$ and $|\Omega_i|$ are the perimeter and area of Ω_i, respectively. The nonlinear time step restriction in terms of the inscribed circle is

$$\Delta t \leq \frac{1}{4}\frac{h_i}{\lambda_i}. \tag{B.23}$$