An Explicitly Conservative Method for Time-Accurate Solution of Hyperbolic Partial Differential Equations on Embedded *Chimera* Grids

by

Louis G. Stern

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

Doctor of Philosophy

University of Washington

1996

Approved by ______________________________
(Chairperson of Supervisory Committee)

____

____

Program Authorized
to Offer Degree ______________________________

Date ______________________________

In presenting this dissertation in partial fulfillment of the requirements for the Doctoral degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of this dissertation is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for copying or reproduction of this dissertation may be referred to University Microfilms, 1490 Eisenhower Place, P.O. Box 975, Ann Arbor, MI 48106, to whom the author has granted "the right to reproduce and sell (a) copies of the manuscript in microform and/or (b) printed copies of the manuscript made from microform."

Signature________________________

Date__________________________

University of Washington

Abstract

An Explicitly Conservative Method for Time-Accurate Solution of Hyperbolic Partial Differential Equations on Embedded *Chimera* Grids

by Louis G. Stern

Chairperson of Supervisory Committee: *Professor Randall J. LeVeque*
Department of Applied Mathematics

In this dissertation I use the h-box method of Berger and LeVeque, originally used on embedded Cartesian grids, to develop a conservative method for Chimera grids. *Chimera*, or overlapping-composite, grids were first introduced in 1983 to speed up the grid-generation preprocessing requirement for numerical solution of partial differential equations (PDEs) in complex geometries. Standard numerical methods for Chimerae are discretely non-conservative because data are multiply-defined where composite grids overlap. For my method I use an *embedded* Chimera grid, in which outer edges of body-fitted grids *cut into* far-field grids, rather than overlap. Data are then singly-defined, but small cut-cells are produced in the far-field grid. To avoid time-steps constrained by the size of cut-cells, I generalize the second-order accurate flux-calculation method of `CLAWPACK` to an enlarged h-box data structure. I show that h-box-`CLAW` is stable for uniform two-dimensional advection. The method extends naturally for solution of non-linear systems of hyperbolic PDEs, and is consistent both with the PDE and with `CLAWPACK` on structured grids; therefore numerical effects at grid interfaces are minimized. I apply h-box-`CLAW` to two scalar equations: non-uniform divergence-free advection, a linear scalar equation; and Burgers' equation, a non-linear scalar equation.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I am deeply grateful to my advisor, Professor Randy LeVeque, for introducing the subject of conservation laws to me with such clarity and insight, and for his continual and patient guidance. He supported travel to conferences and research labs which led to extremely helpful discussions. He has helped to create a stimulating environment here for the study of broader topics of numerical analysis. I have been inspired by his rigor and look forward to future collaborations with him.

I am thankful for the support, comments, careful reading, and suggestions of the members of my supervisory committee, Professors Loyce Adams, Jirair Kevorkian, and Scott Eberhardt. In addition, their courses were instrumental in increasing my understanding of numerical analysis, partial differential equations, fluid dynamics and other applications.

I would like to express my appreciation to Dr. Alan Mueller for his support while I was employed at QUEST, and for his continued confidence in my abilities.

Drs. Geoffrey Chesshire, William Henshaw, Jeffrey Saltzman and Timothy Barth were kind enough to discuss h-box problems and data structures with me—ideas springing from these discussions helped guide my research at critical points. Graduate students Hans-Jörg Forrer from ETH Zürich and Monika Wierse from Freiburg also made very helpful suggestions.

My wife, Jill Goldschneider, read my dissertation closely, helped clarify my expression, took on many tasks beyond the call of duty, and was gracious, patient and loving throughout. I hope I may do no less for her.

I would also like to thank the faculty, students and staff for supporting my education, providing excellent research facilities and for being a great group of fun and intelligent people to work with.

This work was supported in part by NSF Grants DMS-8657319, DMS-9204329, DMS-9303404 and DMS-9505021 and DOE Grant DE-FG06-93ER25181.

DEDICATION

To Jill

Chapter 1

INTRODUCTION

The trend in current engineering design practice is the abandonment of drafting tables and scale models for computational representation of the entire product (Computer Aided Design, or CAD). To avoid the expense of manufacturing and testing prototypes, engineers want to model all aspects of the product's intended use via computer simulation. To be practical, every step of a simulation, including incorporation of design changes, must be completed quickly and easily. Only after the completion of computational modeling is the prototype manufactured for final physical verification. A successful example of this design process is the Boeing 777 jumbo jet.

One simulation in which engineers are particularly interested is aerodynamic modeling. In designing an experimental, high-speed airplane, for example, one is interested in the time-dependent airflow through novel multiple-airfoil or fan-blade configurations. For a wide variety of such applications, airflow can be well modeled by the Euler equations for ideal gases, a non-linear hyperbolic system of partial differential equations (PDEs). This system of equations is derived from integral form of the Navier-Stokes equations, a set of conservation laws for compressible gases, when viscous effects are ignored. The non-linearity of the Euler equations implies that the total aerodynamic effect of an array of airfoils is not equal to the sum of the effects of the individual components. Instead, the complete aerodynamic solution requires the computational simulation of all interacting aerodynamic components. When more than one solid boundary is present in an aerodynamic design, or when the solid boundary is not a straight line or simple box, the geometry is considered complex. A prerequisite for numerical solution of an equation such as the Euler equations is the construction of a computational grid on which state variables (*e.g.*, density, velocity, pressure) are discretized. Grid generation for complex geometries can take almost as much time as the numerical solution. If the time constraint were simply that of initial grid generation, it could be considered to be a start-up cost. But the computational grid must be regenerated for *every* design change.

Therefore, a topic of active research in numerical analysis is the design of solution algorithms to be used in conjunction with adaptable and easily-generated grid types. A numerical method particularly suited to the Euler equations would be used widely, but as applied mathematicians we are interested in developing a general method, suitable for a range of hyperbolic PDEs. That method could then be applied to other physical problems involving complex geometries, such as seismology or ground-water flow of toxic contaminants.

For problems in which shock waves or discontinuities can form, such as high-speed gas flow, a partial differential equation is not valid because derivatives depend on sufficient smoothness of the solution. In these cases we require that the numerical method converge to a *weak solution* of the PDE; in other words, it must converge to a solution of the underlying integral conservation law from which the PDE was derived. A theorem due to Lax and Wendroff [31] tells us that if a stable numerical method for a hyperbolic problem has certain properties, among which is that it is *discretely conservative*, and it converges to a particular solution as the grid is refined, then that numerical solution in fact satisfies the conservation law. This theorem is one of the strongest tools in a numerical analyst's tool-box, because it implies that the numerical solution of hyperbolic problem, if obtainable, is close to what the physical model predicts, even without fine discretization. It is therefore crucial that the numerical quantities be conserved and that the method is stable.

In this dissertation I develop a stable, second-order accurate, discretely conservative algorithm, a modification of the h-box method of Berger and LeVeque [6–8], that makes it possible to take reasonably-sized time-steps in the numerical solution of a hyperbolic system of equations, even when that time-step might be limited due to the presence of a few small cells in an otherwise regularly-structured computational grid. This method has been used for discretely conservative solution of hyperbolic PDEs on a modified Chimera grid, a flexible and easily-generated type of composite grid described in more detail below. This method will make Chimera grid-generation tools more useful for those hyperbolic problems in complex geometries in which discrete conservation is important. Furthermore, because the h-box method need be applied only at the interfaces between structured sub-grids, standard methods can be used in most of the computational domain.

The finite volume method is a standard approach used for explicit solution of hyperbolic problems. The basic idea of the method is to consider a discretized quantity to be a cell average rather than a point value, as in the finite difference method. Then cell-averages are updated using a discrete form of the underlying integral conservation law. While more notation and terminology will be introduced later, it is instructive to apply a simple finite volume method on a Cartesian grid. We discretize the scalar partial differential equation[1],

$$q_t + f(q)_x + g(q)_y = 0 \tag{1.1}$$

where q is a conserved quantity, and f and g are the functions of q describing the flux in the x and y directions, respectively, to obtain the conservative finite volume update formula

$$Q_i^{n+1} = Q_i^n - \frac{\Delta t}{\Delta x \Delta y} \left(\left[F_{i+1/2,j}\, \Delta y - F_{i-1/2,j}\, \Delta y \right] + \left[G_{i,j+1/2}\, \Delta x - G_{i,j-1/2}\, \Delta x \right] \right), \tag{1.2}$$

[1] Subscripts t, x or y indicate a partial derivative with respect to that variable.

where Q_i^n is the average value of q in cell i at time t_n, and F and G are the numerical flux functions evaluated on the cell faces. With care in the calculation of flux functions, a highly accurate and stable method can be designed which converges to a physically correct solution as the grid is refined. While such a numerical method works well on Cartesian grids, and can be adapted easily (as shown in Section 4.3) for use on curvilinear structured grids, the complicated spatial boundaries found in "real-world" problems make generation of a single "reasonable" structured grid far more difficult.

During the 1980's and 1990's, many grid-generation schemes have been developed to deal with complex geometries[2]. While each method may be well-suited to a particular set of applications, almost all of these methods lack the ability to cope with changes in the solid boundaries. In 1983 Steger, Dougherty and Benek [51] introduced the *Chimera* grid scheme, an overlapping composite grid strategy which still shows promise for changing-geometry applications but remains under-explored[3]. Chimera, also known as *overset*, grids do not require grid-boundaries to match (Figure 1.1). They are well-suited for

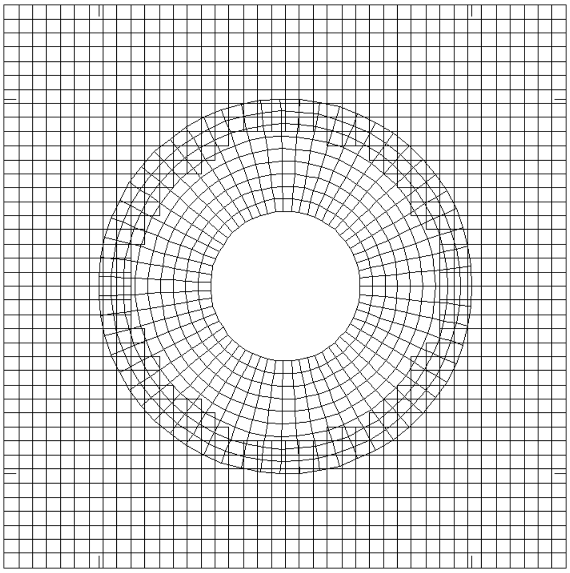


Figure 1.1: Example of a Chimera grid around a cylinder.

use in moving-body problems or situations in which the geometry needs to be changed quickly. They also work well in hybrid solvers, in which viscous effects are incorporated only in grid-cell layers near solid walls and those solutions are merged into inviscid solvers away from walls. However, current methods for Chimera grids are not discretely conservative. This is because data are multiply-defined in the overlap regions at sub-grid interfaces, which precludes explicit discrete conservation. This is addressed in

[2] See Thompson [55] for an excellent survey on the subject.

[3] Impetus for Chimera methods may have been lost due to the untimely death of Joseph Steger in 1992.

more detail in Section 1.2.3.

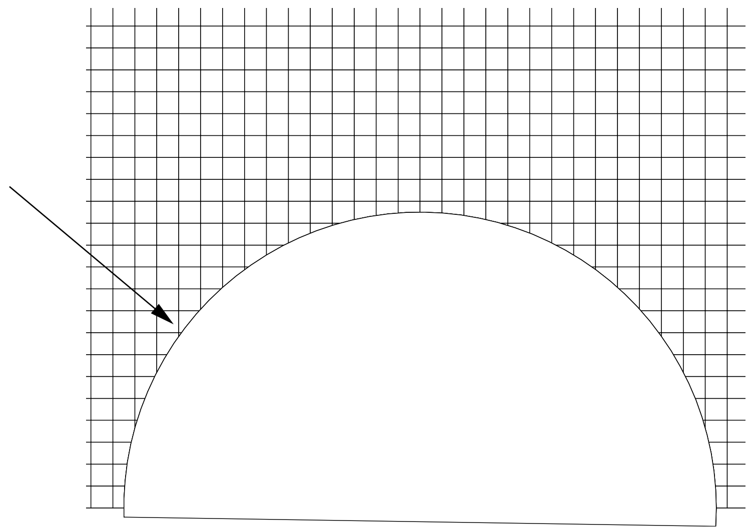


Figure 1.2: An embedded Cartesian grid containing a solid cylinder. Note that small, irregular cells are created by the boundary's "cookie-cutter" effect.

Inspiration for designing an explicitly-conservative Chimera grid method for hyperbolic problems comes from a class of techniques called *Cartesian grid*, *Cartesian boundary* or *embedded grid* methods. In these methods, solid boundaries are simply embedded into a Cartesian grid (Figure 1.2), without special consideration for alignment of the solid boundary with cell faces; thus cells will be partially chopped off by the boundary. This technique avoids almost all of the problems of grid-generation, but can cause difficulties in extending the model to include boundary layers or moving solids. More importantly, special handling of the flux into cells that have been cut by the boundary (referred to as *cut-cells*) is required to maintain stability if an explicit numerical method is used. Current embedded grid techniques do not attain full second-order accuracy for smooth solutions near boundaries, though this is hard to discern from the literature [41]. That is because embedded grids have been used in conjunction with adaptive grid refinement, and lower accuracy due to small cell techniques (or cell-merging) is dealt by refining in cells near the boundary.

This dissertation is concerned with one significant method for maintaining stability in embedded grids, the h-box method of Berger and LeVeque [6–8], because it is unlike other schemes that "fix" stability problems in calculations on small cells with either cell-merging or flux redistribution. The h-box method is based on extending the numerical domain of dependence for small cells. Therefore, it has the ability to create stable fluxes in a way that can be consistent with the overall numerical method. With modification, it is suitable for use on Chimera grids, stable on small cells without excessive time-step restriction, and second-order accurate for smooth solutions.

This work extends the h-box method to solution of hyperbolic partial differential equations on a modified Chimera grid. By modified Chimera, I mean that instead of allowing sub-grids to overlap, body-fitted grids are *embedded*, (or “inset” *vs.* overset), into far-field grids, producing small cells at sub-grid interfaces (Figure 1.3). Consequently, data on that grid are singly-defined, and a flux-based h-box method on that

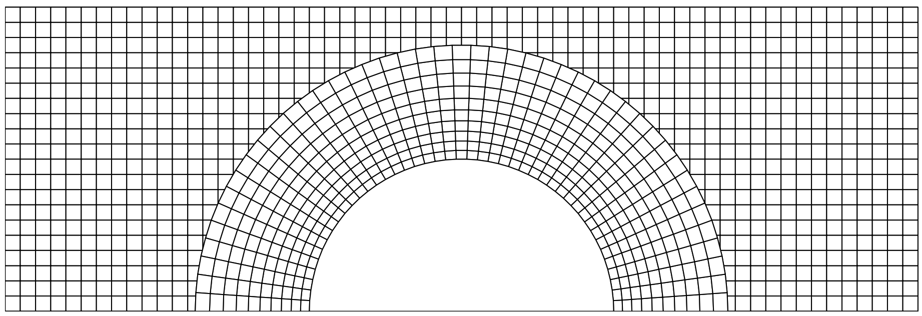


Figure 1.3: An embedded Chimera grid around a cylinder.

grid is discretely conservative. To be competitive in aerodynamic calculations, the method is second-order accurate for smooth solutions and is able to resolve shocks, discontinuities, and non-grid-aligned flow. Such a numerical method makes embedded Chimera grids and Chimera grid-generation programs useful for hyperbolic problems which require conservative algorithms. Application of the h-box method on an embedded Chimera grid *vs.* an embedded Cartesian grid also makes it possible to use a simpler version of the h-box method (Chapter 5).

The rest of this chapter is structured as follows: Section 1.1 begins with a description of the equations we wish to solve, and basic notation is introduced. In Section 1.2, the main grid generation techniques are introduced and advantages and disadvantages of each are summarized. The h-box Cartesian-grid method of Berger and LeVeque is described in Section 1.3, and the context of that method and its limitations are discussed. The topic of this work, an extension of Berger and LeVeque’s h-box method adapted for use on a composite grid, is introduced in Section 1.4. The structure of the remainder of this dissertation is outlined in Section 1.6.

1.1 Hyperbolic equations

Vector notation We use ordinary math script, a, for example, to denote scalar variables or constants. Italicized bold script denotes vector quantities, for example

$$\boldsymbol{a} = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}.$$

Roman bold script refers to a matrix (an array of column vectors):

$$\mathbf{A} = \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} = \begin{pmatrix} \boldsymbol{a}_{\cdot,1} & \boldsymbol{a}_{\cdot,2} \end{pmatrix}.$$

Finally, a triply-indexed tensor (*e.g.*, $\{a_{i,j,k}\}$), is denoted with blackboard-bold script and may be considered as an array of "column" matrices:

$$\mathbb{A} = \begin{pmatrix} \mathbf{A}_{\cdot,\cdot,1} & \boldsymbol{A}_{\cdot,\cdot,2} \end{pmatrix}.$$

We restrict this research to the time-accurate solution of homogeneous hyperbolic systems of partial differential equations in two dimensions, which have the form

$$\boldsymbol{q}_t + \boldsymbol{f}(\boldsymbol{q})_x + \boldsymbol{g}(\boldsymbol{q})_y = 0, \tag{1.3}$$

where $\boldsymbol{f}(\boldsymbol{q})$ and $\boldsymbol{g}(\boldsymbol{q})$ are vector functions of $\boldsymbol{q}$, a vector of conserved state variables. The system is hyperbolic when the matrices

$$\begin{aligned} \mathbf{A} &\equiv \boldsymbol{f}'(\boldsymbol{q}) \\ \mathbf{B} &\equiv \boldsymbol{g}'(\boldsymbol{q}) \end{aligned}$$

are diagonalizable with real eigenvalues and distinct, real eigenvectors for all values of $\boldsymbol{q}$.

Let $\mathbf{F}$ be the matrix whose columns are the flux vectors in the coordinate directions,

$$\mathbf{F} \equiv \begin{pmatrix} \boldsymbol{f} & \boldsymbol{g} \end{pmatrix}. \tag{1.4}$$

Defining the matrix-vector product,

$$\mathbf{F} \cdot \boldsymbol{a} \equiv a_1 \boldsymbol{f} + a_2 \boldsymbol{g},$$

and letting $\boldsymbol{n}$ represent the outward-pointing normal on a contour, we may state concisely that equation (1.3) holds for smooth solutions of the integral conservation law

$$\frac{\partial}{\partial t} \iint\limits_{\Omega} \boldsymbol{q}\, dA = -\int\limits_{\partial\Omega} \mathbf{F} \cdot \boldsymbol{n}\, ds. \tag{1.5}$$

A more general form of the conservation law that is still valid when $\iint\limits_{\Omega} \boldsymbol{q}\, dA$ is discontinuous with respect to time is

$$\left[\iint\limits_{\Omega} \boldsymbol{q}\, dA \right]_{t_1}^{t_2} = -\int_{t_1}^{t_2} \int\limits_{\partial\Omega} \mathbf{F} \cdot \boldsymbol{n}\, ds\, dt. \tag{1.6}$$

The numerical method used to solve (1.3) should be constructed so that in cases where (1.3) is not strictly valid, *e.g.*, where components of $\boldsymbol{q}$ have discontinuities, the calculated solution still satisfies either (1.5) or (1.6) (*i.e.*, it is a weak solution of (1.3)).

To test the numerical method developed in this dissertation, we will examine three types of hyperbolic PDEs in complex two-dimensional geometries: advection of an initial concentration profile in a prescribed incompressible velocity field,

$$\begin{aligned} q_t + u(x,y)\, q_x + v(x,y)\, q_y &= 0 \\ \text{where} \qquad u_x(x,y) + v_y(x,y) &= 0, \end{aligned} \tag{1.7}$$

as an example of a scalar equation whose wave speeds are spatially dependent; and a two-dimensional form of Burgers' equation,

$$q_t + \left(\frac{q^2}{2}\right)_x + \left(\frac{q^2}{2}\right)_y = 0, \tag{1.8}$$

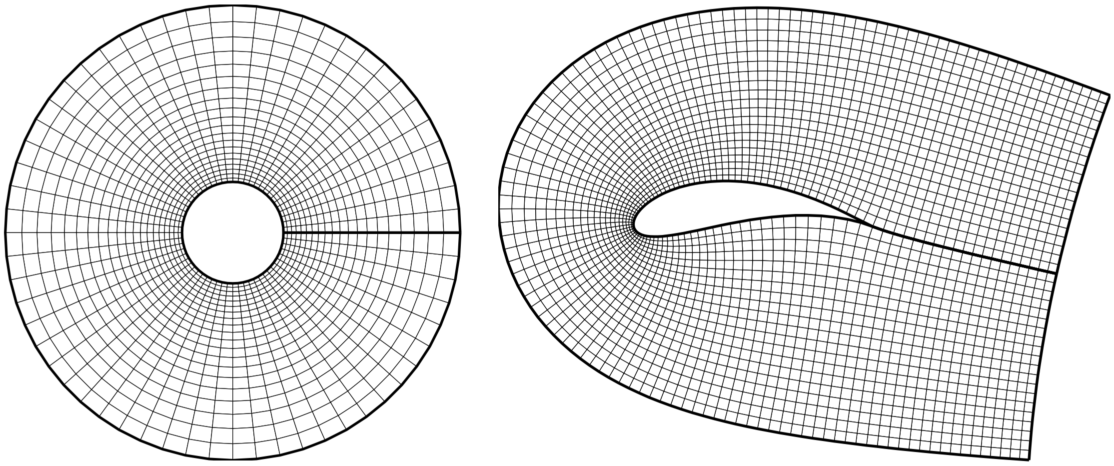
an example of a simple non-linear scalar equation with wave speeds dependent on q.

1.2 Grid structures for computation

Not surprisingly, grid-generation techniques vary widely, and many types of grid structures have been developed for numerical solution of PDEs in complicated geometries. Most methods use one of three main categories of grids: structured, unstructured, or composite. Although this dissertation focuses on a composite type, the Chimera grid, we summarize all of these grid classes, as aspects of each will be incorporated into this work.

1.2.1 Structured grids for numerical methods

Numerical solutions of (1.3) were first computed with structured grids, even before the advent of computers. In its most basic form, a structured grid divides space into rectangular cells of uniform width and height, then (1.3) is solved using any of a variety of techniques (*e.g.*, finite difference, finite volume, or finite element methods). To use these methods in complicated geometries, however, it is necessary to find a transformation from a logically rectangular, uniform computational grid to curvilinear coordinates in such a way that all physical boundaries are aligned with the curvilinear coordinates (discussed in Section 4.3). While this is not overly difficult for simple boundary shapes (Figure 1.4), for more complicated boundaries, or domains with multiple solid bodies this may be a nearly impossible task. Even for simple grids, such as Figure 1.4(a), it is difficult to avoid distortion in cells at a distance from the boundary. There is a rich literature devoted to simulating far-field in-flow, out-flow, or absorbing boundary conditions (*e.g.*, [18]), partly to avoid having to extend grids too far from solid boundaries.



(a)

(b)

Figure 1.4: Simple structured grids, constructed with conformal maps: (a) “O” grid around a cylinder; (b) “C” grid around a Joukowski airfoil.

A constraint on grid generation is that numerical methods for structured grids degrade in accuracy when the physical grid is not orthogonal. Though algebraic or hyperbolic grid generators can produce nearly-orthogonal grids with much less work than fully elliptic grid generators, extra processing may be required to orthogonalize the grid, especially near solid boundaries. Orthogonalization usually involves a number of relaxation steps to approximate the solution of an elliptic equation.

1.2.2 Unstructured mesh[4] generation for numerical methods

Numerical solution on unstructured meshes is directly motivated by the difficulty of fitting structured grids to complicated geometries. Unstructured meshes were originally developed by engineers for use with finite element methods in solid mechanics, since FEMs are formulated so they may be implemented on either structured grids or unstructured meshes, and the shape of individual elements does not have to be rectangular — they may be triangles in two dimensions or tetrahedra in three dimensions. But the availability of unstructured mesh generation for complicated geometries has encouraged development of finite volume methods for unstructured meshes [3]. In two dimensions, unstructured meshes are usually constructed by scattering points in an

[4] We refer to unstructured *meshes* rather than *grids* since the ordering of nodes on an unstructured mesh is not constrained to be logically rectangular. But both structured and unstructured discretizations are commonly referred to as either meshes or grids, and the term Adaptive Mesh Refinement (AMR) is used to denote local subdivision on structured grids, not on unstructured meshes.

unordered fashion and then the points are connected using an efficient triangulation technique; *e.g.*, a Delauney triangulation. The ordering of points and edges is then be adjusted to narrow the bandwidth of the connectivity array.

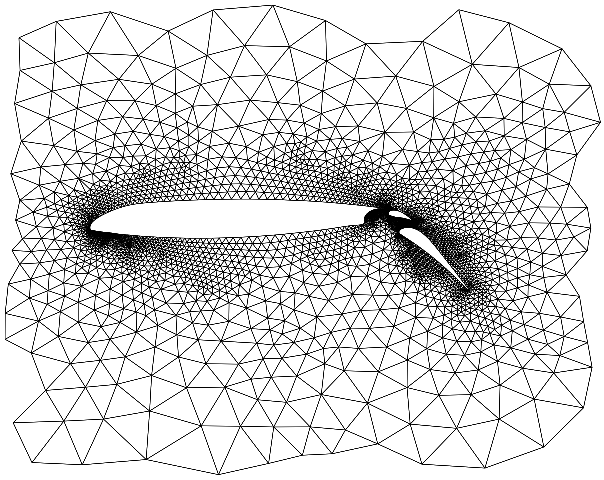


Figure 1.5: An unstructured mesh around a NASA airfoil. This picture was produced by the MATLAB demo program `airfoil.m`.

Several sophisticated methods have been developed to adapt unstructured meshes in regions of complex flow or boundaries (Figure 1.5). Unstructured meshes were slow to gain popularity for hyperbolic problems at first because of the additional connective complexity, *i.e.*, what elements (cells) are connected to a given edge (face). On a structured grid, by contrast, one always knows that the neighbors of cell $C_{i,j}$ are cells $C_{i\pm1,j\pm1}$). On an unstructured mesh it is also difficult to determine spatial derivatives in a simple way, therefore different numerical methods have to be used.

Connectivity complexity also increases the difficulty of numerical solution using parallel computers: partitioning the computational domain to distribute blocks of data to several processors is more difficult because local blocks of cells cannot numbered in a simple sequential way (compare with the partitioning of a structured grid on which one might assign indices with a range in i, j to a single processor).

For hyperbolic problems around solid bodies, one usually is modeling a physical problem that has boundary layers near the body. Boundary layers will be present even in an inviscid solver because of numerical diffusion. To resolve boundary layers cleanly, one therefore needs a body-fitted mesh with cells that are quite thin close to the surface, but relatively long tangential to the surface. This grid modification is called “stretching”. It is hard to specify stretched cells with unstructured mesh generators, so these generators must be combined with structured gridding near solid boundaries when stretched cells are necessary—long rectangular cells from quadrilateral structured grid generation are triangulated.

Finally, unstructured meshes can contain a wide range of cell sizes, making it difficult to calculate explicit time-accurate solutions for hyperbolic problems.

Therefore, though unstructured grids have become quite popular in many applications with complicated geometries, structured grids remain the most attractive for use in time-accurate solvers.

1.2.3 Composite grids

In the past 20 years researchers have tried to side-step the problem of generating a globally structured grid by generating a composite grid consisting of multiple structured sub-grids. Each solid boundary (or a portion of a boundary) has a local sub-grid attached to it. The sub-grid would in a sense represent natural coordinates near the solid wall. The main work of grid-generation then reduces to that required to handle sub-grid interfaces. In this way one might combine the strengths of structured grid methods with the adaptivity of unstructured meshes.

Composite grid methods can be divided into three general categories, which differ in how sub-grid boundaries are joined together: multiblock structured grids; *Chimera* grids; and hybrid grids.

Multi-block composites

Multiblock composites may be the most useful for generating surface-fitted grids, especially for complicated solids in three dimensions. However, the expense of regenerating grids makes them costly to use when changing geometry is important.

A multiblock composite grid is one in which each of the grids is logically rectangular, with no shadowed (blocked out) regions. A classical example of this kind of composite is seen in Figure 1.6. The boundary of each sub-grid adjoins either another sub-grid or a solid boundary.

Multiblock grids are further differentiated as aligned or non-aligned. The difference is clearly illustrated in Figure 1.7: in aligned multiblock grids, the boundary-perpendicular grid-lines from one side of the grid interface meet those from the other side. Most aligned multiblock generators also constrain the spacing normal to the interface to be nearly continuous across the interface. Many commercial and public domain codes have been developed to aid in the production of aligned multiblock grids in two and three dimensions. Some of the more familiar packages are `EAGLE`, `GridPro`, `GRID*`, `CFD-GEOM`, `GRIDGEN` and `ICEM-CFD`. Many more references can be found in [55]. The main advantage of aligned multiblock structured grids is the simple connectivity between sub-grids, which makes boundary conditions relatively easy to apply, *once grids have been generated.* However, generating the grid can still be difficult: in essence one must construct a coarse unstructured mesh with quadrilateral elements. But even after generating the multiblock grid, other problems remain. Boundary conditions may

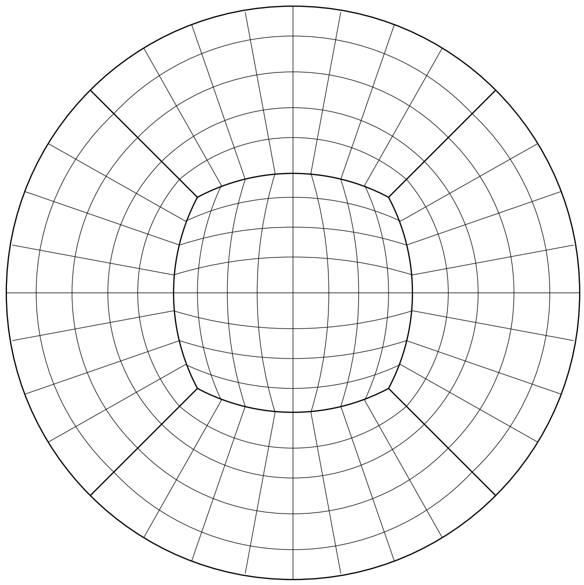


Figure 1.6: A multi-block composite (with aligned boundary points) is used to subdivide a circle and avoid a singularity at the center.

be fairly easy to apply at simple sub-grid interfaces, but can be trickier at corner singularities (*i.e.*, the intersection is not topologically equivalent to a rectangular corner). Multiblocks with aligned grid spacing also share a problem of structured grids: local grid-line clustering can have a global effect. Finally, grids have to be regenerated any time there is a geometry change.

To increase the flexibility of multiblock grids, Furukawa *et al.* [20], Arbogast *et al.* [2] and others have developed methods that use non-aligned blocks (Figure 1.7(b)). Transferring data across sub-grid interfaces in such grids is less straightforward, but such methods are useful for problems in which sub-grid interfaces correspond to faults or other sliding boundaries.

Chimera grids

In Chimera grids the multiblock restriction that the edges of sub-grids match is relaxed. Instead, sub-grids are laid out in a convenient fashion to be aligned with boundaries, flow features, *et al.* As new grids are added to the composite, they *overlap* previously generated grids. In the neighborhood of the solid body, cells on the far-field grid that are overlapped by the new body-fitted grid are not necessary for calculation. We say that these cells are shadowed, or blocked out, by the overset grid; *i.e.*, those cells are marked as non-physical and are not used (except when computationally convenient) for calculation or plotting. For example, Figure 1.8 shows that Cartesian cells underneath the diagonal grid are disregarded, except those required for boundary data interpolated

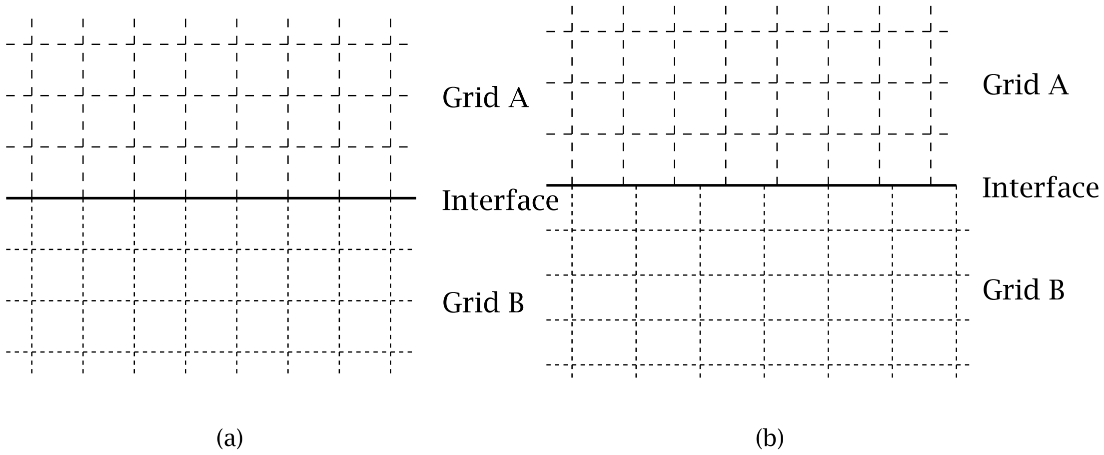


Figure 1.7: (a) Aligned multiblock grids. (b) non-aligned Multiblock grids. In (a) it required only that the normal grid-spacing be C^0 across the boundary, even though the figure indicates that the spacing is C^1.

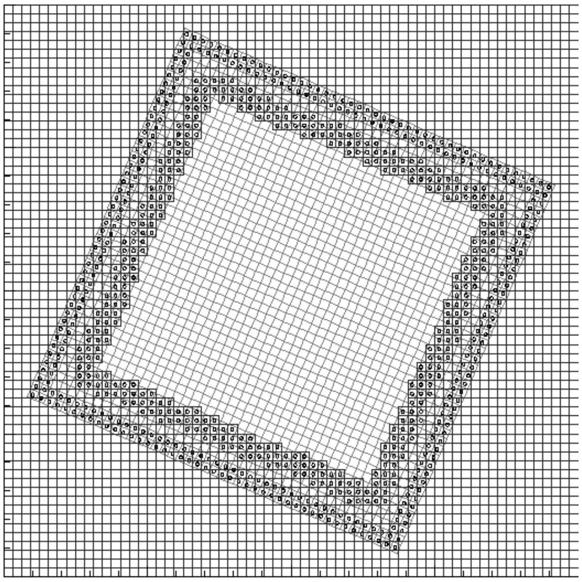


Figure 1.8: This overset grid example (produced with CMPGRD [11] software) is not fitted to any solid boundary, but does demonstrate the overlapping of the oblique grid on top of the original Cartesian grid. Notice that unused original cells have been shadowed (blanked out).

from the diagonal sub-grid.

Chimera grids have been used successfully for the solution of parabolic or elliptic problems, moving boundary problems, and changing geometries. At the boundaries of each grid, boundary cell data is approximated by interpolating information from the other grid. A one-dimensional cross-section of the sub-grid interface, Figure 1.9, demonstrates that the overlap region is a zone in which the information, and consequently conservation, is only as accurate as the interpolation. However, Chimera grids have been used successfully for hyperbolic systems where solutions are smooth. Recent research [13–15, 25, 26, 39, 40, 48] demonstrates that hyperbolic problems with smooth solutions can be solved to 4th and 5th-order accuracy. For these problems the Chimera grid is quite satisfactory, and in the case of iteration of a hyperbolic system to steady state, non-conservative effects are minimized by smoothing, very-high-order methods, and grid refinement.

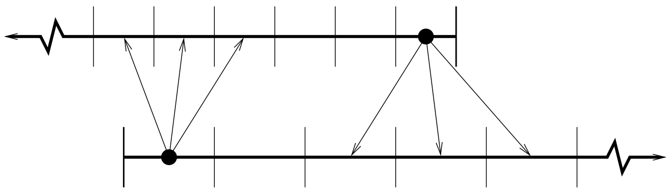


Figure 1.9: One-dimensional slice of overlapping grid interface

For some hyperbolic problems, however, solutions can be discontinuous; *e.g.*, shock waves and contact discontinuities can be present in high-speed compressible gas dynamics. Besides the increased expense of maintaining high accuracy interpolation in these cases, non-conservative numerical solutions on overlapping Chimera grids cannot be guaranteed to converge to weak solutions of (1.3). A numerical method only has this guarantee, if it converges to a solution under refinement at all, if it is discretely conservative and consistent (in the sense that the numerical flux of a constant $q(x,t) = \overline{q}$, $F(\overline{q}, \overline{q})$ is equal to $f(\overline{q})$). This theorem was proved for scalar hyperbolic equations by Lax and Wendroff [31].

Chesshire and Henshaw [14] have developed a conservative interpolation scheme for Chimera grids, in which a linear constraint equation is solved for interpolation coefficients. This step is implicitly dependent on all the cells in a grid-interface region. While conservative interpolation may be useful in conjunction with numerical methods for elliptic or parabolic problems which are also solved implicitly, this technique loses the advantage of simplicity and local domain-of-dependence that one expects in an explicit hyperbolic solver.

The adaptability of Chimera grids to changing geometries, however, makes it desirable to develop a higher-order explicitly conservative method for them. A possible alternative to the Chesshire and Henshaw's constrain interpolation method is to use modified Chimera grids using an idea from embedded grid techniques (Section 1.3).

1.2.4 Hybrid Meshes

Hybrid meshes are anything in between composites of structured grids and unstructured meshes. In particular, Kao, Liou and Chow [27] have developed the so-called "DRAGON" grid technique: standard overset grids are modified by removing a layer of cells in the region between grids and filling that region with an unstructured mesh. I will not discuss this in more detail except to note that this method shares the same disadvantages as unstructured grids for adaptive grid refinement with time-accurate solutions, though in a much smaller portion of the entire spatial domain. Moreover, it is difficult to find a numerical method that can be used identically in both the structured sub-grids and the unstructured mesh grid-interfaces, so some inefficiency in the solution must be accepted at the unstructured interface.

Much research has been done to combine structured near-body cells with far-field unstructured meshes, and in those cases, the same unstructured solver is used in the entire computational domain.

1.3 Embedded grids and the h-box method

Berger and LeVeque [5–8]describe a *Cartesian grid method* for time-accurate solution of the Euler equation in complex geometries. The main idea of their work is that no special treatment is given to generate grids which follow solid boundaries. Instead, solids are simply *embedded* (or inset) into the grid, as seen in Figure 1.2, with no regard for matching grid lines to the body. Several strategies were developed for calculation of stable fluxes on the faces of small cells. One of these, a flux algorithm called the h-box method, successfully calculates stable fluxes for cut-cells. Since this approach was built into an adaptive grid refinement scheme, extremely accurate solutions could be found while bypassing traditional grid generation costs (*i.e.*, those entailed in fitting structured grids to boundaries) almost completely. A cartoon example of a solid boundary embedded into a Cartesian grid is shown in Figure 1.10. In the same figure, we also see 2 pairs of h-boxes associated with a face of a cut-cell. The basic idea of the h-box method is to reconstruct a larger numerical domain of dependence on a face in such a way that the conservative update on a small cell is formed of fluxes which nearly cancel. This will be explained in more detail in Chapter 2 and so we need not go into more detail here. As implemented by Berger and Leveque, however, the h-box method was less than second-order accurate for smooth solutions in small cut-cells. Relatively high-order accuracy for the overall method has been maintained through use of grid refinement. It will shown later in the dissertation that the boundary-aligned h-box method (Figure 1.10) has stability restrictions for flow oblique to the boundary. This result has not been noted previously.

Other researchers have also explored embedded Cartesian grid methods. Quirk [41] developed a similar technique to that of Berger and LeVeque. His *Cartesian boundary scheme* differs from the h-box method in the treatment of large time-steps in cut-cells:

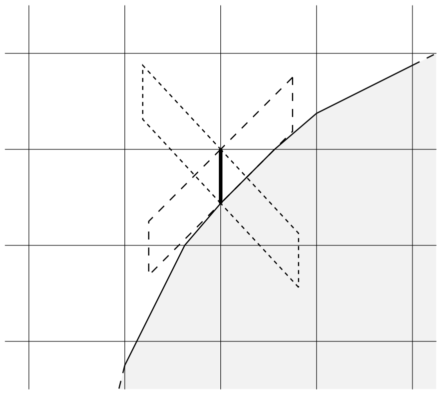


Figure 1.10: A solid body embedded in Cartesian grid, and the h-boxes associated with one of the partial cell-faces.

where cut-cells are too small to ensure stability with a reasonable time-step, they are merged into neighboring cut or regular cells to form a larger conserved region. This method performs reasonably well when incorporated into an adaptive grid refinement method, but like the h-box method, is less than second-order accurate in cut-cells. De Zeeuw and Powell [17], Aftosmis *et al.* [1] and others have developed embedded grid schemes as well, in conjunction with quad-tree (oct-tree in 3D) adaptive grid refinement methods. These researchers use local time-stepping or cell-merging to avoid time-step restrictions due to cut-cells. Pember *et al.* [38] describe a flux-redistribution technique for embedded grids: a cut-cell value is pseudo-updated using a stable, non-conservative method. The non-conservative portion of the pseudo-update is redistributed in a volume-weighted fashion to neighboring cells by adjusting the flux on each face of the cut-cell. Finally, a conservative cell update is made using the fixed fluxes.

1.4 A new composite grid method

While the h-box method as implemented by Berger and LeVeque is not discussed in great detail above, certain complexities and drawbacks of that scheme should be addressed. Their scheme constructs h-boxes in a local coordinate system normal and tangential to the boundary; therefore we denote this version as the *normal-tangential* h-box method. In the case of numerical solution, *e.g.*, for the Euler equations of inviscid gas flow, it is therefore necessary to reconstruct information *inside* the solid body with reflecting boundary conditions, in order that normal h-boxes using information from inside the solid will produce fluxes satisfying zero normal-flux boundary conditions. Furthermore, though the embedded Cartesian grid can be refined near the body, a Cartesian grid precludes any use of boundary-layer modeling except in specific cases where the body is aligned with the grid. Finally, numerical results indicate that the normal-tangential h-box method does not attain full second-order accuracy, though this deficiency may be addressed partly through local grid refinement.

It would be more straightforward to design a flux algorithm on the faces of small cut-cells if the cutting edge were at a distance from the boundary—it would then not be necessary to use a method which satisfies solid-boundary conditions at the same time as it maintains stability with high cell CFL numbers. We therefore consider this alternative scenario: put a body-fitted grid around the solid and cut into the far-field Cartesian grid using the outer edge of the body-fitted grid, instead of the body's edge. This idea has some attractive characteristics. By guaranteeing that h-boxes will always be well-separated from solid boundaries, it is no longer necessary to construct h-boxes in normal-tangential coordinates: we may use a Cartesian orientation for h-boxes instead. Another advantage of the proposed grid is that the body-fitted grid can be used to satisfy solid-boundary conditions without difficulty. We have constructed a Chimera grid, but not an overlapping one. By embedding the body-fitted grid, data is singly-defined. We can use the h-box method to solve hyperbolic equations on this proposed grid with discrete conservation, thus overcoming one of drawbacks noted in Section 1.2.3, while reaping the other advantages of Chimera grids. It is logical to call the newly-proposed grid an *Embedded* Chimera[5], since it is a hybrid of the two grid types. Figure 1.3 shows an example of such a grid.

To distinguish the simplified form of the h-box method on an embedded Chimera grid from the normal-tangential h-box method, we denote it as the *Cartesian-oriented* h-box method. Use of the Cartesian-oriented h-box method on an embedded Chimera brings up an issue that was not present in the original h-box implementation: the difference in orientation of sub-grids on each side of an interface. In embedded Cartesian grids, there is a single grid mapping, and no data ever flows across a solid boundaries, therefore it is never necessary for data to flow from one grid-orientation to a differently-oriented representation on another grid. On an embedded Chimera grid however, this must be considered. The new h-box method will work best, that is with the least discernible effect on the solution, when the numerical solution on each side of an interface is similarly resolved. Furthermore, the numerical method used on each structured sub-grids should produce few grid-dependent anisotropic numerical effects, to minimize distortion in the solution when information propagates across sub-grid interfaces. We choose for the numerical flux calculation algorithm, therefore, an implementation of the algorithm used in `CLAWPACK` [33] (a hyperbolic **C**onservation **LAW**s software **PACK**age). The form of this numerical flux that is used in this dissertation on curvilinear structured sub-grids is discussed in Chapter 4.

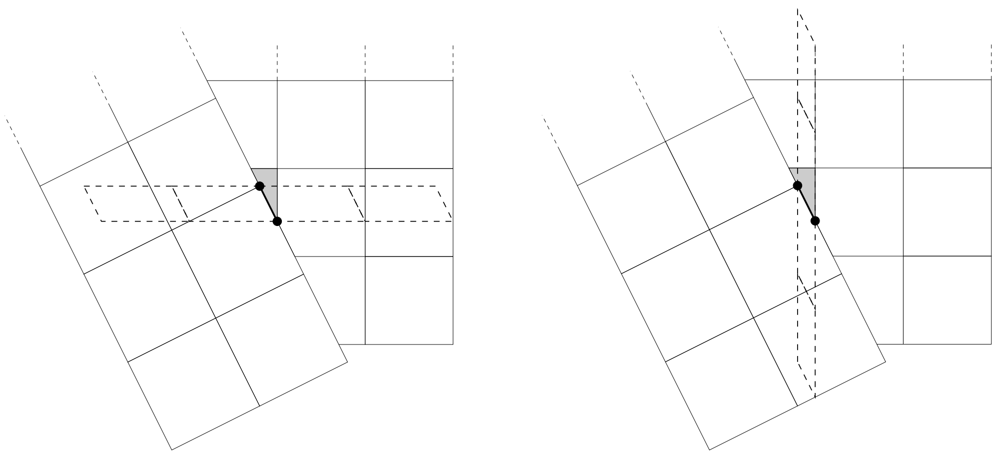
Because this dissertation's work is motivated by the twin goals of simplifying the h-box method and designing a discretely conservative method for hyperbolic problems on Chimera grids it does not proceed farther. However, the numerical method has

[5] Or an *Inset* grid, by contrast with the term *Overset*.

been developed with the intention of eventually incorporating it into an adaptive grid refinement method for Chimera grids, where it might be used for moving body problems.

We now discuss the main topic of this dissertation, implementation of the Cartesian-oriented h-box method on an embedded Chimera grid.

1.4.1 The h-box method on an embedded Chimera grid



(a) (b)

Figure 1.11: h-boxes aligned with (a) x and (b) y coordinate directions on a small cell face at a grid interface.

In the h-box method, the numerical domain of dependence for fluxes on the faces of a cut-cell (or *cut-faces* is extended beyond the nearest-neighbor cells. Because resulting h-box fluxes on cut-faces will be calculated using nearly-overlapping information, flux differences will largely cancel during conservative updates. We illustrate a few of the h-boxes on a small cut-cell face in Figure 1.11[6]. For the calculation of large-time-step-stable fluxes, it will be necessary to associate more h-boxes with a cut-face than those illustrated in Figure 1.11. However, this figure illustrates that much of the flux calculation can be understood by examining the small cell problem in one dimension. We investigate the stability and accuracy of the h-box method on a non-uniform one-dimensional grid in Chapter 2. However, for full second-order accuracy the new h-box-method must incorporate transverse flux corrections; that is, corrections that

[6] It should be noted that h-boxes have uniform (or smoothly varying) length equal to the local value of Δx or Δy on the grid being cut, depending on the axis with which they are aligned.

lead to mixed xy partial derivatives in cell updates. Since this will require a larger array of h-boxes on each cut-face, we call the complete set of h-boxes associated with a cut-face a *FaceGrid*. Transverse flux corrections will require reconstructed $\boldsymbol{q}$ values in FaceGrid h-boxes not shown in Figure 1.11. It is important to design the transverse flux correction algorithm and the data structures necessary for its implementation so that the resultant fluxes are consistent with those calculated on a structured grid. The uniform grid form of transverse correction, as calculated by CLAWPACK, is discussed in Section 4.2.2. We generalize that transverse-correction algorithm for use on a cut-cell face's h-box structure in Section 5.3.3.

1.5 Competitive Embedded Chimera schemes

There are at least two other research projects developing methods for embedded Chimera grids. Saldou [47], working with Colella, adapts the embedded grid flux-redistribution idea (from Pember *et al.* [38]) to keep small cell solutions from exceeding local maxima or minima. This method relies on obtaining a good estimate of the cut-cell solution, but has had difficulty in determining that estimate in a consistent way. As a by-product of the stability analysis for the h-box method in two-dimensional advection, we indicate (Section 5.3.3) what the proper stable estimate for pseudo-updates in cut-cells should be in a flux-redistribution scheme.

An alternative embedded Chimera method is that of Zhi-Jian Wang [58, 59], which uses a cell-merging idea, similar to the Cartesian boundary method of Quirk [42], to maintain stability in small cells. While perhaps useful for quick estimation of solutions, Wang's method is still limited to first-order accuracy near grid interfaces. Furthermore, the time-step for that method is half that of the h-box method developed in this dissertation, since no transverse-flux corrections are used on small cells.

1.6 Outline of the remaining dissertation

The h-box method for one-dimensional non-uniform grids is described in detail in Chapter 2. We analyze the stability and accuracy of the h-box method for uniform advection and discuss stability for Burgers' equation. In Chapter 3 we discuss numerical results for the 1D h-box method.

In Chapter 4 we discuss finite volume methods for structured grids in 2D, introduce LeVeque's CLAWPACK methodology, and describe a flux-based form of CLAWPACK for quadrilateral grids.

In Chapter 5 we introduce notation for the embedded Chimera grid and 2D h-box data structures. Flux-calculation algorithms for cut-cell faces are developed. We show that the h-box method for uniform advection in cut-cells is second-order accurate and stable for CFL numbers ≤ 1 in each dimension.

Numerical results in two dimensions for several partial differential equations are shown in Chapter 6.

Chapter 2

THE h-box METHOD IN ONE DIMENSION

Before discussion of the solution of scalar hyperbolic partial differential equations (PDEs) in one dimension, we define notation in Section 2.1 and present an example of accuracy and stability estimates for the advection equation on a uniform grid in Section 2.2. This discussion elaborates on results previously noted in [9].

2.1 Notation for Discretization

We discuss here the solution of scalar hyperbolic PDEs in one dimension of the form

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

or, in non-conservative form,

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

The particular systems used here as examples are the advection equation,

$$q_t + uq_x = 0, \quad \text{for constant } u > 0, \tag{2.3}$$

and Burgers' Equation,

$$q_t + (q^2/2)_x = 0. \tag{2.4}$$

The integral conservation law form of (2.1) is

$$\left[\int_{x_1}^{x_2} q(x,t)\ dx\right]_{t_n}^{t_{n+1}} = -\left[\int_{t_n}^{t_{n+1}} f(q)\ dt\right]_{x_1}^{x_2}. \tag{2.5}$$

We study a finite-volume discretization of $q(x, t_n)$ in one dimension with variable grid size. The boundaries for cell i, centered at x_i, are $x_{i-1/2}$ and $x_{i+1/2}$. The width of cell i is

$$h_i = x_{i+1/2} - x_{i-1/2}. \tag{2.6}$$

We define the uniform distance h as an upper bound for all cell sizes:[1]

$$h \geq \max_i h_i. \tag{2.7}$$

The center of the i-th cell is

$$\begin{aligned} x_i &= \frac{1}{2}\left(x_{i-1/2} + x_{i+1/2}\right) \\ &= x_{i-1/2} + h_i/2. \end{aligned} \tag{2.8}$$

We refer to the points at uniform distance h to the left and right of cell-boundary $x_{i-1/2}$ as

$$x_{i-1/2,L} \equiv x_{i-1/2} - h \tag{2.9}$$

$$x_{i-1/2,R} \equiv x_{i-1/2} + h. \tag{2.10}$$

We store in $Q_i^n, i = 1, \ldots, n$ numerical approximations to cell averages,

$$\frac{1}{h_i} \int_{x_{i-1/2}}^{x_{i+1/2}} q(x, t_n)\, dx,$$

and use those quantities to determine Q_i^{n+1}, based on a time-step $k = t_{n+1} - t_n$. Without loss of generality, we may assume that $k \equiv \Delta t$ is constant. We denote the value of $q(x_i, t_n)$ as q_i^n, and frequently drop the superscript n. Note that capitalized Q_i refers to the numerical cell average, but it agrees with the cell-center point value q_i to $O({h_i}^2)$. We also refer to the point value $q(x_i - \frac{h}{2}, t_n)$, which we denote as $q_{i-1/2}^n$ or $q_{i-1/2}$. Note that on a non-uniform grid this value does not necessarily correspond to $q(x_{i-1/2}, t_n)$. For the distance between cell centers we use the notation

$$\begin{aligned} h_{i-1/2} &= x_i - x_{i-1} \\ &= \frac{1}{2}\left(h_{i-1} + h_i\right). \end{aligned} \tag{2.11}$$

Total variation (TV) denotes the L^1 norm of a function's derivative:

$$\mathrm{TV}(q) = \int \left| q'(\xi) \right| \, d\xi.$$

Total variation measures the volatility of a function. Here it is used to establish a bounds on the instability of a numerical method. In the discrete case, total variation takes the

---

[1] In forming the random grid used for 1D numerical tests, h is chosen first and then used to determine random h_i's uniformly distributed in the interval $(0, h]$.

form

$$\mathrm{TV}(Q^n) = \sum_i \left| Q^n_{i+1} - Q^n_i \right|,$$

which is independent of the uniformity of the grid. When the numerical method has the property that $\mathrm{TV}(Q^{n+1}) \leq \mathrm{TV}(Q^n)$, the method is said to be *total variation diminishing* (TVD).

2.2 Godunov method for the Advection Equation on a Uniform Grid

In this section we examine the accuracy and stability of an $O(h)$ Godunov upwind method for advection with constant velocity $u > 0$.

For a uniform grid, $h_i = h, i = 1, \ldots, n$, the Godunov upwind method can be written as either

$$\frac{Q^{n+1}_i - Q^n_i}{k} = -u\, \frac{Q^n_i - Q^n_{i-1}}{h} \tag{2.12}$$

or

$$Q^{n+1}_i = Q^n_i - \frac{uk}{h}\left(Q^n_i - Q^n_{i-1}\right). \tag{2.13}$$

For constant u we can rearrange this into a form more suitable for stability analysis. Substituting $\nu = uk/h$ and rearranging,

$$Q^{n+1}_i = (1-\nu)Q^n_i + \nu Q^n_{i-1}. \tag{2.14}$$

This formulation leads to the classical result:

Theorem 2.1. *if ν is less than* 1, *the Godunov method update Q^{n+1} has decreasing norm in both L^1 and TV.*

It is worth reviewing the standard proof for comparison with similar ideas used in our analysis of non-uniform grid methods:

Proof. Assume $0 \leq \nu \leq 1$. If we assume that Q is finite and is bounded in L^1 then we take the L^1 norm of (2.14) and divide both sides by the constant factor h:

$$\begin{aligned}\sum_i \left|Q^{n+1}_i\right| &= \sum_i \left\{ \left| (1-\nu)Q^n_i + \nu Q^n_{i-1} \right| \right\} \\ &= \sum_i \left\{ \left| (1-\nu)Q^n_i \right| + \left| \nu Q^n_{i-1} \right| \right\}.\end{aligned}$$

By the triangle inequality, and since $0 \le \nu \le 1$,

$$\sum_i \left| Q_i^{n+1} \right| \le \sum_i \left\{ (1-\nu) \left| Q_i^n \right| + \nu \left| Q_{i-1}^n \right| \right\}$$

ν constant implies

$$\sum_i \left| Q_i^{n+1} \right| \le (1-\nu) \sum_i \left| Q_i^n \right| + \nu \sum_i \left| Q_{i-1}^n \right|$$

L^1 boundedness implies $\sum_i \left| Q_i^n \right| = \sum_i \left| Q_{i-1}^n \right|$, so

$$\begin{aligned} \sum_i \left| Q_i^{n+1} \right| &\le (1-\nu) \sum_i \left| Q_i^n \right| + \nu \sum_i \left| Q_i^n \right| \\ &\le \sum_i \left| Q_i^n \right| \end{aligned} \tag{2.15}$$

A similar argument holds for the total variation:

$$\sum_i \left| Q_i^{n+1} - Q_{i-1}^{n+1} \right| \le (1-\nu) \sum_i \left| Q_i^n - Q_{i-1}^n \right| + \nu \sum_i \left| Q_{i-1}^n - Q_{i-2}^n \right|,$$

and the sum can be shifted as before because ν is constant. ■

Below we look at how one shows stability in L^2 using von Neumann analysis which will be useful for comparison later.

If we look at this method as a finite difference method instead of a finite volume method, we have

$$q_i^{n+1} = (1-\nu) q_i^n + \nu q_{i-1}^n. \tag{2.16}$$

We derive the amplification factor for von Neumann analysis using standard techniques (see, *e.g.*, Strikwerda [53]): using $I \equiv \sqrt{-1}$ and replacing q_i^n by $g^n e^{Ii\theta}$ we get:

$$\begin{aligned} g^{n+1} e^{Ii\theta} &= (1-\nu) g^n e^{Ii\theta} + \nu g^n e^{I(i-1)\theta} \\ g &= (1-\nu) + \nu e^{-I\theta} \end{aligned}$$

which implies that for constant velocity,

$$\left| g(\theta) \right|^2 = 1 - 4\nu(1-\nu) \sin^2 \frac{1}{2}\theta \tag{2.17}$$

This implies that $\left| g(\theta) \right|$ is bounded by 1 for $0 \le \nu \le 1$, which is the usual Courant-Friedrichs-Lewy (CFL) stability condition for constant flow.

2.3 Finite volume methods for scalar equations on a non-uniform grid

Consider Godunov's method on the simple advection method (2.3) on a non-uniform grid with notation as defined in Section 2.1. If we simply use method (2.13) with variable h_i, we have

$$Q_i^{n+1} = Q_i^n - \frac{uk}{h_i} \left(Q_i^n - Q_{i-1}^n \right). \tag{2.18}$$

This method is stable for $uk/h_{\min} \leq 1$, where $h_{\min} = \min_i h_i$. This stability requirement, which arises from the CFL condition that the numerical domain of dependence must contain the physical domain of dependence of the modeled equation, is common to all "3-point" schemes such as (2.18) which use only nearest-neighbor interactions at cell-interfaces.

However, even if a time step k is chosen for which the method is stable, the method may appear to have global error of $O(1)$. Several researchers ([28, 37]) have shown through so-called *supra-convergence* proofs that the nearest-neighbor flux method has $O(h)$ global error when a stable time-step is chosen; that is, when $uk/h_i \leq 1$ for all i.

Since this work is concerned rather with a large time-step method, which could produce very large cell CFL numbers uk/h_i, it is not helpful to consider supra-convergence methods further. Instead, we focus on calculating a numerical flux with sufficient accuracy so that global error is reduced to $O(h)$ (for Godunov's method) while stability is maintained for time steps such that $uk/h \leq 1$; that is, we want to show stability for an accurate method rather than accuracy for a stable method.

Since the stability of Godunov's method on a non-uniform grid is limited by the small numerical domain of dependence, it is logical to increase stability by extending the region used for flux calculation.

2.3.1 The h-box method in one dimension for the advection equation

One possible solution to the problem of non-uniform grids is to develop a method that in some way locally simulates a uniform grid. In this way the method would naturally be consistent with the uniform-grid method as the grid becomes uniform, and by extending the numerical domain of dependence the time-step limitation would be a function of a uniform grid size rather than the size of the smallest cell.

In finite volume methods, discretized values are interpreted as cell-averages of q, rather than as point-wise values. The conservative method of updating these cell-averages can be derived from (2.5): using the notation of Section 2.1,

$$Q_i^{n+1} = Q_i^n - \frac{k}{h_i} \left[F(Q_i^n, Q_{i+1}^n) - F(Q_{i-1}^n, Q_i^n) \right]. \tag{2.19}$$

We can see from this formulation that it makes sense to "uniformize" the grid at the *cell boundaries*, where the numerical flux is determined. Therefore we propose to calculate the fluxes $F^n_{i-1/2}$ at each cell interface based on reconstructed cell-averages in pseudo-cells of uniform size h formed on each side of the $i-1/2$-th interface, which we call "h-boxes". The conservative update then becomes

$$Q_i^{n+1} = Q_i^n - \frac{k}{h_i}\left[F(\overline{Q}^n_{i+1/2,L},\overline{Q}^n_{i+1/2,R}) - F(\overline{Q}^n_{i-1/2,L},\overline{Q}^n_{i-1/2,R})\right]. \tag{2.20}$$

In Figure 2.1, which illustrates some of the quantities we will be using, the continuous line refers to the underlying function $q(x,t_n)$, the solidly-demarcated boxes are the cell averages Q_i^n, and the boxes demarcated by dotted lines refer to the cell averages reconstructed to the left and right of a particular interface $x_{i-1/2}$ in the boxes of size h. The left interface of the left h-box is located at $x_{i-1/2,L}$; the right interface of the right h-box is located at $x_{i-1/2,R}$. The cell averages in the $i-1/2$-th set of h-boxes are denoted as $\overline{Q}^n_{i-1/2,L}$ and $\overline{Q}^n_{i-1/2,R}$. Methods for the limited reconstruction of $\hat{q}(x,t_n)$ and the h-box averages in one dimension are described in Section A.1.

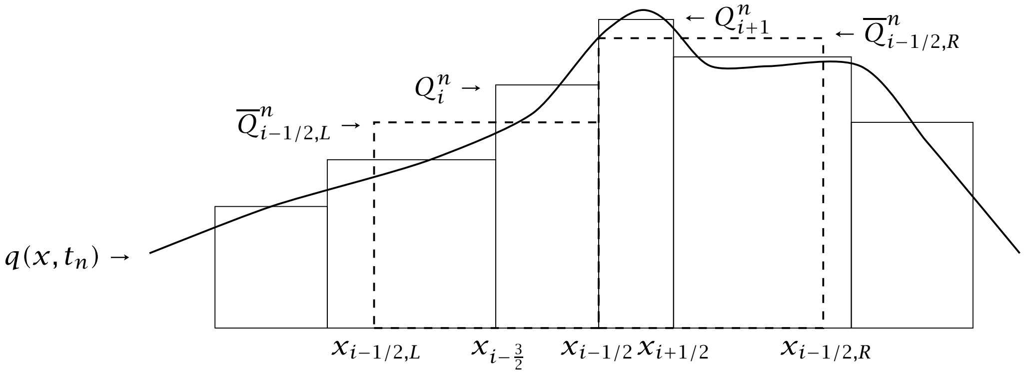


Figure 2.1: Illustration of Cell Average Reconstruction

Here is an outline of the proposed method in 1 dimension:

1. **Reconstruct the function $\hat{q}(x,t)$ from cell averages Q_i.** Depending on the desired accuracy of the solution, a piecewise constant, linear, or higher order representation of the function can be used. The reconstruction is assumed to be "sufficiently" limited (detailed below).

2. **Compute the cell-averages $\overline{Q}_{i-1/2,L}$ and $\overline{Q}_{i-1/2,R}$ in the h-boxes adjoining each cell interface $x_{i-1/2}$.** This is done using weighted sums of the reconstructed values on the underlying non-uniform grid.

3. **Determine flux $F(Q_L,Q_R)$ at $x_{i-1/2}$ based on the cell-averages $\overline{Q}_{i-1/2,L}$ and $\overline{Q}_{i-1/2,R}$.**

4. **Update Q_i^{n+1}.**

Among the advantages of this method are that:

- It simulates a uniform grid at each cell boundary: by extending the numerical domain of dependence at cell faces, the method is inherently consistent as the non-uniform grid becomes uniform with all cells of width h.

- On small cells, the h-boxes formed at each boundary will nearly overlap. As a result fluxes differ on the order of h_i, cancelling the effect of the h_i term in the denominator of (2.20). This effect of the increased domain of numerical dependence gives the necessary stability to increase the size of the time-step k beyond what would be possible in a nearest-neighbor method.

2.3.2 Godunov method using h-boxes on advection equation

We can illustrate some aspects of this method with its application to the Godunov method on (2.3). We assume that uniform h-box cell-averages are computed based on information from the non-uniform grid cells they overlap and that the piece-wise reconstruction in overlapped cells is sufficiently accurate, to some order yet to be specified.

The upwind Godunov method then reduces to cell-average updates in the form

$$Q_i^{n+1} = Q_i^n - \frac{uk}{h_i}\left(\overline{Q}_{i+1/2,L}^n - \overline{Q}_{i-1/2,L}^n\right). \tag{2.21}$$

It is important to note that the method has been factored into a form that contains the difference $\left(\overline{Q}_{i+1/2,L}^n - \overline{Q}_{i-1/2,L}^n\right)$. This difference nearly cancels out when $h_i \ll h$. To aid our description of this cancellation, we define a function to describe a more general form of the cell average of $q(x,t)$ over a region with left boundary ξ of width η at time t_n:

$$\hat{Q}(\xi,\eta) = \frac{1}{\eta}\int_{\xi}^{\xi+\eta} \hat{q}(x,t_n)\,dx. \tag{2.22}$$

The integral is determined from a sufficiently limited piece-wise reconstruction (PWR) $\hat{q}$ of $q(x,t_n)$, either piece-wise constant (PWC) or piece-wise linear (PWL), based on the numerical values Q_i^n. One crucial property of the reconstruction is that

$$\frac{1}{h_i}\int_{x_{i-1/2}}^{x_{i+1/2}} \hat{q}(x,t_n)\,dx = Q_i^n = \frac{1}{h_i}\int_{x_{i-1/2}}^{x_{i+1/2}} q(x,t_n)\,dx$$

which is satisfied trivially by PWC reconstruction and is satisfied by PWL reconstruction if it is viewed as a mean-zero linear correction to the piece-wise constant case. Note that for $\hat{Q}(x_{i-1/2},h_i)$ no special integration technique is required since the averaging region exactly corresponds with cell i. The other important property of the reconstruction is that it be *sufficiently limited.* A piece-wise constant reconstruction is automatically sufficiently limited. A piece-wise linear reconstruction is sufficiently limited if

$$\begin{aligned}\int \left|\hat{q}'(x,t_n)\right| \, dx &\le \sum_i \left|Q^n_{i+1} - Q^n_i\right| \\ &\le \int \left|q'(x,t_n)\right| \, dx;\end{aligned} \tag{2.23}$$

that is, if the total variation of $\hat{q}(x,t_n)$ is non-increasing. The total variation of the discrete solution is as an estimate of the total variation of the underlying function.

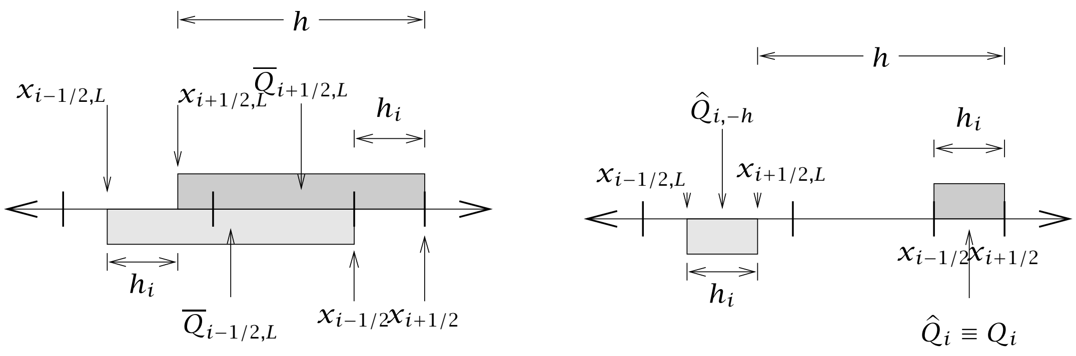
We make the abbreviations

$$\hat{Q}^n_{i,-h} \equiv \hat{Q}(x_{i-1/2} - h, h_i) \tag{2.24}$$

$$\hat{Q}^n_{i,+h} \equiv \hat{Q}(x_{i-1/2} + h, h_i), \tag{2.25}$$

which describe cell-averages over regions of width h_i centered around $x_i \pm h$. Note that on non-uniform grids these reconstructed cell-averages are taken over intervals may not be aligned with cell boundaries.

To continue: in Figure 2.2, we see that because reconstructed h-box averages use the same underlying information in the interval of overlap, the integrals $h\,\overline{Q}_{i+1/2,L}$ and $h\,\overline{Q}_{i-1/2,L}$ will be identical in those intervals and the difference of these quantities will create telescoping cancellation; *i.e.*,



(a) $(\overline{Q}_{i+1/2,L} - \overline{Q}_{i-1/2,L})$ on a small cell before cancellation

(b) After cancellation

Figure 2.2: Left h-box averages on the boundaries of a small cell. Difference of the averages is represented by placement above or below x-axis. Tick marks on x-axis indicate non-uniform cell boundaries. (a) Left h-boxes on small cell use overlapping information. (b) Equivalent difference after telescopic cancellation.

$$\overline{Q}_{i+1/2,L} = \frac{1}{h}\int_{x_{i+1/2}-h}^{x_{i+1/2}} \hat{q}(x,t_n)\,dx$$

$$\overline{Q}_{i-1/2,L} = \frac{1}{h} \int_{x_{i-1/2}-h}^{x_{i-1/2}} \hat{q}(x,t_n)\, dx$$

implies that

$$\begin{aligned}
\overline{Q}_{i+1/2,L} - \overline{Q}_{i-1/2,L} &= \frac{1}{h} \left[\int_{x_{i+1/2}-h}^{x_{i+1/2}} \hat{q}(x,t_n)\, dx - \int_{x_{i-1/2}-h}^{x_{i-1/2}} \hat{q}(x,t_n)\, dx \right] \\
&= \frac{1}{h} \left[\int_{x_{i+1/2}-h}^{x_{i-1/2}} \hat{q}(x,t_n)\, dx + \int_{x_{i-1/2}}^{x_{i+1/2}} \hat{q}(x,t_n)\, dx \right. \\
&\qquad \left. - \int_{x_{i-1/2}-h}^{x_{i+1/2}-h} \hat{q}(x,t_n)\, dx - \int_{x_{i+1/2}-h}^{x_{i-1/2}} \hat{q}(x,t_n)\, dx \right] \\
&= \frac{1}{h} \left[\int_{x_{i-1/2}}^{x_{i+1/2}} \hat{q}(x,t_n)\, dx - \int_{x_{i-1/2}-h}^{x_{i+1/2}-h} \hat{q}(x,t_n)\, dx \right].
\end{aligned}$$

Making substitutions using (2.22) and (2.24),

$$\begin{aligned}
\overline{Q}_{i+1/2,L} - \overline{Q}_{i-1/2,L} &= \frac{1}{h} \left(h_i \left[\hat{Q}(x_{i-1/2}, h_i) - \hat{Q}(x_{i-1/2} - h, h_i) \right] \right) \\
&= \frac{h_i}{h} \left[\hat{Q}_i - \hat{Q}(x_{i-1/2} - h, h_i) \right] \\
&= \frac{h_i}{h} \left(Q_i - \hat{Q}_{i,-h} \right).
\end{aligned} \tag{2.26}$$

The finite volume method update using h-boxes therefore becomes

$$\begin{aligned}
Q_i^{n+1} &= Q_i^n - \frac{uk}{h_i} \left(\overline{Q}_{i+1/2,L}^n - \overline{Q}_{i-1/2,L}^n \right) \\
&= Q_i^n - \frac{uk}{h_i} \left[\frac{h_i}{h} \left(Q_i^n - \hat{Q}_{i,-h}^n \right) \right] \\
&= Q_i^n - \frac{uk}{h} \left[Q_i^n - \hat{Q}_{i,-h}^n \right].
\end{aligned} \tag{2.27}$$

Note that the h_i term from the denominator cancels.

From (2.27) we can derive that stability holds in L^1, but it requires some care with regards to reconstruction.

Proof. From (2.27) we have for uniform advection that

$$Q_i^{n+1} = Q_i^n - \frac{uk}{h} \left[Q_i^n - \hat{Q}_{i,-h}^n \right].$$

We substitute the constant $\nu = uk/h$ to get

$$Q_i^{n+1} = (1 - \nu)\, Q_i^n + \nu\, \hat{Q}_{i,-h}^n.$$

Assuming $\sum_i h_i \left| Q_i^n \right|$ is bounded, we take the L^1 norm of each side:

$$\sum_i h_i \left| Q_i^{n+1} \right| = \sum_i h_i \left\{ \left| (1-\nu)\, Q_i^n + \nu\, \hat{Q}_{i,-h}^n \right| \right\}.$$

For $0 \leq \nu \leq 1$, the triangle inequality gives

$$\sum_i h_i \left| Q_i^{n+1} \right| \leq (1-\nu) \sum_i h_i \left| Q_i^n \right| + \nu \sum_i h_i \left| \hat{Q}_{i,-h}^n \right|.$$

Therefore

$$\sum_i h_i \left| Q_i^{n+1} \right| \leq \sum_i h_i \left| Q_i^n \right| + \nu \left\{ \sum_i h_i \left| \hat{Q}_{i,-h}^n \right| - \sum_i h_i \left| Q_i^n \right| \right\}$$

It is thus sufficient to show that

$$\sum_i h_i \left| \hat{Q}_{i,-h}^n \right| \leq \sum_i h_i \left| Q_i^n \right| \tag{2.28}$$

In the case of piece-wise constant reconstruction, equality holds trivially from (2.22). Piece-wise linear reconstruction is defined as a piece-wise constant reconstruction, $\hat{q}(x, t_n) = Q_i^n$ for $x_{i-1/2} \leq x \leq x_{i+1/2}$, plus a linear function $a\,(x - x_i)$ (therefore having mean zero) with a slope a that approximates the local gradient of $q(x, t_n)$ to $O(h)$. Assume that the slope of the linear function has been limited in each cell so that

$$\min(Q_{i-1}^n, Q_i^n, Q_{i+1}^n) \leq \hat{q}(x, t_n) \leq \max(Q_{i-1}^n, Q_i^n, Q_{i+1}^n)$$
$$\text{for} \quad x_{i-1/2} \leq x \leq x_{i+1/2}$$

If $Q_i^n > 0$ for all i, or if $Q_i^n < 0$ for all i, then the preceding assumptions are sufficient to guarantee that any partition of the piece-wise linear approximation $\hat{q}(x, t_n)$ into $\hat{Q}_{i,-h}$ satisfies property (2.28), and hence the theorem is proved.

However if there are both positive and negative values of Q_i^n, the reconstruction $\hat{q}(x, t_n)$ can cross 0 in a cell, as in Figure 2.3. In such a case it is possible for $\sum_i \left| \hat{Q}_{i,-h}^n \right|$ to exceed $\sum_i \left| Q_i^n \right|$. In the case illustrated, the three small cells i, $i+1$, and $i+2$, when shifted to the left by distance h, produce $h_i \left| \hat{Q}_{i,-h} \right| + h_{i+1} \left| \hat{Q}_{i+1,-h} \right| + h_{i+2} \left| \hat{Q}_{i+2,-h} \right| > h_{i-1} \left| Q_{i-1} \right|$.

If such a case exists, a different method of proof must be used: it is sufficient to show stability in L^1 if the total variation of the solution is non-increasing.

Therefore note that from the effective result of the h-box method on a non-uniform grid,

$$Q_i^{n+1} = Q_i^n - \nu \left[Q_i^n - \hat{Q}_{i,-h}^n \right],$$

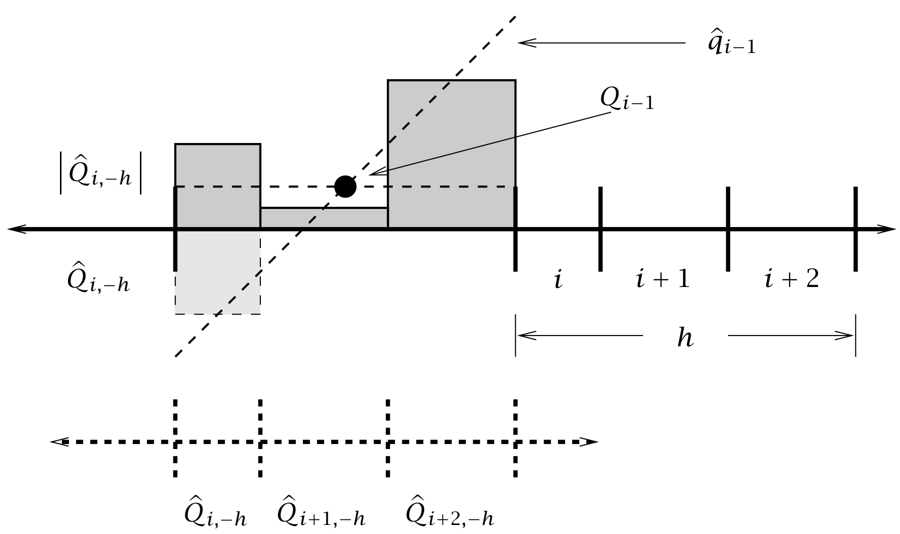


Figure 2.3: A demonstration of how $\sum_i h_i \left|\hat{Q}^n_{i,-h}\right|$ can exceed $\sum_i h_i \left|Q^n_i\right|$.

one can derive

$$Q^{n+1}_{i+1} - Q^{n+1}_i = (1-\nu)\ \left[Q^n_{i+1} - Q^n_i\right] + \nu\ \left[\hat{Q}^n_{i+1,-h} - \hat{Q}^n_{i,-h}\right].$$

Taking the absolute value and summing over i, the total variation at time t_{n+1}, which is independent of the sign of Q_i, can then be shown to be

$$\sum_i \left|Q^{n+1}_{i+1} - Q^{n+1}_i\right| \leq (1-\nu)\ \sum_i \left|Q^n_{i+1} - Q^n_i\right| + \nu\ \sum_i \left|\hat{Q}^n_{i+1,-h} - \hat{Q}^n_{i,-h}\right|$$

or, more succinctly,

$$\mathrm{TV}(Q^{n+1}_i) \leq (1-\nu)\ \mathrm{TV}(Q^n_i) + \nu\ \sum_i \left|\hat{Q}^n_{i+1,-h} - \hat{Q}^n_{i,-h}\right|.$$

It is thus sufficient to show that

$$\sum_i \left|\hat{Q}^n_{i+1,-h} - \hat{Q}^n_{i,-h}\right| \leq \mathrm{TV}(Q^n_i). \tag{2.29}$$

But (2.29) holds if we make the additional requirement on $\hat{q}(x,t_n)$ that $\mathrm{TV}(\hat{q}(x,t_n)) \leq \mathrm{TV}(Q^n_i)$. This requirement may be met, for example, through the use of min-mod piecewise slope limiting on $\hat{q}$. ■

Since ν is not a function of Q_i and $Q^n_i - \hat{Q}^n_{i,-h}$ should not be affected by the addition of a constant, the TVD argument used above is excessively restrictive in the case of uniform advection, and in practice, no instability is seen in uniform advection solutions when Q_i straddles 0.

Not addressed above is the issue of errors introduced by approximation of $\hat{Q}^n_{i,-h}$. I will discuss three considerations:

- Maximum local error.
- Local truncation error in L^∞ for small cells.
- Global error resulting from effect of small cells.
- Local error is reduced through use of piece-wise linear reconstruction.

The maximum local error in the value $\hat{Q}^n_{i,-h}$ introduced by use of piece-wise constant reconstruction will be $O(h)$, as illustrated in Figure 2.4, and will reach that magnitude only when $h_i \ll h$. The actual cause of the error is that when h_i is small, $\hat{Q}^n_{i,-h}$ can "clip" out a section of Q_{i-1} in a region where the piece-wise constant approximation is least accurate. We find that

$$\max\left(\left|\hat{Q}^{\mathrm{PWC}}_{i,\pm h} - \hat{Q}^{\mathrm{PWL}}_{i,\pm h}\right|\right) \le |q_x(x_i - h, t_n)| \; \frac{h}{2}$$

This local error in the approximation is passed through the numerical method, giving a local truncation error (LTE) of $O(h)$ in the L^∞ norm. That error, however, is *strictly* local, in time as well as space: since that error can only occur where a small cell adjoins an $\tilde{h}$-sized cell, such errors only occur in *well-separated* small cells. Therefore in successive time-steps, the $O(h)$ approximation error in the small cell's solution will advect into a neighboring larger cell, where it will be averaged into a much larger region. In this way, the approximation error does *not* build up over time to an $O(1)$ global error, as one might expect from the style of error analysis used on uniform grids. However, the result of piece-wise constant reconstruction will be apparent as spurious bumps of size $q_x \; h/2$ will appear in small cells in every time-step. These bumps can create a large enough error constant in the L^∞ norm that the error may not appear to converge at a rate of $O(h)$ at all. In L^1, though, the effect is that the error constant is increased, but the rate of convergence remains $O(h)$. These effects are seen in the numerical results in Chapter 3.

Using monotone piece-wise linear reconstruction provides one higher order of accuracy in approximating $\hat{Q}^n_{i-1/2,\pm h}$ and hence the Godunov method using h-boxes on non-uniform grids will be $O(h)$ in both L^1 and L^∞.

One may note that stability should not be affected by accuracy of reconstruction as long as reconstructions are with the ranges of neighboring cell values and do not increase the total variation of $q(x,t)$. The information propagated using the h-box method, though possibly less accurate than it would be on a uniform grid, nevertheless uses the identical numerical domain of dependence.

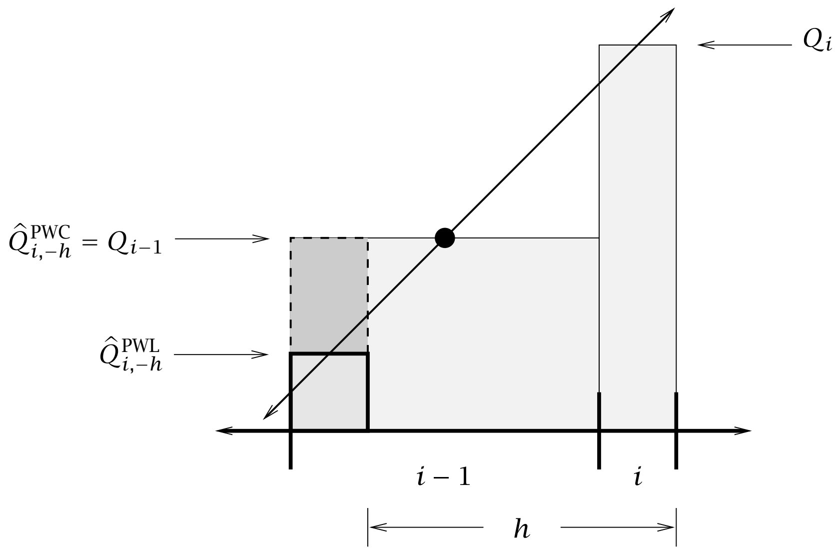


Figure 2.4: the maximum local error from piece-wise constant approximation is $\left|\hat{Q}^{\text{PWC}} - \hat{Q}^{\text{PWL}}\right|$. The oblique line denotes a linear $q(x, t_n)$.

2.4 Godunov's method for Burgers' Equation

We now compare implementation of Godunov methods for solution of (2.4) on both uniform and non-uniform grids, and examine their accuracy and stability. We first discuss the method used for numerical solution of the Riemann problem at cell interfaces.

2.4.1 Roe's approximate Riemann solver

Godunov's method is based on solution of the Riemann Problem between piece-wise constant cell averages on either side of a cell interface:

$$\boldsymbol{q}(x,0) = \begin{cases} \boldsymbol{q}_L & x < 0, \\ \boldsymbol{q}_R & x > 0. \end{cases} \tag{2.30}$$

The resulting intermediate state $\boldsymbol{q}(0,t)$, which is constant over the time-step in the case of the Riemann problem, is used to calculate a flux. For a scalar problem such as Burgers' equation, the intermediate flux is easy to determine:

$$\boldsymbol{q}(x,0) = \begin{cases} 0 & \text{if} \quad q_L < 0 < q_R, \quad \text{otherwise,} \\ q_L^2/2 & \text{if} \quad (q_L + q_R)/2 > 0 \\ q_R^2/2 & \text{if} \quad (q_L + q_R)/2 < 0. \end{cases} \tag{2.31}$$

In other words, the intermediate flux is the *upwind* flux except when a rarefaction straddles the interface.

For general non-linear hyperbolic problems, however, it can be expensive to solve the Riemann problem, even in one dimension. In such cases it is standard to use a technique developed by Roe [44] to approximate the solution to the Riemann problem. This solution uses a locally linearized version of the PDE to be solved,

$$\boldsymbol{q}_t + \hat{\mathbf{A}}(\boldsymbol{q}_L, \boldsymbol{q}_R)\ \boldsymbol{q}_x = 0.$$

This solution process is described in more detail in [34], but the essential points to note are the conditions Roe suggested be imposed on $\hat{\mathbf{A}}$:

$$\hat{\mathbf{A}}(\boldsymbol{q}_L, \boldsymbol{q}_R)\ \ (\boldsymbol{q}_R - \boldsymbol{q}_L) = \boldsymbol{f}(\boldsymbol{q}_R) - \boldsymbol{f}(\boldsymbol{q}_L), \tag{i}$$

$$\hat{\mathbf{A}}(\boldsymbol{q}_L, \boldsymbol{q}_R) \quad \text{is diagonalizable with real eigenvalues,} \tag{ii}$$

$$\hat{\mathbf{A}}(\boldsymbol{q}_L, \boldsymbol{q}_R) \to \boldsymbol{f}'(\overline{\boldsymbol{q}}) \quad \text{smoothly as} \quad \boldsymbol{q}_L, \boldsymbol{q}_R \to \overline{\boldsymbol{q}}. \tag{iii}$$

We define a splitting for the Roe matrix,

$$\hat{\mathbf{A}} = \hat{\mathbf{R}}\ \hat{\mathbf{\Lambda}}\ \hat{\mathbf{L}}^T = \hat{\mathbf{R}}\ \hat{\mathbf{\Lambda}}^+\ \hat{\mathbf{L}}^T + \hat{\mathbf{R}}\ \hat{\mathbf{\Lambda}}^-\ \hat{\mathbf{L}}^T = \hat{\mathbf{A}}^+ + \hat{\mathbf{A}}^- \tag{2.32}$$

where the split-eigenvalue diagonal matrices are

$$\hat{\mathbf{\Lambda}}^+ = \mathrm{diag}\{\max(\hat{\lambda}^p, 0)\}, \qquad \hat{\mathbf{\Lambda}}^- = \mathrm{diag}\{\min(\hat{\lambda}^p, 0)\},$$

the arrays of left- and right-eigenvectors are

$$\hat{\mathbf{L}}^T = \hat{\mathbf{R}}^{-1},$$

and the individual eigenvectors are denoted as

$$\hat{\boldsymbol{r}}^p = \hat{\mathbf{R}}\ \boldsymbol{e}_p \qquad \hat{\boldsymbol{l}}^p = \hat{\mathbf{L}}\ \boldsymbol{e}_p.$$

Using the matrix splitting, one may define the numerical flux in two equivalent ways:

$$\begin{aligned} \boldsymbol{F}(\boldsymbol{Q}_L, \boldsymbol{Q}_R) &= \boldsymbol{f}(\boldsymbol{Q}_R) - \mathbf{A}^+\ (\boldsymbol{Q}_R - \boldsymbol{Q}_L) \\ &= \boldsymbol{f}(\boldsymbol{Q}_L) + \mathbf{A}^-\ (\boldsymbol{Q}_R - \boldsymbol{Q}_L)\,. \end{aligned} \tag{2.33}$$

In the case of a scalar conservation law, the Roe wave-speed can be calculated uniquely to be

$$\begin{aligned} \hat{a} &= \hat{A}(Q_L, Q_R) \\ &= \frac{f(Q_R) - f(Q_L)}{Q_R - Q_L}. \end{aligned} \tag{2.34}$$

Note that in this case the numerical flux function reduces identically to the upwind flux:

$$F(Q_L, Q_R) = \begin{cases} f(Q_L) & \text{where} \quad \hat{a} > 0, \\ f(Q_R) & \text{where} \quad \hat{a} < 0. \end{cases}$$

and if we assume that f' is everywhere positive this reduces further to

$$F(Q_L, Q_R) = f(Q_L).$$

For the scalar case, the Roe solution calculates a speed for the propagation of the discontinuity between Q_R and Q_L which is exactly the same as the shock speed when the Riemann Problem gives rise to a shock. However, rarefactions are also treated as shocks, and so the approximate Roe solution may violate the entropy condition when $f'(Q_L) < 0 < f'(Q_R)$. Several methods exist to make entropy-satisfying corrections for first-order methods (*e.g.*, Harten and Hyman [22]). We note, however, that when *higher-order* corrections are used on uniform grids, the Roe method produces good results that do not deviate noticeably from physical expectations. The higher-order methods used in this research do not contain any entropy fixes.

For scalar equations, we denote the speed $\hat{a}$ as the derivative f' at the *Roe intermediate state* Q^*, where Q^* is a smooth symmetric function of the left and right states:

$$\hat{a}(Q_L, Q_R) = f'\left(Q^*(Q_L, Q_R)\right).$$

In the case of Burgers' equation, it can be shown that

$$Q^* = \frac{Q_L + Q_R}{2}.$$

2.4.2 Godunov's method for Burgers' equation on a uniform grid

For Godunov's method (on both uniform and non-uniform grids) we use the numerical flux function

$$F(Q_L, Q_R) = \begin{cases} Q_L^2/2 & \text{for } (Q_L + Q_R)/2 > 0 \\ Q_R^2/2 & \text{for } (Q_L + Q_R)/2 < 0 \end{cases}. \tag{2.35}$$

This is equivalent to the approximate Roe flux derived in Section 2.4.1 without corrections for the transonic case.

We follow the classical result here to show stability and accuracy of the upwind scheme more generally for non-linear scalar hyperbolic equations

$$q_t + f(q)_x = 0.$$

Assume that $f'(q) > 0$ for all x so that the upwind flux is $F(Q_L, Q_R) = f(Q_L)$. Then the Godunov upwind method can be written as

$$Q_i^{n+1} = Q_i^n - \frac{k}{h}\ \left(f(Q_i) - f(Q_{i-1})\right). \tag{2.36}$$

First order accuracy results from straightforward Taylor series expansion—we are more concerned with showing stability.

Using the intermediate value theorem,

$$Q_i^{n+1} = Q_i^n - \frac{k}{h}\ f'(Q_i^*)\ \left(Q_i - Q_{i-1}\right),$$

where Q_i^* is some value between Q_i and Q_{i-1}. Note that the value Q^*, for scalar equations, can be determined using the Roe approximate solution. Define

$$\nu_i = \frac{k\, f'(Q_i^*)}{h}.$$

Then the Godunov update for (2.2) can be written as

$$Q_i^{n+1} = Q_i - \nu_i \left(Q_i^n - Q_{i-1}^n\right).$$

We can now demonstrate stability of the upwind scheme using a TVD proof. Assume that $\sum_i |Q_i| < \infty$. To find the total variation at time t_{n+1} we take the difference of the Godunov updates in cells i and $i-1$:

$$\begin{aligned} Q_i^{n+1} - Q_{i-1}^{n+1} &= Q_i^n - \nu_i\ \left(Q_i^n - Q_{i-1}^n\right) - Q_{i-1}^n + \nu_{i-1}\left(Q_{i-1}^n - Q_{i-2}^n\right). \\ &= (1-\nu_i)\ \left(Q_i^n - Q_{i-1}^n\right) + \nu_{i-1}\left(Q_{i-1}^n - Q_{i-2}^n\right). \end{aligned}$$

If we impose the standard CFL constraint that $\nu_i \leq 1$ for all i, then the coefficients $(1-\nu_i)$ and ν_{i-1} are positive. We take the absolute value of both sides, apply the triangle inequality, and sum over all i:

$$\sum_i \left|Q_i^{n+1} - Q_{i-1}^{n+1}\right| = \sum_i (1-\nu_i)\ \left|Q_i^n - Q_{i-1}^n\right| + \sum_i \nu_{i-1}\left|Q_{i-1}^n - Q_{i-2}^n\right|.$$

Replacing $i-1$ by i in the last sum,

$$\begin{aligned} \sum_i \left|Q_i^{n+1} - Q_{i-1}^{n+1}\right| &= \sum_i (1-\nu_i)\ \left|Q_i^n - Q_{i-1}^n\right| + \sum_i \nu_i\left|Q_i^n - Q_{i-1}^n\right| \\ &= \sum_i \left\{(1-\nu_i)\left|Q_i^n - Q_{i-1}^n\right| + \nu_i\left|Q_i^n - Q_{i-1}^n\right|\right\} \\ &= \sum_i \left|Q_i^n - Q_{i-1}^n\right|. \end{aligned}$$

2.4.3 Stability of h-box-Godunov for a scalar hyperbolic equation

Assume that we are trying to solve

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

and that $f'(q) \geq 0$ for all q in the region. Then the Godunov upwind flux on a face is defined as

$$F(Q^L, Q^R) = f(Q^L).$$

Using reconstructed values of Q in h-boxes on each face, the numerical update takes the form (in which values at time-step n do not have superscripts):

$$Q_i^{n+1} = Q_i - \frac{k}{h_i} \left[f(\overline{Q}_{i+1/2,L}) - f(\overline{Q}_{i-1/2,L}) \right]. \tag{2.37}$$

Note that this reduces to the uniform grid method when all cells are of uniform size $h_i = h$.

It is not easy to prove stability of the h-box method on a non-uniform grid. From the intermediate value theorem for scalar functions, we have that

$$f(\overline{Q}_{i+1/2,L}) - f(\overline{Q}_{i-1/2,L}) = f'(Q_i^*) \left(\overline{Q}_{i+1/2,L} - \overline{Q}_{i-1/2,L} \right),$$

where Q_i^* is a value between $\overline{Q}_{i-1/2,L}$ and $\overline{Q}_{i+1/2,L}$. As we noted in Section 2.4.1, that intermediate value, in the case of a scalar hyperbolic conservation law, is identical to the Roe intermediate state between the two left-h-box values. Expanding $\overline{Q}_{i+1/2,L}$ in terms of $\overline{Q}_{i-1/2,L}$ as in the case of advection,

$$\begin{aligned} \overline{Q}_{i+1/2,L} &= \overline{Q}_{i-1/2,L} + \left(\overline{Q}_{i+1/2,L} - \overline{Q}_{i-1/2,L} \right) \\ &= \overline{Q}_{i-1/2,L} + \frac{h_i}{h} \left(Q_i - \hat{Q}_{i,-h} \right), \end{aligned}$$

we then obtain

$$\overline{Q}_{i+1/2,L} - \overline{Q}_{i-1/2,L} = \frac{h_i}{h} \left(Q_i - \hat{Q}_{i,-h} \right),$$

because of telescopic cancellation; therefore,

$$\begin{aligned} Q_i^{n+1} &= Q_i^n - \frac{k}{h_i} f'(Q^*) \left[\frac{h_i}{h} \left(Q_i - \hat{Q}_{i,-h} \right) \right], \\ &= Q_i^n - \frac{k}{h} f'(Q^*) \left(Q_i - \hat{Q}_{i,-h} \right). \end{aligned} \tag{2.38}$$

$O(h)$ accuracy follows directly from the approximation of the partial differential equation as it did on a uniform grid. Though we cannot use the same TVD proof methodology as in the uniform grid case (because $(Q_i - Q_{i-1}) \neq (Q_i - \hat{Q}_{i,-h})$), the cancellation of the small quantity h_i from the denominator gives us hope that the h-box method will be stable. We note that for $k\ f'(Q^*)/h \leq 1$, Q_i^{n+1} is a convex combination of uniformly-separated reconstructed values, and the wave-speed ($f'(Q_i^*)$) is derived using overlapping values from the upwind side of the cell, out to distance h from the left cell wall. It may be possible to show a TVD-like proof using the difference $Q_i^{n+1} - \hat{Q}_{i,-h}^n$, and this is being investigated. However, our numerical results (Chapter 3) show that the method is indeed stable when time-step k is chosen so that $k\ \max_i |f'(Q_i)|/h \leq 1$, when $\hat{q}$ is reconstructed using min-mod piece-wise linear slope-limiting.

It might appear unnecessary to use the Roe Riemann solver approximation to define $f'(Q_i^*)$, since the intermediate value theorem is sufficient. However, it is useful to observe that the conservative upwind flux-difference correction to a cell can be found by solution of a Riemann problem using the two upwind cell values, which in the case of the h-box method are the reconstructions in the two upwind h-boxes. This observation aids the design of transverse corrections that will be necessary for stability and second-order accuracy in the two-dimensional h-box method.

Chapter 3

1D COMPUTATIONAL RESULTS

In all one-dimensional tests of the h-box method, periodic boundary conditions are assumed. Some of the results below have been seen previously in [9].

3.1 Flux approximation vs. cell average reconstruction

In examples shown below, the graphs containing numerical solutions to Burgers' equation and the Advection equation depict solutions on grids with $h = 1/40$. Graphs showing $\log(\|GE\|_1)$ vs. $\log h$ use ten h values: $h = 1/20, 1/40, \ldots, 1/200$.

3.1.1 First-order flux approximation and numerical results for smooth solutions

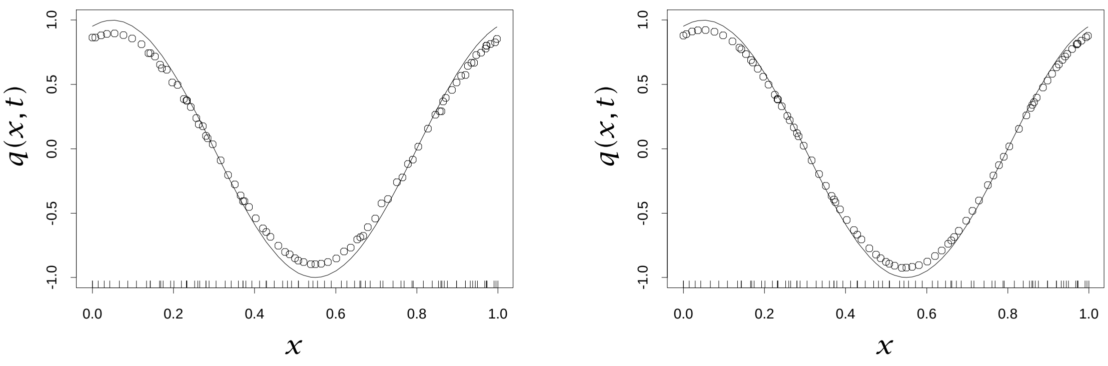
As discussed in Section 2.3.2, we expect that there should be a difference in numerical convergence rates for the h-box extensions of Godunov's method and the Lax-Wendroff method depending on whether piece-wise constant (PWC) or piece-wise linear (PWL) reconstruction is used to interpolate h-box values $\overline{Q}_{i-1/2,L}$ and $\overline{Q}_{i-1/2,R}$. We find the following results for smooth solutions to (2.3) and (2.4). Figures 3.1 and 3.2 show comparisons of the numerical solutions using a Godunov flux in (2.3) and (2.4) with PWC and PWL reconstruction on an irregular grid. Figure 3.11 is a comparison of the log-log plot of the errors in $\|\cdot\|_1$ versus the step-size for the advection equation experiment. Figure 3.13 shows a similar comparison for the Burgers' equation experiment.

In each of these cases, having smooth solutions, we see that while we might expect $O(1)$ error as h decreases, we actually obtain error convergence approaching $O(h)$.

It is easily noted by comparing the log-log plots of the 1st order flux solutions in Figures Figure 3.11 and Figure 3.13, that, though the log-log plots have much straighter lines (denoting a more regular decrease of error with smaller h) the error slopes using piecewise quadratic reconstruction of the primitive function (PWL) are not appreciably steeper than the error slopes of the method with PWC. This indicates that in the PWC experiment, the order of accuracy is limited slightly more by the reconstruction approximation than by the flux approximation, and in the PWL experiment, it is limited by the Godunov Flux approximation.

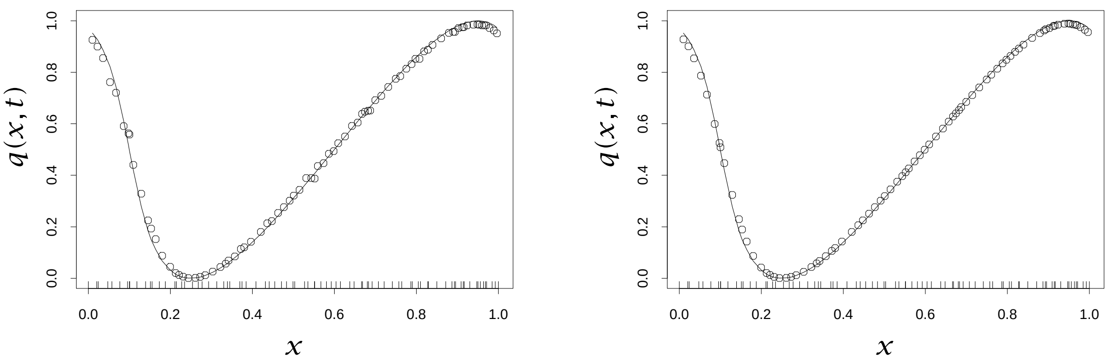
3.1.2 Second order fluxes and numerical results for smooth solutions

The second-order method described in Section 4.2.3 (considered in a one-dimensional sense) is generalized for non-uniform grids by using h-box values instead of neighboring



(a) PWC Reconstruction (b) PWL Reconstruction

Figure 3.1: Comparison of 1D numerical solutions of (2.3), using h-box + Godunov fluxes, for a smooth exact solution.

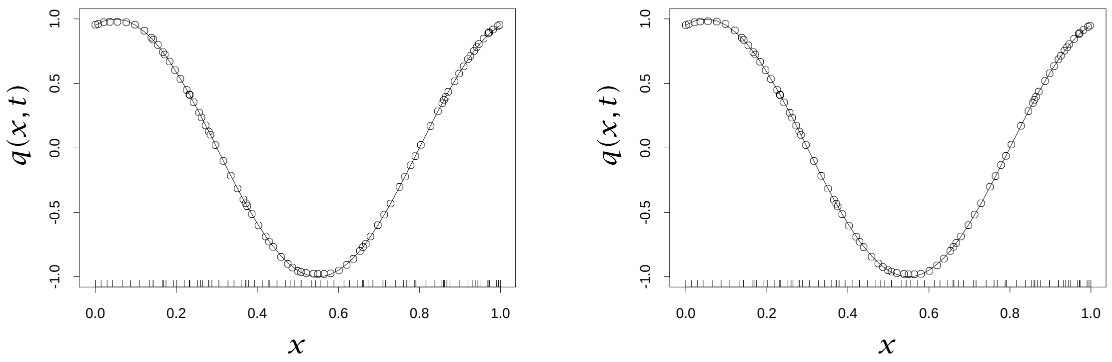


(a) PWC Reconstruction (b) PWL Reconstruction

Figure 3.2: Comparison of 1D numerical solutions of (2.4), using h-box + Godunov fluxes, for a smooth exact solution.

cell values. In this section we examine the results of using upwind-limited Lax-Wendroff fluxes on the same non-uniform grid, and using the same set of solutions, as in the previous section. The examples shown here use a smooth TVD limiter, $\phi(r) = (r + |r|)/(1 + |r|)$, first introduced by van Leer in 1974 (see [54, 56]). This limiter is referred to subsequently as the VL74 limiter.

In figures Figure 3.11 and Figure 3.13, we see that the piece-wise quadratic reconstruction (using piece-wise cubic primitive function reconstruction, see Section A.1) only slightly improves the order of accuracy for smooth solutions when compared to piece-wise linear reconstruction.



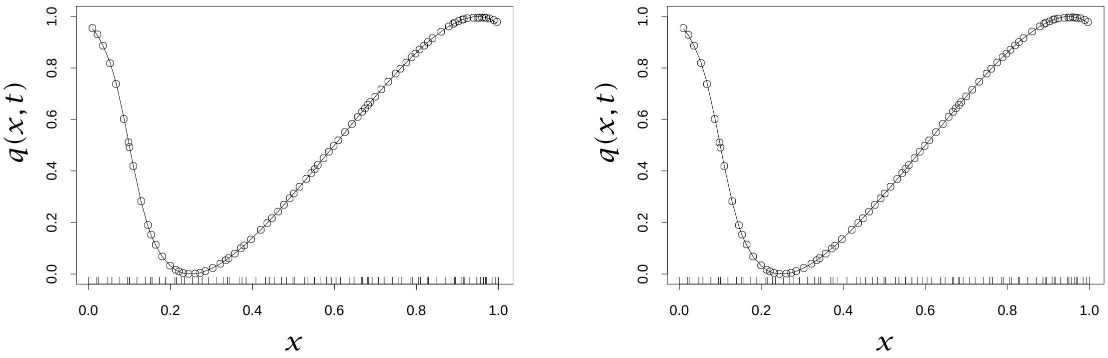
(a) PWL Reconstruction (b) PWQ Reconstruction

Figure 3.3: Comparison of 1D numerical solutions of (2.3), using h-box + Lax-Wendroff fluxes + VL74 limiter, for a smooth exact solution.

For an additional comparison, we present numerical solutions in which the only difference is the form of reconstruction. A second-order technique is used here: the primitive function $W = \int Q \, dx$ is reconstructed using cubic splines under tension (see Section A.1).

3.1.3 h-boxes + Godunov's method and numerical results with discontinuous solutions

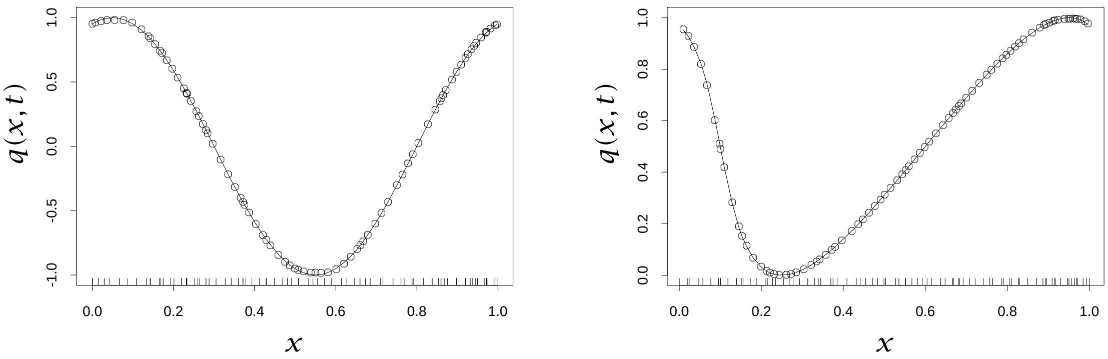
As in previous sections, we note in Figures 3.6, 3.7, 3.12 and 3.14 that use of piece-wise constant reconstruction leads to bumpy solutions. When PWL reconstruction is used, however, we see that the method has good shock resolution without extraneous bumps.



(a) PWL Reconstruction

(b) PWQ Reconstruction

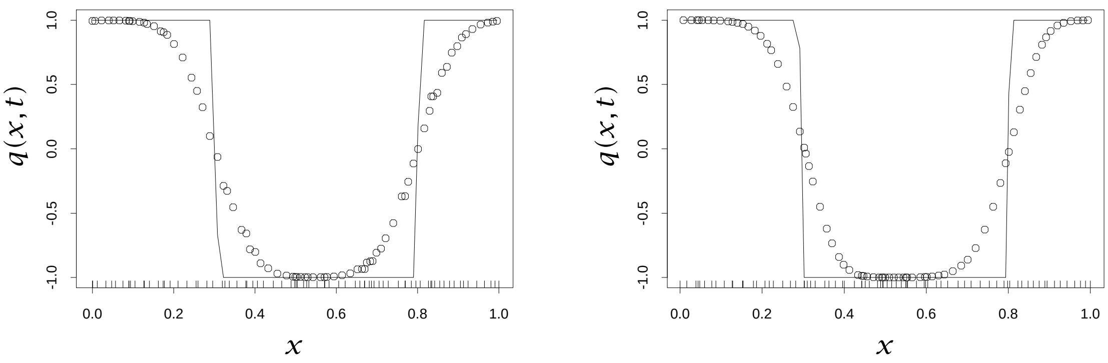
Figure 3.4: Comparison of 1D numerical solutions of (2.4), using h-box + Lax-Wendroff fluxes + VL74 limiter, for a smooth exact solution.



(a) CS Reconstruction on (2.3)

(b) CS Reconstruction on (2.4)

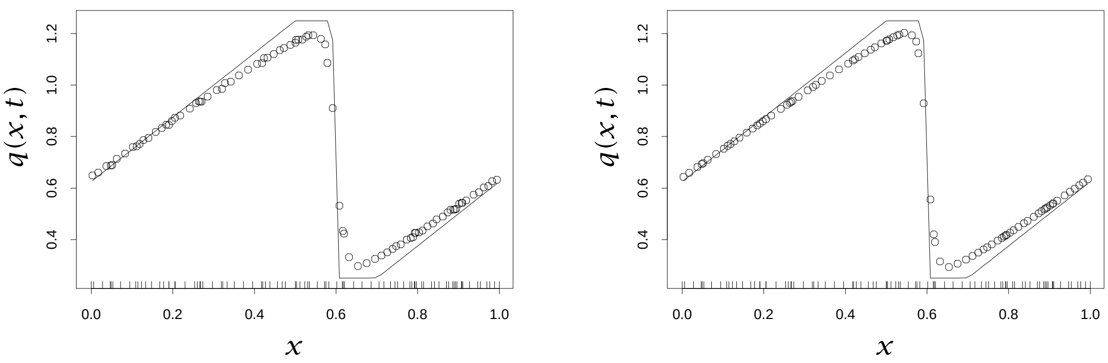
Figure 3.5: Numerical solutions with Lax-Wendroff Fluxes, using cubic spline primitive function reconstruction, on (a) equation (2.3) and (b) equation (2.4), for a smooth exact solution.



(a) PWC Reconstruction

(b) PWL Reconstruction

Figure 3.6: Comparison of 1D numerical solutions of (2.3), using h-box + Godunov fluxes, for a discontinuous exact solution.



(a) PWC Reconstruction

(b) PWL Reconstruction

Figure 3.7: Comparison of 1D numerical solutions of (2.4), using h-box + Godunov fluxes, for a discontinuous exact solution.

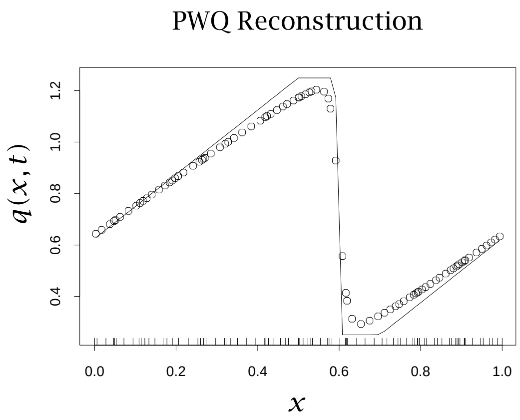


Figure 3.8: Numerical solution of (2.4), using h-box + Godunov flux + piece-wise quadratic reconstruction, for a discontinuous exact solution.

3.1.4 Second-order flux approximation and numerical results with discontinuous solutions

In Figures 3.9, 3.10, 3.12 and 3.14 we compare results using limited Lax-Wendroff fluxes. We note that solutions in which h-box values were reconstructed with piece-wise linear approximations have similar convergence rates to those using piece-wise quadratic reconstruction. We conclude that although piece-wise linear reconstruction is helpful for smoothing out visible inaccuracies in first-order methods, it is not essential to use piece-wise quadratic or higher-order reconstruction when computing a second-order solution.

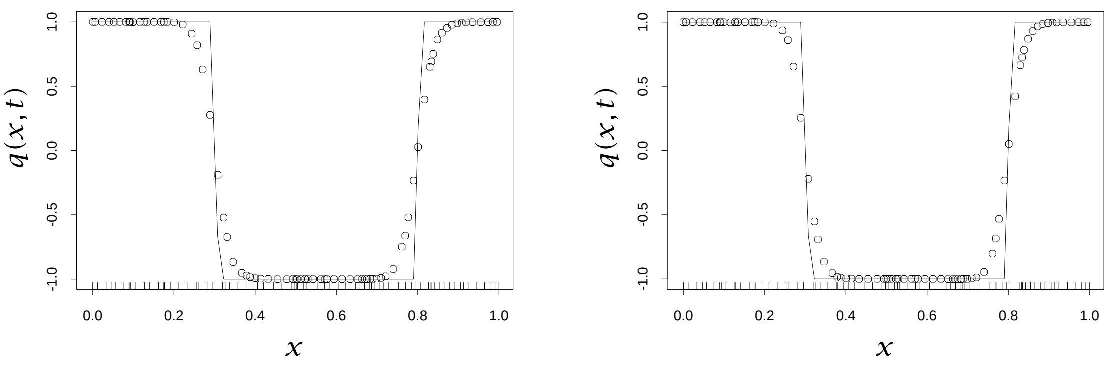
3.2 Numerical results for a non-linear hyperbolic system

The h-box method performs quite well for systems as well as for scalar equations. In this section we examine the behavior of the first and second-order h-box methods on the isothermal gas equations:

$$\begin{pmatrix} \rho \\ \rho u \end{pmatrix}_t + \begin{pmatrix} \rho u \\ (\rho u)^2/\rho + a^2\,\rho \end{pmatrix}_x = 0 \tag{3.1}$$

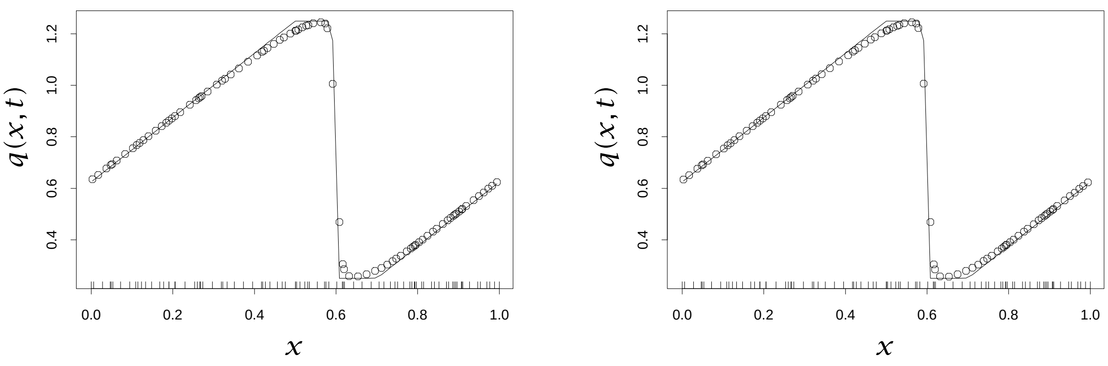
For our approximate Riemann solver we use the derivative matrix $\boldsymbol{f}' = \mathbf{A}$ evaluated at the Roe intermediate state:

$$\boldsymbol{Q}^*(\boldsymbol{Q}^L, \boldsymbol{Q}^R) = \begin{pmatrix} Q_1^* \\ Q_2^* \end{pmatrix} = \begin{pmatrix} \rho_L^{1/2} + \rho_R^{1/2} \\ (\rho_L u_L)/\rho_L^{1/2} + (\rho_R u_R)/\rho_R^{1/2} \end{pmatrix}. \tag{3.2}$$



(a) PWL Reconstruction (b) PWQ Reconstruction

Figure 3.9: Comparison of 1D Numerical solutions with LW Fluxes to (2.3), for a discontinuous exact solution.



(a) PWL Reconstruction (b) PWQ Reconstruction

Figure 3.10: Comparison of 1D Numerical solutions with LW Fluxes to (2.4), for a discontinuous exact solution.

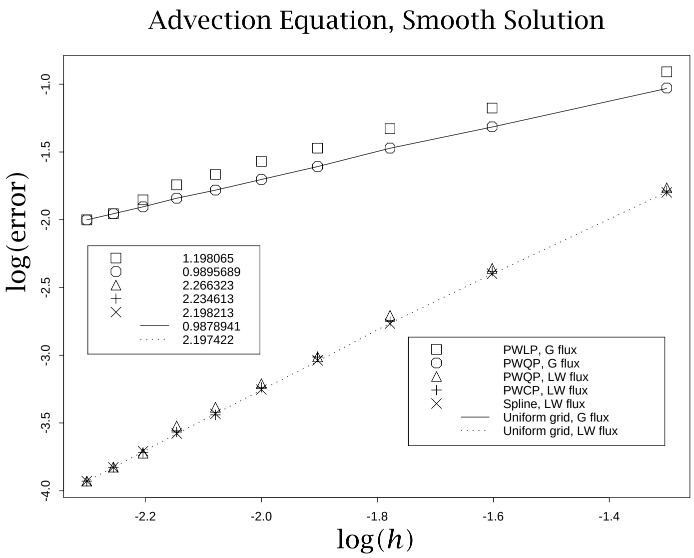


Figure 3.11: Comparison of 1-norm errors for solutions to (2.3) with Godunov Flux and LW Flux approximations and various primitive function reconstructions, for a case having a smooth solution.

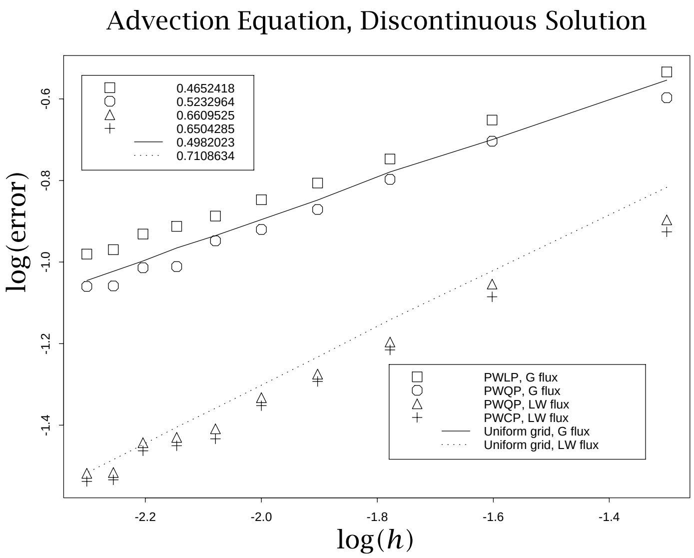


Figure 3.12: Comparison of 1-norm errors for solutions to (2.3) with Godunov Flux and LW Flux approximations and various primitive function reconstructions, for a case having a discontinuous solution.

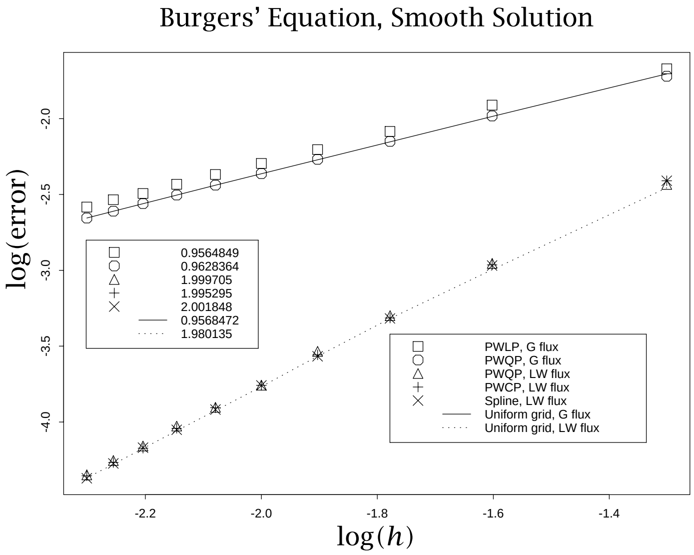


Figure 3.13: Comparison of 1-norm errors for solutions to (2.4) with Godunov Flux and LW Flux approximations and various primitive function reconstructions, for a case having a smooth solution.

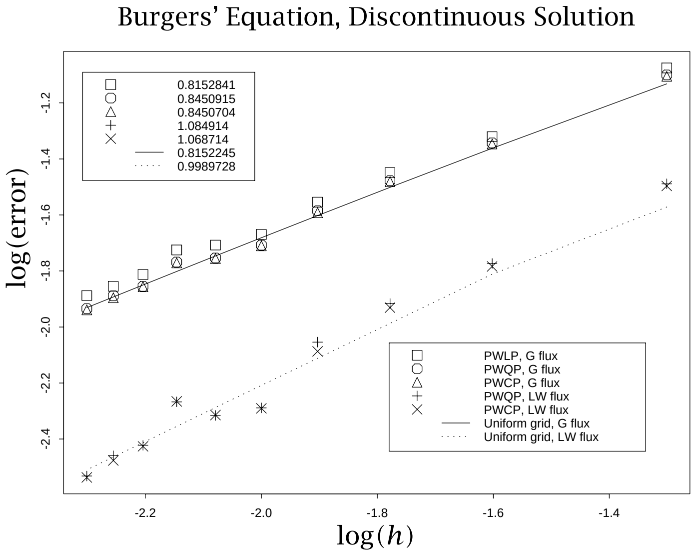


Figure 3.14: Comparison of 1-norm errors for solutions to (2.4) with Godunov Flux and LW Flux approximations and various primitive function reconstructions, for a case having a discontinuous solution.

This gives the matrix

$$\hat{\mathbf{A}}\left(\boldsymbol{Q}^L, \boldsymbol{Q}^R\right) = \begin{pmatrix} 0 & 1 \\ a^2 - (Q_2^*)^2/(Q_1^*)^2 & 2Q_2^*/Q_1^* \end{pmatrix}. \tag{3.3}$$

We see in Figures 3.15, 3.16, 3.17, 3.18, 3.19, and 3.20 that the h-box method compares very well to the uniform grid method. Using piece-wise linear reconstruction, the numerical results show that there is no effective difference between uniform and non-uniform grid solutions, even in a systems case which includes rarefactions and shocks. As in the scalar case, accuracy is reduced, even away from the shock, when a discontinuity is present. Since we are using a Godunov Riemann solver approach, reduction of accuracy away from the shock is present in the non-uniform solution to the same extent as in a uniform solution. This phenomenon has been noted elsewhere by Quirk [43].

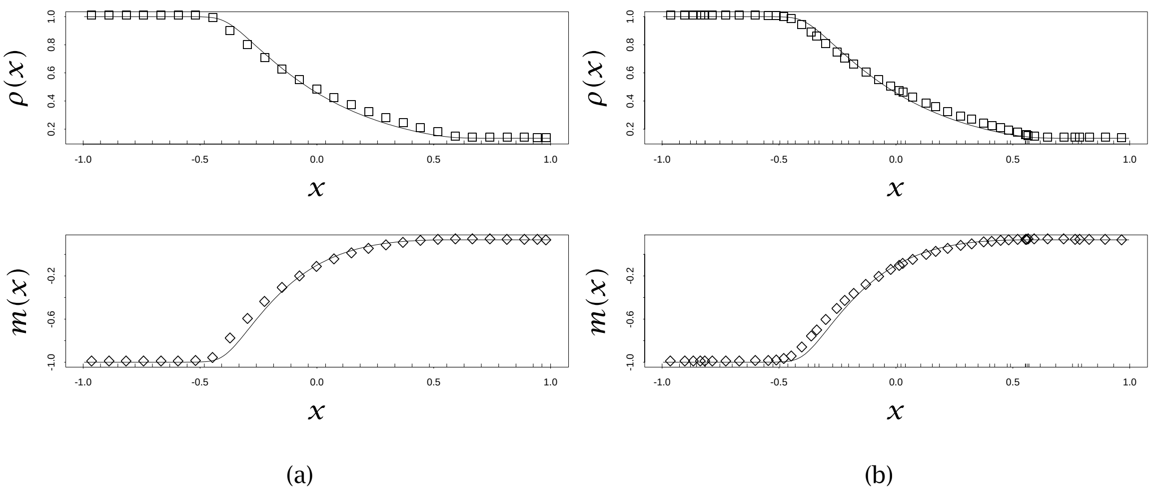
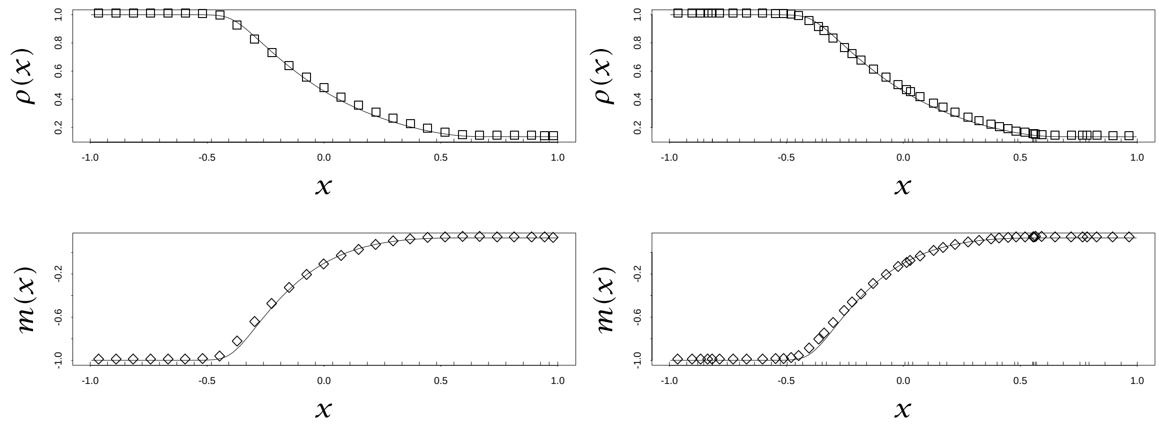
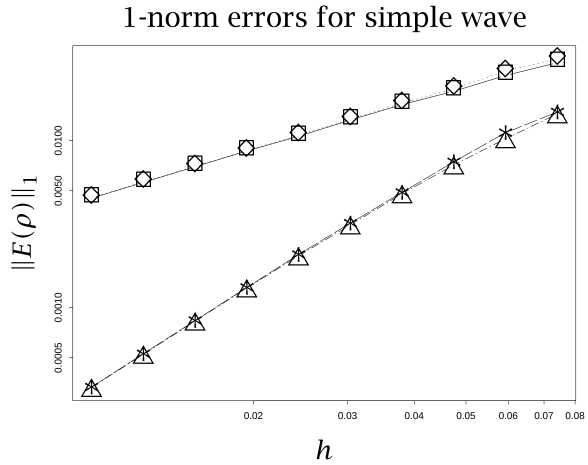


Figure 3.15: h-box method on (a) regular grid and (b) irregular grid using Godunov flux, for a 1-simple-wave solution of isothermal equations. □: Numerical solution of ρ. ◇: Numerical solution of momentum m.



(a) (b)

Figure 3.16: h-box method on regular grid and irregular grid using Lax-Wendroff flux, for a 1-simple-wave solution of isothermal equations. □: Numerical solution of ρ. ◇: Numerical solution of momentum m.



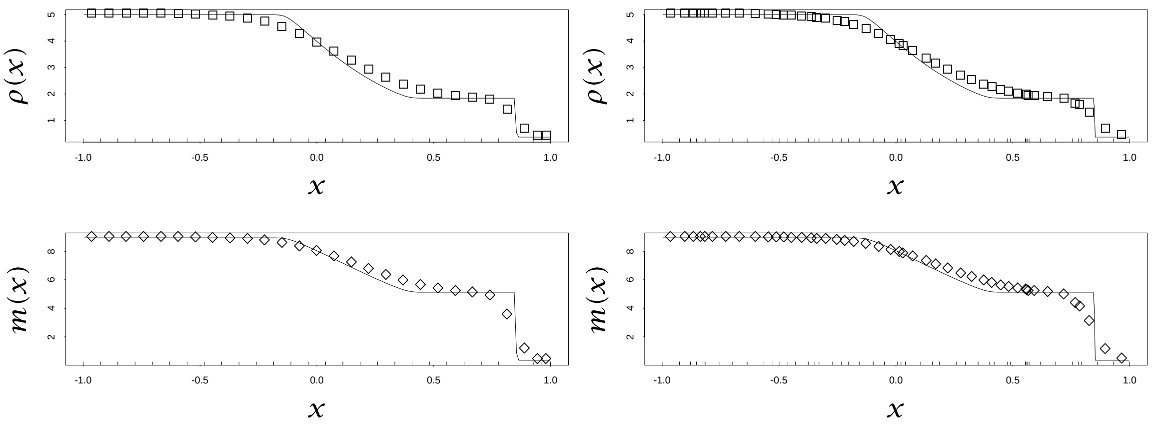
(a)

Order of accuracy

Method	$\|E(\rho)\|_1$	$\|E(m)\|_1$
□	0.96	0.98
◇	0.97	0.96
△	2.03	2.01
$*$	2.04	2.03

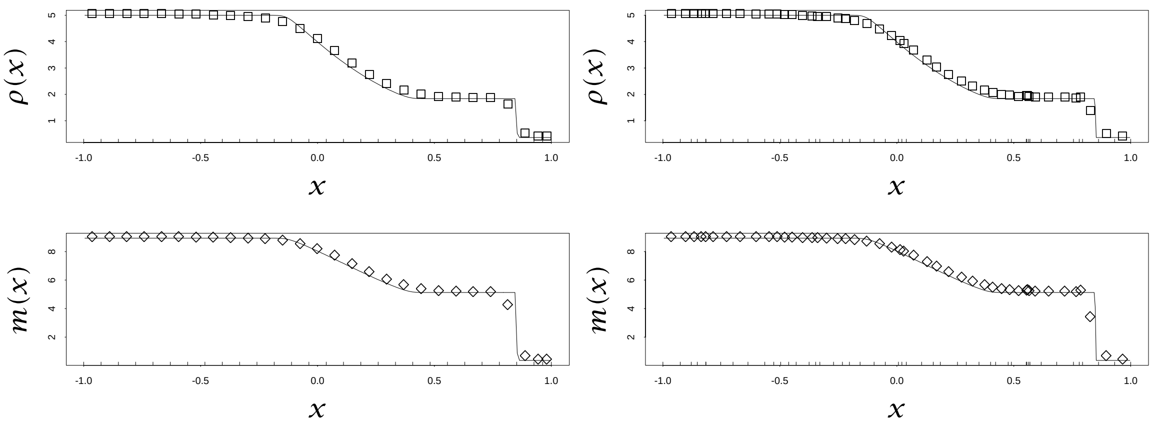
(b)

Figure 3.17: Comparison of 1-norm accuracy in a 1-simple wave in the isothermal gas equations. □: Godunov fluxes, uniform grid. ◇: Godunov fluxes, non-uniform grid. △: Lax-Wendroff fluxes with Min-Mod limiter, uniform grid. $*$: Lax-Wendroff fluxes with Min-Mod limiter, non-uniform grid.



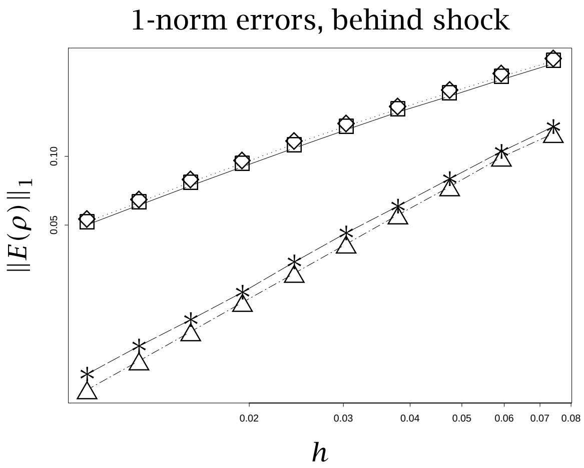
(a) (b)

Figure 3.18: h-box method on (a) regular grid and (b) irregular grid using Godunov flux, for a 1-wave/2-shock solution of isothermal equations. □: Numerical solution of ρ. ◇: Numerical solution of momentum m.



(a) (b)

Figure 3.19: h-box method on regular grid and irregular grid using Lax-Wendroff flux, for a 1-wave/2-shock solution of the isothermal gas equations. □: Numerical solution of ρ. ◇: Numerical solution of momentum m.



(a)

Order of accuracy

Method	$\|E(\rho)\|_1$	$\|E(m)\|_1$
$\square$	0.87	0.86
$\diamond$	0.88	0.87
$\triangle$	1.32	1.37
$*$	1.26	1.30

(b)

Figure 3.20: Comparison of 1-norm accuracy in a 1-rarefaction that has passed through a 2-shock for the isothermal gas equations. $\square$: Godunov fluxes, uniform grid. $\diamond$: Godunov fluxes, non-uniform grid. $\triangle$: Lax-Wendroff fluxes with Min-Mod limiter, uniform grid. $*$: Lax-Wendroff fluxes with Min-Mod limiter, non-uniform grid.

Chapter 4

UPWIND FINITE VOLUME METHODS FOR STRUCTURED GRIDS

Since a composite grid uses multiple structured grids, we need some term to describe the individual sub-grids and the geometrical data structures associated with them. In reference to the computational-to-physical coordinate *mapping* those grids represent, they are denoted as *MapGrids*.

The proposed h-box-CLAW method for embedded Chimera grids is a generalization of the MapGrid algorithm used in CLAWPACK. There are two advantages of using CLAWPACK-style fluxes on both MapGrids and at sub-grid interfaces. The first is that MapGrid fluxes can be calculated efficiently without worrying about location of cut-cells. Then fluxes on cut-cell faces can be replaced by those calculated using h-box-CLAW. The second advantage is that CLAWPACK is both stable and nearly *grid-isotropic*—it produces few numerical artifacts due to grid orientation. This means that h-box-CLAW derivatives will not be degraded in accuracy through use of data that is mismatched across the sub-grid interface.

After introducing some notation, we describe the basic steps of our implementation of CLAWPACK on a rectangular MapGrid, and then generalize the method for use on non-rectangular quadrilateral grids.

4.1 Notation for two-dimensional problems

Recall (1.3) from Chapter 1:

$$\boldsymbol{q}_t + \boldsymbol{f}(\boldsymbol{q})_x + \boldsymbol{g}(\boldsymbol{q})_y = 0$$

A standard set of finite volume techniques to solve this problem involve local solution of the one-dimensional problems

$$\boldsymbol{q}_t + \boldsymbol{f}(\boldsymbol{q})_x = 0 \tag{4.1}$$
$$\boldsymbol{q}_t + \boldsymbol{g}(\boldsymbol{q})_y = 0. \tag{4.2}$$

For example, one could calculate an $O(h)$-error solution using a dimensionally-split algorithm:

- Solve (4.1) using $\boldsymbol{Q}^n_{i,j}$ as starting data with a time-step of Δt to get an intermediate solution $\boldsymbol{Q}^*_{i,j}$.

- Solve (4.2) using $\boldsymbol{Q}^{*}_{i,j}$ as starting data with a time-step of Δt to get the solution $\boldsymbol{Q}^{n+1}_{i,j}$.

With Strang-splitting [52] and higher-order approximations in the one-dimensional flux calculations, it is possible to develop an $O(h^2)$-error solution. But a dimensionally-split method would be impractical at the interface of two grids, since the intermediate solutions on each side of the interface could be mismatched — as noted before, the sub-grids need not be aligned, and an x oriented solution on one grid would not merge well with an $\tilde{x}$ solution on another.

Alternatively, one could calculate the fluxes on x-normal and y-normal cell faces using local (first-order) solutions to (4.1) and (4.2), and then make the cell update in a single step. This is an example of a single-step or non-dimensionally-split method, here with $O(h)$-error and some orientation-dependent time-step restrictions. In order to create a non-dimensionally-split algorithm of higher-order accuracy that is stable in the presence of transverse flow, however, one must incorporate cross-derivative terms. In $O(h^2)$ dimensionally-split schemes cross-derivative terms are included as a by-product of using the intermediate solution $\boldsymbol{Q}^{*}_{i,j}$ in a solver for the other one-dimensional problem. In a single-step method, the cross-derivative terms must be included explicitly. Notation necessary for describing these methods is now introduced.

First assume that $\boldsymbol{f}'(\boldsymbol{q})$ and $\boldsymbol{g}'(\boldsymbol{q})$ are constant diagonalizable matrices $\mathbf{A}$ and $\mathbf{B}$ respectively, with

$$\mathbf{A} = \mathbf{R}_x \boldsymbol{\Lambda}_x {\mathbf{L}_x}^T = \mathbf{R}_x {\boldsymbol{\Lambda}_x}^+ {\mathbf{L}_x}^T + \mathbf{R}_x {\boldsymbol{\Lambda}_x}^- {\mathbf{L}_x}^T = \mathbf{A}^+ + \mathbf{A}^- \tag{4.3}$$

$$\mathbf{B} = \mathbf{R}_y \boldsymbol{\Lambda}_y {\mathbf{L}_y}^T = \mathbf{R}_y {\boldsymbol{\Lambda}_y}^+ {\mathbf{L}_y}^T + \mathbf{R}_y {\boldsymbol{\Lambda}_y}^- {\mathbf{L}_y}^T = \mathbf{B}^+ + \mathbf{B}^- \tag{4.4}$$

where

$$\begin{aligned} {\boldsymbol{\Lambda}_x}^+ &= \mathrm{diag}\{\max(\lambda_x^p, 0)\}, & {\boldsymbol{\Lambda}_x}^- &= \mathrm{diag}\{\min(\lambda_x^p, 0)\} \\ {\boldsymbol{\Lambda}_y}^+ &= \mathrm{diag}\{\max(\lambda_y^\sigma, 0)\}, & {\boldsymbol{\Lambda}_y}^- &= \mathrm{diag}\{\min(\lambda_y^\sigma, 0)\}, \end{aligned}$$

and the arrays of left- and right-eigenvectors are defined as

$$\begin{aligned} \boldsymbol{r}_x^p &= \mathbf{R}_x \boldsymbol{e}_p & \boldsymbol{r}_y^\sigma &= \mathbf{R}_y \boldsymbol{e}_\sigma \\ \boldsymbol{l}_x^p &= \mathbf{L}_x \boldsymbol{e}_p & \boldsymbol{l}_y^\sigma &= \mathbf{L}_y \boldsymbol{e}_\sigma \end{aligned}$$

where

$${\mathbf{L}_x}^T = {\mathbf{R}_x}^{-1} \qquad {\mathbf{L}_y}^T = {\mathbf{R}_y}^{-1}.$$

These linear operators will be used to transport the differences, or "jumps", of $\boldsymbol{Q}$ at cell interfaces. To denote these jumps we define

$$([\boldsymbol{Q}]_x)_{i-1/2,j} = \boldsymbol{Q}_{i,j} - \boldsymbol{Q}_{i-1,j} \qquad \left([\boldsymbol{Q}]_y\right)_{i,j-1/2} = \boldsymbol{Q}_{i,j} - \boldsymbol{Q}_{i,j-1};$$

i.e., a increment of $\pm 1/2$ indicates the index in which the difference is taking place. The x or y subscript indicates that the jump is proportional to the discrete form of an x or y partial derivative. The components of these jumps in terms of either $\mathbf{A}$ or $\mathbf{B}$ eigenvectors are called *waves*:

$$\boldsymbol{w}_x^p = \left(\boldsymbol{l}_x^{p\,T}\ [\boldsymbol{Q}]_x\right)\ \boldsymbol{r}_x^p \qquad \boldsymbol{w}_y^\sigma = \left(\boldsymbol{l}_y^{\sigma\,T}\ [\boldsymbol{Q}]_y\right)\ \boldsymbol{r}_y^\sigma$$

Note that if one denotes the number of eigenvectors as M_w (*i.e.*, the number of waves), one can then calculate the products $\mathbf{A}^+\,[\boldsymbol{Q}]_x$ and $\mathbf{A}^-\,[\boldsymbol{Q}]_x$ as follows:

$$\begin{aligned}
\mathbf{A}^+\,[\boldsymbol{Q}]_x &= \sum_{p=1}^{M_w} \max(\lambda_x^p, 0)\boldsymbol{w}_x^p \\
\mathbf{A}^-\,[\boldsymbol{Q}]_x &= \sum_{p=1}^{M_w} \min(\lambda_x^p, 0)\boldsymbol{w}_x^p.
\end{aligned}$$

The $\mathbf{B}^+\,[\boldsymbol{Q}]_y$ and $\mathbf{B}^-\,[\boldsymbol{Q}]_y$ products can be calculated in a similar way. Since these products are important for their use as flux-difference corrections, but may be used in a more general way than simply as flux-difference corrections, LeVeque [36] calls them by the slightly less cumbersome name of *fluctuations*, a usage originally coined in early work of Roe [45,46].

4.2 Summary of CLAWPACK in Two Dimensions

The grid-isotropic method we use on `MapGrids` is a re-implementation of the method use in CLAWPACK, developed by LeVeque [32,33,35,36]. This approach uses an upwinded form of the augmented second-order expansion

$$\begin{aligned}
\boldsymbol{Q}(x, t_{n+1}) = \boldsymbol{Q}(x, t_n) &- \Delta t\left[(\boldsymbol{f})_x + (\boldsymbol{g})_y\right] \\
&+ \frac{\Delta t^2}{2}\left[\left(\mathbf{AA}\boldsymbol{Q}_x + \mathbf{AB}\boldsymbol{Q}_y\right)_x + \left(\mathbf{BA}\boldsymbol{Q}_x + \mathbf{BB}\boldsymbol{Q}_y\right)_y\right] \\
&- \frac{\Delta t^3}{6}\left[\left(\mathbf{AAB}\boldsymbol{Q}_{yx} + \mathbf{ABA}\boldsymbol{Q}_{xy} + \mathbf{ABB}\boldsymbol{Q}_{yy}\right)_x\right. \\
&\qquad \left. + \left(\mathbf{BAA}\boldsymbol{Q}_{xx} + \mathbf{BAB}\boldsymbol{Q}_{yx} + \mathbf{BBA}\boldsymbol{Q}_{xy}\right)_y\right].
\end{aligned} \tag{4.5}$$

Note that this expansion, which takes account of non-commutativity of **A** and **B**, is in conservation form, and though it has some $O(\Delta t^3)$ terms, it does not have all the $O(\Delta t^3)$ terms necessary to be a third-order scheme. When implemented in an upwinded form this method has improved stability characteristics because it has a larger, more circular stencil than the ordinary second-order upwind scheme. For simplicity of implementation, however, CLAWPACK does not take account of non-commutativity of operators; all third-order terms with two coefficients **AA** are treated as if they come from the $\mathbf{BAA}\boldsymbol{Q}_{xxy}$ correction. A similar simplification is used for higher-order terms with two **BB** coefficients. In practice, this does not adversely affect the accuracy of the method, and the method retains the desired stability.

CLAWPACK uses expansion (4.5) with split operators of the form (4.3), (4.4) and suitably up-winded difference approximations to derivatives to create a numerical method which is both stable and nearly grid-isotropic. It can be interpreted as forming correction waves for normal flux corrections and then adding the appropriate transversely propagated components of those correction waves to neighboring perpendicular faces.

The following descriptions of CLAWPACK-style flux algorithms are essentially the same as those used in the distributed code, though the actual implementation differs slightly. We use a modified finite-volume update formula that generalizes to quadrilateral grids:

$$\boldsymbol{Q}_{i,j}^{n+1} = \boldsymbol{Q}_{i,j}^{n} - \frac{\Delta t}{\kappa_{i,j}} \Big((\boldsymbol{F}\Delta s)_{i+1/2,j} - (\boldsymbol{F}\Delta s)_{i-1/2,j} + (\boldsymbol{G}\Delta s)_{i,j+1/2} - (\boldsymbol{G}\Delta s)_{i,j-1/2} \Big), \quad (4.6)$$

where $\kappa_{i,j}$ is the cell size, numerical fluxes $\boldsymbol{F}$ and $\boldsymbol{G}$ are evaluated in the face-normal direction, and Δs is the face length.

Table 4.1: Summary of flux-correction descriptors used in this research.

Flux correction notation	Highest-order on x-sweeps	Highest-order on y-sweeps
1	$\mathbf{A}\ \boldsymbol{q}_x$	$\mathbf{B}\ \boldsymbol{q}_y$
1T	$\mathbf{B}\ \mathbf{A}\ \boldsymbol{q}_{xy}$ †	$\mathbf{A}\ \mathbf{B}\ \boldsymbol{q}_{yx}$ ‡
2	$\mathbf{A}\ \mathbf{A}\ \boldsymbol{q}_{xx}$	$\mathbf{B}\ \mathbf{B}\ \boldsymbol{q}_{yy}$
2TT	$\mathbf{B}\ \mathbf{A}\ \mathbf{A}\ \boldsymbol{q}_{xxy}$ †	$\mathbf{A}\ \mathbf{B}\ \mathbf{B}\ \boldsymbol{q}_{yyx}$ ‡
2NT	$\mathbf{A}\ \mathbf{A}\ \mathbf{B}\ \boldsymbol{q}_{yxx}$	$\mathbf{B}\ \mathbf{B}\ \mathbf{A}\ \boldsymbol{q}_{xyy}$
2TN	$\mathbf{A}\ \mathbf{B}\ \mathbf{A}\ \boldsymbol{q}_{xyx}$	$\mathbf{B}\ \mathbf{A}\ \mathbf{B}\ \boldsymbol{q}_{yxy}$

† Corrected on x-sweeps, but regarded as a correction in the y-direction.

‡ Corrected on y-sweeps, but regarded as a correction in the x-direction.

Notes on flux-correction notation. The convention used in this research for flux corrections is to use a number (*e.g.*, **1** or **2**) and matrix symbols of **N** (for *Normal*) or **T** (for *Transverse*) afterwards. Because the flux-corrections are calculated in x-oriented or y-oriented sweeps, this notation generalizes the correction due to the normal or transverse operators. The initial number refers to the highest degree derivative present in the normal direction. At least one **N** operator is always assumed (and is applied last). Flux corrections that contain non-normal terms have operators **N** or **T** afterward to indicate the non-commutative order of matrix operations in the correction. Table 4.2 summarizes the abbreviations and the highest-order partial-derivative approximations resulting from those corrections on x-oriented and y-oriented sweeps. Note that the terms marked with footnotes are considered to be corrections in the *transverse* direction to the sweep.

The notation here is more general than what is needed in practice. In CLAWPACK, the three higher-order corrections, **2TT**, **2NT** and **2TN**, are lumped together as if matrices and spatial derivatives were commutative. The resulting correction is applied to transverse fluxes. For convenience, however, the higher-order transverse correction in CLAWPACK is called a **2TT**,correction.

4.2.1 *Donor-cell upwind flux corrections*

Algorithm 4.1 (Godunov normal fluxes in CLAWPACK).

for all i, j **do** {Initialize $\boldsymbol{F}$ and $\boldsymbol{G}$ fluxes}

$$\boldsymbol{F}_{i-1/2,j} \leftarrow \begin{pmatrix} \boldsymbol{f}(\boldsymbol{Q}_{i,j}) & \boldsymbol{g}(\boldsymbol{Q}_{i,j}) \end{pmatrix} \cdot \boldsymbol{n}_{i-1/2,j}$$

{Initialize $\boldsymbol{F}$ with flux in cell on right of face}

$$\boldsymbol{G}_{i,j-1/2} \leftarrow \begin{pmatrix} \boldsymbol{f}(\boldsymbol{Q}_{i,j}) & \boldsymbol{g}(\boldsymbol{Q}_{i,j}) \end{pmatrix} \cdot \boldsymbol{n}_{i,j-1/2}$$

{Initialize $\boldsymbol{G}$ with flux in cell above face}

end for

for all i, j **do** {Subtract normal fluctuations to get Godunov (DCU) fluxes}

$$\boldsymbol{F}_{i-1/2,j} \mathrel{-}= (\mathbf{A}^+ [\boldsymbol{Q}]_x)_{i-1/2,j}$$

$$\boldsymbol{G}_{i,j-1/2} \mathrel{-}= (\mathbf{B}^+ [\boldsymbol{Q}]_y)_{i,j-1/2}$$

end for

Algorithm 4.1

A finite volume method which uses only first-order Godunov fluxes is known as a *donor-cell upwind* (DCU) method. We therefore denote the first-order fluxes (Algorithm 4.1) used in that method as DCU fluxes. The effect of DCU flux updates is shown

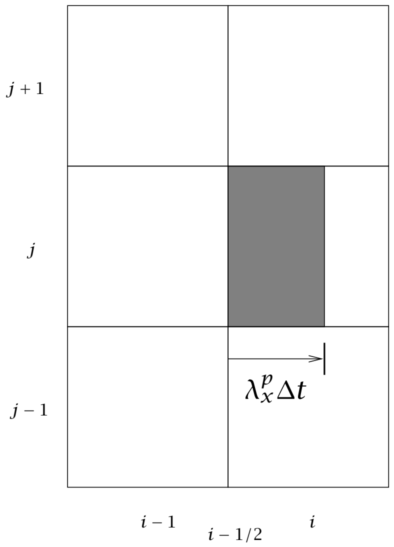


Figure 4.1: The effect of normal wave component $\boldsymbol{w}_x^p$ of $\mathbf{A}^+ [\boldsymbol{Q}]_x$, where $\lambda_x^p > 0$

in Figure 4.1. Such a method is stable only up to

$$\left| \max_p \frac{\lambda_x^p \ \Delta t \ \Delta s_\eta}{\kappa} \right| + \left| \max_\sigma \frac{\lambda_y^\sigma \ \Delta t \ \Delta s_\xi}{\kappa} \right| \leq 1.$$

In addition, the DCU method is not grid-isotropic; symmetric initial conditions for isotropic equations give rise to grid-oriented numerical results.

Note that the CFL number is defined here in terms of the maximum fractional area (κ) of the downwind cell that is updated by waves from $\mathbf{A}$. This generalized form can be used even when the physical grid is not uniform.

4.2.2 Corner transport upwind flux corrections

Algorithm 4.2 (Corner Transport Upwind in CLAWPACK).

for all j **do** {Sweep over rows}

  for all i **do** {Transverse propagations of waves from each $(i - 1/2, j)$}

$$\boldsymbol{G}_{i,j+1/2} \ -= \ \frac{\Delta t}{2 \ \kappa_{i,j}} \ \mathbf{B}^+ (\Delta s \ \mathbf{A}^+ \ [\boldsymbol{Q}]_x)_{i-1/2,j}$$

$$\boldsymbol{G}_{i,j-1/2} \ -= \ \frac{\Delta t}{2 \ \kappa_{i,j}} \ \mathbf{B}^- (\Delta s \ \mathbf{A}^+ \ [\boldsymbol{Q}]_x)_{i-1/2,j}$$

$$\boldsymbol{G}_{i-1,j+1/2} -= \ \frac{\Delta t}{2 \ \kappa_{i-1,j}} \ \mathbf{B}^+ (\Delta s \ \mathbf{A}^- \ [\boldsymbol{Q}]_x)_{i-1/2,j}$$

6:
$$G_{i-1,j-1/2} -= \frac{\Delta t}{2\ \kappa_{i-1,j}}\ \mathbf{B}^-(\Delta s\ \mathbf{A}^-\ [\boldsymbol{Q}]_x)_{i-1/2,j}$$

7: **end for**
8: **end for**
9: Correct $\boldsymbol{F}$ fluxes in similar way, sweeping over columns i

Algorithm 4.2

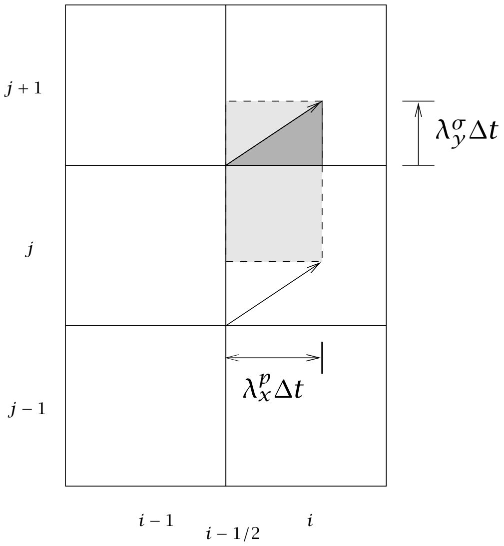


Figure 4.2: Geometrical interpretation of CTU correction. The portion of wave $\boldsymbol{w}_x^p$ of $\mathbf{A}^+\ [\boldsymbol{Q}]_x$ that is transversely propagated through face $(i, j+1/2)$ in the σ-th wave family of $\mathbf{B}^+$.

In Algorithm 4.2 we see the CLAW-like correction to DCU fluxes. This correction makes updates more accurate (though still first-order), grid-isotropic and stable up to

$$\max\left(\left|\max_p \frac{\Delta t\ \Delta s_\eta\ \lambda_x^p}{\kappa}\right|, \left|\max_\sigma \frac{\Delta t\ \Delta s_\xi\ \lambda_y^\sigma}{\kappa}\right|\right) \le 1.$$

For simplicity, I describe the corrections in the y directions that arise from transverse propagation of $\mathbf{A}^\pm\ [Q]$ fluctuations in the x direction (during "x-sweeps"). The corrections arising from y-sweeps are calculated in a similar way. Figure 4.2 shows a geometrical interpretation of this transverse correction. Because the numerical update using these fluxes is equivalent, in the case of advection, to a first-order method known as *corner transport upwind* (CTU) [16], we will denote the resulting fluxes after Algorithm 4.2 as CTU fluxes, and the CLAW-like method using CTU fluxes as a CTU method. The CTU method is a "first-order" transverse correction because the order of error in the update $\boldsymbol{Q}^{n+1}$ is not decreased by the flux correction in Algorithm 4.2, even though it results in second-order ($\boldsymbol{Q}_{xy}$) derivative corrections and the overall error constant is reduced. Another reason to denote the CTU correction as first-order is that it is equivalent to the transverse propagation of piece-wise constant reconstructions of Q.

Details of CLAWPACK's transverse corrections

In LeVeque's CLAWPACK code examples, a different form of the transverse fluctuation correction is used depending on whether the flux functions $\boldsymbol{f}$ and $\boldsymbol{g}$ depend explicitly on position $\boldsymbol{x}$ or not. In the CTU flux-correction algorithm it is not specified what values are used for setting up approximate Riemann solvers. For example, in the correction factor for interface $(i, j+1/2)$,

$$\frac{\Delta t}{2\,\kappa}\,\mathbf{B}^{+}(\Delta s_{\eta}\mathbf{A}^{+}\,[\boldsymbol{Q}]_{x})_{i-1/2,j},$$

, we did not specify whether $\mathbf{B}$ is calculated at position $(i-1/2, j)$ or $(i, j+1/2)$ For uniform $\mathbf{A}$ and $\mathbf{B}$ on a rectangular grid this is not an issue. However, if $\mathbf{A}$ and $\mathbf{B}$ are functions of $(\boldsymbol{x}, \boldsymbol{q}, \boldsymbol{n})$ (*i.e.*, position, $\boldsymbol{q}$, and orientation), one should calculate the corrective factor as

$$\boldsymbol{G}_{i,j+1/2} -= \frac{\Delta t}{2\,\kappa_{i,j}}\,\mathbf{B}^{+}(\boldsymbol{x}_{1,j+1/2}, \boldsymbol{Q}^{\mathrm{avg}}_{i-1/2,j}, \boldsymbol{n}_{1,j+1/2})\ (\Delta s_{\eta}\mathbf{A}^{+}\,[\boldsymbol{Q}]_{x})_{i-1/2,j}.$$

In other words, $\mathbf{B}$ should be evaluated using Roe-averaged $\boldsymbol{q}$ values from the face originating the wave, but then $\mathbf{B}$ (really $\widetilde{\mathbf{B}}$) should be rotated into the direction of the face through which the wave is transversely propagated. In addition, any spatially-dependent components of the transverse wave speed must also taken from the *transversely-affected* face rather than the originating face. Finally, by using the area from the cell which is upwind from the transverse face, here (i, j), the triangular correction area is stretched correctly in the case of a non-orthogonal grid. This can be demonstrated in the case of non-uniform advection: the quantity

$$\text{Cell fraction} = \frac{(\boldsymbol{V}\cdot\Delta s)_{i-1/2,j}\ \Delta t}{\kappa_{i,j}}$$

is the fraction of cell (i, j) that is filled up by the normal wave from face $(i-1/2, j)$, so the width of the transverse triangle is

$$(\Delta s)_{1,j+1/2}\ \times\ \text{Cell fraction.}$$

But the factor $(\Delta s)_{1,j+1/2}$ is incorporated into $(\boldsymbol{V}\cdot\Delta s)_{i,j+1/2}$, so the base of the transverse correction triangle is calculated automatically. This is shown geometrically by Figure 4.3.

To clarify, one may consider two scalar equations: two-dimensional advection in a divergence-free velocity field, an example of a spatially-dependent wave speed; and a

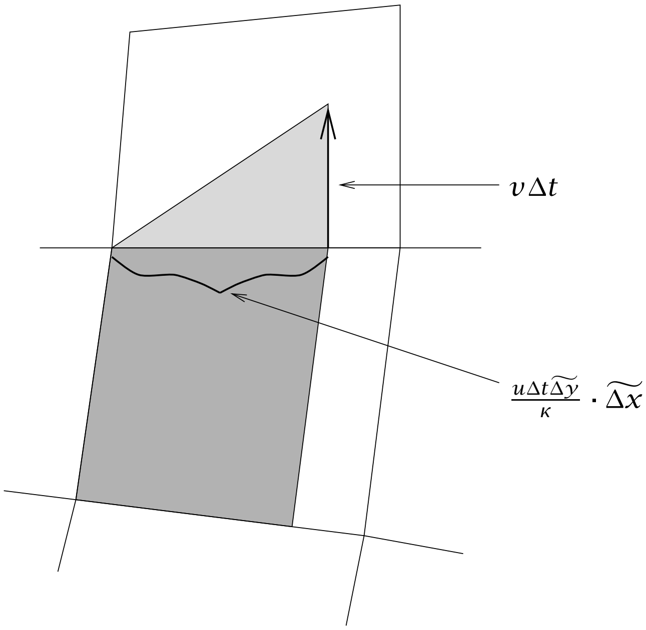


Figure 4.3: Interpretation of transverse correction on non-orthogonal grids

two-dimensional form of Burgers' equation, an example of a q-dependent wave speed.

In the advection case, one often knows only the normal velocity on cell faces scaled by the length of the side, $\boldsymbol{V}\cdot\Delta s$, calculated in such a way to enforce the discrete divergence-free condition. It is then appropriate to use transverse velocities *from the transverse face* to make the corrections. For divergence-free velocity fields this choice ensures that zero-flux boundary conditions are enforced. In this example, no other modifications have to be made to incorporate the face direction into flux-corrections.

In the case of $\boldsymbol{f}$ and $\boldsymbol{g}$ not having any explicit dependence on x, say in the case of a scalar 2-D Burgers' equation,

$$q_t + \left(\frac{q^2}{2}\right)_x + \left(\frac{q^2}{2}\right)_y = 0, \tag{4.7}$$

the local speed of a wave $\boldsymbol{w}$ across a face is $q^{\text{avg}} = (q_{\text{left}} + q_{\text{right}})/2$. If the transverse value of q^{avg} is used rather than the normal face value in the CTU correction, fluxes giving rise to spurious humps in the solution can occur. But the wave speed on the transverse face, though q-dependent, can still depend on the orientation of the transverse face and needs to be adjusted accordingly. For the 2-D Burgers' equation example,

$$\text{transverse wavespeed}_{i,j+1/2} = q^{\text{average}}_{i-1/2,j}\,(n_1 + n_2)_{i,j+1/2},$$

where $\boldsymbol{n}_{i,j+1/2} = (n_1, n_2)$ is the normal vector on the transverse face. Here one can see that the transverse-correction wave speed is a function of orientation on the transversely-affected face and q on the wave-originating face.

4.2.3 Second-order normal flux corrections

Algorithm 4.3 (Second-order normal corrections in CLAWPACK).

1: **for all** i, j **do** {Higher order normal correction}

2:
$$\boldsymbol{F}_{i-1/2,j} \mathrel{+}= \frac{1}{2} \sum_{p=1}^{M_w} \left|\lambda_x^p\right| \left(1 - \frac{\Delta t\ \Delta s_\eta}{\kappa} \left|\lambda_x^p\right|\right) \widehat{\boldsymbol{w}}_x^p$$

{where $\widehat{\boldsymbol{w}}_x^p$ is the limited form of $\boldsymbol{w}_x^p$}

3:
$$\boldsymbol{G}_{i,j-1/2} \mathrel{+}= \frac{1}{2} \sum_{\sigma=1}^{M_w} \left|\lambda_y^\sigma\right| \left(1 - \frac{\Delta t\ \Delta s_\xi}{\kappa} \left|\lambda_y^\sigma\right|\right) \widehat{\boldsymbol{w}}_y^\sigma$$

{where $\widehat{\boldsymbol{w}}_y^\sigma$ is the limited form of $\boldsymbol{w}_y^\sigma$}

4: **end for**

Algorithm 4.3

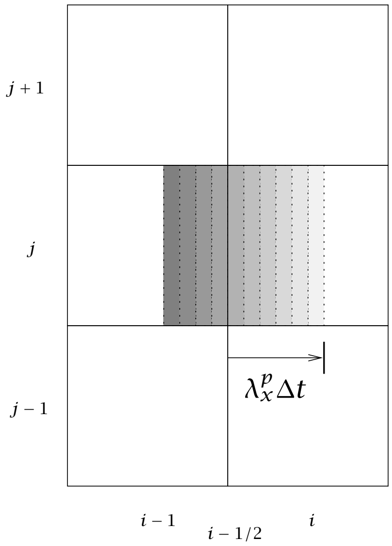


Figure 4.4: Geometrical interpretation of the higher-order normal flux correction due to a piece-wise linear approximation of the upwind value $\boldsymbol{Q}_{i-1,j}$ in wave $\boldsymbol{w}_x^p$ of $\mathbf{A}^+$.

The CLAWPACK algorithm for adding higher-order normal corrections to bring updates to second-order accuracy is seen in Algorithm 4.3. As seen in Figure 4.4, this correction can be interpreted as a limited reconstruction of the piece-wise linear correction to the $\boldsymbol{Q}_{i,j-1}$ portion of $([\boldsymbol{Q}]_x)_{i-1/2,j}$ that is carried across the $(i-1/2, j)$-th

x-perpendicular face by $\mathbf{A}^+$. Note that the reuse of the $\widehat{\cdot}$ notation for the limited wave is consistent with its use in Chapter 2 since $\widehat{\boldsymbol{w}}_x$ and $\widehat{\boldsymbol{w}}_y$ also represent reconstructions with sufficient limiting.

4.2.4 CLAWPACK's higher-order transverse corrections

Algorithm 4.4 (Higher-order transverse corrections in CLAWPACK).

for all j **do** {Sweep over rows}
  for all i **do** {Modify $(i-1/2, j)$ fluxes during x-sweeps}

$$(\mathbf{A}^+ [\boldsymbol{Q}]_x)_{i-1/2,j} \;-\!= \sum_{p=1}^{M_w} \left|\lambda_x^p\right| \left(1 - \frac{\Delta t\ \Delta s_\eta}{\kappa} \left|\lambda_x^p\right|\right) \widehat{\boldsymbol{w}}_x^p$$

$$(\mathbf{A}^- [\boldsymbol{Q}]_x)_{i-1/2,j} \;+\!= \sum_{p=1}^{M_w} \left|\lambda_x^p\right| \left(1 - \frac{\Delta t\ \Delta s_\eta}{\kappa} \left|\lambda_x^p\right|\right) \widehat{\boldsymbol{w}}_x^p$$

  end for
end for
Modify $(i, j-1/2)$ fluxes in similar way in y-sweeps

Algorithm 4.4

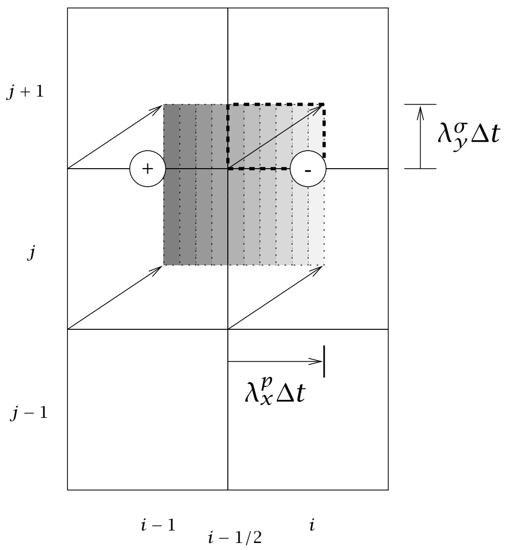


Figure 4.5: Higher order correction due to $\mathbf{A}$ wave is transversely propagated using $\mathbf{B}^+$. To have the same subtract-from-right and add-to-left cell correction effect as a change in $\boldsymbol{F}_{i-1/2,j}$, the higher-order transverse correction must be subtracted from $\boldsymbol{G}_{i,j+1/2}$ and added to $\boldsymbol{G}_{i,j-1/2}$.

The last type of `CLAWPACK`-style flux correction is used to incorporate part of the higher-order normal correction into the transverse corrections. This correction increases overall accuracy (though the method remains second-order), helps to decrease numerical dispersion near transversely-propagated discontinuities, and improves the stability and grid-isotropic behavior of the solution. As described in Algorithm 4.4, this correction is actually made in the quantities $\mathbf{A}^* [\boldsymbol{Q}]_x$ and $\mathbf{B}^* [\boldsymbol{Q}]_y$ *before* those fluctuations are split up for use in making the transverse corrections in Algorithm 4.2. In this way the correct second-difference in the normal direction is propagated to transverse faces. Note also that these corrections can use part of the same non-linear correction factor that is used for the higher-order normal corrections, thus avoiding much extra computation. For simplicity, Algorithm 4.4 describes only the corrections made during x-sweeps — the corrections to $\mathbf{B}^* [\boldsymbol{Q}]_y$ are made in a similar way during y-sweeps.

We should note that for non-linear hyperbolic systems of equations, an approximate Roe solver is used to find the **A** or **B** matrix on a face that corresponds to the solution of the Riemann problem in the face-normal direction. The approximate Roe solver usually uses "Roe-averages" of the quantities on each side of the face, a procedure which typically requires taking square roots and is one of the more computationally intensive procedures of the algorithm. The version of `CLAWPACK` for systems of equations calculates **A** and **B** matrices for waves from a single face using the *same* set of Roe-averaged variables, and then uses those matrices in Algorithms 4.1-4.4 to distribute the fluctuations to neighboring faces. Because **1T** corrections are required for a second-order method, higher-order modifications of the $\mathbf{A}^* [\boldsymbol{Q}]$ fluctuations does not increase work appreciably.

4.3 General Curvilinear Coordinates in Two dimensions[1]

If one denotes computational coordinates (ξ, η) by $\boldsymbol{\xi}$ and physical coordinates (x, y) by $\boldsymbol{x}$, one can transform a hyperbolic PDE from a grid in the $\boldsymbol{\xi}$ plane to a grid in the $\boldsymbol{x}$ plane, as illustrated in Figure 4.6[2]. Since this transformation is from physical to computational coordinates, we define J as the determinant of the Jacobian matrix of the function $\boldsymbol{\xi}(\boldsymbol{x})$; it is preferable, however, to define J in terms of the transformation $\boldsymbol{x}(\boldsymbol{\xi})$, which is more easily determined from physical grid quantities:

$$J^{-1} = \left| \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\xi}} \right| = x_\xi y_\eta - x_\eta y_\xi \tag{4.8}$$

$$J = \left| \frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{x}} \right| = \xi_x \eta_y - \xi_y \eta_x = \frac{1}{x_\xi y_\eta - x_\eta y_\xi}. \tag{4.9}$$

[1] The development which follows uses the conventions and notation of [19]; where this work differs is in modifications to better agree with finite volume method notation.

[2] The superscript $\tilde{\cdot}$ will be used to denote quantities that have been transformed from one coordinate system to another.

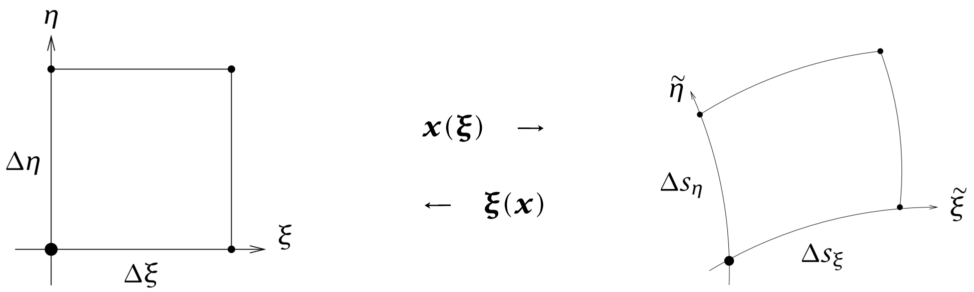


Figure 4.6: Transformation from uniform computational to physical coordinates and visa-versa.

The notation Δs_ξ, as seen in Figure 4.6 for example, is used as an abbreviation for the *Euclidean* distance between two cell vertices, which agrees to $O(\Delta s_\xi^2)$ with the arc-length of the curve $\boldsymbol{x}(\xi, \eta = \text{constant})$ between those cell vertices. The notation Δs_η is defined similarly. The use of subscripts with finite differences is analogous to the use of the ξ or η subscript on dependent variables to denote a partial derivative.

We henceforth assume that the computational grid is uniform. Using previously defined notation, the area of a grid cell in the computational domain is simply $\Delta\xi\Delta\eta$. In the physical domain the area of the curvilinear grid cell is $J^{-1}\Delta\xi\Delta\eta$, which can be derived from the change of variables theorem, or alternatively with tensor notation, as in [19]. If an explicit mapping function is not known, it may be possible to determine J^{-1} only by viewing the cell as a polygon with straight edges, and solving for J^{-1} with the second-order approximation

$$\kappa \approx J^{-1}\Delta\xi\Delta\eta, \tag{4.10}$$

where κ represents the polygonal cell-area.

It is helpful to derive other mesh quantities as well in terms of those that can be measured directly. First we denote the indices at cell centers as (i, j). Cell corners, or vertices, have indices $(i \pm 1/2, j \pm 1/2)$. Similarly, the centers of cell faces on which ξ is constant have indices $(i \pm 1/2, j)$, and centers of cell faces on which η is constant have indices $(i, j \pm 1/2)$. It can be assumed generally that only the vertex locations are known, not the mapping. We may then define

$$\left(\Delta x_\xi\right)_{i,j-1/2} = x_{i+1/2,j-1/2} - x_{i-1/2,j-1/2} \tag{4.11}$$

$$\left(\Delta y_\xi\right)_{i,j-1/2} = y_{i+1/2,j-1/2} - y_{i-1/2,j-1/2} \tag{4.12}$$

$$(\Delta x_\eta)_{i-1/2,j} = x_{i-1/2,j+1/2} - x_{i-1/2,j-1/2} \tag{4.13}$$

$$(\Delta y_\eta)_{i-1/2,j} = y_{i-1/2,j+1/2} - y_{i-1/2,j-1/2}. \tag{4.14}$$

Note that we may drop either the indices or the ξ or η subscripts for cell-face differences without ambiguity. We can derive the relationships

$$\Delta x_\eta \approx x_\eta \Delta\eta \tag{4.15}$$
$$\Delta y_\eta \approx y_\eta \Delta\eta \tag{4.16}$$
$$\Delta x_\xi \approx x_\xi \Delta\xi \tag{4.17}$$
$$\Delta y_\xi \approx y_\xi \Delta\xi, \tag{4.18}$$

which are accurate to $O(\Delta^2)$ when Euclidean *vs.* curvilinear distances are used. One can then represent the physical face sizes Δs, which are face *lengths* in two dimensions, in terms of both straight-edge distances and curvilinear differentials as follows,

$$\Delta s_\eta = \sqrt{\Delta x_\eta{}^2 + \Delta y_\eta{}^2} \qquad \approx \sqrt{x_\eta^2 + y_\eta^2}\,\Delta\eta \quad = ds_\eta\,\Delta\eta \tag{4.19}$$
$$\Delta s_\xi = \sqrt{\Delta x_\xi{}^2 + \Delta y_\xi{}^2} \qquad \approx \sqrt{x_\xi^2 + y_\xi^2}\,\Delta\xi \quad = ds_\xi\,\Delta\xi. \tag{4.20}$$

Finally, we can derive the quantities $\partial\xi_i/\partial x_j$ in terms of $\partial x_i/\partial\xi_j$:

$$\begin{aligned} \xi_x &= y_\eta/J^{-1} & \qquad \eta_x &= -y_\xi/J^{-1} \\ \xi_y &= -x_\eta/J^{-1} & \qquad \eta_y &= x_\xi/J^{-1}. \end{aligned}$$

4.3.1 Hyperbolic partial differential equations in general curvilinear coordinates

We will consider a general first-order hyperbolic system of equations in two dimensions:

$$\boldsymbol{q}_t + \boldsymbol{f}_x + \boldsymbol{g}_y = 0.$$

Introducing

$$\frac{\partial}{\partial x} = \xi_x \frac{\partial}{\partial \xi} + \eta_x \frac{\partial}{\partial \eta} \tag{4.21}$$

and

$$\frac{\partial}{\partial y} = \xi_y \frac{\partial}{\partial \xi} + \eta_y \frac{\partial}{\partial \eta} \tag{4.22}$$

and multiplying the system by J^{-1} (assumed to be constant with respect to time), we get

$$\left(\boldsymbol{q}J^{-1}\right)_t + \xi_x \boldsymbol{f}_\xi J^{-1} + \xi_y \boldsymbol{g}_\xi J^{-1} + \eta_x \boldsymbol{f}_\eta J^{-1} + \eta_y \boldsymbol{g}_\eta J^{-1} = 0. \tag{4.23}$$

If one makes substitutions such as

$$\xi_x \boldsymbol{f}_\xi J^{-1} = \left(\xi_x \boldsymbol{f} J^{-1}\right)_\xi - \boldsymbol{f}\left(\xi_x J^{-1}\right)_\xi = \left(\xi_x \boldsymbol{f} J^{-1}\right)_\xi - \boldsymbol{f}\,(y_\eta)_\xi$$

the resulting cancellation yields

$$\tilde{\boldsymbol{q}}_t + \tilde{\boldsymbol{f}}_\xi + \tilde{\boldsymbol{g}}_\eta = 0. \tag{4.24}$$

where

$$\tilde{\boldsymbol{q}} = \boldsymbol{q}\, J^{-1} \qquad \tilde{\boldsymbol{f}} = \left(\xi_x\, \boldsymbol{f} + \xi_y\, \boldsymbol{g}\right) J^{-1} \qquad \text{and} \qquad \tilde{\boldsymbol{g}} = \left(\eta_x\, \boldsymbol{f} + \eta_y\, \boldsymbol{g}\right) J^{-1}.$$

One may therefore solve the PDE numerically on a curvilinear grid using a method very much like that used on a uniform rectangular grid, though it should be emphasized that one does not usually update $\tilde{\boldsymbol{q}}$ but rather $\boldsymbol{q}$ in order to avoid transformational approximation error.

Before continuing, we will redefine $\tilde{\boldsymbol{f}}$ and $\tilde{\boldsymbol{g}}$ in a way that is more easily calculated using physical grid values. We define the cell-face perpendiculars

$$\boldsymbol{\Delta s}_\eta = \begin{pmatrix} \Delta y_\eta \\ -\Delta x_\eta \end{pmatrix} \tag{4.25}$$

$$\boldsymbol{\Delta s}_\xi = \begin{pmatrix} -\Delta y_\xi \\ \Delta x_\xi \end{pmatrix}, \tag{4.26}$$

with magnitudes $|\Delta s_\eta|$ and $\left|\Delta s_\xi\right|$ respectively, and the vectors perpendicular to curvilinear coordinate lines (evaluated at face centers),

$$\boldsymbol{ds}_\eta = \begin{pmatrix} y_\eta \\ -x_\eta \end{pmatrix} \approx \frac{\boldsymbol{\Delta s}_\eta}{\Delta \eta} \tag{4.27}$$

$$\boldsymbol{ds}_\xi = \begin{pmatrix} -y_\xi \\ x_\xi \end{pmatrix} \approx \frac{\boldsymbol{\Delta s}_\xi}{\Delta \xi}. \tag{4.28}$$

Then

$$\begin{aligned} \mathbf{F} \cdot \boldsymbol{ds}_\eta &= y_\eta\, \boldsymbol{f} - x_\eta\, \boldsymbol{g} \\ &= \xi_x\, J^{-1}\, \boldsymbol{f} + \xi_y\, J^{-1}\, \boldsymbol{g} \\ &= \left(\xi_x\, \boldsymbol{f} + \xi_y\, \boldsymbol{g}\right)\, J^{-1} \\ &= \tilde{\boldsymbol{f}} \\ &\approx \frac{\mathbf{F} \cdot \boldsymbol{\Delta s}_\eta}{\Delta \eta}. \end{aligned} \tag{4.29}$$

Similarly,

$$
\begin{aligned}
\mathbf{F} \cdot \boldsymbol{ds}_{\xi} &= -y_{\xi}\, \boldsymbol{f} + x_{\xi}\, \boldsymbol{g} \\
&= \eta_x\, J^{-1}\, \boldsymbol{f} + \eta_y\, J^{-1}\, \boldsymbol{g} \\
&= \Big(\eta_x\, \boldsymbol{f} + \eta_y\, \boldsymbol{g}\Big)\, J^{-1} \\
&= \tilde{\boldsymbol{g}} \\
&\approx \frac{\mathbf{F} \cdot \boldsymbol{\Delta s}_{\xi}}{\Delta \xi}.
\end{aligned}
\tag{4.30}
$$

As we will see below, while the definitions of $\tilde{\boldsymbol{f}}$ and $\tilde{\boldsymbol{g}}$ are important for the numerical method, it will be more important to use the derivative matrices $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$ for terms that derive more directly from (4.23). With that end in view, note that (4.23) can be transformed to

$$
\tilde{\boldsymbol{q}} + \tilde{\mathbf{A}}\, \boldsymbol{q}_{\xi} + \tilde{\mathbf{B}}\, \boldsymbol{q}_{\eta} = 0 \tag{4.31}
$$

where

$$
\tilde{\mathbf{A}} = \Big(\xi_x\, \mathbf{A} + \xi_y\, \mathbf{B}\Big)\, J^{-1} \tag{4.32}
$$

$$
\tilde{\mathbf{B}} = \Big(\eta_x\, \mathbf{A} + \eta_y\, \mathbf{B}\Big)\, J^{-1}. \tag{4.33}
$$

If we define a general tensor form of the flux-derivative matrix,

$$
\mathbb{A} = \begin{pmatrix} \mathbf{A} & \mathbf{B} \end{pmatrix}, \tag{4.34}
$$

and the tensor-vector product (restricted to vectors $\boldsymbol{b}$ of length 2)

$$
\mathbb{A} \cdot \boldsymbol{b} = b_1\, \mathbf{A} + b_2\, \mathbf{B}, \tag{4.35}
$$

then (4.32) and (4.33) can be written more conveniently as

$$
\begin{aligned}
\tilde{\mathbf{A}} &= \mathbb{A} \cdot \boldsymbol{ds}_{\eta} \\
&= \frac{\mathbb{A} \cdot \boldsymbol{\Delta s}_{\eta}}{\Delta \eta}
\end{aligned}
\tag{4.36}
$$

$$
\begin{aligned}
\tilde{\mathbf{B}} &= \mathbb{A} \cdot \boldsymbol{ds}_{\xi} \\
&= \frac{\mathbb{A} \cdot \boldsymbol{\Delta s}_{\xi}}{\Delta \xi}.
\end{aligned}
\tag{4.37}
$$

We now write the finite-volume, conservative update formula that results from (4.24), with some modifications. It is more convenient to use finite difference quantities from the grid than the derived partial derivative terms such as ξ_x, η_y, *etc.*

Beginning with

$$\tilde{Q}^{n+1}_{i,j} = \tilde{Q}^{n}_{i,j} - \frac{\Delta t}{\Delta \xi}\left(\tilde{F}_{i+1/2,j} - \tilde{F}_{i-1/2,j}\right) - \frac{\Delta t}{\Delta \eta}\left(\tilde{G}_{i,j+1/2} - \tilde{G}_{i,j-1/2}\right),$$

we multiply the $\tilde{F}$ quantities by $\Delta\eta/\Delta\eta$ and the $\tilde{G}$ quantities by $\Delta\xi/\Delta\xi$ to put all terms over a common denominator:

$$\begin{aligned}\tilde{Q}^{n+1}_{i,j} = \tilde{Q}^{n}_{i,j} - \frac{\Delta t}{\Delta\xi\Delta\eta}\Big(&\left(\tilde{F}\,\Delta\eta\right)_{i+1/2,j} - \left(\tilde{F}\,\Delta\eta\right)_{i-1/2,j} \\ &+ \left(\tilde{G}\,\Delta\xi\right)_{i,j+1/2} - \left(\tilde{G}\,\Delta\xi\right)_{i,j-1/2}\Big).\end{aligned}$$

Then we substitute $\tilde{q} = q\,J^{-1}$ and divide through by J^{-1} to get

$$\begin{aligned}Q^{n+1}_{i,j} = Q^{n}_{i,j} - \frac{\Delta t}{J^{-1}\Delta\xi\Delta\eta}\Big(&\left(\tilde{F}\,\Delta\eta\right)_{i+1/2,j} - \left(\tilde{F}\,\Delta\eta\right)_{i-1/2,j} \\ &+ \left(\tilde{G}\,\Delta\xi\right)_{i,j+1/2} - \left(\tilde{G}\,\Delta\xi\right)_{i,j-1/2}\Big).\end{aligned}$$

Finally, we make substitutions from (4.10), (4.29), (4.30), (4.25), and (4.26) to get

$$\begin{aligned}Q^{n+1}_{i,j} = Q^{n}_{i,j} - \frac{\Delta t}{\kappa}\Big(&(\mathbf{F}\cdot\boldsymbol{\Delta s}_{\eta})_{i+1/2,j} - (\mathbf{F}\cdot\boldsymbol{\Delta s}_{\eta})_{i-1/2,j} \\ &+ \left(\mathbf{F}\cdot\boldsymbol{\Delta s}_{\xi}\right)_{i,j+1/2} - \left(\mathbf{F}\cdot\boldsymbol{\Delta s}_{\xi}\right)_{i,j-1/2}\Big).\end{aligned} \tag{4.38}$$

We recognize this equation as the finite volume form of the general integral conservation law (1.5) on a quadrilateral cell. This form differs from the usual curvilinear coordinate update for finite difference methods in that the quantities that are differenced are calculated in units of *total flux per unit time*,

$$\frac{[q]}{[t]}$$

rather than *flux per unit distance per unit time*,

$$\frac{[q]}{[x][t]}.$$

This form is more appropriate for use in composite grid methods since fluxes will then be calculated with the same scale on all cell faces.

To summarize: on curvilinear or quadrilateral grids, fluxes and their corrections can be calculated on cell faces much as they would be on a uniform grid. The main difference is that Riemann Problem solutions and fluxes for each cell face must be calculated in a locally rotated coordinate system, normal to the face, and the resulting fluxes must then be multiplied by the length of the face to have proper dimensions.

It is helpful to look at how a particular PDE would be solved on a curvilinear grid. Consider the case of plane advection in an incompressible flow field with no source terms. If we represent the velocity field as

$$\boldsymbol{V} = \begin{pmatrix} u(\boldsymbol{x}) \\ v(\boldsymbol{x}) \end{pmatrix},$$

then plane advection can be described by the equation

$$q_t + [u\ q]_x + [v\ q]_y = 0 \qquad \text{where} \quad \nabla \cdot \boldsymbol{V} = 0. \tag{4.39}$$

As we did above in the general case, we may transform this to

$$\tilde{q}_t + \left(\frac{\hat{u}\ \tilde{q}}{J^{-1}}\right)_\xi + \left(\frac{\hat{v}\ \tilde{q}}{J^{-1}}\right)_\eta = 0, \tag{4.40}$$

where

$$\begin{aligned} \tilde{q} &= q\ J^{-1} \\ \hat{u} &= u\ \xi_x + v\ \xi_y \\ &= \tilde{u}\ J^{-1} \\ \hat{v} &= u\ \eta_x + v\ \eta_y \\ &= \tilde{v}\ J^{-1}. \end{aligned}$$

This makes sense because on a non-rectangular grid, one cannot cancel out cell-area from the system by assuming that cell-sizes are the same everywhere. One must actually solve for conservation of mass, density times area. Since we would prefer to calculate using q directly, thus avoiding numerical errors that might occur from transformation to $\tilde{q}$, we rewrite (4.40) as follows, assuming for now that J^{-1} is constant:

$$q_t + (\hat{u}\ q)_\xi + (\hat{v}\ q)_\eta = 0. \tag{4.41}$$

It can be shown easily that if $\boldsymbol{V}$ is divergence-free in $\boldsymbol{x}$, $\tilde{\boldsymbol{V}}$ is also divergence-free in $\boldsymbol{\xi}$. In this case (4.36) and (4.37) simplify to

$$\begin{aligned} \hat{u}\Delta\eta &= \boldsymbol{V} \cdot \boldsymbol{\Delta s}_\eta \\ \hat{v}\Delta\xi &= \boldsymbol{V} \cdot \boldsymbol{\Delta s}_\xi. \end{aligned}$$

We will normally leave off indices on J^{-1}. When it is being used to update a cell quantity, J^{-1} will be assumed to have the same indices as the cell. In that case, it may be calculated as the value of $J^{-1}(\boldsymbol{x})$ at the physical coordinates corresponding to the computational center of the cell, which agrees to second order with the cell area when $\Delta\xi\Delta\eta = 1$. When associated with an interface quantity (*e.g.*, a flux), $J^{-1}(\boldsymbol{x})$ will be assumed to be evaluated at the midpoint of that interface. For smoothly varying grids, that value agrees to second order with the average of $J_{i,j}^{-1}$ in the cells on each side of the interface.

Chapter 5

THE *H*-BOX METHOD ON AN EMBEDDED CHIMERA GRID

In this chapter the steps necessary for calculation of stable, accurate and consistent fluxes on interfaces of cells at sub-grid interfaces are described. We begin with discussion of generation of the embedded Chimera grid data structures that store information necessary for these flux calculations.

5.1 Construction of embedded Chimera data structures

To implement a multidimensional analog of the one-dimensional h-box method on an embedded Chimera grid, one must first generate an embedded Chimera grid. We denote the data structure containing information about structured sub-grid geometry as a *MapGrid*, to indicate that the structured grid is a general curvilinear, or *mapped*, grid. Embedded Chimera grid data, containing pointers to MapGrid structures and all information about the connectivity of sub-grids and the geometry of irregular cut-cells contained in a structure called a *CompGrid*[1].

5.1.1 CompGrid geometry and connectivity generation

The h-box-CLAW method requires piece-wise reconstruction of gradient information in both irregular cut-cells and structured grid cells neighboring cut-cells. Piece-wise linear reconstruction requires connectivity information for cell faces on irregular cells. The simplest type of connectivity information is to have two cells on either side of an edge (henceforth called a *CompGrid face*), together with an orientation for the face. Therefore, the data structure for MapGrid intersection stores each CompGrid face segment that has two distinct neighbors separately; *i.e.*, a CompGrid face is not allowed to have more than two neighbors. The CompGrid face is also associated with oriented connectivity information (MapGrid and CompGrid cells on face, vertices on face), the normal vector on the face, and the index of the structured sub-grid face on which the CompGrid face sits. Note that cut-cells may have several faces which are not cut by the outer boundary of the embedded MapGrid, but even these faces are stored as CompGrid faces, and MapGrid fluxes on those faces must be replaced also. This last step is necessary to ensure that the stable flux algorithm will be implemented on *all* faces of a small cell. The vertices at the endpoints of CompGrid faces are *CompGrid vertices*, and one may

[1] The names MapGrid and CompGrid are deliberately chosen to be similar to those used in the CMPGRD [11] and Overture [10] software packages.

observe trivially that in two dimensions each `CompGrid` vertex is shared between two or more `CompGrid` faces. `CompGrid` vertices are indexed and their positions are stored.

To simplify `MapGrid` intersection in this research, we assume that the edges of all cells in all sub-grids are straight lines. The edges of body-fitted grids therefore cut directly into the underlying grid. In general, this requirement may lead to cut-cells which are not convex. We are concerned with polygon convexity because extremely simple and robust intersection algorithms can be used when both polygons are convex. In this research we make some mild restrictions to ensure each non-convex cut-cell is the difference of two convex parts. However, these straight-edge and convexity restrictions are not requirements of the method, and may be modified in future work as necessary.

A `CompGrid` data structure is generated recursively by successive embedding of `MapGrids`. For the purpose of this discussion, we assume that there are only two `MapGrids` in the composite, and that `MapGrid`-2 is embedded into `MapGrid`-1. As `MapGrid`-2 is laid down, its outer edges cut into cells in `MapGrid`-1. Cells in `MapGrid`-1 that are completely inside the boundaries of `MapGrid`-2 are marked as non-physical. Then `MapGrid`-2's outer edge is traced through individual cut-cells, and the geometries and connectivity of `CompGrid` cells, faces and vertices are stored.

5.1.2 h-box and ghost cell construction

Once the basic geometry of grid-intersection has been laid out, an array of h-box polygons must be generated on each `CompGrid` face. We will be interested mainly in how those h-boxes overlap `MapGrid` and `CompGrid` cells—overlap information will be used for data reconstruction.

It is also necessary to mark certain cells in both `MapGrid`-1 and `MapGrid`-2 as *ghost* cells; *i.e.*, data must be reconstructed in those cells. Ghost cell data are used as boundary conditions for unmasked portions of `MapGrids`.

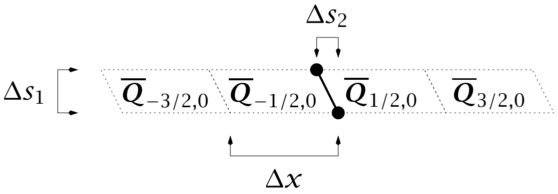
The intersections of ghost cells, and h-boxes with physical cells are computed during grid generation. The centroids and areas of those intersections are stored. Overlap information also includes the grid number and index of the physical cell being overlapped.

The algorithm used to reconstruct piece-wise gradient information in irregular cells and the mapped-grid cells adjacent to them is described in Section A.2.

*5.2 h-**box structures on a non-aligned** `CompGrid` **face***

The h-box structures on `CompGrid` faces are more complicated in two dimensions than they were in one dimension. In 1-D, we simply store 2 h-boxes to the left and right of each face (a total of 4 h-boxes). The h-boxes adjacent to the face are used for Godunov flux calculation, and the outer h-boxes are used to for flux-limiting in higher-order flux-correction. For non x- or y-aligned `CompGrid` faces in two dimensions, however, we require 4 h-boxes in each coordinate direction for Godunov fluxes and higher-order normal corrections. These h-boxes were shown in Chapter 1 in Figure 1.11. That figure illustrates that we really are reconstructing data in a uniform pseudo-grid around each `CompGrid` face in order to extend the numerical domain of dependence. It is therefore logical to call this new `CompGrid`-face-centered grid a *`FaceGrid`*. As in one dimension, it will be necessary to refer to quantities in both the cells and on the interfaces between cells of the `FaceGrid`. It is convenient to adopt an index coordinate system for the `FaceGrid` that is locally centered on the `CompGrid`-face, which is given index $(0,0)$. Then, for example, we may refer without ambiguity to $\overline{\boldsymbol{Q}}_{1/2,0}$ to indicate the x-aligned `FaceGrid` cell immediately to the right of the `CompGrid` face. As in Chapter 2, we will mark `FaceGrid` quantities with an over-bar to indicate that they are connected with a `CompGrid` face.

The quantities that will be used for coordinate-aligned corrections are illustrated in Figure 5.1.



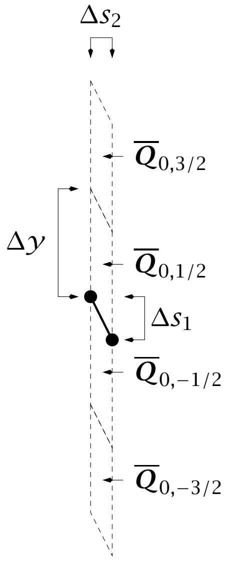


Figure 5.1: Coordinate aligned `FaceGrid`s on a non-aligned face.

5.3 Normal fluxes on a non-aligned `CompGrid` face

Here we describe the generalized multidimensional h-box algorithm on a non-grid-aligned interface. Observe that for corrections in a single coordinate, *i.e.*, to model $\boldsymbol{q}_x$, $\boldsymbol{q}_y$, $\boldsymbol{q}_{xx}$, and $\boldsymbol{q}_{yy}$ derivatives, the algorithm is essentially the same as the one-dimensional form shown in Chapter 2. We call these "normal" flux-difference corrections, even though the coordinates are not necessarily aligned with a `CompGrid` face, since these corrections correspond to the normal corrections in `CLAWPACK` on a Cartesian grid.

5.3.1 "Donor-cell upwind" fluxes on a non-aligned `CompGrid` face

The flux-difference corrections corresponding to first-order (DCU) corrections on the normal `FaceGrid` h-boxes, are nearly the same as those in Algorithm 4.1. We denote the x- and y-components of the flux through a `CompGrid` face as $\overline{\boldsymbol{F}}$ and $\overline{\boldsymbol{G}}$, respectively. Note that for a non-aligned face, there are both x and y jumps on the same face (and with the same index). Another difference between the $[\cdot]$ notation on `FaceGrid`s and that on `MapGrid`s is that since `FaceGrid` cells do not correspond to physical cells, they are given $1/2$ indices to reference their locations with respect to an index origin on the *face*:

$$\left(\left[\overline{\boldsymbol{Q}}\right]_x\right)_{i,j} = \overline{\boldsymbol{Q}}_{i+1/2,j} - \overline{\boldsymbol{Q}}_{i-1/2,j} \qquad \left(\left[\overline{\boldsymbol{Q}}\right]_y\right)_{i,j} = \overline{\boldsymbol{Q}}_{i,j+1/2} - \overline{\boldsymbol{Q}}_{i,j-1/2}.$$

We denote the total flux through the non-aligned face as $\check{\boldsymbol{F}}$. It is the combination of the component fluxes $\overline{\boldsymbol{F}}$ and $\overline{\boldsymbol{G}}$ weighted by the respective h-box cross-sections, Δs_1 and Δs_2. We use the $\boldsymbol{\Delta s}$ notation here, as in Section 4.3, to refer to a vector normal to the `CompGrid` face with magnitude equal to the length of the face:

$$\boldsymbol{\Delta s} = \begin{pmatrix} \Delta s_1 \\ \Delta s_2 \end{pmatrix} = \begin{pmatrix} \Delta y \\ -\Delta x \end{pmatrix} \tag{5.1}$$

where

$$\Delta x = x^{\textcircled{2}} - x^{\textcircled{1}} \tag{5.2}$$
$$\Delta y = y^{\textcircled{2}} - y^{\textcircled{1}}. \tag{5.3}$$

We also use the flux-matrix notation from Section 4.3.1:

$$\overline{\mathbf{F}} = \begin{pmatrix} \overline{\boldsymbol{F}} & \overline{\boldsymbol{G}} \end{pmatrix} \tag{5.4}$$
$$\overline{\mathbf{F}} \cdot \boldsymbol{b} = b_1\, \overline{\boldsymbol{F}} + b_2\, \overline{\boldsymbol{G}} \tag{5.5}$$

We are now prepared to formulate the first-order flux algorithm for `CompGrid` faces:

Algorithm 5.1 (Godunov fluxes on `CompGrid`).

for all $k = 1 : N_{\texttt{CompGrid face}}$ **do** {Initialize fluxes on all `CompGrid` faces}

$$\overline{\boldsymbol{F}}^k \leftarrow \boldsymbol{f}(\overline{\boldsymbol{Q}}^k_{1/2,0})\, \Delta s_1$$

{flux from right `FaceGrid` cell for x-component}

$$\overline{\boldsymbol{G}}^k \leftarrow \boldsymbol{g}(\overline{\boldsymbol{Q}}^k_{0,1/2})$$

{flux from upper `FaceGrid` cell for y-component}

end for

for all `CompGrid` faces **do** {Subtract normal fluctuations to get DCU fluxes}

$$\overline{\boldsymbol{F}}^k \mathrel{-}= (\mathbf{A}^+ \left[\overline{\boldsymbol{Q}}\right]_x)^k_{0,0}$$

$$\overline{\boldsymbol{G}}^k \mathrel{-}= (\mathbf{B}^+ \left[\overline{\boldsymbol{Q}}\right]_y)^k_{0,0}$$

$$\breve{\boldsymbol{F}}^k \leftarrow \begin{pmatrix} \overline{\boldsymbol{F}}^k & \overline{\boldsymbol{G}}^k \end{pmatrix} \cdot \boldsymbol{\Delta s}$$

{Scale flux components by components of face-size}

end for

Algorithm 5.1

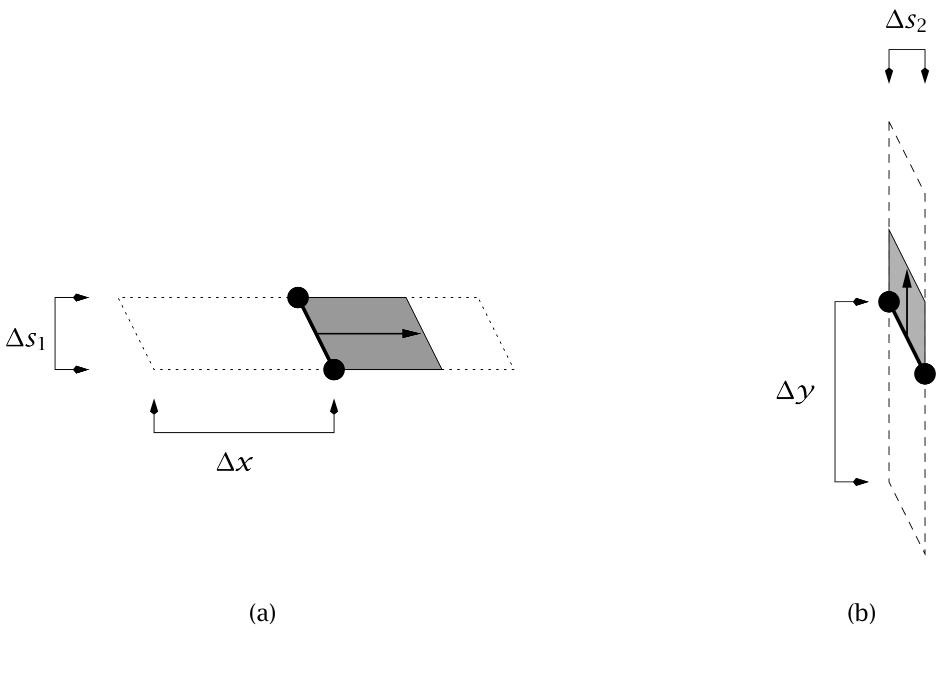


Figure 5.2: (a) DCU correction from x-oriented `FaceGrid` h-box. (b) DCU correction from y-oriented `FaceGrid` h-box.

Analysis following the method of Section 2.3.2 shows that using $\langle 1 \rangle$ fluxes in our numerical $\boldsymbol{F}$ and $\boldsymbol{G}$ fluxes produces solutions with global error of $O(\Delta x)$ and stability up to $\|\mathrm{CFL}\|_1 = 1$; *i.e.*, the sum of the CFL number magnitudes in each direction may not exceed 1. This is the same stability limitation as that of the DCU method on a structured grid, hence the "normal" flux correction terminology. The analysis follows below.

Analysis for `CompGrid` first-order method

In two dimensions, small cells on a non-uniform grid (that is, the cut-cells in our embedded Chimera) do not have just two sides. For the sake of simplicity of argument, consider for now a small cell with just 3 sides, as illustrated in Figure 5.3. We will show

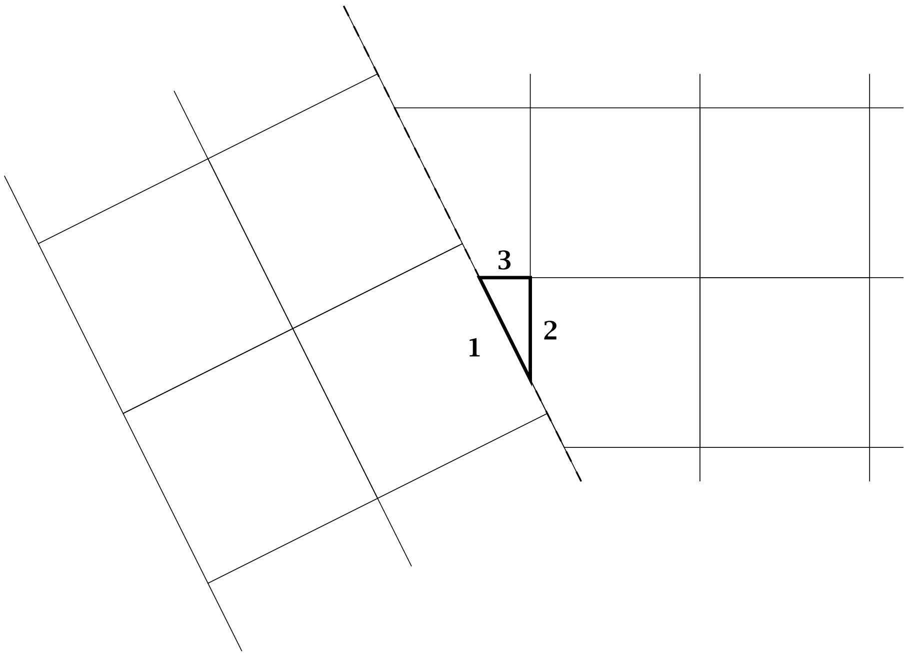


Figure 5.3: Typical small cell on `CompGrid`. Faces on cell are numbered for reference in stability proof. Note that Δs_1 from face **1** is equal to the total height of the cell, while Δs_2 from face **1** is equal to the total width of the cell.

that the finite volume update using fluxes calculated by Algorithm 5.1 is stable and $O(h)$ accurate for uniform advection:

$$q_t + (u\ q)_x + (v\ q)_y = 0.$$

We assume that u and v are constant and positive. Note that in this case the total flux through any `CompGrid` face is just the sum of coordinate-wise upwind fluxes:

$$\check{F}^k = u\ \overline{Q}^k_{-1/2,0}\Delta s_1 + v\ \overline{Q}^k_{0,-1/2}\Delta s_2.$$

On the small `CompGrid` cell (un-indexed for now), the finite-volume update is

$$Q_{\text{cell}}^{n+1} = Q_{\text{cell}}^{n} - \frac{\Delta t}{\kappa} \sum_{\text{faces on cell}} \breve{F}^{\text{face}} \cdot \ (\text{sign of flux on face})\,,$$

where κ is the cell area. The "sign of flux on face" in this formulation refers to the orientation of the face's normal vector $\boldsymbol{n}$: if $\boldsymbol{n}$ points into the cell, the flux has negative sign (*i.e.*, -1) on that cell; if $\boldsymbol{n}$ points out of the cell, the flux has positive sign on that cell. This is simply a generalization of the finite volume formulation on quadrilateral grids (4.6). In the case of the small cell in Figure 5.3, faces **2** and **3** both have positive sign, while face **1** has negative sign.

The resulting update for the small cell in our example is

$$Q_{\text{cell}}^{n+1} = Q_{\text{cell}}^{n} - \frac{\Delta t}{A^{\text{cell}}} \left(u \ \Delta s_1 \ \left(\overline{Q}^{\mathbf{2}}_{-1/2,0} - \overline{Q}^{\mathbf{1}}_{-1/2,0} \right) + v \ \Delta s_2 \ \left(\overline{Q}^{\mathbf{3}}_{0,-1/2} - \overline{Q}^{\mathbf{1}}_{0,-1/2} \right) \right),$$

where we take advantage of the fact that Δs_1 is the same on both face **1** and face **2**, and Δs_2 is the same on both face **1** and face **3**. The difference of the `FaceGrid` h-boxes in each coordinate direction leads to telescopic cancellation. In the x-direction, the difference of x-oriented h-boxes becomes

$$\overline{Q}^{\mathbf{2}}_{-1/2,0} - \overline{Q}^{\mathbf{1}}_{-1/2,0} = \frac{A^{\text{cell}}}{\Delta s_1 \ \Delta x} \ \left(Q_{\text{cell}} - \hat{Q}_{-\Delta x} \right),$$

where we use notation analogous to the one-dimensional case, $\hat{Q}_{-\Delta x}$ to describe the reconstructed cell average in an image of the small cell offset distance Δx to the left. The scaling $A^{\text{cell}}/(\Delta s_1 \ \Delta x)$ is the ratio of the small cell area to the area of x-`FaceGrid` h-boxes. Similarly, the difference of y-`FaceGrid`-box cell averages becomes

$$\overline{Q}^{\mathbf{3}}_{0,-1/2} - \overline{Q}^{\mathbf{1}}_{0,-1/2} = \frac{A^{\text{cell}}}{\Delta s_2 \ \Delta y} \ \left(Q_{\text{cell}} - \hat{Q}_{-\Delta y} \right).$$

A geometrical interpretation of these telescopic cancellations is shown in Figure 5.4.

The result of telescopic cancellation is that the update formula becomes

$$\begin{aligned} Q_{\text{cell}}^{n+1} &= Q_{\text{cell}}^{n} - \frac{\Delta t}{A^{\text{cell}}} \left(\frac{u \ \Delta s_1 \ A^{\text{cell}}}{\Delta s_1 \ \Delta x} \ \left(Q_{\text{cell}} - \hat{Q}_{-\Delta x} \right) + \frac{v \ \Delta s_2 \ A^{\text{cell}}}{\Delta s_2 \ \Delta y} \ \left(Q_{\text{cell}} - \hat{Q}_{-\Delta y} \right) \right) \\ &= Q_{\text{cell}}^{n} - \frac{\Delta t}{\Delta x \ \Delta y} \ \left(u \ \Delta y \ \left(Q_{\text{cell}} - \hat{Q}_{-\Delta x} \right) + v \ \Delta x \ \left(Q_{\text{cell}} - \hat{Q}_{-\Delta y} \right) \right). \end{aligned} \tag{5.6}$$

As in the case of the one-dimensional h-box method, the resulting update formula for uniform advection no longer contains any small cell-area quantities in the denominator. In fact, the update formula is consistent with the donor-cell upwind method on a uniform grid, for which $\hat{Q}_{-\Delta x} = Q_{i-1,j}$ and $\hat{Q}_{-\Delta y} = Q_{i,j-1}$.

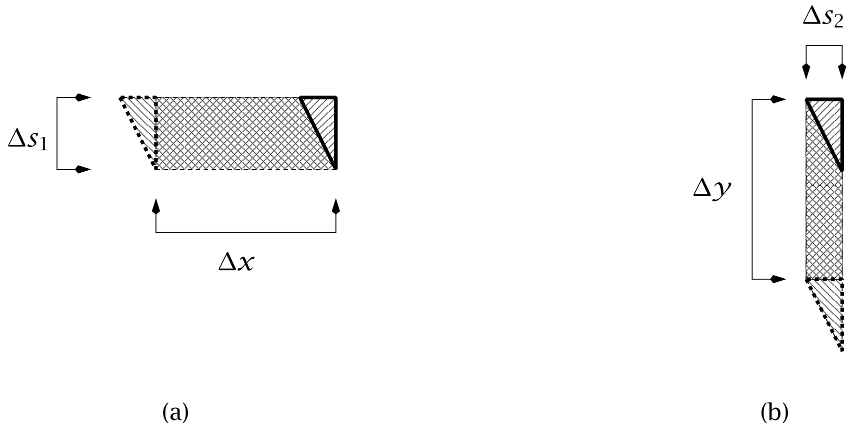


Figure 5.4: Overlap of FaceGrid boxes in first-order update. (a) $\overline{Q}$ in left x-FaceGrid-box from **1** has negative hatching; $\overline{Q}$ in left x-FaceGrid-box from **2** has positive hatching. The difference of FaceGrid h-box values cancels in the cross-hatched overlap region. (b) Similar cancellation for y-FaceGrid h-boxes.

For more general small cells, the sum of all left x-FaceGrid h-boxes and lower y-FaceGrid h-boxes will cancel in a similar fashion; therefore, without loss of generality, the same update formula using offset-reconstructed cell-averages holds true for *any* small cell. In the case of uniform advection, we define coordinate-wise CFL numbers $\nu_x = u\Delta t\Delta y/(\Delta x\Delta y)$ and $\nu_y = v\Delta t\Delta x/(\Delta x\Delta y)$ and reduce the general finite-volume update formula to

$$\begin{aligned} Q_{\text{cell}}^{n+1} &= Q_{\text{cell}}^{n} - \nu_x \left(Q_{\text{cell}} - \hat{Q}_{-\Delta x}\right) - \nu_y \left(Q_{\text{cell}} - \hat{Q}_{-\Delta y}\right) \\ &= (1 - \nu_x - \nu_y)\, Q_{\text{cell}}^{n} + \nu_x\, \hat{Q}_{-\Delta x} + \nu_y\, \hat{Q}_{-\Delta y}. \end{aligned} \tag{5.7}$$

It is apparent that for uniform advection in which $\nu_x + \nu_y \le 1$, the new small cell value will be a convex combination of the old cell value and the reconstructed values $\hat{Q}_{-\Delta x}$ and $\hat{Q}_{-\Delta y}$. To show stability of the method in small cells, we will assume that we apply the first-order h-box method in a general unstructured mesh (with cells indexed by k) in which cell sizes may vary from 0 to uniform $\Delta x\Delta y$, and the same uniform Δx and Δy will be used for all h-boxes on all cell faces. This is analogous to the worst-case non-uniform grid used for the one-dimensional proofs in Chapter 2.

Theorem 5.1 (Stability in L^1 for piece-wise constant reconstruction). *Assume that the reconstruction $\hat{q}$ has piece-wise constant value equal to the cell-average inside any cell; that the L^1 norm $\sum_k \left|Q_k^n\right| < \infty$; and that $Q_k^n > 0$ for all k. Then the cell update Q_k^{n+1} using fluxes from Algorithm 5.1 satisfies $\sum_k \left|Q_k^{n+1}\right| < \infty$ when $\nu_x + \nu_y \le 1$.*

Proof. Under our assumptions, we can sum (5.7) over all cells and use the triangle inequality to get

$$\sum_k \left|Q_k^{n+1}\right| \leq (1 - \nu_x - \nu_y) \sum_k |Q_k| + \nu_x \sum_k \left|\hat{Q}^k_{-\Delta x}\right| + \nu_y \sum_k \left|\hat{Q}^k_{-\Delta y}\right|.$$

By construction, and using the fact that $Q_k > 0$, we have that

$$\sum_k \left|\hat{Q}^k_{-\Delta x}\right| = \sum_k \left|Q_k^n\right| \qquad \text{and} \qquad \sum_k \left|\hat{Q}^k_{-\Delta y}\right| = \sum_k \left|Q_k^n\right|$$

Therefore,

$$\begin{aligned} \sum_k \left|Q_k^{n+1}\right| &\leq (1 - \nu_x - \nu_y) \sum_k |Q_k| + \nu_x \sum_k |Q_k| + \nu_y \sum_k |Q_k| \\ &\leq \sum_k \left|Q_k^n\right|. \quad \blacksquare \end{aligned}$$

We conclude that the L^1 stability limitation of the DCU method using h-boxes is the same as that of the method on a uniform grid with spacings $\Delta x, \Delta y$.

Our proof of Theorem 5.1 rests on the assumption that the same update formula is used everywhere. In a `CompGrid`, however, we may be using this update on small cells while a differently-oriented update (*not* along x and y coordinates) is being used in an adjacent grid. But by proving stability for a worst-case grid, as we did for the 1D case, we have tested the stability of the h-box method in the worst possible case. When h-box DCU fluxes are used to update adjoining body-fitted grid cells, we expect one of two cases:

- h-box sizes are smaller than adjoining body-fitted grid cells. Therefore the time-step limitation from the h-box method will be the most restrictive, and time-step restrictions in body-fitted grid cells will be satisfied with room to spare.

- h-box sizes are larger than adjoining body-fitted grid cells. The time-step restriction in this case will come from body-fitted grid cells, so h-box fluxes will be well within their limitations. Body-fitted cells will receive information from a much larger upwind region than they would otherwise, and therefore are much less likely to be unstable.

The total variation functional in two dimensions is defined for a piece-wise constant $\hat{q}$ as follows:

$$\begin{aligned} \mathrm{TV}(\hat{q}) &= \int |\hat{q}_x| + \left|\hat{q}_y\right| \; dx\, dy \\ &= \sum_{\text{faces } j} \left|[\hat{q}]_j\right| \; \left|(\Delta s_1)_j\right| + \left|(\Delta s_2)_j\right|. \end{aligned} \tag{5.8}$$

In other words, since there is no variation inside a cell with piece-wise constant reconstruction, the total variation along a face is the L^1 norm of the face-length multiplied by the absolute value of the jump across the face. The problem with this definition of total variation is that it is not apparent how to use it in conjunction with the finite-volume update formula for non-uniform grids in two dimensions. But the stability result for one dimension indicates that a possible requirement for stability is that the reconstruction used for h-box values be TVD. This requirement is satisfied trivially by piece-wise constant reconstruction. As in the 1D case, we anticipate that piece-wise linear reconstruction will be required to eliminate bumpy solutions in small cells adjacent to normal-sized cells. Therefore limitation on cell-gradients will be required for piece-wise linear reconstruction. In our two-dimensional tests, however, we find that, for smooth solutions, average gradients perform quite well without extra limiting. But in the case of discontinuous solutions, spurious bumps form unless we use a min-mod-type limiter of the form described in Section A.2.2.

Berger and LeVeque's h-box implementation

We may infer from the results above that the implementation of the h-box method used by Berger and LeVeque [6–8],a rotated version of the method just discussed, cannot have greater stability than a donor-cell upwind scheme on a regular grid. In the rotated coordinate system near the embedded boundary used by Berger and LeVeque, we have rotated-h-box CFL numbers ν_n and ν_τ, in the normal and tangential directions, respectively. The stability restriction on a first-order rotated-h-box solution that must be satisfied becomes $|\nu_n| + |\nu_\tau| \leq 1$. One would therefore expect instabilities to result in the case of transverse propagation near the boundary, which could occur, for example, in solving the linear acoustic equation when acoustic waves bounce off a solid. In the application studied by Berger and LeVeque, supersonic gas flow modeled by the Euler equations, normal and tangential CFL numbers near the boundary will be dominated by tangential flow, and thus this particular grid-orientation-dependent instability may have escaped notice. However, our stability result indicates that the Berger and Leveque h-box implementation should be tested in a very simple case—acoustic waves bouncing off a 45° embedded boundary.

5.3.2 $\langle 2 \rangle$ fluxes on a non-aligned `CompGrid` face

In a similar way to the previous section, we can define the $\langle 2 \rangle$ corrections through the non-aligned face in an entirely analogous way to Algorithm 4.3. Note that here we make the definitions

$$\boldsymbol{w}_x^p = (\left(\boldsymbol{l}_x^{p\,T} \left[\overline{\boldsymbol{Q}}\right]_x\right) \boldsymbol{r}_x^p)_{0,0} \qquad\qquad \boldsymbol{w}_y^\sigma = (\left(\boldsymbol{l}_y^{\sigma\,T} \left[\overline{\boldsymbol{Q}}\right]_y\right) \boldsymbol{r}_y^\sigma)_{0,0}.$$

These x and y waves are limited using the waves at h-box interfaces one unit Δx or Δy in the upwind direction from the `CompGrid` face. For example, if

$$\operatorname{sign}(\Delta s_1)\ (\boldsymbol{V} \cdot \boldsymbol{\Delta s}) > 0,$$

$\boldsymbol{w}_x^p$ is limited using

$$(\left(\boldsymbol{l}_x^{pT}\ \left[\overline{\boldsymbol{Q}}\right]_x\right)\ \boldsymbol{r}_x^p)_{-1,0}.$$

Again, note in Algorithm 5.2 that the total second-order correction to the flux through the `CompGrid` face uses a componentially-weighted combination of the flux corrections in each direction.

Algorithm 5.2 (Second-order corrections on `CompGrid`).

for all $k = 1 : N_{\texttt{CompGrid}\ \text{faces}}$ **do** {Higher order correction}

$$\overline{\boldsymbol{F}}^k \mathrel{+}= \frac{1}{2} \sum_{p=1}^{M_w} \left|\lambda_x^p\right| \left(1 - \frac{\Delta t\ \Delta s_1}{\Delta x \Delta y} \left|\lambda_x^p\right|\right) \widehat{\boldsymbol{w}}_x^p$$

{where $\widehat{\boldsymbol{w}}_x^p$ is the limited form of $\boldsymbol{w}_x^p$}

$$\overline{\boldsymbol{G}}^k \mathrel{+}= \frac{1}{2} \sum_{\sigma=1}^{M_w} \left|\lambda_y^\sigma\right| \left(1 - \frac{\Delta t\ \Delta s_2}{\Delta x \Delta y} \left|\lambda_y^\sigma\right|\right) \widehat{\boldsymbol{w}}_y^\sigma$$

{where $\widehat{\boldsymbol{w}}_y^\sigma$ is the limited form of $\boldsymbol{w}_y^\sigma$}

$$\breve{\boldsymbol{F}} = \overline{\mathbf{F}}^{\langle 2\rangle}\ \cdot\ \boldsymbol{\Delta s}$$

{Higher-order fluxes used on face instead of DCU fluxes}

end for

Algorithm 5.2

x and y `FaceGrid` h-box cancellations follow as they did in the case of first-order fluxes, and so we do not consider accuracy and stability of this method further. The crucial flux correction that is necessary for both second-order accuracy and stability of the two-dimensional h-box method is the analog of *corner-transport upwind*.

*5.3.3 Transverse corrections (***1T** *fluxes)*

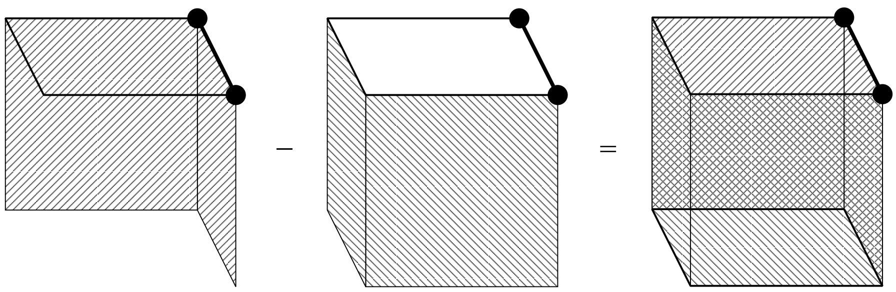
On a non-uniform 2D grid, it is necessary to model $\boldsymbol{q}_{xy}$ and $\boldsymbol{q}_{yx}$ derivatives for a fully second-order scheme. It is not obvious how to do so in a way that is both a logical extension of the one-dimensional method and also consistent with the algorithm on a mapped grid.

We make transverse corrections using a generalized form of the CTU corrections on a uniform grid. Since we will make the corrections by looking backward from the

CompGrid face into a reconstructed data structure to determine incoming waves, we will need information "adjacent" to the FaceGrid h-boxes we have already defined. This requires reconstruction of Q in two additional data structures. For the first, we define *VertGrid* cell values as shown in Figure 5.5. We will call the numerical approximations to the cell averages of $\boldsymbol{q}$ in the 4 cells of the VertGrid $\dot{\boldsymbol{Q}}$ (the dot denoting here a vertex-associated quantity), indexed with respect to the associated vertex at index $(0,0)$. When referring to the two VertGrids on a face, we use a ① or ② superscript to indicate quantities associated with VertGrids ① and ② respectively; *e.g.*, $\dot{\boldsymbol{Q}}^{①}_{1/2,1/2}$, using indices with respect to the vertex (Figure 5.5(c)).

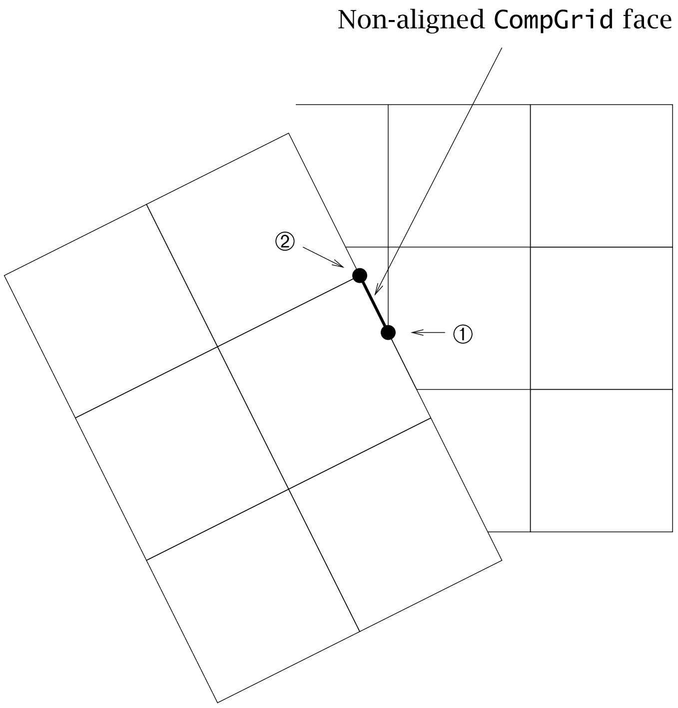
For the second new data structure we generalize the FaceGrid definition to include extra boxes that will be needed for transverse corrections. The first part of the extended FaceGrid, shown in Figure 5.6, has uniform Δx extent in the x-direction. The second part, shown in Figure 5.7 is defined in a similar manner. The indices for both FaceGrid quantities are incremented with respect to the center of the obliquely-aligned face (which has index $(0,0)$), and are hence unambiguous. The complete FaceGrid for a non-aligned face, composed of both x- and y-aligned components, is shown in Figure 5.8.

The CTU corrections will use extended regions of uniform extent in the transverse upwind direction from the FaceGrid h-box used for first-order corrections. We are motivated in this approach by our non-linear 1D method: we found that the upwind flux-difference correction was equivalent to $f'(Q_i^*)\ \left(\overline{Q}_{i+1/2,-h} - \overline{Q}_{i-1/2,-h}\right)$. This is similar to the fluctuation wave that is transversely propagated in the CLAWPACK CTU method. Here we show a schematic of the boxes used for correcting $\overline{F}$ on face **1** with q_{yx} terms:

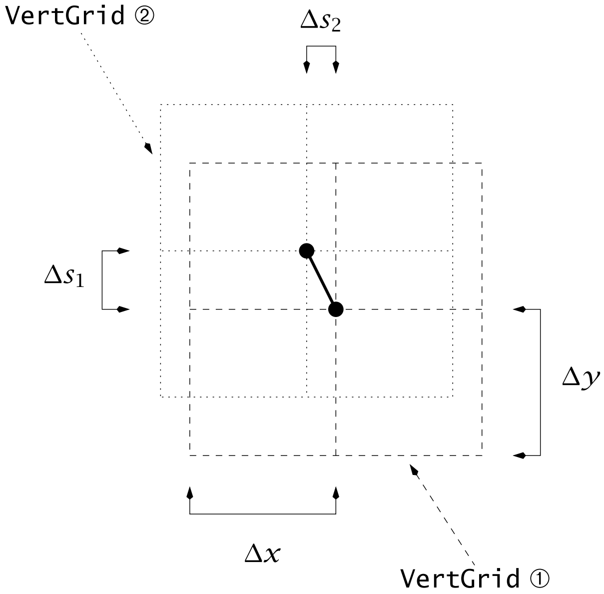


(5.9)

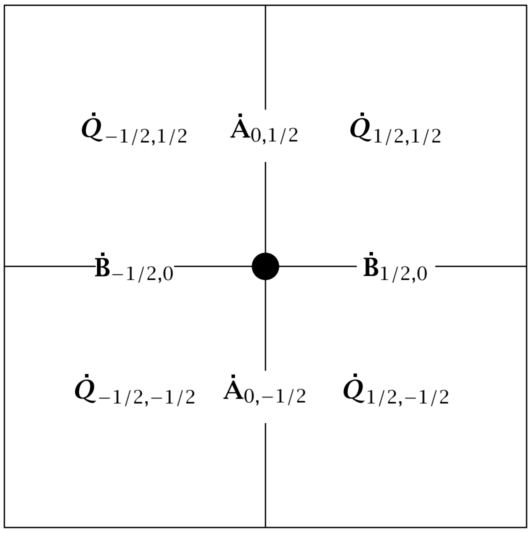
In this "equation" we see that the reconstruction (indicated with positively-sloped hatching) hanging down from the upper sides of FaceGrid box $-1/2, 0$ is the area-weighted sum of $\dot{Q}^{\text{top}}_{-1/2,-1/2}$ and $\overline{Q}_{0,-1/2}$. Similarly, a reconstruction hanging from the bottom sides of $\overline{Q}_{-1/2,0}$ (marked with negatively-sloped hatching) is the area-weighted sum of $\dot{Q}^{\text{bottom}}_{-1/2,-1/2}$ and $\overline{Q}_{-1,-1/2}$. The difference of these two reconstructed quantities effectively cancels in their overlap region (indicated by cross-hatching), to give a scaled difference



(a) Typical non-aligned face arising from grid inset



(b) `VertGrid`s on a non-aligned face



(c) Index notation for `VertGrid` quantities

Figure 5.5: `VertGrid`s associated with `CompGrid` vertices ① and ② on a typical non-aligned `CompGrid` face.

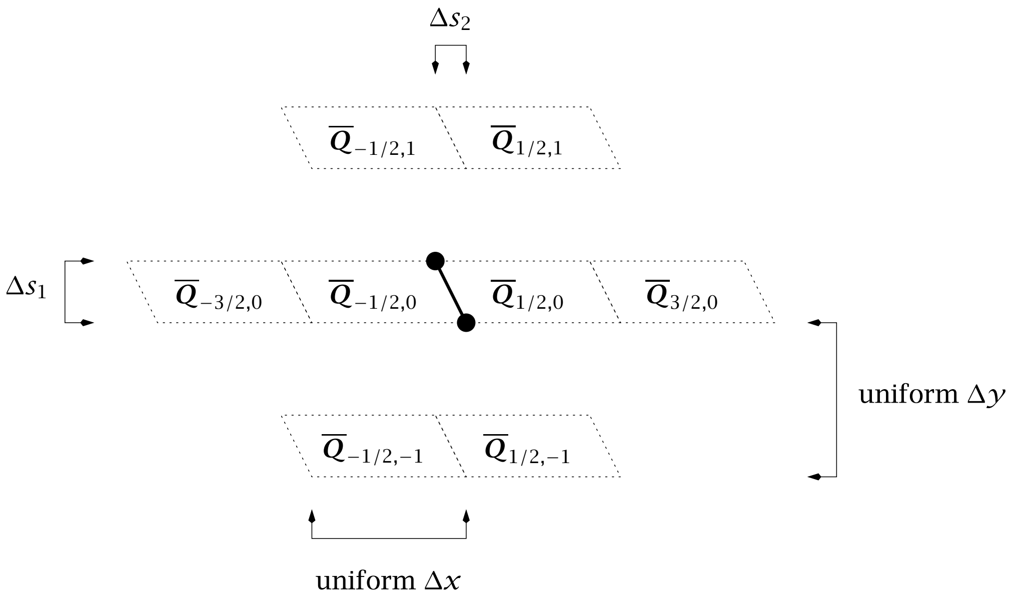


Figure 5.6: x-`FaceGrid` for a non-aligned interface.

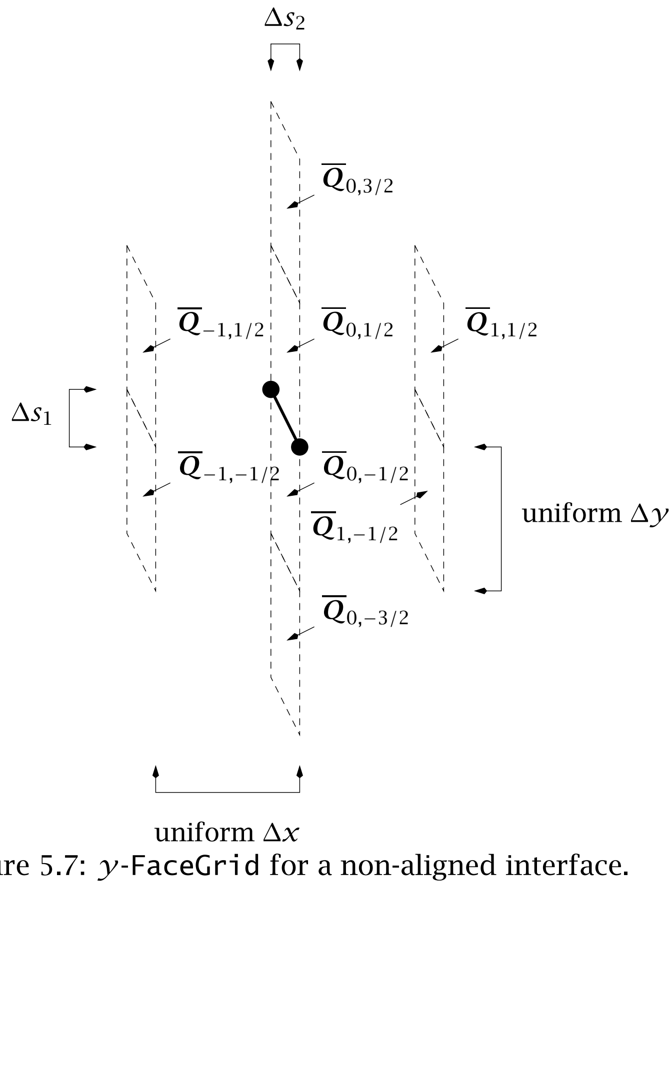


Figure 5.7: y-`FaceGrid` for a non-aligned interface.

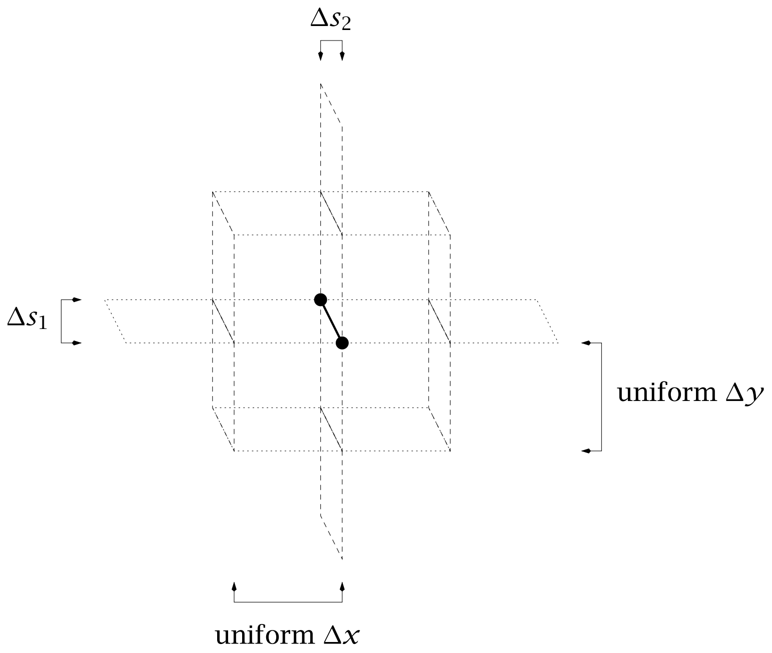


Figure 5.8: Both FaceGrids around a non-aligned face.

of the FaceGrid values, $\left(\overline{Q}_{-1/2,0} - \overline{Q}_{-1/2,-1}\right)$. More precisely,

$$\dot{Q}^{\text{top}}_{-1/2,-1/2} + \frac{\Delta s_2 \Delta y}{\Delta x \Delta y}\overline{Q}_{0,-1/2} - \left(\dot{Q}^{\text{bottom}}_{-1/2,-1/2} + \frac{\Delta s_2 \Delta y}{\Delta x \Delta y}\overline{Q}_{-1,-1/2}\right) = \frac{\Delta s_1 \Delta x}{\Delta x \Delta y}\left(\overline{Q}_{-1/2,0} - \overline{Q}_{-1/2,-1}\right).$$

The composite reconstruction formulation has to be modified slightly if the FaceGrid face has positive slope instead of negative slope as in this case, but the idea remains the same. Therefore we denote the combined VertGrid + FaceGrid h-box values as $\overline{\dot{Q}}^{T}_{-1/2,-1/2}$ and $\overline{\dot{Q}}^{B}_{-1/2,-1/2}$, to indicate that these values are put together with respect to the top and bottom vertices on an x-oriented FaceGrid box. Similarly, the left and right combined values $\overline{\dot{Q}}^{L}_{-1/2,-1/2}$ and $\overline{\dot{Q}}^{R}_{-1/2,-1/2}$ are formed for corrections to y-oriented fluxes. In the latter case, using orientations as in previous figures,

$$\begin{aligned}\overline{\dot{Q}}^{R}_{-1/2,-1/2} - \overline{\dot{Q}}^{L}_{-1/2,-1/2} &= \dot{Q}^{\text{right}}_{-1/2,-1/2} + \frac{\Delta s_1 \Delta x}{\Delta x \Delta y}\overline{Q}_{-1/2,0} - \left(\dot{Q}^{\text{left}}_{-1/2,-1/2} + \frac{\Delta s_1 \Delta x}{\Delta x \Delta y}\overline{Q}_{-1/2,-1}\right)\\ &= \frac{\Delta s_2 \Delta y}{\Delta x \Delta y}\left(\overline{Q}_{0,-1/2} - \overline{Q}_{-1,-1/2}\right).\end{aligned}$$

We can now state the transverse correction algorithm for CompGrid faces. Here we state corrections due only to $\mathbf{B}^+\mathbf{A}^+$ and $\mathbf{A}^+\mathbf{B}^+$ terms, but corrections from the other 3 quadrants are implemented in a similar way

Algorithm 5.3 (Corner transport upwind on `CompGrid` faces).

1: **for all** $k = 1 : N_{\texttt{CompGrid faces}}$ **do** {Corner transport}

2:
$$\overline{\boldsymbol{F}}^{k} -= \frac{\Delta t}{2\Delta x \Delta s_1}\,\mathbf{A}^{+}\left(\Delta x\ \mathbf{B}^{+}\left(\dot{\boldsymbol{Q}}^{T}_{-1/2,-1/2} - \dot{\boldsymbol{Q}}^{T}_{-1/2,-1/2}\right)\right)$$

{Correct $\overline{\boldsymbol{F}}^{k}$ with transverse flux from lower left}

3:
$$\overline{\boldsymbol{G}}^{k} -= \frac{\Delta t}{2\Delta y \Delta s_2}\,\mathbf{B}^{+}\left(\Delta y\ \mathbf{A}^{+}\left(\dot{\boldsymbol{Q}}^{R}_{-1/2,-1/2} - \dot{\boldsymbol{Q}}^{L}_{-1/2,-1/2}\right)\right)$$

{Correct $\overline{\boldsymbol{G}}^{k}$ with transverse flux from lower left}

4:
$$\check{\boldsymbol{F}} = \overline{\mathbf{F}}^{\langle 2\rangle}\ \cdot\ \boldsymbol{\Delta s}$$

{Scale flux components by components of face-size}

5: **end for**

Algorithm 5.3

In the case of uniform advection, we can derive the effective flux corrections:

$$\begin{aligned}\overline{\boldsymbol{F}}^{k}\Delta s_1 &-= \frac{\Delta t \Delta s_1}{2\Delta x \Delta s_1}\, u\ \Delta x\ v \frac{\Delta s_1 \Delta x}{\Delta x \Delta y}\left(\overline{Q}_{-1/2,0} - \overline{Q}_{-1/2,-1}\right)\\ &-= \frac{uv\Delta t \Delta s_1}{2\Delta y}\left(\overline{Q}_{-1/2,0} - \overline{Q}_{-1/2,-1}\right).\end{aligned}$$

Similarly,

$$\begin{aligned}\overline{\boldsymbol{G}}^{k}\Delta s_s &-= \frac{\Delta t \Delta s_2}{2\Delta y \Delta s_2}\, v\ \Delta y\ u \frac{\Delta s_2 \Delta y}{\Delta x \Delta y}\left(\overline{Q}_{0,-1/2} - \overline{Q}_{-1,-1/2}\right)\\ &-= \frac{uv\Delta t \Delta s_2}{2\Delta x}\left(\overline{Q}_{0,-1/2} - \overline{Q}_{-1,-1/2}\right).\end{aligned}$$

Note that these corrections are *not* how the terms are calculated but rather what terms are used (implicitly) in the correction.

When CTU flux corrections are used in addition to DCU fluxes, we can show for uniform advection on an unstructured grid that we have a first-order method with CFL=1 time-step allowed in both x and y coordinates. The basis of this proof is to show that these fluxes give rise to overlap cancellation, with the resulting update being a convex combination of four piece-wise reconstruction $\hat{Q}_{-\Delta x,\Delta y}$, $\hat{Q}_{-\Delta y}$, $\hat{Q}_{-\Delta x}$ and Q.

First, we add the corrections to the DCU update (5.7). We note that `FaceGrid` h-box differences cancel as in the DCU update above. The result is

$$\begin{aligned}Q^{n+1}_{\text{cell}} &= (1 - \nu_x - \nu_y)\ Q^{n}_{\text{cell}} + \nu_x\ \hat{Q}_{-\Delta x} + \nu_y\ \hat{Q}_{-\Delta y}\\ &+ \frac{\Delta t}{A^{\text{cell}}}\left(\frac{A^{\text{cell}}}{\Delta s_1\ \Delta x}\ \frac{u\ v\ \Delta t\ \Delta s_1}{2\Delta y}\left(Q_{\text{cell}} - \hat{Q}_{-\Delta x} - Q_{-\Delta y} + \hat{Q}_{-\Delta x,\Delta y}\right)\right.\end{aligned}$$

$$
\begin{aligned}
&+ \frac{A^{\text{cell}}}{\Delta s_2\ \Delta y}\ \frac{u\ v\ \Delta t\ \Delta s_2}{2\Delta x}\left(Q_{\text{cell}} - \hat{Q}_{-\Delta y} - Q_{-\Delta x} + \hat{Q}_{-\Delta x,-\Delta y}\right)\Bigg)\\
&= (1-\nu_x)\ (1-\nu_y)\ Q^n_{\text{cell}} + \nu_x\ (1-\nu_y)\ \hat{Q}_{-\Delta x}\\
&+ \nu_y\ (1-\nu_x)\ \hat{Q}_{-\Delta y} + \nu_x\ \nu_y\ \hat{Q}_{-\Delta x,-\Delta y}
\end{aligned}
$$

As in the L^1 proof of stability for the DCU (first-order) method, we note that the method will be stable in L^1 for piece-wise constant reconstruction whenever the coefficients are positive. This will be true when $0 \le \nu_x \le 1$ and $0 \le \nu_y \le 1$, which means that by adding corner correction terms in this way, we have increased the stability of the method in the same way as the uniform method.

It should be noted that the effective numerical update with CTU corrections is completely consistent with `CLAWPACK` CTU as the grid becomes uniformly rectangular.

A note on flux redistribution methods

We may infer from our results thus far that the correct stable update for a small cell should be a convex combination of the reconstructed values $\hat{Q}$ over regions of identical shape to the cell, at uniform distances from the cell. Such a combination may be useful in conjunction with a method such as the flux-redistribution scheme of Pember *et al.* [38]; *i.e.*, the stable non-conservative update used to redistribute fluxes could be calculated using a standard upwinded finite-difference method, but using the equally spaced reconstructions $\hat{Q}$ instead of nearest neighbor values.

5.4 The Embedded Chimera Grid Method

Now that the method for calculating fluxes on faces near the grid interface has been described, we can formulate the entire embedded Chimera grid (ECG) method, Algorithm 5.4.

Algorithm 5.4 (Embedded Chimera grid method).

Construct Composite Grid structures.
Initialize data on individual mapped grids.
Initialize velocity on composite structures:
for all `FaceGrid` faces **do**
  Find u and v on `FaceGrid` faces such that discrete divergence-free condition is preserved on `CompGrid` cells.
end for
Initialize q on composite structures:
for all `CompGrid` cells **do**
  Calculate q at centroids
end for
repeat

Apply regular boundary conditions to mapped grids.
Calculate slopes in `CompGrid` cells.
Calculate slopes in mapped cells that affect ghost cells.
Reconstruct q in `VertGrids` and `FaceGrids`.
Reconstruct q in ghost cells. This may overwrite some of the regular boundary conditions on the intersecting grid boundary.
for all `MapGrids` **do**
  Do mapped grid flux calculation.
  Zero out fluxes on marked faces — they will be calculated by `CompGrid` method.
end for
for all `VertGrids` **do**
  Do Flux calculations on `VertGrid` faces
end for
for all `FaceGrids` **do**
  for all $k = 1 : 2$ (iterate over x and y) **do**
    $\langle 1 \rangle$,$\langle 1\mathrm{T} \rangle$, and $\langle 2 \rangle$ corrections for i-th coordinate.
    $\langle 1\mathrm{T} \rangle$ fluxes use pre-calculated `VertGrid` fluxes.
    Scale $\boldsymbol{F}_k$ flux correction by $n_k\ ds$
  end for
  Use total $\boldsymbol{F}$ correction to correct $\boldsymbol{F}$ on underlying `MapGrid` faces.
  Also use total $\boldsymbol{F}$ correction to update ΔQ on neighboring `CompGrid` cells.
end for
for all `MapGrids` **do**
  Do conservative updates on all cells using corrected fluxes.
end for
for all `CompGrid` cells **do**
  Do conservative updates using accumulated ΔQ's.
end for
until $t > t_{\mathrm{end}}$

Algorithm 5.4

Chapter 6

2D NUMERICAL RESULTS

In this chapter we examine the behavior of the h-box-CLAW method for two problems, advection and a two-dimensional form of Burgers' equation. In the case of advection, it is possible to obtain a high-order estimate of the exact solution for comparison. For advection problems using reflecting or zero-extrapolated boundary conditions, or for Burgers' equation problems, the exact solution is more difficult to determine. In those cases we compare our numerical results to results obtained on a finer grid. In all examples below, we use a more finely-discretized version of Chimera grid shown in Figure 6.1 (identical to Figure 1.3 but redisplayed here fore convenience).

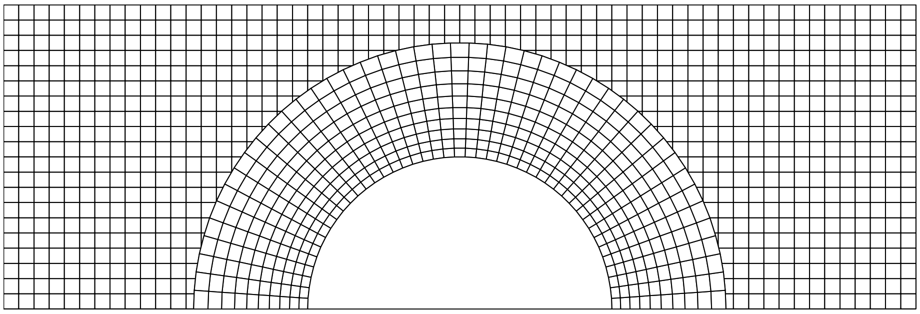


Figure 6.1: Chimera grid around a cylinder.

6.1 Advection with cylindrical flow

A good test for the method's performance for general two-dimensional advection is to use a non-uniform divergence-free velocity field. In the following examples I use flow around a cylinder, which has an exact analytical solution. Given an incompressible flow, the scalar stream function $\psi(x, y)$ can be calculated at any point. We use ψ to set the discretely incompressible velocity field. Note that on a line segment between points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, $\psi(\boldsymbol{x}_2) - \psi(\boldsymbol{x}_1)$ is equal to $\boldsymbol{V}\,\boldsymbol{\Delta s}$, the average component of velocity normal to $\overline{\boldsymbol{x}_1\boldsymbol{x}_2}$, multiplied by the length of the side. By Green's law, the divergence of velocity in a cell will be equivalent to

$$\sum_j (\boldsymbol{V} \cdot \boldsymbol{n})_j$$

where j is an index that iterates over the cell sides. Using the stream-function formulation above, the sum of normal velocity components will cancel to round-off error.

In these examples, ψ is set to

$$\psi(x, y) = \frac{\kappa}{4\pi R} \log\left(\frac{R^2}{x^2 + y^2}\right) + (U\,y - V\,x)\left(1 - \frac{R^2}{x^2 + y^2}\right), \tag{6.1}$$

where R is the cylindrical radius, the velocity at infinity is (U, V) with magnitude $W = \sqrt{U^2 + V^2}$, and counter-clockwise circulation is κ. For the examples we use $\kappa = 0$, $R = 1$, $(U, V) = (1, 0)$. The streamlines of the flow (contours of ψ) on the composite grid are shown in Figure 6.2.

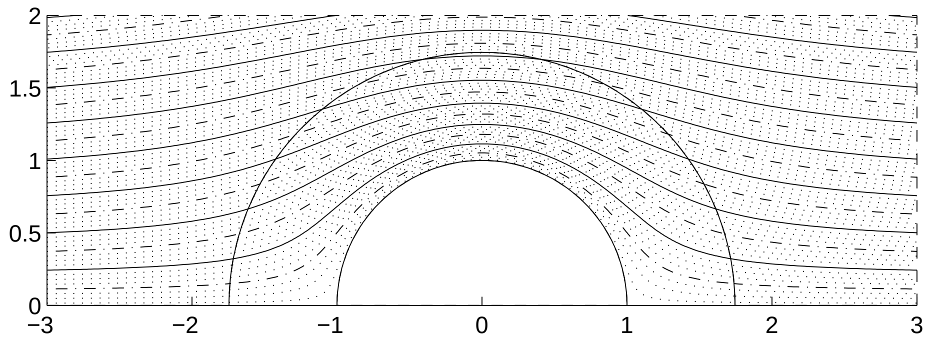


Figure 6.2: Incompressible cylindrical flow, on a half-cylinder embedded Chimera grid.

Results illustrated use a Cartesian grid with 240×80 cells, and cell sizes in the body-fitted grid that are of comparable size at the grid interface. The boundary conditions at the top and right sides of the Cartesian grid are zero-order-extrapolation, while reflecting boundary conditions are used along the y-axis.

For the first set of tests, we track the propagation of a hump of concentration through the cylinder-Chimera. In this case, we can use a Runge-Kutta solver to trace through the velocity field. Starting at the final time at a cell center, we trace back to time 0. The initial condition at the time 0 position determines what the final state should be.

In this test we examine the 2nd-order method with no slope-limiting on gradient reconstruction or flux-limiting. Contours of the initial condition at time 0 are shown in Figure 6.3; the numerical solution at approximately half-way through the test (at time 1.9) is shown in Figure 6.4; and the numerical state at stop-time 3.8 is shown in Figure 6.5. As expected, there is some trailing oscillation due to the un-limited Lax-Wendroff-style fluxes, but the contour lines match quite well across the two grids. The initial cosine hump of concentration ranges from 0 to 1, has a radius of 0.5, and is located at position $(-2, 1)$ at time 0. The same contour levels are used in all three plots.

We chart the numerical errors against h-box size in Table 6.1. In this type of chart,

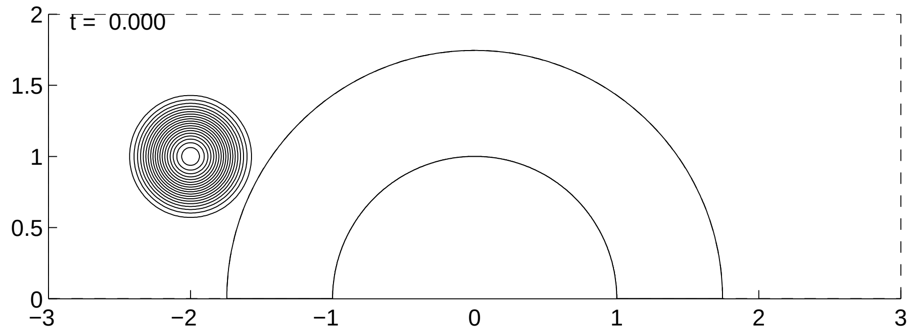


Figure 6.3: Initial contours of the cosine hump concentration.

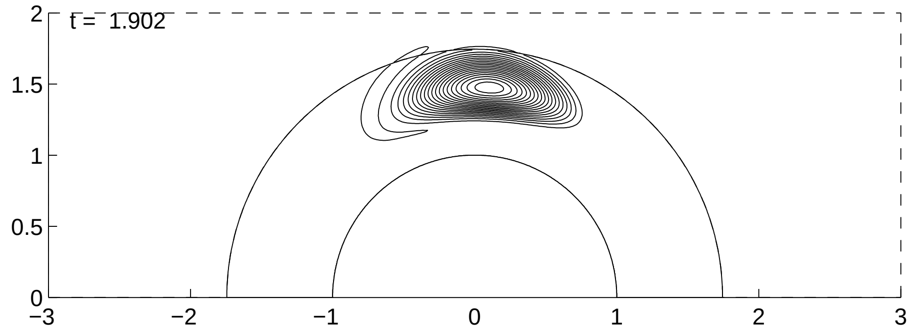


Figure 6.4: Contours of the hump concentration halfway through the test.

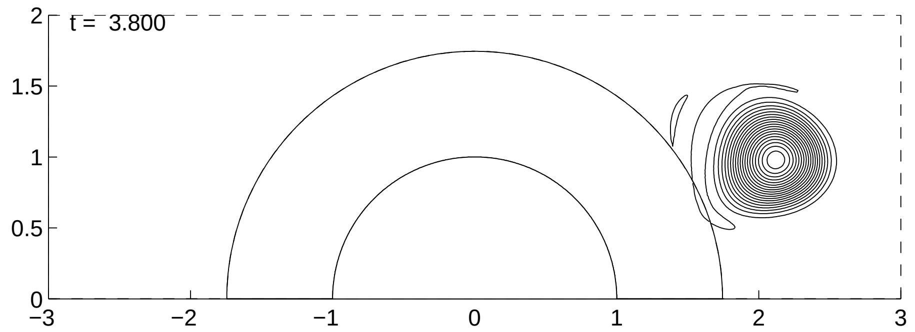


Figure 6.5: Concentration contours at time 3.8.

one expects that the ratio of errors should approach 4.0, since we are using a second-order method and the grid-sizes are being halved at each refinement. We see that as the grids are refined, this ratio in fact exceeds 4.0. This may be explained by the hump traveling through a portion of the cylindrical grid in which the flow is quite well aligned, thus avoiding some transverse propagation errors. In any case, the results are encouraging: even if the error ratio converges to 3.0, it means the method has a rate of approximately 1.73, which is close to second-order accurate.

Table 6.1: Errors for advection of a hump in cylindrical flow.

h	L^1 error	L^1 ratio	L^∞ error	L^∞ ratio
0.1000	2.038E-01	—	4.380E-01	—
0.0500	7.383E-02	2.76	2.217E-01	1.98
0.0250	2.410E-02	3.06	7.737E-02	2.87
0.0125	5.120E-03	4.71	1.524E-02	5.08

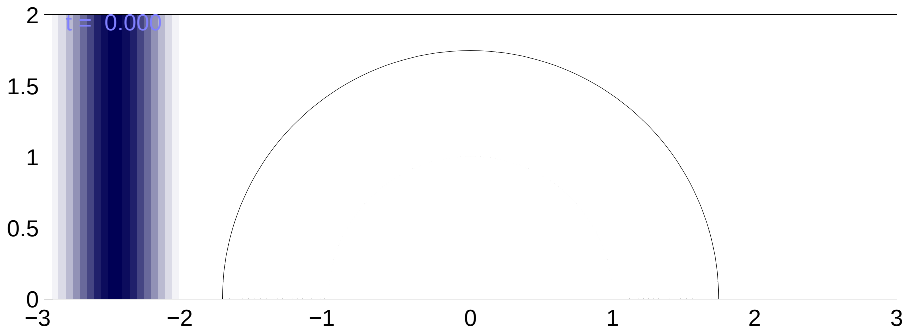
As a comparison, we examine the propagation of a band function with smooth cosine cross section. q ranges from 0 at the left and right sides of the band to 1 at the center. The band is of width 1, centered at position $x = -2.5$ at time 0. This is illustrated in Figure 6.6. We show the solution at two other times in Figures 6.7 and 6.8.

In this case, because of the boundary conditions specified above, we cannot trace back to an exact initial condition—zero-extrapolation BC's imply that after the hump passes the middle of the grid, information is entering the solution that was not initially on a straight vertical band. Therefore, we compare solutions at several levels of refinement against a fine grid solution. We tabulate the errors, compared to a fine grid solution in Table 6.2. In both the fine grid solution and the coarse grid solutions, the VL74 limiter is used. From this table, it appears that the best L^1 ratio implies a convergence

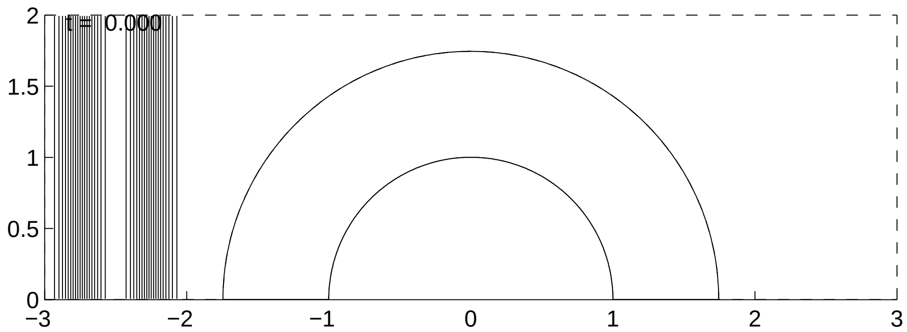
Table 6.2: Errors for advection with cylindrical flow.

h	L^1 error	L^1 ratio	L^∞ error	L^∞ ratio
0.100	2.583E-01	—	4.527E-01	—
0.050	1.030E-01	2.51	3.695E-01	1.23
0.025	3.933E-02	2.62	2.249E-01	1.64

rate of approximately 1.62. But as we can see from Figure 6.9, the largest magnitude of the error is concentrated next to the cylinder, rather than at the grid interface. Since `CLAWPACK` would have even more error in this region because of discretization error, this is a very good result—we expect that with more refinement we would approach convergence rates closer to the hump result above.

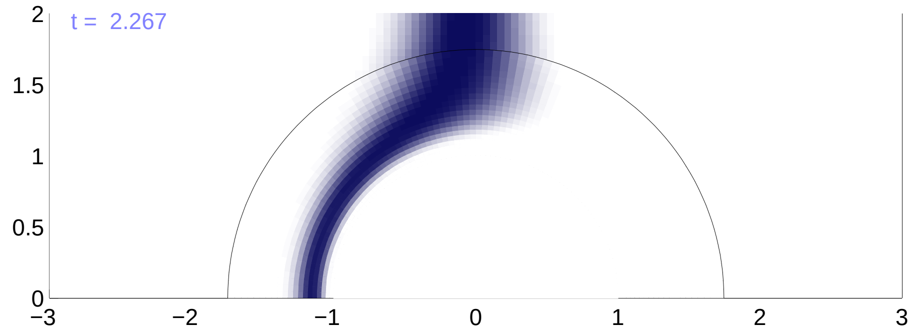


(a)

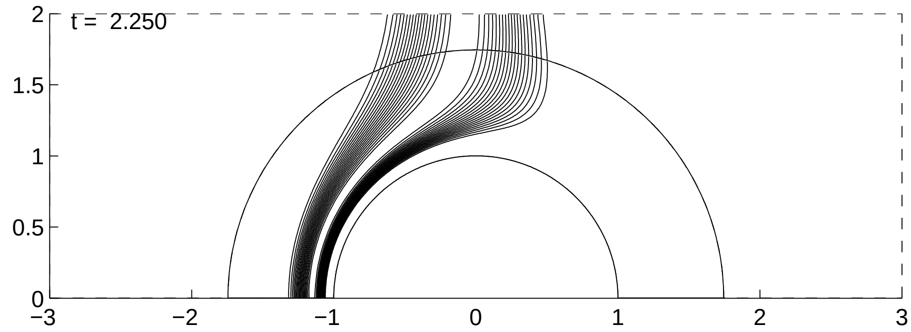


(b)

Figure 6.6: Initial condition for smooth solution test. (a) Solution on embedded Chimera. (b) Contours on embedded Chimera (20 levels).

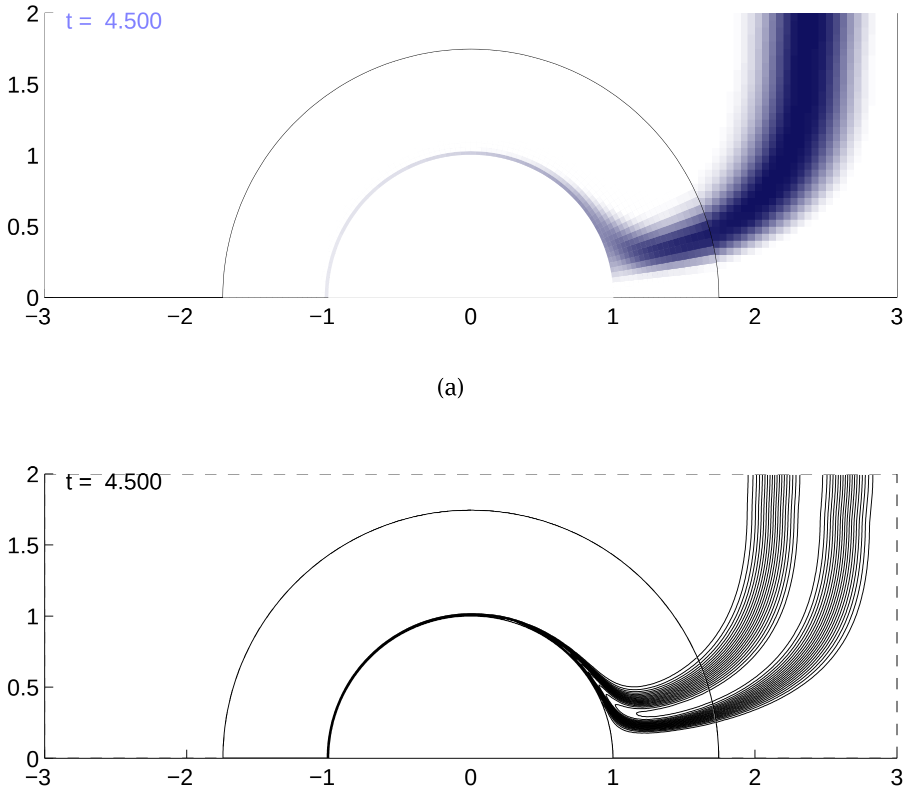


(a)



(b)

Figure 6.7: Smooth solution at middle time-step.



(a)

(b)

Figure 6.8: Smooth solution at final time-step.

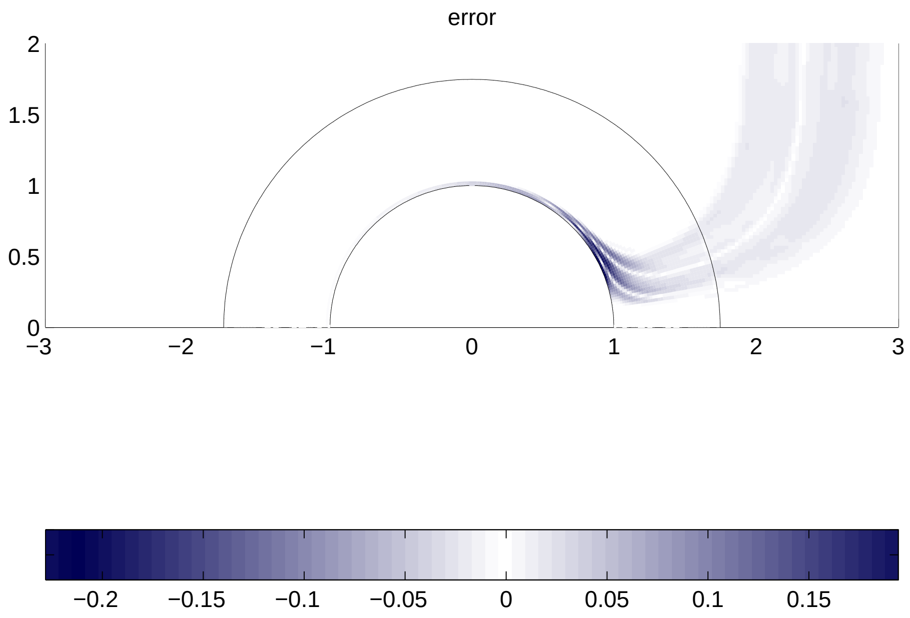


Figure 6.9: Errors of the numerical advection solution are concentrated near the cylinder.

6.2 Burgers' equation with smooth solutions

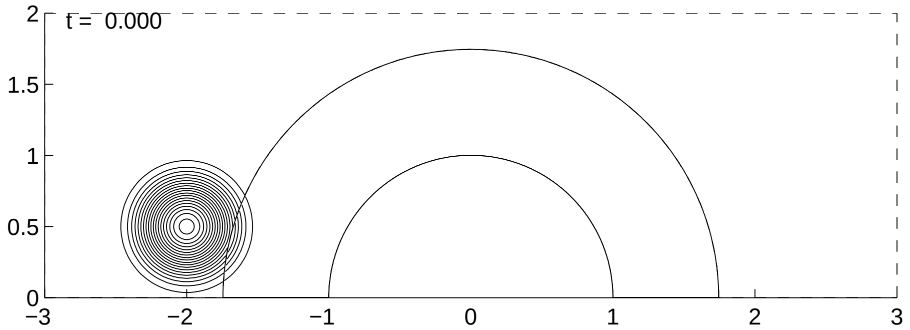
For error analysis, we examine the propagation of a cosine hump function The hump has radius 0.5, and starts at position $(-2, 0.5)$. The base value of q is 1.0, and the maximum value at the top of the hump is 2.0. To prevent the hump from steepening up into a shock, we examine the solution at a small time, but after the hump has begun crossing the grid interface. For this example, we use a modification of the standard Burgers' equation,

$$q_t + (u\ q^2/2)_x + (v\ q^2/2)_y = 0. \tag{6.2}$$

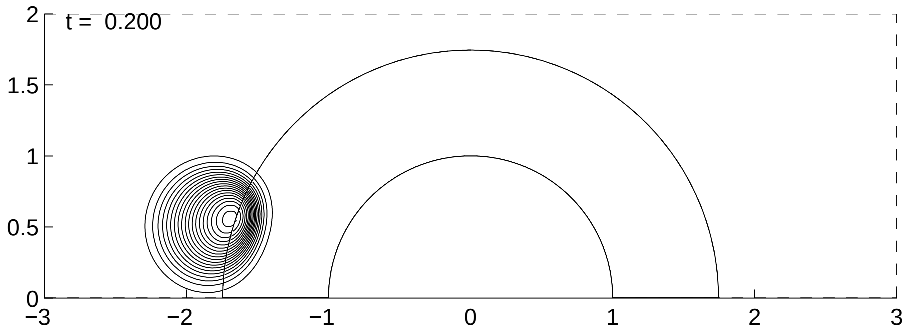
This is simply the one-dimensional form of Burgers' equation, but evaluated on streamlines. The advantage of this formulation is that boundary conditions are easier to apply near the cylinder, since streamlines will be tangent to the surface. We use the same cylindrical flow field (u, v) as in the advection example above. Contours of the initial condition and the solution at time 0.2 are shown in Figure 6.10.

Table 6.3: Errors for Burgers' in cylindrical flow.

h	L^1 error	L^1 ratio	L^∞ error	L^∞ ratio
0.100	1.321E-02	—	9.411E-02	—
0.050	3.321E-03	3.977	7.019E-02	1.34



(a)



(b)

Figure 6.10: (a) Initial condition. (b) Solution at time $t = 0.5$, before shock develops.

As in the case of advection we trace back along streamlines to get the true solution for error comparisons. We compare the errors at different resolutions in Table 6.3. In all solutions we use the monotonized-centered limiter [54, 57],

$$\phi = \max\left(0, \min\left(2 * r, (1 + r)/2, 2\right)\right).$$

6.3 Discontinous solutions

Since one of the purposes of to develop a conservative method for hyperbolic equations on Chimera grids, we should examine the behaviour of the method on discontinuous solutions. In addition, we would like to check the behavior of the method at various places around the grid intersection. To do so, we use two types of examples: In the first, we let a circular plug of concentration be carried through the interface in cylindrical flow. In the second, we look at a discontinuity that is wrapped around the cylinder by that flow, and in the process, the discontinuity is carried through the entire grid-interface region.

6.3.1 First discontinuous examples

This first set of examples shows that discontinuities are handled well by the method.

Advection of a cylindrical plug

We examine a cylindrical plug concentration or radius 0.5 centered at position $(x, y) = (-2, 1)$ that is inserted into the flow at time 0. Inside the cylinder $q = 1$, outside $q = 0$. The contour plots show 5 snapshots of the solution. Figure 6.11(a) shows the initial position of the cylindrical concentration. At a slightly later time in Figure 6.11(b) the profile of the concentration has distorted, and it is passing through the `CompGrid` interface into the body-fitted grid. Note that contour lines look virtually continuous across the grid interface. Other contour plots (Figures 6.11(c)–6.11(e)) confirm that the method does well as the concentration plug emerges from the other side. As before, there are 20 evenly-spaced contour lines. In this case, we use the superbee limiter to better resolve the discontinuity.

Burgers' along streamlines using a cylindrical plug

To examine the behaviour of a shock and rarefaction going through the interface, we can use the same cylindrical plug solution as in the advection case with the streamline form of Burgers' equations in (6.2). In order to move the entire plug through the interface, however, we set up a base state of 1 and make the plug concentration 2. The behavior is illustrated in the series of Figures 6.12(a)–6.12(e).

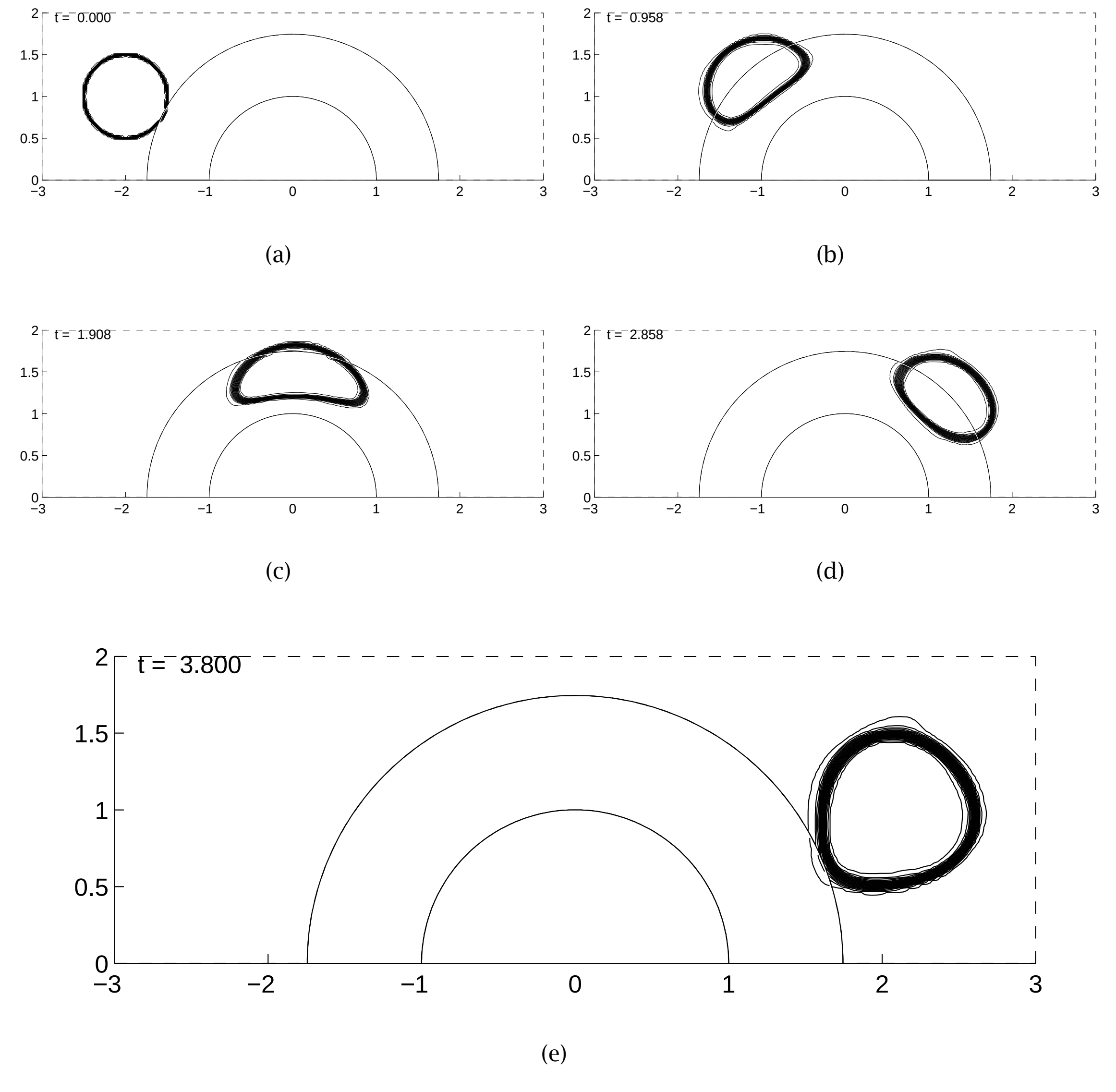


Figure 6.11: Contours of discontinuous advection concentration profile at (a) initial time, (b) entering radial grid, (c) inside radial grid, (d) exiting radial grid, (e) final time.

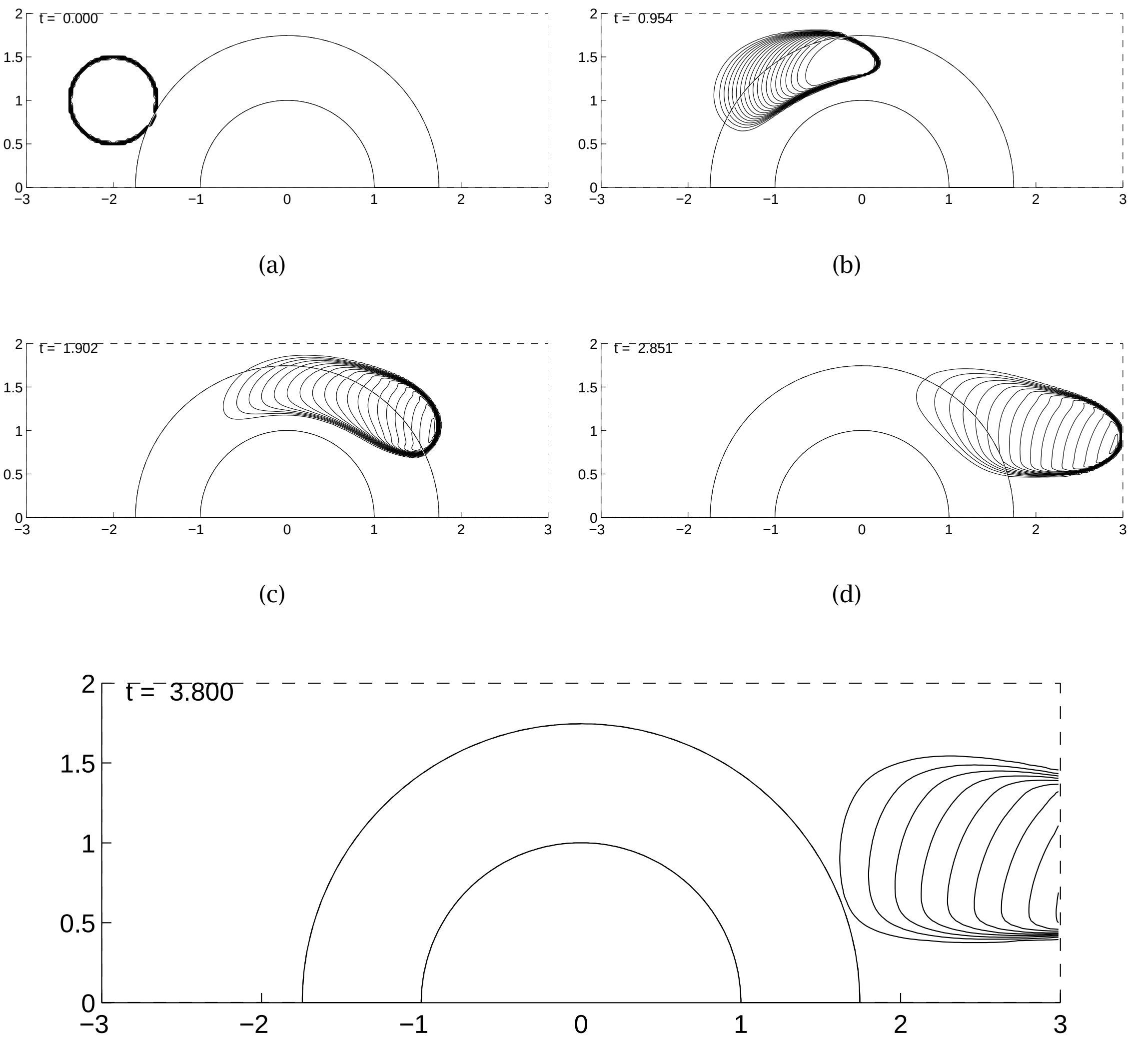


Figure 6.12: Contours of discontinuous Burgers' concentration profile at (a) initial time, (b) entering radial grid, (c) inside radial grid, (d) exiting radial grid, (e) final time.

6.3.2 Shock-front examples

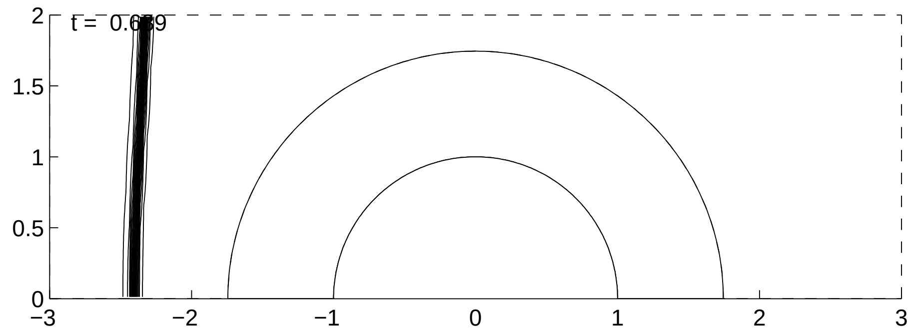
In this section we examine the effect of propagating a discontinuous front through the grid-interface. In this way we can see how discontinuities and shocks behave over a range of small cut-cells.

Advection of a discontinuous front

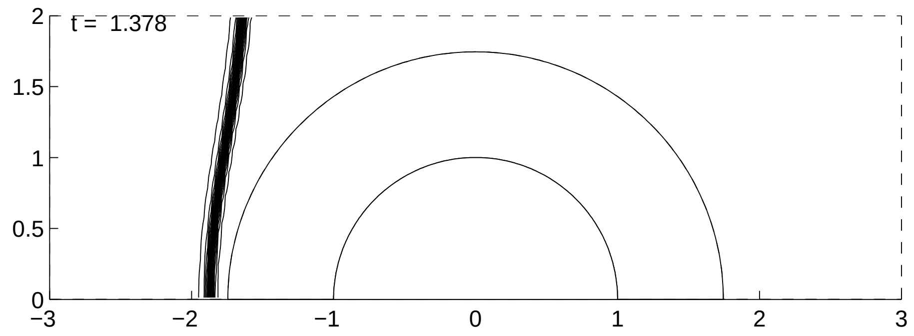
For this example, we impose a boundary condition of $q = 1$ at the left side of the Cartesian grid, with $q = 0$ everywhere else. As above, zero-extrapolated BC's are used on the upper and right boundaries, while reflecting boundary conditions are used on the x-axis. The contours at eight separate time intervals are shown in Figures 6.13, 6.14 and 6.15. From these figures we see that the discontinuity's contours match extremely well across the discontinous boundary, at all portions of the grid interface.

Burgers' equation on streamlines with shocks

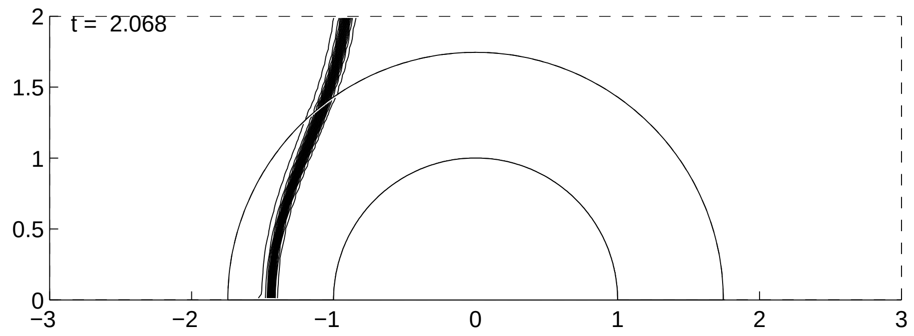
An example of Burgers' equation on streamlines with a single shock would look very much like the previous advection example. If anything, the discontinuity would be even more sharply resolved, as characteristics would be continuously entering the shock and steepening its profile. Therefore, to show a more interesting example, we examine the effect of (6.2) on a combination of a smooth hump steepening into a shock which is being trailed by a faster shock. The initial conditions are $q = 1$ as a base, with a cosine hump from 1 to 2 in a vertical band of width 1, centered around $x = -2$. At the left boundary a condition of $q = 2$ is imposed, and zero-extrapolation and reflecting boundary conditions are used as in the previous advection example. The solution contours at several different time steps are shown in Figures 6.16, 6.17 and 6.18. It is interesting to note that both the leading shock and trailing shocks are resolved well and the contours of the rarefaction are quite clear.



(a)

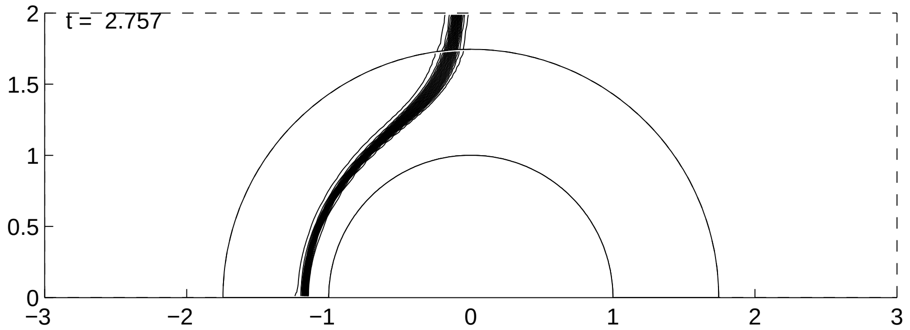


(b)

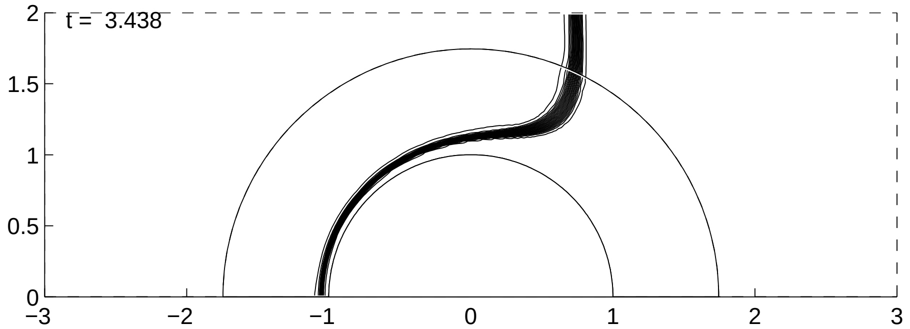


(c)

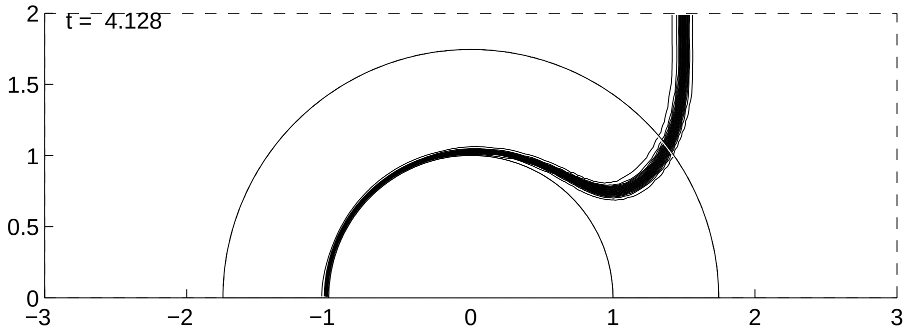
Figure 6.13: Contours of an advected discontinuous front at the first three time steps.



(a)

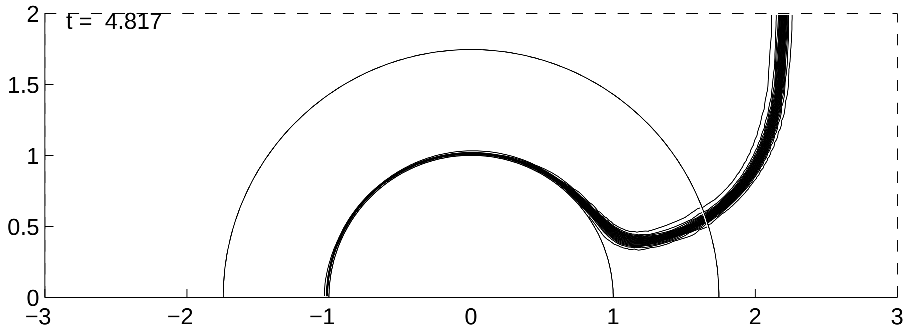


(b)

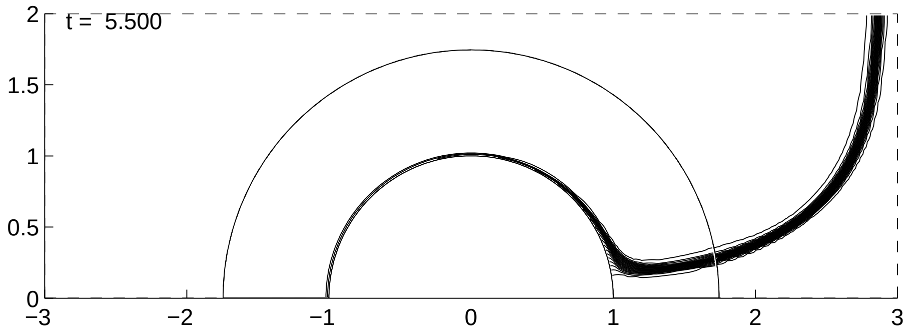


(c)

Figure 6.14: Contours of an advected discontinuous front at the next three time steps.

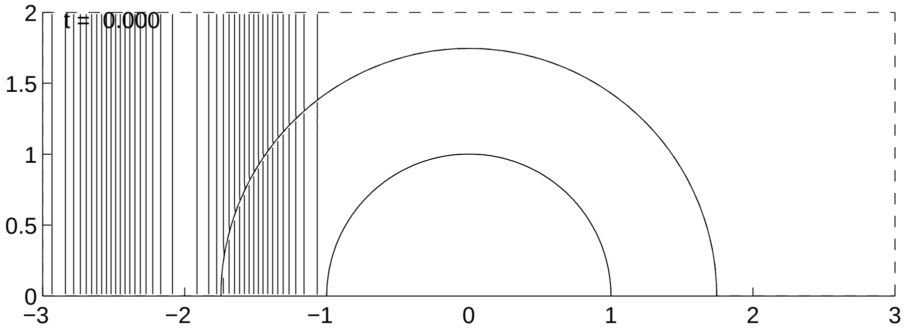


(a)

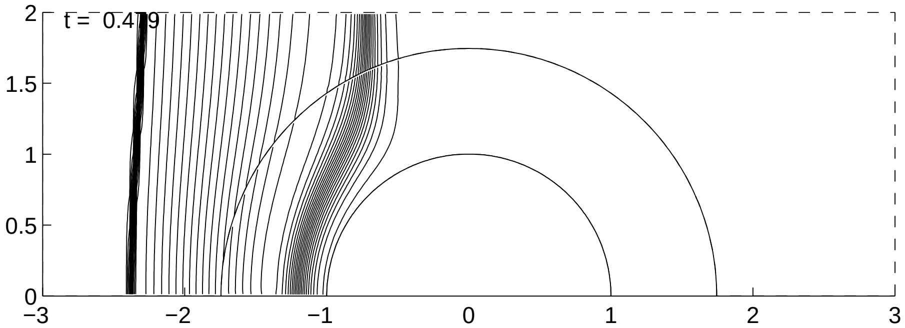


(b)

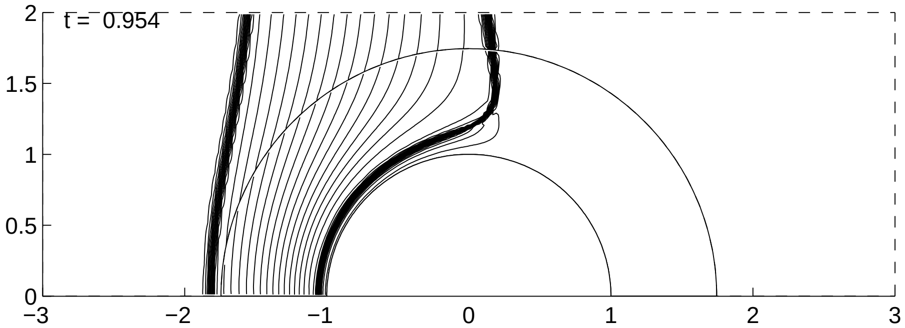
Figure 6.15: Contours of an advected discontinuous front at the next three time steps.



(a)

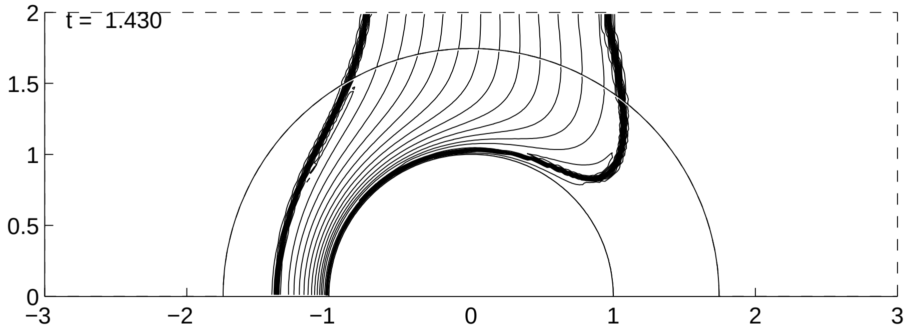


(b)

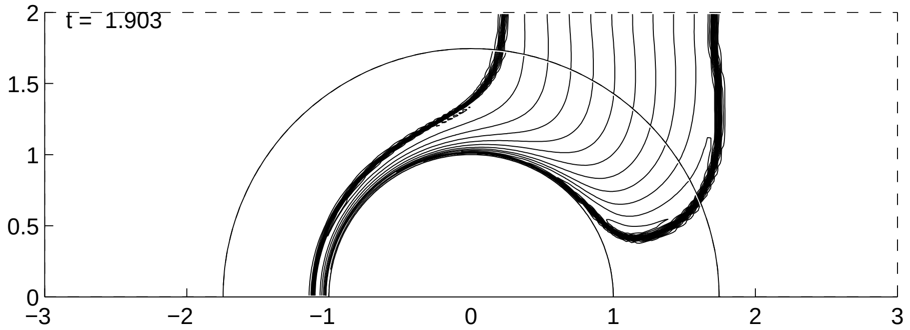


(c)

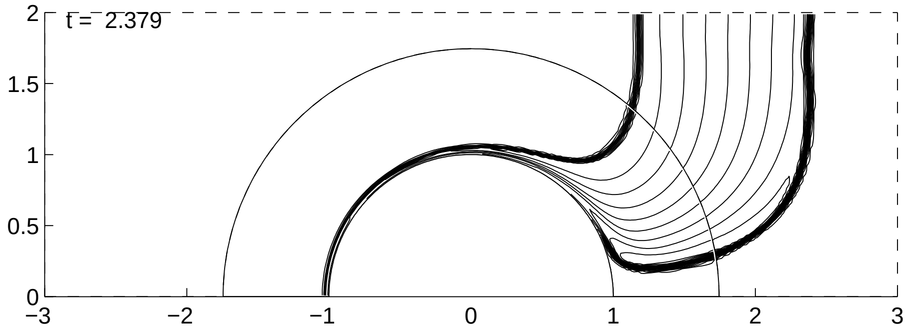
Figure 6.16: Contours of shocks and rarefactions in the streamline-Burgers' example at the first three time steps.



(a)

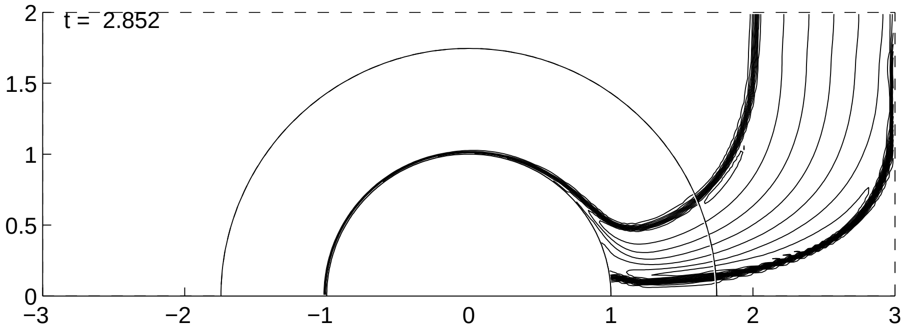


(b)

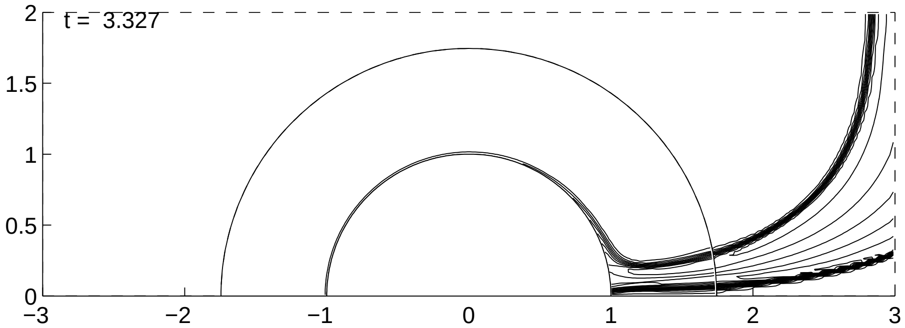


(c)

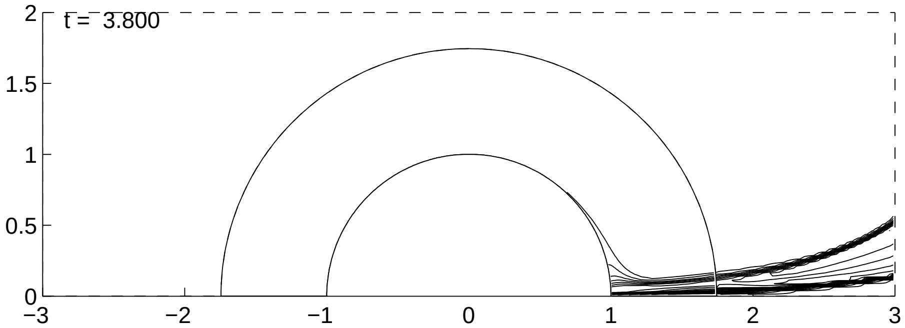
Figure 6.17: Contours of shocks and rarefactions in the streamline-Burgers' example at the next three time steps.



(a)



(b)



(c)

Figure 6.18: Contours of shocks and rarefactions in the streamline-Burgers' example at the last three time steps.

Chapter 7

CONCLUSIONS

7.1 Contributions of this work

We have developed a discretely conservative method, h-box-CLAW, for the solution of hyperbolic equations on Chimera grids in two dimensions. This method generalizes the CLAWPACK conservative flux-calculation algorithm to non-uniform cell-faces, and is consistent with CLAWPACK on Cartesian grids. While the method has been tested and analyzed primarily on the advection equation, it has been formulated in a way that can be extended naturally to non-linear hyperbolic equations and hyperbolic systems of equations.

We have developed necessary structures, FaceGrids and VertGrids, to store geometry, connectivity and data for the two-dimensional version of h-box-CLAW. For gradient recovery near grid interfaces (required for piece-wise linear reconstruction), we use a modified version of an algorithm from unstructured mesh techniques (Section A.2). A standard scalar multi-dimensional limiter is used to prevent the introduction of new maxima or minima.

We have proved that the h-box-CLAW method is TVD stable for uniform scalar advection in 1D, and that it is L^1 stable for uniform scalar advection in 2D. As noted in Section 5.3.1, our stability analysis of the h-box equivalent of the donor-cell upwind method implies a tighter time-step restriction for the earlier version of the h-box method as used by Berger and LeVeque. Because their version does not use transverse corrections, we show that it can be unstable in the case of transverse flows at solid boundaries when the CFL sum $|\tilde{\nu}_n| + |\tilde{\nu}_\tau| > 1$ (in coordinates with respect to the boundary).

In Chapter 2 we show for both advection and non-linear scalar equations in one dimension that accuracy of the h-box method is limited only by the accuracy of the reconstruction of data in h-boxes—piece-wise linear reconstruction is sufficient for second-order accuracy when a second-order algorithm is used. Our one-dimensional numerical results in Chapter 3 bear out our analytical result that piece-wise linear reconstruction is necessary for smooth accurate solutions of linear and non-linear scalar equations on general non-uniform grids. We show that the one-dimensional h-box method works well numerically for a non-linear system, the isothermal gas dynamics equations.

The CLAWPACK flux-calculation algorithm used for structured grid calculation in two dimensions has been extended to general quadrilateral grids. The solver can be modified for other equations through inclusion of Riemann Problem solver and flux initialization subroutines. The quadrilateral generalization makes it possible to use the same code for combinations of advection and non-linear systems; *i.e*, hyperbolic systems of equations of the form

$$\boldsymbol{q}_t + [u(x,y)\, \boldsymbol{f}\,(\boldsymbol{q})]_x + [v(x,y)\, \boldsymbol{g}\,(\boldsymbol{q})]_y = 0.$$

This formulation may be convenient for analysis of spatially anisotropic problems. The velocity field can be calculated separately, and may vary with time.

The h-box-CLAW method has been formulated so that, like CLAWPACK, it may be extended to non-linear hyperbolic systems using additional Riemann Problem solver and a flux initializer routines. The composite grid code is designed so that the same routines can be used on both structured grids faces and also h-box structures at grid interfaces. Note that even for a scalar equation on a quadrilateral grid, Riemann problem solutions will be dependent on face orientation.

In two dimensions, we verify numerically that the h-box-CLAW method is second-order accurate both for non-uniform divergence-free advection and for two-dimensional Burgers' equation, a non-linear equation.

7.2 Future Work

In this dissertation the h-box-CLAW method has been used to solve hyperbolic equations in two dimensions. There are many directions for future research.

7.2.1 Complete extension to systems

The h-box-CLAW method can be extended to linear and non-linear systems of equations. Acoustics, elasticity and the Euler equations are systems for which a composite grid formulation would provide immediate utility.

7.2.2 Hybrid solvers

Chimera grids can be used with hybrid solvers. For example, the full Navier-Stokes equations can be solved on a body-fitted grid in a concentrated grid-layer near the body, then that solution can be merged with solution of the Euler equations at some distance from the body. Finally, h-box-CLAW could be used as the interface between the Euler equation solution on the body-fitted grid and the far-field grid. This hybrid technique could be used to improve capturing of separation points or modeling of unusual or irregular surfaces in aerodynamic design.

7.2.3 Multiple-grid intersection

In general Chimera schemes, it is not always possible to completely separate body-fitted grids from each other. There will be some cases for which two body-fitted grids may intersect. In such cases, a far-field Cartesian grid cell may be cut twice. The h-box-CLAW method will generalize to this case without change, but more work is required to handle special cases such as these when a cutting grid's outer boundary intersects a previously generated cutting boundary.

7.2.4 Adaptive grid refinement

We would like to incorporate the h-box-CLAW method with adaptive grid refinement (AMR) methods. This would allow simpler generation of both body and far-field grids, and then those grids could be refined in regions of strong gradients, shocks, discontinuities or boundary layers. It would be necessary, however, to set up h-box structures for grid intersections at each level of refinement, and h-box size will also have to change at each level. Slope reconstruction techniques would have to be extended to handle possible intersections of a coarse grid, a fine grid and a cutting body-fitted grid with refinement.

7.2.5 Embedded Cartesian grids

The h-box-CLAW method on embedded Chimera grid has been presented as an alternative to embedded Cartesian grid methods, and the advantages of Chimera body-fitted grids have been discussed extensively. However, embedded Cartesian grids have a place in the calculation of "quick and dirty" approximate solutions. The properties of the h-box-CLAW method that make it useful for interfacing body-fitted with Cartesian grid solutions, *i.e.*, extension to second-order accuracy and consistency of the formulation with Cartesian flux calculation, also make our method an attractive alternative to Berger and LeVeque's body-aligned h-box method for embedded Cartesian grids. We also expect that the Cartesian-aligned h-box-CLAW method would stable for slightly larger time-steps than the body-aligned h-box method, since the latter does not incorporate transverse corrections.

7.2.6 Three dimensions

In two dimensions we have seen that stability requires the reconstruction of h-box information in `FaceGrids` and `VertGrids`. In three dimensions it will be necessary to reconstruct data in a third data structure, one constructed with respect to an edge on an irregular face; one might call the new set of h-boxes *`EdgeGrids`*. Furthermore, it will be necessary to incorporate more transverse-propagation terms to maintain stability. An additional complication in three dimensions is that all hexahedral and intersected cell faces must be triangulated to guarantee that cells are made up of a collection of flat faces, and a polyhedron intersection algorithm will be required to determine cell overlaps. However, the h-box method's basic concept of data overlap and uniform numerical domain of dependence extends naturally to three dimensions, and 3-D CLAWPACK [29,30] can be generalized for h-boxes in a similar manner as in two-dimensions.

BIBLIOGRAPHY

[1] M. J. Aftosmis, J. E. Melton, and M. J. Berger. Adaptation and surface modeling for Cartesian mesh methods. AIAA Paper 95-1725-CP, June 1995.

[2] T. Arbogast, M. F. Wheeler, and I. Yotov. Logically rectangular mixed methods for flow in irregular, heterogeneous domains. In A. A. Aldama et al., editors, *Computational Methods in Water Resources XI*, volume 1, pages 621–628. Computational Mechanics publications, Southampton, 1996.

[3] T. J. Barth. Aspects of unstructured grids and finite-volume solvers for the Euler and Navier-Stokes equations. In *VKI Lecture Series, 1994-05*, number 5 in Computational Fluid Dynamics. Von Karman Institute for Fluid Dynamics, Rhode Saint Genese, Belgium, 1994.

[4] Timothy J. Barth and Dennis C. Jesperson. The design and application of upwind schemes on unstructured meshes. AIAA Paper 89-0366, January 1989.

[5] M. J. Berger and R. J. LeVeque. An adaptive Cartesian mesh algorithm for the Euler equations in arbitrary geometries. AIAA Paper 89-1930, 1989.

[6] M. J. Berger and R. J. LeVeque. Cartesian meshes and adaptive mesh refinement for hyperbolic partial differential equations. In *Proc. Third Int'l Conf. Hyperbolic Problems, Uppsala*, 1990.

[7] M. J. Berger and R. J. LeVeque. Stable boundary conditions for Cartesian grid calculations. *Computing Systems in Engineering*, 1:305–311, 1990.

[8] M. J. Berger and R. J. LeVeque. A rotated difference scheme for Cartesian grids in complex geometries. AIAA Conference on Computational Fluid Dynamics, CP-91-1602, 1991.

[9] M. J. Berger, R. J. LeVeque, and L. G. Stern. Finite volume methods for irregular one-dimensional grids. *Proc. Symp. Appl. Math.*, 48:255–259, 1994. `ftp://amath.washington.edu/pub/leveque/papers/mjb:rjl:lgs:vancouver.ps.%Z` .

[10] K. Brislawn, D. Brown, G. Chesshire, W. Henshaw, K. Pao, D. Quinlan, E. Randall, and J. Saltzman. Overture: The grid classes. Los Alamos National Labs internal report, LA-UR-96-3708, 1996. `http://www.c3.lanl.gov/~henshaw/Overture/Overture.html`.

[11] D. L. Brown, G. Chesshire, and W. D. Henshaw. CMPGRD software. http://www.c3.lanl.gov/~geoff/docs/docs.html. Initial development of CMPGRD was supported at IBM's Thomas J. Watson Research Laboratory.

[12] S. R. Chakravarthy, A. Harten, and S. Osher. Essentially non-oscillatory shock-capturing schemes of arbitrarily-high accuracy. AIAA Paper 86-0339, January 1986.

[13] G. Chesshire and W. D. Henshaw. Composite overlapping meshes for the solution of partial differential equations. *J. Comput. Phys.*, 90(1):1–64, September 1990.

[14] G. Chesshire and W. D. Henshaw. A scheme for conservative interpolation on overlapping grids. *SIAM J. Sci. Comput.*, 15(4):819–45, July 1994.

[15] G. Chesshire and V. K. Naik. An environment for parallel and distributed computation with application to overlapping grids. *IBM J. Res. Develop.*, 38(3):285–300, May 1994.

[16] P. Colella. Multidimensional upwind methods for hyperbolic conservation laws. *J. Comput. Phys.*, 87:171–200, 1990.

[17] Darren De Zeeuw and Kenneth G. Powell. An adaptively refined Cartesian mesh solver for the Euler equations. *J. Comput. Phys.*, 104(1):56–68, January 1993.

[18] B. Engquist and A. J. Majda. Absorbing boundary conditions for the numerical solution of waves. *Math. Comp.*, 31:629–651, 1977.

[19] C. A. J. Fletcher. *Computational Techniques for Fluid Dynamics*, volume II, chapter 12, pages 46–51. Springer-Verlag, New York, 1988.

[20] M. Furukawa, M. Yamasaki, and M. Inoue. A zonal approach for solving the Navier-Stokes equations using a TVD finite volume method. In *Collection of Technical Papers ISCFD NAGOYA 1989*, pages 566–71, Nagoya, Japan, 28-31 August 1989. ISCFD NAGOYA, Nagoya Trade and Ind. Center.

[21] A. Harten, B. Engquist, S. Osher, and S. R. Chakravarthy. Uniformly high order accurate essentially non-oscillatory schemes III. *J. Comput. Phys.*, 71(2):231–303, August 1987.

[22] A. Harten and J. M. Hyman. Self-adjusting grid methods for one-dimensional hyperbolic conservation laws. *J. Comput. Phys.*, 50:235–269, 1983.

[23] A. Harten and S. Osher. Uniformly high-order accurate non-oscillatory schemes I. *SIAM J. Numer. Anal.*, 24(2):279–309, April 1987.

[24] A. Harten, S. Osher, B. Engquist, and S. R. Chakravarthy. Some results on uniformly high-order accurate essentially nonoscillatory schemes. *Appl. Numer. Math.*, 2(3-5):347–77, October 1986.

[25] W. D. Henshaw. A fourth-order accurate method for the incompressible Navier-Stokes equations on overlapping grids. *J. Comput. Phys.*, 113(1):13–25, July 1994.

[26] W. D. Henshaw, H.-O. Kreiss, and L. G. M. Reyna. A fourth-order-accurate difference approximation for the incompressible Navier-Stokes equations. *Comput. & Fluids*, 23(4):575–93, May 1994.

[27] Kai-Hsiung Kao, Meng-Sing Liou, and Chuen Yen Chow. Grid adaptation using chimera composite overlapping meshes. *AIAA J.*, 32(5):942–9, May 1994.

[28] H.-O. Kreiss, B. Swartz, B. Wendroff, and A. B. White, Jr. Supra-convergent schemes on irregular grids. *Math. Comp.*, 47(176):537–54, October 1986.

[29] J. O. Langseth and R. J. LeVeque. Three-dimensional Euler computations using CLAWPACK. In P. Arminjon, editor, *Conf. on Numer. Meth. for Euler and Navier-Stokes Eq., Montreal*, 1995. to appear.

[30] J. O. Langseth and R. J. LeVeque. A wave-propagation method for three-dimensional hyperbolic problems. to appear, 1996.

[31] P. D. Lax and B. Wendroff. Systems of conservation laws. *Comm. Pure Appl. Math.*, 13:217–237, 1960.

[32] R. J. LeVeque. CLAWPACK software. Available from `netlib.att.com` in `netlib/pdes/claw/doc`, or `http://www.amath.washington.edu/~rjl/clawpack.html`.

[33] R. J. LeVeque. CLAWPACK User Notes. Available from `netlib.att.com` in `netlib/pdes/claw/doc`, or `http://www.amath.washington.edu/~rjl/clawpack.html`.

[34] R. J. LeVeque. *Numerical Methods for Conservation Laws.* Birkhäuser-Verlag, 1990.

[35] R. J. LeVeque. CLAWPACK — a software package for solving multi-dimensional conservation laws. In *Proc. 5'th Int'l Conf. Hyperbolic Problems*, 1994. `ftp://amath.washington.edu/pub/leveque/papers/hyp94.ps.Z`.

[36] R. J. LeVeque. Wave propagation algorithms and the clawpack software for multi-dimensional hyperbolic systems. submitted, 1995. `ftp://amath.washington.edu/pub/leveque/papers/rjl:clawpack`.

[37] T. A. Manteuffel and A. B. White, Jr. The numerical solution of second-order boundary value problems on nonuniform meshes. *Math. Comp.*, 47:511–535, 1986.

[38] Richard B. Pember, Ann S. Almgren, William Y. Crutchfield, Louis H. Howell, John B. Bell, Phillip Colella, and Vincent E. Beckner. An embedded boundary method for the modeling of unsteady combustion in an industrial gas-fired furnace. LLNL preprint, October 18 1995.

[39] N. A. Petersson. Computing periodic gravity waves on water by using moving composite overlapping grids. *SIAM J. Sci. Comput.*, 14(6):1339–58, Nov. 1993.

[40] N. A. Petersson and J. F. Malmliden. Computing the flow around a submerged body using composite grids. *J. Comput. Phys.*, 105(1):47–57, March 1993.

[41] James J. Quirk. An alternative to unstructured grids for computing gas dynamic flows around arbitrarily complex two-dimensional bodies. *Comput. & Fluids*, 23(1):125–142, January 1992.

[42] James J. Quirk. An alternative to unstructured grids for computing gas dynamic flows around arbitrarily complex two-dimensional bodies. *Comput. & Fluids*, 23(1):125–42, January 1994.

[43] James J. Quirk. A contribution to the great Riemann solver debate. *Internat. J. Numer. Methods Fluids*, 18(6):555–74, March 1994.

[44] P. L. Roe. Approximate Riemann solvers, parameter vectors, and difference schemes. *J. Comput. Phys.*, 43:357–372, 1981.

[45] P. L. Roe. Fluctuations and signals – a framework for numerical evolution problems. In K. W. Morton and M. J. Baines, editors, *Numerical Methods for Fluid Dynamics*, pages 219–257. Academic Press, 1982.

[46] P. L. Roe. Upwind scemes using various formulations of the Euler equations. In F. Angrand et al., editors, *Numerical Methods for the Euler Equations of Fluid Dynamics*, pages 14–31. INRIA Workshop, SIAM, 1985.

[47] Patrick Saldou. `http://barkley.me.berkeley.edu/~saldou/`.

[48] T. I. P. Shih, M. J. Rimlinger, and W. J. Chyu. Three-dimensional shock-wave/boundary-layer interactions with bleed. *AIAA J.*, 31(10):1819–26, October 1993.

[49] C.-W. Shu and S. Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes. *J. Comput. Phys.*, 77(2):439–71, August 1988.

[50] C.-W. Shu and S. Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes II. *J. Comput. Phys.*, 83(1):32–78, July 1989.

[51] J. L. Steger, F. C. Dougherty, and J. A. Benek. A chimera grid scheme. In K. N. Ghia and U. Ghia, editors, *Advances in Grid Generation*, pages 59–69. ASME, New York, asme fed-5 edition, 1983.

[52] G. Strang. On the construction and comparison of difference schemes. *SIAM J. Numer. Anal.*, 5:506–517, 1968.

[53] John C. Strikwerda. *Finite Difference Schemes and Partial Differential Equations*. Wadsworth & Brooks/Cole Advanced Books & Software, Pacific Grove, California, 1989.

[54] P. K. Sweby. High resolution schemes using flux limiters for hyperbolic conservation laws. *SIAM J. Numer. Anal.*, 21:995–1011, 1984.

[55] Joe K. Thompson. A reflection on grid generation in the 90s: trends, needs, and influences. In B. K. Soni, J. F. Thompson, J. Häuser, and P. R. Eiseman, editors, *Numerical Grid Generation in Computational Field Simulations*, volume I, pages 1029–1110. NSF Engineering Research Center for Computational Field Simulations, 1996. Proceedings of the 5th International Conference, held at Mississippi State University, April 1 – Apr 5, 1996.

[56] B. van Leer. Towards the ultimate conservative difference scheme II. Monotonicity and conservation combined in a second order scheme. *J. Comput. Phys.*, 14:361–370, 1974.

[57] B. van Leer. Towards the ultimate conservative difference scheme V. A second order sequel to Godunov's method. *J. Comput. Phys.*, 32:101–136, 1979.

[58] Zhi-Jian Wang. A fully conservative structured/unstructured Chimera grid scheme. AIAA Paper 95-0671, January 1995.

[59] Zhi-Jian Wang and H. Q. Yang. A unified conservative zonal interface treatment for arbitrarily patched and overlapped grids. AIAA Paper 94-0320, January 1994.

Appendix A

RECONSTRUCTION METHODS IN ONE AND TWO DIMENSIONS

A.1 One-dimensional reconstruction

In one dimension, we use a reconstruction method inspired by the primitive function formulation of the ENO techniques of Harten, Osher, *et al.* [12, 21, 23, 24, 49, 50].

We will denote the index of the cell containing $x_{i-1/2,L}$ by $p_{i-1/2} \equiv p(x_{i-1/2,L})$, and that of the cell containing $x_{i-1/2,R}$ by $q_{i-1/2} \equiv q(x_{i-1/2,R})$. This is a somewhat cumbersome notation forced on us by the possibility that the outer boundaries of the h-boxes may be several cells away from $x_{i-1/2}$ if the grid is highly irregular.

The primitive function w is defined as

$$w(x, t_n) \equiv \int_{x_{-1/2}}^{x} q(x, t_n)\, dx. \tag{A.1}$$

We also define the following approximation to w

$$\begin{aligned} W_i^n &\equiv \sum_{i=1}^{j} Q_i^n h_i && \text{(A.2)} \\ &\approx \int_{x_{-1/2}}^{x_{i-1/2}} q(x, t_n)\, dx = w(x_{i-1/2}, t_n) && \text{(A.3)} \end{aligned}$$

which will be used in determining reconstructions to q. For function reconstruction, we make use of Newton's divided difference approximations, which use the following notation:

$$\begin{aligned} y[x] &= y(x) \\ y_i \equiv y[x_i] &= y(x_i) \\ y[x_0, x_1] &= (y_1 - y_0)/(x_1 - x_0) \\ &= y[x_1, x_0] \\ y[x_0, x_1, x_2] &= (y[x_1, x_2] - y[x_0, x_1])/(x_2 - x_0) \\ &\cdots \\ y[x_0, \ldots, x_m] &= (y[x_1, x_2, \ldots, x_m] - y[x_0, \ldots, x_{m-1}])/(x_m - x_0) \end{aligned}$$

Using this notation, we can interpolate $y(x)$ in the interval $x_{-1/2} \le x \le x_{N+1/2}$ by

$$y(x) = P_m(x) + R_m(x) \tag{A.4}$$

where $P_m(x)$ is the polynomial

$$\begin{aligned} P_m(x) &= P_m(x_0, x_1, \ldots, x_m, x) \\ &= y_0 \\ &\quad + (x - x_0)\ y[x_0, x_1] \\ &\quad + (x - x_0)(x - x_1)\ y[x_0, x_1, x_2] \\ &\quad + \quad \ldots \\ &\quad + (x - x_0)(x - x_1)\ldots(x - x_{m-1})\ y[x_0, \ldots, x_m], \end{aligned}$$

and $R_m(x)$, the error in approximating $y(x)$ by the order m polynomial $P_m(x)$, is defined as

$$\begin{aligned} R_m(x) &= R_m(x_0, x_1, \ldots, x_m, x) \\ &= (x - x_0)(x - x_1)\ldots(x - x_m)\ y[x_0, \ldots, x_m, x] \\ &= \frac{(x - x_0)(x - x_1)\ldots(x - x_m)\ y^{(m+1)}(\xi)}{(m+1)!} \end{aligned}$$

where $x_0 < \xi < x_m$ (from the Mean Value Theorem). We will call the truncated approximation $y(x) = P_m(x)$ the **order-m DDA** (Divided Difference Approximation) to $y(x)$.

It is possible to be more precise about the location of ξ in the interval $[x_0, x_m]$ by examining the term $y[x_0, \ldots, x_m, x]$ in more detail. Let us look, for instance at error in the order-1 DDA for $x \in [x_0, x_1]$, $R_1(x)$, with $x_1 - x_0 = O(h)$:

$$\begin{aligned} R_1(x) &= R_1(x_0, x_1, x) \\ &= (x - x_0)(x - x_1)\ y[x_0, x_1, x] \\ &= (x - x_0)(x - x_1)\ \frac{\frac{(y(x_1) - y(x))}{(x_1 - x)} - \frac{(y(x_1) - y(x_0))}{(x_1 - x_0)}}{x - x_0} \\ &= (x - x_0)\ (x - x_1)\ \frac{y'\left(\frac{x_1 + x}{2}\right) - y'\left(\frac{x_1 + x_0}{2}\right)}{2\left(\frac{x_1 + x}{2} - \frac{x_1 + x_0}{2}\right)} + O(h^3) \\ &= (x - x_0)\ (x - x_1)\ \frac{y''\left(\frac{x_0 + 2x_1 + x}{2}\right)}{2} + O(h^3) \end{aligned}$$

An inductive process serves to show that in the general case,

$$R_m(x) = (x - x_0)\ (x - x_1)\ldots(x - x_m)\ \frac{y^{(m+1)}(\xi)}{(m+1)!} + O(h^{m+2}), \tag{A.5}$$

where

$$\xi = \frac{x_0 + \binom{m+1}{1} x_1 + \cdots + \binom{m+1}{m} x_m + x}{2^{m+1}}. \tag{A.6}$$

with $h \geq \max_{i=1,\ldots,m} h_i$.

The one-dimensional reconstruction techniques explored here use an **ENO**-style reconstruction: first, the values of the primitive function $W_{i-1/2}^n$ are determined at $x_{i-1/2}$ as in (A.3). Next, if we assume that at a particular time-step t_n our Q_i^n's are exact cell-averages, then it follows that the $W_{i-1/2}^n$'s are exact point values of $w(x, t_n)$ as well. In addition, $w(x, t_n)$ has one more degree of continuity in x than $q(x, t_n)$, so it should be possible to interpolate w with some continuous function. In this research we use piece-wise polynomial reconstruction of degree m, using the order m divided-difference approximation. For this technique it is convenient to define the reconstruction of $w(x, t_n)$ to be $W_{i-1/2,L}^n$ at $x_{i-1/2,L}$, and $W_{i-1/2,R}^n$ at $x_{i-1/2,R}$. Once these reconstructions of $w(x, t_n)$ are computed, we can find the neighboring interpolated cell-averages from

$$Q_{i-1/2,L}^n = (W_{i-1/2}^n - W_{i-1/2,L}^n)/h \tag{A.7}$$
$$Q_{i-1/2,R}^n = (W_{i-1/2,R}^n - W_{i-1/2}^n)/h. \tag{A.8}$$

In this section we will show how $Q_{i-1/2,L}^n$ is determined using PWP reconstruction. A similar procedure could be followed to find $Q_{i-1/2,R}^n$. We can find $W_{i-1/2,L}^n$ with a truncation error of $O(h^{m+1})$ with the order-m DDA, using the values of W_i^n at the $m+1$ points $\{x_{p_{i-1/2}-k}, \ldots, x_{p_{i-1/2}+m-k}\}$. The index k refers to the choice of stencil for the reconstruction, and can range from 1 to m. In practice, it will depend on the cell in which $x_{i-1/2,L}$ sits; i.e., $k = k(p_{i-1/2})$. It should be further noted that the stencil index k chosen for finding $W_{i-1/2,L}^n$ will not necessarily be the same stencil index k chosen for $W_{i-1/2,R}^n$. Continuing with our example, a piece-wise linear polynomial (PWLP) reconstruction of $w(x_{i-1/2,L}, t_n)$ would use $m = 1$, $k = 1$, and we would find $W_{i-1/2,L}^n$ using

$$W_{i-1/2,L}^n = W_{p_{i-1/2}-1}^n + (x_{i-1/2,L} - x_{p_{i-1/2}-1})\, W\left[x_{p_{i-1/2}-1}, x_{p_{i-1/2}}\right], \tag{A.9}$$

with a truncation error of $O(h^2)$. Similarly the piece-wise quadratic (PWQP) and cubic (PWCP) polynomial reconstruction forms for $W_{i-1/2,L}^n$ are

- Piece-wise quadratic reconstruction form:

$$\begin{aligned} W^n_{i-1/2,L} = W^n_{p_{i-1/2}-k} & \\ &+ (x_{i-1/2,L} - x_{p_{i-1/2}-k}) \\ &\quad W\left[x_{p_{i-1/2}-k}, x_{p_{i-1/2}-k+1}\right] \\ &+ (x_{i-1/2,L} - x_{p_{i-1/2}-k})\ (x_{i-1/2,L} - x_{p_{i-1/2}-k+1}) \\ &\quad W\left[x_{p_{i-1/2}-k}, x_{p_{i-1/2}-k+1}, x_{p_{i-1/2}-k+2}\right] \end{aligned} \tag{A.10}$$

  having a truncation error of $O(h^3)$, with $k = 1$ or 2.

- Piece-wise cubic reconstruction form:

$$\begin{aligned} W^n_{i-1/2,L} = W^n_{p_{i-1/2}-k} & \\ &+ (x_{i-1/2,L} - x_{p_{i-1/2}-k})\ W\left[x_{p_{i-1/2}-k}, x_{p_{i-1/2}}\right] \\ &+ (x_{i-1/2,L} - x_{p_{i-1/2}-k})\ (x_{i-1/2,L} - x_{p_{i-1/2}-k+1}) \\ &\quad W\left[x_{p_{i-1/2}-k}, x_{p_{i-1/2}-k+1}, x_{p_{i-1/2}-k+2}\right] \\ &+ (x_{i-1/2,L} - x_{p_{i-1/2}-k}) \\ &\quad (x_{i-1/2,L} - x_{p_{i-1/2}-k+1}) \\ &\quad (x_{i-1/2,L} - x_{p_{i-1/2}-k+2}) \\ &\quad W\left[x_{p_{i-1/2}-k}, \ldots, x_{p_{i-1/2}-k+3}\right] \end{aligned} \tag{A.11}$$

  having a truncation error of $O(h^4)$, with $k = 1, 2$ or 3.

Consequently, when we use these reconstruction formulas to determine the cell-averages, in the h boxes on either side of x_i, we get truncation errors for $Q^n_{i-1/2,L}$ of $O(h^1)$, $O(h^2)$, and $O(h^3)$, respectively.

A factor to consider when using this type of reconstruction formula is how we choose k. In Harten, *et al.*, the stencil is chosen to be essentially non-oscillatory — that is, k is chosen so that the numerical approximations to the derivatives of $w(x)$ are minimized for the reconstructions in cell $p_{i-1/2}$. This is the implementation we use in our method.

For comparison with standard techniques we note the following correspondences: piece-wise linear primitive function reconstruction (PWLP) is equivalent to piece-wise constant reconstruction (PWC) using Q; piece-wise quadratic primitive function reconstruction (PWQP) is equivalent to piece-wise linear reconstruction (PWL) using Q; and piece-wise cubic primitive function reconstruction (PWCP) is equivalent to piece-wise quadratic reconstruction (PWQ) using Q.

A.2 Two-dimensional reconstruction

In this section we describe the algorithm used for piece-wise linear reconstruction of $\overline{\nabla q}$ on CompGrid and MapGrid cells using physical connectivity information.

A.2.1 Problem

To reconstruct gradient information, we use the Green-Gauss theorem:

$$\begin{aligned} \overline{\nabla q} &= \frac{1}{A_{\Omega_c}} \int_{\Omega_c} \nabla q \; dA \\ &= \frac{1}{A_{\Omega_c}} \int_{\partial\Omega_c} q \, \boldsymbol{n} \; dl \end{aligned} \tag{A.12}$$

Exampes of typical Ω_c's over which we construct this average gradient are seen in Figure A.1.

An efficient way to calculate $\overline{\nabla q}$ in each type of Ω_c is to first calculate $A_{\Omega_v} \; \overline{\nabla q}$, where Ω_v is the region about a vertex determined by connecting the centroids of cells neighboring the vertex. This region is sometimes called the *cell-midpoint* dual of the space partition. $\overline{\nabla q}$ on Ω_c is found by summing up values of $A_{\Omega_v} \; \overline{\nabla q}$ from each vertex on the cell and dividing the result by A_{Ω_c}.

We can precalculate certain geometrical constants, *viz.*,

- `FaceStat` on `MapGrid` faces,
- $\int_{C^-}^{C^+} \mathbf{n} \; dl$ on each `MapGrid` and `CompGrid` face,
- Ω_c on `MapGrid` and `CompGrid` cells and Ω_v on their vertices,

and store them in the overall `CompGrid` structure because they will not change during time-evolution of the PDE.

We note that some `CompGrid` vertices are incomplete; *i.e.*, they are not completely surrounded by `CompGrid` faces. In the same sense, some `MapGrid` vertices are incomplete; they are not completely surrounded by physical `MapGrid` faces. We therefore complete `CompGrid` vertex calculations by adding in information from the underlying `MapGrid` vertices and then using the resulting `CompGrid` information to overwrite that on the `MapGrid` vertex. In this way the connectivity information is stored in the `CompGrid` structure, not in the `MapGrid` structure.

We further note that the iterations used to implement this strategy can be done almost entirely with iterations over faces, using connectivity structures on faces. The only exception to this is that `CompGrid` vertices also have to know what `MapGrid` vertex they sit upon, which introduces an iteration over `CompGrid` vertices.

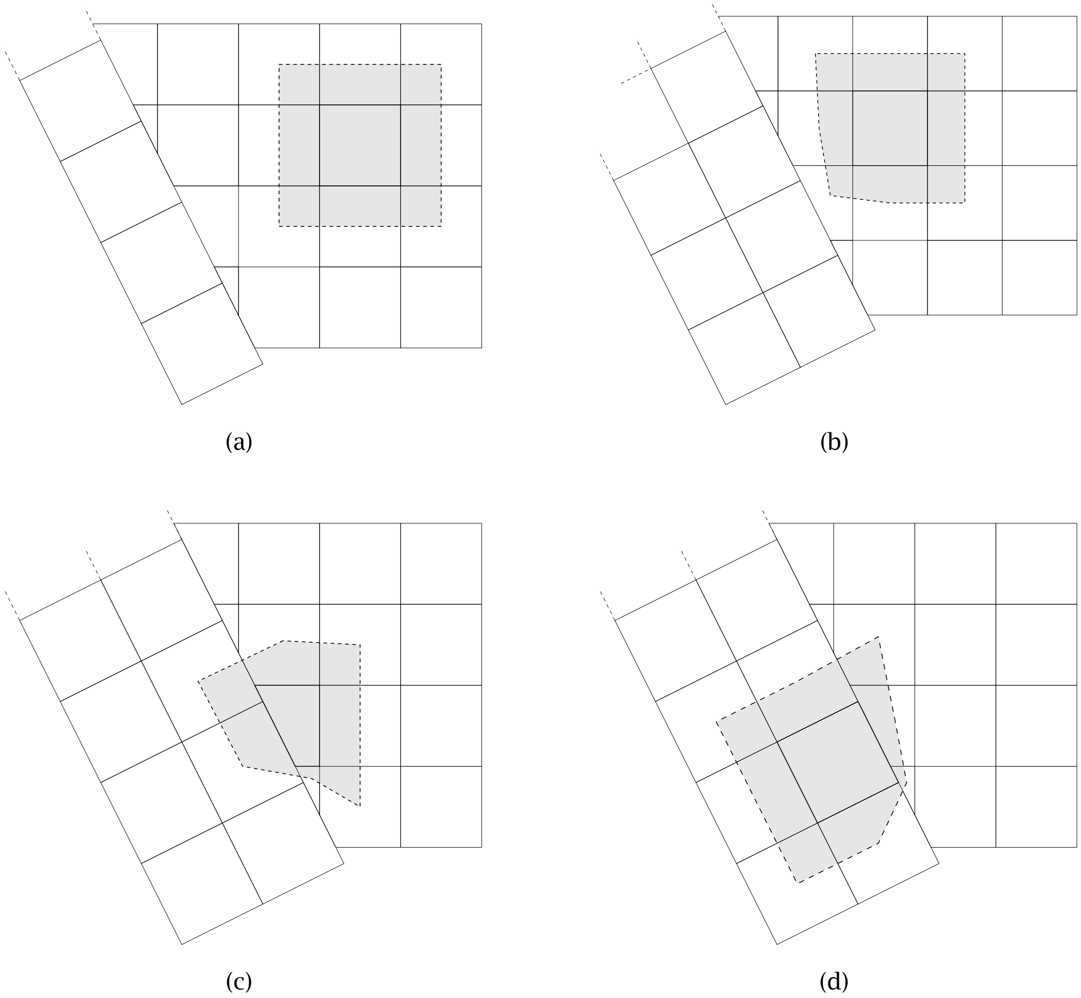


Figure A.1: Ω_c examples:
(a) Ω_c around an ordinary `MapGrid` cell.
(b) Ω_c around an `MapGrid` cell from cut-into grid next to `CompGrid` Cells.
(c) Ω_c around a `CompGrid` cell.
(d) Ω_c around an `MapGrid` cell from cutting grid next to `CompGrid` cells.

Algorithm A.1 (piece-wise linear reconstruction for `CompGrid`).

for all physical `MapGrid` faces **do**
  Scatter $\mp q\,\mathbf{n}\,ds$ to `MapGrid` vertices 1 and 2 on face, respectively.
end for
for all `CompGrid` faces **do**
  Scatter $\mp q\,\mathbf{n}\,ds$ to `CompGrid` vertices 1 and 2 on face, respectively.
end for
for all `CompGrid` vertices **do**
  if `CompGrid` vertex has a `MapGrid` vertex underneath it **then**

$$
\begin{aligned}
A_v\ \overline{\nabla q}_{\texttt{CompGrid}\,\mathrm{vert}} &\mathrel{+}= A_v\ \overline{\nabla q}_{\texttt{MapGrid}\,\mathrm{vert}}\\
A_v\ \overline{\nabla q}_{\texttt{MapGrid}\,\mathrm{vert}} &:= A_v\ \overline{\nabla q}_{\texttt{CompGrid}\,\mathrm{vert}}
\end{aligned}
$$

  end if
end for
for all `CompGrid` faces **do**
  Gather $A\ \overline{\nabla q}$ from `CompGrid` vertices on face.
  Scatter $A\ \overline{\nabla q}$ to `CompGrid` cells on face.
end for
for all `CompGrid` cells **do**
  Divide $A\ \overline{\nabla q}$ by Ω_c
end for
for all physical `MapGrid` faces **do**
  Gather $A\ \overline{\nabla q}$ from `MapGrid` vertices on face.
  Scatter $A\ \overline{\nabla q}$ to `MapGrid` cells on face.
end for
for all physical `MapGrid` cells **do**
  Divide $A\ \overline{\nabla q}$ by Ω_c
end for

Algorithm A.1

A.2.2 Gradient limiting

The problem of limiting reconstructed gradients in order to prevent non-physical oscillations and instabilities in the numerical solution is still a topic of active research. However, min-mod-type scalar limiters of diffusive type [54] which give reasonable results can be found in either Barth and Jesperson [4] or De Zeeuw and Powell [17]. These limiters are very similar to each other (though De Zeeuw and Powell do not mention Barth and Jesperson's work)—the cell-centered values $W_{i,j}$ (where cell index $\{i, j\}$ is used in the general sense of a two-dimensional reference) are used to find average gradients

$\nabla W_{i,j}$. Then piece-wise reconstruction takes the form

$$W(\boldsymbol{x}, \boldsymbol{x}_{i,j}) = W_{i,j} + \phi \nabla W_{i,j} \cdot (\boldsymbol{x} - \boldsymbol{x}_{i,j}),$$

where ϕ is a scalar between 0 and 1. ϕ is constructed so that the maximum value of $W(\boldsymbol{x}, \boldsymbol{x}_{i,j})$, attained at cell vertices, does not exceed the bounds of the data used to evaluate the gradient.

The Barth-Jesperson (BJ89) limiter is evaluated by taking the minimum value of all vertex values of ϕ in a cell , where each vertex limiter ϕ^v is determined using

$$\phi^v = \begin{cases} 1 & W^v_{i,j} - W_{i,j} = 0, \\ \min\left(1, \frac{W^v_{\max} - W_{i,j}}{W^v_{i,j} - W_{i,j}}\right) & W^v_{i,j} - W_{i,j} > 0, \\ \min\left(1, \frac{W^v_{\min} - W_{i,j}}{W^v_{i,j} - W_{i,j}}\right) & W^v_{i,j} - W_{i,j} < 0. \end{cases} \tag{A.13}$$

Here $W^v_{i,j}$ is the reconstruction at vertex v using the unlimited gradient, and $W^v_{\max}$ and $W^v_{\min}$ are the maximum and minimum, respectively, of W on cells neighboring the vertex. As formulated, this limiting procedure is applied to each component of a vector of conserved (or primitive variable) quantities, and a different minimum-to-maximum range is used at each vertex.

In the De Zeeuw-Powell limiter (DZP93), the limiting procedure is modified:

1. Primitive variables are used; *e.g.*, $\{\rho, u, v, p\}$ in the case of the Euler equations.

2. A larger minimum-to-maximum range is used for comparison—instead of using the max and min at each vertex separately, the limiter uses the maximum over all vertices. This ensures that the maximum and minimum reconstructed values in the cell will lie between the bounds of values used in the path integral used to reconstruct the gradient. However, variation of the reconstructed function could increase, because the limiting does not check variation at cell faces.

3. For a vector of primitive variables, the minimum of *all* component-wise limiters is used to limit the total vector reconstruction. This more restrictive requirement balances the greater range in vertex values allowed by the component-wise limiter.

These modifications may lead to gradient values which increase the total variation of the reconstruction. Therefore, in this research we have chosen to use the BJ89 limiter, which is somewhat simpler to implement and is closer to being TVD; however, the DZP93 limiter may be used in future work.

VITA

Louis G. Stern

Master of Science in Mathematics awarded October 1990 from New York University.

Bachelor of Arts in Physics awarded May 1984 from Reed College, Portland, Oregon.

Other Publications

"Fractional Step Methods Applied to a Chemotaxis Model," R. Tyson, L. G. Stern, R. J. LeVeque. *Submitted to Journal of Mathematical Biology*.

"Finite Volume Methods for Irregular One-Dimensional Grids," M. J. Berger, R. J. LeVeque, L. G. Stern. Proceedings of Symposia in Applied Mathematics, volume 48, 1994: *Mathematics of Computation, 1943–1993*.

"A nonlinear dosimetric model for hemoglobin adduct formation by the neurotoxic agent acrylamide and its genotoxic metabolite glycidamide." C. J. Calleman, L. G. Stern, E. Bergmark, L. G. Costa. *Environmental Health Perspectives*, March 1993.

"Linear versus nonlinear models for hemoglobin adduct formation by acrylamide and its metabolite glycidamide: implications for risk estimation." C. J. Calleman, L. G. Stern, E. Bergmark, L. G. Costa. *Cancer Epidemiology, Biomarkers and Prevention*, July-Aug. 1992.