Accurate Residual-Distribution Schemes for Accelerated Parallel Architectures

by

Stephen M. J. Guzik

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Aerospace Science and Engineering
University of Toronto

Abstract

Accurate Residual-Distribution Schemes for Accelerated Parallel Architectures

Stephen M. J. Guzik

Doctor of Philosophy

Graduate Department of Aerospace Science and Engineering

University of Toronto

2010

Residual-distribution methods offer several potential benefits over classical methods, such as a means of applying upwinding in a multi-dimensional manner and a multi-dimensional positivity property. While it is apparent that residual-distribution methods also offer higher accuracy than finite-volume methods on similar meshes, few studies have directly compared the performance of the two approaches in a systematic and quantitative manner. In this study, comparisons between residual distribution and finite volume are made for steady-state smooth and discontinuous flows of gas dynamics, governed by hyperbolic conservation laws, to illustrate the strengths and deficiencies of the residual-distribution method. Deficiencies which reduce the accuracy are analyzed and a new nonlinear scheme is proposed that closely reproduces or surpasses the accuracy of the best linear residual-distribution scheme. The accuracy is further improved by extending the scheme to fourth order using established finite-element techniques. Finally, the compact stencil, arithmetic workload, and data parallelism of the fourth-order residual-distribution scheme are exploited to accelerate parallel computations on an architecture consisting of both CPU cores and a graphics processing unit. Numerical experiments are used to assess the gains to efficiency and possible monetary savings that may be provided by accelerated architectures.

Acknowledgements

I would like to express my thanks to my supervisor, Professor Clinton Groth, for his teachings and especially for allowing me the freedom to pursue many of my own research interests. I am also grateful to the members of my doctoral committee for their guidance during the course of my studies. My time at the university was significantly enhanced by interactions with my colleagues, especially Xinfeng Gao for intellectual and enjoyable discussions and Scott Northrup for exceptional technical support.

Outside of the university, I owe much to my parents for their support during my study.

Contents

List of Symbols

Math

Ω	-	area
(ξ, η)	-	two-dimensional coordinate system aligned with advection vector
d	-	number of dimensions
$\hat{\mathbf{I}}$	-	identity matrix
$\hat{n}$	-	inwards pointing unit normal
$\vec{n}$	-	inwards pointing normal
s	-	distance along an edge
$\mathcal{S}$	-	surface (or path in 2-D) in space
t	-	time
(x, y)	-	two-dimensional Cartesian coordinate system
$\vec{x}$	-	spatial vector (for any x)
$\mathbf{X}$	-	vector of solution variables (for any X)

Fluids

β	-	Mach number parameter
γ	-	specific heat ratio
ρ	-	density
θ	-	flow angle
a	-	x-direction advection velocity/speed of sound
$\vec{\mathbf{A}}$	-	flux Jacobians
b	-	y-direction advection velocity
e_T	-	specific total energy
$\vec{f}$	-	flux vector

$\vec{\mathbf{F}}$	-	flux dyad
h_T	-	specific total enthalpy
M	-	Mach number
p	-	pressure
q	-	flow magnitude
$\mathbf{Q}$	-	symmetrizing variables
$\breve{\mathbf{Q}}$	-	symmetrizing variables with velocity expressed by a magnitude and an angle
R	-	specific gas constant
s	-	entropy
S	-	entropy proportionality
u	-	scalar variable/x-direction velocity
$\mathbf{U}$	-	conserved variables
v	-	y-direction velocity
$\mathbf{V}$	-	primitive variables
$\mathbf{W}$	-	characteristic variables
$\mathbf{Z}$	-	Roe's parameter vector

Algorithm

α	-	coefficient of error magnitude/preconditioner parameter
β	-	distribution coefficient/spatial order of accuracy
χ	-	preconditioner parameter
ϵ	-	small additional floating point term
η	-	preconditioner parameter
λ	-	advection speed/eigenvalue
$\mathbf{\Lambda}$	-	diagonal matrix
ϕ	-	fluctuation
ψ	-	projection of the fluctuation
Ψ	-	symmetric limiter
τ	-	time parameter
θ	-	blending coefficient
h	-	time step
Δh	-	mesh spacing

E	-	element
H	-	solution time
i	-	vertex
k	-	scalar inflow parameter
$\mathbf{K}$	-	matrix inflow parameter
L	-	length
L_i	-	barycentric coordinates normalized over a reconstruction element
$\mathbf{L}$	-	rows of left eigenvectors
N	-	number of grid points representing discrete unknowns
N_D	-	number of grid points representing discrete unknowns in a single dimension
$\mathbf{N}_Q$	-	inverse of matrix inflow parameters
$\mathbf{P}$	-	preconditioner
$\mathbf{R}$	-	columns of right eigenvectors
s	-	mesh aspect ratio
S	-	a particular scalar variable/speedup
S_i	-	barycentric coordinates normalized over a primary element
$\mathbf{S}$	-	a particular vector of variables
T	-	normalized primary element/total execution time for an algorithm

List of Tables

List of Figures

Chapter 1

Introduction

Computational methods for solving hyperbolic partial differential equations are well established and computational fluid dynamics (CFD) has gained widespread acceptance as a powerful tool for engineering design. While classical methods, such as finite-difference and finite-volume ($\mathcal{FV}$), have found considerable success, research towards improving the core algorithms is still very active. New classes of methods such as residual distribution ($\mathcal{RD}$), also known as fluctuation splitting, are specifically designed to address known shortcomings of the classical methods and may offer several additional improvements as well.

The first objective of this thesis is to systematically quantify the performance of the $\mathcal{RD}$ method relative to the Godunov-type $\mathcal{FV}$ method. The latter method is chosen as a reference because the $\mathcal{FV}$ method is mature and has demonstrated an exceptional capability for computing solutions to discontinuous flows both robustly and accurately. Despite these characteristics, there is some dissatisfaction with current finite volume ($\mathcal{FV}$) methods. While elegant and physical in one dimension, $\mathcal{FV}$ methods do not extend readily to multiple dimensions because the Riemann problem itself does not extend readily to multiple dimensions. The usual workaround is to apply the one dimensional scheme in multiple directions, a process in which the splitting of the flux becomes biased in directions normal to the faces of the computational cells. Consequently, the schemes are no longer quite as physical and this causes a corresponding decrease in the accuracy via excess numerical dissipation. As shown by Roe and Sidilkover [51], dimensional splitting is about the worst thing one can do for first-order solutions. Residual distribution methods attempt to correct this deficiency by explicitly modelling the underlying multidimensional physics.

The most significant advantages of the $\mathcal{RD}$ method are regularly advertised in the literature and generally illustrate their excellent shock-capturing capabilities. The comparison considered herein between $\mathcal{RD}$ and $\mathcal{FV}$ serves additional purposes. It provides insight into the practicality of $\mathcal{RD}$ as a mainstream solution method by identifying any disadvantages with respect to both the accuracy in smooth regions and in terms of robustness. It is our opinion that the notable disadvantages in the $\mathcal{RD}$ method are a result of its relative immaturity and that most can be overcome. Indeed, much of the current literature simply focuses on addressing problematic issues [3, 6, 53] rather than applying or extending the $\mathcal{RD}$ method. The second objective of this thesis to address these deficiencies, by highlighting them and proposing solutions.

Residual distribution methods have a number of ancillary advantageous characteristics, many of which are borrowed from finite-element methods and make it easier to extend the method to orders of accuracy greater than two. These high-order schemes are explored herein as candidates for processing on parallel heterogeneous architectures. The final objective of this thesis is to explore methods for adapting an $\mathcal{RD}$ scheme to new “accelerators” such a graphics processing units (GPU). Specifically, hybrid parallel methods (those consisting of several levels of parallelism) are considered for heterogeneous architectures featuring concurrent execution on both central processing units (CPU) and GPUs.

The three main objectives of the thesis are thus a quantitative comparison with $\mathcal{FV}$, identification and correction of deficiencies, and a high-order extension on parallel accelerated architectures. In the next sections, an introduction to the $\mathcal{RD}$ method and

GPU acceleration are provided along with the governing equations and several canonical test problems used to evaluate the computational performance of the method.

1.1 Residual Distribution

Residual distribution already has a long history; in 1981, a Lax-Wendroff $\mathcal{RD}$ scheme was proposed by Ni [37] and, by 1986, the multidimensional framework for $\mathcal{RD}$ had been developed by Roe [50]. Discussion of the early history is available in [4, 22] and in many early theses, most notably [43, 57]. Techniques for $\mathcal{RD}$ were formalized in the early 1990s through a collaborative effort between Roe and co-workers at the University of Michigan and Deconinck and co-workers at the Von Kármán Institute for Fluid Dynamics, resulting in the publication of three notable theses [35, 43, 57]. Significant contributions were also due to efforts by Sidilkover [51, 54].

Around the turn of the century, efforts at the University of Michigan concentrated on achieving solutions of the Euler equations via hyperbolic/elliptic splitting. Following research by Mesaros [35] and Nishikawa [38], the thesis by Rad [46] is perhaps illustrative of an "ultimate" Euler solver where hyperbolic parts of the system are treated by wave-based multidimensional upwind methods and elliptic portions are solved by least-squares minimization. Constraints on the minimization force the solver to preserve potential flow and result in a scheme that can effectively handle flow regimes ranging from incompressible to supersonic, all while retaining the excellent shock-capturing properties of $\mathcal{RD}$. This approach highlights the physics of the governing PDEs and yields very accurate results. Drawbacks of Rad's $\mathcal{RD}$ scheme include fine tailoring to the Euler equations and the lack of a straightforward extension to three dimensional flows.

Efforts at the Von Kármán Institute have instead generally focused on matrix distribution techniques [58, 59] for solution of the Euler system of equations. The waves crossing a face of an element are described in matrix form, rather than by first performing a characteristic decomposition, and matrix-vector analogues of scalar techniques are used to determine the solution. In this approach, the physics of the governing PDEs are "hidden" in the matrices such that one cannot tune the solution procedure as explicitly as with hyperbolic/elliptic splitting. Nevertheless, the results are still quite accurate and more importantly, the technique is easily extended to any number of dimensions or to systems of equations.

Since the year 2000, notable advancements include relaxation of the requirement for a conservative linearization [19] and the discovery of *mapped* distribution schemes which are essentially a generalization of limiters to multiple space dimensions [2, 4, 9]. There has also been a significant amount of work on systems other than the Euler equations [41, 44] and the solution of unsteady flows [2, 18, 48]. A wide variety of techniques have been proposed for extending the $\mathcal{RD}$ method to orders of accuracy greater than two [13, 16, 32]; the most promising simply follows techniques established for the finite-element method [9].

The mathematical foundations of $\mathcal{RD}$ have been primarily driven by a large number of publications by Abgrall and co-workers. These efforts generally involve proofs of the stability of the schemes [5, 8], however, study of the more intricate mathematics has also shed light on some accuracy and convergence issues [3, 6]. The thesis by Ricchiuto [47] and related works [21, 48] provide a very complete description of the fundamental mathematics behind $\mathcal{RD}$.

The rationale behind the development of $\mathcal{RD}$ methods is to find a technique that is superior in terms of accuracy per unit computational cost over existing methods. There are several advantages to the $\mathcal{RD}$ method which help achieve this goal [22]:

- The first order scheme features much less cross-diffusion than a dimensionally split $\mathcal{FV}$ scheme [51]. This is a consequence of the narrow stencil used by the first-order $\mathcal{RD}$ scheme for updating the unknowns. Since its inception, this feature has been well documented. The first-order scheme is not examined herein in much detail

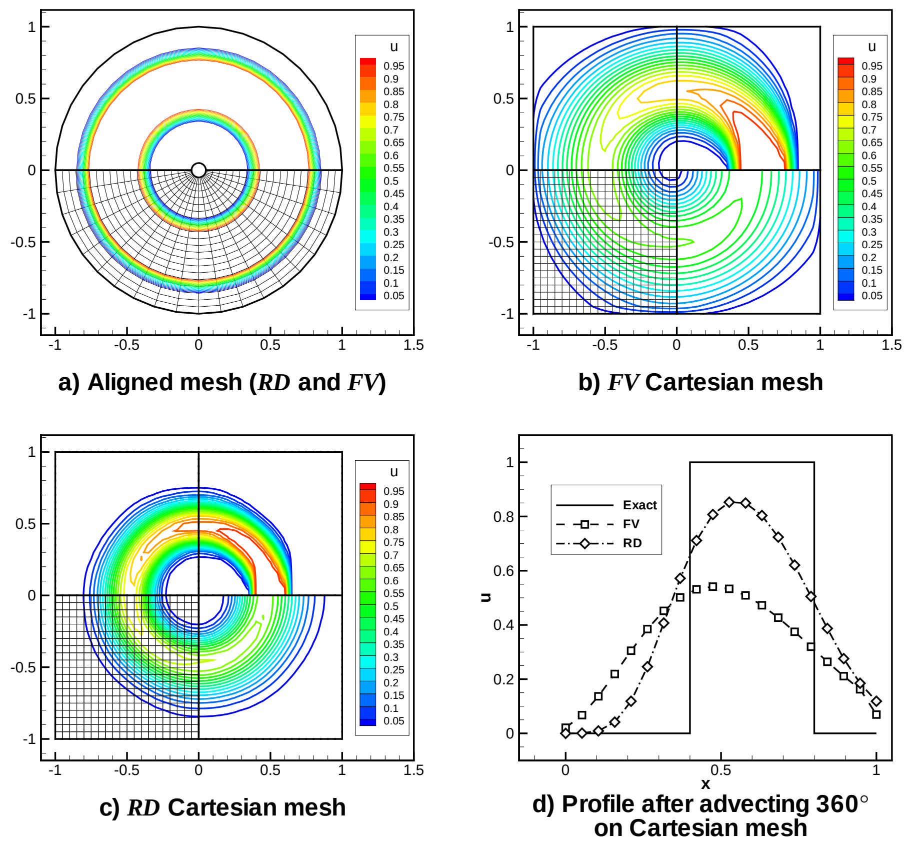


Figure 1.1 Dissipation of first-order $\mathcal{FV}$ and $\mathcal{RD}$ schemes on aligned and Cartesian meshes for circular advection of a "top-hat" profile.

as we will concentrate the high-order behaviour of the method. Nevertheless, the first-order scheme is very important because it serves as a basis for the construction of many $\mathcal{RD}$ schemes. A visual summary of the dissipative characteristics of the first-order $\mathcal{RD}$ scheme compared to a first-order $\mathcal{FV}$ scheme is provided in Fig. 1.1. For circular advection of a "top-hat" profile, both schemes closely reproduce the exact solution if the mesh is aligned with the advection vector (Fig. 1.1a). However, when the flow becomes multidimensional within the elements, as on a Cartesian mesh, significant dissipation is observed. The better accuracy of the $\mathcal{RD}$ scheme is illustrated in Fig. 1.1d after advecting the profile 360° on a Cartesian mesh.

- Residual distribution schemes have an inherent multidimensional positivity property allowing for strict nonlinear stabilization. While nonlinear stabilization can be proven in one dimension for classical methods such as finite-difference and finite-volume, the extension to multiple dimensions is less rigorous even though application to multiple dimensions has been shown to work quite well for most practical CFD applications. Finite-element methods, while allowing for proven multidimensional stability properties, can suffer from a lack of accuracy or require problem-dependent tunable constants [22, 47]. For application of the $\mathcal{RD}$ method to scalar equations, positivity of the monotone and linear N scheme is shown herein. The energy stability of the first-order $\mathcal{RD}$ scheme (N scheme) applied to linear equations, both scalar and systems, is proven by Abgrall and Barth [5], Abgrall and Mezine [8], and Ricchiuto [47]. The results extend to any symmetrizable system. While entropy stability can be shown for nonlinear scalar equations [47], the results for nonlinear systems are less comprehensive and only indicate that an entropy condition is satisfied for certain quadratures and in the limit of mesh-refinement [5]. Numerical results, however, invariably demonstrate the non-oscillatory behaviour of the N scheme [2, 3, 21, 22].

- A compact reconstruction stencil (different from the update stencil), allows for second-order accuracy on a stencil of only one element. The compact reconstruction stencil simplifies application of boundary conditions, eases parallelization of the algorithm, and increases the efficiency of the algorithm, especially for higher orders of accuracy.

Within the literature, most of the results emphasize the accurate shock-capturing properties of the $\mathcal{RD}$ schemes. This is indeed one of the most advantageous characteristics but accuracy in smooth regions is also a requirement for practical computation, especially when constrained by limited computational resources (and hence mesh size). While the literature has shown that $\mathcal{RD}$ methods are more accurate than $\mathcal{FV}$ methods when applied to the solution of problems on the same size mesh [43, 46, 63, 64, 65], most of these previous studies were more qualitative in nature, usually providing a visual comparison

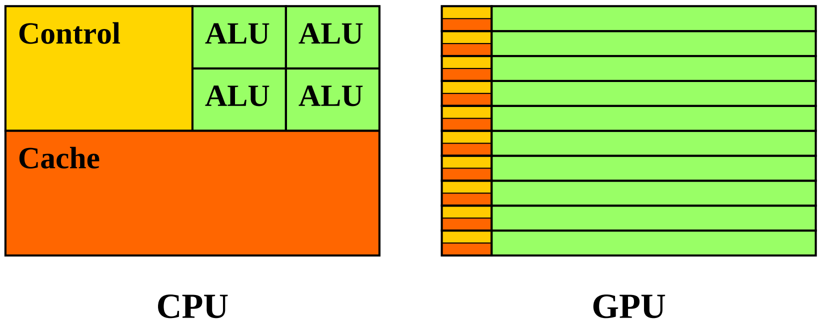


Figure 1.2 The GPU devotes more transistors to data processing than the CPU [42].

of a given flow feature. There is little to be found on the quantitative comparative accuracy of $\mathcal{RD}$ and $\mathcal{FV}$ methods, especially with respect to the more practical nonlinear distribution methods which are both second-order and monotone. The results obtained by Abgrall [1] provide some insight, but the quantitative comparisons made therein only describe solution minimums and maximums. With respect to the advancement of residual distribution methods, this thesis focuses on accuracy in smooth flows. Of course, in achieving maximum accuracy, it is desired to not compromise the advantages stated above, most notably the nonlinear stability characteristics. This is a challenging task since the accuracy of a method always seems to be inversely proportional to the nonlinear stability of the method.

1.2 Parallel Accelerated Architectures

The potential for using graphics processing units (GPU) to assist with CFD has recently generated considerable interest [12, 17, 55]. As shown in Fig. 1.2, the GPU devotes more transistors to data processing than the CPU, which diverts significant resources to data caching and flow control [42]. While the CPU can efficiently process conditional instructions (branching) and dispersed data, the GPU is specialized for computation of highly parallel data using instructions with a high arithmetic intensity. Programs that exhibit these characteristics can benefit from the massive computational power of the

GPU over the CPU. Perhaps most importantly, the GPU is a very cost-effective parallel processor since research and development are supported by the graphics visualization market.

Previously, using the GPU for computation has been difficult because code for the GPU could only be expressed using application programming interfaces (API) designed for graphics. NVIDIA has recently provided the Compute Unified Device Architecture (CUDA), essentially a programming model, for "issuing and managing computations on the GPU as a data-parallel computing device without the need of mapping them to a graphics API [42]." With CUDA, the GPU can be programmed using the C programming language and a library is provided for managing the GPU from code running on the CPU. The introduction of CUDA also included changes to the hardware that facilitate general-purpose computing. In particular, general scatter and gather operations to global memory (RAM) are now permitted and a parallel data cache consisting of on-chip shared memory increases the flexibility in which an algorithm can be designed [42]. Alternative programming standards such as OpenCL [36] are emerging and may resolve many of the concerns related to portability that arise from using a vendor-specific API.

In this work, the potential of using GPUs to assist with the computation of discrete solutions to systems of partial differential equations using high-order residual-distribution ($\mathcal{RD}$) techniques is explored. The $\mathcal{RD}$ method is an attractive candidate for GPU computing because it provides a compact reconstruction stencil (lowering memory operations) and the high-order extension increases the arithmetic intensity. We take the perspective that the different parts of the algorithm, depending on their characteristics, are better suited to either the GPU or CPU. Suitability is determined both by the computational efficiency that can be achieved on a processor and by the complexity of writing and maintaining a particular section on a processor. Consequently, the resulting algorithm illustrates the use of several levels of parallelism including a heterogeneous level featuring simultaneous processing by the CPU and GPU; at this level, the algorithm is split between the two processors depending on the aforementioned suitability. A similar attempt at concurrent processing is described by Stock and Gharakhani [56] for a vortex-particle method using an N-body approach.

1.3 Governing Equations and Canonical Problems

To study $\mathcal{RD}$, we apply the method to hyperbolic systems of conservation laws having the general form

$$\frac{\partial \mathbf{U}}{\partial t} + \vec{\nabla} \cdot \vec{\mathbf{F}} = 0\,, \tag{1.1}$$

where $\mathbf{U}$ is the solution vector and $\vec{\mathbf{F}}$ is the flux dyad. The scalar advection equation, nonlinear Burgers equation, and Euler equations of inviscid compressible gas dynamics are all of this form and will be considered here. The latter are especially interesting, not only because of the relevance to practical work, but because it is in the solution of systems that $\mathcal{RD}$ methods can become more complex and face greater challenges. Only steady solutions in two dimensions are considered as part of this research.

Several canonical problems are repeatedly used to evaluate the methods and are introduced below. Each has an exact analytical solution allowing for the error to be described by a particular norm given by

$$L_p\text{-error} = \left(\frac{\int |u - u_{\text{exact}}|^p \, \mathrm{d}\Omega}{\Omega} \right)^{\frac{1}{p}}\,, \tag{1.2}$$

where $\mathrm{d}\Omega$ is the area associated with a discrete error measure. The integration is performed numerically using a rectangle rule around each vertex (the location of the discrete unknowns in $\mathcal{RD}$). By evaluating a sequence of grids with an increasing number of discrete unknowns, N, the error can be fit to the relation

$$L_p\text{-error} = \alpha {N_D}^{\beta}\,, \tag{1.3}$$

where N_D is the dimensional spacing of the computational grid, $N_D = \sqrt{N}$. In (1.3), β denotes the spatial order of accuracy (for second order, $\beta \approx -2$) and α describes the

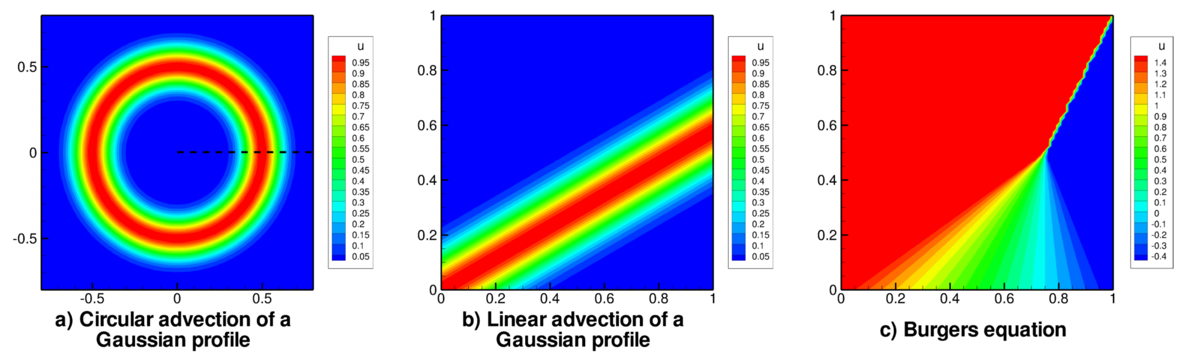


Figure 1.3 Scalar problems for study.

absolute magnitude of the error. This quantitative description of the error is used as a basis for all analysis in this work and is supplemented by qualitative contour plots of computed distributions where appropriate.

1.3.1 Scalar problems

The computational methods are applied to the scalar problems of circular advection of a Gaussian profile, linear advection of a Gaussian profile, and nonlinear Burger's equation. Analytical solutions for the three cases are shown in Fig. 1.3. The linear advection equation is given by

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} + b\frac{\partial u}{\partial y} = 0\,, \tag{1.4}$$

where $a = a(x, y)$ and $b = b(x, y)$ are the components of the advection velocity field. The Gaussian profile, $u = \mathrm{e}^{-0.5[(x-0.5)/0.08]^2}$, prescribes a smooth solution. For the circular advection problem, Fig. 1.3a, the Gaussian profile is assigned at the $(0 \leq x \leq 1,\ y = 0)$ boundary and then advected in a counter-clockwise direction on a domain extending from -1 to 1 in both dimensions. For the linear advection problem, Fig. 1.3b, the Gaussian profile is advected at an angle of 30° on a domain extending from 0 to 1.

The exact solution to the nonlinear Burgers equation, given by

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + 1\frac{\partial u}{\partial y} = 0\,, \tag{1.5}$$

is shown in Fig. 1.3c and used to evaluate the shock-capturing properties of both the $\mathcal{RD}$ and $\mathcal{FV}$ schemes for scalar equations. A steady problem was studied on a square solution domain in which the boundary values of the solution u were specified to vary linearly from 1.5 to -0.5 along the x-axis. This results in the formation of a compression wave that strengthens and produces a shock at $(x=0.75, y=0.5)$. The shock then progresses upward and leaves the solution domain at the top right corner. The extent of the domain is from 0 to 1 in both dimensions.

For all scalar problems, Dirichlet boundary conditions are assigned whenever the advection vector points into the computational domain.

1.3.2 Euler system problems

The two-dimensional Euler equations are given by (1.1) with

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho e_T \end{bmatrix}, \quad \mathbf{F}_x = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho u v \\ \rho u h_T \end{bmatrix}, \quad \mathbf{F}_y = \begin{bmatrix} \rho v \\ \rho u v \\ \rho v^2 + p \\ \rho v h_T \end{bmatrix}, \tag{1.6}$$

where e_T is the specific total energy and h_T is the specific total enthalpy. In this work, gaseous flows of air are considered and the preceding partial differential equations are supplemented with the ideal gas law, $p = \rho R T$, as an equation of state. The specific gas constant, R, is taken to be 287 J/(kg $\cdot$ K) and a perfect gas is assumed with a specific heat ratio of $\gamma = 1.4$.

The problems to be considered are shown in Fig. 1.4. Problems of supersonic flow with shocks are in the first two sub-figures. An oblique shock is shown in Fig. 1.4a. The incident flow is at Mach 2 and orientated at an angle of $-13.8978\,^\circ$ to an inviscid wall at $y = 0$. The resulting shock is inclined at $30\,^\circ$ to the wall. A supersonic inflow condition is specified at the left boundary, supersonic outflow at the right, and Dirichlet at the top. Figure 1.4b shows a supersonic flow at Mach 3 past a diamond-shaped aerofoil.

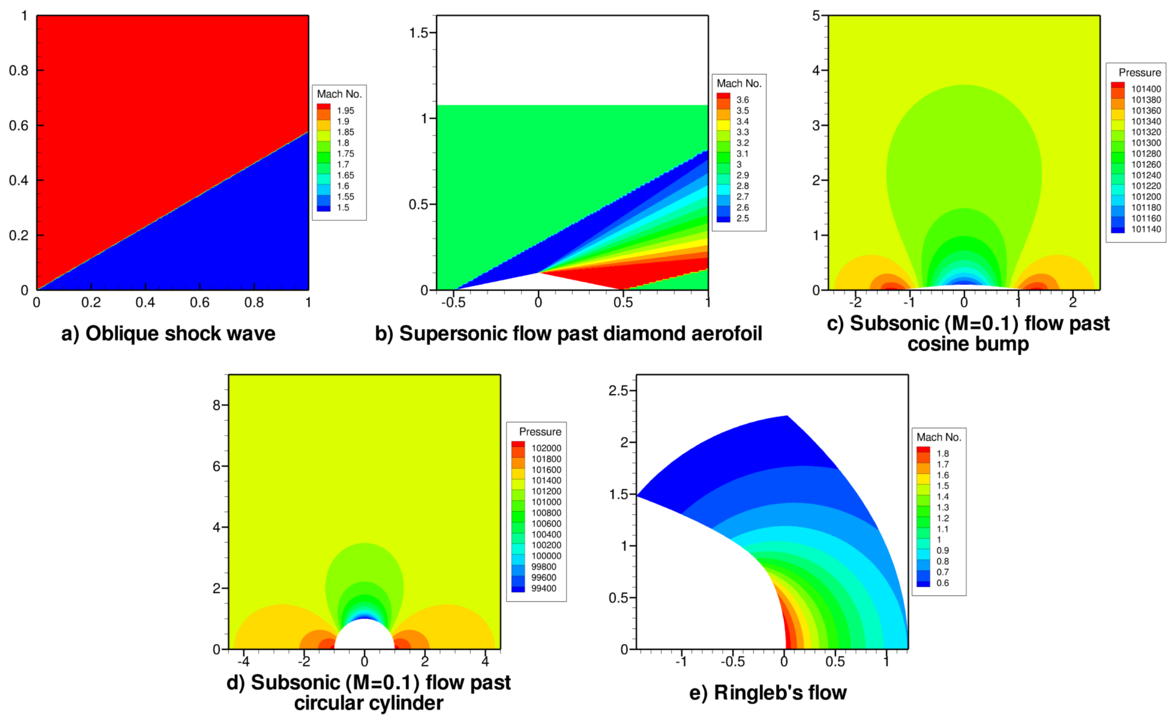


Figure 1.4 Euler system problems for study.

Freestream pressure and temperature corresponding to standard atmospheric conditions are used and the angle of attack is $0\,^\circ$. The solution domain features a horizontal axis of symmetry and the outflow boundary is close enough to prevent any interaction between the otherwise simple and centered waves. Hence, an exact solution can be determined for the entire domain. The left boundary is prescribed as supersonic inflow and the right as supersonic outflow. A Dirichlet condition is specified at the top.

Figures 1.4c and 1.4d consist of subsonic flow over a smooth cosine bump and past a circular cylinder, respectively. The flow is at standard atmospheric conditions and the freestream Mach number was taken to be 0.1. A horizontal axis of symmetry is imposed along with an inviscid wall condition over the object and a far-field condition is used away from the object. The smooth bump has a chord of 3 and the profile is defined by

$$y = \frac{1}{20}\left(\cos\left(\frac{\pi x}{1.5}\right) + 1\right)\,, \qquad -1.5 \leq x \leq 1.5\,. \tag{1.7}$$

The far-field boundary is placed at 7 times the chord. The circular cylinder has a diameter of 2 and the far-field is placed at 10.5 times this diameter. As both these inviscid flows are homentropic, any deviations from the freestream entropy are a result of numerical solution error. Changes in entropy, given by

$$\delta s = \frac{R}{\gamma - 1} \ln\left(\frac{p}{\rho^\gamma}\right) - s_\infty \,, \tag{1.8}$$

were therefore used to define solution error where s_∞ is the value of entropy in the freestream. The problem of subsonic flow past a cylinder introduces the complexity of stagnation regions compared to the bump problem.

Ringleb's flow, Fig. 1.4e, is a hodograph solution to the Euler equations [52] that involves an isentropic and irrotational flow contained between two streamlines. The availability of the analytic solution for this case makes it useful for demonstrating the accuracy of the spatial discretization. The left and right boundaries are delimited by streamlines and there is subsonic inflow at the top and mixed subsonic/supersonic outflow at the bottom of the domain. Dirichlet boundary conditions are applied at all boundaries.

1.4 Thesis Outline

The remainder of this thesis is organized as follows. In Chapter 2, details of the $\mathcal{RD}$ method are presented for scalar equations. Techniques for systems of equations are described in Chapter 3. Details of the implementation are discussed in Chapter 4 including the method of time-marching and the imposition of boundary conditions. The high-order approach is also presented in Chapter 4. In Chapter 5, the second-order $\mathcal{RD}$ method is compared against a Godunov-type $\mathcal{FV}$ method using the canonical problems described in the previous section. The advantages and disadvantages of the $\mathcal{RD}$ scheme are highlighted. In Chapter 6, a technique is proposed for enhancing the accuracy of nonlinear $\mathcal{RD}$ schemes. Chapter 7 introduces the target architecture and how the fourth-order $\mathcal{RD}$ scheme is adapted to run on heterogeneous systems consisting of GPUs and CPUs. The efficiency of using a GPU is analyzed in detail. Our final results are presented in Chapter 8 where second and fourth-order solutions are obtained using the proposed techniques for improving the accuracy. The fourth-order results are additionally obtained on the heterogeneous architecture using adaptations necessary for computing at reduced precision. Comparisons are made to solutions predicted using trusted but less comprehensive $\mathcal{RD}$ techniques relevant to the problem at hand. Conclusions, including the contributions of the thesis, are the subject of Chapter 9.

Chapter 2

Residual Distribution Methods for Scalar Equations

Residual distribution methods calculate the residual (or fluctuation) on an element, E, of an unstructured mesh and then, by some appropriate method, distribute the fluctuation to the nodes of that element to advance the solution in time. There are three distinct steps: computation of the fluctuation, distribution of the fluctuation, and evolution of the solution. Residual distribution methods are cell-vertex methods that are usually solved on simplexes (triangles in two space dimensions). A typical element is illustrated in Fig. 2.1 and highlights several relevant features: solution unknowns at the vertices of the triangle, inwards normals of edges numbered according to the opposing vertices and scaled by the lengths of the edges, and contributions to the dual mesh constructed from the element centroid and edge mid-points.

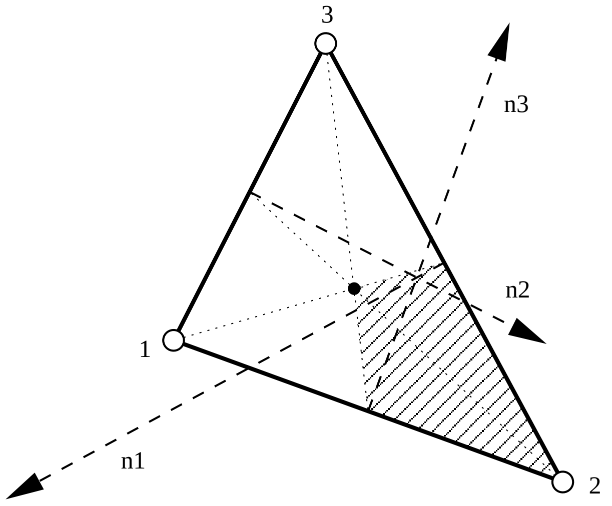


Figure 2.1 A simplex element for an $\mathcal{RD}$ scheme.

In this chapter, $\mathcal{RD}$ methods will be explained for scalar equations. There is a very large number of $\mathcal{RD}$ schemes, most of which feature one or more desirable properties and one or more undesirable properties. Only standard schemes, those which are well established in the literature and have the most desirable behaviour or characterize a particular property, are presented for completeness. The details of these methods are well documented in a variety of related theses [35, 43, 46, 47, 57] and at least summarized in much of the literature. The mathematical notation expressed by Csík *et al.* [19] and Ricchiuto [47] is closely followed for consistency.

2.1 Theory for Scalar Equations

For the scalar advection equation, one can express the PDE as

$$\frac{\partial u}{\partial t} + \sum_{j=1}^{d} \left(\lambda_j \frac{\partial u}{\partial x_j} \right) = 0\,, \tag{2.1}$$

where d is the number of dimensions and λ_j is the advection speed in the jth coordinate direction. The *fluctuation* on a simplex element, E, is defined as

$$\phi^E = -\int_E \frac{\partial u}{\partial t} \mathrm{d}\Omega_E = \int_E \sum_{j=1}^{d} \left(\lambda_j \frac{\partial u}{\partial x_j} \right) \mathrm{d}\Omega_E = \left[\sum_{j=1}^{d} \left(\bar{\lambda}_j \hat{x}_j \right) \right] \cdot \int_E \vec{\nabla} u \, \mathrm{d}\Omega_E\,, \tag{2.2}$$

where Ω_E is the element area, $\hat{x}_j$ defines a unit vector in the jth coordinate direction, and $\bar{\lambda}_j$ is linearized over the element. For nonlinear equations, $\bar{\lambda}_j$ is determined via a conservative linearization such that

$$\left[\sum_{j=1}^{d} \left(\bar{\lambda}_j \hat{x}_j \right) \right] \cdot \int_E \vec{\nabla} u \, \mathrm{d}\Omega_E = -\oint_{\partial E} \vec{f}(u) \cdot \hat{n} \, \mathrm{d}\mathcal{S}\,, \tag{2.3}$$

where $\hat{n}$ is the inwards-pointing unit normal vector of surface element $\mathrm{d}\mathcal{S}$ and $\vec{f}(u)$ is the flux vector. For second-order schemes, the solution, u, is assumed to vary linearly in the element. The integral in (2.2) can then be evaluated exactly to obtain

$$\phi^E = \left[\sum_{j=1}^{d} \left(\bar{\lambda}_j \hat{x}_j \right) \right] \cdot \frac{1}{d} \sum_{i=1, i\in E}^{d+1} u_i \vec{n}_i = \sum_{i=1, i\in E}^{d+1} k_i u_i\,. \tag{2.4}$$

The index i loops over each vertex of an element and the vector $\vec{n}_i$ defines the inwards normal of the edge opposite node i and scaled by the length of the edge. The *inflow parameters*, k_i, are defined by

$$k_i = \frac{1}{d}\left[\sum_{j=1}^{d}\left(\bar{\lambda}_j \hat{x}_j\right)\right] \cdot \vec{n}_i \, , \tag{2.5}$$

and describe whether edge i sees the inflow (positive k) or outflow (negative k) of the solution quantity. Because $\bar{\lambda}_j$ is linearized, $\sum_{i=1,i\in E}^{d+1} k_i = 0$. The inflow parameters can be used to interpolate the solution at inflow and outflow points according to

$$u_{in} = \frac{\displaystyle\sum_{i=1,i\in E}^{d+1} k_i^- u_i}{\displaystyle\sum_{i=1,i\in E}^{d+1} k_i^-} \, , \tag{2.6}$$

and

$$u_{out} = \frac{\displaystyle\sum_{i=1,i\in E}^{d+1} k_i^+ u_i}{\displaystyle\sum_{i=1,i\in E}^{d+1} k_i^+} \, , \tag{2.7}$$

where $k_i^+ = \max(0, k_i)$ and $k_i^- = \min(0, k_i)$. Using these definitions, the fluctuation defined by (2.4) can be cast into the form

$$\phi^E = \sum_{i=1,i\in E}^{d+1} \max(0, k_i)\left(u_{out} - u_{in}\right) \, , \tag{2.8}$$

which shows that the fluctuation is zero when the inflow and outflow points are at the same value (or streamwise invariant) [43].

The distribution of the fluctuation to the nodes, ϕ_i^E, is governed by distribution coefficients, β_i, with $\phi_i^E = \beta_i \phi^E$ and, for consistency, $\sum_{i,i\in E}^{d+1} \beta_i = 1$. The distribution can be represented graphically as shown in Fig. 2.2; the barycentric coordinates of a "distribution point" define the distributions to each vertex, e.g., $\beta_1 = \Omega_1/\Omega_E$.

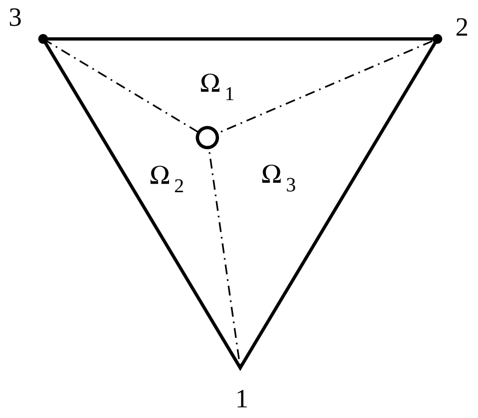


Figure 2.2 Geometrical representation of a distribution. The barycentric coordinates of the distribution point equal the distribution coefficients.

The nodal residual is defined as the sum of all fluctuations distributed to node i from all elements, E, that share node i as a vertex. The semi-discrete update formula is then

$$\Omega_i \frac{\mathrm{d}u_i}{\mathrm{d}t} + \sum_E \beta_i^E \phi^E = 0\,, \tag{2.9}$$

where Ω_i is the area of the dual mesh, shown in Fig. 2.3, associated with node i of the unstructured mesh. Various time-marching schemes can be applied to the solution of the coupled ordinary differential equations given above for the nodal values of the solution.

In two dimensions, one can envision two possible orientations of the elemental triangles: those with one downstream vertex (one positive inflow parameter - type I triangle in Fig. 2.4) and those with two downstream vertices (two positive inflow parameters - type II triangle in Fig. 2.4). In the former case, an upwind scheme sends all the fluctuation to the downstream vertex. In the latter case, the fluctuation is split between the two downstream nodes.

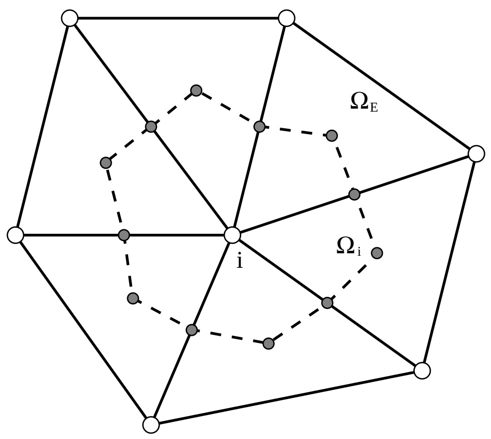


Figure 2.3 Primary elements (solid lines) and dual mesh (dashed lines) created from element centroids and edge midpoints.

There are a variety of $\mathcal{RD}$ schemes, each distinguished by a different method for determining the distribution coeffi-

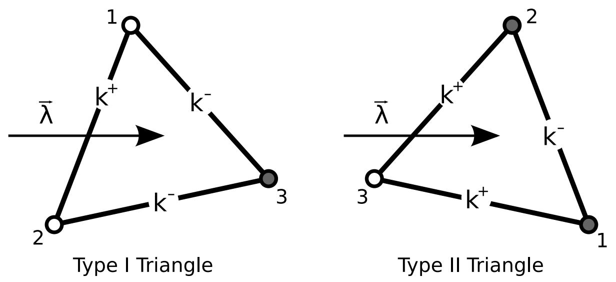


Figure 2.4 Possible triangle orientations in a two-dimensional flow. $\vec{\lambda}$ is the linearized characteristic vector

cients, β_i^E. The different residual distribution schemes can be classified according to various properties that they may satisfy. The properties relevant to this work are as follows:

Upwind ($\mathcal{U}$) — No fluctuation is sent to an upstream vertex. This implies that the distribution point in Fig. 2.2 must lie on the perimeter of the element.

Positivity ($\mathcal{P}$) — A nonlinear bound on solution extrema is achieved if the residual update can be written as a sum of the surrounding vertices with positive coefficients,

$$u_i^{n+1} = \sum_j^N c_{ij} u_j^n \,, \qquad c_{ij} > 0 \,, \qquad j = 1 \ldots N \,. \tag{2.10}$$

Schemes of this form are "local extremum diminishing" (local maxima are non-increasing and local minima are non-decreasing) and thus monotonicity-preserving. An extensive discussion of the subject including proofs of energy and entropy stability is available in the thesis by Ricchiuto [47]. In practice, positivity is prescribed within each element rather than on the dual mesh. This stricter condition still ensures (2.10) and allows for completion of the distribution step within the stencil of an element.

Linearity preservation ($\mathcal{LP}$) — An exact linear solution, if imposed at the discrete grid points, will not be altered by the scheme. This property allows for orders of accuracy greater than one. It is sufficient that the distribution coefficients are bounded for a scheme to be $\mathcal{LP}$ [22]. If the exact solution of a linear problem is assigned at the vertices, then the fluctuation in the element will be determined as $\phi^E = 0$. Bounded distribution coefficients will ensure that the solution does not change since $\phi_i^E = \beta_i \phi^E = 0$.

2.2 Linear $\mathcal{RD}$ Schemes

A number of linear $\mathcal{RD}$ schemes are presented next. Linear schemes cannot be both $\mathcal{P}$ and $\mathcal{LP}$. They may or may not possess property $\mathcal{U}$.

2.2.1 N Scheme

The N scheme is a linear scheme that preserves monotonicity. The N scheme was originally proposed by Roe [50] and is formulated by ensuring that positivity is preserved [20, 43, 57]. It is one of the most important schemes since it serves as a basis for most nonlinear schemes that are also $\mathcal{LP}$.

Consider three separate equations describing the updates to the vertices of an element,

$$\Omega_1 \frac{\mathrm{d}u_1}{\mathrm{d}t} = -\beta_1 \left(k_1 u_1 + k_2 u_2 + k_3 u_3\right) \tag{2.11a}$$

$$\Omega_2 \frac{\mathrm{d}u_2}{\mathrm{d}t} = -\beta_2 \left(k_1 u_1 + k_2 u_2 + k_3 u_3\right) \tag{2.11b}$$

$$\Omega_3 \frac{\mathrm{d}u_3}{\mathrm{d}t} = -\beta_3 \left(k_1 u_1 + k_2 u_2 + k_3 u_3\right) , \tag{2.11c}$$

with the fluctuation written using (2.4). For a type I triangle, as shown in Fig. 2.4, all the fluctuation is sent to the downwind vertex; from (2.11), $\beta_1 = \beta_2 = 0$ and $\beta_3 = 1$. With an explicit-Euler time-marching algorithm, (2.11c) becomes

$$u_3^{n+1} = u_3^n - \frac{h}{\Omega_3} \left(k_1 u_1^n + k_2 u_2^n + k_3 u_3^n\right) . \tag{2.12}$$

The inflow parameters k_1 and k_2 are negative yielding positive coefficients for u_1^n and u_2^n. The coefficient for u_3^n will be positive under an appropriate time-step constraint. The time step is determined independently for each sub-element E that connects to vertex 3, a more restrictive condition for enforcing $\mathcal{P}$ than considering together all the updates from all triangles that connect to vertex 3. In (2.12), the update from a triangle is applied to the area, $\Omega^E/3$, contributed by the triangle to the dual mesh, resulting in the time-step constraint

$$h^E = \frac{1}{3}\frac{\Omega^E}{k_3} \, . \tag{2.13}$$

As noted by Struijs [57], the distribution in one target triangles is bounded and satisfies both properties $\mathcal{P}$ and $\mathcal{LP}$.

For type II triangles (see Fig. 2.4) the fluctuation must be split between the two downstream vertices, 1 and 2. Using the fact that $\sum_{i=1,i\in E}^{d+1} k_i = 0$, the fluctuation can be written $\phi^E = k_1 \left(u_1 - u_3\right) + k_2 \left(u_2 - u_3\right)$. With an explicit-Euler time-marching algorithm, (2.11a) and (2.11b) become

$$\begin{aligned} u_1^{n+1} &= u_1^n - \frac{h}{\Omega_1} k_1 \left(u_1 - u_3\right) \\ u_2^{n+1} &= u_2^n - \frac{h}{\Omega_2} k_2 \left(u_2 - u_3\right) \, , \end{aligned} \tag{2.14}$$

where β_1 and β_2 only have an implicit meaning and, for (2.11c), $\beta_3 = 0$. Since $k_1, k_2 > 0$, the coefficients for u_3 are positive. With the same considerations as for the type I triangle, the coefficients for u_1 and u_2 are positive under the time-step constraint

$$h^E = \frac{1}{3}\frac{\Omega^E}{\max(k_1, k_2)} \, . \tag{2.15}$$

Equation (2.14) is actually derived from the more general distribution

$$u_1^{n+1} = u_1^n - \frac{h}{\Omega_1}\left[k_1 \left(u_1 - u_3\right) + p_1 \left(u_1 - u_3\right) - p_2 \left(u_2 - u_3\right)\right] \tag{2.16}$$

$$u_2^{n+1} = u_2^n - \frac{h}{\Omega_2}\left[k_2 \left(u_2 - u_3\right) - p_1 \left(u_1 - u_3\right) + p_2 \left(u_2 - u_3\right)\right] \, . \tag{2.17}$$

In [57], it is shown that the largest time step of a positive update is achieved if $p_1 = p_2 = 0$, yielding (2.14). Note that in the two-target case, the distribution coefficients are given

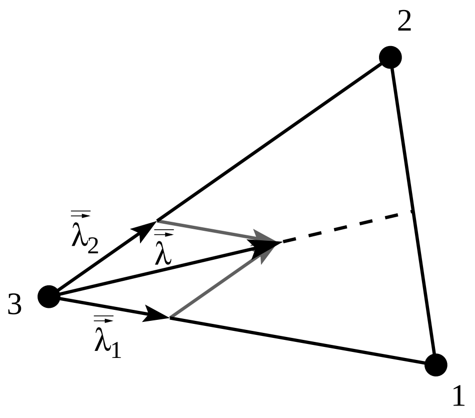


Figure 2.5 Geometrical distribution of the N scheme for a type II triangle.

by $\beta_i = \phi_i/\phi^E$ and become unbounded as $\phi^E \to 0$. Hence, for a type II triangle, the N scheme is not $\mathcal{LP}$.

A general formulation for the N scheme is given by

$$\phi_j^N = k_j^+ \left(u_j - u_{in}\right) , \tag{2.18}$$

where u_{in} is from (2.6). For an explicit-Euler update, a time-step restriction enforcing local positivity is

$$h^E = \frac{1}{3}\frac{\Omega^E}{\max\left(k_j^+\right)} , \quad j \in E . \tag{2.19}$$

Geometrically, the distribution of the N scheme in a type II triangle is equivalent to splitting the advection vector into two components parallel to edges 1 and 2 (see Fig. 2.5). Because of this, vertex 2, for example, will have no influence on the distribution to vertex 1; $\vec{\bar{\lambda}}_1$, being parallel to edge 2 results in $k_2 = 0$:

$$\begin{aligned}\phi_1 &= k_1|_{\vec{\bar{\lambda}}_1} u_1 + k_2|_{\vec{\bar{\lambda}}_1} u_2 + k_3|_{\vec{\bar{\lambda}}_1} u_3 \\ &= k_1|_{\vec{\bar{\lambda}}_1} u_1 + k_3|_{\vec{\bar{\lambda}}_1} u_3 \\ &= k_1|_{\vec{\bar{\lambda}}_1} \left(u_1 - u_3\right) .\end{aligned}$$

Note that $k_1 = k_1|_{\vec{\bar{\lambda}}_1}$, reproducing (2.14). This leads to the very narrow stencil for which the N scheme is named. In Fig. 2.6, while triangles $(1,2,3)$, $(1,3,4)$, and $(1,4,5)$ are involved in the update to vertex 1, the update is written as a function of only vertices 1, 3, and 4. The nature of the scheme excludes the outermost vertices, 2 and 5, which are perpendicular to the advection vector.

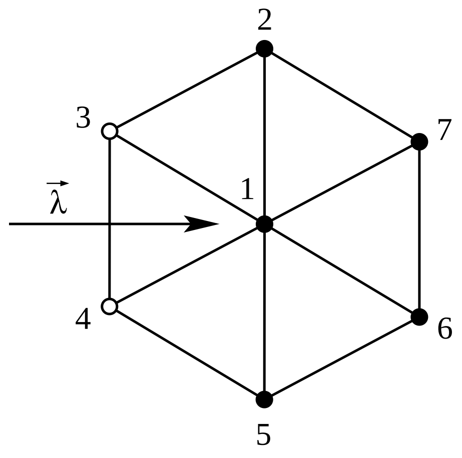


Figure 2.6 Vertices involved in an N-scheme update.

Formulation of the N scheme was based on the use of (2.4) which requires the conservative linearization for $\vec{\bar{\lambda}}$ in (2.3). A conservative linearization may be undesirable or even impossible for some equations; an alternative is to use a contour-integration-based $\mathcal{RD}$ (CRD) scheme [19]. In this technique, the fluctuation is computed via a contour integral given by

$$\phi^E = -\oint_{\partial E} \vec{f}(S) \cdot \hat{n}\, \mathrm{d}\mathcal{S}\,, \tag{2.20}$$

where $\vec{f}$ is the flux vector. In a numerical implementation, a Gauss quadrature integration rule is used to evaluate the integral. In the subsequent distribution step, any set of variables S may be used for the linearization. Since it may no longer be true that $\phi^E = \sum_{i=1,i\in E}^{d+1} k_i u_i$, the N scheme must be modified to ensure conservation. For substitution into (2.18), equation (2.6) is replaced by

$$u_{in} = \frac{\phi^E - \displaystyle\sum_{i=1,i\in E}^{d+1} k_i^+ u_i}{\displaystyle\sum_{i=1,i\in E}^{d+1} k_i^-}\,. \tag{2.21}$$

Note that if a conservative linearization is still used, this modification does not change anything because $\sum_{i=1,i\in E}^{d+1} k_i^- u_i = \phi^E - \sum_{i=1,i\in E}^{d+1} k_i^+ u_i$. Although the CRD technique is rather simple, the effects are profound because $\mathcal{RD}$ solutions are no longer restricted to a particular linearization. For scalar equations solved in this study, the CRD technique is used with a linearization based on

$$S = u\,. \tag{2.22}$$

2.2.2 LDA Scheme

The low diffusion A (LDA) scheme is a linear scheme which is $\mathcal{LP}$ but not $\mathcal{P}$. As with the N scheme, all the fluctuation in sent to the downstream vertex in a type I triangle. In a type II triangle, shown in Fig. 2.7, the distribution is governed by the location at which the linearized characteristic vector intersects the outflow edge. This identifies the distribution point defined by Fig. 2.2. In the example of Fig. 2.7, $\beta_1 = L_1/L$ and $\beta_2 = L_2/L$. A general formula for the LDA scheme is

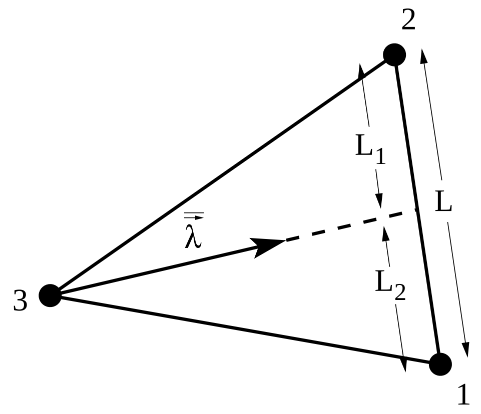


Figure 2.7 Geometrical distribution of the LDA scheme for a type II triangle.

$$\beta_i^{LDA} = \frac{k_i^+}{\displaystyle\sum_{j=1, j\in E}^{d+1} k_j^+}\,. \tag{2.23}$$

There is also an LDB scheme in which the splitting of ϕ^E is based on angles [43, 57] but this scheme has fallen out of favour and is not considered here.

If a non-conservative linearization is employed and the fluctuation is computed by a contour integral, (2.20), no modifications are required for the LDA scheme since $\sum_{i=1,i\in E}^{d+1} \beta_i^{LDA} = 1$, irrespective of the linearization, and therefore the distributed fluctuation is conserved.

Taylor-Series Analysis of the LDA Scheme

The LDA scheme performs very well on smooth flows and provides the expected spatial order of accuracy. In some cases, super-convergence is observed, a property which can be

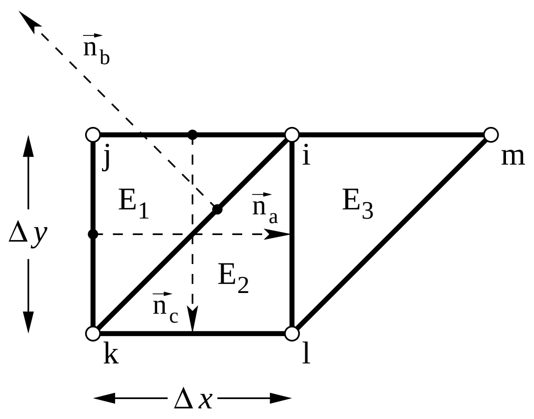


Figure 2.8 Mesh elements used for Taylor series expansion about vertex i.

understood by examining the cross-diffusion of a Taylor-series expansion of the solution around a vertex. This type of analysis is performed extensively by Paillère [43] at $O(\Delta h^3)$ truncation for both the LDA and N scheme, with h being representative of the grid spacing. The technique is repeated here at $O(\Delta h^4)$ truncation to investigate the super-convergence. Taylor series expansion will also be used in Chapter 6 to study the accuracy of nonlinear schemes.

A grid of three elements is considered and shown in Fig. 2.8. The update for vertex i is considered and the downstream elements above vertex i can be ignored if the y-component of the advection vector is assumed positive. The scaled inwards normals are shown for E_1 and have values

$$\vec{n}_a = \begin{bmatrix} \Delta y \\ 0 \end{bmatrix} \quad \vec{n}_b = \begin{bmatrix} -\Delta y \\ \Delta x \end{bmatrix} \quad \vec{n}_c = \begin{bmatrix} 0 \\ -\Delta x \end{bmatrix}.$$

The fluctuations for each element may be expressed as

$$\begin{aligned} \phi^{E_1} &= k_a u_i + k_b u_j + k_c u_k \\ \phi^{E_2} &= -k_c u_i - k_a u_k - k_b u_l \\ \phi^{E_3} &= k_b u_i + k_c u_l + k_a u_m \end{aligned} \tag{2.24}$$

where

$$k_a = \frac{1}{2} a\Delta y \qquad k_b = \frac{1}{2}\left(b\Delta x - a\Delta y\right) \qquad k_c = -\frac{1}{2} b\Delta x\,.$$

Using a Taylor series expansion, all nodal values are expressed relative to u_i as follows:

$$\begin{aligned}
u_j &= u_i - \Delta x\frac{\partial u}{\partial x} + \frac{\Delta x^2}{2}\frac{\partial^2 u}{\partial x^2} - \frac{\Delta x^3}{6}\frac{\partial^3 u}{\partial x^3} + O(\Delta h^4)\\
u_k &= u_i - \Delta x\frac{\partial u}{\partial x} - \Delta y\frac{\partial u}{\partial y} + \frac{\Delta x^2}{2}\frac{\partial^2 u}{\partial x^2} + \Delta x\Delta y\frac{\partial^2 u}{\partial x\partial y} + \frac{\Delta y^2}{2}\frac{\partial^2 u}{\partial y^2}\\
&\quad - \frac{\Delta x^3}{6}\frac{\partial^3 u}{\partial x^3} - \frac{\Delta x^2\Delta y}{2}\frac{\partial^3 u}{\partial x^2\partial y} - \frac{\Delta x\Delta y^2}{2}\frac{\partial^3 u}{\partial x\partial y^2} - \frac{\Delta y^3}{6}\frac{\partial^3 u}{\partial y^3} + O(\Delta h^4) \qquad (2.25)\\
u_l &= u_i - \Delta y\frac{\partial u}{\partial y} + \frac{\Delta y^2}{2}\frac{\partial^2 u}{\partial y^2} - \frac{\Delta y^3}{6}\frac{\partial^3 u}{\partial y^3} + O(\Delta h^4)\\
u_m &= u_i + \Delta x\frac{\partial u}{\partial x} + \frac{\Delta x^2}{2}\frac{\partial^2 u}{\partial x^2} + \frac{\Delta x^3}{6}\frac{\partial^3 u}{\partial x^3} + O(\Delta h^4)\,.
\end{aligned}$$

Substituting (2.25) into (2.24) results in

$$\begin{aligned}
\phi^{E_1} &= \frac{\Delta x\Delta y}{2}\left(a\frac{\partial u}{\partial x} + b\frac{\partial u}{\partial y} - \frac{a\Delta x}{2}\frac{\partial^2 u}{\partial x^2} - b\Delta x\frac{\partial^2 u}{\partial x\partial y} - \frac{b\Delta y}{2}\frac{\partial^2 u}{\partial y^2}\right.\\
&\qquad \left. + \frac{a\Delta x^2}{6}\frac{\partial^3 u}{\partial x^3} + \frac{b\Delta x^2}{2}\frac{\partial^3 u}{\partial x^2\partial y} + \frac{b\Delta x\Delta y}{2}\frac{\partial^3 u}{\partial x\partial y^2} + \frac{b\Delta y^2}{6}\frac{\partial^3 u}{\partial y^3}\right) + O(\Delta h^5)\\
\phi^{E_2} &= \frac{\Delta x\Delta y}{2}\left(a\frac{\partial u}{\partial x} + b\frac{\partial u}{\partial y} - \frac{a\Delta x}{2}\frac{\partial^2 u}{\partial x^2} - a\Delta y\frac{\partial^2 u}{\partial x\partial y} - \frac{b\Delta y}{2}\frac{\partial^2 u}{\partial y^2}\right. \qquad (2.26)\\
&\qquad \left. + \frac{a\Delta x^2}{6}\frac{\partial^3 u}{\partial x^3} + \frac{a\Delta x\Delta y}{2}\frac{\partial^3 u}{\partial x^2\partial y} + \frac{a\Delta y^2}{2}\frac{\partial^3 u}{\partial x\partial y^2} + \frac{b\Delta y^2}{6}\frac{\partial^3 u}{\partial y^3}\right) + O(\Delta h^5)\\
\phi^{E_3} &= \frac{\Delta x\Delta y}{2}\left(a\frac{\partial u}{\partial x} + b\frac{\partial u}{\partial y} + \frac{a\Delta x}{2}\frac{\partial^2 u}{\partial x^2} - \frac{b\Delta y}{2}\frac{\partial^2 u}{\partial y^2} + \frac{a\Delta x^2}{6}\frac{\partial^3 u}{\partial x^3} + \frac{b\Delta y^2}{6}\frac{\partial^3 u}{\partial y^3}\right) + O(\Delta h^5)\,.
\end{aligned}$$

A transformation is made from Cartesian coordinates, (x, y), to a coordinate system aligned with the advection vector, (ξ, η), according to

$$\hat{\xi} = \frac{a}{|\lambda|}\hat{x} + \frac{b}{|\lambda|}\hat{y} \quad \text{and} \quad \hat{\eta} = -\frac{b}{|\lambda|}\hat{x} + \frac{a}{|\lambda|}\hat{y}\,.$$

Derivatives in the streamline coordinate system are given by the following relations:

$$\begin{aligned}
\frac{\partial u}{\partial x} &= \frac{1}{|\lambda|}\left[a\frac{\partial u}{\partial \xi} - b\frac{\partial u}{\partial \eta}\right] \\
\frac{\partial u}{\partial y} &= \frac{1}{|\lambda|}\left[b\frac{\partial u}{\partial \xi} + a\frac{\partial u}{\partial \eta}\right] \\
\frac{\partial^2 u}{\partial x^2} &= \frac{1}{|\lambda|^2}\left[a^2\frac{\partial^2 u}{\partial \xi^2} - 2ab\frac{\partial^2 u}{\partial \xi\partial \eta} + b^2\frac{\partial^2 u}{\partial \eta^2}\right] \\
\frac{\partial^2 u}{\partial x\partial y} &= \frac{1}{|\lambda|^2}\left[ab\left(\frac{\partial^2 u}{\partial \xi^2} - \frac{\partial^2 u}{\partial \eta^2}\right) + \left(a^2-b^2\right)\frac{\partial^2 u}{\partial \xi\partial \eta}\right] \\
\frac{\partial^2 u}{\partial y^2} &= \frac{1}{|\lambda|^2}\left[b^2\frac{\partial^2 u}{\partial \xi^2} + 2ab\frac{\partial^2 u}{\partial \xi\partial \eta} + a^2\frac{\partial^2 u}{\partial \eta^2}\right] \\
\frac{\partial^3 u}{\partial x^3} &= \frac{1}{|\lambda|^3}\left[a^3\frac{\partial^3 u}{\partial \xi^3} - 3a^2b\left(\frac{\partial^3 u}{\partial \xi^2\partial \eta} + \frac{\partial^3 u}{\partial \xi\partial \eta^2}\right) - b^3\frac{\partial^3 u}{\partial \eta^3}\right] \\
\frac{\partial^3 u}{\partial x^2\partial y} &= \frac{1}{|\lambda|^3}\left[a^2b\frac{\partial^3 u}{\partial \xi^3} + \left(a^3-2ab^2\right)\frac{\partial^3 u}{\partial \xi^2\partial \eta} + \left(b^3-2a^2b\right)\frac{\partial^3 u}{\partial \xi\partial \eta^2} + ab^2\frac{\partial^3 u}{\partial \eta^3}\right] \\
\frac{\partial^3 u}{\partial x\partial y^2} &= \frac{1}{|\lambda|^3}\left[ab^2\frac{\partial^3 u}{\partial \xi^3} - \left(b^3-2a^2b\right)\frac{\partial^3 u}{\partial \xi^2\partial \eta} + \left(a^3-2ab^2\right)\frac{\partial^3 u}{\partial \xi\partial \eta^2} - a^2b\frac{\partial^3 u}{\partial \eta^3}\right] \\
\frac{\partial^3 u}{\partial y^3} &= \frac{1}{|\lambda|^3}\left[b^3\frac{\partial^3 u}{\partial \xi^3} + 3ab^2\frac{\partial^3 u}{\partial \xi^2\partial \eta} + 3a^2b\frac{\partial^3 u}{\partial \xi\partial \eta^2} + a^3\frac{\partial^3 u}{\partial \eta^3}\right] .
\end{aligned}$$

Based on the exact solution, derivatives in the streamline direction are set to zero,

$$\frac{\partial()}{\partial \xi} = \frac{\partial()}{\partial \xi^2} = \frac{\partial()}{\partial \xi\partial \eta} = \frac{\partial()}{\partial \xi^3} = \frac{\partial()}{\partial \xi^2\partial \eta} = \frac{\partial()}{\partial \xi\partial \eta^2} = 0\,,$$

thereby simplifying the previous transformation derivatives. After substituting in the transformation derivatives, and letting $s = \Delta y/\Delta x$, the new expressions for the fluctuation in each element are

$$\begin{aligned}
\phi^{E_1} &= \frac{sab\Delta x^3}{4|\lambda|^2}\left[(b-as)\frac{\partial^2 u}{\partial \eta^2} + \frac{\Delta x}{3|\lambda|}(as-2b)(as-b)\frac{\partial^3 u}{\partial \eta^3}\right] + O(\Delta h^5) \\
\phi^{E_2} &= \frac{sab\Delta x^3}{4|\lambda|^2}\left[(as-b)\frac{\partial^2 u}{\partial \eta^2} + \frac{\Delta x}{3|\lambda|}(b-2as)(as-b)\frac{\partial^3 u}{\partial \eta^3}\right] + O(\Delta h^5) \qquad (2.27) \\
\phi^{E_3} &= \frac{sab\Delta x^3}{4|\lambda|^2}\left[(b-as)\frac{\partial^2 u}{\partial \eta^2} + \frac{\Delta x}{3|\lambda|}(as+b)(as-b)\frac{\partial^3 u}{\partial \eta^3}\right] + O(\Delta h^5)\,.
\end{aligned}$$

With explicit-Euler time-marching, the update scheme is written

$$\frac{u_i^{n+1}-u_i^n}{\Delta t}+\frac{1}{\Omega_i}\sum_{p=1}^{3}\beta_i^{E_p}\phi^{E_p}=0 \quad \text{with} \quad \Omega_i=\Delta x\Delta y=s\Delta x^2\,, \tag{2.28}$$

and a converged steady solution is obtained when

$$\frac{1}{s\Delta x^2}\sum_{p=1}^{3}\beta_i^{E_p}\phi^{E_p}=0\,. \tag{2.29}$$

The truncation error, (TE), for the scheme is then given by

$$\begin{aligned}\text{TE}=\frac{ab(as-b)\Delta x}{4|\lambda|^2}&\left\{\left[-\beta_i^{E_1}+\beta_i^{E_2}-\beta_i^{E_3}\right]\frac{\partial^2 u}{\partial\eta^2}\right.\\ &\left.+\frac{\Delta x}{3|\lambda|}\left[\beta_i^{E_1}(as-2b)-\beta_i^{E_2}(2as-b)+\beta_i^{E_3}(as+b)\right]\frac{\partial^3 u}{\partial\eta^3}\right\}+O(\Delta h^3)\,.\end{aligned} \tag{2.30}$$

For an LDA scheme and an advection vector at an angle of $45^\circ \leq \delta \leq 90^\circ$ from the x-axis, one has

$$\beta_i^{E_1}=\frac{k_a}{-k_c}=\frac{as}{b}\,,\qquad \beta_i^{E_2}=1\,,\qquad \beta_i^{E_3}=\frac{k_b}{-k_c}=1-\frac{as}{b}\,, \tag{2.31}$$

and (2.30) reduces to

$$\text{TE}=\frac{ab(as-b)\Delta x^2}{12|\lambda|^3}\left[2b-4as\right]\frac{\partial^3 u}{\partial\eta^3}+O(\Delta h^3)\,. \tag{2.32}$$

Equation (2.32) only has terms greater than or equal to $O(\Delta h^2)$ indicating that the LDA scheme is at least second-order accurate. Additionally, the scheme is consistent since the truncation error tends to zero as $\Delta x \to 0$. It is also apparent that the LDA scheme is at least third-order accurate if either:

$a=0$	advection vector aligned with vertical mesh lines.
$b=0$	advection vector aligned with horizontal mesh lines.
$s=\frac{b}{a}$	advection vector aligned with mesh diagonal.
$s=\frac{1}{2}\frac{b}{a}$	unique cell aspect ratio.

The first three conditions describe the advection vector aligning with the mesh. The last condition is the most curious; if a Cartesian mesh with a regular triangulation has an aspect ratio $s = \frac{1}{2}\frac{b}{a}$, then the scheme will be third-order accurate. This accounts for the super-convergence that is sometimes observed with the LDA scheme and is a result of error cancellation between the three elements of Fig. 2.8.

2.2.3 LDC Scheme

It is possible to adapt the error cancellation of the super-convergent LDA scheme to mesh elements of any aspect ratio, resulting in a new distribution scheme we have labelled LDC. Starting from (2.30), the three relations

$$-\beta_i^{E_1} + \beta_i^{E_2} - \beta_i^{E_3} = 0 \tag{2.33a}$$

$$\beta_i^{E_1}(as - 2b) - \beta_i^{E_2}(2as - b) + \beta_i^{E_3}(as + b) = 0 \tag{2.33b}$$

$$\beta_i^{E_2} = 1 \tag{2.33c}$$

are used to determine the distribution coefficients of the new scheme. Full upwinding is assumed for type I triangles, such as E_2, thereby providing the last relation. Solving (2.33) for $\beta_i^{E_1}$ and $\beta_i^{E_3}$ gives

$$\beta_i^{E_1} = \frac{2}{3} + \frac{k_a}{3k_c} \qquad \text{and} \qquad \beta_i^{E_3} = \frac{2}{3} + \frac{k_b}{3k_c}\,. \tag{2.34}$$

A general distribution formula for type II triangles is

$$\beta_i^{LDC} = \begin{cases} \dfrac{2}{3} - \dfrac{k_i}{3\sum_{j=1,j\in E}^{d+1} k_j^+} & \text{if } i \text{ is a downstream vertex,} \\ 0 & \text{if } i \text{ is an upstream vertex.} \end{cases} \tag{2.35}$$

Compared to the LDA scheme, the distribution coefficients are bounded between $1/3 \leq L_i/L \leq 2/3$ (see Fig. 2.7) and, quite surprisingly, the relative weighting between the downstream vertices is *opposite* the weighting of the LDA scheme for a given advection vector. In other words, if the advection vector points towards vertex i in triangle E_1 of Fig. 2.8, the LDC scheme will set $\beta_j^{E_1} > \beta_i^{E_1}$.

The LDC scheme is quite similar to a high-order technique proposed by Hubbard and Laird [32] where the stencil for the distribution is expanded such that an element can distribute its own fluctuation to vertices that it does not share. Both this technique and the LDC scheme are derived from Taylor-series expansions and both require a regular grid. The main difference is that the LDC scheme still retains a stencil of one element for distribution of the fluctuation.

The LDC scheme is only of academic interest. It relies upon a mesh with a regular structure and a generic formula for both type I and type II triangles is cumbersome. Nevertheless, the LDC scheme is a demonstration of an upwind method that achieves an extra order of accuracy on certain meshes by error cancellation. The LDC scheme is a linear scheme which, similar to the LDA scheme, is $\mathcal{LP}$ but not $\mathcal{P}$.

2.2.4 Central Scheme

A central scheme is given by $\beta_i = 1/3$. Although $\mathcal{LP}$, the central scheme respects neither of the $\mathcal{U}$ or $\mathcal{P}$ properties. As such, it is notoriously unstable for problems of wave propagation. This scheme is identical to Jameson's central finite-volume scheme [34] and the Galerkin finite-element method. It is only considered in this thesis as a basis for the construction of certain nonlinear schemes described later in Chapter 6.

2.2.5 Lax-Wendroff Scheme

The formulation of the Lax-Wendroff (LW) scheme in a $\mathcal{RD}$ framework is described in [35, 43, 50, 57] and recalled here for completeness. Two relations are required based on what amounts to mass lumping in the finite-element method [57]:

$$\phi^E = -\int_E \frac{\partial u}{\partial t} \mathrm{d}\Omega_E \Rightarrow \frac{\partial u}{\partial t} = -\frac{\phi^E}{\Omega_E}\,, \tag{2.36}$$

and

$$\int_D \left(u_i^{n+1} - u_i^n\right) \mathrm{d}\Omega_i = \left(u_i^{n+1} - u_i^n\right) \Omega_i\,, \tag{2.37}$$

where D is used to denote the dual mesh around vertex i having area Ω_i. Equation (2.36) defines the relation which expresses that $\partial u/\partial t$ is taken to be constant in a primary element and (2.37) states that the evolution of the solution in an element of the dual mesh is given by the change at node i.

The formulation starts with a Taylor series expansion in time around u_i^n,

$$u_i^{n+1} = u_i^n + \Delta t \frac{\partial u}{\partial t} + \frac{\Delta t^2}{2}\frac{\partial^2 u}{\partial t^2} + O(\Delta t^3) \tag{2.38}$$

$$u_i^{n+1} - u_i^n = \Delta t \frac{\partial u}{\partial t} + \frac{\Delta t^2}{2}\frac{\partial^2 u}{\partial t^2} + O(\Delta t^3)\,, \tag{2.39}$$

followed by integration over an element of the dual mesh,

$$\underbrace{\int_D \left(u_i^{n+1} - u_i^n\right) \mathrm{d}\Omega_i}_{\text{Use (2.37)}} = \int_D \Bigg(\underbrace{\Delta t \frac{\partial u}{\partial t}}_{\mathcal{A}} + \underbrace{\frac{\Delta t^2}{2}\frac{\partial^2 u}{\partial t^2}}_{\mathcal{B}} + \dots \Bigg) \mathrm{d}\Omega_i\,. \tag{2.40}$$

The temporal derivatives in the RHS are now replaced with spatial derivatives using the original equation. Consider a contribution to vertex i from primary element E. The term $\mathcal{A}$ in (2.40) is evaluated using (2.36),

$$\int_{E\cap D} \Delta t \frac{\partial u}{\partial t} \mathrm{d}\Omega_E = -\frac{1}{3}\left(\Delta t \frac{\phi^E}{\Omega_E}\Omega_E\right) = -\frac{1}{3}\Delta t \phi^E\,. \tag{2.41}$$

The relation

$$\frac{\partial^2 u}{\partial t^2} = -\bar{a}\frac{\partial^2 u}{\partial x \partial t} - \bar{b}\frac{\partial^2 u}{\partial y \partial t} = -\vec{\bar{\lambda}} \cdot \nabla \frac{\partial u}{\partial t}$$

is used to rewrite term $\mathcal{B}$ in (2.40) as

$$\begin{aligned}\int_{E\cap D} \frac{\Delta t^2}{2}\frac{\partial^2 u}{\partial t^2} \mathrm{d}\Omega_E = -\frac{\Delta t^2}{2}\vec{\bar{\lambda}} \cdot \int_{E\cap D} \nabla \frac{\partial u}{\partial t} \mathrm{d}\Omega_E = \frac{\Delta t^2}{2}\vec{\bar{\lambda}} \cdot \oint_{\partial E\cap D} \frac{\partial u}{\partial t} \hat{n} \mathrm{d}\mathcal{S} \\ = -\frac{\Delta t^2}{2}\frac{\phi^E}{\Omega_E}\vec{\bar{\lambda}} \cdot \oint_{\partial E\cap D} \hat{n} \mathrm{d}\mathcal{S} = -\frac{\Delta t^2}{2}\frac{\phi^E}{\Omega_E} k_i\,.\end{aligned} \tag{2.42}$$

The $\frac{1}{2}$ required to form k_i follows from an integration over the path CDE instead of path AB in Fig. 2.9 ($\vec{EC} = \frac{1}{2}\vec{AB}$). Substitution of (2.41) and (2.42) into (2.40) and summing over all primary elements, E, that connect to vertex i results in

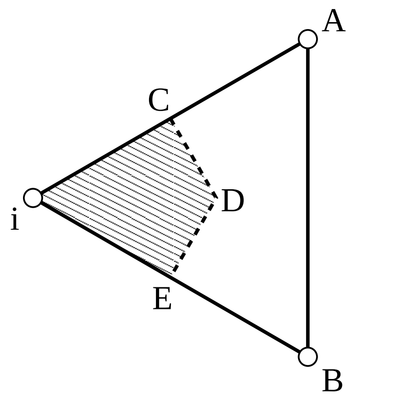


Figure 2.9 Integration path for (2.42).

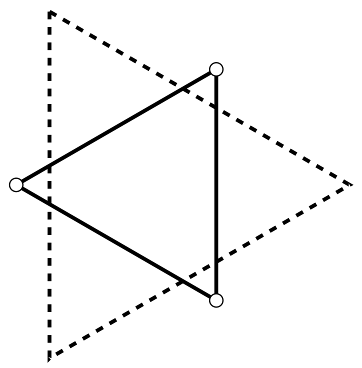


Figure 2.10 Geometrical representation of possible distributions for the LW scheme. The distribution point (see Fig. 2.2) may exist along the dashed lines.

$$\left(u_i^{n+1} - u_i^n\right) = -\frac{\Delta t}{\Omega_i} \sum_{E,i\in E} \left(\frac{1}{3} + \frac{\Delta t}{2\Omega_E} k_i\right) \phi^E \,, \tag{2.43}$$

or in semi-discrete form,

$$\Omega_i \frac{\mathrm{d}u_i}{\mathrm{d}t} + \sum_{E,i\in E} \left(\frac{1}{3} + \frac{\Delta t}{2\Omega_E} k_i\right) \phi^E = 0 \,. \tag{2.44}$$

From (2.44), it is obvious that the distribution coefficients for this "semi-discrete" form of the LW scheme are

$$\beta_i^{LW} = \frac{1}{3} + \frac{\Delta t}{2\Omega_E} k_i \,, \tag{2.45}$$

which is effectively a central scheme plus a dissipation term. In (2.45), Δt is a cell-based time step and in this work, it is obtained from

$$\Delta t = \frac{\Omega_E}{\max(k_j^+)}, \qquad j \in E \,, \tag{2.46}$$

(see (2.19)).

With the use of (2.46), the distribution coefficients are able to vary between $-2/3 \leq \beta_i^{LW} \leq 5/6$. The possible distributions, as defined by the distribution point in Fig. 2.2, are shown geometrically along the dashed lines of Fig. 2.10. The LW scheme is $\mathcal{LP}$ but not $\mathcal{P}$ or $\mathcal{U}$. Although usually not upwind, there is sufficient dissipation to ensure a convergent scheme.

2.2.6 Upwind Control Volume Scheme

The upwind control volume (UCV) scheme of Giles *et al.* [25] (and described in [43]) is similar to the LW scheme but with a modified dissipation term. The UCV scheme restricts the distribution coefficients so that they cannot be negative, essentially leading to an upwind formulation for type II triangles. For example, compare the possible distributions of Fig. 2.11 to those in Fig. 2.10. A generic formula for the UCV scheme is

$$\beta_i^{UCV} = \frac{1}{3} + \frac{k_i}{3\sum_{j=1,j\in E}^{d+1} k_j^+} \tag{2.47}$$

As with the LW scheme, the UCV scheme is $\mathcal{LP}$ but not $\mathcal{P}$ or $\mathcal{U}$.

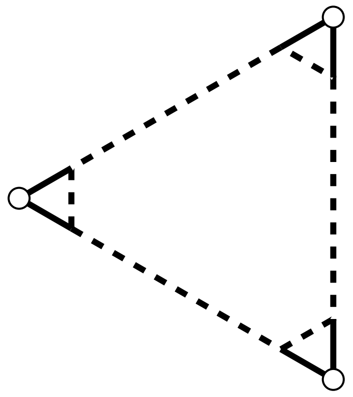


Figure 2.11 Possible distributions for the UCV scheme. The distribution point may exist along the dashed lines.

2.2.7 Accuracy of Linear Schemes

The performance of the linear schemes is shown in Fig. 2.12 for linear advection of a Gaussian profile (Fig. 1.3b). The L_1, L_2, and L_∞ error norms, computed on the line segment of the two finest meshes, are listed in Table 2.1. These results are illustrative of the interior scheme on a uniform Cartesian mesh that has been tessellated in the direction of the advection vector; boundary conditions are avoided by expanding the domain and imposing the exact solution around the exterior of the domain.

The N, LDA, and LDC schemes achieve the expected orders of accuracy. Interestingly, the LW and UCV schemes also indicate perfect third-order convergence, a result that is generally not indicated in the literature. However, the construction of the LW scheme

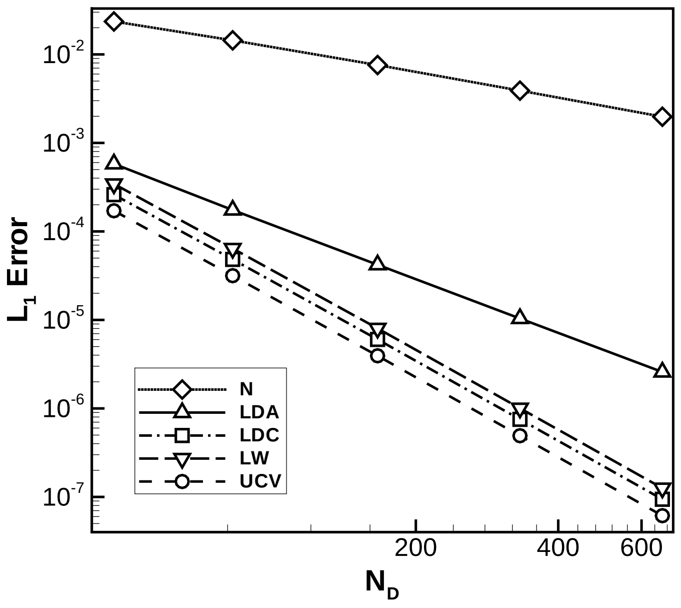


Figure 2.12 Accuracy of linear schemes (β in (1.3)) for linear advection of a Gaussian profile.

Table 2.1 Spatial convergence of linear schemes (β in (1.3)) for linear advection of a Gaussian profile.

Scheme	L_1-error	L_2-error	L_∞-error
N	-0.98	-0.97	-0.95
LDA	-2.01	-2.01	-2.00
LDC	-3.00	-3.00	-3.00
LW	-3.00	-3.00	-2.99
UCV	-3.00	-3.00	-3.00

is indeed at $O(\Delta h^3)$, and results obtained by Paillère [43] do indicate that the SUPG scheme, which is essentially the LW scheme with a unique cell-based time step, has better accuracy than the LDA scheme.

2.3 Nonlinear Schemes

Analogous to Godunov's theorem [26], only a nonlinear scheme can satisfy both the $\mathcal{P}$ and $\mathcal{LP}$ properties [22]. The nonlinear distribution schemes described next depend on and adapt themselves to the solution.

2.3.1 Limited N Scheme

The first scheme to satisfy all of properties $\mathcal{U}$, $\mathcal{P}$, and $\mathcal{LP}$ was the positive streamwise-invariant (PSI) scheme devised by Struijs [57]. In 1995, Sidilkover and Roe [54] made the observation that the PSI scheme is identical to a limited N (LN) scheme using a minmod limiter. The distribution coefficients for the N scheme are not bounded and may tend to $\pm\infty$ as $\phi^E \rightarrow 0$. The LN scheme is based upon the N scheme but limits the distribution coefficients when one of them (in the case of a type II triangle) becomes negative. If ϕ_1^N and ϕ_2^N denote the fluctuation that would be distributed to the two downstream nodes by the N scheme and $r = -\phi_1^N/\phi_2^N$, then the limited distribution given by the LN scheme is

$$\phi_1^{LN} = \phi_1^N \left[1 - \frac{\Psi(r)}{r}\right] , \qquad \phi_2^{LN} = \phi_2^N \left[1 - \Psi(r)\right] , \tag{2.48}$$

where $\Psi(r)$ is a symmetric limiter. Use of a minmod limiter,

$$\Psi(r) = \max\left[0, \min\left(r, 1\right)\right] , \tag{2.49}$$

reproduces the PSI scheme. Any symmetrical limiter may be used in the LN scheme but the minmod limiter is preferred because it is the only limiter that strictly maintains positivity [35, 43].

2.3.2 Blended Scheme

The blended scheme is a blending of the N and LDA schemes, $\phi_i^B = \theta\phi_i^N + (1-\theta)\phi_i^{LDA}$. There are several possible definitions of the blending coefficient θ; see [1] for a definition that reproduces the PSI scheme. In this work, the heuristic definition proposed by Deconinck *et al.* [22],

$$\theta = \frac{\left|\phi^E\right|}{\displaystyle\sum_{l=1, l\in E}^{d+1} \left|\phi_l^N\right| + \epsilon}\,, \qquad \epsilon = 10^{-10}\,, \tag{2.50}$$

is used where θ is defined to switch to the LDA scheme when divergence of the nodal fluctuation, as computed by the N-scheme, is detected. Positivity of this particular blended scheme has not been formally shown; however, numerical experiments generally produce solutions that are satisfactorily monotone. The blended scheme has the $\mathcal{LP}$ and $\mathcal{U}$ properties.

2.3.3 Map A Scheme

The mapped schemes proposed by Abgrall and co-workers [2, 4, 9] extend the limiting concepts of the LN scheme to multiple dimensions. Whereas the LN scheme is only applicable to a type II triangle, mapped schemes can be used at any number of dimensions and also if the distribution to be limited is not upwind. The latter may occur in the matrix distribution schemes to be discussed in the next chapter. Other than the oddity of limiting for accuracy instead of monotonicity, mapped schemes are very similar to the multidimensional limiting concepts discussed by Berger *et al.* [11]. However, the constraint or target of the limiting is more obvious since geometrically, it equals the perimeter of the element.

As with the LN scheme, the basis for a mapped scheme is the $\mathcal{P}$ distribution, β_i^N, predicted by the N scheme. Next an $\mathcal{LP}$ distribution, β^*, is sought that satisfies

$$\beta_i^N \beta_i^* \geq 0, \tag{2.51}$$

such that the sign of the distribution does not change and the $\mathcal{P}$ property is retained. A mapping which reproduces the PSI scheme [2], hereon referred to as "map A"[1], is

$$\beta_i^{\text{map}A} = \beta_i^* = \frac{\max(0, \beta_i^N)}{\sum_{j=1, j\in E}^{d+1} \beta_j^+} \,. \tag{2.52}$$

As shown in Fig. 2.13, this mapping is always a translation toward one of the vertices and essentially bounds the distribution to the perimeter of the triangle, ensuring that the distribution also has the $\mathcal{P}$ and $\mathcal{U}$ properties. In the case where the distribution point (see Fig. 2.2) predicted by the N scheme is within the perimeter of the element, no changes are made to the distribution.

An alternative mapping described by Abgrall and Roe [9] is to move the distribution point in an orthogonal direction towards an edge. Although this minimizes the distance the distribution point is moved, it also alters the ratio of the distribution coefficients. We label this mapping "map B" but it is not considered in the remainder of this thesis.

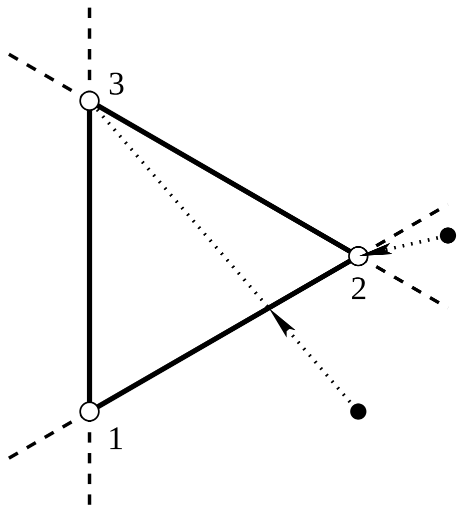


Figure 2.13 The map A scheme translates a distribution point outside the element towards a vertex until the perimeter of the element is reached.

2.3.4 Accuracy of Nonlinear Schemes

The performance of the nonlinear schemes at the interior is shown in Fig. 2.14 for the same problem used to analyze the linear schemes. The L_1, L_2, and L_∞ error norms, computed on the line segment of the two finest meshes, are listed in Table 2.2.

As expected by design, the LN scheme with a minmod limiter is identical to the map A scheme. More interesting is the performance of the blended scheme which follows the LN (minmod) scheme very closely even though the construction is significantly different.

[1]This mapping scheme, being a reinterpretation of the PSI scheme, is often referred to as PSI in the literature.

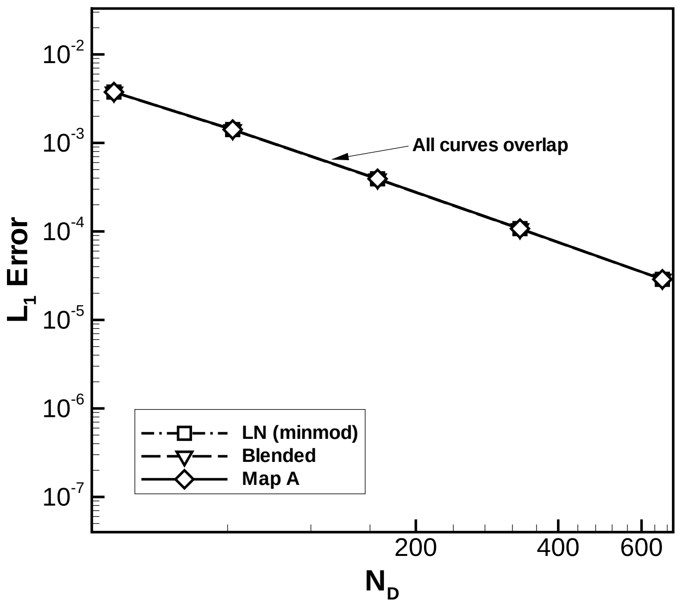


Figure 2.14 Accuracy of nonlinear schemes (β in (1.3)) for linear advection of a Gaussian profile.

Table 2.2 Spatial convergence of nonlinear schemes (β in (1.3)) for linear advection of a Gaussian profile.

Scheme	L_1-error	L_2-error	L_∞-error
LN (minmod)	-1.90	-1.67	-1.32
Blended	-1.89	-1.67	-1.32
Map A	-1.90	-1.67	-1.32

The fact that the spatial accuracy is less than second-order is examined more thoroughly in the chapters that follow.

Chapter 3

Residual Distribution Methods for Systems of Equations

The scalar $\mathcal{RD}$ techniques are extended in this chapter to systems of equations. The Euler system of equations is used to present the various methods.

3.1 Theory for Systems of Equations

There are currently two approaches to distributing the fluctuation for a system: system decomposition and matrix schemes [22]. The most physically satisfactory approach is to decompose the system into scalar equations. This is achieved via hyperbolic-elliptic splitting where, for the steady Euler equations, the addition of a preconditioner allows for diagonalization of the equations in characteristic form [35, 43, 46]. Matrix schemes are a generalization of scalar techniques to matrix-vector equations. Although not as intuitive nor physically meaningful as equation decomposition techniques, the resulting schemes can still provide very accurate results.

In quasi-linear form, (1.1) with (1.6) becomes

$$\frac{\partial \mathbf{U}}{\partial t} + \sum_{j=1}^{d} \left(\mathbf{A}_{U,j} \frac{\partial \mathbf{U}}{\partial x_j} \right) = 0\,, \tag{3.1}$$

where $\mathbf{A}_{U,j} = \partial \mathbf{F}_j / \partial \mathbf{U}$ are the conservative-variable flux Jacobians.

For systems of equations, the scalar inflow parameters, k_i, of the residual distribution schemes become matrices, $\mathbf{K}_i$, since the linearized scalar wave speed, $\bar{\lambda}_j$, of (2.3) is now a matrix that depends on the linearized flux Jacobians

$$\bar{\mathbf{A}}_{U,j} = \left.\frac{\partial \mathbf{F}_j}{\partial \mathbf{U}}\right|_{\bar{\mathbf{S}}} . \tag{3.2}$$

The notation of (3.2) denotes that the Jacobian is linearized at the average state $\bar{\mathbf{S}} = \frac{1}{d+1}\sum_{i,i\in E}^{d+1} \mathbf{S}_i$ of a set of, as yet undefined, solution variables $\mathbf{S}(\mathbf{U})$. Equation (2.5) then has the form

$$\mathbf{K}_i = \frac{1}{d}\left[\sum_{j=1}^{d} \left(\bar{\mathbf{A}}_j \hat{x}_j\right)\right] \cdot \vec{n}_i \, . \tag{3.3}$$

Having matrices as inflow parameters creates two primary issues. The first is to find a method of linearization, i.e., a definition of $\mathbf{S}$, that is still conservative. The second is the technique for computing the distribution coefficients, $\boldsymbol{\beta}_i$, which are now matrices.

3.1.1 Linearization

A conservative linearization has been found for the Euler equations. The Roe-Struijs-Deconinck (RSD) linearization [57] is an extension of Roe's parameter vector [49] to multiple dimensions $\mathbf{S} = \mathbf{Z} = [\sqrt{\rho}, \sqrt{\rho}u, \sqrt{\rho}v, \sqrt{\rho}h]^T$. This linearization ensures that a second-order conservative fluctuation may be computed as

$$\boldsymbol{\phi}_U^E = \sum_{i=1,i\in E}^{d+1} \mathbf{K}_{U,i}\tilde{\mathbf{U}}_i \, , \tag{3.4}$$

where $\tilde{\mathbf{U}}_i = \frac{\partial \mathbf{U}}{\partial \mathbf{S}}\big|_{\bar{\mathbf{S}}} \mathbf{S}_i$. Use of a conservative linearization leads to a linearization-based $\mathcal{RD}$ (LRD) scheme. Unfortunately, conservative linearizations are not available for all systems of equations. Also, in some cases the use of $\mathbf{Z}$ as the linearized state may be undesirable. In particular, non-existent pressure gradients may be detected for shear flows [35].

The alternative to using a conservative linearization is to use the CRD technique described in section 2.2.1. For the Euler equations, the fluctuation is computed via a contour integral given by

$$\phi_U^E = -\oint_{\partial E} \vec{\mathbf{F}}(\mathbf{S}) \cdot \hat{n}\, \mathrm{d}\mathcal{S}\,, \tag{3.5}$$

where, again, a Gauss quadrature integration rule is used to evaluate the integral. The modifications required for the matrix N scheme will be noted in the following section on matrix distribution. The Euler results in this study are all obtained using a CRD scheme with a linearization based on the primitive variables given by

$$\mathbf{S} = \mathbf{V} = [\rho, u, v, p]^T\,. \tag{3.6}$$

3.2 Hyperbolic/Elliptic Splitting

Hyperbolic-elliptic splitting provides a method for decomposing the Euler equations. Supersonic flows decouple into four scalar equations while subsonic flows decouple into two scalar equations plus an acoustic subset. In this work, only the details relevant to the implementation are described. Details regarding the development of the related preconditioner are available elsewhere [35, 43, 46].

For simplicity, the Euler equations are expressed in a coordinate system aligned with the streamline, (ξ, η), and in terms of a particular set of symmetrizing variables, $\breve{\mathbf{Q}}$, as follows

$$\frac{\partial \breve{\mathbf{Q}}}{\partial t} + \mathbf{A}_{\breve{Q}} \frac{\partial \breve{\mathbf{Q}}}{\partial \xi} + \mathbf{B}_{\breve{Q}} \frac{\partial \breve{\mathbf{Q}}}{\partial \eta} = 0\,, \tag{3.7}$$

before applying the preconditioner. The symmetrizing variables have the form

$$\partial \breve{\mathbf{Q}} = \begin{bmatrix} \partial p/(\rho a) & \partial q & q\partial\theta & \partial S \end{bmatrix}^T\,, \tag{3.8}$$

where q is the flow speed

$$q = \sqrt{u^2 + v^2}\,, \qquad \partial q = \frac{u\partial u + v\partial v}{q}\,, \tag{3.9}$$

θ is the local flow direction

$$\theta = \tan^{-1}\frac{v}{u}, \qquad \partial\theta = \frac{u\partial v - v\partial u}{q^2}\,, \tag{3.10}$$

and ∂S is proportional to the change in entropy

$$\partial S = \partial p - a^2 \partial \rho \,. \tag{3.11}$$

With symmetrizing variables, the Jacobians have the form

$$\mathbf{A}_{\breve{Q}} = \begin{bmatrix} q & a & 0 & 0 \\ a & q & 0 & 0 \\ 0 & 0 & q & 0 \\ 0 & 0 & 0 & q \end{bmatrix} , \qquad \mathbf{B}_{\breve{Q}} = \begin{bmatrix} 0 & 0 & a & 0 \\ 0 & 0 & 0 & 0 \\ a & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} . \tag{3.12}$$

The preconditioner, $\mathbf{P}$, is added such that

$$\frac{\partial \breve{\mathbf{Q}}}{\partial t} + \mathbf{P} \left(\mathbf{A}_{\breve{Q}} \frac{\partial \breve{\mathbf{Q}}}{\partial \xi} + \mathbf{B}_{\breve{Q}} \frac{\partial \breve{\mathbf{Q}}}{\partial \eta} \right) = 0 \,. \tag{3.13}$$

Note that the preconditioning does not affect steady-state solutions. This system may be re-written as

$$\mathbf{A}_{\breve{Q}}^{-1} \mathbf{P}^{-1} \frac{\partial \breve{\mathbf{Q}}}{\partial t} + \frac{\partial \breve{\mathbf{Q}}}{\partial \xi} + \mathbf{A}_{\breve{Q}}^{-1} \mathbf{B}_{\breve{Q}} \frac{\partial \breve{\mathbf{Q}}}{\partial \eta} = 0 \,. \tag{3.14}$$

The matrix $\mathbf{A}_{\breve{Q}}^{-1} \mathbf{B}_{\breve{Q}}$ can be diagonalized as $\mathbf{\Lambda} = \mathbf{L} \mathbf{A}_{\breve{Q}}^{-1} \mathbf{B}_{\breve{Q}} \mathbf{R}$ and thereby defining the characteristic variables $\partial \mathbf{W} = \mathbf{L} \partial \breve{\mathbf{Q}}$. After diagonalization and a change of variables from $\breve{\mathbf{Q}}$ to $\mathbf{W}$, (3.14) assumes the form

$$\mathbf{L} \mathbf{A}_{\breve{Q}}^{-1} \mathbf{P}^{-1} \mathbf{R} \frac{\partial \mathbf{W}}{\partial t} + \frac{\partial \mathbf{W}}{\partial \xi} + \mathbf{\Lambda} \frac{\partial \mathbf{W}}{\partial \eta} = 0 \,. \tag{3.15}$$

The preconditioner is defined to diagonalize the matrix $\mathbf{D} = \mathbf{L} \mathbf{A}_{\breve{Q}}^{-1} \mathbf{P}^{-1} \mathbf{R}$, completely for supersonic flows and as much as possible for subsonic flows. A preconditioner, valid for both subsonic and supersonic flows, can be written as

$$\mathbf{P} = \begin{bmatrix} \frac{\alpha \chi M^2}{\beta} & -\frac{\alpha \chi M}{\beta} & 0 & 0 \\ -\frac{\alpha \chi M}{\beta} & \frac{\alpha \chi}{\beta} + \alpha & 0 & 0 \\ 0 & 0 & \beta \chi & 0 \\ 0 & 0 & 0 & \alpha \end{bmatrix} , \tag{3.16}$$

where $\beta = \sqrt{\max(\epsilon^2, |M^2 - 1|)}$ and $\chi = 1/\max(M, 1)$. To avoid problems at the sonic point, the parameter ϵ assumes a small value, typically 0.05 [43]. The quantity α was introduced by Mesaros [35] to reduce sensitivity to the flow angle in stagnation regions. It is defined as

$$\alpha = \begin{cases} \frac{1}{2}\,, & \text{for} \quad M \leq \frac{1}{3}\,, \\ \frac{1}{2} + \frac{27}{2}\left(M - \frac{1}{3}\right)^2 - 27\left(M - \frac{1}{3}\right)^3\,, & \text{for} \quad \frac{1}{3} < M < \frac{2}{3}\,, \\ 1\,, & \text{for} \quad M \geq \frac{2}{3}\,. \end{cases}$$

Equation (3.15) then takes the form

$$\frac{\partial \mathbf{W}}{\partial t} + \mathbf{A}_W \frac{\partial \mathbf{W}}{\partial \xi} + \mathbf{B}_W \frac{\partial \mathbf{W}}{\partial \eta} = 0\,, \tag{3.17}$$

where $\mathbf{A}_W = \mathbf{D}^{-1}$ and $\mathbf{B}_W = \mathbf{D}^{-1}\mathbf{\Lambda}$. The characteristic variables are given by

$$\partial \mathbf{W} = \begin{bmatrix} \partial S \\ \partial p + \rho q \partial q \\ \partial p + \frac{\rho q^2}{\beta} \partial \theta \\ \partial p - \frac{\rho q^2}{\beta} \partial \theta \end{bmatrix}. \tag{3.18}$$

A prescription of $\mathbf{A}_W$ and $\mathbf{B}_W$ that is valid for both supersonic and subsonic flows and encompasses a smooth transition between the two regimes is given by

$$\mathbf{A}_W = \begin{bmatrix} \alpha q & 0 & 0 & 0 \\ 0 & \alpha q & 0 & 0 \\ 0 & 0 & \frac{1}{2}\chi q \beta(\alpha\eta + 1) & \frac{1}{2}\chi q \beta(\alpha\eta - 1) \\ 0 & 0 & \frac{1}{2}\chi q \beta(\alpha\eta - 1) & \frac{1}{2}\chi q \beta(\alpha\eta + 1) \end{bmatrix}, \tag{3.19}$$

$$\mathbf{B}_W = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2}\chi q(\alpha + 1) & \frac{1}{2}\chi q(\alpha - 1) \\ 0 & 0 & \frac{1}{2}\chi q(\alpha - 1) & \frac{1}{2}\chi q(\alpha + 1) \end{bmatrix}, \tag{3.20}$$

where $\eta = (M^2 - 1)/\beta^2$. Note that η is defined as -1 for subsonic flows and $+1$ for supersonic flow but it smoothly transitions between the two values over the range $1-\epsilon^2 < M^2 < 1+\epsilon^2$. In supersonic flows, matrices (3.19) and (3.20) are diagonal and have the form

$$\mathbf{A}_w = \begin{bmatrix} q & 0 & 0 & 0 \\ 0 & q & 0 & 0 \\ 0 & 0 & a\beta & 0 \\ 0 & 0 & 0 & a\beta \end{bmatrix}, \qquad \mathbf{B}_w = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & a & 0 \\ 0 & 0 & 0 & -a \end{bmatrix}. \tag{3.21}$$

The following steps summarize the numerical implementation of the $\mathcal{RD}$ scheme resulting from the hyperbolic-elliptic equation decomposition procedure:

1. The fluctuation in conservative variables, $\boldsymbol{\phi}_U^E$, is computed using (3.5).

2. The average linearized state is determined and used to compute the transformation matrices $\frac{\partial \breve{\mathbf{Q}}}{\partial \mathbf{U}}$, $\frac{\partial \mathbf{U}}{\partial \breve{\mathbf{Q}}}$, $\frac{\partial \mathbf{W}}{\partial \breve{\mathbf{Q}}}$, $\frac{\partial \breve{\mathbf{Q}}}{\partial \mathbf{W}}$, and $\frac{\partial \mathbf{U}}{\partial \mathbf{V}}$ as well as the preconditioner $\mathbf{P}$.

3. The fluctuation is preconditioned and transformed into characteristic variables via
$$\boldsymbol{\phi}_W^E = \frac{\partial \mathbf{W}}{\partial \breve{\mathbf{Q}}} \mathbf{P} \frac{\partial \breve{\mathbf{Q}}}{\partial \mathbf{U}} \boldsymbol{\phi}_U^E .$$

4. The solution state at each vertex in the element is converted from the linear state (primitive variables in this work) to characteristic variables via
$$\tilde{\mathbf{W}}_i = \frac{\partial \mathbf{W}}{\partial \breve{\mathbf{Q}}} \frac{\partial \breve{\mathbf{Q}}}{\partial \mathbf{U}} \frac{\partial \mathbf{U}}{\partial \mathbf{V}} \mathbf{V}_i .$$

5. The fluctuation is distributed using scalar distribution everywhere except for the elliptic subset which arises from the acoustic equations in a subsonic flow. In this work, the elliptic subset is distributed using a Lax-Wendroff scheme [35] with
$$\boldsymbol{\beta}_i^{LW} = \frac{\hat{\mathbf{I}}}{3} + \frac{\tau}{2\Omega_E} \mathbf{K}_{W,i} , \tag{3.22}$$

where $\hat{\mathbf{I}}$ is the identity matrix and $\frac{\Omega_E}{\tau}$ is set to the maximum eigenvalue of matrix $\mathbf{K}_{W,i}$. The acoustic inflow parameter $\mathbf{K}_{W,i}$ is formulated using (3.3) but only using the 2×2 acoustic subsets of $\mathbf{A}_W$ and $\mathbf{B}_W$. Note that more sophisticated elliptic distribution techniques exist, but are not considered as part of this work. The most advanced is probably the least-squares minimization technique advocated by Rad [46] as described earlier in Chapter 1.

6. The fluctuation sent to each vertex is converted to conservative variables for a subsequent summation and update of the solution. At this point, there is the option to retain or remove the preconditioner [35, 43]. The most consistent method is to remove the preconditioner from the distributed element fluctuation and reapply it at the vertex. This conversion is performed as follows:

$$\boldsymbol{\phi}_{U,i} = \frac{\partial \mathbf{U}}{\partial \breve{\mathbf{Q}}} \mathbf{P}^{-1} \frac{\partial \breve{\mathbf{Q}}}{\partial \mathbf{W}} \boldsymbol{\phi}_{W,i} \, .$$

   This method is fully conservative but less computationally robust. The preconditioner may alternatively be retained in the fluctuation by the following conversion

$$\boldsymbol{\phi}_{U,i} = \frac{\partial \mathbf{U}}{\partial \breve{\mathbf{Q}}} \frac{\partial \breve{\mathbf{Q}}}{\partial \mathbf{W}} \boldsymbol{\phi}_{W,i} \, .$$

   This implies an assumption that the preconditioners in each of the elements sharing vertex i are sufficiently close in value. In other words, the flow is smooth. This approach is not conservative but numerical experiments indicate that it is more robust (about the same as a matrix $\mathcal{RD}$ scheme). The two approaches yield similar results, even in the vicinity of strong discontinuities, but the conservative method was found to be more accurate. In this work, the conservative approach is used for all the decomposed solutions.

7. The desired time-marching algorithm is applied to the discrete solution at each vertex governed by the semi-discrete form of the governing equations given by (2.9). For explicit time-marching, the preconditioner, computed at the vertex state, is

typically reapplied to the vertex residual if it was removed in the previous step. This ensures that the update scheme is consistent with the modified wave-speeds of the system and theoretically provides convergence that is almost independent of the Mach number for subsonic problems.

3.3 Matrix Distribution

Matrix distribution schemes are generalizations of scalar techniques [59]. The linear schemes considered here are all invariant under a similarity transformation, meaning that the same conservative fluctuation is sent to the vertices, irrespective of the variables in which the distribution is actually performed [22]. It is therefore beneficial to switch to the symmetrizing variables

$$\partial \mathbf{Q} = \begin{bmatrix} \partial p/(\rho a) & \partial u & \partial v & \partial S \end{bmatrix}^T , \tag{3.23}$$

where $\partial S = \partial p - a^2 \partial \rho$, when formulating the distribution scheme. The matrices $\mathbf{A}_Q$ and $\mathbf{B}_Q$ assume the form

$$\mathbf{A}_Q = \begin{bmatrix} u & a & 0 & 0 \\ a & u & 0 & 0 \\ 0 & 0 & u & 0 \\ 0 & 0 & 0 & u \end{bmatrix}, \qquad \mathbf{B}_Q = \begin{bmatrix} v & 0 & a & 0 \\ 0 & v & 0 & 0 \\ a & 0 & v & 0 \\ 0 & 0 & 0 & v \end{bmatrix}. \tag{3.24}$$

In symmetrizing form, the last equation completely decouples from the system. This is the entropy advection equation. As a result, only the upper 3×3 subset remains to be solved by matrix distribution. In what follows, the inflow parameters $\mathbf{K}_Q$ are formulated using (3.3) but only for the upper 3×3 subset of $\mathbf{A}_Q$ and $\mathbf{B}_Q$.

The following steps summarize the numerical implementation of the residual distribution scheme resulting from the matrix distribution procedure applied to the Euler equations:

1. The fluctuation in conservative variables, $\boldsymbol{\phi}_U^E$, is computed using (3.5).

2. The average linearized state is determined and used to compute the transformation matrices $\frac{\partial \mathbf{Q}}{\partial \mathbf{U}}$, $\frac{\partial \mathbf{U}}{\partial \mathbf{Q}}$, and $\frac{\partial \mathbf{U}}{\partial \mathbf{V}}$.

3. The fluctuation is transformed into symmetrizing variables via
$$\boldsymbol{\phi}_Q^E = \frac{\partial \mathbf{Q}}{\partial \mathbf{U}} \boldsymbol{\phi}_U^E \,.$$

4. The solution state at each vertex in the element is converted from the linear state (primitive variables in this work) to symmetrizing variables via
$$\tilde{\mathbf{Q}}_i = \frac{\partial \mathbf{Q}}{\partial \mathbf{U}} \frac{\partial \mathbf{U}}{\partial \mathbf{V}} \mathbf{V}_i \,.$$

5. The entropy equation is distributed using a scalar method and the remaining 3×3 subset by a matrix technique.

6. The fluctuation at each vertex is converted to conservative variables for the update
$$\boldsymbol{\phi}_{U,i} = \frac{\partial \mathbf{U}}{\partial \mathbf{Q}} \boldsymbol{\phi}_{Q,i} \,.$$

7. The desired time-marching algorithm is applied at each vertex using the semi-discrete update formula (2.9).

Matrix versions of the various $\mathcal{RD}$ schemes are now described. For all the schemes, except LW, the inflow parameters $\mathbf{K}_Q$ must first be split into positive and negative components. Matrix $\mathbf{K}_Q$ is diagonalized via $\boldsymbol{\Lambda} = \mathbf{L}\mathbf{K}_Q\mathbf{R}$. Using $\boldsymbol{\Lambda}^{\pm} = (\boldsymbol{\Lambda} \pm |\boldsymbol{\Lambda}|)/2$, the split inflow parameters are defined as $\mathbf{K}_Q^{\pm} = \mathbf{R}\boldsymbol{\Lambda}^{\pm}\mathbf{L}$. As the sum of the negative (or positive) distribution coefficients is an oft-required denominator, it is common to define

$$\mathbf{N}_Q = \left(\sum_{j=1, j\in E}^{d+1} \mathbf{K}_{Q,j}^{-} \right)^{-1} \tag{3.25}$$

for matrix schemes. At stagnation points, the matrix $\mathbf{N}_Q$ may become singular. Analytically, it has been shown that $\mathbf{K}_{Q,i}^{+}\mathbf{N}_Q$, the manner in which $\mathbf{N}_Q$ is used, always has meaning [1]. Numerically, a small modification is made to avoid the singularity. For the problems considered in this thesis (Fig. 1.4), a linearized stagnation region is never encountered in the interior. However, at inviscid wall boundaries, the normal component of the velocity is explicitly forced to zero and the inversion required in (3.25) can be difficult in the degenerate ghost elements used to impose boundary conditions. At stagnation points, $\sum_{j=1,j\in E}^{d+1} \mathbf{K}_{Q,j}^{-}$ assumes the form

$$\begin{bmatrix} s_{11} & 0 & 0 \\ 0 & s_{22} & s_{23} \\ 0 & s_{23} & s_{33} \end{bmatrix} .$$

and becomes problematic when $s_{22}s_{33} - s_{23}^2 = 0$. For the ghost elements, a simple modification to avoid the singularity is introduced:

$$K_{Q,i,22}^{+} = K_{Q,i,22}^{+} + \epsilon \qquad K_{Q,i,33}^{+} = K_{Q,i,33}^{+} + \epsilon \qquad K_{Q,i,22}^{-} = K_{Q,i,22}^{-} - \epsilon \qquad K_{Q,i,33}^{-} = K_{Q,i,33}^{-} - \epsilon ,$$

where ϵ assumes a small value, typically 1×10^{-6}. Note that this procedure has no effect on the overall value of $\mathbf{K}_{Q,i}$.

3.3.1 Linear Schemes

N Scheme

The matrix N scheme defines the fluctuation distributed to each node i of an element as

$$\boldsymbol{\phi}_{Q,i}^{N} = \mathbf{K}_{Q,i}^{+} \left(\tilde{\mathbf{Q}}_i - \mathbf{Q}_{in} \right) . \tag{3.26}$$

The modified state $\mathbf{Q}_{in}$ for the CRD scheme is

$$\mathbf{Q}_{in} = \mathbf{N} \left(\boldsymbol{\phi}_{Q}^{E} - \sum_{j=1,j\in E}^{d+1} \mathbf{K}_{Q,j}^{+} \tilde{\mathbf{Q}}_j \right) . \tag{3.27}$$

LDA Scheme

For the matrix LDA scheme, $\boldsymbol{\phi}_{Q,i}^{LDA} = \boldsymbol{\beta}_i^{LDA} \boldsymbol{\phi}_Q^E$ where

$$\boldsymbol{\beta}_i^{LDA} = -\mathbf{K}_{Q,i}^{+} \mathbf{N}_Q . \tag{3.28}$$

LW Scheme

The matrix LW scheme is given by (3.22). This is the only matrix scheme (considered in this thesis) that does not require the use of $\mathbf{N}_Q$.

UCV Scheme

The UCV scheme has a matrix formulation similar to that of the LW scheme:

$$\boldsymbol{\beta}_i^{UCV} = \frac{1}{3}\left(\hat{\mathbf{I}} + \mathbf{K}_{Q,i}\mathbf{N}_Q\right) . \tag{3.29}$$

Accuracy of Linear Matrix Schemes

An overview of linear matrix schemes is given by examining the performance of the interior scheme on a square subset of Ringleb's flow using a Cartesian mesh and a uniform tessellation. The results are also relevant to hyperbolic/elliptic splitting which uses the LW matrix scheme for the acoustic subset (in fact, both approaches use a combination of scalar and matrix techniques). Actual comparisons between hyperbolic/elliptic splitting and matrix distribution, as described earlier in this section, are presented in the next chapter for more practical problems. For this problem, the flow is the same as illustrated in Fig. 1.4e but only solved on the square domain extending from a bottom left corner of $(-0.35, 1)$ to a top right corner of $(0.45, 1.8)$. The flow in this subset is entirely subsonic. This test is only indicative of the interior scheme; vertices around the exterior of the domain are fixed to the exact solution. The results, illustrated in Fig. 3.1 and listed for several error norms in Table 3.1, show that all the $\mathcal{LP}$ schemes achieve similar second-order behaviour. For systems, no super-convergence is observed. The N scheme also behaves as expected giving first-order accuracy.

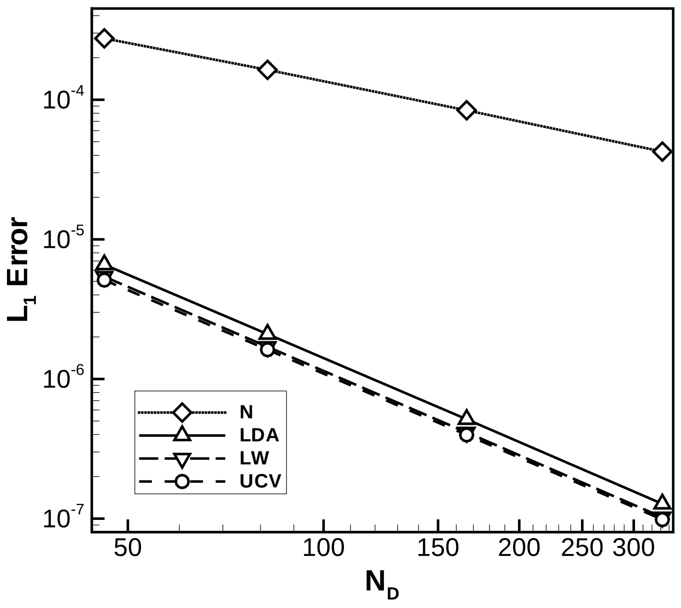


Figure 3.1 Accuracy of linear matrix schemes applied to a Cartesian grid on a subset of Ringleb's flow.

Table 3.1 Spatial convergence of linear matrix schemes applied to a Cartesian grid on a subset of Ringleb's flow.

Scheme	L_1-error	L_2-error	L_∞-error
N	-0.98	-0.98	-0.98
LDA	-2.01	-2.01	-2.04
LW	-2.02	-2.03	-2.00
UCV	-2.01	-2.02	-1.99

3.3.2 Nonlinear Schemes

Blended Scheme

For systems, use of the LN scheme is not straightforward so nonlinear distributions are commonly obtained by applying blends of the N and LDA schemes. The fluctuation distributed by the blended B scheme is

$$\boldsymbol{\phi}^{B}_{Q,i} = \boldsymbol{\Theta}\boldsymbol{\phi}^{N}_{Q,i} + (\hat{\mathbf{I}} - \boldsymbol{\Theta})\boldsymbol{\phi}^{LDA}_{Q,i}\,, \tag{3.30}$$

where $\hat{\mathbf{I}}$ is the identity matrix. The entries of the diagonal nonlinear blending matrix $\boldsymbol{\Theta}$ are given by

$$\boldsymbol{\Theta}_{k,k} = \frac{\left|\phi^E_{Q,k}\right|}{\sum_{l=1,l\in E}^{d+1} \left|\phi^N_{Q,l,k}\right| + \epsilon}, \qquad \epsilon = 10^{-10}, \tag{3.31}$$

where index k refers the the kth equation of the system and l loops over the vertices of the element [19].

Map A Scheme

The scalar map A scheme is also used for systems of equations. However, the system is first cast into a set of scalar equations by projecting the fluctuation, and distributions of the system N scheme, onto the left eigenvectors, $\boldsymbol{l}^E_\sigma$, of the one-dimensional, 3×3 subset of system

$$\mathbf{A}_Q m_x + \mathbf{B}_Q m_y . \tag{3.32}$$

In (3.32), $\hat{m}$ is a unit vector aligned with the linearized velocity in the element, $\bar{\vec{q}}$ [8]. Following the notation used by Ricchiuto [47] and with $\sigma = 1 \ldots 3$, the projected fluctuation is determined as

$$\psi^N_{i,\sigma} = \boldsymbol{l}^E_\sigma \phi^N_{i,\sigma} \qquad \text{and} \qquad \psi^E_\sigma = \boldsymbol{l}^E_\sigma \phi^E_\sigma . \tag{3.33}$$

For each scalar component, the fluctuation is then limited using the scalar map A scheme,

$$\psi^{\mathrm{map}A}_{i,\sigma} = f^{\mathrm{map}A}\left(\psi^N_{j,\sigma}, \psi^E_\sigma\right) . \tag{3.34}$$

Finally, the limited results are projected back into conservative variables by

$$\boldsymbol{\phi}^{\mathrm{map}A}_i = \sum_{\sigma=1}^{3} \psi^{\mathrm{map}A}_{i,\sigma} \boldsymbol{r}^E_\sigma . \tag{3.35}$$

The left (rows) and right (columns) eigenvectors are given by

$$\boldsymbol{l}^E_\sigma = \begin{bmatrix} 0 & -m_y & m_x \\ \frac{1}{2} & \frac{m_x}{2} & \frac{m_y}{2} \\ -\frac{1}{2} & \frac{m_x}{2} & \frac{m_y}{2} \end{bmatrix}, \qquad \text{and} \qquad \boldsymbol{r}^E_\sigma = \begin{bmatrix} 0 & 1 & -1 \\ -m_y & m_x & m_x \\ m_x & m_y & m_y \end{bmatrix}, \tag{3.36}$$

respectively.

This method for producing a $\mathcal{P}$ and $\mathcal{LP}$ mapping scheme for systems of equations does suffer from a known flaw. The symptoms of the flaw are poor convergence and "the occurrence of wiggles in the smooth part of the solution" [4]. Repeating the comments of Ricchiuto [47], matrix distribution techniques may prevent the occurrence of type I triangles and the dissipation associated with the upwinding of a one-target distribution. In the absence of type I triangles, limiting produces destabilizing effects such that convergence to machine zero is never achieved. More information and proposed corrections involving the addition of dissipation in smooth regions are discussed by Abgrall [3] and Abgrall *et al.* [6].

Accuracy of Nonlinear Matrix Schemes

The nonlinear matrix schemes are solved on the same subset of Ringleb's flow that was used to test the linear matrix schemes. Results are illustrated in Fig. 3.2 and listed in Table 3.2.

Poor convergence was observed for residuals of the blended scheme but this was easily rectified by freezing the blending coefficient after the convergence stalled. The performance of the blended scheme for systems is similar to the scalar case with only the L_∞-error being significantly worse. The L_1-error is below second-order accurate and this degradation will be explored in more detail in Chapters 5 and 6. The order of the L_∞-error is unity in this example indicating that the solution is not being recognized as smooth in at least one element. The results from the map A scheme, on the other hand, are unsatisfactory. The performance is actually worse than the N scheme. It is suspected that this test case exacerbates the flaw known to exist in the scheme. The symptom of poor convergence is indeed observed; both the L_2 and L_∞ norms of the residual only converge by one order of magnitude. A type of limiter freezing was not employed because there is no parameter to freeze aside from the distribution coefficients themselves.

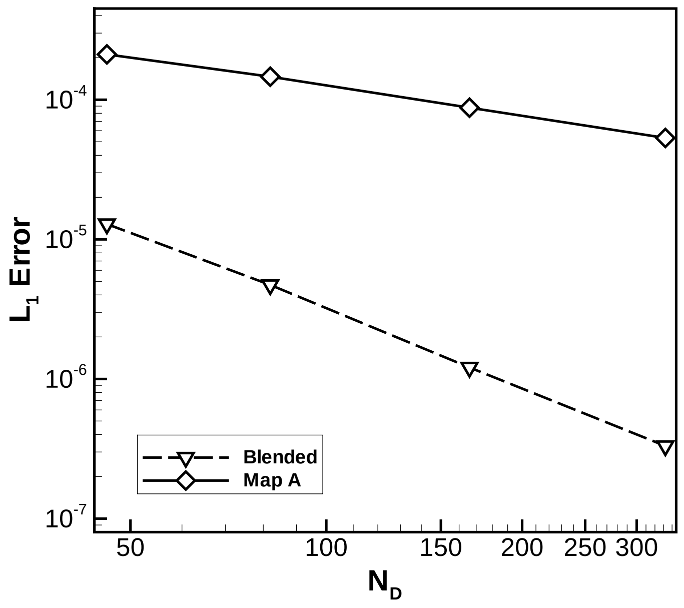


Figure 3.2 Accuracy of nonlinear matrix schemes applied to a Cartesian grid on a subset of Ringleb's flow.

Table 3.2 Spatial convergence of nonlinear matrix schemes applied to a Cartesian grid on a subset of Ringleb's flow.

Scheme	L_1-error	L_2-error	L_∞-error
Blended	-1.87	-1.65	-1.00
Map A	-0.72	-0.66	-0.83

Because of the extremely poor accuracy of the map A scheme, we rely on the blended scheme for accuracy estimates of $\mathcal{P}$ and $\mathcal{LP}$ solutions to systems. However, it is important to note that recent work on the map A scheme may entirely correct the flaw [3, 6].

Chapter 4

Implementation of the Numerical Algorithm

Details of the implementation including the meshes, imposition of boundary conditions, and the time-marching algorithm are described in this chapter. The approach for obtaining orders of accuracy greater than two is then presented and discussed.

4.1 Structured Mesh

Standard $\mathcal{RD}$ methods are solved on a grid of simplexes (triangles in two dimensions). Both structured and unstructured meshes are used for the solutions presented in this thesis. In Chapter 5, our $\mathcal{FV}$ solutions are obtained on structured quadrilateral grids and, with some special modifications, the same grids are used to obtain the $\mathcal{RD}$ solutions. In later chapters, unstructured grids are used for the subsequent analysis and for the extension to fourth-order accuracy. The approach adopted here for applying the $\mathcal{RD}$ schemes to quadrilateral grids is to insert a diagonal into each quadrilateral and thereby triangulate the mesh. It is therefore possible to take advantage of the freedom to optimally align the diagonal with the characteristic vector. In Chapter 5, the effects of choosing an optimal direction for the diagonal are examined for scalar equations. Another method for applying an $\mathcal{RD}$ scheme to a quadrilateral mesh is discussed by Abgrall and Marpeau [7].

For scalar equations, the optimal diagonal is aligned with the advection vector. For solutions of the Euler equations by matrix distribution or subsonic decomposition, the diagonal is aligned with the streamline vector. The same procedure can be applied when decomposing the Euler equations in supersonic flows, but in some cases this will cause the diagonal to be inserted in a direction that is opposite to the direction of the dominant wave. An example of this is illustrated in Fig. 4.1 for the oblique shock problem of Fig. 1.4a. In this example, the incident supersonic flow is oriented in a direction from the top left to the bottom right. An oblique shock produced by a solid wall aligned with the x-axis turns the flow to the horizontal or x-direction. The streamline in the incident flow (and through a finite shock) is therefore oriented in a direction that is opposite to the direction of the shock wave. The shock is the only significant wave in the flow and ideally the diagonals should be aligned with this wave. Figure 4.1a and 4.1b show the grid and a discrete representation of the exact solution, respectively. A matrix distribution using a blended scheme with a streamline tessellation is shown in Fig. 4.1c. A decomposed solution (the flow is entirely supersonic so the Euler equations decouple into four scalar equations) using an LN scheme with a streamline tessellation is shown in Fig. 4.1d. Both of these are more dissipative than if the tessellation had been fixed in the orientation of the shock prior to the solution. In Fig. 4.1e, each scalar wave resulting from the decoupled Euler equations is solved on a tessellation aligned with its own characteristic velocity. The shock is much more compact but numerous spurious waves are produced behind the shock. The reasons for these waves are currently not fully understood. In Fig. 4.1f, both streamline waves are solved on a tessellation aligned with the streamline, and both acoustic waves are solved on a tessellation aligned with the dominant acoustic wave. This technique seems to eliminate the spurious waves and is used for all results obtained on quadrilateral grids.

Note that when a separate tessellation is used for the streamline and acoustic waves,

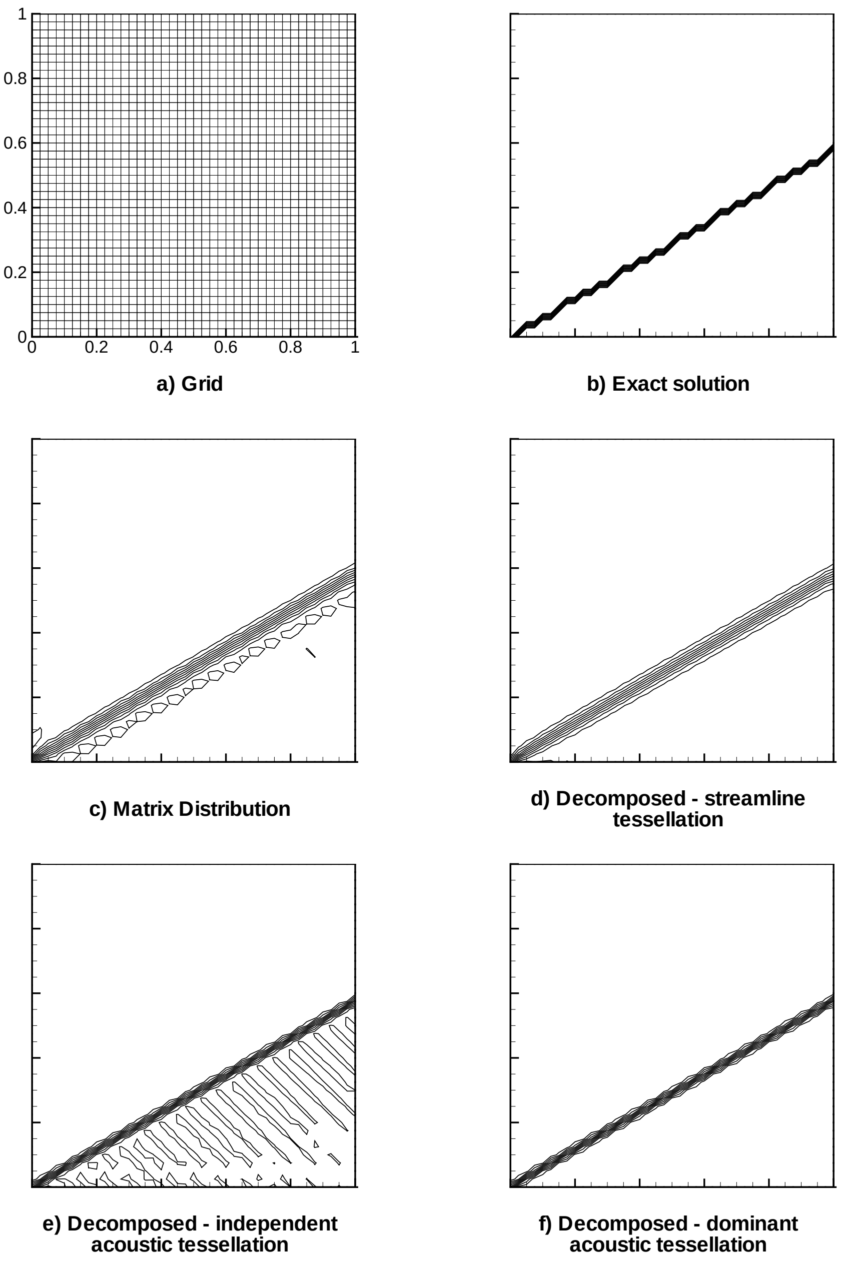


Figure 4.1 Distribution of pressure for solution of an oblique shock. The effect of various tessellations are examined.

the linear state is averaged over the entire quadrilateral and conservation is maintained on the quadrilateral. Since all transformation matrices are computed at the same state, and since the CRD linearization technique allows conservation to be independent of the linear state, the scheme is still conservative. The dominant acoustic wave is determined by comparing the difference in the acoustic characteristic variables between opposite

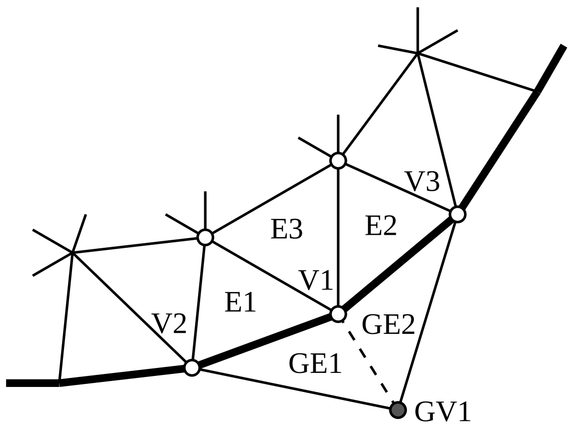


Figure 4.2 Weak boundary condition for vertex V1 consisting of ghost elements, GE1 and GE2, and ghost vertex GV1.

vertices in the quadrilateral. The dominant wave is assumed to run counter to the largest difference. Note also that the tessellation is frozen at a prescribed level of convergence so as not to interfere with the convergence of the solution.

4.2 Boundary Conditions

Boundary conditions (BC) are implemented using the weak formulation originally proposed by Paillère [43]. In contrast with a strong formulation, where the required boundary state is imposed directly at the vertices, a weak formulation indirectly enforces the boundary condition by using supplementary ghost elements. In Fig. 4.2, the boundary conditions for vertex V1 are prescribed indirectly via two ghost elements, GE1 and GE2. Vertex V1 is completely surrounded by physical elements E1, E2, and E3 and ghost elements GE1 and GE2. The ghost elements are degenerate with the dashed line having zero length. The ghost vertex, GV1, therefore lies directly on top of vertex V1. Because of the degeneracy, no fluctuation is sent from the ghost elements to V2 or V3. The states in the ghost vertices are set to produce the desired results, e.g., farfield conditions for a farfield BC, reflected velocity for a symmetry or inviscid wall BC, or the desired conditions for a Dirichlet BC. For the purpose of calculating the time-step in the ghost element, the area is taken to be half the area of the physical dual mesh associated with vertex V1. Except for special handling of the zero-length edge, which only involves explicitly setting the inflow parameter to zero for that edge, the ghost elements are treated the same as any of the interior elements.

Paillère [43] recommends to use all three nodes in the ghost element to set the linearized state. This is reasonable if the state in the ghost vertex is carefully maintained. However, it is convenient to minimize this maintenance by letting the scheme itself determine which characteristic information is needed. Consider, for example, a farfield BC. It is desirable to set the farfield state in the ghost vertex and not change it. If the boundary experiences supersonic outflow, the ghost vertex should not have any influence on the linearized state. Otherwise it could, depending on its value, change the linearized flow to subsonic. For this scenario, it is preferable to only use the interior vertices to compute the linearized state (vertices V1 and V2 for element GE1 in Fig. 4.2). If the boundary experiences anything other than supersonic outflow, then information from the ghost vertex is required. In this situation, usage of only the interior vertices for calculation of the linearized state is still valid since the interior vertices should eventually adopt the appropriate value due to the incoming waves. For inviscid wall BC, on the other hand, the ghost vertex is instrumental in altering the flow. This is especially true for supersonic flows normal to the wall, such as may be encountered during the impulsive starting of a solution. For wall or symmetry BC, all three vertices are used to compute the linearized state whereas for all other BC, only the interior vertices are used.

In contrast to ghost cells commonly used in $\mathcal{FV}$ schemes, it is straightforward to apply the ghost elements in corners. Vertex GV1 can be split into two separate vertices and given different states if either the normals are different (at a corner) or the left and right boundary conditions differ.

For all the problems considered in this thesis, this manner of imposing boundary conditions has proven robust. However, we have experienced difficulty with other cases, such as the supersonic forward-facing step described by Woodward and Colella [66] and originally proposed by Emery [23]. For that problem, the solution can be impulsively started with supersonic flow normal to the wall, but eventually negative density and pressure will result behind the corner of the step. Robust handling of supersonic expanding flows at walls is still elusive to our implementation and certainly requires more investigation. Other options that may improve results at walls include setting the normal velocity to zero in the ghost vertex (instead of reflecting it) and finely tailoring the linearized state in the ghost elements. Another curiosity we have noticed, especially at walls, is that instabilities may develop if the linearized state is computed from only one vertex (e.g., set the linearized state to that of vertex V1 for elements GE1 and GE2). It seems necessary to include information tangential to the boundary when computing the linearized state. Hence, vertex V2 must be used to help compute the linearized state for element GE1 and vertex V3 for element GE2.

4.3 Time Marching Algorithm

A simple explicit-Euler time-marching algorithm is used to advance the solution in time. The time step in each element is computed as

$$h^E = \frac{1}{3}\frac{\Omega^E}{k_{max}}\,, \tag{4.1}$$

a restriction that ensures positivity for a scalar N scheme. For scalar equations, k_{max} is the maximum inflow parameter in the element. For matrix-distribution techniques, k_{max} is the maximum eigenvalue over all the $\mathbf{K}_{i,i\in E}$ matrices. For the scalar equations, a Courant-Friedrichs-Lewy (CFL) number of 1.8 is used. For supersonic Euler solutions, a CFL number of 1.5 is used. In all other flows, a CFL number of unity is used. CFL numbers above unity were determined by trials and are acceptable in some cases because the positivity condition is more restrictive than necessary (see Eq. (2.13) and the text preceding it).

4.4 Non-dimensional Variables

For solutions of the Euler equations, the state variables are all made non-dimensional by

$$\tilde{\rho} = \frac{\rho}{\rho_\infty}\,, \quad \tilde{u} = \frac{u}{a_\infty}\,, \quad \tilde{v} = \frac{v}{a_\infty}\,, \quad \text{and} \quad \tilde{e}_T = \frac{e_T}{a_\infty^2}\,, \tag{4.2}$$

where a denotes the speed of sound. Equation (1.1) is changed to

$$\frac{1}{a_\infty}\frac{\partial \mathbf{U}}{\partial t} + \vec{\nabla} \cdot \vec{\mathbf{F}} = 0\,, \tag{4.3}$$

and written with the non-dimensional variables given above instead of the dimensional variables. The physical time step from solving (4.3) in an element is given by

$$h^E = \frac{\tilde{h}^E}{a_\infty} \tag{4.4}$$

where $\tilde{h}^E$ results from (4.1) applied to non-dimensional variables. For the most part, using non-dimensional variables has no effect on the solution. However, errors from finite floating-point precision were encountered during use of the GPU, and having all variables at the same magnitude allows for better identification of the round-off errors.

4.5 Construction of High-Order $\mathcal{RD}$ Schemes

Schemes with an order of accuracy greater than two are constructed by following a framework similar to that of finite-element theory. Abgrall and Roe [9] used such an approach to increase the number of degrees of freedom in an element by inserting nodes in the interior. This leads to the construction of P^2 and P^3 elements for third and fourth order solutions, respectively. In this work, we adopt a slightly different perspective where *reconstruction elements* are defined as an ordered collection of *primary elements* with the desired degrees of freedom. Figure 4.3 shows a reconstruction element with 9 primary elements and 10 degrees of freedom (vertices) for a fourth-order solution. Although the primary elements should have an arrangement similar to that shown

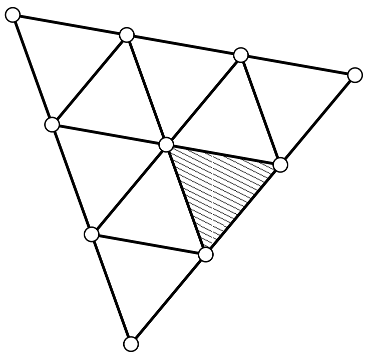


Figure 4.3 P^3 reconstruction element consisting of an ordered collection of primary elements (shaded).

in Fig. 4.3, the reconstruction elements can have a completely unstructured connectivity. The solution is obtained on the primary elements and the reconstruction elements only serve to compute the high-order reconstruction. The fluctuation over the entire reconstruction element is never of interest. Ultimately, the same approach is taken by Abgrall and Roe [9]; the different perspectives only affect the moment in the CFD process when P^3 elements are introduced. In [9], extra degrees of freedom are introduced into a pre-existing triangular mesh; in our case [29], it is at the time of grid generation. All degrees of freedom are introduced as discrete unknowns before starting the solution and retained afterwards. This allows for additional tailoring of the mesh; the details are discussed later in this section. The results we present all have fourth-order spatial accuracy but the technique can be extended to any desired order. Note that alternative approaches to constructing high-order $\mathcal{RD}$ schemes do exist in the literature [13, 16, 31, 32].

Lagrange basis functions are used to define a cubic interpolating polynomial for the entire P^3 element. Within this *Lagrange* element, both the coordinates and the solution are interpolated by the Lagrange basis functions. The solution is interpolated in the variables chosen for the linear state, (2.22) for advection and (3.6) for the Euler equations. These polynomials are used to integrate the fluctuation and the linearized state in each of the primary elements that are members of the reconstruction element. The Lagrange basis functions provide C^0 continuity along the edges of the reconstruction-elements, thereby ensuring consistent evaluations of the fluctuation through an edge. The coordinates may be interpolated by Lagrange basis functions at the same order as the solution, leading to iso-parametric Lagrange-elements [67]. This allows for curved edges in the reconstruction elements so that they may be fitted to boundaries of the domain. Alter-

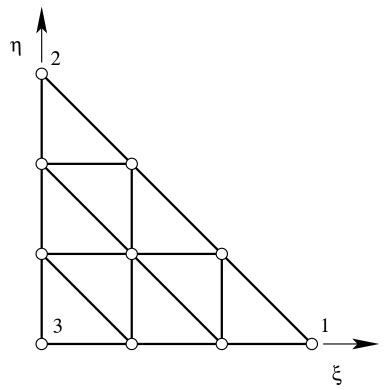


Figure 4.4 Canonical coordinate system for triangular Lagrange elements.

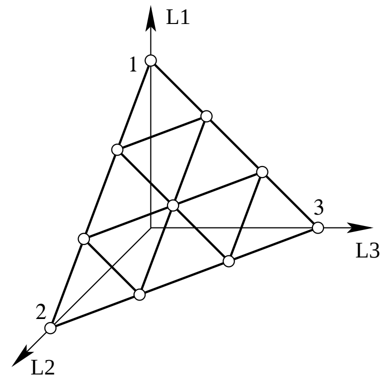


Figure 4.5 Natural coordinate system for triangular Lagrange elements.

natively, the coordinates may be linearly interpolated in each primary element leading to sub-parametric Lagrange elements.

The Lagrange basis functions can be defined in canonical, Fig. 4.4, or natural, Fig. 4.5, normalized coordinate systems. The canonical coordinates, ξ and η, are orthogonal while the natural coordinates, L_1, L_2, and L_3, are related by the expression $L_1 + L_2 + L_3 = 1$. The symmetry of the natural coordinate system allows for application of triangular Gauss quadrature rules [67]. All the integrations required in a primary element are performed using numerical Gauss quadrature. For integration in a specific primary element, a mapping is performed such that the natural coordinate system, typically normalized for the reconstruction element, is instead normalized over the primary element. We define the natural coordinates, normalized over a primary element, as S_1, S_2, and S_3.

Within each primary element, two quantities must be integrated: the linearized state and the fluctuation. Components of the linearized state, here denoted by U, are given by

$$\bar{U} = \frac{1}{\Omega_E} \int_E U \, \mathrm{d}\Omega_E \,, \tag{4.5}$$

where Ω_E is the area of the primary element. A sub-parametric Lagrange element is assumed for computation of the linearized state and after transformation into a normalized primary element, T, (4.5) becomes

$$\bar{U} = \int_T U\left(L_1, L_2, L_3\right) \,\mathrm{d}\Omega_T \,. \tag{4.6}$$

Equation (4.6) is integrated using a Gauss quadrature rule where $\vec{S}_j$ defines the location of Gauss point j in natural coordinates in a primary element and $\vec{L}_j = \vec{L}(\vec{S}_j)$. The linearized state is assumed to be no more than a linear function of the solution, and therefore, four Gauss points are sufficient to ensure exact integration of each component.

The fluctuation is integrated over each face, F, (edge) by evaluating the line integral

$$\boldsymbol{\phi}^F = -\int_F \vec{\mathbf{F}}\left(x, y\right) \cdot \hat{n}\left(x, y\right) \mathrm{d}\mathcal{S} \,. \tag{4.7}$$

In terms of the normalized parameter, s, the fluctuation is given by

$$\boldsymbol{\phi}^F = -\int_{-1}^{1} \vec{\mathbf{F}}\left(L_1, L_2, L_3\right) \cdot \left(-\frac{\mathrm{d}y}{\mathrm{d}s}\hat{\imath} + \frac{\mathrm{d}x}{\mathrm{d}s}\hat{\jmath}\right) \mathrm{d}s \,, \tag{4.8}$$

for a counter-clockwise integration around the element and with $\vec{S}_j = \vec{S}(s_j)$ and $\vec{L}_j = \vec{L}(S_j)$ at Gauss point j. The coordinate derivative $\mathrm{d}x/\mathrm{d}s$ can be obtained from

$$\frac{\mathrm{d}x}{\mathrm{d}s} = \frac{\partial x}{\partial L_1}\frac{\mathrm{d}L_1}{\mathrm{d}s} + \frac{\partial x}{\partial L_2}\frac{\mathrm{d}L_2}{\mathrm{d}s} + \frac{\partial x}{\partial L_3}\frac{\mathrm{d}L_3}{\mathrm{d}s} \tag{4.9}$$

and similarly for $\mathrm{d}y/\mathrm{d}s$. At each Gauss point, the solution in linearized form is interpolated and then used to compute $\vec{\mathbf{F}}$. For scalar advection, the flux, $\vec{f}$ ($\vec{\mathbf{F}}$ in (4.7) and (4.8)), is assumed to be no more than a quadratic function of the solution, u and, because all problems considered herein have straight edges, the coordinate derivatives are constants (i.e., giving sub-parametric Lagrange elements). Therefore, four Gauss points are sufficient to ensure exact integration of the fluctuation along each edge,

$$\underbrace{\left(u^3\right)^2}_{f_i}\left(\frac{\mathrm{d}x_i}{\mathrm{d}s}\right)^0 \;=\; \text{degree } 6 \;<\; 2(\mathbf{4}) - 1 \,.$$

For the Euler equations, the flux, $\vec{\mathbf{F}}$, may be up to a quartic function of the primitive (linearized) variables and the possibility of curved edges is considered. Eight Gauss points are sufficient to ensure exact integration of the fluctuation,

$$\underbrace{\left(U^3\right)^4}_{\mathbf{F}_i}\left(\frac{\mathrm{d}x_i}{\mathrm{d}s}\right)^2 \;=\; \text{degree } 14 \;<\; 2(\mathbf{8})-1\,.$$

The high-order method only alters the integration of ϕ^E and the linearized state within each primary element. The distribution of the fluctuation and the evolution of the solution are the same in the primary elements as for standard second-order schemes. The CRD technique is used to ensure conservation. This is especially important for high-order schemes because (3.4) will certainly not hold true.

A unique characteristic of the high-order reconstruction is that, compared with a typical high-order $\mathcal{FV}$ scheme [33], the interpolation is not necessarily centered around a primary element. However, the asymmetry of the stencil allows for the solution in the 10 primary elements to be interpolated by one reconstruction, a process which should be very efficient. Additionally, the impact of the high-order scheme on parallelization and boundary conditions is minimal.

Currently, our approach to achieving high-order is very comprehensive in terms of accuracy but does not weigh the accuracy gains against the computational expense. For many of the algorithmic details, we have not thoroughly explored improving the efficiency for a given level of solution error. Gains might be realized, for example, by using a lower-order interpolation for integrating the linear state or by using sub-parametric Lagrange elements everywhere. Although such modifications may increase the error, their use may be more efficient on a refined mesh that compensates for the increased error. Alternatively, these modifications may not even adversely affect the error.

4.5.1 High-Order Mesh

All high-order solutions for this work are obtained on unstructured meshes. As mentioned previously, the P^3 elements are defined during mesh generation. While more demanding of mesh-generation software, this does allow for fitting the reconstruction elements

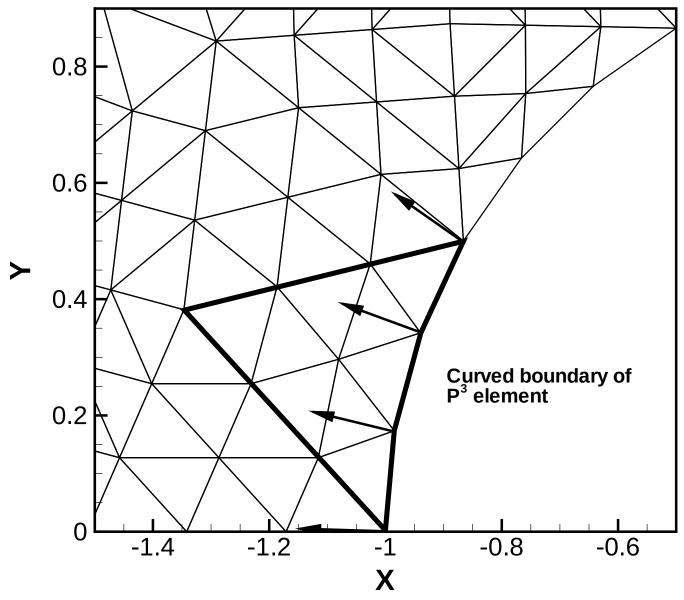


Figure 4.6 High-order mesh elements respect the curvature of the boundary.

around the curvature of the domain boundaries as shown in Fig. 4.6. Note that the interior reconstruction elements all still have straight boundaries and could be interpolated using sub-parametric Lagrange elements. Although not studied quantitatively, the use of curved boundaries seems to have a large effect on minimizing the error. We have made modifications to the open-source mesh generator Gmsh [24] to support the output of boundary-fitted P^3 meshes in CGNS [14] format. An alternative would be to build the P^3 elements within the solver from a mesh of triangles by inserting the extra degrees of freedom and mapping them to the domain boundary. However, this requires more than a discrete understanding of the geometry within the solver code. It is our opinion that such complexity is best left within the mesh generator.

4.5.2 High-Order Boundary Conditions

For weak ghost cells, the solution on the edges is interpolated using one-dimensional Lagrange elements that follow a certain path, depending on which edge is being integrated.

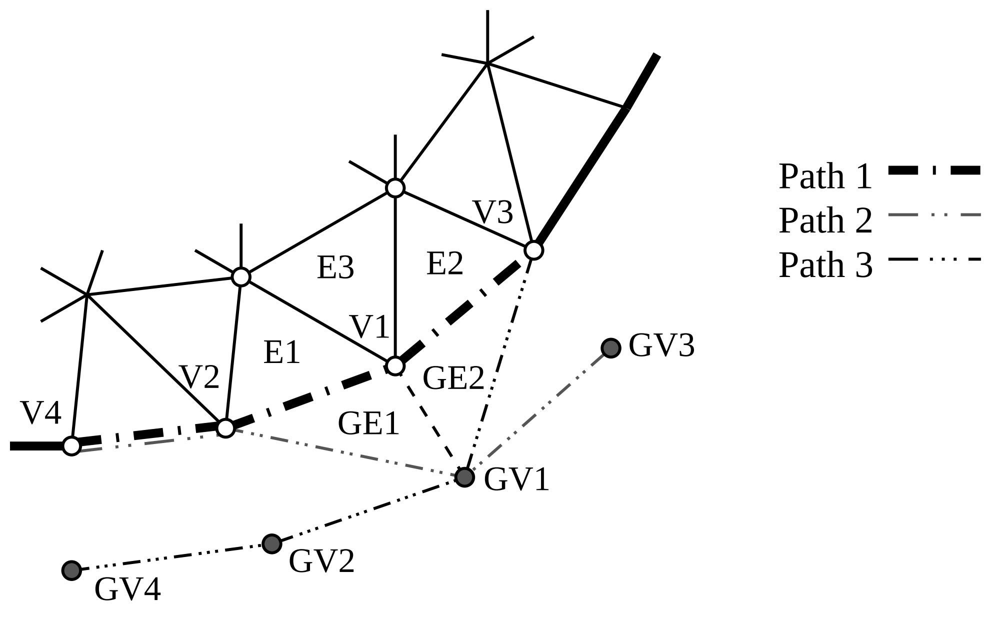


Figure 4.7 Paths for interpolating the solution in weak ghost elements of a fourth-order scheme.

Several possible paths are shown in Fig. 4.7. Note that interpolating along the interior path, path 1 in Fig. 4.7, is equivalent to interpolating on the edge of the two-dimensional Lagrange element that borders that path.

4.5.3 Accuracy of High-Order Scheme

The experiment involving a square subset of Ringleb's flow from section 3.3.1 is repeated here at fourth-order accuracy for various schemes. An example of the unstructured P^3 mesh is shown in Fig. 4.8 along with density contours. As before, vertices around the exterior of the domain are fixed to the exact solution to allow testing of only the interior scheme. Linear schemes using both hyperbolic/elliptic splitting and matrix techniques are illustrated in Fig. 4.9. The corresponding error norms are listed in Table 4.1. The notation LDA-LW for the decomposed schemes describes the distribution method for the decoupled scalar equations followed by the method for the acoustic subset (always LW). All the $\mathcal{LP}$ schemes achieve the expected fourth-order spatial convergence and have very

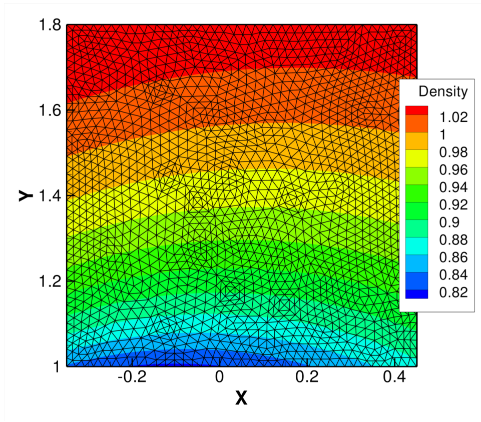


Figure 4.8 Unstructured mesh for high-order solutions on a subset of Ringleb's flow along with density contours.

similar error levels. Because it lacks the $\mathcal{LP}$ property, the matrix N scheme is still first-order accurate whereas the partial $\mathcal{LP}$ distribution of the decomposed N-LW scheme ($\mathcal{LP}$ for LW distribution of the acoustic subset) averages to near second-order accuracy.

Predictions from nonlinear schemes using both hyperbolic/elliptic splitting and matrix techniques are illustrated in Fig. 4.10 and the error norms are listed in Table 4.2. The decomposed LN-LW scheme has similar accuracy to the LDA-LW scheme with only the L_∞-error indicating a lower order of convergence. Although the matrix blended scheme performs very well on the two coarsest mesh samples, the results are more erratic as the mesh is further refined. This behaviour is only apparent at higher orders of accuracy. The reason is probably

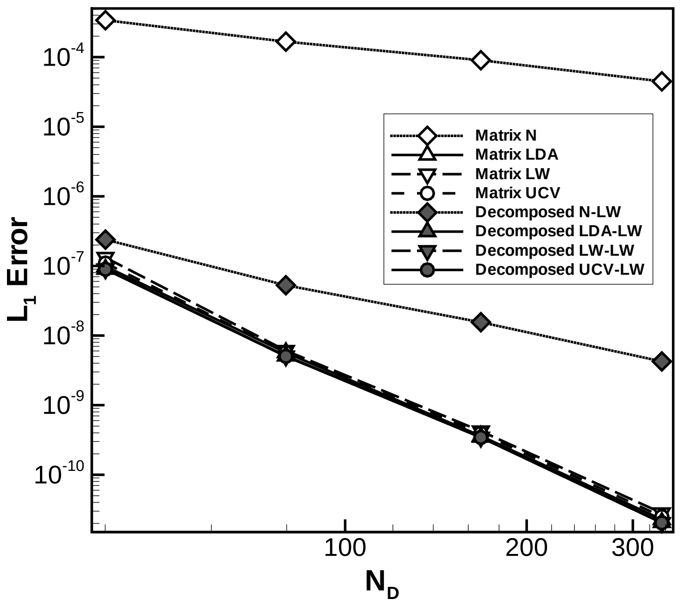


Figure 4.9 L_1-density error of high-order linear schemes applied to a Cartesian grid on a subset of Ringleb's flow.

Table 4.1 Spatial convergence of high-order linear schemes applied to a Cartesian grid on a subset of Ringleb's flow.

Scheme	L_1-error	L_2-error	L_∞-error
Matrix N	-1.01	-1.02	-0.83
Matrix LDA	-4.00	-3.97	-4.37
Matrix LW	-3.92	-3.94	-4.07
Matrix UCV	-3.99	-3.98	-4.19
Decomposed N-LW	-1.87	-1.95	-1.95
Decomposed LDA-LW	-4.08	-4.04	-3.88
Decomposed LW-LW	-4.09	-4.05	-3.80
Decomposed UCV-LW	-4.09	-4.05	-3.84

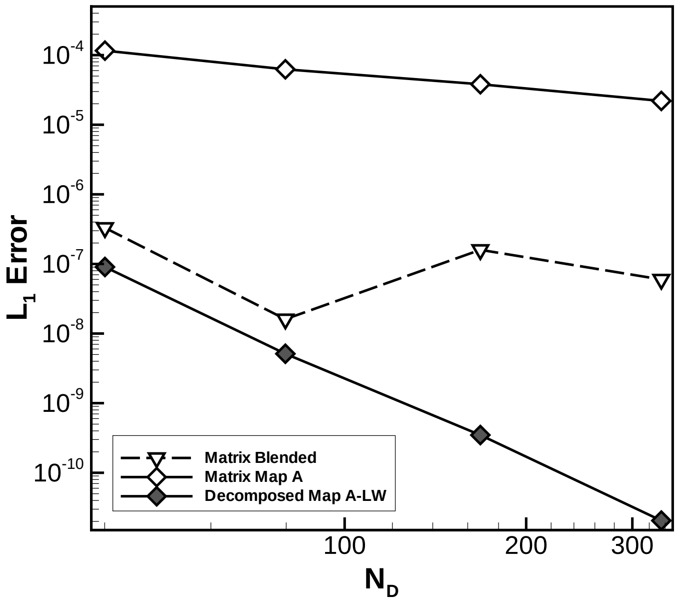


Figure 4.10 Accuracy of high-order nonlinear schemes applied to a Cartesian grid on a subset of Ringleb's flow.

Table 4.2 Spatial convergence of high-order nonlinear schemes applied to a Cartesian grid on a subset of Ringleb's flow.

Scheme	L_1-error	L_2-error	L_∞-error
Matrix map A	-0.79	-0.80	-0.16
Matrix blended[a]	-4.39	-4.46	-4.61
Decomposed LN-LW	-4.09	-4.07	-3.66

[a] Results from line segment between two coarsest meshes.

different from what normally affects the map A scheme as the blended scheme is fully converged for all these results. As for the map A scheme, switching to fourth order provides no significant change from what was seen at second order accuracy. For this case, the performance is not much better than the N scheme.

Chapter 5

Comparison of the Residual Distribution and Finite Volume Methods

A quantitative comparison of the accuracy of the $\mathcal{RD}$ method versus a Godunov-type $\mathcal{FV}$ method was performed on the set of canonical problems described in Chapter 1 (Figs. 1.3 and 1.4) [28]. These comparisons make exclusive use of structured quadrilateral meshes, but it is shown in the results that $\mathcal{RD}$ methods can also benefit from the structure by optimally tessellating the mesh. Graphs of spatial L_1-error norms are supplemented by contour plots of the computed distributions where informative. All comparisons are made using second-order versions of the numerical schemes. No direct comparisons are made concerning the accuracy per computational cost. A quantitative analysis of the costs was not possible at the time due to issues of optimization for the $\mathcal{RD}$ and $\mathcal{FV}$ code implementations. However, since both second-order schemes were observed to have approximately similar runtimes, differences in the solution accuracy obtained for the same number of unknowns does provide some indication of the accuracy per computational cost.

For smooth solutions (i.e., continuously differentiable solutions), it is expected that both the $\mathcal{RD}$ and $\mathcal{FV}$ schemes will exhibit second-order spatial accuracy, i.e., $\beta \approx -2$. For solutions with discontinuities, it is expected that both methods will reduce to first-order spatial accuracy to preserve monotonicity, i.e., $\beta \approx -1$. Where informative, the slope of the line-segment connecting the two finest grids is denoted by β in the error graphs to show the spatial order of accuracy. Of primary interest in (1.3) is the coefficient α which describes the absolute magnitude of the error. Although of the same formal order of accuracy, it is anticipated that the multidimensional $\mathcal{RD}$ schemes will have an error with an absolute magnitude that is lower than that of the $\mathcal{FV}$ scheme.

5.1 Godunov-Type $\mathcal{FV}$ Schemes

As mentioned above, a Godunov-type finite-volume scheme serves as a reference for evaluating the performance of the $\mathcal{RD}$ method. Godunov-type finite volume schemes perform an integration of the solution flux at the boundaries of a cell to compute the cell residual when advancing the solution in time using a time-marching method [26]. The solution is often stored and updated at the cell centers. Before computing the flux, the solution in the cell is reconstructed, possibly using neighbour cells. A piece-wise constant reconstruction leads to a first-order scheme while a piece-wise linear reconstruction leads to a second-order scheme. To maintain monotonicity, second-order schemes limit the reconstruction, reducing it towards piece-wise constant when there are large changes in the local solution gradient. The solution fluxes at the cell interfaces are evaluated in terms of the possibly discontinuous reconstructed solution values by solving a Riemann problem, providing an appropriate upwinding of the hyperbolic flux. In this study, a Godunov-type finite-volume method developed for body-fitted multiblock meshes is used as the basis for all of the comparisons to the $\mathcal{RD}$ schemes. The method incorporates a least-squares piece-wise linear reconstruction, the slope limiter of Venkatakrishnan [61], and the exact Riemann solver flux function of Gottlieb and Groth [27]. Refer to the paper by Sachdev *et al.* [52] for a complete description of the finite-volume method used herein.

5.2 Performance Comparisons for Scalar Equations

Aside from directly comparing the $\mathcal{RD}$ and $\mathcal{FV}$ schemes, the effect of grid tessellation is also examined for scalar equations. In each case, a relevant $\mathcal{RD}$ distribution scheme was used to compute solutions on grids with an optimal tessellation (diagonal aligned with characteristic vector), a reverse tessellation (diagonal opposite the characteristic vector), and a random unstructured triangular grid.

5.2.1 Linear Advection Equation

The $\mathcal{RD}$ and $\mathcal{FV}$ methods are first compared for the linear advection equation applied to the time-invariant problem of circular advection (Fig. 1.3a). Solutions to this smooth scalar flow were computed on simple Cartesian grids with uniform spacing. An interesting aspect of this test case is that the advection velocity is generally not aligned with the Cartesian grid. This challenges the dimensional-splitting of the $\mathcal{FV}$ scheme. Because the solution is smooth, the LDA scheme was used to obtain the majority of the $\mathcal{RD}$ results. The LN scheme is also solved to compare its performance relative to the LDA scheme. The $\mathcal{FV}$ solution was obtained without using a limiter. However, had one been used, experiments indicate it would have had virtually no effect on the computed solution. Numerical results were obtained for grid densities ranging from $N_D = 40$ to $N_D = 640$.

Figure 5.1 shows a Cartesian mesh of size 80×80 and the various solutions obtained on that mesh. The grid is shown in Fig. 5.1a and a representation of the exact solution on the mesh is shown in Fig. 5.1b. Notable dissipation is observable in solutions generated by the $\mathcal{RD}$ LN (minmod) scheme, $\mathcal{FV}$ scheme, and $\mathcal{RD}$ LDA scheme with a reverse tessellation. Figure 5.2 depicts the variation of the spatial accuracy with the mesh size.

From the results given in these two figures, it is apparent that the effects of the tessellation are very significant. When using an $\mathcal{RD}$ LDA scheme, the optimal tessellation is over half an order of magnitude more accurate that the reverse tessellation. The

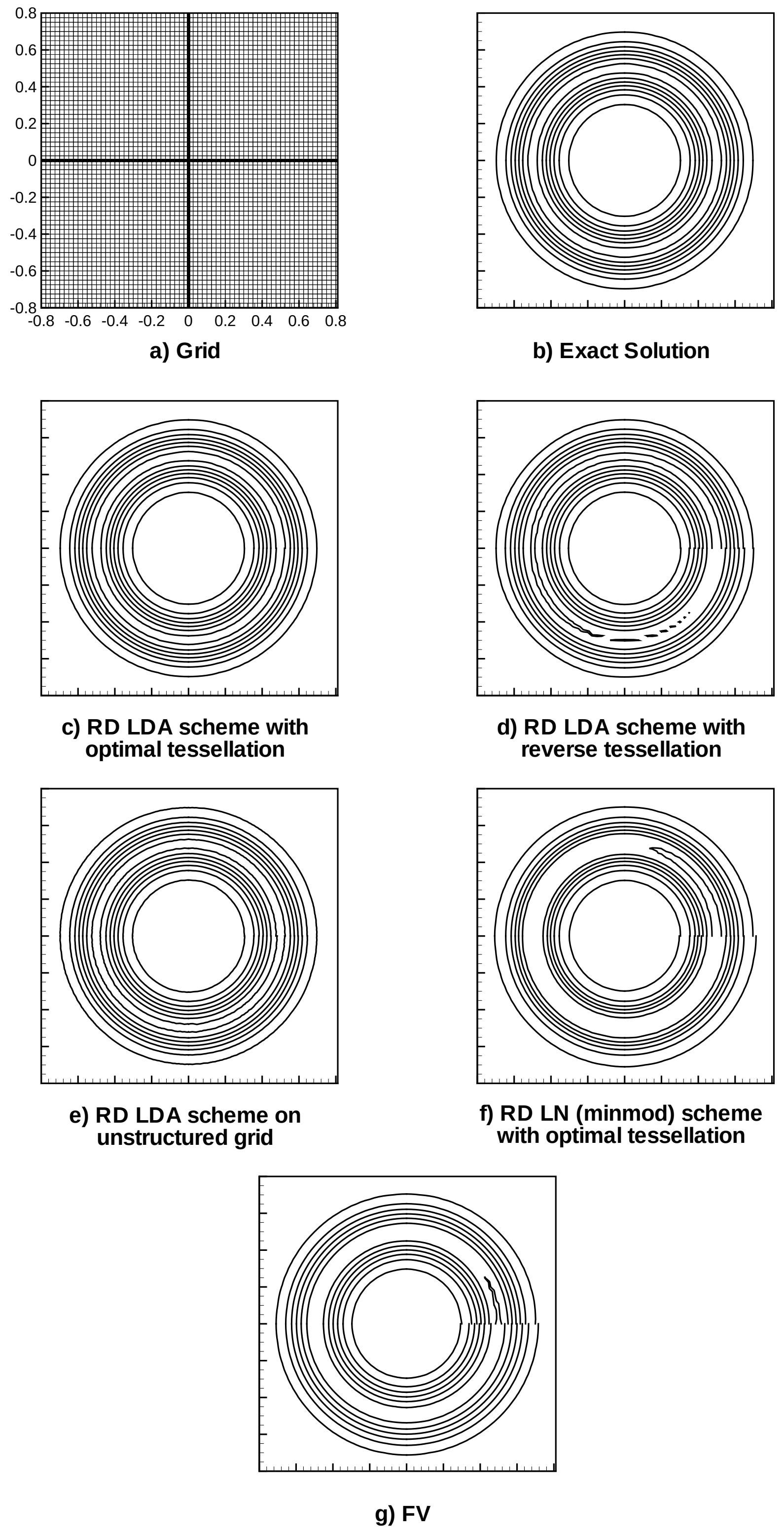


Figure 5.1 Solutions of circular advection obtained using a 80×80 uniform Cartesian mesh.

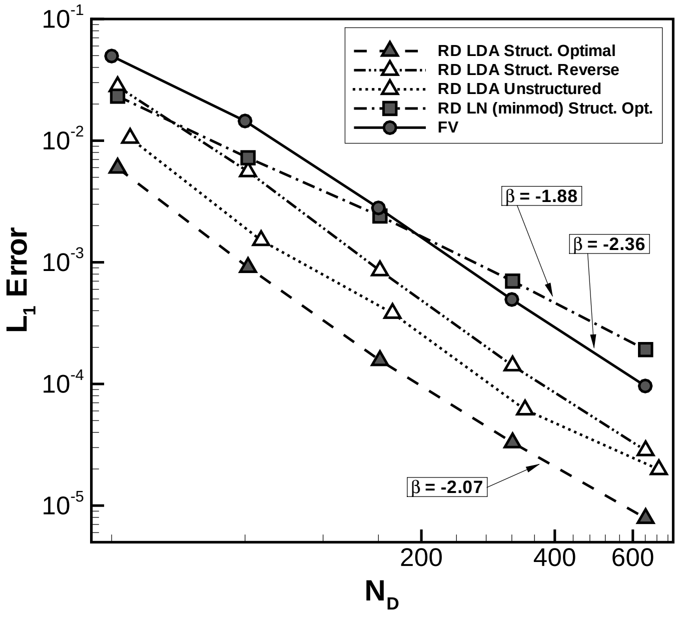


Figure 5.2 L_1-error as a function of mesh density for circular advection.

accuracy achieved on the unstructured grid with the random tessellation lies in between. It is also quite apparent from the predicted errors that, while both the LDA scheme and the $\mathcal{FV}$ scheme achieve second-order accuracy for this problem, the absolute error of the LDA scheme with the optimal tessellation is more than an order of magnitude less than that of the $\mathcal{FV}$ scheme.

The results for the $\mathcal{RD}$ LN scheme with a minmod limiter are somewhat less impressive. Although the LN (minmod) scheme offers a marginal improvement over the $\mathcal{FV}$ method on coarser meshes, this quickly disappears because of its lower order-of-accuracy ($\beta \approx -1.88$). Other nonlinear formulations, such as the blended scheme, also exhibit a similar degraded order of accuracy.

5.2.2 Nonlinear Burgers Equation

The solution to the nonlinear Burgers equation (Fig. 1.3c) is considered next and used to evaluate the shock-capturing properties of both the $\mathcal{RD}$ and $\mathcal{FV}$ schemes for scalar

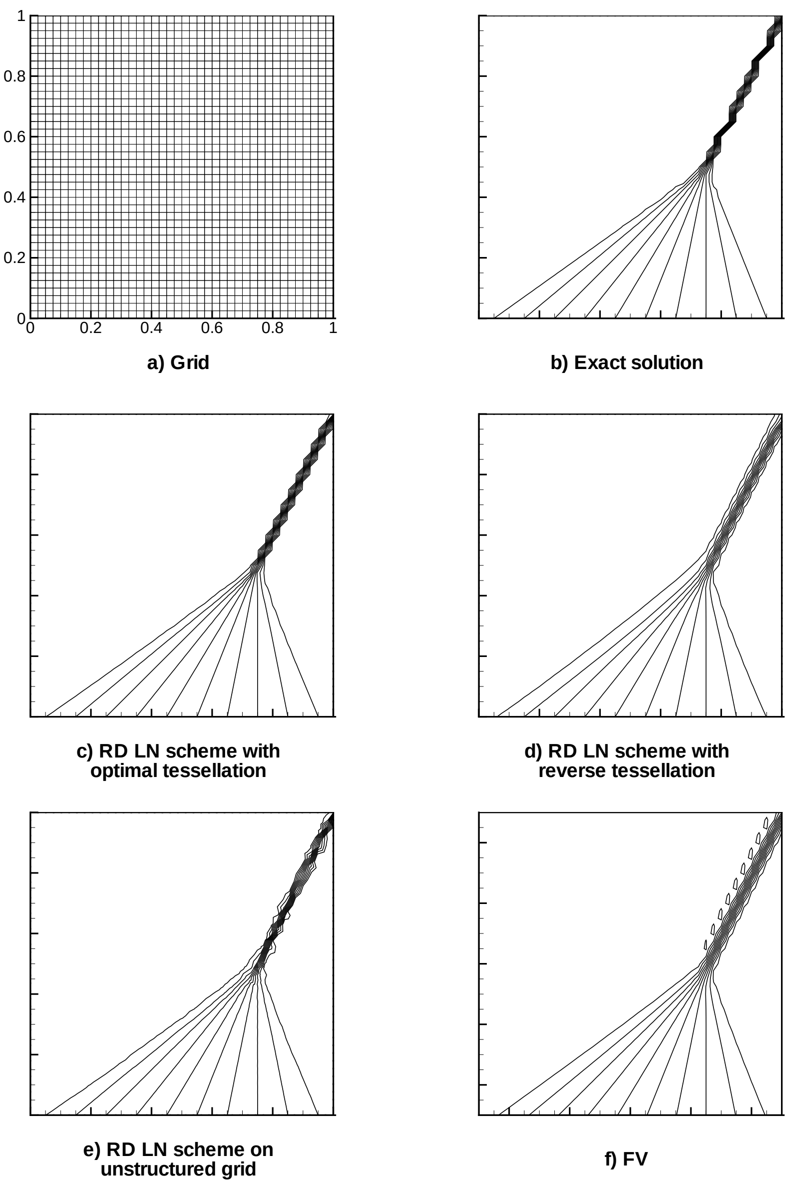


Figure 5.3 Solutions to Burgers equation obtained using a 40×40 uniform Cartesian mesh.

equations. A Cartesian mesh with uniform spacing is again used to obtain the numerical solutions. The compression waves and the shock run at angles to the Cartesian quadrilateral grid, again challenging the dimensional-splitting of the $\mathcal{FV}$ scheme. The $\mathcal{RD}$ results were obtained using an LN distribution scheme to preserve monotonicity. The $\mathcal{FV}$ results used the slope-limiter of Venkatakrishnan [61] for the same purpose. Numerical results were obtained for grid densities ranging from $N_D = 40$ to $N_D = 640$.

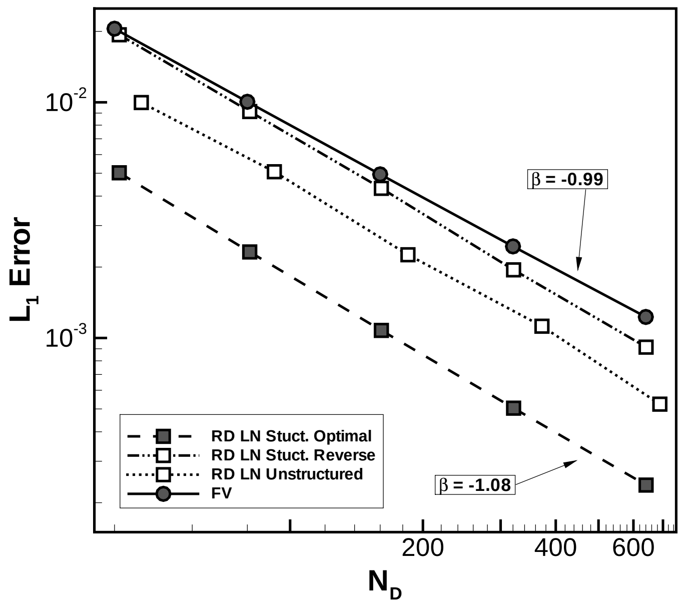


Figure 5.4 L_1-error as a function of mesh density for Burgers equation.

Figure 5.3 shows the coarsest Cartesian mesh used in the calculations and the various solutions obtained on that mesh. Figure 5.3b depicts a representation of the exact solution on the discrete mesh of Fig. 5.3a and numerical solutions obtained using the $\mathcal{RD}$ LN scheme with various tessellations are given in Figs. 5.3c-e. The solution obtained using the $\mathcal{FV}$ scheme is shown in Fig. 5.3f. It is evident from these results that, at least qualitatively, the $\mathcal{RD}$ LN scheme using an optimal tessellation provides the most compact shock.

Figure 5.4 depicts the variation of the spatial accuracy with the mesh size. All schemes have near first-order accuracy in the L_1-error norm. The spatial accuracy of the best $\mathcal{RD}$ scheme is greater than the $\mathcal{FV}$ scheme by more than half an order of magnitude. Interestingly, this is entirely dependent upon proper tessellation of the grid; the $\mathcal{RD}$ scheme solved on the reverse tessellation is about equivalent to the $\mathcal{FV}$ scheme. As should be expected, the $\mathcal{RD}$ solution on the unstructured mesh is somewhere between the optimal and reverse tessellations.

5.3 Performance Comparisons for the Euler Equations

Solutions to the Euler equations are now considered. The performance of the $\mathcal{RD}$ and $\mathcal{FV}$ schemes are evaluated for a discontinuous supersonic flow, a smooth subsonic flow, a flow with stagnation regions, and a smooth transonic flow. The quadrilateral grid used to obtain the $\mathcal{RD}$ solutions is always optimally tessellated. For the decomposed $\mathcal{RD}$ schemes applied to supersonic flow regimes, a split streamline and dominant acoustic tessellation is used. For the other $\mathcal{RD}$ schemes and flow regimes, the tessellation direction is aligned with the flow streamline.

5.3.1 Supersonic Flow Past a Diamond-Shaped Aerofoil

The first problem considered related to the Euler equations involves supersonic flow past the diamond-shaped aerofoil of Fig. 1.4b. Numerical solutions were obtained using a $\mathcal{RD}$ decomposed LN scheme (the flow fully decouples everywhere), a $\mathcal{RD}$ matrix blended scheme, and, for comparison purposes, the $\mathcal{FV}$ scheme. The computational domain was divided into four blocks with the body-fitted multiblock grids ranging in size from 160×40 to 640×160.

Figures 5.5 and 5.6 depict both qualitative and quantitative results for the error in the computed density of the $\mathcal{RD}$ and $\mathcal{FV}$ schemes. Figure 5.5 shows the coarsest grid used in the accuracy study and the various solutions obtained on that grid for this supersonic flow problem. It is quite apparent from this figure that the numerical shocks obtained with the $\mathcal{FV}$ code are somewhat thicker than those of the $\mathcal{RD}$ schemes due to its more dissipative nature. The qualitative results for the two $\mathcal{RD}$ schemes (decomposed and matrix) appear quite similar. The L_1-norm of the solution error of the various schemes as a function of the mesh size is provided in Fig. 5.6. As expected, all of the schemes exhibit close to first-order accuracy ($\beta = -1$). However, small differences in the absolute

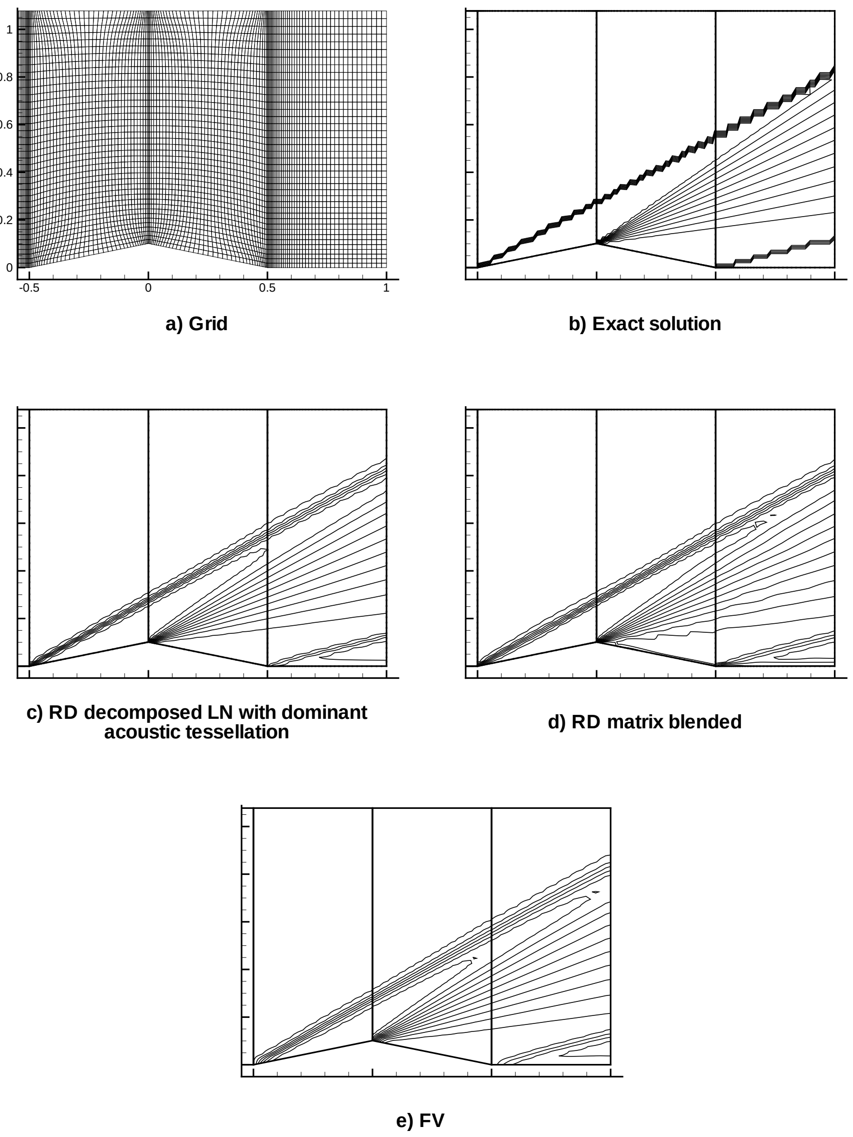


Figure 5.5 Density distributions for $M = 3$ supersonic flow past a diamond-shaped aerofoil obtained using 160×40 body-fitted multiblock mesh.

error are evident. The decomposed $\mathcal{RD}$ solution is the most accurate, followed by the matrix $\mathcal{RD}$ solution and then the $\mathcal{FV}$ solution.

5.3.2 Subsonic Flow Past a Smooth Bump

The performance of the $\mathcal{RD}$ and $\mathcal{FV}$ algorithms is now considered for subsonic flow past a smooth bump (Fig. 1.4c). The computational grid was divided into six blocks and

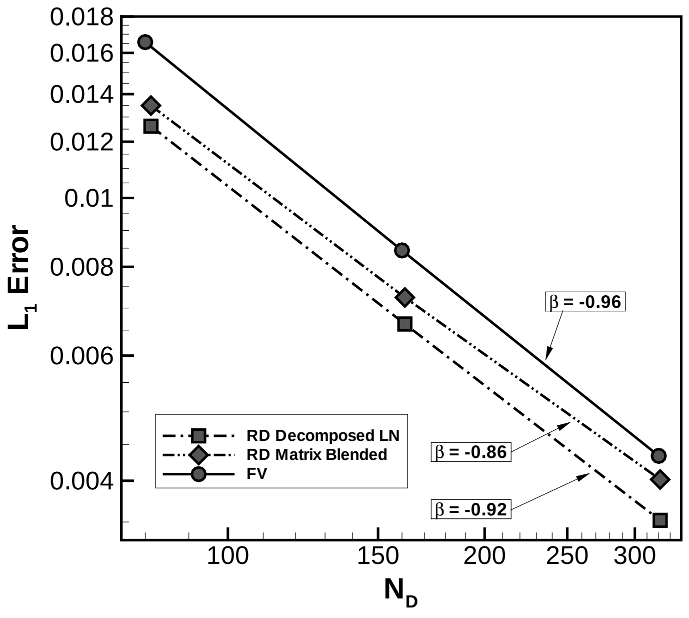


Figure 5.6 L_1-density-error as a function of grid spacing for $M = 3$ supersonic flow past a diamond-shaped aerofoil.

ranges in size from 120 × 80 to 480 × 320. The $\mathcal{RD}$ solutions were obtained using a decomposed LDA-LW (meaning an LDA distribution for the decoupled scalar equations and Lax-Wendroff distribution for the remaining subsonic acoustic subset) scheme, a decomposed LN-LW scheme, a matrix LDA scheme, and a matrix blended scheme.

Contours of the entropy change are shown in Fig. 5.7 for solutions obtained on the coarsest mesh. Qualitatively, the decomposed $\mathcal{RD}$ schemes show the least entropy production, followed by the matrix $\mathcal{RD}$ schemes and finally, the $\mathcal{FV}$ scheme. The spatial convergence is illustrated in Fig. 5.8. The $\mathcal{FV}$ scheme and the $\mathcal{RD}$ schemes that use an LDA distribution all indicate spatial orders of accuracy near $\beta = -2.64$. The reason for the super-convergence is not well understood but the most likely explanation is that the width of the layer in which entropy deviations occur is also a function of the mesh spacing (the L_∞-error is not super-convergent). The matrix $\mathcal{RD}$ schemes perform similarly to the $\mathcal{FV}$ scheme. However, as observed with the LN (minmod) scheme for scalar equations, the matrix blended scheme shows a degraded order of accuracy compared to the matrix LDA scheme. An interesting result in Fig. 5.8 is that the decomposed $\mathcal{RD}$

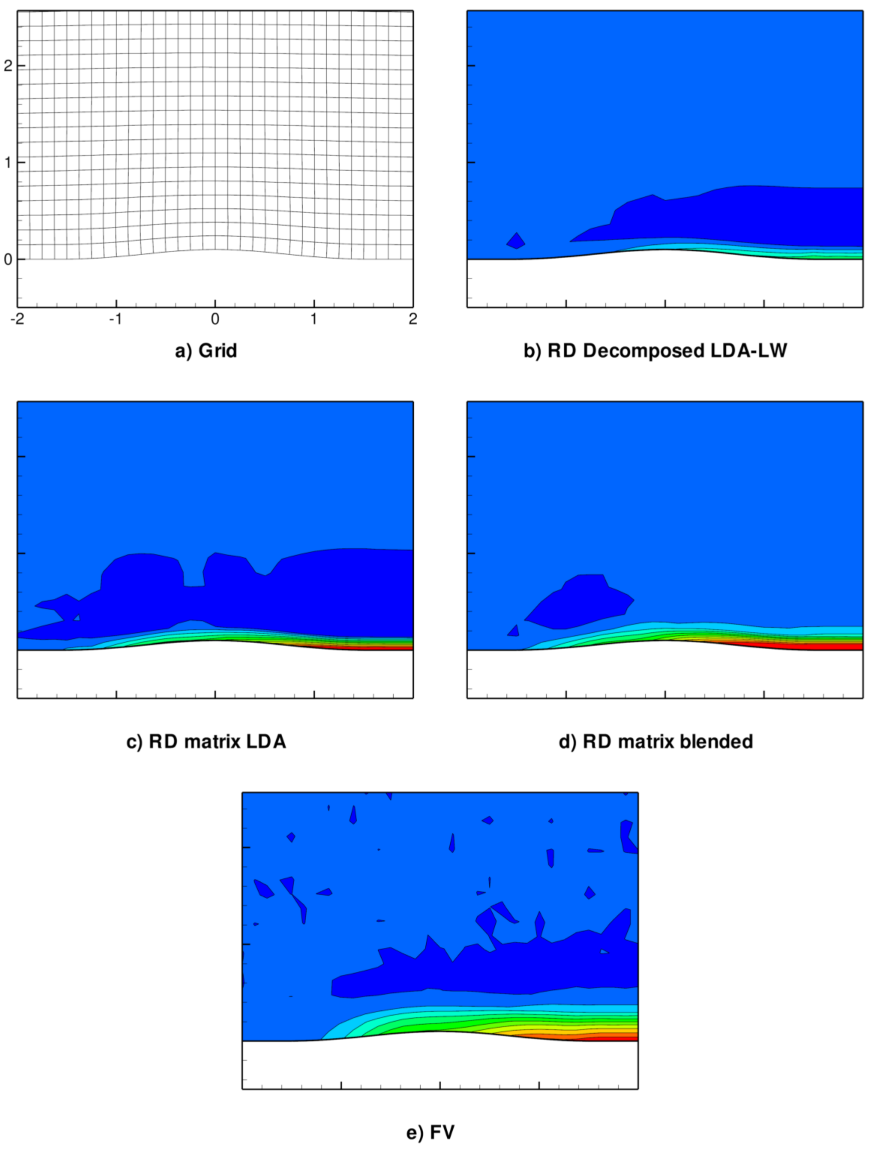


Figure 5.7 Distributions of entropy change for $M = 0.1$ subsonic flow past a smooth bump (mesh size 120×80).

schemes are much more accurate than the other approaches. The decomposed LDA-LW scheme and the decomposed LN-LW scheme provide nearly identical results, probably because this flow is dominated by the acoustic subsystem. In other words, the results are mostly indicative of how well the subsonic acoustic subset is treated. Experiments revealed that it is the hyperbolic-elliptic splitting, rather than the distribution scheme, that is providing most of the benefit; resolving the acoustic subset with an LDA matrix

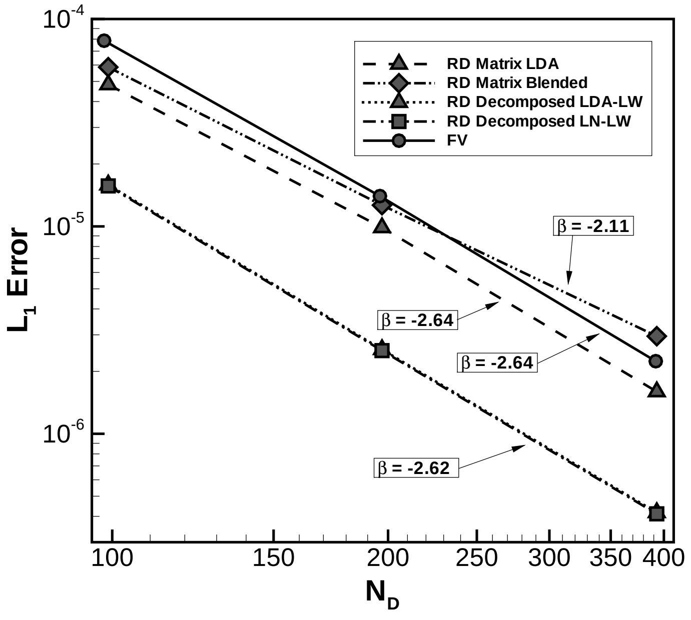


Figure 5.8 L_1-error as a function of grid density for $M = 0.1$ subsonic flow past a smooth bump.

scheme provides similar results to using the LW scheme.

5.3.3 Subsonic Flow Past a Circular Cylinder

The more difficult problem of subsonic flow past a circular cylinder was also considered to include the added complexity of stagnation regions (Fig. 1.4d). System decomposition via hyperbolic/elliptic splitting fails for this case as large instabilities develop near the stagnation regions. Numerical solutions were obtained using both matrix LDA and matrix blended $\mathcal{RD}$ schemes along with the $\mathcal{FV}$ method on multiblock grids ranging from 40×40 to 160×160 in each quadrant.

Contours of the entropy change are shown in Fig. 5.9 for a mesh size of 40×40 in each quadrant. From this figure, it should be quite evident that the $\mathcal{RD}$ LDA scheme produces much less entropy than the $\mathcal{FV}$ scheme. Qualitatively at least, the $\mathcal{RD}$ schemes provide a significantly improved result as compared to the $\mathcal{FV}$ method. Naturally, even greater improvements would be expected if the decomposed $\mathcal{RD}$ scheme could have been

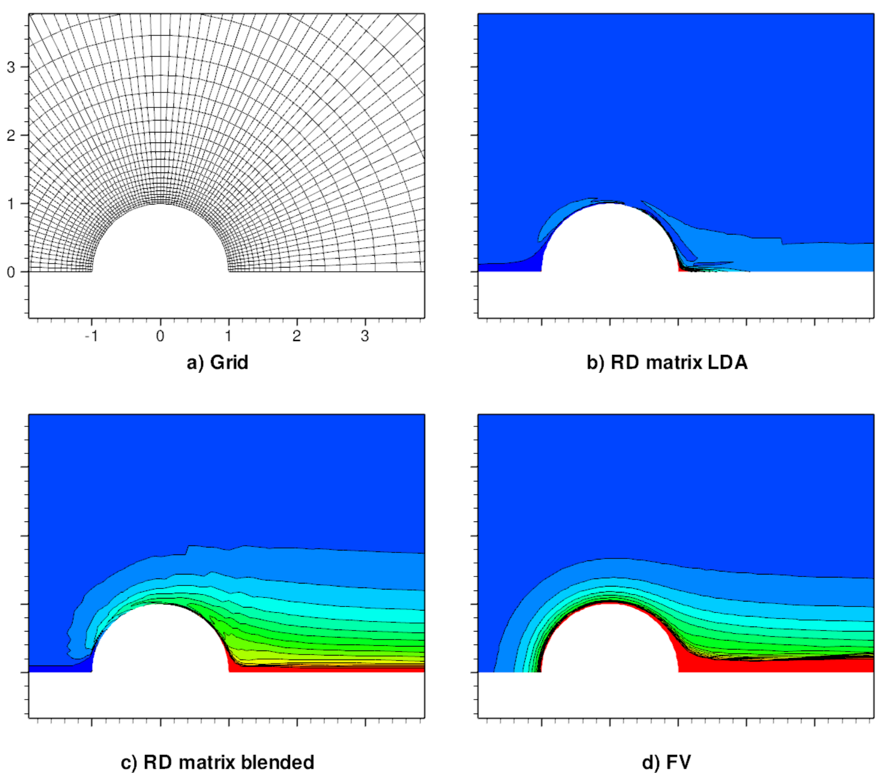


Figure 5.9 Distributions of entropy change for $M = 0.1$ subsonic flow past a circular cylinder (mesh size 40×40 in each quadrant).

used. Quantitative comparisons of the methods are given by the variation of the solution accuracy with respect to mesh density as shown in Fig. 5.10. The results depicted in the figure are not quite as expected. Neither of the $\mathcal{RD}$ schemes yield straight lines. The blended scheme displays an order of accuracy of only 0.77 on the finest meshes, a value much worse than observed for the bump case. While the absolute error of the LDA scheme is indeed less than that of the $\mathcal{FV}$ scheme, the order of accuracy is significantly lower. Although it could be argued that the asymptotic regime for the solution error in terms of the mesh resolution has not yet been achieved on the range of grids considered, this explanation is not supported by the $\mathcal{FV}$ results, for which asymptotic-like behaviour is observed. Instead, a cause for the behaviour of the $\mathcal{RD}$ schemes may be some form of numerical instability. This instability is especially noticeable on the finer meshes.

Figure 5.11 shows the predicted Mach number and entropy distributions on a grid with

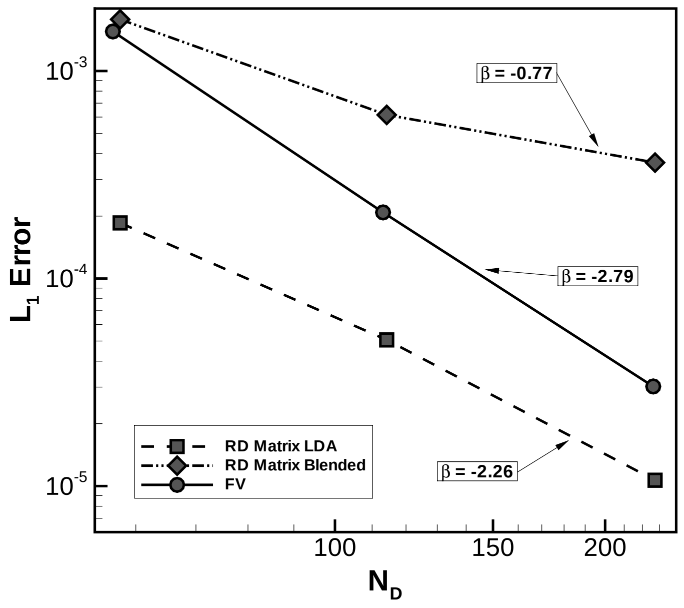


Figure 5.10 L_1-error as a function of grid density for $M = 0.1$ subsonic flow past a circular cylinder.

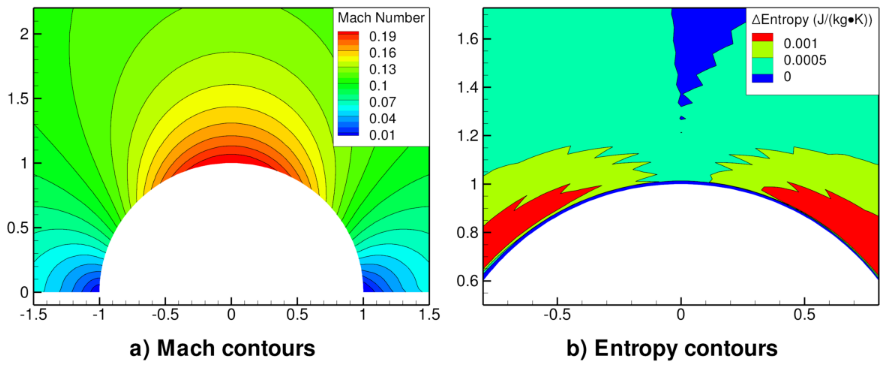


Figure 5.11 Perturbations in the subsonic cylinder flow generated by the $\mathcal{RD}$ LDA scheme (mesh size 80×80 in each quadrant).

80×80 cells in each quadrant. This solution was obtained using the LDA scheme. The L_2 norm for all solution residuals was reduced by 11 orders of magnitude indicating that a steady solution was indeed achieved. Although the Mach number contours are smooth and symmetrical, the entropy solution displays a number of perturbations. Similar, but less significant, perturbations were also observed in the blended results. The oscillations seem to be highly sensitive to the grid. A pre-defined tessellation can exacerbate the situation. Letting the tessellation optimally adapt to the solution tends to minimize the oscillations. An unstructured mesh disrupts the regularity of the perturbations, but their effect is still otherwise present. To our knowledge, this behaviour has not been reported elsewhere in the literature and may be a result of boundary conditions.

5.3.4 Ringleb's Flow

Numerical solutions were obtained for Ringleb's flow using both the $\mathcal{RD}$ and $\mathcal{FV}$ schemes on body fitted grids ranging in size from 40×40 to 320×320. An example grid, coarser than the coarsest grid solved and shown for illustrative purposes only, is displayed in Fig. 5.12. The $\mathcal{RD}$ solutions were obtained using a matrix LDA scheme, a matrix blended scheme, a decomposed LDA scheme, and a decomposed LN scheme. In subsonic regions, the elliptic acoustic subset of the decomposed Euler equations was solved using the Lax-Wendroff approach described previously.

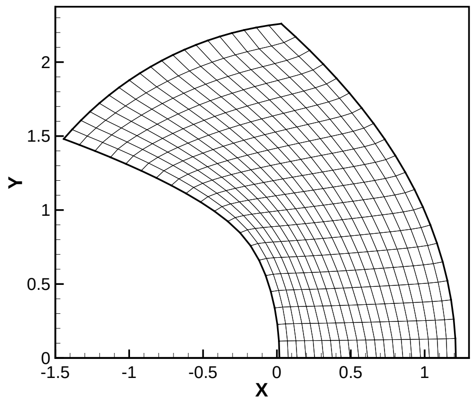


Figure 5.12 Grid for solution of Ringleb's flow.

The blending coefficient or limiter of the nonlinear $\mathcal{RD}$ schemes was frozen when the solution was fully developed. In the previous problems, freezing the limiter would avoid a convergence stall but not appreciably influence the overall accuracy of the results. It was therefore not used. For this problem, freezing the limiter and extending the convergence did actually lead to more accurate estimations of the solution accuracy of the nonlinear $\mathcal{RD}$ schemes.

The L_1-error norms of the computed difference in the solution densities are shown in Fig. 5.13 for the Ringleb's flow problem. The results indicate that the $\mathcal{FV}$ and both of the $\mathcal{RD}$ LDA schemes have spatial orders of accuracy near 2, as should be expected. However, in this case, it is the $\mathcal{FV}$ scheme that yields the best results, providing solutions

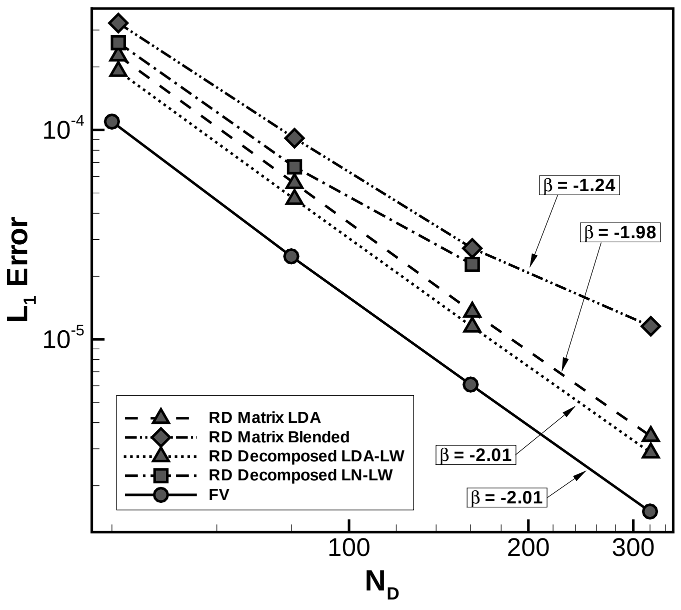


Figure 5.13 L_1-density-error as a function of grid density for Ringleb's flow.

that appear to be almost twice as accurate as the decomposed LDA scheme. We explore this issue further in the next chapter. The performance of the nonlinear $\mathcal{RD}$ schemes is inferior to that of the linear LDA schemes. For the matrix blended method, a degradation in the order of accuracy is seen, especially on finer grids, similar to that observed for other problems. Between the two finest meshes, the order of accuracy is only -1.24. On the finest mesh, 320×320, the decomposed LN method produced oscillations that severely corrupt the accuracy. However, the decomposed LN results that were obtained seem to follow a trend similar to that of the matrix blended results. As with the supersonic case, the decomposed $\mathcal{RD}$ schemes show a slight improvement in absolute accuracy over the $\mathcal{RD}$ matrix schemes.

Chapter 6

Analysis and Corrections of Deficiencies in Residual Distribution Schemes

In the previous chapter, a comparison of the $\mathcal{RD}$ method with the $\mathcal{FV}$ method identified several deficiencies in the $\mathcal{RD}$ method:

1. Degraded spatial accuracy of the nonlinear $\mathcal{RD}$ schemes.
2. Lower accuracy than $\mathcal{FV}$ for Ringleb's flow.
3. Perturbations in solutions of subsonic flow around a cylinder.
4. Instability of the hyperbolic/elliptic decoupling method at stagnation points.

The first two deficiencies relate to accuracy and are examined in this chapter. The latter two relate to issues of robustness and are not addressed. The perturbations generated by the LDA scheme are not reported elsewhere in the literature and no correction is known. More information on the stagnation point instabilities resulting from hyperbolic/elliptic splitting is available in the theses by Mesaros [35] and Rad [46]. In particular, the use of least-square minimization, although complex, appears to avoid the problem.

6.1 Degraded Spatial Accuracy

The most troubling deficiency of the $\mathcal{RD}$ method is the degraded accuracy of the non-linear $\mathcal{RD}$ schemes that are both $\mathcal{P}$ and $\mathcal{LP}$. Monotone and second-order accurate $\mathcal{RD}$ schemes are essential for most practical applications of CFD and the accuracy must be better than or equivalent to the $\mathcal{FV}$ method for the $\mathcal{RD}$ method to be considered as a viable alternative. The problem with limiters degrading the accuracy is also apparent in the $\mathcal{FV}$ framework where it is understood in a one-dimensional context [62] and the solution there applies equally well to the $\mathcal{RD}$ method. The problem is related to the behaviour of symmetric limiters, $\Psi(r)$, in the region where $r \approx 1$. Any limiter which deviates from a constant slope of 1/2 in this region can exhibit first-order error terms; the closer a scheme preserves the slope in this region, the better the accuracy will be. While all the classical limiters are at least tangential to $d\Psi/dr = 1/2$, only the MUSCL limiter [60] (equivalent to the Barth-Jespersen slope limiter in one dimension [10]),

$$\Psi(r) = \max\left[0, \min\left(2r, \frac{r+1}{2}, 2\right)\right] , \tag{6.1}$$

features this constant slope; in a one-dimensional $\mathcal{FV}$ framework, the MUSCL limiter recovers Fromm's scheme for $1/3 \leq r \leq 3$.

The effect of simply using a MUSCL limiter in place of the minmod limiter on an optimally tessellated structured mesh is shown in Fig. 6.1 for circular advection of a Gaussian profile. For the most part, the LN scheme with a MUSCL limiter achieves better accuracy than the linear LDA scheme.

Using a geometrical interpretation of the distribution based on the barycentric coordinates of a distribution point in a triangle, as described by Fig. 2.2, the behaviour of the minmod and MUSCL limiters on a type-II triangle (see Fig. 2.4) are shown in Fig. 6.2. An upwind distribution will not send any fluctuation to vertex 3, hence the distribution point lies along the line connecting vertices 1 and 2. Because the distribution for the N scheme is unbounded as $\phi^E \to 0$, the N scheme may send opposing positive

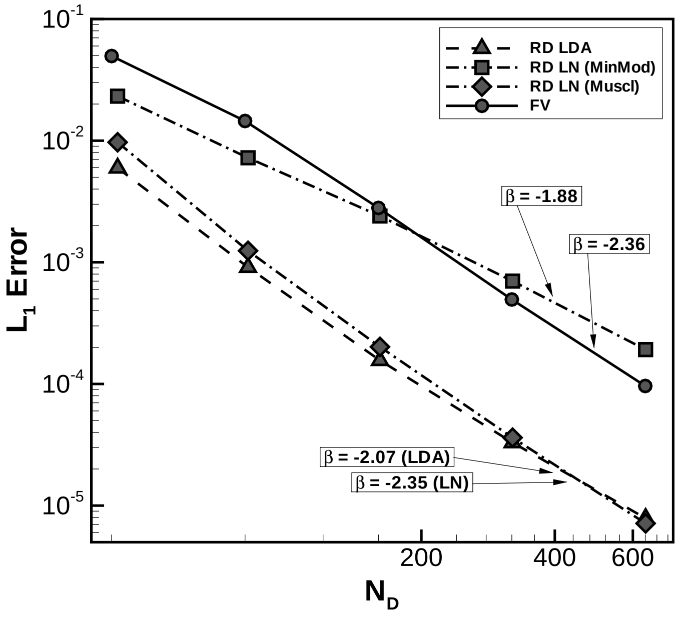


Figure 6.1 Effect of the MUSCL limiter on spatial accuracy for circular advection of an Gaussian profile on an optimally tessellated Cartesian mesh.

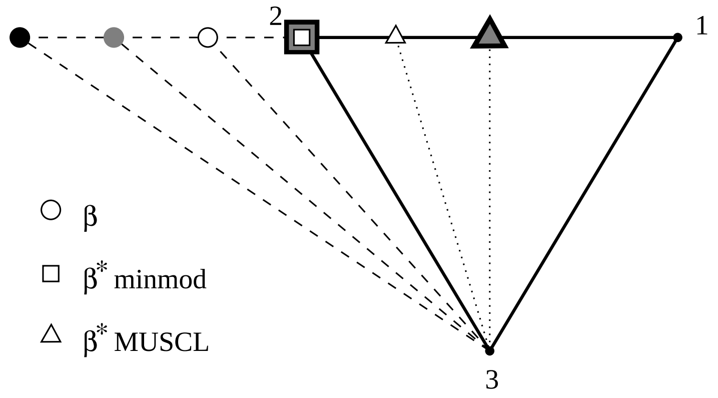


Figure 6.2 Limited distributions, β^*, arising from various "unbounded" distributions of the N scheme, β.

and negative fluctuations to the downstream vertices. In Fig. 6.2, this is represented by the distribution point, as determined by the N Scheme (circle symbol), moving outside the triangle. The limited distribution is shown for the minmod limiter (square symbol) and MUSCL limiter (triangle symbol). Consistent with the map A scheme, the minmod limiter moves the distribution point to the nearest vertex; in doing so, the signs of the distribution coefficients are not changed and the scheme is still positive. The MUSCL limiter reflects the distribution point about vertex 2 until an equal distribution between vertices 1 and 2 is reached. The limited distribution point remains at the center for all unlimited distributions $1/3 \leq r \leq 3$. The MUSCL limiter can change the sign of the distribution coefficients and hence, use of this limiter no longer guarantees property $\mathcal{P}$.

With $r = -\phi_1^N/\phi_2^N$, the MUSCL limiter obtains the same distribution of the fluctuation in the vicinity of

$$r = 1 + \epsilon \tag{6.2}$$

for all small values of $\pm\epsilon$. The minmod limiter, on the other hand will set the distribution point to vertex 2 for $\epsilon < 0$ and to vertex 1 for $\epsilon > 0$. From a geometrical perspective, this seems rational and intuitive (the distribution point is to the left of vertex 2 in Fig. 6.2 for $\epsilon < 0$ and vice-versa), but when written as (6.2), the limiting appears discontinuous. This behaviour is the first hint of trouble but only indicates the possibility for limiter chatter in regions of near constant solution for which $\phi^E \to 0$. Insight into the accuracy can be gained by performing an analysis similar to that of section 2.2.2 where a Taylor series expansion is performed on a representative grid (Fig. 2.8) followed by determination of the truncation error. In this case, however, the distribution is split early in the development for the two-target triangles according to (2.18) and limited via (2.48) with (2.49). The development is provided in Appendix A. By varying the type of limiter and the sign of ϵ in (6.2) for input to the minmod limiter in triangles 1 and 3, the truncation errors displayed in Table 6.1 were obtained. The notation for the truncation error is the same as in section 2.2.2; in particular, s is the aspect ratio of the element and $\lambda = a\hat{\imath} + b\hat{\jmath}$ is the advection velocity. For the minmod limiter, when $\epsilon < 0$, the limiter evaluates to $\Psi(r) = r$, and when $\epsilon > 0$, the limiter evaluates to $\Psi(r) = 1$. It is shown in Table 6.1 that when triangles E_1 and E_3 have values of ϵ with different signs, first-order error terms are introduced into the truncation error. Although less concise, this result is essentially the same as that presented by Waterson and Deconinck [62] for one-dimensional $\mathcal{FV}$ methods.

Table 6.1 Truncation error of various limiters for the LN scheme.

Limiter	$\Psi(r_{E_1})$	$\Psi(r_{E_3})$	Truncation Error
MUSCL	$\frac{r_{E_1}+1}{2}$	$\frac{r_{E_3}+1}{2}$	$O\left(\Delta x^2\right)$
minmod	r_{E_1}	r_{E_3}	$O\left(\Delta x^2\right)$
	1	1	$O\left(\Delta x^2\right)$
	r_{E_1}	1	$\frac{ab(as-b)\Delta x}{4\lvert\lambda\rvert^2}\frac{\partial^2 u}{\partial \eta^2}+O\left(\Delta x^2\right)$
	1	r_{E_3}	$\frac{ab(b-as)\Delta x}{4\lvert\lambda\rvert^2}\frac{\partial^2 u}{\partial \eta^2}+O\left(\Delta x^2\right)$

The LN scheme with a MUSCL limiter is very effective in the two-dimensional case; however, as mentioned in section 2.3.3, the LN scheme cannot be extended to higher dimensions. It is therefore of interest to find a mapped scheme which geometrically extends the limiting concept to any number of dimensions. The most logical extension of the MUSCL limiter to multiple dimensions is to similarly reflect the negative distribution coefficients to positive values until a centered distribution, $\beta_i^{\text{Central}} = 1/(d+1) = 1/3$, is reached. In this proposed approach, from here on labelled "map C", three distribution points are used: β^{Central}, β^N, and $\beta^{\,\text{map}A}$. From the geometrical interpretation, the lengths of the two line segments, $L_1 = \left|\beta^N - \beta^{\,\text{map}A}\right|$ and $L_2 = \left|\beta^{\,\text{Central}} - \beta^{\,\text{map}A}\right|$ as shown in Fig. 6.3, are used to define a blending coefficient,

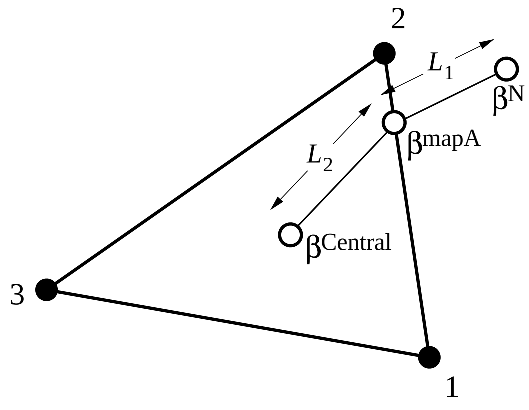


Figure 6.3 Geometrical lengths used to computed the blending coefficient for the map C scheme.

$$\theta = \min\left(1, \frac{\left|\beta^N - \beta^{\,\text{map}A}\right|}{\left|\beta^{\,\text{Central}} - \beta^{\,\text{map}A}\right|}\right). \tag{6.3}$$

The limited distribution of the map C scheme is then given by

$$\beta_i^{\text{map}C} = \beta_i^* = \theta\beta^{\mathcal{LP}} + (1-\theta)\,\beta_i^{\text{map}A}, \tag{6.4}$$

where $\beta^{\mathcal{LP}}$ is either β^{Central} or, optionally, any other linear $\mathcal{LP}$ scheme. When the distribution point given by β^N is inside the triangle, $\beta^N = \beta^{\text{map}A}$, resulting in $\theta = 0$ and $\beta^{\text{map}C} = \beta^N$. When $L_1 \geq L_2$, $\theta = 1$ and $\beta^{\text{map}C} = \beta^{\mathcal{LP}}$. This should only occur when $\phi^E \approx 0$. Otherwise, the distribution point is moved along a linear path from $\beta^{\text{map}A}$ to $\beta^{\mathcal{LP}}$ a distance corresponding to the ratio given by (6.3). The resulting scheme is similar to the blended scheme except with a positive component of $\beta_i^{\text{map}A}$ in place of β_i^N and with a blending coefficient instead derived according to geometrical similarities with the LN scheme and a MUSCL limiter.

A scheme bearing some similarities to map C was suggested by Abgrall and Roe [9] where a modification was made to the map A scheme as follows:

$$\hat{\beta}_i^{\text{map}A} = \frac{2\phi_i^{N+} + \epsilon}{\sum_{j=1, j\in E}^{3} (2\phi_j^{N+} + 3\epsilon)} \qquad \phi_j^{N+} = \begin{cases} \phi_j^N & \phi_j^N \phi^E \geq 0 \\ 0 & \phi_j^N \phi^E < 0 \end{cases}, \tag{6.5}$$

where $\epsilon = 10^{-10}$ in double precision. In regions of constant solution, where both $\phi^E \approx 0$ and $\phi_i^N \approx 0$, this scheme will set $\hat{\beta}_i^{\text{map}A} = 1/3$. For elements where the distribution coefficients of the N scheme diverge, it will closely reproduce the map A scheme. At least in near constant regions, this will have a similar effect to the map C scheme and the authors of [9] state that the modification was required to obtain the expected orders of accuracy. However, the rationale given for (6.5) was to ensure that $\sum_i \hat{\beta}_i^{\text{map}A} = 1$ exactly. Based on the analysis given in this chapter, it may be that limiting towards an $\mathcal{LP}$ distribution is instead the reason for obtaining better accuracy.

6.1.1 Accuracy

The accuracy of the map C scheme is assessed here for linear advection and Ringleb's flow at both second and fourth orders of accuracy. Possible candidates for the $\beta^{\mathcal{LP}}$ target of (6.4) are the Central, LW, LDA, and UCV scheme. The central scheme is excluded because it leads to an unstable map C scheme when applied to to Ringleb's flow. However

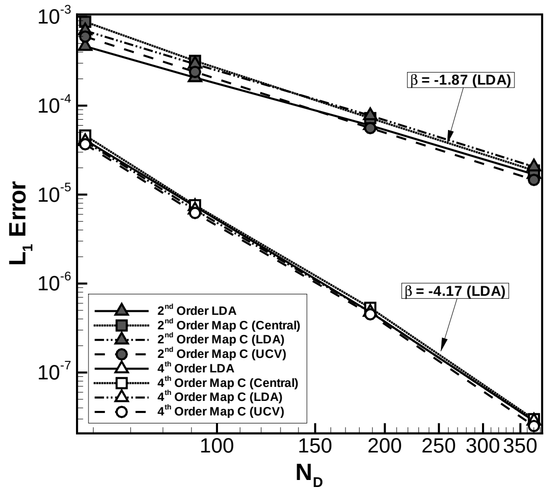


Figure 6.4 L_1-Error norm of map C algorithms for linear advection of a Gaussian profile on unstructured meshes.

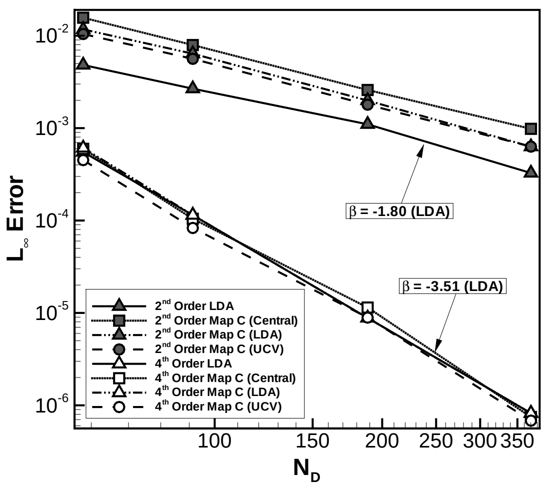


Figure 6.5 L_∞-Error norm of map C algorithms for linear advection of a Gaussian profile on unstructured meshes.

results are shown for linear advection. The LW scheme is also not considered because it may place the distribution point outside the geometry of the triangle, something that is unsettling, even if workable, because our entire effort is to get the point inside the triangle. Unlike in Chapters 2 and 3, solutions here were obtained using the full interior and boundary schemes.

Figures 6.4 and 6.5 illustrate the L_1-error and L_∞-error, respectively, of the map C scheme applied to a problem of linear advection of a Gaussian profile on *unstructured* meshes. Both second and fourth-order results are shown for the different $\mathcal{LP}$ schemes as a central target of the limiting. For reference, the error of the LDA scheme is also shown. All the map C schemes provide similar accuracy to the LDA scheme and only for second order L_∞-error is the LDA scheme slightly more accurate. From this case, it appears that all $\beta^{\mathcal{LP}}$ targets would be suitable candidates as far as accuracy is concerned.

The LDA and UCV target $\mathcal{LP}$ schemes are additionally compared in Figs. 6.6 and 6.7 for matrix solutions of Ringleb's flow. Each equation is limited using the scalar method defined by (6.3) and (6.4) by first projecting the fluctuations onto the eigenvectors of a one-dimensional system in the direction of the velocity vector, exactly as described for the

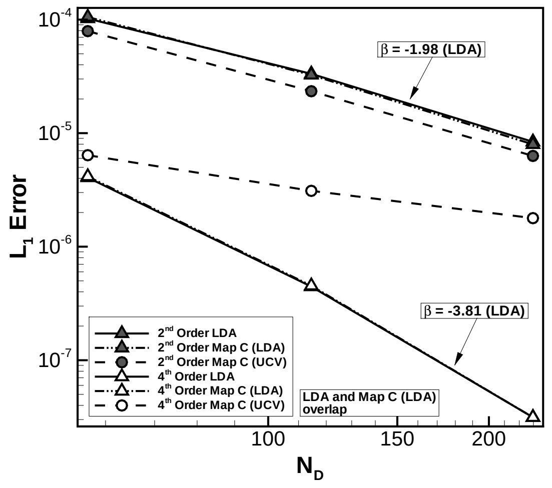


Figure 6.6 L_1-Error norm of map C algorithms for Ringleb's flow on unstructured meshes.

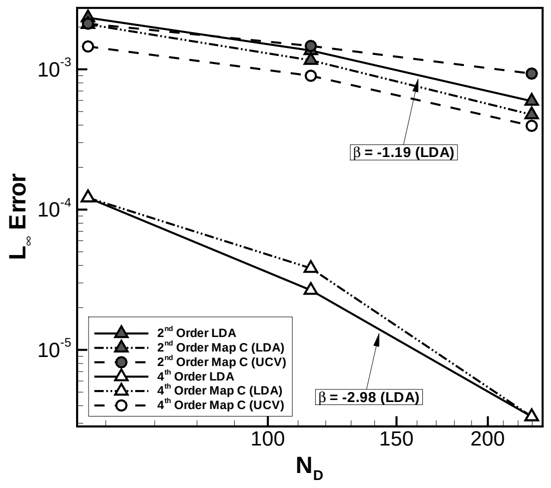


Figure 6.7 L_∞-Error norm of map C algorithms for Ringleb's flow on unstructured meshes.

map A scheme in section 3.3.2. From the figure, limiting towards an LDA distribution is almost identical to using the LDA scheme itself, with only minor variations in the L_∞-error. For unknown reasons, perhaps because it is not fully upwind, using a UCV distribution as the target for the map C limiting was not successful at higher orders of accuracy. In Figs. 6.6 and 6.7, the order of convergence for both mapC (UCV) error norms are close to first-order when fourth-order is expected. Based on these results, an LDA distribution was chosen as the $\mathcal{LP}$ target of the map C scheme (as defined in Eq. (6.4)) for all remaining numerical experiments.

6.2 Monotonicity

The construction of the map C scheme clearly sacrifices monotonicity. In this section, the severity of the compromise is examined by considering solutions of Burgers equation (Fig. 1.3c) and the Euler equations for a single shock wave (Fig. 1.4a). Contours from second-order predictions of Burgers equation are shown in Fig. 6.8. The contours vary by 0.2 around a solution minimum and maximum of -0.5 and 1.5, respectively. The map

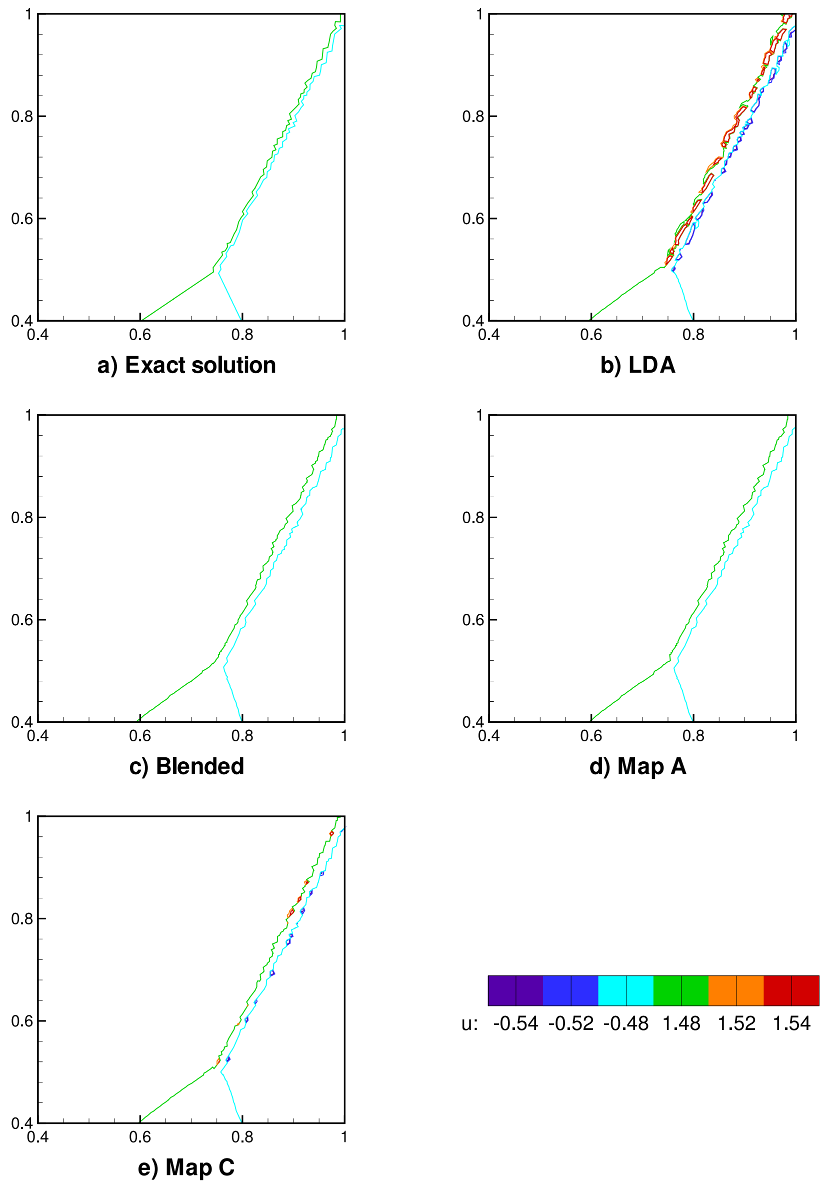


Figure 6.8 Monotonicity of various $\mathcal{RD}$ schemes for solutions of Burgers equation.

C scheme shows significantly more oscillations than either the blended or map A schemes but significantly less than the LDA scheme. Table 6.2 lists the global minimum and maximum in each solution. Note the small undershoot by the blended scheme and the perfect monotonicity of the map A scheme. The extrema predicted by the map C scheme are a bit disappointing with all values being a significant fraction of the LDA scheme.

The monotonicity of Euler equation solutions computed for an oblique shock is shown

Table 6.2 Global extrema for solutions of Burgers equation.

Scheme	Minimum	Maximum
Exact	-0.5	1.5
LDA	-0.77	1.88
Blended	-0.52	1.5
Map A	-0.5	1.5
Map C	-0.62	1.61

by pressure contours in Fig. 6.9. The contours vary by intervals of 1000 Pa around the solution minimum and maximum of 101325 Pa and 210444 Pa. The results are similar to the advection case for both hyperbolic/elliptic splitting and the matrix method. The global minimum and maximum for each solution are available in Table 6.3. For systems, all schemes exhibit at least some minor oscillations. Again, we had hoped for better monotonicity in solutions obtained by the proposed map C scheme.

Table 6.3 Global extrema (Pa) for solutions of an oblique shock.

Scheme		Minimum	Maximum
Exact		101325	210444
H/E Split	LDA	72462	246052
	Blended	101038	214397
	Map A	101325	213972
	Map C	94669	227700
Matrix	Blended	101095	215180
	Map A	101286	218844
	Map C	96462	223180

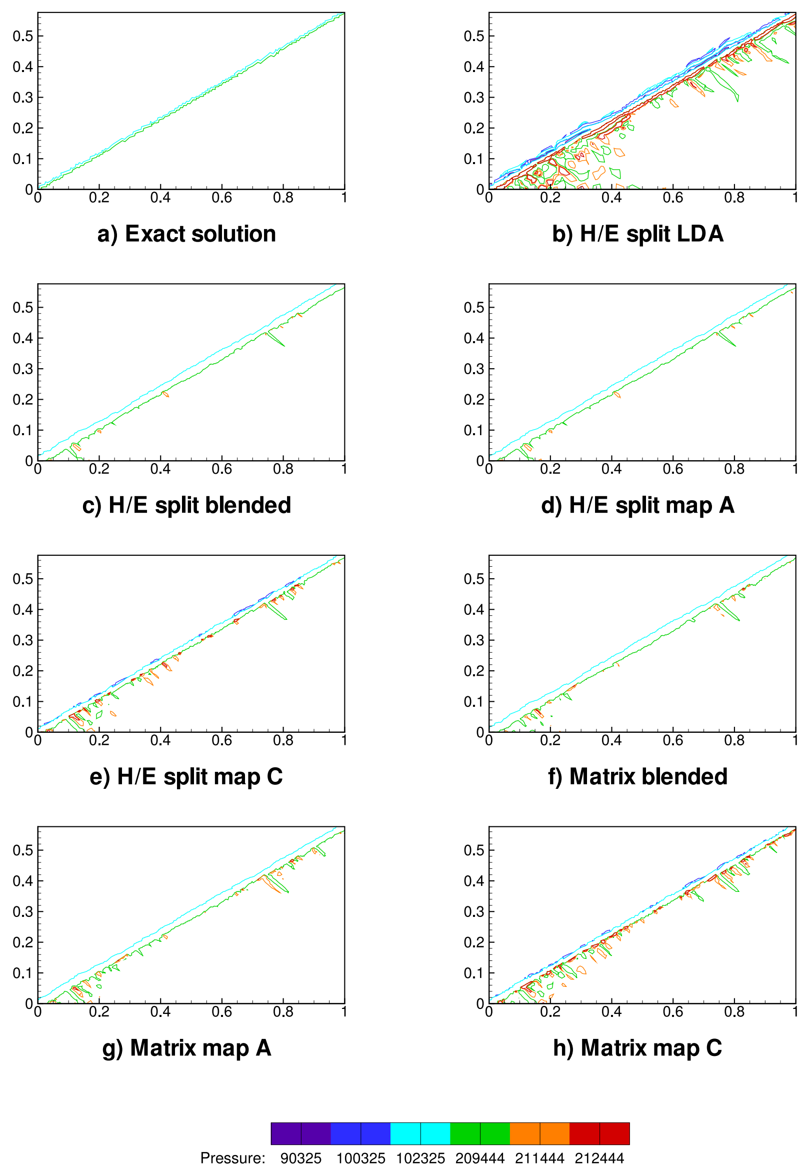


Figure 6.9 Monotonicity of various $\mathcal{RD}$ scheme for solutions of an oblique shock wave.

There are techniques one can use to increase the monotonicity-preserving characteristics of the map C scheme. A simple modification is to increase the distance that a distribution point, as computed by the N scheme, must be outside the perimeter of the triangle before the limited distribution point is blended with the chosen $\mathcal{LP}$ scheme. For example, replace the numerator in (6.3) with

$$\max\left(0, \left|\beta^N - \beta^{\mathrm{map}A}\right| - \alpha\right) \,, \tag{6.6}$$

where α assumes some appropriate value. This will improve the monotonicity of a solution but can have a large impact on the accuracy. We found it nearly impossible to obtain satisfactory monotonicity and still obtain the fourth-order accuracy shown in Figs. 6.6 and 6.7 for Ringleb's flow.

Another possible approach is to use some form of explicit discontinuity detection; possible techniques are discussed by Abgrall [3] and Guzik and Groth [29]. For elements where discontinuities exist, the blending coefficient can be determined using the blended scheme. Since both schemes blend between a $\mathcal{LP}$ scheme and a $\mathcal{P}$ scheme, it may be possible to easily interchange the coefficients.

However, for this study, (6.3) remains unaltered and we make no further effort to improve the monotonicity. Instead, in Chapter 8, the performance of the map C scheme defined by (6.3) and (6.4) is examined for the canonical test problems at both second and fourth order accuracy. Improvements to the monotonicity of the scheme are left to future work.

6.3 Poor performance of Ringleb's Flow

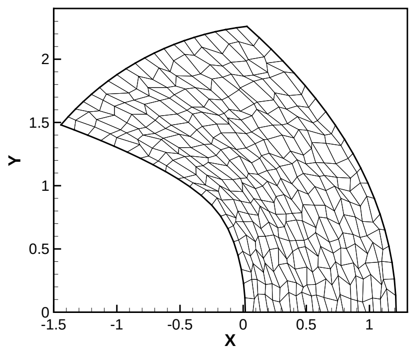


Figure 6.10 Distorted grid for solution of Ringleb's flow.

In Chapter 5, it was shown that the $\mathcal{FV}$ method delivers solutions that are consistently more accurate than the $\mathcal{RD}$ solutions. The discrepancy regarding the relative accuracies of $\mathcal{RD}$ and $\mathcal{FV}$ methods is known to not be related to mesh alignment. This was proven by generating solutions on randomly distorted grids, similar to that shown in Fig. 6.10, in place of the smooth mesh shown in Fig. 5.12. The results for an unlimited $\mathcal{FV}$ method are compared to those for the matrix-LDA $\mathcal{RD}$ scheme on the distorted mesh in Fig. 6.11. It should be

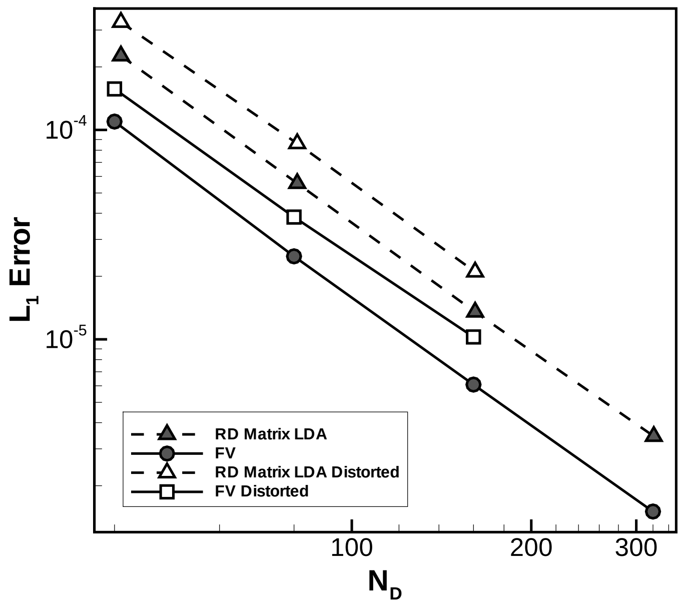


Figure 6.11 L_1-density-error as a function of grid density for Ringleb's flow on a distorted mesh.

evident from this figure that while second-order accuracy is still achieved on the distorted grids, there is a notable decrease in the solution accuracy of both methods. However, the overall effect on each of the methods is quite similar such that the $\mathcal{FV}$ scheme is still more accurate than the $\mathcal{RD}$ scheme.

The poor performance of the $\mathcal{RD}$ schemes for Ringleb's flow is thought to be related to how well the schemes correct solutions in regions that violate the entropy condition. Entropy violations result from the discretization and typically occur when a vertex receives no update because all waves move away from the vertex. This commonly occurs at sonic points and, in some cases, can lead to the formation of expansion shocks [53].

Figures 6.12a and 6.12b show the absolute magnitude of the density error for the $\mathcal{FV}$ and $\mathcal{RD}$ schemes, respectively. It is apparent that the $\mathcal{RD}$ scheme suffers from higher error levels near sonic points. The entropy fix proposed by Sermeus and Deconinck [53] was implemented for the matrix scheme. Regions where an entropy fix may be required, as detected by this scheme, are shown in Fig. 6.12c. The effect of applying the entropy fix is displayed in Fig. 6.12d. Although almost indistinguishable from Fig. 6.12b, there is

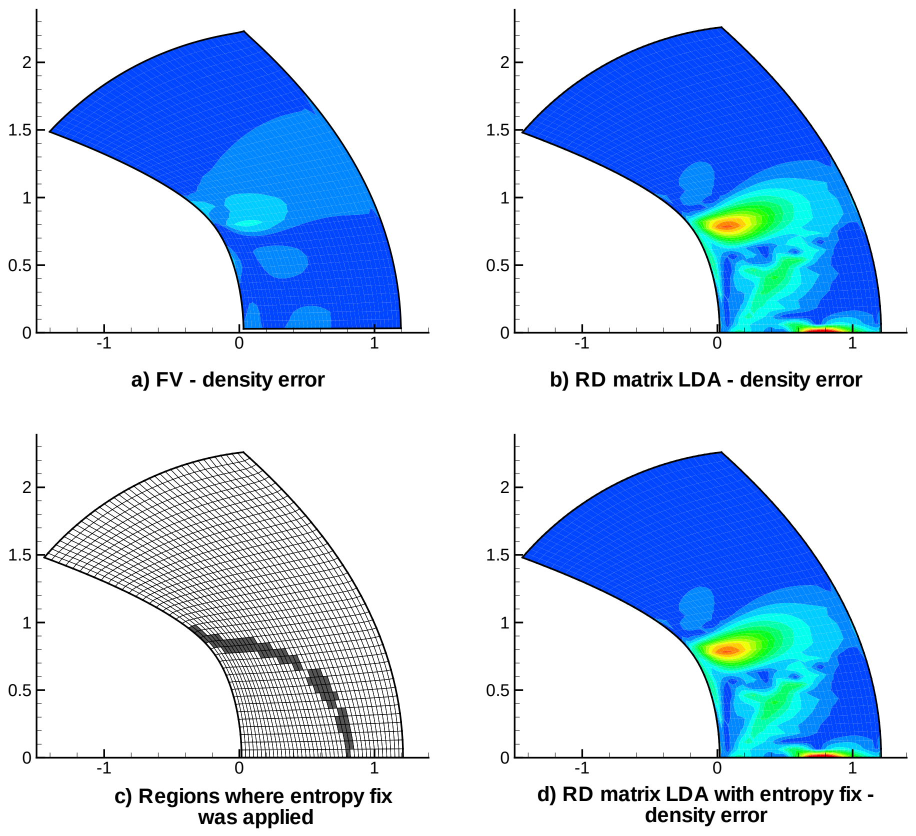


Figure 6.12 Distributions of the density error generated by the $\mathcal{RD}$ and $\mathcal{FV}$ schemes and performance of an entropy fix.

a small improvement to the accuracy. Most of the improvement is obtained in the region of large error at the center of the domain. The large error at the outflow boundary is caused by an entropy violation within the degenerate ghost elements and is not dealt with at all by the entropy fix.

The $\mathcal{FV}$ scheme is more accurate than the $\mathcal{RD}$ scheme even when an entropy fix is applied. It seems likely that the results reported in Fig. 5.13 are more indicative of how well the two schemes satisfy the appropriate entropy condition than anything else. The $\mathcal{FV}$ method is much more successful, probably because of its larger stencil. In one dimension, a second-order reconstruction will tend to remove most entropy violations. The $\mathcal{RD}$ scheme, on the other hand, must completely rely on accurate multidimensional analytical approximations of the solution in regions of entropy violation. Besides the entropy correction considered here, fixes have also been proposed by Wood and Kleb [65] and Nishikawa and Roe [40].

Chapter 7

Heterogeneous Parallelism Using Central and Graphics Processing Units

High-order $\mathcal{RD}$ schemes are attractive candidates for computing on graphics processing units (GPUs) because the compact stencil provides parallel data and the high-order extension dramatically increases the arithmetic intensity of the computation. With these two characteristics, data parallelism and arithmetic intensity, an algorithm may be better suited to the highly parallel architecture of a GPU rather than the serial architecture of the CPU. In this chapter, the potential for computing portions of a $\mathcal{RD}$ algorithm on a GPU is explored. The algorithm is tested by applying an LDA matrix scheme to Ringleb's flow (Fig. 1.4e). The results in this chapter often relate to the three principal steps of the $\mathcal{RD}$ algorithm as described in Chapter 2 and repeated here: computation of the fluctuation, distribution of the fluctuation, and evolution of the solution. The target architecture for this research is a distributed cluster with an arbitrary number of identical compute nodes. As illustrated in Fig. 7.1, each node is assumed

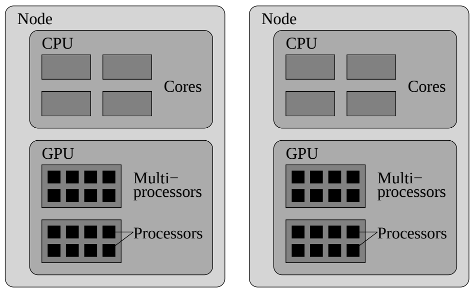


Figure 7.1 Parallel architecture.

to contain a number of CPU cores sharing random-access memory (RAM) and a GPU with its own dedicated RAM. The numerical implementation features four levels of parallelism. The computational grid for the problem is divided into zones and the coarsest level (1st) of parallelism is defined by the set of zones distributed to each node. Within a node, the zones in the set are distributed among the CPU cores. An intermediate level (2nd) of parallelism is characterized by the simultaneous processing of the GPU and CPU cores. The finest levels (3rd and 4th) of parallelism are expressed by the vector processing of the GPU.

The 1st level of parallelism, involving communication between the nodes, is implemented with MPI. The 2nd level involves communication on a single node and is entirely contained within a process. The parallelism at this level is expressed using POSIX threads. A single thread is fixed to each CPU core residing on the node. As well, a single thread, not assigned to any particular core, is created on the node for the sole purpose of controlling the processing of the GPU; this thread remains mostly idle. The next two sections described the complete parallel implementation for a cluster of nodes. However, having access to only one node for this thesis work, our results focus only on the parallel interaction between the CPU cores and the GPU within a single node. Specifically, our test node consists of a quad-core AMD Phenom II 940 CPU and an NVIDIA GTX 260 (27 multiprocessors and 896 MB) GPU. Predictions of computational performance are presented in the last section.

7.1 Parallel Implementation

For the 1st and 2nd levels of parallelism, the domain is partitioned into a number of zones. An example of a partitioned mesh is shown in Fig. 7.2. The meshes used in this study were all generated with Gmsh [24] and partitioned between reconstruction elements using the spectral partitioning algorithm of Chaco [30].

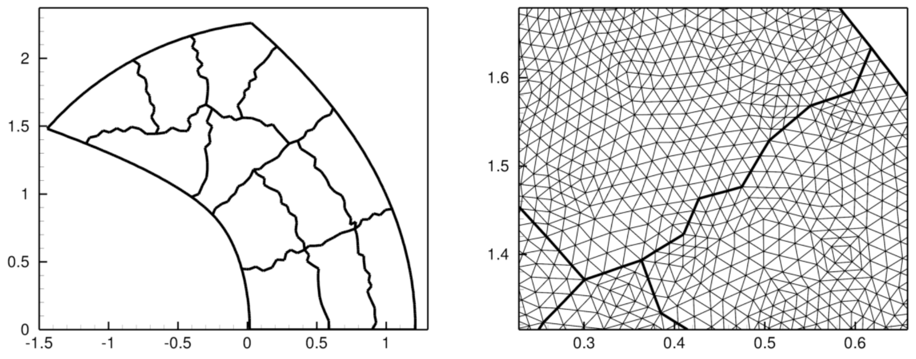


Figure 7.2 Sample mesh and partitions used for parallel computation of Ringleb's flow.

7.1.1 Parallel Communication Strategy

The partitioning of the domain into zones splits elements of the dual mesh (see Fig. 2.3) for vertices that reside on interior zone boundaries. In each zone, the dual mesh elements need to be rebuilt with the missing information from neighbour zones. This is done by locally accumulating and then communicating the required information: the fluctuation distributed to the vertex, the area associated with the vertex, and the time-step constraint. Figure 7.3 illustrates how this is accomplished using MPI for communication between the compute nodes and POSIX threads for communication between the threads executing on the CPU cores within a compute node. Five zones are shown in Fig. 7.3 with zones 1 and 2 belonging to node 1 and zones 3–5 belonging to node 2. An *interface* is defined for all vertices on interior boundaries that belong to a common set of nodes. In Fig. 7.3, a dual interface connects the five vertices between zones 1 and 5 and a multiple interface connects all the zones to one vertex. For two-dimensional problems, a dual interface only connects two zones while a multiple interface can connect any number of zones, but only for one vertex. A copy of the interface exists on all nodes holding zones that share the interface. We label these as *sibling* interfaces. Within each interface there are two buffers for accumulating data: a local buffer and a window buffer. Once the distribution of the fluctuation has been completed for a zone, the necessary data is

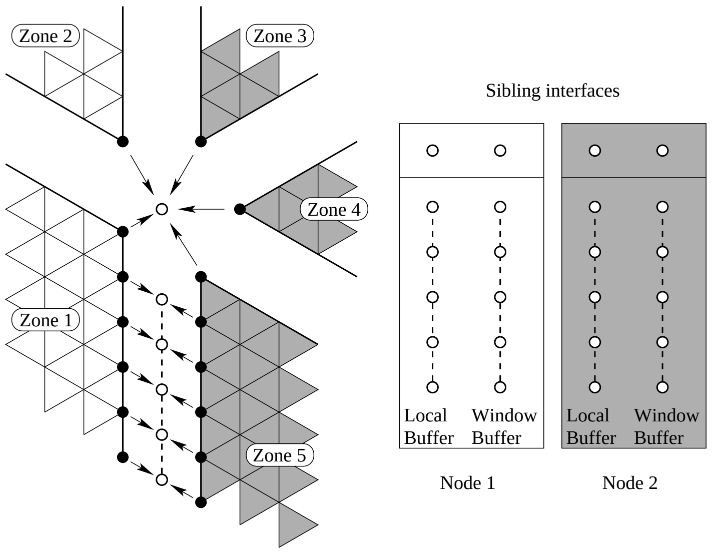


Figure 7.3 Communication between zones is accomplished by defining interfaces between the zones.

accumulated in the local buffer of all interfaces shared by the zone. A mutex[1] is assigned to each interface to isolate the updates from the threads on the node. When an interface has received updates from all zones local to the node, the last thread to update the zone initiates an MPI remote memory access (RMA) operation that accumulates information from the local buffer of the local interface into the window buffer of any sibling interfaces on remote nodes. The RMA operations are one-sided and do not require cooperation from the receiving nodes. However, the window buffer is marked as a special *window* of memory that can be written to by remote processes. Once all interfaces have been updated by all nodes, the window buffer is added to the local buffer in each interface. The local buffer now holds all information required to advance the solution on the dual mesh. This information is relayed to the zones belonging to an interface allowing for each zone to independently advance the solution at all of its vertices.

An example is given for the multiple interface shown in Fig. 7.3. Node 2 will process zones 3–5. After completing the distribution of the fluctuation for zone 3, the fluctuation sent to the shared vertex, the area associated with the shared vertex, and the time-step

[1]A locking algorithm that allows only one thread to update the interface at a time.

constraint are all accumulated into the local buffer of the interface on node 2. The same is done for zones 4 and 5. Once the information from zone 5 is added, the local buffer is complete on node 2. The local buffer is then added to the window buffer of the sibling interface on node 1 using RMA operations defined by MPI. Meanwhile, node 1 has processed and gathered information from zones 1 and 2 (and, when complete, added the data to the window buffer of the sibling interface on node 2). By adding the window buffer to the local buffer, node 1 now has information from all five zones that share the vertex. The information from the interface is now copied by node 1 to zones 1 and 2 before proceeding with the evolution of the solution.

Although RMA operations are relatively new to MPI, and perhaps not as efficient as well-established cooperative instructions, the preceding communication pattern has some advantages. First, it is simple to implement because coordination is not required between pairs of processes. Second, completion of the local buffers should occur at irregular time intervals. Hence, the RMA operations will be issued at irregular intervals while zones are being processed. This allows for some overlap between communication and computation. As well, we speculate that sending messages at irregular intervals throughout the entire duration of the processing should alleviate saturation of the communication network. However, we have not had the opportunity to investigate the actual performance of this communication pattern.

7.1.2 Subdivision of the Algorithm Between CPU and GPU

For execution at the 2nd level of parallelism, the algorithm is subdivided depending on which type of processor a portion of the algorithm is best suited for. The imposition of fourth-order accuracy provides for high arithmetic intensity and the regular compact structure of the $\mathcal{RD}$ stencil provides highly parallel data. For example, the integrations in 9 primary elements can be completed on the data of one reconstruction element. Figure 7.4 shows a profile of the $\mathcal{RD}$ algorithm executing on a quad-core CPU. The three

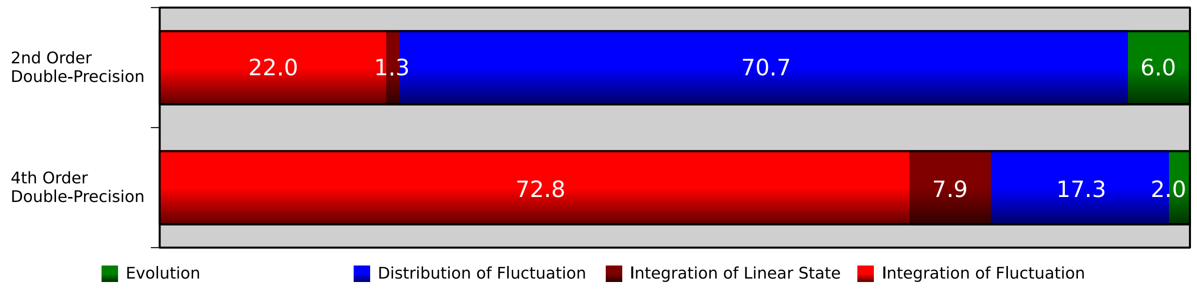


Figure 7.4 Profile of the fourth-order $\mathcal{RD}$ algorithm solving Ringleb's flow on a quad-core CPU.

distinct steps of the $\mathcal{RD}$ method, as described in Chapter 2, are identified in the figure: computation/integration of the fluctuation (red), distribution of the fluctuation (blue), and evolution of the solution (green). The computational cost of the two quadratures (integration of the fluctuation and integration of the linear state) increases dramatically when switching from second-order accuracy to fourth-order accuracy. While the quadratures are extremely expensive, the algorithms involved are both simple and easily parallelized. At the 2nd level of parallelism, we therefore chose to perform the quadratures on the GPU while all other tasks concurrently execute on the CPU cores. The results in Fig. 7.4 were generated using profiling capabilities provided with the compiler; precise timings of the code indicate that the quadratures in the fourth-order algorithm require 75 % of the processing time. According to Amdahl's law [15], which can be written as

$$S = \frac{1}{1 - f_p + \frac{f_p}{S_p}}, \tag{7.1}$$

where f_p is the parallel fraction and S_p is the speedup of the parallel fraction, the maximum possible speedup, S, of the overall algorithm for infinite speedup of the quadratures ($S_p = \infty$) is four.

During each iteration of the algorithm, a streaming process is defined where the GPU integrates the fluctuation and linear state in a *set of zones*, equivalent to the number of CPU cores, before distributing the zones to the CPU cores. The CPU cores then distribute the fluctuation while the GPU simultaneously begins integration of the next set of zones. Because of global time-stepping (which is retained, as opposed to local time-stepping, only to allow experimentation with the added complexity), the solution cannot be evolved until the residual has been distributed for all zones and the global minimum of the time-step is determined. To ensure concurrent processing, the GPU is tasked to integrate the first set of zones for the *next* iteration as soon as the CPUs finish evolving the solution for the first set. Meanwhile, the CPUs can evolve the solution for the remaining zones in the *current* iteration.

Synchronization between the threads which process the zones on the CPU cores (labelled CPU threads) and the thread that controls the GPU (labelled GPU host thread) is achieved using the series of barriers shown in Fig. 7.5. Before entry at the top of the figure, the CPU threads are processing the distribution for a set of zones while the GPU is performing the quadratures for the next set of zones. As the CPU threads finish computing, they enter the top of the figure. The first CPU thread to arrive sets barrier C and removes barrier A. This allows the GPU host thread, which has been idle while the CPU threads compute, to activate and check the status of the GPU. Once the computation on the GPU is complete and the results transferred to main memory, the GPU thread sets barrier D and removes barrier B. Meanwhile, the CPU threads

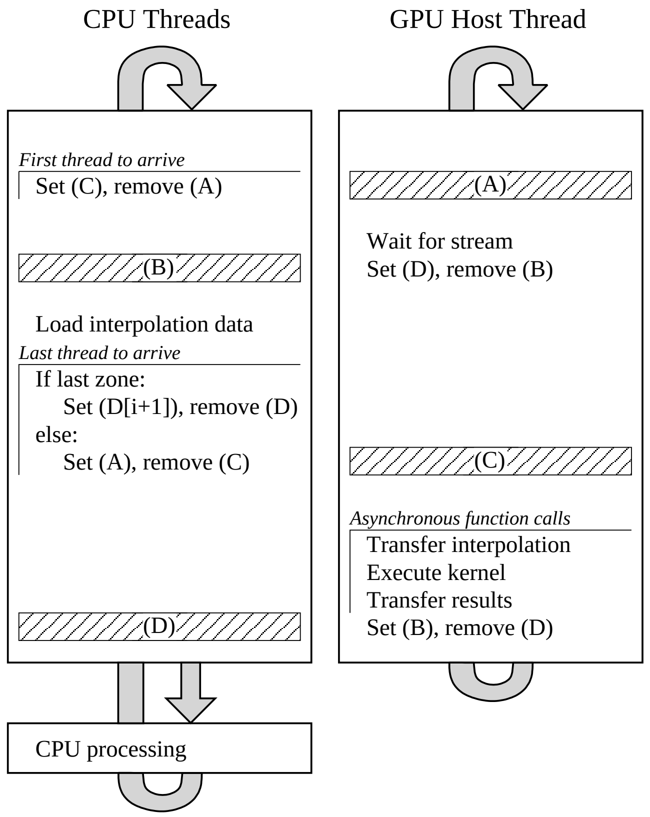


Figure 7.5 Synchronization between threads executing on the CPU cores and a thread managing the GPU.

have been collecting at barrier B. They are now able to proceed with loading interpolation data that the GPU will process next. The last thread to load the data will set barrier A and remove barrier C. The GPU host thread now invokes a number of asynchronous functions (the functions do not wait for completion before returning). These functions initiate the transfer of the interpolation to global memory on the GPU, execute the kernel (the program) on the GPU, and transfer the results back to main CPU memory. After executing these functions, the GPU host threads sets barrier B and removes barrier D. Because the CPU threads are waiting idle at barrier D, the GPU host thread does not have to compete for resources on the CPU. When barrier D is lifted, the CPU threads proceed to distribute the fluctuation using the data just obtained from the previous execution on the GPU. Meanwhile, the GPU host thread goes to sleep at barrier A.

If the CPU threads are about to process the last set of zones received from the GPU, then there is nothing for the GPU to do; it cannot process the next iteration until the global time step is known and the solution has been advanced. In this event, the GPU host thread is left idle at barrier C and the last CPU thread to arrive at barrier D sets barrier D for the *next* entry into the synchronization routine and removes barrier D for *this* entry. The CPU threads will distribute the fluctuation for the last set of zones and compute the global time step. Once the solution has been evolved for the first set of zones, all CPU threads will re-enter the synchronization routine. They will fall through barrier B and proceed to load the interpolation for the first set of zones for the GPU. The GPU will now compute the fluctuation for the first set of zones while the CPU threads finish advancing the solution in all remaining zones. Ideally, the GPU will finish its task before the CPU threads finish theirs. Global time-stepping can be seen to add complexity here and possibly cause inefficiency if concurrent processing is inhibited.

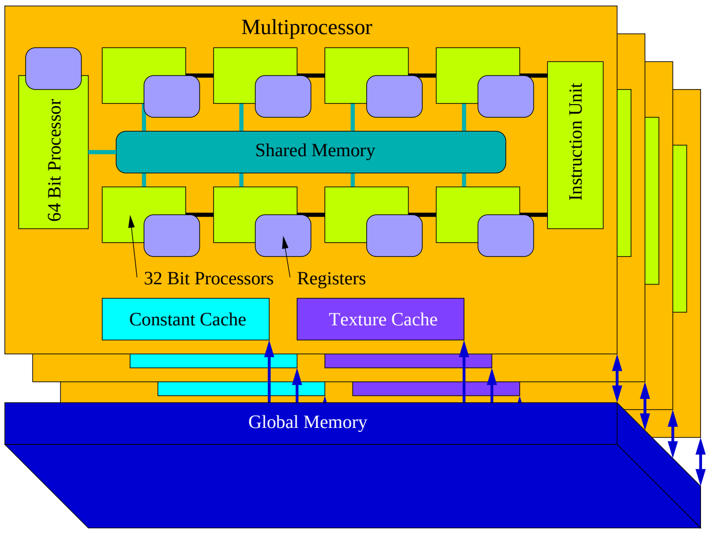


Figure 7.6 Main components of NVIDIA GeForce 200 series GPUs.

7.2 Numerical Quadrature on the GPU

The 3rd and 4th levels of parallelism are expressed entirely on the GPU. The programming of the GPU is closely aligned with the architecture of the hardware. A description is provided here to illustrate how an algorithm executes on a GPU and is based upon information given in the CUDA programming guide [42].

7.2.1 GPU Architecture

Figure. 7.6 summarizes the main components of NVIDIA GeForce 200 series GPUs. The main feature is a set of *multiprocessors*, the number of which primarily determines the performance (and cost) of the GPU. In the example of our test node, the GPU contains 27 multiprocessors. Each multiprocessor, features a single-instruction, multiple-data (SIMD) architecture with 8 processors that each execute a single instruction on different data. The SIMD processors are all 32-bit and only operate on single-precision data. For double-precision computations, a single 64-bit processor exists on the multiprocessor. In terms of computational performance, double-precision operations are significantly handicapped; the theoretical double-precision performance is actually twelve times slower than that of the theoretical single-precision performance.

The memory system consists of on-chip memory and global memory, the latter being similar to RAM for a CPU. The individual multiprocessors have four types of on-chip memory: registers, shared memory, a read-only constant cache, and a read-only texture cache. The read-write global memory for the GPU can be read from or written to by the CPU. Additionally, the CPU can write items belonging to texture or constant memory to the GPU. Reading from constant or texture memory, versus directly from global memory, on the GPU is generally more efficient because of the cache. Although limited in size, the shared memory has very fast read and write access and it can be accessed from any processor within a multiprocessor. It is useful for temporary storage of data or sharing intermediate results within a multiprocessor. Because accessing global memory is orders of magnitude slower than accessing shared memory, the recommended programming strategy is to stage data from global memory to shared memory, perhaps using the texture or constant caches, perform as many arithmetic operations as possible on the data in shared memory, and then write the results back to global memory. Memory can be accessed in any manner but there are penalties if optimal access patterns are not respected. The details of optimal patterns are intricate [42], but a simplified example of a rule is to require data with a stride of 32 bits.

7.2.2 Algorithm for Numerical Quadrature

The programming model is closely aligned with the hardware of the GPU. GPU *blocks* are defined which are distributed to the multiprocessors of the GPU. Within each block, GPU *threads* (not to be confused with the POSIX threads running on the CPUs) are defined which execute in parallel on the processors following the SIMD pattern. The code written for the GPU, called a kernel, is programmed from the perspective of a thread. The threads in a block can be synchronized and can share data with each other via the shared memory. The blocks, on the other hand are completely independent units and may not communicate with each other. Because of this, the number of multiprocessors on a GPU can vary and have no effect, besides performance, on the execution. A grouping of blocks, labelled a GPU *grid*, defines the complete problem to be computed during a kernel execution.

The number of threads, and if possible blocks, should massively outnumber the hardware resources. For example, the threads are issued in batches of 32, called a *warp*, to the processors. However, at least 192 to 256 threads per block are recommended [42]. In addition, the device attempts to run as many blocks as possible on a multiprocessor. The actual number is limited by hardware resources, usually either the number of available registers or the amount of available shared memory. The overall concept is to have as many active *warps* as possible. "The ratio of the number of active warps per multiprocessor to the maximum number of active warps (limited by hardware) is called the multiprocessor *occupancy*" [42]. With a large enough occupancy, the GPU is able to hide the memory access latencies of some warps behind the computation of other warps; in doing so, it can avoid requirements for large local memory caches.

In our example of high-order $\mathcal{RD}$, the quadrature of the linearized state and fluctuation is adapted to this GPU programming model. The grid consists of all the zones that must be evaluated during one kernel execution (the number of zones evaluated per kernel execution is equivalent to the number of available CPU cores in the compute node). Each block contains two reconstruction elements and 144 threads are used per reconstruction element to perform the integration. Dummy reconstruction elements are added if needed to fully complete all GPU blocks. In total, 288 threads are assigned per block which is evenly divisible into 9 warps per block. As stated in Section 4.5, four Gauss points are required in each primary element for integration of the linearized state and eight Gauss points are required on each edge for integration of the flux. During integration of the linearized state, 16 threads are used in each element to simultaneously evaluate all Gauss points for all variables:

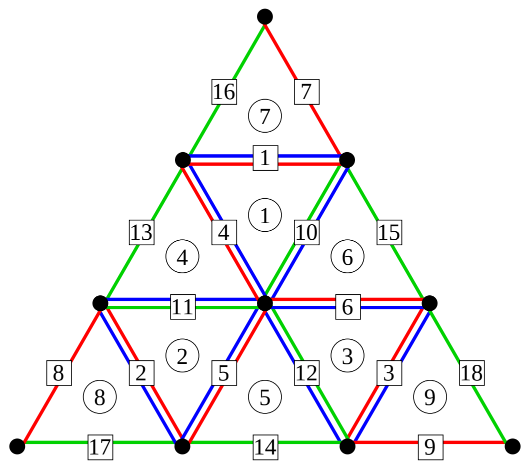
$$4 \text{ Gauss points } \times \text{ 4 variables } \times \text{ 9 primary elements } \times \text{ 2 reconstruction elements} = 288\,.$$

During integration of the flux, 8 threads simultaneously interpolate the solution at all Gauss points on the edge for a single variable, and store the result in a register. This is repeated four times for each variable in the linearized state. The flux is then evaluated at each Gauss point and summed to numerically solve the integral. This is also repeated four times to assemble the flux for each equation in the two-dimensional Euler system of equations:

$$8 \text{ Gauss points } \times \text{ 1 variable } \times \text{ 18 edges } \times \text{ 2 reconstruction elements} = 288\,.$$

The fluctuation in a primary element is completed by a summation of the flux on the surrounding edges. The particular ordering of the edges and elements shown in Fig. 7.7 simplifies the reduction. Assuming sequential ordering of the elements and edges in memory, the red and green edges can be read from consecutive memory locations and written to consecutive memory locations. Only addition of the blue edges results in a disordered access pattern.

Results for the Euler equations are presented in this chapter but a kernel has been written for high-order solutions of the scalar advection equations as well. In the advection case, each block contains eight reconstruction elements and 36 threads are used per reconstruction element to perform the integration. Four threads are as-



Element	1	2	3	4	5	6	7	8	9
🟥 Edge 🟥	1	2	3	4	5	6	7	8	9
🟦 Edge 🟦	4	5	6	11	12	10	1	2	3
🟩 Edge 🟩	10	11	12	13	14	15	16	17	18

Figure 7.7 Reduction patterns for summation of the flux on the edges into the primary-element fluctuation using the GPU.

signed to each primary element for quadrature of the linearized state and two threads are assigned to each edge for quadrature of the fluctuation. Each block then consists of 288 threads or exactly 9 warps.

The usage of the GPU memory is illustrated in Fig. 7.8. The connectivity of the reconstruction elements is loaded into global memory by the CPUs only once at the start of the solution. Before each invocation of the kernel, the CPU threads load the current solution for the set of zones to be integrated into global memory. As shown in Fig. 7.8 both the solution and connectivity are bound to, and read via, textures so that the data can be read using the texture cache. When the kernel executes, the connectivity for a Lagrange element is read and this information is used to load the solution at the vertices. The solution is stored in shared memory (2 reconstruction elements at a time per block or multiprocessor) for fast access throughout the kernel execution. To facilitate the quadrature, several constants are required. These include the 1-D and 2-D Gauss point locations and weights, maps for transforming normalized coordinates of the edges and primary elements into the natural coordinates of the Lagrange element, tables of vertices belonging to the edges for linear interpolation of the coordinates (used only for sub-parametric Lagrange elements), and a map for summing the fluctuation integrated along an edge into the total for a primary element (the indices of the blue edges in Fig. 7.7). Some shared memory is also allocated for reduction operations (e.g, summation of Gauss points). The computed values of the linearized state and the fluctuation are then written to global memory.

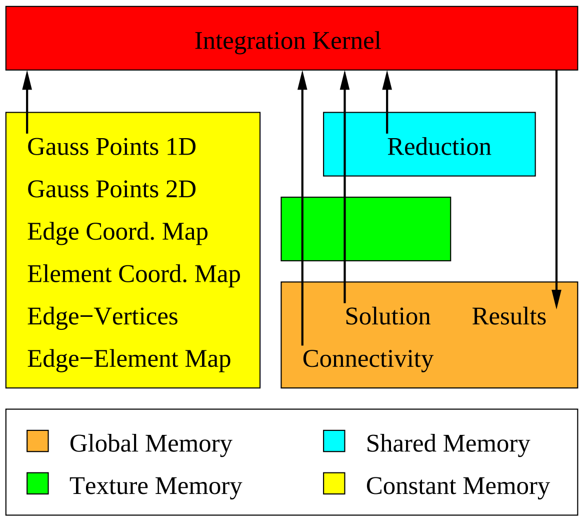


Figure 7.8 Memory types used for quadrature on the GPU.

In order to reduce the complexity of the quadrature algorithm on the GPU, approximately 33% additional work, versus quadrature on the CPU, is performed during integration of the fluctuation. On the CPU, the flux on each edge is integrated only once, a task made simpler because one does not have to worry about parallel execution or dispersed memory fetches. On the GPU, each edge in a Lagrange element (Fig. 4.3) is integrated, but the outer edges are commonly shared with another Lagrange element. Therefore, while the CPU integrates an average of 13.5 edges per Lagrange element, the GPU integrates all 18 edges per Lagrange element. Perhaps a similar algorithm to that for the CPU could be developed for the GPU, but it would be even more difficult to isolate the blocks and maintain data parallelism.

A summary of the overall relationship between the hardware and the implementation of the algorithm in software is shown in Fig. 7.9. The domain is decomposed into a number of zones and the coarsest level of parallelism is defined by a set of zones (in our numerical experiments, all the zones reside on one node). The second level of parallelism is a split in the algorithm allowing for concurrent processing on the CPU cores and the GPU. The CPU cores distribute the fluctuation and evolve the solution while the GPU concentrates on integrating the fluctuation and the linearized state. The third level of parallelism defines blocks as two reconstruction elements. The finest level of parallelism is expressed by threads on the GPU which compute the contributions from a single Gauss point.

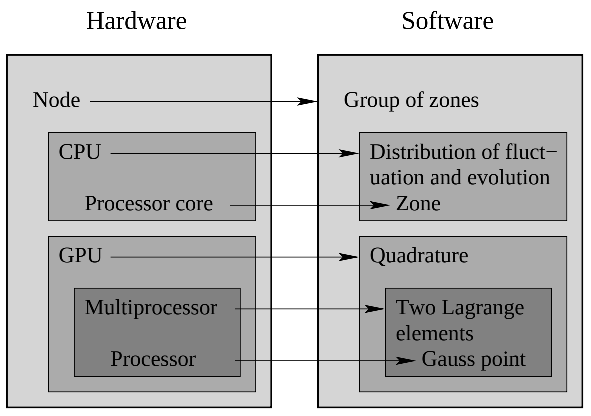


Figure 7.9 Relationship between hardware and software for the heterogeneous architecture.

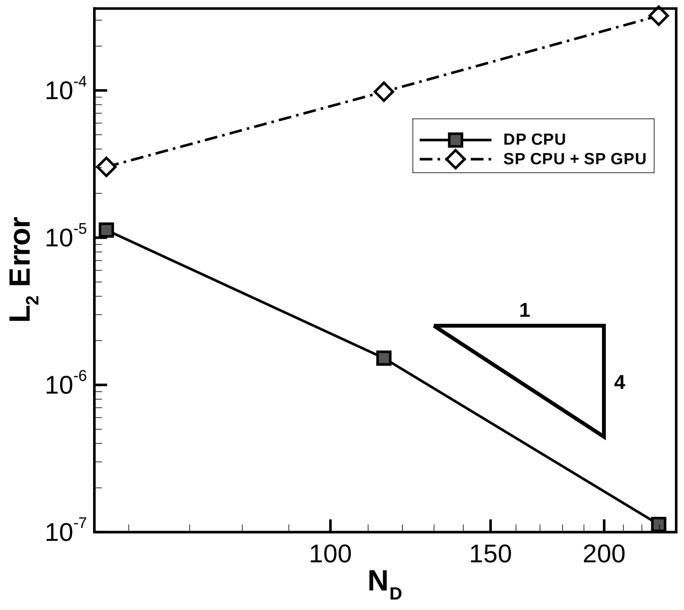


Figure 7.10 Spatial convergence of error in density for solutions of Ringleb's flow. Single-precision results are obtained on co-processing of the GPU and CPU while double-precision results are obtained on only the CPU.

7.2.3 Floating-Point Precision

Our expectation was that single-precision would be sufficient to solve the Euler equations. However, the single-precision results for Ringleb's flow in Fig. 7.10 suggest otherwise. In single-precision, as the number of grid points increases, the error also increases. The reason for this behaviour is subtractive cancellation, an error related to the finite precision of floating point numbers, during computation of the fluctuation. The flux on the edges is well defined, possibly using all of the available precision. However, summation of the flux on the three edges will often produce a fluctuation near zero. The fluctuation is known to frequently approach zero because only in this limit does the N scheme become unbounded. An example of subtractive cancellation is given below for six digits of floating-point precision:

$$\begin{array}{r} 1.00002\times 10^{5} \\ -1.00000\times 10^{5} \\ \hline 2.\mathrm{xxxxx}\times 10^{0} \end{array}.$$

The subtraction here has eliminated 5 digits of precision leaving nothing but round-off error in the positions marked by ‘x’. A similar effect is corrupting the computed fluctuation and, unless the entire method can be radically reformulated, the only solution is to increase the precision. Another problem is simply the lack of sufficient precision to represent the shape of the solution. If differences in the range of solution values are six orders of magnitude lower than the average value of the solution, then with only six digits of precision, the shape will appear flat.

These issues present a dilemma. It is possible to use the double-precision processor on the GPU and obtain results identical to those that would be obtained by only using the CPU. However, this leaves most of the arithmetic resources on the GPU idle and has a large negative impact on the efficiency of the GPU. On the other hand, single-precision computation, while extremely fast, introduces unacceptable round-off errors. To find a compromise, several changes have been made. First, the linearized state is computed entirely in single-precision. This has no effect on the accuracy; it is likely that even a linear shape function could be used in the primary elements but we have not investigated this possibility. Second, the interpolation of the solution on the edges is performed in single-precision, without much loss of accuracy, by first subtracting off the average solution value in the Lagrange element. This concept is illustrated in Fig. 7.11 for a one-dimensional Lagrange element. The original solution is given in double-precision, as represented by approximately 15 base-10 digits at the top of the figure. This is broken down into a single-precision average value and single-precision shape, each using approximately 6 base-10 digits. When the average value is subtracted from the Lagrange polynomial defining the solution, the shape of the solution then varies about an average of zero. Consequently all six digits can be used to evaluate the shape during interpolation.

In the algorithm, the average solution in a reconstruction element is computed in double-precision and subtracted from the solution at the vertices. Both the average and solution at the vertices are stored in 32-bit variables. During evaluation of a Gauss point,

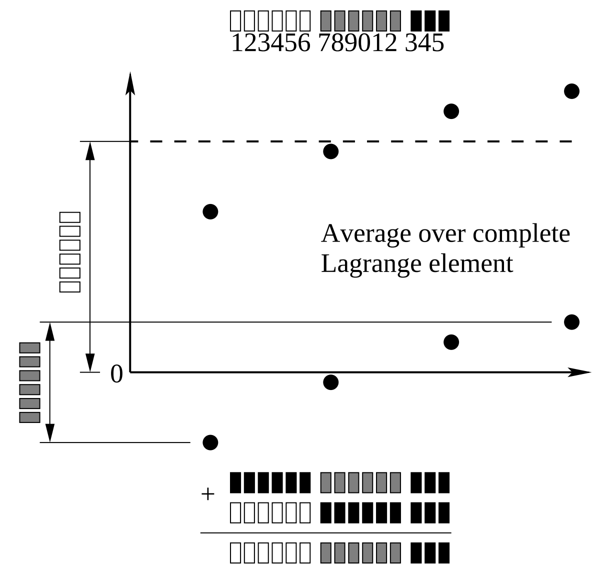


Figure 7.11 Decomposition of a double-precision solution into a single-precision average value and a single-precision shape.

the interpolation is performed completely in single-precision. The average solution and interpolated solution at the Gauss point are converted to double-precision before being summed as shown at the bottom of Fig. 7.11. The final result achieves a precision near that of the full double-precision computation.

The above method is appropriate for solution variables. For spatial coordinates, the same technique is used but, since derivatives of the coordinates are needed, an average derivative is subtracted instead.

Although introducing a fair amount of extra computation, this technique, which is referred to as mixed precision in the following sections, allows for all interpolations to be performed using single-precision. High-order interpolations require a large amount of arithmetic and by utilizing the full SIMD capability of the GPU, large gains in efficiency can be realized while still retaining an accuracy similar to that of double-precision computations.

Table 7.1 Mesh sizes for evaluation of GPU efficiency.

Mesh Number	Vertices	Primary Elements	Reconstruction Elements	Partitions
1	3220	6210	690	12
2	13111	25758	2862	12
3	52738	104533	11616	16

7.3 Computational Performance

The computational performance of a single compute node was evaluated with and without use of the GPU. In all numerical experiments without the use of the GPU, results were obtained in double-precision. When the GPU kernel was run at double-precision or using the mixed-precision technique, the code on the CPU also made use of double-precision arithmetic. When the GPU kernel was executed using single-precision arithmetic, single-precision values were used for *storage* by the algorithm executed on the CPU; however, as specified by the x86 compilers that were used, operations on the CPU were still performed using the double-precision floating-point unit with intermediate results being stored in 64-bit registers.

Three meshes with the sizes given in Table 7.1 were used to determine the spatial convergence rates for solutions of Ringleb's flow. Comparisons of execution time and synchronization charts were obtained using the intermediate mesh size. All timings are reported as an average of three runs using a wall-time encapsulating only the solver portion of the algorithm. The errors in the timings are estimated to be less than 0.5%.

The computational efficiency of the GPU was first tested for an isolated quadrature of the reconstruction elements using mesh 2 from Table 7.1. Figure 7.12 shows the speedups obtained by using double-precision, single-precision, and the mixed precision processing on the GPU versus both 1 core and all 4 cores of the CPU . Computing in double precision, the GPU offers a speedup of 3.9 times over the 4-core CPU whereas in

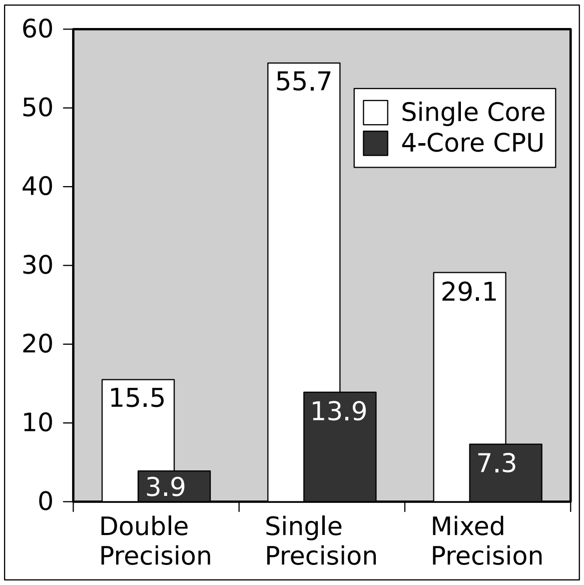


Figure 7.12 Speedup of the quadrature by the GPU.

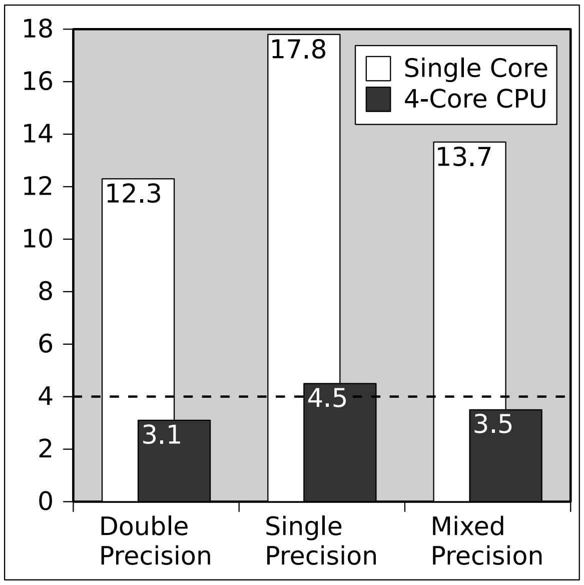


Figure 7.13 Speedup of the complete $\mathcal{RD}$ algorithm by concurrently processing quadratures on the GPU.

single-precision, the speedup is 13.9. With mixed precision, the speedup falls in-between at 7.3.

The speedups obtained for execution of the complete $\mathcal{RD}$ algorithm, again on mesh 2 from Table 7.1, are shown in Fig. 7.13. In this chart, parallel heterogeneous processing of the GPU and all four cores of the CPU are compared against one core and all four cores of only the CPU. Parallelization of the (double-precision) integration was predicted by Amdahl's law to allow a maximum speedup of 4, versus processing on only four cores of the CPU. This is shown by the dashed line in Fig. 7.13 and all speedups (versus four cores of the CPU) are near this maximum value.

The speedups for the complete algorithm are near the maximum because the processing performed by the GPU is hidden behind the concurrent processing of the CPUs, i.e., the second level of parallelism. The synchronization and processor utilization are illustrated in Figs. 7.14 to 7.16 for a single iteration (derived from an average of 500 iterations) of the algorithm at double precision, single precision, and mixed precision, respectively. Zones are numbered in these figures as they stream between the proces-

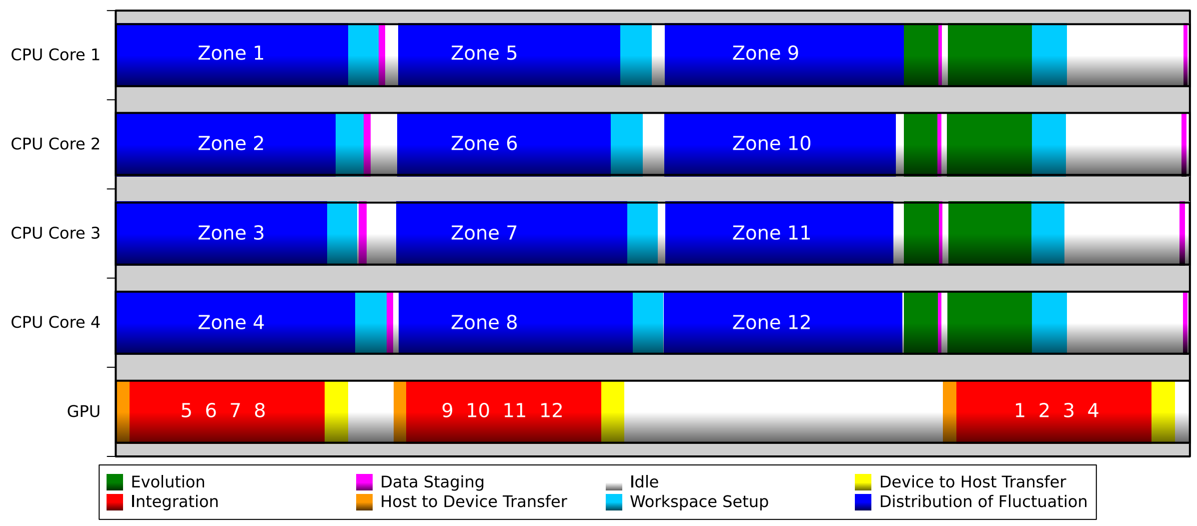


Figure 7.14 Processor synchronization and usage for one iteration of the fourth-order $\mathcal{RD}$ algorithm at double precision.

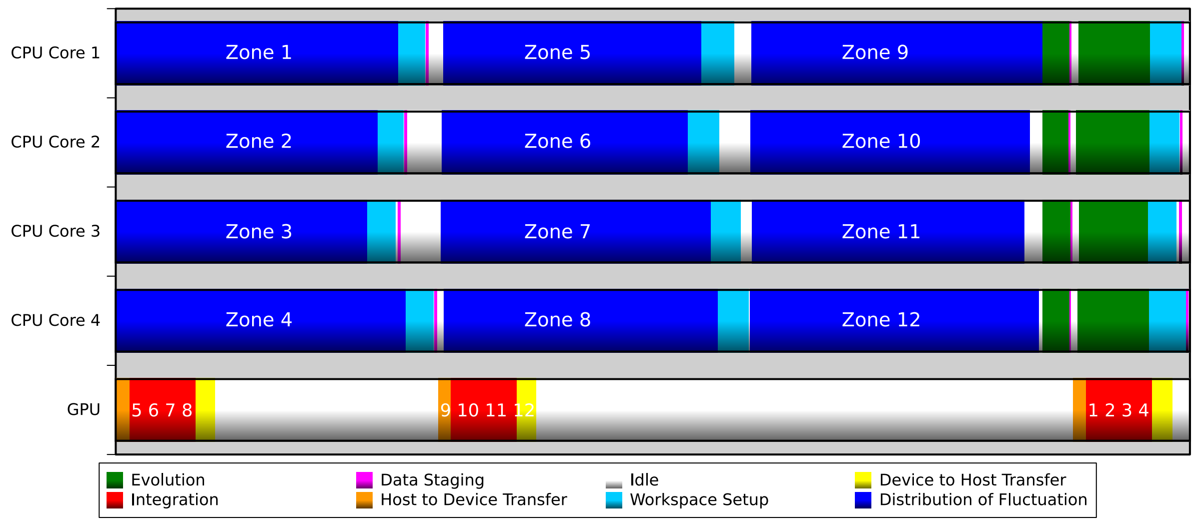


Figure 7.15 Processor synchronization and usage for one iteration of the fourth-order $\mathcal{RD}$ algorithm at single precision.

sors. Integration of linearized state and fluctuation is performed for four zones at once on the GPU (red). The results are transferred (yellow) to the CPU cores so that distribution of the fluctuation can be performed (blue). Upon completion, the CPU sets up the workspace for the next zone (light blue). This involves tasks such as computing the linearized variables and constructing weak ghost elements for the application of boundary conditions. The solution for the next set of zones is packaged (magenta) and sent

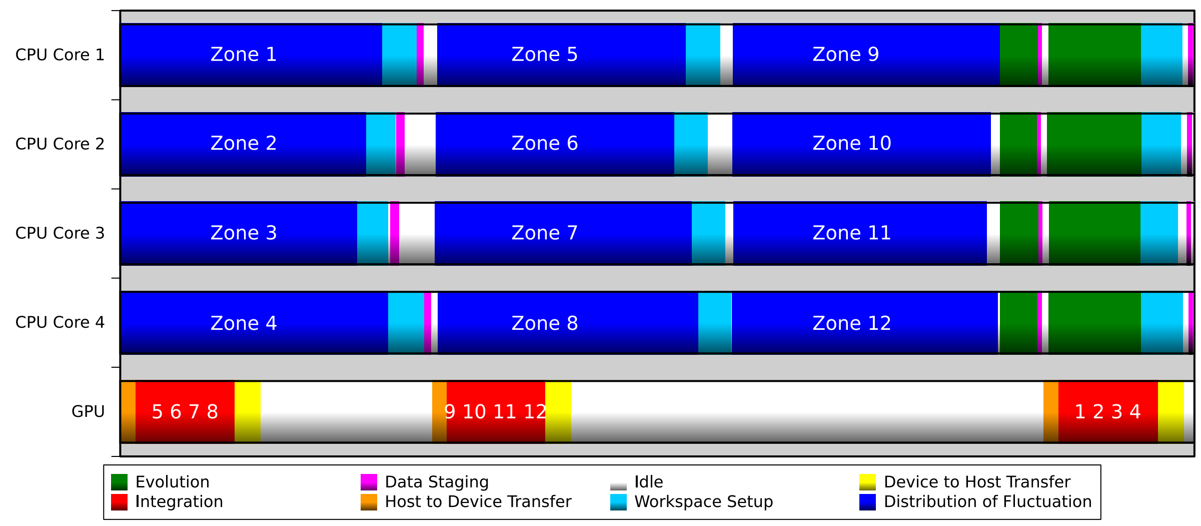


Figure 7.16 Processor synchronization and usage for one iteration of the fourth-order $\mathcal{RD}$ algorithm using mixed precision on the GPU.

to the GPU (orange). Thus, the four cores and the GPU process concurrently. Because global time stepping is used, the solution cannot be evolved until the residual has been distributed for all zones, and the global time step is known. As soon as the CPU cores evolve the first set of zones, the GPU is tasked to integrate these zones for the *next* iteration while the CPU cores continue to evolve the solution (green) for the remaining zones in the *current* iteration. The load balancing of the four CPU cores is not perfectly even because of extra computations associated with boundary conditions in some zones.

The only change resulting from the precision of the GPU is the amount of time the GPU takes to perform the integration (as described in Fig. 7.12). However, note that CPU remains idle after evolving the solution in Fig. 7.14 while it waits for data from the GPU. In Figs. 7.15 and 7.16, the CPU does not have to wait. This accounts for the different speedups seen in Fig. 7.13.

The GPU is more than 50 % idle in Fig. 7.16. There are two approaches towards achieving better utilization of the GPU. The first is to give it more work, although remaining parts of the algorithm do not adapt as well as the quadrature to the GPU architecture. The second is to simply add more CPU cores; the GPU can easily co-process with 8 CPU cores. This would have the effect of idling the CPU after evolution of the solution, as was seen in Fig. 7.14, and reducing the speedup (over the 8 CPU cores alone) to approximately three.

It is interesting to consider the capital costs of the hardware. We assume an ideal configuration with 2 quad-core CPUs, 1 GPU, and an overall speedup of 3. The GPU costs about the same as a quad-core CPU (the quad-core CPU is defined as having a unit cost). In an ideal configuration, the GPU would co-process with 8 cores (total of 3 units cost), providing a 6 times speedup over processing with only one quad-core CPU. To match the co-processing power with only CPUs requires 6 CPUs (total of 6 units cost). Therefore, equivalent computing on only CPUs would cost twice as much. Similar savings could be expected for operational (electricity) costs. These cost estimates are subject to some considerations. First, although, programmed thoughtfully, the GPU code has not been profiled or optimized based on such information. More detailed optimizations could have a large effect. Secondly, the vector capabilities of the the arithmetic units on the CPU (SSE registers) were unused in this study. With a width of 128 bits, usage of the SSE registers is probably only worthwhile for single-precision operations. Nevertheless, the same techniques for enabling single-precision computation on the GPU could also be used on the CPU. With consideration of extra data packaging efforts, we roughly estimate that use of the SSE registers could reduce the speedups in Fig. 7.12 by a factor of two.

7.3.1 Accuracy

Regarding spatial accuracy, the results from using either double-precision or mixed-precision on a GPU overlap the double-precision CPU results shown in Fig. 7.10. However, because an explicit-Euler time-marching algorithm is used with a global time-step, a relation can be made between the mesh-spacing and computation time using the CFL condition (4.1). This allows us to graph error versus computation time for the various methods. The total time, T, for a simulation can be estimated by

$$T = \underbrace{2 * \frac{L}{\Delta x}\frac{L}{\Delta x}}_{\text{No. cell}} \underbrace{\frac{H}{h}}_{\text{No. time step}} \quad , \tag{7.2}$$

where L is the length of the domain, H is the desired solution time (which prescribes the number of iterations required to converge this steady-state simulation), and h is the time step. Substituting in the CFL condition, $h = \Delta x^2/(6 * k_{max})$ results in

$$\Delta x = \left(\frac{12L^2 H k_{max}}{T}\right)^{\frac{1}{4}} \tag{7.3}$$

which, when placed in the spatial error relation (1.3), becomes[2]

$$L_p\text{-error} = \alpha T^{-\frac{\beta}{4}} \, . \tag{7.4}$$

Therefore, a scheme with fourth-order spatial accuracy should have a slope of -1 if the error is plotted versus time on a log-log scale graph. Such a graph is given in Fig. 7.17 for various architectures and floating-point precisions. With some extrapolation, it is evident from this figure that, for a given level of solution error, the fourth-order scheme is several orders of magnitude faster than the second-order scheme. The difference in computational time increases dramatically for lower levels of solution error. Usage of the GPU, on the other hand, provides a constant speedup to the fourth-order scheme for any level of solution error. More sophisticated algorithms for converging a steady state problem should also realize this constant speedup while improving upon the slope of the curve. Although seemingly of similar performance to double-precision heterogeneous computation in Fig. 7.17, remember that the GPU is underutilized in mixed-precision and could theoretically process with more CPU cores.

[2]This relationship is more applicable to unsteady problems but is used here to help quantify the performance of the high-order algorithm and the GPU.

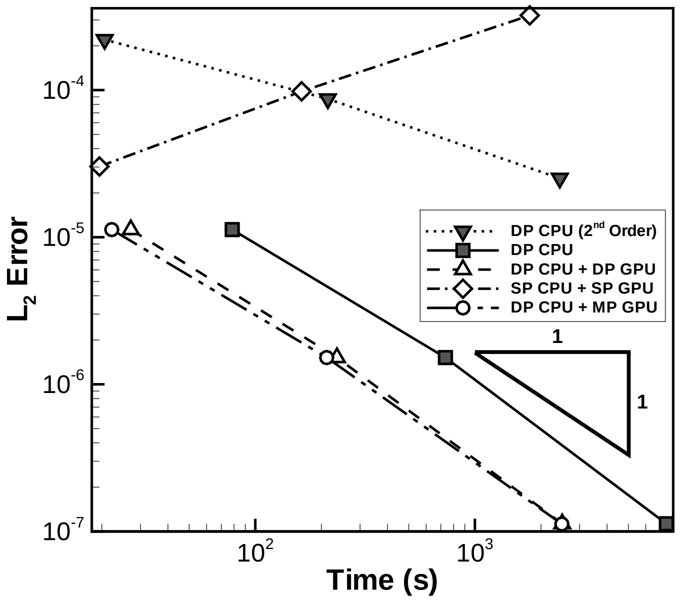


Figure 7.17 Accuracy of a scheme versus total runtime. Each solution is converged to the same solution time and data points represent different mesh sizes.

Chapter 8

Results

The set of problems examined for the comparison between $\mathcal{RD}$ and $\mathcal{FV}$ are revisited using the map C scheme at both second and fourth-order accuracy. At fourth-order accuracy, the results are obtained on the heterogeneous architecture described in chapter 7. Where accuracy is of concern, the results are compared to an LDA scheme using full double-precision on the GPU. Where monotonicity is more important, a matrix blended scheme or decomposed map A-LW (equivalent to PSI-LW) is used as a basis for comparison, again computed using full double-precision on the GPU for fourth-order results. Only accuracy is reported in this chapter; speedups do not differ significantly from that reported in chapter 7. All results are computed on unstructured grids as described in section 4.5.1. The grids are manually refined around known flow structures and are scaled by adjusting the characteristic length of an element.

8.1 Scalar Equations

The scalar problems of circular advection and Burgers equations were solved at both second and fourth-order accuracy. Although solutions of scalar equations on the heterogeneous architecture have not been thoroughly discussed, a double-precision kernel has been written and the fourth-order results shown here were obtained with assistance

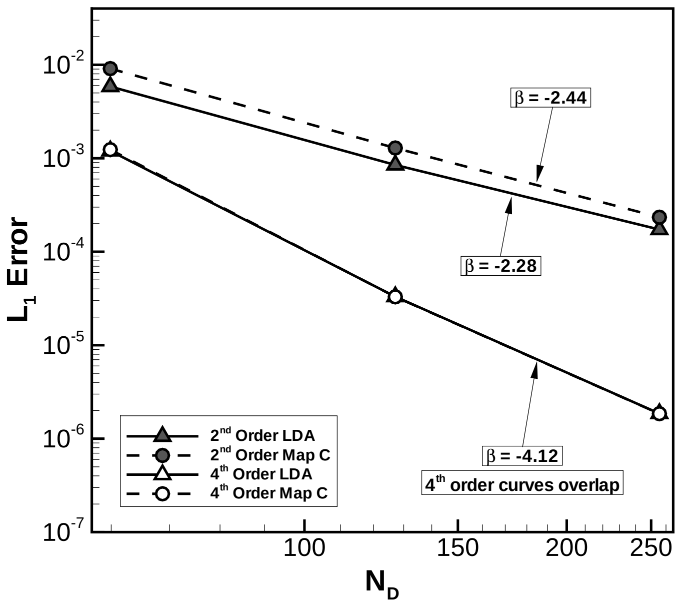


Figure 8.1 L_1-error of the map C scheme as a function of mesh density for circular advection.

from the GPU. However, a mixed precision kernel was not developed for scalar advection, hence the results presented here are equivalent to what would be obtained by using only a CPU.

Circular Advection

Second and fourth-order spatial convergence are shown for circular advection in Fig. 8.1. The map C scheme is showing the desired order of accuracy and almost exactly overlaps the LDA scheme at fourth-order accuracy.

Non-Linear Burgers Equation

The steady shock-wave problem for Burgers equation has already been used in Chapter 6 to explore the monotonicity of solutions computed with the map C scheme at second-order accuracy. Those results are supplemented here by plots of spatial convergence and results obtained at fourth-order accuracy. As a reference, solutions have also been obtained with the scalar map A scheme (equivalent to a PSI scheme or N scheme limited

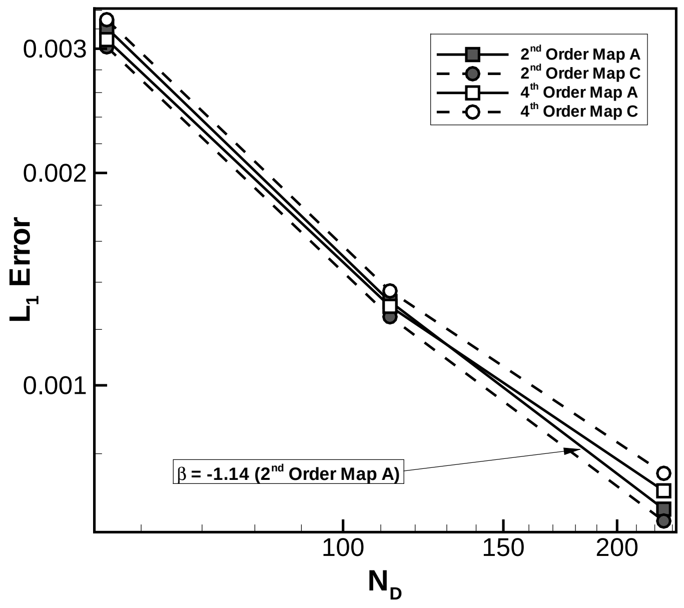


Figure 8.2 L_1-error of the map C scheme as a function of mesh density for Burgers equation.

with a minmod limiter). The spatial convergence is illustrated in Fig. 8.2 and shows all schemes achieve essentially first-order accuracy. As the mesh is refined, the fourth-order results exhibit slightly less accuracy than the second-order results. This is likely a result of oscillations being introduced by the reconstruction itself; at fourth-order, there are no restrictions on the shape of the cubic polynomials in the vicinity of discontinuities. Ultimately, it is probably necessary to enforce monotonicity over the entire reconstruction element. An approach that reduces the reconstruction to linear in the primary elements when discontinuities are detected is discussed by Guzik and Groth [29], albeit with a cumbersome approach for explicitly detecting discontinuities.

Distributions of the solution are shown in Fig. 8.3 for a section of the shock wave. At second order, both the map A and map C schemes seem to produce the same result (a clearer picture of the monotonicity, which is *not* the same, is given in chapter 6). At fourth-order accuracy, both schemes introduce some oscillations with the map C scheme being noticeably worse.

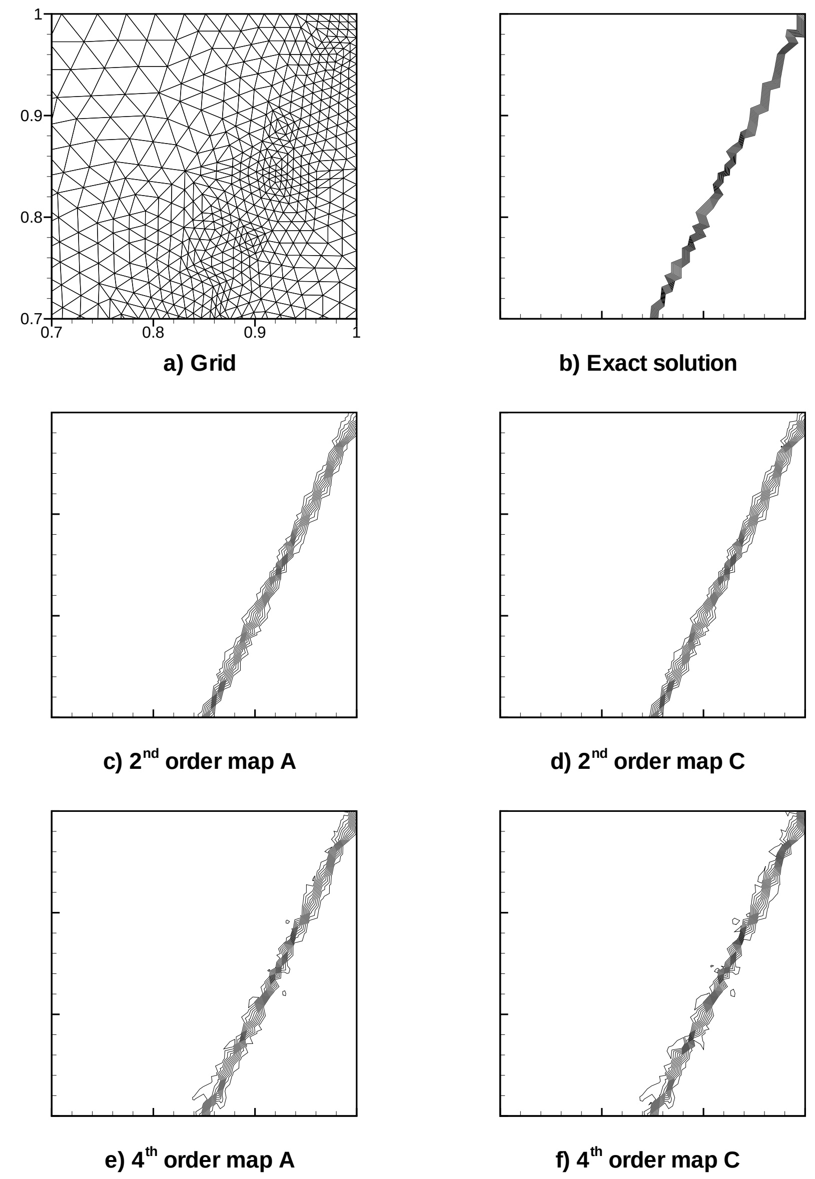


Figure 8.3 Solutions of Burgers equation obtained using the map C scheme.

8.2 Euler Equations

The canonical flow problems for the Euler equations are re-visited next. All fourth-order map C results are obtained on the heterogeneous architecture using mixed-precision computation on the GPU. Unless otherwise noted, these results are almost indistinguishable from using full double-precision on the GPU.

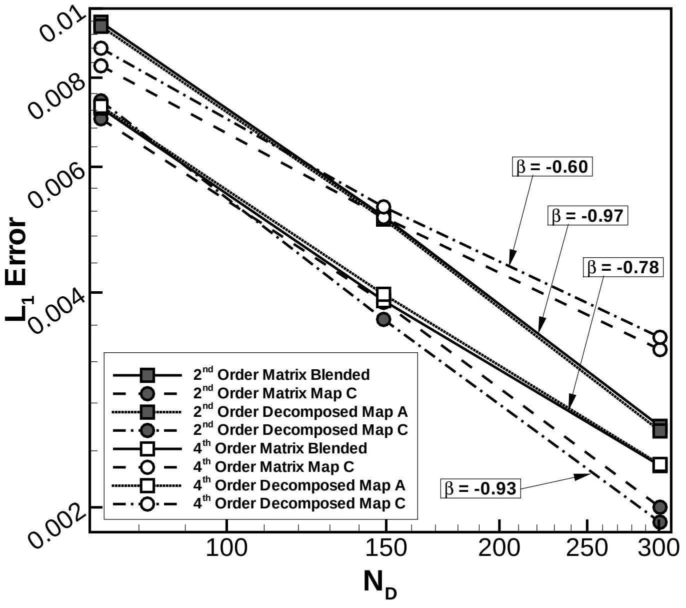
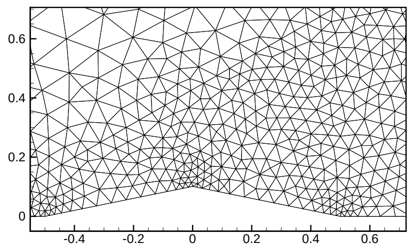


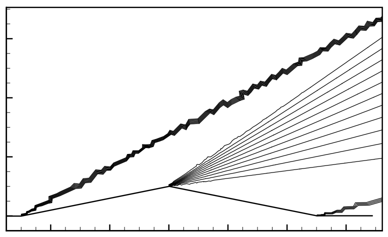
Figure 8.4 L_1-error of the map C scheme as a function of mesh density for supersonic flow past a diamond aerofoil.

Supersonic Flow Past a Diamond-Shaped Aerofoil

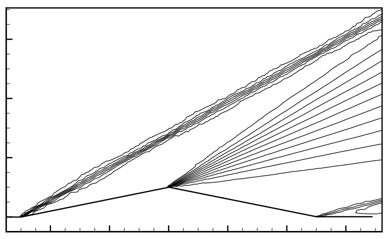
The spatial accuracy of the map C scheme for supersonic flow past a diamond-shaped aerofoil is presented in Fig. 8.4. A notable characteristic of this graph is the similarity between the matrix and hyperbolic/elliptic splitting techniques, for a given distribution scheme. Interestingly, on the finest mesh, the map C scheme is more accurate than the blended or map A scheme at second order, and less accurate at fourth-order. The order of convergence for the fourth-order schemes is less than that observed at second order, presumably for the same reasons discussed for Burgers equation: oscillations introduced by the cubic reconstruction polynomials. The contours of density shown in Fig. 8.5 indicate oscillations around the shocks at fourth-order accuracy, with the map C scheme being noticeably worse than the map A scheme.



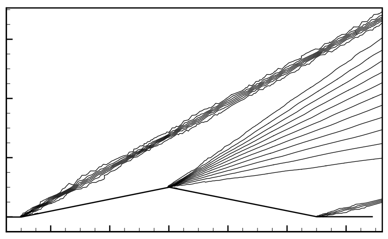
a) Grid of reconstruction elements



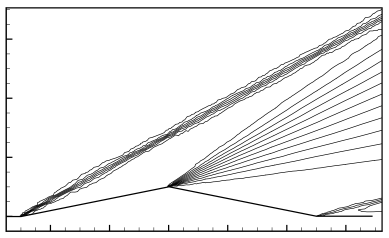
b) Exact solution



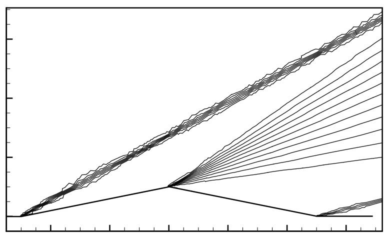
c) 2nd order matrix blended



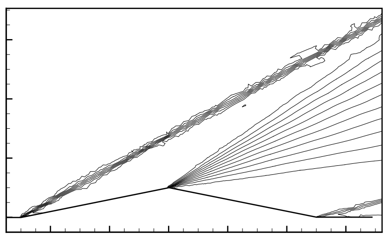
d) 2nd order matrix map C



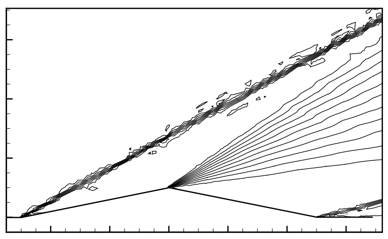
e) 2nd order decomposed map A



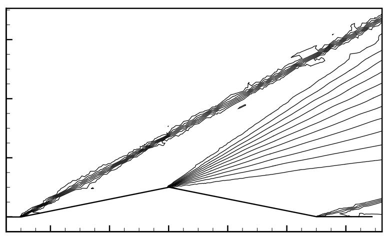
f) 2nd order decomposed map C



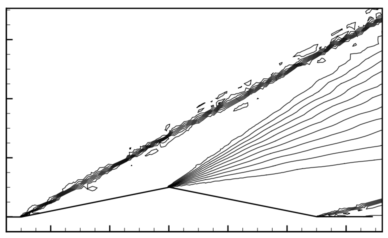
g) 4th order matrix blended



h) 4th order matrix map C



i) 4th order decomposed map A



j) 4th order decomposed map C

Figure 8.5 Density contours of supersonic flow past a diamond aerofoil using the map C scheme.

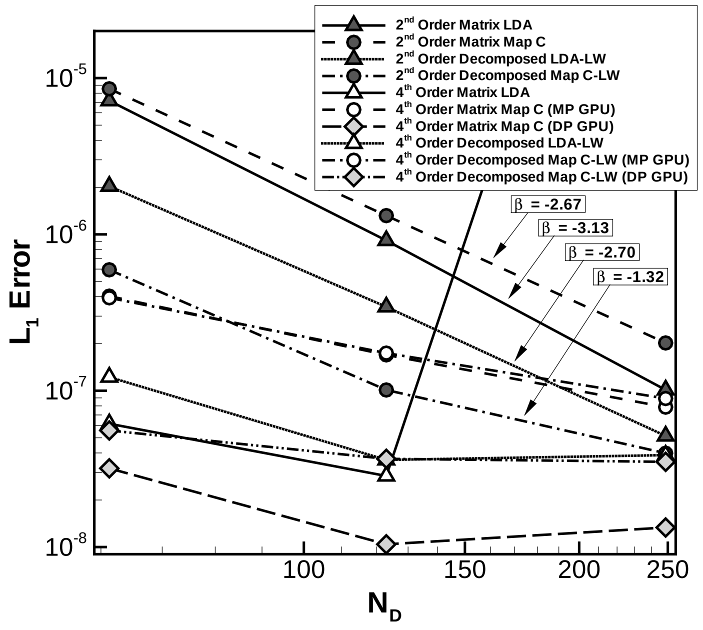


Figure 8.6 L_1-error of the map C scheme as a function of mesh density for subsonic flow past a smooth bump.

Subsonic Flow Past a Smooth Bump

Surprisingly, the relatively benign flow of a subsonic gas over a smooth bump presented some unexpected results. As shown in Fig. 8.6 convergence of the second-order map C schemes are, for the most part, on par with those of the LDA schemes. However, fourth-order spatial convergence is not apparent for the higher-order interpolations. Instead all curves seem to reach a threshold at 3.5×10^{-8} J/(kg · K). The notable exception is the fourth-order, double-precision, matrix map C scheme but for it, the threshold simply appears a bit lower. We have no conclusive explanation, but believe this behaviour to be a limit of the floating-point precision. Unlike every other problem, results obtained using mixed-precision on the GPU are significantly worse than those obtained using double-precision on the GPU. The small difference in precision, estimated at 3 base-10 digits, is having a large effect on the accuracy. Even the second-order decomposed map C-LW scheme, the only second-order scheme to reach the threshold, is unable to penetrate it. Before reaching the threshold, this scheme performs as expected reaching the appropriate order of convergence. Additionally, full convergence of the residuals could be achieved for all the considered distribution schemes. We do not believe these results to be indicative of a flaw in the scheme, only that some inherent limitation of the hardware with respect to floating-point accuracy has been reached when evaluating the solution error. In any case, results of the map C scheme, at least at full double-precision, are no worse than those of the LDA scheme.

The final peculiarity is a large error in the solution of the fourth-order matrix LDA scheme on the finest mesh. The error is obvious in the solution and appears as perturbations away from the bump near the downstream farfield boundary. For the moment, this error is not understood but may be related to the boundary implementation. Note that the map C scheme seems to overcome this instability, demonstrating enhanced robustness.

Subsonic Flow Past a Circular Cylinder

Subsonic flow past a circular cylinder also presented some challenges for the high-order schemes. Because of instabilities near the stagnation limit, hyperbolic/elliptic splitting is not used and all results are by matrix distribution techniques. In Fig. 8.7, the second-order map C scheme closely follows the second-order LDA scheme and the order of accuracy is near the expected value. At fourth-order, however, the spatial accuracy diverges for the LDA scheme. The reason for this is the development of unstable recirculation regions near the stagnation points on the circular cylinder. Contours of pressure and streamlines are shown near the aft stagnation point in Fig. 8.8. It seems that the recirculation patterns develop when there is insufficient dissipation near the stagnation point. Both the second-order LDA scheme and the fourth-order map C scheme are able to suppress these patterns. We would have expected the fourth-order map C scheme to resume at least first-order spatial convergence but the convergence seems stalled in Fig. 8.7. This is unlikely to be related to a similar spatial-convergence stall observed

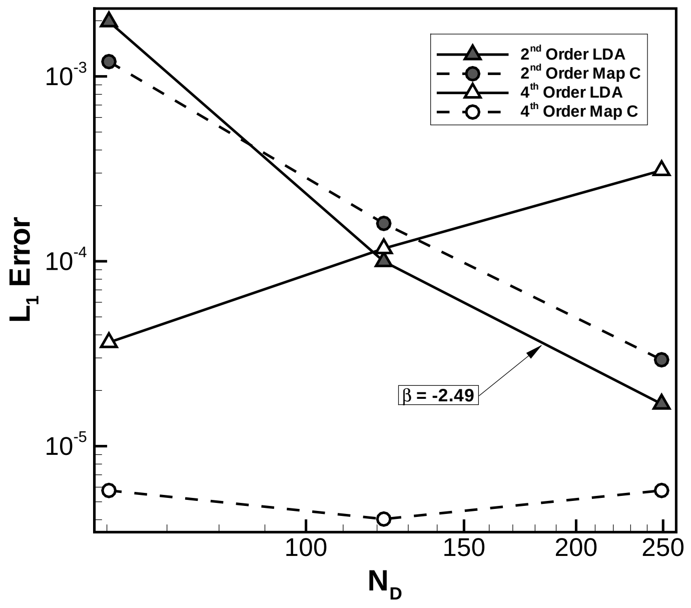


Figure 8.7 L_1-error of the map C scheme as a function of mesh density for subsonic flow past a circular cylinder.

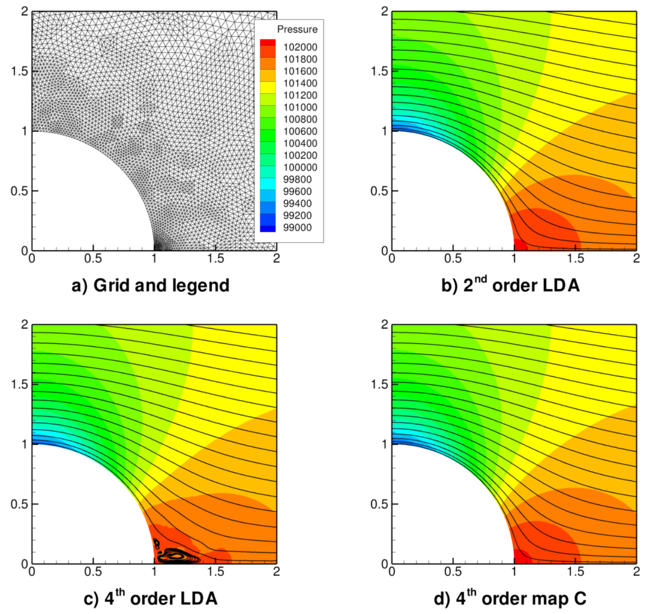


Figure 8.8 Regions of recirculation produced by high-order solutions of subsonic flow past a circular cylinder.

for subsonic flow past a bump. For the cylinder, results from using mixed precision and double-precision on the GPU are nearly identical. Also for this case, convergence of the residuals stalled for all fourth-order schemes. The residual convergence stalls very quickly

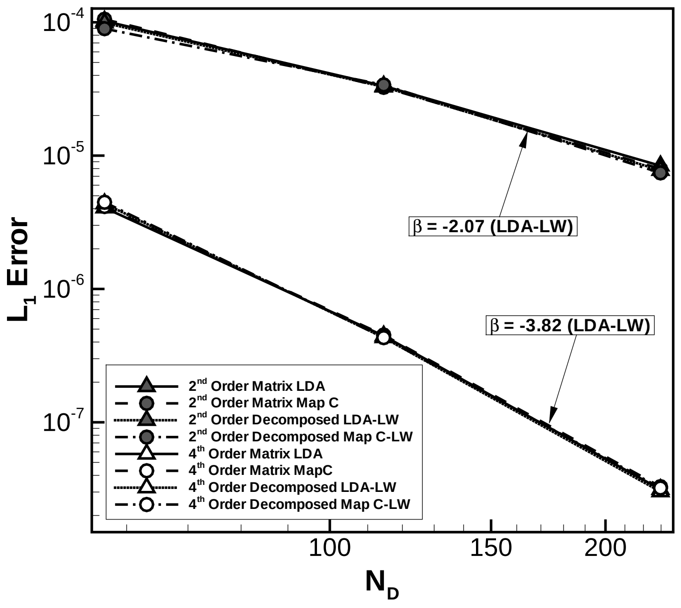


Figure 8.9 L_1-error of the map C scheme as a function of mesh density for Ringleb's flow.

for the LDA scheme and is clearly related to the recirculation zones. The residuals for the fourth-order map C scheme stall after converging about five orders of magnitude for no obvious reason. Full convergence could be achieved at second order accuracy. The exact reason for the poor spatial convergence of the fourth-order map C scheme is not understood. However, as for the bump flow, results obtained with the map C scheme are similar to, or even improve upon, results obtained with an LDA scheme.

Ringleb's Flow

Ringleb's flow (Fig. 1.4e), having an exact analytical solution, allows for direct measurement of the solution error. The map C and LDA schemes produce nearly identical results indicating that map C scheme admits the same entropy violations. We assume that the solutions for Ringleb's flow are still less accurate than what could be achieved using a $\mathcal{FV}$ scheme, but otherwise, the results are exactly as intended. As shown in Fig. 8.9 both second and fourth-order accuracy are obtained as expected for all distribution schemes.

Chapter 9

Conclusion

The $\mathcal{RD}$ method has revealed itself as a candidate for overcoming well-known deficiencies in the $\mathcal{FV}$ method, while maximizing accuracy per unit computational cost. The excellent capability of the $\mathcal{RD}$ method at capturing discontinuities is well demonstrated in the literature and has strong mathematical foundations. Herein, it was shown that the $\mathcal{RD}$ method can also surpass the accuracy of the $\mathcal{FV}$ method in smooth regions, especially when combined with techniques such as solution-dependent tessellation of the domain. At second-order, this should translate into better accuracy per computational cost since both second-order methods have similar runtimes. Unfortunately, a quantitative comparison of the $\mathcal{RD}$ method with the $\mathcal{FV}$ method indicates a number of deficiencies relating to both accuracy and computational robustness. The deficiencies are thought to result from the relative immaturity of the method and it is very likely that all may eventually be overcome.

The most significant deficiency is an observed degradation in the order of convergence of the spatial accuracy for nonlinear $\mathcal{RD}$ schemes. These schemes retain the linearity preservation and positivity properties enabling solutions that are accurate in smooth regions and monotone around discontinuities. While excelling at the latter, the degradation of accuracy in smooth regions makes the uncorrected $\mathcal{RD}$ method less viable than the $\mathcal{FV}$ method. Analysis of the problem revealed the limiting technique as the culprit and a new nonlinear mapped scheme, labelled map C, was proposed and shown to correct the deficiency. Although the map C scheme either closely reproduces or surpasses the accuracy of the best linear $\mathcal{RD}$ schemes, too much is sacrificed in terms of monotonicity near discontinuities. An interesting situation now exists with two nonlinear schemes at opposite extents of a compromise between accuracy and monotonicity. The map A scheme, already well known for some time, produces solutions with excellent monotonicity but degrades the accuracy in smooth regions. The new map C scheme achieves excellent accuracy, but permits too much (in our opinion) oscillation around discontinuities. Perhaps a nonlinear $\mathcal{RD}$ scheme can be found that excels at both or perhaps more involved techniques such as explicit discontinuity detection are required to bridge this divide.

Extension of the $\mathcal{RD}$ method to higher orders of accuracy improves the accuracy per computational cost and also allows for effective usage of highly parallel graphics processors which have recently been marketed as devices for general-purpose computing. The monetary costs for GPUs are subsidized by the graphics visualization market and therefore, they have a strong potential for increasing the cost effectiveness of computational fluid dynamics. Because an inherent compact reconstruction stencil increases the data parallelism and evaluation of the high-order interpolations increases the arithmetic intensity, parts of the $\mathcal{RD}$ algorithm are well suited to computation on a GPU. An architecture was proposed featuring four levels of parallelism including heterogeneous processing on both GPUs and CPUs. It was shown that significant speedups could be achieved by porting only simple, yet expensive, portions of the $\mathcal{RD}$ algorithm to the GPU. This has a major impact on the ease at which a GPU can be utilized. Using a single node of the proposed architecture, experiments on the efficiency of solving the Euler equations indicated that adding a GPU could improve performance by a factor between two and four over using only CPUs.

In the final results, the parallel architecture was used to obtain solutions with the new map C scheme for a set of canonical problems. Although the results are still not perfect, large steps have been made at both improving the accuracy, and computational efficiency at which solutions can be realized using the residual distribution method.

9.1 Original Contributions

A number of original contributions form an integral part of this thesis. Some of the more notable contributions are as follows:

1. On regular structured meshes, a linear scheme was proposed that uses error cancellation between elements to achieve third-order accuracy from a linear interpolation. Although other well-known schemes perform similarly, the proposed LDC scheme is the only one that features the upwind property. The development of the scheme also highlights why an LDA scheme may exhibit super-convergent properties.

2. Although probably unworkable at higher dimensions (due to exponentially increasing logistical complexities), a method was introduced for solving a decomposed system of equations on a quadrilateral mesh by optimally tessellating a quadrilateral element for each of the decomposed scalar equations according to its characteristic velocity.

3. A qualitative comparison of the $\mathcal{RD}$ and $\mathcal{FV}$ methods was performed for several canonical problems. The comparison has highlighted and documented several deficiencies in the $\mathcal{RD}$ method. The most significant, the degraded accuracy of the nonlinear schemes was seemingly unnoticed in the literature.

4. A new mapped scheme, labelled map C, was developed to correct the degraded accuracy observed in nonlinear schemes. Although the monotonicity of solutions generated by this scheme are not entirely satisfactory, it is the only nonlinear mapped scheme known by the author to obtain similar or better accuracy than the LDA scheme for all problems considered.

5. The methodologies outlined herein for adapting a CFD algorithm to a heterogeneous architecture consisting of a GPU and CPU are quite novel. This includes finding techniques that compromise between the efficiency of single-precision arithmetic and the accuracy of double-precision arithmetic.

6. The fourth-order results presented in this thesis are thought to be some of the first obtained for the Euler system of equations with a nonlinear distribution scheme using P^3 elements as proposed by Abgrall and Roe [9].

7. A technique was outlined for using a set of interfaces, POSIX threads, and RMA operations defined by MPI to enable communication both within the shared memory of a node and between the distributed memory of a cluster of nodes. The technique is logistically simple, allows for overlap of communication and computation, and somewhat distributes the communication messages throughout an iteration instead of sending them all at once.

9.2 Future Research

We still feel that the residual method is somewhat immature but this leaves much opportunity for future research. Having two nonlinear schemes, one which excels at achieving monotone solutions and one which excels at reducing error in smooth regions, the most obvious avenue for future work is to find a nonlinear scheme that excels at both. As mentioned earlier, this may require more involved, if less elegant, approaches such as explicit discontinuity detection. It would also be desirable to have a limiter that does not adversely affect convergence of the residuals and is not dependent upon freezing techniques.

Some robustness issues have also been identified, both at second and fourth-order accuracy, which require further research. Unless solutions are obtained by minimization, hyperbolic/elliptic splitting suffers from instability near stagnation regions. Additionally, it was shown that $\mathcal{RD}$ schemes can have higher solution error than $\mathcal{FV}$ schemes in flows where entropy violations may develop. However, the errors are not excessive and it is perhaps incorrect to believe that $\mathcal{RD}$ should be better than $\mathcal{FV}$ in all cases. Several robustness issues were also observed for the problem of subsonic flow past a circular cylinder. These include minor perturbations at second-order and recirculation patterns and stalling of the residual convergence at fourth-order. Problems related to the circular cylinder may be resolved by a more mature nonlinear scheme as described in the previous paragraph.

The method for imposing boundary conditions has proven effective for the problems considered herein. However, it can be problematic in other flows, such as strong expansion near a wall, and a more rigorous study on methods for applying boundary conditions is certainly required. At interior boundaries, the simplicity of the interface should allow for an efficient parallel implementation of solution-adaptive mesh refinement.

Finally, practical CFD with $\mathcal{RD}$ will not be realized until steady and unsteady viscous solutions are possible in three spatial dimensions. Many components, such as unsteady and viscous flows have already been examined in two-spatial dimensions. Unsteady approaches ensure consistent mass matrices [45] or construct either simplex [18] or prismatic [48] elements in space-time. For viscous flows, uniform accuracy of the advective and diffusive components can be achieved by casting the problem as a first-order system [39] or following a Petrov-Galerkin analogy [21]. While the hyperbolic/elliptic splitting approach still requires more research for extension to three-dimension, there are no such limitations for matrix schemes. As such, the road to practical CFD may not involve much more than the assembly of the constituent parts. An extensive discussion of the future research requirements for $\mathcal{RD}$ in general is given by Deconinck and Ricchiuto [21].

References

[1] Abgrall, R., "Toward the Ultimate Conservative Scheme: Following the Quest," *Journal of Computational Physics*, 167, 2001, pp. 277–315.

[2] Abgrall, R., "Construction of Second Order Accurate Monotone and Stable Residual Distribution Schemes for Unsteady Flow Problems," *Journal of Computational Physics*, 168, 2003, pp. 16–55.

[3] Abgrall, R., "Essentially Non-Oscillatory Residual Distribution Schemes for Hyperbolic Problems," *Journal of Computational Physics*, 214, 2006, pp. 773–808.

[4] Abgrall, R., "Residual Distribution Schemes: Current Status and Future Trends," *Computers & Fluids*, 35, 2006, pp. 641–669.

[5] Abgrall, R. and Barth, T., "Residual Distribution Schemes for Conservation Laws Via Adaptive Quadrature," *SIAM Journal of Scientific Computing*, 24(3), 2002, pp. 732–769.

[6] Abgrall, R., Larat, A., Ricchiuto, M., and Tavé, C., "A Simple Construction of Very High Order Non-Oscillatory Compact Schemes on Unstructured Meshes," *Computers & Fluids*, 38, 2009, pp. 1314–1323.

[7] Abgrall, R. and Marpeau, F., "Residual Distribution Schemes on Quadrilateral Meshes," *Journal of Scientific Computing*, 30(1), 2007, pp. 131–175.

[8] Abgrall, R. and Mezine, M., "Construction of Second-Order Accurate Monotone and Stable Residual Distribution Schemes for Steady Problems," *Journal of Computational Physics*, 195, 2004, pp. 474–507.

[9] Abgrall, R. and Roe, P. L., "High Order Fluctuation Schemes on Triangular Meshes," *Journal of Scientific Computing*, 19(1-3), 2003, pp. 3–36.

[10] Barth, T. J. and Jespersen, D., "The Design and Application of Upwind Schemes on Unstructured Meshes," AIAA 89-0366, 1989.

[11] Berger, M., Aftosmis, M. J., and Murman, S. M., "Analysis of Slope Limiters on Irregular Grids," AIAA 2005-0490, 43rd AIAA Aerospace Sciences Meeting, 2005.

[12] Brandvik, T. and Pullan, G., "Acceleration of a 3D Euler Solver Using Commodity Graphics Hardware," AIAA 2008-607, 46th AIAA Aerospace Sciences Meeting, 2008.

[13] Caraeni, D. and Fuchs, L., "Compact Third-Order Multidimensional Upwind Discretization for Steady and Unsteady Flow Simulations," *Computers & Fluids*, 34, 2005, pp. 419–441.

[14] CGNS, *http://cgns.sourceforge.net/* and/or *http://www.grc.nasa.gov/WWW/cgns/index.html*, CGNS Steering Committee, 2009.

[15] Chapman, B., Jost, G., and Pas, R. V. D., *Using OpenMP*, The MIT Press, 2008.

[16] Chou, C.-S. and Shu, C.-W., "High Order Residual Distribution Conservative Finite Difference WENO Schemes for Convection-Diffusion Steady State Problems on Non-Smooth Meshes," *Journal of Computational Physics*, 224, 2007, pp. 992–1020.

[17] Cohen, J. M. and Molemaker, M. J., "A Fast Double Precision CFD Code using CUDA," *Proceedings of the 21st International Conference on Parallel Computational Fluid Dynamics*, ParCFD, 2009.

[18] Csík, A. and Deconinck, H., "Space-Time Residual Distribution Schemes for Hyperbolic Conservation Laws on Unstructured Linear Finite Elements," *Int. J. Numer. Meth. Fluids*, 40, 2002, pp. 573–581.

[19] Csík, A., Ricchiuto, M., and Deconinck, H., "A Conservative Formulation of the Multidimensional Upwind Residual Distribution Schemes for General Nonlinear Conservation Laws," *Journal of Computational Physics*, 179, 2002, pp. 286–312.

[20] Deconinck, H., Powell, K. G., Roe, P. L., and Struijs, R., "Multi-Dimensional Schemes for Scalar Advection," AIAA 91-1532, AIAA CFD Conference, 1991.

[21] Deconinck, H. and Ricchiuto, M., *Encyclopedia of Computational Mechanics*, John Wiley & Sons, Ltd., vol. 3, chap. Residual Distribution Schemes: Foundations and Analysis, 2007.

[22] Deconinck, H., Sermeus, K., and Abgrall, R., "Status of Multidimensional Upwind Residual Distribution Schemes and Applications in Aeronautics," AIAA 2000-2328, AIAA Fluids Conference, 2000.

[23] Emery, A. F., "An Evaluation of Several Differencing Methods for Inviscid Fluid Flow Problems," *Journal of Computational Physics*, 2, 1968, pp. 306–331.

[24] Geuzaine, C. and Remacle, J.-F., "Gmsh: A Three-Dimensional Finite Element Mesh Generator with Built-in Pre- and Post-Processing Facilities," *International Journal for Numerical Methods in Engineering*, 79(11), 2009, pp. 1309–1331.

[25] Giles, M., Anderson, W. K., and Roberts, T. W., "Upwind Control Volumes: A New Upwind Approach," AIAA 90-0104, 1990.

[26] Godunov, S. K., "Finite-Difference Method for Numerical Computations of Discontinuous Solutions of the Equations of Fluid Dynamics," *Mat. Sb.*, 47, 1959, pp. 271–306.

[27] Gottlieb, J. J. and Groth, C. P. T., "Assessment of Riemann Solvers for Unsteady One-Dimensional Inviscid Flows of Perfect Gases," *Journal of Computational Physics*, 78, 1988, pp. 437–458.

[28] Guzik, S. M. J. and Groth, C. P. T., "Comparison of Solution Accuracy of Multidimensional Residual Distribution and Godunov-Type Finite-Volume Methods," *International Journal of Computational Fluid Dynamics*, 22(1–2), 2008, pp. 61–83.

[29] Guzik, S. M. J. and Groth, C. P. T., "A High-Order Residual-Distribution Scheme with Variation-Bounded Nonlinear Stabilization," AIAA 2008-0777, 46th AIAA Aerospace Sciences Meeting, 2008.

[30] Hendrickson, B. and Leland, R., "The Chaco User's Guide: Version 2.0," SAND 94–2692, Sandia, 1994.

[31] Hubbard, M. E., "Non-Oscillatory Third Order Fluctuation Splitting Schemes for Steady Scalar Conservation Laws," *Journal of Computational Physic*, 222, 2007, pp. 740–768.

[32] Hubbard, M. E. and Laird, A. L., "Achieving High-Order Fluctuation Splitting Schemes by Extending the Stencil," *Computers & Fluids*, 34, 2005, pp. 443–459.

[33] Ivan, L. and Groth, C. P. T., "High-Order Central ENO Scheme for Hyperbolic Conservation Laws on Body-Fitted Multi-Block Mesh," *Proceedings of the 14th Annual Conference of the CFD Society of Canada*, CFD Society of Canada, 2006, pp. 317–324.

[34] Jameson, A. and Mavriplis, D., "Finite-Volume Solution of the Two-Dimensional Euler Equations on a Regular Triangular Mesh," *AIAA Journal*, 24, 1986, pp. 611–618.

[35] Mesaros, L. M., *Multi-Dimensional Fluctuation Splitting Schemes for the Euler Equations on Unstructured Grids*, Ph.D. thesis, University of Michigan, 1995.

[36] Munshi, A., *The OpenCL Specification*, Khronos OpenCL Working Group, version 1.0, 2009.

[37] Ni, R. H., "A Multiple Grid Scheme for Solving the Euler Equations," *AIAA Journal*, 20, 1981, pp. 1565–1571.

[38] Nishikawa, H., *On Grids and Solutions from Residual Minimization*, Ph.D. thesis, University of Michigan, 2001.

[39] Nishikawa, H., "A First-Order System Approach for Diffusion Equation. I: Second-Order Residual-Distribution Schemes," *Journal of Computational Physics*, 227, 2007, pp. 315–352.

[40] Nishikawa, H. and Roe, P. L., "Adaptive-Quadrature Fluctuation-Splitting Schemes for the Euler Equations," AIAA 2005-4865, AIAA CFD Conference, 2005.

[41] Nishikawa, H., Roe, P. L., Suzuki, Y., and van Leer, B., "A General Theory of Local Preconditioning and Its Application to the 2D Ideal MHD Equations," AIAA 2003-3704, AIAA CFD Conference, 2003.

[42] NVIDIA Corporation, *NVIDIA CUDA Compute Unified Device Architecture Programming Guide*, NVIDIA Corporation, 2008, version 2.1.

[43] Paillère, H., *Multidimensional Upwind Residual Distribution Schemes for the Euler and Navier-Stokes Equations on Unstructured Grids*, Ph.D. thesis, Université Libre de Bruxelles, 1995.

[44] Paillère, H., Degrez, G., and Deconinck, H., "Multidimensional Upwind Schemes for the Shallow Water Equations," *Int. J. Numer. Meth. Fluids*, 26, 1998, pp. 987–1000.

[45] Palma, P. D., Pascazio, G., Rossiello, G., and Napolitano, M., "A Second-Order Accurate Monotone Implicit Fluctuation Splitting Scheme for Unsteady Problems," *Journal of Computational Physics*, 208, 2005, pp. 1–33.

[46] Rad, M., *A Residual Distribution Approach to the Euler Equations that Preserves Potential Flow*, Ph.D. thesis, University of Michigan, 2001.

[47] Ricchiuto, M., *Construction and Analysis of Compact Residual Discretizations for Conservation Laws on Unstructured Meshes*, Ph.D. thesis, von Karman Institute for Fluid Dynamics, 2005.

[48] Ricchiuto, M., Csík, A., and Deconinck, H., "Residual Distribution for General Time-Dependent Conservation Laws," *Journal of Computational Physics*, 209, 2005, pp. 249–289.

[49] Roe, P. L., "Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes," *Journal of Computational Physics*, 43, 1981, pp. 357–372.

[50] Roe, P. L., "Linear advection schemes on triangular meshes," CoA Report 8720, 1987.

[51] Roe, P. L. and Sidilkover, D., "Optimum Positive Linear Schemes for Advection in Two and Three Dimensions," *SIAM Journal of Numerical Analysis*, 29(6), 1992, pp. 1542–1568.

[52] Sachdev, J. S., Groth, C. P. T., and Gottlieb, J. J., "A Parallel Solution-Adaptive Scheme for Predicting Multi-Phase Core Flows in Solid Propellant Rocket Motors," *International Journal of Computational Fluid Dynamics*, 19(2), 2005, pp. 157–175.

[53] Sermeus, K. and Deconinck, H., "An Entropy Fix for Multi-Dimensional Upwind Residual Distribution Schemes," *Computers & Fluids*, 34, 2005, pp. 617–640.

[54] Sidilkover, D. and Roe, P. L., "Unification of some Advection Schemes in Two Dimensions," Technical Report 95-10, ICASE, 1995.

[55] Smith, M. R., Kuo, F. A., Chou, C. Y., and Wu, J. S., "Application of a Kinetic Theory Based Solver of the Euler Equations using GPU," *Proceedings of the 21st International Conference on Parallel Computational Fluid Dynamics*, ParCFD, 2009.

[56] Stock, M. J. and Gharakhani, A., "Toward Efficient GPU-Accelerated N-body Simulations," AIAA 2008-608, 46th AIAA Aerospace Sciences Meeting, 2008.

[57] Struijs, R., *A Multi-Dimensional Upwind Discretization Method for the Euler Equations on Unstructured Grids*, Ph.D. thesis, Technische Universiteit Delft, 1994.

[58] van der Weide, E. and Deconinck, H., "Positive Matrix Distribution Schemes for Hyperbolic Systems, with Application to the Euler Equations," *Computational Fluid Dynamics '96, Proceedings of the 3rd ECCOMAS CFD Conference*, J. Wiley, 1996, pp. 496–502.

[59] van der Weide, E., Deconinck, H., Issman, E., and Degrez, G., "A Parallel, Implicit, Multi-Dimensional Upwind, Residual Distribution Method for the Navier-Stokes Equations on Unstructured Grids," *Computational Mechanics*, 23, 1999, pp. 199–208.

[60] van Leer, B., "Towards the Ultimate Conservative Difference Scheme. V. A Second-Order Sequel to Godunov's Method," *Journal of Computational Physics*, 32, 1979, pp. 101–136.

[61] Venkatakrishnan, V., "On the Accuracy of Limiters and Convergence to Steady State Solutions," Paper 93-0880, AIAA, 1993.

[62] Waterson, N. P. and Deconinck, H., "Design Principles for Bounded Higher-Order Convection Schemes — a Unified Approach," *Journal of Computational Physics*, 224, 2007, pp. 182–207.

[63] Wood, W. A., *Multi-dimensional Upwind Fluctuation Splitting Scheme with Mesh Adaption for Hypersonic Viscous Flow*, Ph.D. thesis, Virginia Polytechnic Institute and State University, 2001.

[64] Wood, W. A. and Kleb, W. L., "Diffusion Characteristic of Finite Volume and Fluctuation Splitting Schemes," *Journal of Computational Physics*, 153, 1999, pp. 353–377.

[65] Wood, W. A. and Kleb, W. L., "2-D/Axisymmetric Formulation of Multi-dimensional Upwind Scheme," AIAA 2001-2630, 15th AIAA CFD Conference, 2001.

[66] Woodward, P. and Colella, P., "The Numerical Simulation of Two-Dimensional Fluid Flow with Strong Shocks," *Journal of Computational Physics*, 54, 1984, pp. 115–173.

[67] Zienkiewicz, O. C., *Finite Element Method in Engineering Science*, McGraw-Hill Education, 2nd ed., 1972.

Appendix A

Truncation Error of the LN Scheme

The Taylor series expansion of the LN scheme is performed on the three triangles shown in Fig. 2.8. The fluctuation is defined by (2.24) with the same inflow parameters. All nodal values are expressed relative to u_i using the Taylor series defined by (2.25) but only to $O(\Delta h^3)$. By switching to a streamline coordinate system and assuming derivatives in the streamline direction are zero, the expansions become

$$\begin{aligned}
u_j &= u_i + \frac{1}{|\lambda|}\left[b\Delta x\frac{\partial u}{\partial \eta} + \frac{b^2\Delta x^2}{2|\lambda|}\frac{\partial^2 u}{\partial \eta^2}\right] + O(\Delta h^3)\\
u_k &= u_i + \frac{1}{|\lambda|}\left[(b\Delta x - a\Delta y)\frac{\partial u}{\partial \eta} + \frac{(b\Delta x - a\Delta y)^2}{2|\lambda|}\frac{\partial^2 u}{\partial \eta^2}\right] + O(\Delta h^3)\\
u_l &= u_i + \frac{1}{|\lambda|}\left[-a\Delta y\frac{\partial u}{\partial \eta} + \frac{a^2\Delta y^2}{2|\lambda|}\frac{\partial^2 u}{\partial \eta^2}\right] + O(\Delta h^3)\\
u_m &= u_i + \frac{1}{|\lambda|}\left[-b\Delta x\frac{\partial u}{\partial \eta} + \frac{b^2\Delta x^2}{2|\lambda|}\frac{\partial^2 u}{\partial \eta^2}\right] + O(\Delta h^3)\,.
\end{aligned} \tag{A.1}$$

For element E_1, the fluctuation sent to vertices i and j by the N scheme are $\phi_i^N = k_a(u_i - u_k)$ and $\phi_j^N = k_b(u_j - u_k)$, respectively. Substituting into the limiting formula defined by (2.48), $\phi_i^{LN} = \phi_i^N + \Psi_1(r_1)\phi_j^N$ with $r_1 = -\phi_i^N/\phi_j^N$, results in

$$\begin{aligned}
\phi_i^{LN,E_1} = \frac{a\Delta y(b\Delta x - a\Delta y)}{2|\lambda|}\Bigg[&-\frac{\partial u}{\partial \eta} - \frac{b\Delta x - a\Delta y}{2|\lambda|}\frac{\partial^2 u}{\partial \eta^2}\\
&+\Psi_1(r_1)\left(\frac{\partial u}{\partial \eta} + \frac{2b\Delta x - a\Delta y}{2|\lambda|}\frac{\partial^2 u}{\partial \eta^2}\right)\Bigg] + O(\Delta h^4)\,.
\end{aligned} \tag{A.2}$$

From element E_2, the entire fluctuation, $\phi^{E_2} = -k_c u_i - k_a u_k - k_b u_l$, is sent by the LN scheme to vertex i,

$$\phi_i^{LN,E_2} = -\frac{ab^2\Delta x^2 \Delta y}{4|\lambda|^2}\frac{\partial^2 u}{\partial \eta^2} + O(\Delta h^4)\,. \tag{A.3}$$

For element E_3, the fluctuation sent to vertices i and m by the N scheme are $\phi_i^N = k_b(u_i - u_l)$ and $\phi_m^N = k_a(u_m - u_l)$, respectively. Substituting into the limiting formula defined by (2.48), $\phi_i^{LN} = \phi_i^N - \Psi_3(r_3)\phi_i^N$ with $r_3 = -\phi_m^N/\phi_i^N$, results in

$$\phi_i^{LN,E_3} = \frac{a\Delta y}{2|\lambda|}\left[1 - \Psi_3(r_3)\right]\left(b\Delta x - a\Delta y\right)\left(\frac{\partial u}{\partial \eta} - \frac{a\Delta y}{2|\lambda|}\frac{\partial^2 u}{\partial \eta^2}\right) + O(\Delta h^4)\,. \tag{A.4}$$

With the substitution $\Delta y = s\Delta x$, the total fluctuation sent to vertex i from all upwind elements is

$$\phi_i^{LN} = \frac{as(as-b)\Delta x^2}{2|\lambda|}\left\{\left[\Psi_3(r_3) - \Psi_1(r_1)\right]\frac{\partial u}{\partial \eta} + \frac{\Delta x}{2|\lambda|}\left[2b + \Psi_1(r_1)\left(as - 2b\right) - \Psi_3(r_3)as\right]\frac{\partial^2 u}{\partial \eta^2}\right\} + O(\Delta h^4)\,. \tag{A.5}$$

For element 1, the input to the limiter is given by

$$r_1 = \frac{-\dfrac{\partial u}{\partial \eta} + \dfrac{(as-b)\Delta x}{2|\lambda|}\dfrac{\partial^2 u}{\partial \eta^2}}{-\dfrac{\partial u}{\partial \eta} + \dfrac{(as-2b)\Delta x}{2|\lambda|}\dfrac{\partial^2 u}{\partial \eta^2}}\,. \tag{A.6}$$

For element 3, the input to the limiter is given by

$$r_3 = \frac{\dfrac{\partial u}{\partial \eta} - (as+b)\dfrac{\partial^2 u}{\partial \eta^2}}{\dfrac{\partial u}{\partial \eta} - as\dfrac{\partial^2 u}{\partial \eta^2}}\,. \tag{A.7}$$

Assuming a smooth solution such that $r \approx 1$, the accuracy of using a MUSCL limiter, (6.1), can be determined by substituting $\Psi_1(r_1) = (r_1+1)/2$ and $\Psi_3(r_3) = (r_3+1)/2$ into (A.5). The result, $\phi_i^{LN} = O(\Delta h^4)$, indicates a second-order scheme.

In the region $r \approx 1$, the minmod limiter returns the minimum of r and 1. If either $r_1 \leq 1$ and $r_3 \leq 1$, or $r_1 \geq 1$ and $r_3 \geq 1$, then the LN scheme is also second-order accurate.

However, if $r_1 < 1$ and $r_3 > 1$, then $\Psi_1(r_1) = r_1$ and $\Psi_3(r_3) = 1$ are substituted into (A.5) yielding

$$\phi_i^{LN} = \frac{sab(as - b)\Delta x^3}{4|\lambda|^2} \frac{\partial^2 u}{\partial \eta^2} + O(\Delta h^4)\,,$$

a result that retains first-order terms. Similarly, if $r_1 > 1$ and $r_3 < 1$, then $\Psi_1(r_1) = 1$ and $\Psi_3(r_3) = r_3$ such that

$$\phi_i^{LN} = \frac{sab(b - as)\Delta x^3}{4|\lambda|^2} \frac{\partial^2 u}{\partial \eta^2} + O(\Delta h^4)\,.$$

Therefore, the LN scheme with a minmod limiter can, in certain cases, introduce first-order terms into the truncation error.