ON SUPPRESSING SPURIOUS OSCILLATIONS IN THE NUMERICAL SOLUTION OF HYPERBOLIC CONSERVATION LAWS

by

Joel Patterson

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

AT

DALHOUSIE UNIVERSITY
HALIFAX, NOVA SCOTIA
AUGUST 3, 2004

DALHOUSIE UNIVERSITY

DEPARTMENT OF MATHEMATICS AND STATISTICS

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled "**On Suppressing Spurious Oscillations in the Numerical Solution of Hyperbolic Conservation Laws**" by **Joel Patterson** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: August 3, 2004

Supervisor: ______________________________
Raymond Spiteri

Reader: ______________________________
Patrick Keast

DALHOUSIE UNIVERSITY

Date: **August 3, 2004**

Author: **Joel Patterson**

Title: **On Suppressing Spurious Oscillations in the Numerical Solution of Hyperbolic Conservation Laws**

Department: **Mathematics and Statistics**

Degree: **M.Sc.** Convocation: **October** Year: **2004**

Table of Contents

List of Tables

List of Figures

Abstract

During the numerical solution of a hyperbolic conservation law, signs of nonlinear instability are most often indicated by the presence of numerical artifacts known as spurious oscillations.This thesis discusses some spatial discretization schemes and time integration methods that are designed to reduce the likelihood of spurious oscillations in the numerical solution of a one-dimensional hyperbolic conservation law. The Essentially Non-Oscillatory and Weighted Essentially Non-Oscillatory schemes as well as the Monotonic Upstream Centred Scheme for Conservation Laws are considered for the spatial discretization. Strong-Stability-Preserving Runge-Kutta methods are considered for the time evolution. In addition to the results given by these methods, some stability restrictions are considered inluding linear stability step size restrictions, the CFL condition, and the theoretical and practical monotonicity conditions for a class of second-order explicit Runge-Kutta methods.

Chapter 1

Introduction

Classically partial differential equations (PDEs) often fall into one of three distinct categories, parabolic, elliptic, or hyperbolic; see, e.g., [1]. In this thesis we focus on a particular class of hyperbolic PDEs known as one-dimensional (1D) hyperbolic conservation laws, having the general form

$$u_t(x,t) + f(u(x,t))_x = 0,\ x \in (a,b),\ t > 0, \tag{1.1}$$

with initial condition

$$u(x,0) = u_0(x), \tag{1.2}$$

and either periodic boundary conditions,

$$u(a,t) = u(b,t), \tag{1.3}$$

or Dirichlet boundary condtiions

$$u(a,t) = A, \quad u(b,t) = B. \tag{1.4}$$

Hyperbolic PDEs arise in many applications including fluid dynamics, aerodynamics, astrophysics, ultrasound modeling, traffic dynamics, and also in the study of fiber optics, acoustic waves, seismic waves, and electromagnetic waves; see, e.g., [9].

In this thesis we examine some difficulties that arise when attempting to solve hyperbolic conservation laws numerically. In Chapter 3 and Chapter 4 we consider some numerical methods that can be used to correct these difficulties.

1.1 The Method of Lines

One general approach to the numerical solution of PDEs is called the *method of lines* (MOL), see, e.g. [13], [1]. This approach generates a system of ordinary differential equations (ODEs) from the original system of PDEs by discretizing the PDEs over a discrete set of x values, $\{x_i\}_{i=0}^{N}$, which partitions the domain (a, b). We call this discrete set of x values a *grid*. The idea behind the MOL is to reduce the PDEs, which depend on time and space, to a system of ODEs that depend only on time. The benefit of this procedure is that there is an abundance of high quality software to solve systems of ODEs; see, e.g., [2]. Discretization schemes based on finite differences are commonly used to reduce the PDEs to a system of ODEs. Finite differences use approximations to the solution at discrete values of x in approximations to the spatial derivatives. When the spatial derivatives are replaced by discrete solution values we then have ODEs which depend only on time.

When we have generated the ODEs from (1.1) we then have what is known as an initial value problem (IVP). In an IVP, conditions on the solutions are provided at some initial point. Consider, as an example, an IVP of the form

$$y' = g(t, y), \tag{1.5}$$

with initial condition

$$y(t_0) = \alpha. \tag{1.6}$$

Note that the initial condition is located at the initial point t_0.

Given a solution approximation y_n at t_n, the forward Euler [2] method gives an approximation to the solution y_{n+1} at t_{n+1} of the form

$$y_{n+1} = y_n + \Delta t g(t_n, y_n), \tag{1.7}$$

where

$$\Delta t = t_{n+1} - t_n.$$

When we integrate ODEs using the forward Euler method, there are restrictions on the step size, Δt, to consider. The restrictions on the step size maintain stability of the integration and the accuracy of the approximated solution. Provided $g(t, y)$

satisfies certain conditions, the error in the solution of an ODE in an IVP using the forward Euler method is bounded globally and locally by terms which are $O(\Delta t)$ and $O(\Delta t^2)$ respectively; see, e.g., [2]. For the test equation $y' = \lambda y$ with $\lambda < 0$, *linear stability* for the forward Euler method requires $-2 < |\Delta t \lambda| < 0$; see, e.g., [2].

If we consider an approximation to the solution $u(x,t)$ of (1.1) based on the discrete grid values , we represent this approximation as

$$\mathbf{U}(t) = \begin{bmatrix} U_0(t) \\ U_1(t) \\ \vdots \\ U_N(t) \end{bmatrix}, \text{ where } U_j(t) \approx u(x_j, t). \tag{1.8}$$

As well we define $\mathbf{U}^n \equiv \mathbf{U}(t_n)$ and $U_j^n \equiv U_j(t_n)$.

1.1.1 The Courant, Friedrichs, and Lewy condition

In addition to the standard IVP ODE stepsize restrictions mentioned in the previous section, the numerical solution of hyperbolic PDEs introduces another step size restriction which couples the spatial step and the time step; it is called the Courant, Friedrichs, Lewy (CFL) condition; see, e.g., [11]. The CFL condition states that the domain of dependence of the PDE must fall within the domain of dependence of the numerical scheme.

Characteristic lines are curves of constant value of the solution within the domain of the problem. One can think of characteristic lines as contour lines. Knowing the solution value at any point along the line specifies the constant solution value along the whole characteristic line. In particular, this constant value can be specified by the solution value at the point where a characteristic line meets the boundary.

If we consider Figure 1.1 and the characteristic line which traces back from the point C to the point Q on the initial value line, we can see there an example of a violation of the CFL condition. We observe in this figure that the numerical scheme has a combination of spatial subinterval size, Δx, and time step, Δt, such that the domain of dependence of the numerical scheme fails to include the domain of dependence (the characteristic line) of the PDE. Formally stated, the CFL condition for

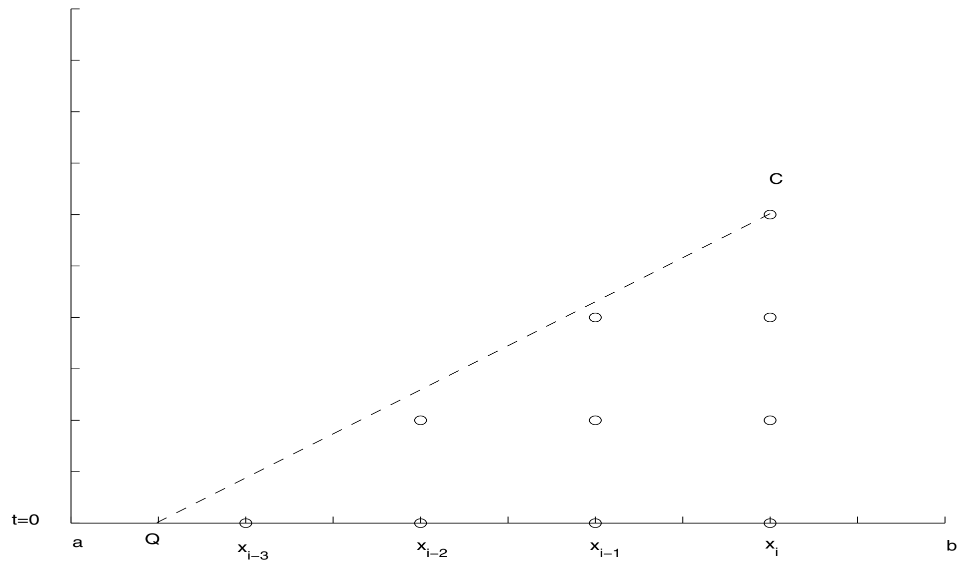

Figure 1.1: An example of a violation of the CFL condition.

the scalar advection equation, with α a constant,

$$u_t(x,t) + \alpha u_x(x,t) = 0,$$

is as follows [11]:

$$|\alpha|\Delta t \leq \Delta x. \tag{1.9}$$

In simple terms, if we advance too quickly in time, i.e., with too large a time step, to a point C (as in Figure 1.1) then the domain of dependence of the numerical scheme fails to include the characteristic line which passes through the point C. If this happens then we can expect the numerical scheme to be unstable, leading to large errors.

1.2 Nonlinear Stability and Spurious Oscillations

In this chapter we have discussed the linear stability step size requirement for the forward Euler method. Linear stability analysis is appropriate when applied to a linear hyperbolic PDE. However, most hyperbolic conservation laws are nonlinear and are subject to a different type of stability constraint known as *nonlinear stability* ([8]).

During the numerical solution of a hyperbolic conservation law, signs of nonlinear instability are most often indicated by the presence of numerical artifacts known as *spurious oscillations*. Spurious oscillations are overshoots in the numerical solution that are unrepresentative of the true solution. An example of a spurious oscillation can be seen in Figure 1.2. In order to help suppress spurious oscillations in the numerical solution, we need to require some type of nonlinear stability condition for the numerical methods used to solve hyperbolic conservation laws. A well known nonlinear stability condition is the total variation diminishing (TVD) condition [8]. The total variation of a numerical solution, $\mathbf{U}^n$, is defined as

$$TV(\mathbf{U}^n) = \sum_{j=1}^{N} |U_j^n - U_{j-1}^n|. \tag{1.10}$$

The total variation may be thought of as a rough approximation to the fluctuation in the solution at a given time. TVD methods have the property that the total variation of the numerical solution does not increase over time. In other words

$$TV(\mathbf{U}^{n+1}) \leq TV(\mathbf{U}^n). \tag{1.11}$$

The condition (1.11) is a restriction associated with nonlinear stability rather than linear stability. A stronger condition for nonlinear stability is the strong-stability-preserving (SSP) property. We refer to SSP with respect to the time integration; the SSP property guarantees that a time discretization is TVD. Unlike the TVD property (1.11), the SSP property holds in any norm; i.e.,

$$||U^{n+1}|| \leq ||U^n||. \tag{1.12}$$

1.3 Summary of Thesis

We begin by discussing some spatial discretizations which are designed to suppress spurious oscillations in the numerical solution of a hyperbolic conservation law. The theory behind these methods is presented and their respective convergence rates are

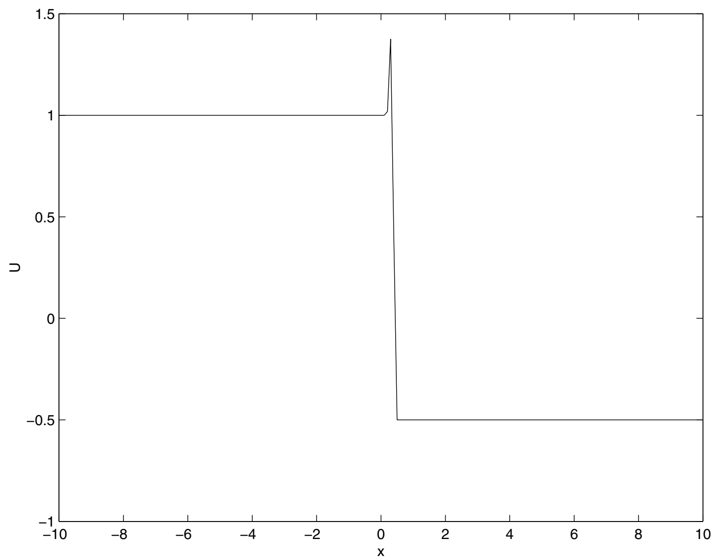

Figure 1.2: An example of a spurious oscillation

confirmed. Specifically, the Monotonic Upstream-Centered Scheme for Conservation Laws (MUSCL), Essentially Non-Oscillatory (ENO), Weighted Essentially Non-Oscillatory (WENO) spatial discretization schemes are discussed.

Once spatial discretizations are discussed we concern ourselves with the time integration methods we use to integrate the system of ODEs resulting from a discretization of the hyperbolic conservation law (1.1). We assume that a spatial discretization that suppresses spurious oscillations is employed, and then we consider a class of Runge-Kutta methods designed to suppress spurious oscillations during the time integration. Some common methods of this class are introduced and some of their properties discussed.

Finally, in Chapter 4 we introduce and examine some stability step size restrictions associated with the numerical solution of hyperbolic conservation laws. The analysis in this chapter includes determining step size restrictions for some general classes of methods. We consider step size restrictions associated with the time integration methods and the PDE problem.

Chapter 2

Spatial Discretizations

In this chapter, after establishing some basic definitions and the finite volume formulation for hyperbolic conservation laws, we verify the order of convergence for the MUSCL, ENO, and WENO spatial discretization schemes. These schemes are commonly used to provide the spatial discretization for hyperbolic conservation laws. We discuss the theory behind these schemes as well as their implementation.

2.1 Definitions

We assume a discretization of the spatial domain based on a discrete set of points that we refer to as the *grid*. We represent this grid with the points

$$a = x_{\frac{1}{2}} < x_{\frac{3}{2}} < \cdots < x_{N+\frac{1}{2}} = b,$$

where a and b are the endpoints of the spatial domain, and N is the number of subintervals into which the spatial domain is divided.

We define a *cell* $I_i = [x_{i-1/2}, x_{i+1/2}]$ and its mid-point $x_i = \frac{1}{2}(x_{i-1/2} + x_{i+1/2})$, for $i = 1, \ldots, N$. We also define a *stencil*, $S(i)$, which is a collection of cells built around the cell I_i, given by

$$S(i) \equiv \{I_{i-l}, \ldots, I_{i+r}\}.$$

In a stencil, $S(i)$, we have l cells to the left of the cell I_i and r cells to the right. The values l and r are referred to as the *left shift* and the *right shift* respectively, and the *size* of $S(i)$ is the number of cells contained in $S(i)$. The size of $S(i)$ is equal to $l + r + 1$.

We also define *cell averages*, $\bar{v}_i$, of some function $v(x)$ over a grid to be

$$\bar{v}_i \equiv \frac{1}{\Delta x_i} \int_{x_{i-1/2}}^{x_{i+1/2}} v(\xi) d\xi, \quad i = 1, 2, \ldots, N, \tag{2.1}$$

where $\Delta x_i = x_{i+1/2} - x_{i-1/2}$.

2.2 A Finite Volume Formulation for the 1D Conservation Law

The ENO and WENO spatial discretization schemes, which are designed to prevent the growth of spurious oscillations in the numerical solution due to the spatial discretization, can be applied after (1.1) is transformed using a finite volume formulation. For simplicity we will assume that (1.1) is subject to periodic boundary conditions, (1.3).

Rather than treating (1.1) directly by using a traditional spatial discretization based on point values (e.g., involving solution approximations at points in the grid), we first integrate (1.1) in space over the cell I_i,

$$\int_{x_{i-1/2}}^{x_{i+1/2}} \frac{\partial}{\partial t} u(x,t)dx + \int_{x_{i-1/2}}^{x_{i+1/2}} f(u(x,t))_x dx = 0, \quad i = 1, \ldots, N,$$

to obtain

$$\frac{\partial}{\partial t} \int_{x_{i-1/2}}^{x_{i+1/2}} u(x,t)dx + \big(f(u(x_{i+1/2},t)) - f(u(x_{i-1/2},t))\big) = 0, \quad i = 1, \ldots, N.$$

After dividing through by Δx_i, we obtain

$$\frac{\partial}{\partial t} \frac{1}{\Delta x_i} \int_{x_{i-1/2}}^{x_{i+1/2}} u(x,t)dx \quad + \quad \frac{f(u(x_{i+1/2},t)) - f(u(x_{i-1/2},t))}{\Delta x_i} = 0, \tag{2.2}$$

$$i = 1, \ldots, N.$$

Using the definition of a cell average (2.1) we can rewrite (2.2) as the set of ODEs that we will refer to as the *finite volume formulation*:

$$\frac{d\bar{u}_i(t)}{dt} = -\frac{1}{\Delta x_i}(f(u(x_{i+1/2},t)) - f(u(x_{i-1/2},t))), \quad i = 1, \ldots, N, \tag{2.3}$$

where $\bar{u}_i(t)$ is the cell average (2.1) of the solution $u(x,t)$ over cell I_i at time t; i.e.,

$$\bar{u}_i(t) \equiv \frac{1}{\Delta x_i} \int_{x_{i-1/2}}^{x_{i+1/2}} u(\xi,t)d\xi.$$

During the $(n+1)$st time step from t_n to t_{n+1}, we approximate the right-hand side of (2.3), by approximating $f(u(x_{i+1/2},t))$ and $f(u(x_{i-1/2},t))$ by $f_{i+1/2}(t)$ and $f_{i-1/2}(t)$, which are defined below in (2.6). Since we are approximating the right-hand side of (2.3), as in (2.4), we replace $\bar{u}_i(t)$ by the solution approximations $\bar{U}_i(t)$ and get the ODE system

$$\frac{d\bar{U}_i(t)}{dt} = -\frac{1}{\Delta x_i}(f_{i+1/2}(\bar{\mathbf{U}}(t)) - f_{i-1/2}(\bar{\mathbf{U}}(t))), \quad i = 1, \ldots, N, \tag{2.4}$$

where

$$\bar{\mathbf{U}}(t) = \begin{bmatrix} \bar{U}_0(t) \\ \bar{U}_1(t) \\ \vdots \\ \bar{U}_N(t) \end{bmatrix}. \tag{2.5}$$

In (2.4), $f_{i+1/2}(\bar{\mathbf{U}}(t))$ and $f_{i-1/2}(\bar{\mathbf{U}}(t))$ are given by

$$f_{i+1/2}(\bar{\mathbf{U}}(t)) = h(U^-_{i+1/2}(\bar{\mathbf{U}}(t)), U^+_{i+1/2}(\bar{\mathbf{U}}(t))) \text{ and} \tag{2.6a}$$

$$f_{i-1/2}(\bar{\mathbf{U}}(t)) = h(U^-_{i-1/2}(\bar{\mathbf{U}}(t)), U^+_{i-1/2}(\bar{\mathbf{U}}(t))) \tag{2.6b}$$

where $U^+_{i+1/2}(\bar{\mathbf{U}}(t))$ and $U^-_{i+1/2}(\bar{\mathbf{U}}(t))$ are linear combinations of the $\bar{U}_i(t)$ values; see (2.27), and where $h(\tilde{u}_1, \tilde{u}_2)$ is a *monotone flux* (to be defined below in (2.7)).

In this thesis the values $U^-_{i+1/2}(\bar{\mathbf{U}}(t))$, $U^+_{i+1/2}(\bar{\mathbf{U}}(t))$, $U^-_{i-1/2}(\bar{\mathbf{U}}(t))$, $U^+_{i-1/2}(\bar{\mathbf{U}}(t))$ are given by either an ENO or a WENO spatial discretization scheme (to be discussed in sections 2.4 and 2.5 respectively). We will later see that $U^-_{i+1/2}(\bar{\mathbf{U}}(t))$, $U^+_{i+1/2}(\bar{\mathbf{U}}(t))$, $U^-_{i-1/2}(\bar{\mathbf{U}}(t))$, $U^+_{i-1/2}(\bar{\mathbf{U}}(t))$ are obtained from linear combinations of the cell averages $\bar{U}_j(t)$, $j = 1, \ldots, N$; therefore the right-hand side of (2.4) in fact depends on $\bar{U}_j(t)$, $j = 1, \ldots, N$. $h(\tilde{u}_1, \tilde{u}_2)$ is a monotone flux used in order to help achieve appropriate upwinding. In [14] a monotone flux is defined as a function satisfying three properties. First, $h(\tilde{u}_1, \tilde{u}_2)$ is a Lipschitz continuous function in both arguments. Second, $h(\tilde{u}_1, \tilde{u}_2)$ is a nondecreasing function in $\tilde{u}_1$ and a nonincreasing function in $\tilde{u}_2$. Finally, $h(\tilde{u}_1, \tilde{u}_2)$ must be consistent with the physical flux $f(u(x,t))$; i.e., $h(\tilde{u}_1, \tilde{u}_1) = f(\tilde{u}_1)$.

In this thesis the monotone flux we consider is the Godunov flux [14]

$$h(\tilde{u}_1, \tilde{u}_2) = \begin{cases} \min_{\tilde{u}_1 \le U \le \tilde{u}_2} f(U), & \text{if } \tilde{u}_1 \le \tilde{u}_2, \\ \max_{\tilde{u}_1 \le U \le \tilde{u}_2} f(U), & \text{if } \tilde{u}_1 > \tilde{u}_2. \end{cases} \tag{2.7}$$

It is assumed that the physical flux, $f(u(x,t))$, is convex ($f(u(x,t))_{xx} \neq 0$). By making this assumption, $f(u(x,t))$ is often a simple function whose maximum or minimum in the monotone flux (2.7) is easily calculated. For example, the linear advection equation has $f(u(x,t)) = u(x,t)$, and $h(\tilde{u}_1, \tilde{u}_2)$ is thus given by the minimum of $\tilde{u}_1$ and $\tilde{u}_2$ when $\tilde{u}_1 \leq \tilde{u}_2$, or by the maximum of $\tilde{u}_1$ and $\tilde{u}_2$ when $\tilde{u}_1 > \tilde{u}_2$.

Once we have a way of generating the ODE system (2.4), we may then use an initial-value problem solver to integrate the solution from the initial time t_0 to some final time t_f. For every point $x_{i+1/2}$ of the grid, at the nth time step, t_n, an ENO or a WENO scheme will give us a way of determining an approximation to the solution from the left and right of $x_{i+1/2}$, for $i = 1, \ldots, N$.

2.3 The MUSCL Discretization Scheme

In this section we describe the MUSCL spatial discretization scheme and confirm its order of convergence. This spatial discretization is taken from [4], where it is claimed to be second order in the 1-norm, where the 1-norm for a vector $\mathbf{z}$,

$$\mathbf{z} = \begin{bmatrix} z_0 \\ z_1 \\ \vdots \\ z_N \end{bmatrix},$$

is given by

$$\frac{\sum_{i=0}^{N} |z_i|}{N+1}. \tag{2.8}$$

The MUSCL scheme is intended to prevent the introduction of spurious oscillations in the numerical solution due to the spatial discretization. Using this method for the spatial discretization allows us to perform experiments and attribute any spurious oscillations that arise to the time discretization used.

The MUSCL spatial discretization requires an approximation of (1.1) different from (2.3). The MUSCL spatial discretization first approximates the physical flux in (1.1) using the conservative difference

$$f(u(x,t))_x \approx \frac{\hat{f}_{i+1/2}(t) - \hat{f}_{i-1/2}(t)}{\Delta x}, \text{ for } i = 1, \dots, N,$$

where the values $\hat{f}_{i+1/2}(t)$ and $\hat{f}_{i-1/2}(t)$ are defined in terms of the Godunov flux (2.7), as

$$\hat{f}_{i+1/2}(t) = h(\hat{U}^{-}_{i+1/2}(t), \hat{U}^{+}_{i+1/2}(t)) \text{ and } \hat{f}_{i-1/2}(t) = h(\hat{U}^{-}_{i-1/2}(t), \hat{U}^{+}_{i-1/2}(t)). \tag{2.9}$$

The solution approximations in (2.9),

$$\hat{U}^{+}_{i+1/2}(t) \approx \lim_{x \to x^{+}_{i+1/2}} u(x,t), \text{ and } \hat{U}^{-}_{i+1/2}(t) \approx \lim_{x \to x^{-}_{i+1/2}} u(x,t),$$

are calculated using

$$\hat{U}^{-}_{i+1/2}(t_n) = U^n_i + \frac{1}{2}\text{minmod}(U^n_{i+1} - U^n_i, U^n_i - U^n_{i-1}),$$

and

$$\hat{U}^{+}_{i+1/2}(t_n) = U^n_{i+1} - \frac{1}{2}\text{minmod}(U^n_{i+2} - U^n_{i+1}, U^n_{i+1} - U^n_i),$$

where $U^n_i \approx u(x_i, t_n)$ and the function $\text{minmod}(\tilde{u}_1, \tilde{u}_2)$ is given by

$$\text{minmod}(\tilde{u}_1, \tilde{u}_2) = \frac{\text{sign}(\tilde{u}_1) + \text{sign}(\tilde{u}_2)}{2} \min(|\tilde{u}_1|, |\tilde{u}_2|),$$

where

$$\text{sign}(\tilde{u}_1) = \begin{cases} 1 & \text{if } \tilde{u}_1 > 0, \\ -1 & \text{if } \tilde{u}_1 < 0, \\ 0 & \text{if } \tilde{u}_1 = 0. \end{cases}$$

The MUSCL discretization is TVD under the condition [4],

$$\Delta t_{FE} \leq \frac{\Delta x}{2 \max_j |U^n_j|}, \tag{2.10}$$

for any one-dimensional conservation law [5], where Δt_{FE} is the step size taken by the forward Euler method (1.7).

$$
\begin{array}{c|cccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{5} & \frac{1}{5} & 0 & 0 & 0 & 0 & 0 \\
\frac{3}{10} & \frac{3}{40} & \frac{9}{40} & 0 & 0 & 0 & 0 \\
\frac{4}{5} & \frac{44}{55} & -\frac{56}{15} & \frac{32}{9} & 0 & 0 & 0 \\
\frac{8}{9} & \frac{19372}{6561} & -\frac{25360}{2187} & \frac{64448}{6561} & -\frac{212}{729} & 0 & 0 \\
1 & \frac{9017}{3168} & -\frac{355}{33} & \frac{46732}{5247} & \frac{49}{176} & -\frac{5103}{18656} & 0 \\
\hline
 & \frac{35}{384} & 0 & \frac{500}{1113} & \frac{125}{192} & -\frac{2187}{6784} & \frac{11}{84}
\end{array}
$$

Table 2.1: Butcher tableau for the Dormand-Prince fifth-order Runge-Kutta method.

To measure the convergence rate, we consider the advection equation which has

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

in (1.1) on $(0, 2\pi)$ with smooth initial data $u(x, 0) = \sin(x)$, and periodic boundary conditions $u(0, t) = u(2\pi, t)$. We integrate from time $t_0 = 0$ to time $t_f = 1$ with $\frac{\Delta t}{\Delta x} = \frac{1}{2}$ so that (2.10) is satisfied[1]. We integrate in time using the fifth-order Dormand-Prince Runge-Kutta method having the Butcher tableau in Table 2.1[2]. The solution to this problem has a smooth initial condition and develops no shocks, hence a SSP time integrator is not required. The number of spatial (and hence temporal) subintervals is doubled for each trial, and the convergence rate is then calculated by taking $\log_2$ of the ratio of the errors. The results found in Table 2.2 show that the MUSCL spatial discretization scheme is second-order in the 1-norm, as predicted by the theory for these schemes.

2.4 ENO Schemes

2.4.1 Reconstruction

We begin with a discussion of ENO schemes in terms of a given function $v(x)$. The following discussion is largely taken from [14] and discusses the *reconstruction* of solution approximations at the grid points starting from the cell average data.

---

[1]The exact solution for this problem is $u(x,t) = \sin(x - t)$, so we have $\max_j |U_j^n| = 1$, and by (2.10) we choose the upper bound on the step size restriction, giving $\Delta t = \frac{\Delta x}{2}$.

[2]The connection between a Runge-Kutta method and the corresponding Butcher tableau is explained in Chapter 3.

Test for Convergence		
N	L_1	rate
10	6.9545×10^{-2}	-
20	2.8808×10^{-2}	1.27
40	8.3711×10^{-3}	1.78
80	2.3289×10^{-3}	1.85
160	6.2380×10^{-4}	1.90
320	1.6711×10^{-4}	1.90

Table 2.2: Convergence of the MUSCL scheme: Advection equation, time integration from $t_0 = 0$ to $t_f = 1$ using the Dormand-Prince fifth-order Runge-Kutta method, $u(x,0) = \sin(x)$, $x \in [0, 2\pi]$, $\frac{\Delta t}{\Delta x} = \frac{1}{2}$, and periodic boundary conditions.

Let $p(x)$ be the unique interpolating polynomial of degree at most $k-1$, specified by the integral interpolation conditions

$$\frac{1}{\Delta x_j} \int_{x_{j-1/2}}^{x_{j+1/2}} p(\xi) d\xi = \bar{v}_j, \quad j = i-l, ..., i+r, \tag{2.11}$$

where $k = l + r + 1$. These interpolation conditions are applied over the cells of the stencil, $S(i)$, associated with the cell I_i. Then, $p(x)$ is a kth-order-accurate approximation of $v(x)$ in I_i [14]; i.e.,

$$p(x) = v(x) + O(\Delta x^k), \quad x \in I_i. \tag{2.12}$$

(We will prove this later in this section.)

We want to use $p(x)$ to give approximations to the function values at the cell boundaries; i.e., $v(x_{i+1/2}) \approx p(x_{i+1/2})$ and $v(x_{i-1/2}) \approx p(x_{i-1/2})$. Let

$$v_{i+1/2} := p(x_{i+1/2}), \quad v_{i-1/2} := p(x_{i-1/2}), \quad i = 1, 2, ..., k. \tag{2.13}$$

Note that since $p(x)$ is determined from the imposition of the interpolation conditions (2.11), the coefficients of $p(x)$ depend on the cell averages $\bar{v}_i$ associated with $S(i)$. It then follows that evaluation of $p(x)$ at $x_{i+1/2}$ or $x_{i-1/2}$ leads to expressions for $v_{i+1/2}$ and $v_{i-1/2}$ as linear combinations of the cell averages associated with $S(i)$:

$$v_{i+1/2} = \sum_{j=0}^{k-1} c_{l,j} \bar{v}_{i-l+j}, \quad v_{i-1/2} = \sum_{j=0}^{k-1} c_{l-1,j} \bar{v}_{i-l+j}. \tag{2.14}$$

We have just explained how it is possible to construct a kth-order accurate approximation to the function values at the grid points using a linear combination of the cell averages associated with the k cells of $S(i)$. Next we look at how to calculate the coefficients $\{c_{l,j}\}$ arising in (2.14). First consider the *primitive function*

$$V(x) := \int_{-\infty}^{x} v(\xi)d\xi. \tag{2.15}$$

We can express the primitive function at $x_{i+1/2}$ as

$$V(x_{i+1/2}) = \sum_{j=-\infty}^{i} \int_{x_{j-1/2}}^{x_{j+1/2}} v(\xi)d\xi = \sum_{j=-\infty}^{i} \bar{v}_j \Delta x_j, \tag{2.16}$$

using (2.1).

This gives us the value of the $V(x)$ at the cell boundaries and in particular at the $k+1$ grid points,

$$x_{i-l-1/2}, \ldots, x_{i+r+1/2},$$

which represent the boundaries of the k cells included in $S(i)$. Next we observe that there exists a unique polynomial $P(x)$ of degree at most k which interpolates $(x_j, V(x_j)), \quad j = i-l-1/2, \ldots, i+r+1/2$. Note that from (2.11), $p(x) \equiv P'(x)$. To verify this, substitute $P'(x)$ for $p(x)$ in (2.11) to get

$$\begin{aligned}
\frac{1}{\Delta x_j} \int_{x_{j-1/2}}^{x_{j+1/2}} P'(x)dx &= \frac{1}{\Delta x_j} \left(P(x)\big|_{x_{j-1/2}}^{x_{j+1/2}} \right) \\
&= \frac{1}{\Delta x_j} (P(x_{j+1/2}) - P(x_{j-1/2})) \\
&= \frac{1}{\Delta x_j} (V(x_{j+1/2}) - V(x_{j-1/2})) \\
&= \frac{1}{\Delta x_j} \left(\sum_{l=-\infty}^{j} \bar{v}_l \Delta x_l - \sum_{l=-\infty}^{j-1} \bar{v}_l \Delta x_l \right) \\
&= \frac{1}{\Delta x_j} (\bar{v}_j \Delta x_j) = \bar{v}_j.
\end{aligned}$$

Therefore $P'(x)$ satisfies the interpolation conditions (2.11), and this implies $p(x) \equiv P'(x)$, due to the uniqueness of $p(x)$.

We next express $P(x)$ in terms of a Lagrange interpolating polynomial:

$$P(x) = \sum_{m=0}^{k} V(x_{i-l+m-1/2}) \prod_{s=0, s\neq m}^{k} \frac{x - x_{i-l+s-1/2}}{x_{i-l+m-1/2} - x_{i-l+s-1/2}}. \tag{2.17}$$

Next, we use the property[3] that

$$\sum_{m=0}^{k} \prod_{s=0, s\neq m}^{k} \frac{x - x_{i-l+s-1/2}}{x_{i-l+m-1/2} - x_{i-l+s-1/2}} \equiv 1. \tag{2.18}$$

Equations (2.17) and (2.18) allow us to express the subtraction of $V(x_{i-l-1/2})$ from $P(x)$ in the following way:

$$\begin{aligned} P(x) - V(x_{i-l-1/2}) = & \sum_{m=0}^{k} V(x_{i-l+m-1/2}) \prod_{s=0, s\neq m}^{k} \frac{x - x_{i-l+s-1/2}}{x_{i-l+m-1/2} - x_{i-l+s-1/2}} \\ & - V(x_{i-l-1/2}) \sum_{m=0}^{k} \prod_{s=0, s\neq m}^{k} \frac{x - x_{i-l+s-1/2}}{x_{i-l+m-1/2} - x_{i-l+s-1/2}} \end{aligned}$$

$$= \sum_{m=0}^{k} (V(x_{i-l+m-1/2}) - V(x_{i-l-1/2})) \prod_{s=0, s\neq m}^{k} \frac{x - x_{i-l+s-1/2}}{x_{i-l+m-1/2} - x_{i-l+s-1/2}}, \tag{2.19}$$

which is a Lagrange interpolant of the data $(x_j, V(x_j) - V(x_{i-l-1/2}))$, $j = i - l - 1/2, \ldots, i + r + 1/2$. Taking the derivative of (2.19) we get

$$p(x) = \sum_{m=0}^{k} (V(x_{i-l+m-1/2}) - V(x_{i-l-1/2})) \left(\frac{\displaystyle\sum_{s=0, s\neq m}^{k} \prod_{q=0, q\neq m,s}^{k} (x - x_{i-l+q-1/2})}{\displaystyle\prod_{s=0, s\neq m}^{k} (x_{i-l+m-1/2} - x_{i-l+s-1/2})} \right). \tag{2.20}$$

[3]The proof of (2.18) follows by considering a polynomial of degree at most k, given by $Q(x) := f(x) - 1$ where $f(x)$ is the left-hand side of (2.18). $Q(x)$ has $k+1$ distinct roots, one at each grid point. By the fundamental theorem of algebra, a polynomial of degree at most k, with $k+1$ distinct roots must be identically zero. Thus, $f(x) \equiv 1$.

From (2.16) we see that

$$V(x_{i-l+m-1/2}) - V(x_{i-l-1/2}) = \sum_{j=0}^{m-1} \bar{v}_{i-l+j}\Delta x_{i-l+j}.$$

We then substitute the above into (2.20) to obtain

$$p(x) = \sum_{m=0}^{k}\sum_{j=0}^{m-1} \bar{v}_{i-l+j}\Delta x_{i-l+j} \left(\frac{\displaystyle\sum_{s=0,s\neq m}^{k} \prod_{q=0,q\neq m,s}^{k} (x - x_{i-l+q-1/2})}{\displaystyle\prod_{s=0,s\neq m}^{k} (x_{i-l+m-1/2} - x_{i-l+s-1/2})} \right). \tag{2.21}$$

If we evaluate (2.21) at $x = x_{i+1/2}$ and rearrange the summations, we get

$$p(x_{i+1/2}) = \sum_{j=0}^{k-1} \left(\sum_{m=j+1}^{k} \frac{\displaystyle\sum_{s=0,s\neq m}^{k} \prod_{q=0,q\neq m,s}^{k} (x_{i+1/2} - x_{i-l+q-1/2})}{\displaystyle\prod_{s=0,s\neq m}^{k} (x_{i-l+m-1/2} - x_{i-l+s-1/2})} \right) \Delta x_{i-l+j}\bar{v}_{i-l+j}. \tag{2.22}$$

Comparing equation (2.22) to (2.14) (and noting (2.13)) we easily see the coefficients $c_{l,j}$ are

$$c_{l,j} = \left(\sum_{m=j+1}^{k} \frac{\displaystyle\sum_{s=0,s\neq m}^{k} \prod_{q=0,q\neq m,s}^{k} (x_{i+1/2} - x_{i-l+q-1/2})}{\displaystyle\prod_{s=0,s\neq m}^{k} (x_{i-l+m-1/2} - x_{i-l+s-1/2})} \right) \Delta x_{i-l+j}. \tag{2.23}$$

Once the coefficients $c_{l,j}$ are available, it is a simple matter of using (2.14) in order to calculate the function approximations at the boundaries of the cell I_i given the cell averages $\bar{v}_{i-l+j}$, where $l = 0 \ldots k-1$ and $j = 0 \ldots k-1$. If Δx_i is a constant, Δx, then (2.23) simplifies to

$$c_{l,j} = \left(\sum_{m=j+1}^{k} \frac{\displaystyle\sum_{s=0,s\neq m}^{k} \prod_{q=0,q\neq m,s}^{k} (l-q+1)}{\displaystyle\prod_{s=0,s\neq m}^{k} (m-s)} \right). \tag{2.24}$$

2.4.2 Choosing an Appropriate Stencil

Given a stencil $S(i)$ built around the cell I_i and the corresponding cell averages, we have seen that a kth order approximation to an unknown function, $v(x)$, can be obtained over I_i and in particular at the cell boundaries. However, this raises the question of how to appropriately build such a stencil. For the cell I_i there are k possible stencils (consisting of consecutive cells) of size k which can be built that include I_i. We wish to choose the candidate stencil over which there is the least change in the underlying function $v(x)$ (this is quantified later in this section). In order to gain a measure of the change in the underlying function over a stencil, we compute a higher-order divided difference of the primitive function (2.15) over the stencil. This divided difference is the coefficient of the error term of the Newton interpolant of the primitive function and is an indicator of the smoothness of the interpolant. The jth divided difference over the stencil is calculated recursively using

$$V[x_{i-1/2},...,x_{i+j-1/2}] := \frac{V[x_{i+1/2},...,x_{i+j-1/2}] - V[x_{i-1/2},...,x_{i+j-3/2}]}{x_{i+j-1/2} - x_{i-1/2}}, \qquad (2.25)$$

where $j = 2, \ldots, k$. Using (2.1) and (2.15), the 2-point divided differences ($j = 1$) are equal to the cell averages:

$$V[x_{i-1/2}, x_{i+1/2}] = \bar{v}_i.$$

Intuitively, the candidate stencil with the least amount of change suggested in the underlying function is the least likely to contain a discontinuity. This stencil is the one we wish to select for the construction of our function approximations at the boundaries of cell I_i. This is because if an interpolant is constructed using data from a function $v(x)$ containing a discontinuity, then the interpolant may contain oscillations near the region of the discontinuity which are likely unrepresentative of the function being approximated.

If the primitive function $V(x)$ is discontinuous within the region of the problem domain covered by a given stencil, then the numerator of any divided difference (2.25) involving points on opposite sides of the discontinuity will be $O(1)$, and the jth divided differences associated with the stencil behave like $O\left(\Delta x^{-j}\right)$ where Δx is the largest cell size. A large divided difference associated with a stencil indicates the presence of a discontinuity within that stencil. Hence, the stencil to be associated with the cell I_i is chosen to be the one with the smallest divided differences.

Rather than determining the divided differences for each of the k stencils of size k that include I_i, we perform a sequence of steps to determine the appropriate stencil. We begin the ENO scheme algorithm by initializing the current stencil to include only the cell I_i. We then consider the two candidate stencils formed by including either I_{i-1} or I_{i+1}, and calculate the respective divided differences ((2.25) with $j = 2$) over each. Then the current stencil is updated to be the candidate stencil with the smallest divided difference (i.e., the smoothest one). Once this is done, the process is repeated until the current stencil consists of k cells, as desired. This stencil construction and the subsequent reconstruction of function approximations at the cell boundaries of I_i that define ENO schemes.

2.4.3 ENO Schemes for the 1D Conservation Law

Assume that the hyperbolic conservation law (1.1) has been rewritten in terms of the finite volme formulation (2.3) and that the ODEs in (2.3) are approximated by the ODEs (2.4), which employ flux approximations (2.6). Also, assume from the initial condition specifying $u(x, 0)$ that the initial cell averages $\bar{U}_i^0$, defined by (2.1), are calculated from

$$\bar{U}_i^0 \equiv \frac{1}{\Delta x_i} \int_{x_{i-1/2}}^{x_{i+1/2}} u(\xi, 0) d\xi, \quad i = 1, 2, \ldots, N. \tag{2.26}$$

These $\bar{U}_i^0$ values provide the initial values at $t = 0$ for the ODEs (2.4). Note that we have identified the underlying function $v(x)$ in the ENO theory as the true solution of (1.1), where at some time fixed time t_n, $u(x, t_n)$ can be viewed as a function of only x.

Given cell averages $\bar{U}_i^n$ at time t_n, we calculate the solution approximations at the endpoints of the cell I_i. We denote these solution approximations by $U_{i+1/2}^-(\bar{\mathbf{U}}(t_n)) \approx u(x_{i+1/2}, t_n)$ and $U_{i-1/2}^+(\bar{\mathbf{U}}(t_n)) \approx u(x_{i-1/2}, t_n)$. Using (2.14), this yields

$$U^{-}_{i+1/2}(\bar{\mathbf{U}}(t_n)) = \sum_{j=0}^{k-1} c_{l,j}\bar{U}^n_{i-l+j}, \quad U^{+}_{i-1/2}(\bar{\mathbf{U}}(t_n)) = \sum_{j=0}^{k-1} c_{l-1,j}\bar{U}^n_{i-l+j}, \tag{2.27}$$

where the $c_{l,j}$ are found using (2.24). The solution approximations $U^{n(-)}_{i+1/2}$ and $U^{n(+)}_{i-1/2}$ are found for $i = 1, \ldots, N$. These values are then used in the flux approximation (2.6) to calculate the right-hand sides of the ODEs (2.4), which are integrated from t_n to t_{n+1} using an appropriate initial-value solver. This gives the cell average approximations $\bar{U}^{n+1}_i$, $i = 1, \ldots, N$, at the time t_{n+1}. We then repeat the above steps until the final time t_f is reached.

2.4.4 ENO Scheme Experiments

In this section we apply the principles we have discussed concerning ENO spatial discretizations in some numerical experiments. Consider the advection equation

$$u_t(x,t) + u_x(x,t) = 0, \ x \in (0,2), \ 0 < t < t_f, \tag{2.28}$$

subject to the initial condition $u(x,0) = \sin(\pi x)$ and periodic boundary conditions, $u(0,t) = u(2,t)$. We integrate the ODEs (2.4) in the following three experiments to time $t_f = 1$. The *refinement path* $\frac{\Delta t}{\Delta x} = \frac{1}{5}$ is used; it is chosen in order not to violate any time step size restrictions (to be discussed later in Chapters 3 and 4). We note that our refinement path specifies a fixed ratio between the time step Δt, and the cell width, Δx. In these experiments we wish to test the convergence rate of ENO spatial discretization schemes for various stencils of size k.

We integrate the ODEs in time for $k = 2$, 3, and 4 using the Dormand-Prince fifth-order Runge-Kutta method (Table 2.1). We use this high-order Runge-Kutta method for the time integration so that any significant contribution to the error of the approximate solution can be attributed to the spatial discretization, which is of lower order for the chosen k values. There is no need to employ a SSP Runge-Kutta method because we have smooth initial conditions and no shocks form in the solution to this problem.

The tests for convergence were run using the stencil sizes $k = 2$, 3, 5. The results may be seen in Tables 2.3, 2.4, and 2.5 respectively. We expect convergence rates of

Test for Convergence		
N	L_1	rate
10	1.1426×10^{-1}	-
20	3.9190×10^{-2}	1.54
40	1.0846×10^{-2}	1.85
80	2.8585×10^{-3}	1.92
160	7.3603×10^{-4}	1.96

Table 2.3: Convergence of ENO schemes: Advection equation, time integration from $t_0 = 0$ to $t_f = 1$ using Dormand-Prince fifth-order Runge-Kutta method, $u(x, 0) = \sin(\pi x)$, $x \in [0, 2]$, $\frac{\Delta t}{\Delta x} = \frac{1}{5}$, periodic boundary conditions, and $k = 2$.

Test for Convergence		
N	L_1	rate
10	3.8137×10^{-2}	-
20	5.0491×10^{-3}	2.91
40	6.4303×10^{-4}	2.97
80	8.0657×10^{-5}	2.99
160	1.0091×10^{-5}	2.99

Table 2.4: Convergence of ENO schemes: Advection equation, time integration from $t_0 = 0$ to $t_f = 1$ using Dormand-Prince fifth-order Runge-Kutta method, $u(x, 0) = \sin(\pi x)$, $x \in [0, 2]$, $\frac{\Delta t}{\Delta x} = \frac{1}{5}$, periodic boundary conditions, and $k = 3$.

approximately k for ENO spatial discretization schemes because ENO schemes are $O(\Delta x^k)$. In Tables 2.3, 2.4, and 2.5 we observe the expected convergence rates of approximately k in the 1-norm.

2.4.5 Conclusions for ENO Schemes

We now summarize the results presented regarding ENO spatial discretizations. First, given a set of cell averages for a function $v(x)$ over a grid $\{x_{i+1/2}\}_{i=0}^{N}$ on the problem domain, we employ Newton divided differences of the cell averages to construct a kth-order stencil, $S(i)$, for each cell I_i. From this stencil for I_i we use (2.11) (derived from the associated interpolating polynomial to the cell averages over the stencil), where the coefficients come from (2.23), in order to yield an $O(\Delta x^k)$ approximation to the function values at the endpoints of I_i.

In order to solve the 1D hyperbolic conservation law (1.1) subject to periodic

Test for Convergence		
N	L_1	rate
10	2.2909×10^{-2}	-
20	1.4623×10^{-3}	3.97
40	1.0767×10^{-4}	3.76
80	7.2628×10^{-6}	3.89
160	4.8032×10^{-7}	3.92

Table 2.5: Convergence of ENO schemes: Advection equation, time integration from $t_0 = 0$ to $t_f = 1$ using Dormand-Prince fifth-order Runge-Kutta method, $u(x, 0) = \sin(\pi x)$, $x \in [0, 2]$, $\frac{\Delta t}{\Delta x} = \frac{1}{5}$, periodic boundary conditions, and $k = 4$.

boundary conditions, we first assume that the PDE has been converted to the finite volume formulation (2.3). We approximate the finite volume formulation by the ODEs (2.4), having the monotone flux (2.6), where the solution approximations $U^-_{i+1/2}(\bar{\mathbf{U}}(t_n))$ and $U^+_{i+1/2}(\bar{\mathbf{U}}(t_n))$ are provided by the summations (2.27). Once the ODEs (2.4) have been advanced from t_n to t_{n+1}, the approximate cell averages at the time t_{n+1} are used to calculate solution approximations $U^-_{i+1/2}(\bar{\mathbf{U}}(t_{n+1}))$ and $U^+_{i+1/2}(\bar{\mathbf{U}}(t_{n+1}))$, which are again used in the monotone flux (2.6) of the ODEs (2.4). This process is repeated until the final time t_f is reached.

We observe that in each of the above three experiments, the expected convergence rate is achieved.

2.5 Weighted Essentially Non-Oscillatory Schemes

We have seen in the previous section how to construct an $O(\Delta x^k)$ approximation to a function described in terms of k cell averages, with cell width Δx. This approach provides a $O(\Delta x^k)$ approximation at the grid points but also at any point within a cell (assuming sufficient smoothness of the underlying function). In this section we show how to apply a WENO scheme to obtain an $O(\Delta x^{2k-1})$ approximation to the solution at the cell boundaries of I_i (but not throughout the cell).

The WENO schemes of Liu, Osher, and Chan [10] were designed to make some improvements over ENO schemes. One of the problems with ENO schemes is that round-off errors can affect the process of selecting a stencil. If the divided differences of two candidate stencils in the ENO scheme stencil construction process are very close, then the selection of a stencil may be subject to round-off error. Moreover, WENO schemes are easily parallelizable and perform at a comparable cost to ENO schemes when implemented on parallel processors.

The WENO schemes presented in [10] achieved $O(\Delta x^k)$ accuracy, and in fact after some refinement of the approach, $O(\Delta x^{k+1})$ accuracy was achieved. However, Jiang [6] improved upon these schemes, producing WENO schemes that achieve $O(\Delta x^{2k-1})$ accuracy. These latter schemes are the WENO schemes that we consider in this thesis.

2.5.1 Reconstruction

As we have seen, the ENO scheme reconstruction considers the stencil over which the underlying function is the smoothest in order to calculate solution approximations at the cell boundaries of cell I_i. Rather than selecting a single stencil over which the underlying function is the smoothest, WENO considers a convex combination of the function approximations provided by each of the k possible stencils that could be built about I_i.

Recall that the k candidate stencils associated with I_i are

$$S_l(i) \equiv \{x_{i-l}, \ldots, x_{i-l+k-1}\}, \quad l = 0, \ldots, k-1, \tag{2.29}$$

and each stencil leads to an approximation of $O(\Delta x^k)$ for $v(x_{i+1/2})$ (see (2.14)); i.e.,

$$\sum_{j=0}^{k-1} c_{l,j} \bar{v}_{i-l+j} = v(x_{i+1/2}) + O(\Delta x^k), \quad l = 0, \ldots, k-1. \tag{2.30}$$

After each of the approximations (2.30) are calculated, a WENO scheme takes a convex combination of these k approximations to determine an $O(\Delta x^{2k-1})$ approximation; i.e.,

$$\sum_{l=0}^{k-1} \omega_l \left(\sum_{j=0}^{k-1} c_{l,j} \bar{v}_{i-l+j} \right) = v(x_{i+1/2}) + O(\Delta x^{2k-1}), \tag{2.31}$$

where the coefficients in (2.31) satisfy $\omega_l > 0$ and

$$\sum_{l=0}^{k-1} \omega_l = 1. \tag{2.32}$$

Having $\omega_l > 0$ reduces the likelihood of cancellation problems. To see why (2.32) is necessary, substitute from (2.30) in (2.31), to get

$$\sum_{l=0}^{k-1} \omega_l \left(v(x_{i+1/2}) + O(\Delta x^k) \right) = v(x_{i+1/2}) + O(\Delta x^{2k-1}),$$

or

$$\left(\sum_{l=0}^{k-1} \omega_l \right) v(x_{i+1/2}) + O(\Delta x^k) = v(x_{i+1/2}) + O(\Delta x^{2k-1}). \tag{2.33}$$

We then see that in order to have agreement, to at least $O(\Delta x^k)$, we must have (2.32).

Note that from (2.31), we have a weighted sum of the cell averages $\bar{v}_{i-(k-1)} \dots \bar{v}_{i+(k-1)}$ that gives an $O(\Delta x^{2k-1})$ approximation to $v(x_{i+1/2})$. In this section we show how to find appropriate coefficients ω_l in order for (2.33) to hold.

We would like coefficients, ω_l, such that when the function $v(x)$ over $S_l(i)$ contains a discontinuity, the corresponding coefficient $\omega_l \ll 1$, in which case the contribution to the approximation (2.31) from that stencil is negligible.

The weights ω_l are guaranteed to satisfy (2.32) if we write them as

$$\omega_l = \frac{\alpha_l}{\sum_{s=0}^{k-1} \alpha_s}, \quad l = 0, \dots, k-1. \tag{2.34}$$

The α_l are given by

$$\alpha_l = \frac{d_l}{(\epsilon + \beta_l)^k}, \tag{2.35}$$

where ϵ is some small constant introduced in order to avoid division by zero if $\beta_l = 0$, where β_l is the indicator of smoothness specified by Jiang [6], and given by

$$\beta_l = \sum_{s=1}^{k-1} \int_{x-1/2}^{x+1/2} \Delta x^{2s-1} \left(\frac{\partial^s p_l^k(x)}{\partial^s x} \right)^2 dx. \tag{2.36}$$

Written out explicitly, the smoothness indicators for $k = 2$ are

$$\beta_0 = (\bar{v}_{i+1} - \bar{v}_i)^2, \tag{2.37}$$

$$\beta_1 = (\bar{v}_i - \bar{v}_{i-1})^2, \tag{2.38}$$

and for $k = 3$ are

$$\beta_0 = \frac{13}{12}(\bar{v}_i - 2\bar{v}_{i+1} + \bar{v}_{i+2})^2 + \frac{1}{4}(3\bar{v}_i - 4\bar{v}_{i+1} + \bar{v}_{i+2})^2, \tag{2.39}$$

$$\beta_1 = \frac{13}{12}(\bar{v}_{i-1} - 2\bar{v}_i + \bar{v}_{i+1})^2 + \frac{1}{4}(\bar{v}_{i-1} - \bar{v}_{i+1})^2, \tag{2.40}$$

$$\beta_2 = \frac{13}{12}(\bar{v}_{i-2} - 2\bar{v}_{i-1} + \bar{v}_i)^2 + \frac{1}{4}(\bar{v}_{i-2} - 4\bar{v}_{i-1} + 3\bar{v}_i)^2. \tag{2.41}$$

These expressions are taken from [7]. The smoothness indicators are larger in value for a stencil containing cells with discontinuities, leading to a smaller α_l value for the contribution of that stencil to the overall WENO reconstruction.

With a choice of ϵ and d_l values we can define the weights (2.34), (2.35), and calculate the higher-order function approximation (2.31). The ϵ value is chosen to be 1×10^{-6} [7]. The coefficients, d_l, must satisfy

$$\sum_{l=0}^{k-1} d_l = 1, \tag{2.42}$$

and for $k = 2$ the coefficients are $d_0 = \frac{2}{3}$, $d_1 = \frac{1}{3}$ [6]. Likewise, for $k = 3$, we have $d_0 = \frac{3}{10}$, $d_1 = \frac{3}{5}$, and $d_2 = \frac{1}{10}$ [6]. These values lead to $O(\Delta x^{2k-1})$ accuracy. The reason for this claim as well as the way in which one determines the d_l's is explained in more detail in the next sub-section.

2.5.2 Improved WENO Schemes

The following discussion outlines the improvements made to the WENO spatial discretization scheme to obtain $O(\Delta x^{2k-1})$ accuracy. It is largely taken from [7], to which the reader is referred for more detail.

If we consider every possible stencil of size k built about a cell I_i, then there are $2k-1$ possible cells to consider. Recall that the ith cell has a cell average $\bar{v}_i$ associated with it. Now consider an interpolant to all $2k-1$ cell averages. Let this interpolant be called $p_{k-1}^{2k-1}(x)$, where the superscript represents the number of cells in the stencil over which the interpolant is defined, and the subscript represents the left shift of the stencil. The interpolant $p_{k-1}^{2k-1}(x)$ is of degree less than $2k-1$ and satisfies $2k-1$ interpolation conditions of the form

$$\frac{1}{\Delta x}\int_{x_{j-1/2}}^{x_{j+1/2}} p_{k-1}^{2k-1}(\xi)d\xi = \bar{v}_j, \quad j = i-(k-1),\ldots,i+(k-1).$$

As we saw earlier during the discussion on ENO schemes, we have (see (2.12))

$$p_{k-1}^{2k-1}(x) = v(x) + O(\Delta x^{2k-1}), \quad x \in \cup S_l(i),$$

and in particular

$$p_{k-1}^{2k-1}(x_{i+1/2}) = v(x_{i+1/2}) + O(\Delta x^{2k-1}). \tag{2.43}$$

Consider k interpolants, $p_l^k(x)$, $l = 0,\ldots,k-1$, one for each of the candidate stencils, with the cell averages as the interpolation data and the interpolation conditions

$$\frac{1}{\Delta x}\int_{x_{j-1/2}}^{x_{j+1/2}} p_l^k(\xi)d\xi = \bar{v}_j, \quad j = i-l,\ldots,i+r.$$

It follows that

$$p_l^k(x) = v(x) + O(\Delta x^k), \quad x \in S_l(i),$$

and in particular

$$p_l^k(x_{i+1/2}) = v(x_{i+1/2}) + O(\Delta x^k). \tag{2.44}$$

These interpolants are of degree less than k. We wish to choose a weighted combination of the $p_l^k(x_{i+1/2})$ values, for $l = 0,\ldots,k-1$, such that

$$p_{k-1}^{2k-1}(x_{i+1/2}) = \sum_{l=0}^{k-1} d_l p_l^k(x_{i+1/2}). \tag{2.45}$$

The significance of (2.45) is that it gives us a linear combination of $p_l^k(x_{i+1/2})$ values, each an $O(\Delta x^k)$ approximation to $v(x_{i+1/2})$, such that the sum of these values is an $O(\Delta x^{2k-1})$ approximation to $v(x_{i+1/2})$. The d_l values that satisfy (2.45) are used to define the corresponding α_l values in (2.35). From Section 2.4.1 we saw that

$$p_l^k(x_{i+1/2}) = \sum_{j=0}^{k-1} c_{l,j}^k \bar{v}_{i-l+j}, \quad l = 0,\ldots,k-1,$$

where the superscript on the coefficient $c^k_{l,j}$ will denote the size of the stencil over which corresponding ENO interpolant is defined. Similarly

$$p^{2k-1}_{k-1}(x_{i+1/2}) = \sum_{j=0}^{2k-2} c^{2k-1}_{k-1,j}\bar{v}_{i-(k-1)+j}.$$

By matching the cell average coefficients from the left-hand side and the right-hand side of (2.45), we can determine d_l, $l = 0,\ldots,k-1$, by solving the upper triangular system

$$\begin{bmatrix} c^k_{0,0} & c^k_{1,1} & \cdots & c^k_{k-1,k-1} \\ 0 & c^k_{1,0} & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & \cdots & c^k_{k-1,0} \end{bmatrix} \begin{pmatrix} d_0 \\ \vdots \\ \vdots \\ d_{k-1} \end{pmatrix} = \begin{pmatrix} c^{2k-1}_{k-1,k-1} \\ \vdots \\ \vdots \\ c^{2k-1}_{k-1,0} \end{pmatrix}. \tag{2.46}$$

Consider the following example for determining the d_l when $k = 2$. When we consider the WENO scheme with $k = 2$, the lower-order interpolating polynomials evaluated at $x_{i+1/2}$ are

$$p^2_0(x_{i+1/2}) = \frac{1}{2}\bar{v}_i + \frac{1}{2}\bar{v}_{i+1} \quad \text{and} \quad p^2_1(x_{i+1/2}) = -\frac{1}{2}\bar{v}_{i-1} + \frac{3}{2}\bar{v}_i,$$

and the corresponding higher-order interpolant is

$$p^3_1(x_{i+1/2}) = -\frac{1}{6}\bar{v}_{i-1} + \frac{5}{6}\bar{v}_i + \frac{1}{3}\bar{v}_{i+1}.$$

By applying (2.45) we get

$$-\frac{1}{6}\bar{v}_{i-1} + \frac{5}{6}\bar{v}_i + \frac{1}{3}\bar{v}_{i+1} = d_0\left(\frac{1}{2}\bar{v}_i + \frac{1}{2}\bar{v}_{i+1}\right) + d_1\left(-\frac{1}{2}\bar{v}_{i-1} + \frac{3}{2}\bar{v}_i\right),$$

and by comparing coefficients of the cell averages in the above equation we see that $d_0 = \frac{2}{3}$ and $d_1 = \frac{1}{3}$. Likewise, for $k = 3$ we have the lower-order interpolants

$$\begin{aligned} p^3_0(x_{i+1/2}) &= \frac{1}{3}\bar{v}_i + \frac{5}{6}\bar{v}_{i+1} - \frac{1}{6}\bar{v}_{i+2}, \\ p^3_1(x_{i+1/2}) &= -\frac{1}{6}\bar{v}_{i-1} + \frac{5}{6}\bar{v}_i + \frac{1}{3}\bar{v}_{i+1}, \end{aligned}$$

and

$$p_2^3(x_{i+1/2}) = \frac{1}{3}\bar{v}_{i-2} - \frac{7}{6}\bar{v}_{i-1} + \frac{11}{6}\bar{v}_i.$$

The corresponding higher-order interpolating polynomial, along with the corresponding sum from (2.45) is

$$\begin{aligned} p_2^5(x_{i+1/2}) &= \frac{1}{30}\bar{v}_{i-2} - \frac{13}{60}\bar{v}_{i-1} + \frac{47}{60}\bar{v}_i + \frac{9}{20}\bar{v}_{i+1} - \frac{1}{20}\bar{v}_{i+2} \\ &= d_0 p_0^3(x_{i+1/2}) + d_1 p_1^3(x_{i+1/2}) + d_2 p_2^3(x_{i+1/2}). \end{aligned}$$

By once again comparing the coefficients of the cell averages on either side of the above equation, it follows easily that $d_0 = \frac{3}{10}$, $d_1 = \frac{3}{5}$, and $d_2 = \frac{1}{10}$.

Assuming that ϵ and the β_l are already available, given the d_l we can then compute the α_l and then the ω_l values from (2.34) and (2.35). We have just seen how the d_l values lead to a linear combination of the $p_l^k(x_{i+1/2})$ values that provides an $O(\Delta x^{2k-1})$ approximation to $v(x_{i+1/2})$. We must now show how the ω_l values, defined in terms of the d_l values by (2.34) and (2.35), can also give a linear combination of the $p_l^k(x_{i+1/2})$ values that is an $O(\Delta x^{2k-1})$ approximation to $v(x_{i+1/2})$; i.e.,

$$\sum_{l=0}^{k-1} \omega_l p_l^k(x_{i+1/2}) = v(x_{i+1/2}) + O(\Delta x^{2k-1}). \tag{2.47}$$

Subtracting (2.45) from (2.47) gives

$$v(x_{i+1/2}) - p_{k-1}^{2k-1}(x_{i+1/2}) = \sum_{l=0}^{k-1} (\omega_l - d_l) p_l^k(x_{i+1/2}) + O(\Delta x^{2k-1}). \tag{2.48}$$

It is now sufficient to show that *each term* in the sum

$$\sum_{l=0}^{k-1} (\omega_l - d_l) p_l^k(x_{i+1/2}) \tag{2.49}$$

is an $O(\Delta x^{2k-1})$ approximation of $v(x_{i+1/2})$. Recall (2.32) and (2.42); it then follows that

$$\sum_{l=0}^{k-1} \omega_l - \sum_{l=0}^{k-1} d_l = \sum_{l=0}^{k-1} (\omega_l - d_l) = 0,$$

and therefore

$$\sum_{l=0}^{k-1} (\omega_l - d_l) v(x_{i+1/2}) = 0. \tag{2.50}$$

Thus (2.49) can be written as

$$\sum_{l=0}^{k-1} (\omega_l - d_l) \left(p_l^k(x_{i+1/2}) - v(x_{i+1/2}) \right). \tag{2.51}$$

From (2.44) we know that $p_l^k(x_{i+1/2}) - v(x_{i+1/2})$ is $O(\Delta x^k)$. Hence, in order to make each term in (2.49) $O(\Delta x^{2k-1})$, we require

$$\omega_l = d_l + O(\Delta x^{k-1}) \implies \omega_l - d_l = O(\Delta x^{k-1}), \tag{2.52}$$

so that each term is

$$O(\Delta x^{k-1}) O(\Delta x^k) = O(\Delta x^{2k-1}).$$

The ω_l coefficients are obtained from the d_l coefficients through (2.34) and (2.35). We next show how (2.52) follows. Because we wish to find weights ω_l such that (2.52) holds then, according to page 205 of [7], we need a smoothness term such that

$$\beta_l = D(1 + O(\Delta x^{k-1})), \tag{2.53}$$

where D is some non-zero value which is not dependent on the size of the stencil, k.

Let us assume that the smoothness term β_l is of the form (2.53) and that we have an ϵ value such that $\epsilon \ll \beta_l$. Under these assumptions

$$\alpha_l \approx \frac{d_l}{\beta_l^k} = \frac{d_l}{D^k(1 + O(\Delta x^{k-1}))^k} = \frac{d_l}{D^k}(1 + O(\Delta x^{k-1})). \tag{2.54}$$

and

$$\sum_{l=0}^{k-1} \alpha_l = \frac{1}{D^k} \sum_{l=0}^{k-1} (d_l + O(\Delta x^{k-1})) = \frac{1}{D^k} \left(\sum_{l=0}^{k-1} d_l \right) + O(\Delta x^{k-1}) = \frac{1}{D^k} + O(\Delta x^{k-1}). \tag{2.55}$$

Substituting (2.54) and (2.55) in (2.34) gives

$$\begin{aligned} \omega_l &= \frac{\frac{d_l}{D^k}(1 + O(\Delta x^{k-1}))}{\frac{1}{D^k} + O(\Delta x^{k-1})} \\ &= \frac{d_l(1 + O(\Delta x^{k-1}))}{(1 + O(\Delta x^{k-1}))} \\ &= d_l + O(\Delta x^{k-1}). \end{aligned} \tag{2.56}$$

Hence, having a smoothness term which satisfies (2.53) gives (2.52). It is shown in [7], page 207 that the smoothness term (2.36) satisfies (2.53).

2.5.3 WENO Scheme Experiments

We now examine a result obtained from the implementation of the discretization (2.4) using the WENO spatial discretization scheme with $k = 3$. The ODEs are evolved in time using the Matlab routine `ode45`, which uses adaptive time step control. The relative error tolerance and absolute error tolerance for `ode45` used in these experiments are both 1×10^{-10}, so that the time error will not contribute to the overall error. The solution to this problem has a smooth initial condition and develops no shocks, hence a SSP time integrator is not required.

We consider the advection equation (2.28) on the domain $(-1, 1)$, with the initial condition

$$u(x,0) = \begin{cases} 1, & \text{if } -\frac{1}{5} \le x \le \frac{1}{5}, \\ 0, & \text{otherwise,} \end{cases} \tag{2.57}$$

and periodic boundary conditions, $u(-1,t) = u(1,t)$. The problem is integrated from $t_0 = 0$ to $t_f = 1$. The results of this experiment are presented in Figure 2.1. We also note the absence of spurious oscillations in the numerical solution.

We next test for convergence of the WENO spatial discretization using the advection equation (2.28) on the domain $-1 \le x \le 1$, with the initial condition

$$u(x,0) = \sin(\pi x),$$

Test for Convergence		
N	L_1	rate
10	1.6192×10^{-2}	-
20	7.4356×10^{-4}	4.44
40	2.2577×10^{-5}	5.04
80	6.9766×10^{-7}	5.02
160	2.1782×10^{-8}	5.00

Table 2.6: Convergence of WENO Schemes: Advection equation, time integration from $t_0 = 0$ to $t_f = 1$ using the Dormand-Prince fifth-order Runge-Kutta method, $u(x, 0) = \sin(\pi x)$, $x \in (-1, 1)$, $\frac{\Delta t}{\Delta x} = 5 \times 10^{-2}$, periodic boundary conditions, and $k = 3$.

and periodic boundary conditions, $u(-1, t) = u(1, t)$. We integrate the problem from $t_0 = 0$ to $t_f = 1$, using $\Delta x = 0.04$. We will integrate in time using the refinement path $\frac{\Delta t}{\Delta x} = 5 \times 10^{-2}$ and the fifth-order Dormand-Prince Runge-Kutta method (Table 2.1). The solution to this problem has a smooth initial condition and develops no shocks, hence a SSP time integrator is not required.. For the Δt and Δx ratio chosen, this time stepping scheme is of sufficient accuracy that the total error is dominated by the spatial error. For $k = 3$, we expect a rate of convergence of approximately 5 in the 1-norm. The results are presented in Table 2.6, where we observe the expected rate of convergence.

2.5.4 Conclusions for WENO Schemes

In summary, WENO schemes allow us to deal with the discontinuities that are characteristic of hyperbolic PDEs, and through a careful selection of the coefficients in the method, they allow us to achieve higher order than ENO schemes.

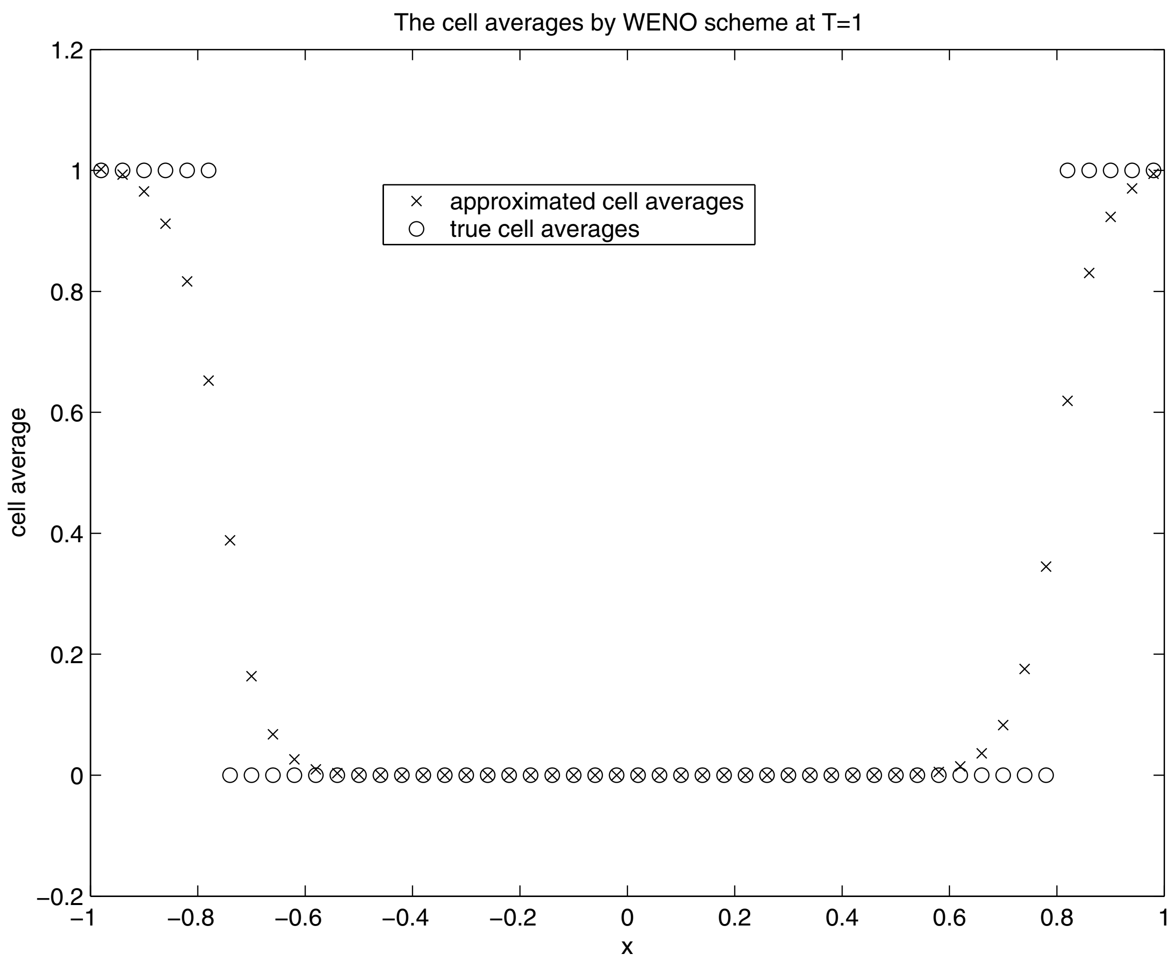

Figure 2.1: Advection equation, time integration to $t_f = 1$ using `ode45`, initial condition (2.57), $x \in (-1, 1)$, $\Delta x = 0.04$, periodic boundary conditions, and $k = 3$. The cell averages are plotted at the midpoints of the cells.

Chapter 3

Strong-Stability-Preserving Runge-Kutta Methods

In this chapter we consider some experiments involving SSP Runge-Kutta (SSPRK) methods. For the spatial discretization, we use a MUSCL [16] scheme which is a second-order TVD method in space. Also, we will produce an example in which a linearly stable method [9] produces a spurious oscillation, whereas an SSPRK method does not. This experiment motivates the use of SSPRK methods by illustrating a situation in which a SSPRK method is preferable to an arbitrary Runge-Kutta method. Specifically we reproduce the results on page 77 of [4].

Finally, the monotonicity numbers of some second-order time discretization methods are confirmed experimentally. A series of three experiments will be performed with the goal of testing to see for which Δt values each method becomes unstable. More extensive tests are done in [15], to which we refer the reader for further information on this subject.

3.1 A Description of Strong-Stability-Preserving Runge-Kutta Methods

Hyperbolic conservation laws such as (1.1) are interesting because discontinuities or shocks often occur in their solutions [9]. When we attempt to solve hyperbolic conservation laws numerically, the discontinuities present in the true solution may cause standard numerical integration methods to fail [9]. Errors resulting from the spatial discretization can propagate through the solution and may even cause the numerical computation to become unstable. The time integration method used in the numerical solution of the ODE system resulting from the spatial discretization of a hyperbolic conservation law may exhibit nonlinear instabilities in the form for spurious oscillations [8]. Rewriting (1.1) as $u_t(x,t) = -f(u(x,t))_x$, we define an approximation to $-f(u(x,t))_x$ in (1.1) in terms of the spatial discretization operator, $\mathbf{L}(\mathbf{U}(t))$, applied over the spatial mesh $\{x_i\}_{i=0}^{N}$, where $\mathbf{U}(t)$ is given by (1.8).

When the spatial discretization is applied to the PDE (1.1), we get the system of ODEs

$$\dot{\mathbf{U}}(t) = \mathbf{L}(\mathbf{U}(t)). \tag{3.1}$$

It is possible that the operator $\mathbf{L}$ could depend explicitly on t; however we suppress this dependence in order to simplify the notation. The rest of this chapter is based on the following fundamental assumption regarding the spatial discretization. *We assume that the spatial discretization used to generate (3.1) does not introduce spurious oscillations when the forward Euler method (which is of first-order accuracy; i.e., the error is $O(\Delta t)$ where Δt is the time step) is applied with a sufficiently small time step.* Higher-order time integration methods have been developed that allow (3.1) to be integrated in time without the introduction of spurious oscillations. Originally these methods were proven to be TVD [4] but more recently they have been shown to be SSP.

If we speak of a spatial discretization being TVD, it means that when the spatial discretization is applied, then the numerical solution obtained from the forward Euler method applied to the resulting ODEs is TVD for some suitably restricted step size Δt.

One way to think about Runge-Kutta methods is that they are able to take larger steps of higher order by taking linear combinations of smaller steps of lower order. Among the methods we will be considering, namely the Strong-Stability-Preserving Runge-Kutta (SSPRK) methods, the underlying idea is that these Runge-Kutta methods are SSP if the smaller steps upon which they are based are SSP as well [4]. SSPRK methods are composed of a convex combination of forward Euler steps [15]. *We assume that the spatial discretization is such that when we integrate the ODE system (3.1) using the forward Euler method with Δt sufficiently small, the resulting approximate solution satisfies the SSP condition (1.12).*

We will now explain what we mean by "Δt sufficiently small". We recall that the forward Euler method applied to (3.1) is

$$\mathbf{U}^{n+1} = \mathbf{U}^n + \Delta t \mathbf{L}(\mathbf{U}^n), \text{ where } \Delta t = t_{n+1} - t_n. \tag{3.2}$$

When this method is used to integrate the discretized hyperbolic PDE, there is a step size restriction on the method. Exceeding the step size restriction for the forward Euler method leads to an unstable numerical computation. This restriction on the step size is called the CFL condition [11][1]. Consider the hyperbolic PDE

$$u_t(x,t) + \alpha(x,t)u_x(x,t) = 0; \tag{3.3}$$

then the CFL condition for the forward Euler method applied to (3.3) is

$$|\alpha(x,t)|\Delta t_{FE} \leq \Delta x, \tag{3.4}$$

where Δx is the (assumed) uniform distance between discrete points in the spatial dimension [9]. We define Δt_{FE} as the maximum allowable step size for the forward Euler method, for the given spatial discretization and mesh spacing Δx, according to the CFL condition (3.4). As well, we define the *CFL number* for the forward Euler method to be the ratio

$$\frac{\Delta t_{FE}}{\Delta x}.$$

As mentioned earlier, assuming a time step satisfying the CFL condition, we need to assume that the spatial discretization is such that the resulting solution is SSP (1.11) when the forward Euler method is used for the time integration. We can then capitalize on this to construct SSPRK methods. When the forward Euler method is SSP, and we consider a numerical method that is made up of a convex combination of forward Euler steps (i.e., a SSPRK method) then that method is SSP as well [4].

For the ODE system (3.1) and for the nth time step $[t_n, t_{n+1}]$ we now consider the explicit Runge-Kutta method having s stages and written in the (non-standard) *Shu-Osher form* [4]

[1]Note that this restriction is distinct from the standard ODE linear stability step size.

$$\mathbf{U}^{(0)} = \mathbf{U}^n, \tag{3.5a}$$

$$\mathbf{U}^{(i)} = \sum_{k=0}^{i-1} \alpha_{ik}\mathbf{U}^{(k)} + \Delta t \beta_{ik}\mathbf{L}(\mathbf{U}^{(k)}), \quad i = 1, \ldots, s \tag{3.5b}$$

$$\mathbf{U}^{n+1} = \mathbf{U}^{(s)}. \tag{3.5c}$$

We assume we have a solution approximation $\mathbf{U}(t_n)$, where $\mathbf{U}(t_n) := \mathbf{U}^n$; then the Runge-Kutta method is used to obtain a solution approximation $\mathbf{U}^{n+1}$ at t_{n+1}. It is important to note that the Runge-Kutta method given by the coefficients α_{ik} and β_{ik} as in (3.5) can be uniquely expressed in terms of the standard Butcher array (see Table 3.1) through a mapping relating the entries α_{ik} and β_{ik} to the entries of the Butcher tableau (see [15], [4]). For example, the two-stage Runge-Kutta method in Shu-Osher form (3.5) gives the corresponding Butcher tableau seen in Table 3.2, where

$$c_2 = a_{21} = \beta_{10}, \quad b_1 = \beta_{20} + \alpha_{21}\beta_{10}, \quad \text{and} \quad b_2 = \beta_{21}. \tag{3.6}$$

$$\begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b}^T \end{array} := \begin{array}{c|ccc} c_1 & a_{11} & \cdots & a_{1s} \\ \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & \cdots & a_{ss} \\ \hline & b_1 & \cdots & b_s \end{array}$$

Table 3.1: Butcher tableau representation of the standard Runge-Kutta method.

$$\begin{array}{c|cc} 0 & 0 & 0 \\ c_2 & a_{21} & 0 \\ \hline & b_1 & b_2 \end{array}$$

Table 3.2: Butcher tableau representation of the two-stage, explicit Runge-Kutta method.

Recall that a Runge-Kutta method having the Butcher tableau shown in Table 3.1

has the general form

$$\mathbf{K}_i = \mathbf{L}\left(\mathbf{U}^n + \Delta t \sum_{j=1}^{s} a_{ij}\mathbf{K}_j\right), \quad i = 1, \dots, s, \tag{3.7a}$$

$$\mathbf{U}^{n+1} = \mathbf{U}^n + \Delta t \sum_{i=1}^{s} b_i \mathbf{K}_i. \tag{3.7b}$$

However, it is not true that a Runge-Kutta method expressed in terms of the standard Butcher tableau can be written *uniquely* in Shu-Osher form (3.5). Rather, a given Runge-Kutta method written in the standard form (3.7) represents an *equivalence class* of Shu-Osher methods [15]. In fact we can associate an infinite number of Shu-Osher methods with a given Runge-Kutta method. For a given Shu-Osher representation of a Runge-Kutta method, define M as follows,

$$M \equiv \min_{i,k} \frac{\alpha_{ik}}{\beta_{ik}}, \text{ for } \beta_{ik} \neq 0. \tag{3.8}$$

If we encounter a $\beta_{ik} = 0$ in (3.8) we ignore that particular ratio when calculating M. Of all possible Shu-Osher forms we can associate with a given Runge-Kutta method, only a subset will have coefficients which give the maximum value for M as defined in (3.8). We will call these *optimal Shu-Osher representations* of the given Runge-Kutta method. This maximum value of M associated with these optimal Shu-Osher representations is called the *monotonicity coefficient* for the Runge-Kutta method.

Theorem 2.3 from [15] states that if we integrate (3.1) with a Runge-Kutta method using a fixed time step, Δt, then the method will be SSP provided that the *monotonicity condition*,

$$\Delta t \leq M \Delta t_{FE}, \tag{3.9}$$

is satisfied, where Δt_{FE} is defined by the CFL condition for the forward Euler method for a given problem (as in (3.4) for example), and M is the monotonicity coefficient for the Runge-Kutta method.

An algorithm is given in [3] that calculates M for a Runge-Kutta method given **A** and **b** in Table 3.1. This algorithm however does not provide the coefficients for a corresponding optimal Shu-Osher representation of the Runge-Kutta method.

However, [3] also describes another approach that makes it possible to calculate an optimal Shu-Osher representation for the given Runge-Kutta method.

If we let Δt_{SO} be the maximum step size that satisfies (3.9) for a given optimal Shu-Osher method, then the *monotonicity number* for the Shu-Osher method is given as the ratio

$$\frac{\Delta t_{SO}}{\Delta x}.$$

Thus the monotonicity number for the Shu-Osher method, is the CFL number for the forward Euler method multiplied by M, the monotonicity coefficient of the Shu-Osher method. What is important to notice about (3.9) is that the step size restriction of the SSPRK method is dependent on the coefficients α_{ik} and β_{ik}. Some SSPRK methods have less restrictive monotonicity conditions (3.9) than others, and it is possible that a larger monotonicity coefficient could be more desirable since a larger monotonicity coefficient allows larger steps to be taken by the SSPRK method. However, a larger monotonicity coefficient may come at a cost; namely, more function evaluations. The paper [15] discusses a new group of SSPRK methods which are more efficient than the traditional SSPRK methods. These new SSPRK methods have a greater number of stages than order but are more efficient because the increased allowable stepsize of the method offsets the additional computational cost per step.

3.2 Examples of Runge-Kutta Methods

In this thesis we use four specific second-order Runge-Kutta methods. They are listed here for easy reference.

3.2.1 The SSP(2,2) Runge-Kutta Method

The first method is a two-stage second-order SSPRK method given in Shu-Osher form (3.5):

$$\mathbf{U}^{(0)} = \mathbf{U}^n \tag{3.10a}$$

$$\mathbf{U}^{(1)} = \mathbf{U}^{(0)} + \Delta t \mathbf{L}(\mathbf{U}^{(0)}), \tag{3.10b}$$

$$\mathbf{U}^{(2)} = \frac{1}{2}\mathbf{U}^{(0)} + \frac{1}{2}\mathbf{U}^{(1)} + \frac{1}{2}\Delta t \mathbf{L}(\mathbf{U}^{(1)}), \tag{3.10c}$$

$$\mathbf{U}^{n+1} = \mathbf{U}^{(2)}, \tag{3.10d}$$

with the corresponding Butcher tableau given in Table 3.3. We refer to this scheme as SSP(2,2). For this SSPRK method $\alpha_{10} = 1$, $\alpha_{20} = \frac{1}{2}$, $\alpha_{21} = \frac{1}{2}$, $\beta_{10} = 1$, $\beta_{20} = 0$, and $\beta_{21} = \frac{1}{2}$. This method has the Runge-Kutta representation seen in Table 3.2 with coefficients as defined in (3.6).

By (3.8) the SSPRK method (3.10) has $M = 1$. Referring to (3.9), this means that we are able to take stable time steps of the same size as the forward Euler method, but with improved accuracy since this is a second-order method.

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & 1/2 & 1/2 \end{array}$$

Table 3.3: Butcher tableau for SSP(2,2).

3.2.2 The Explicit Midpoint Method

The second method is the Explicit Midpoint method given in the standard Runge-Kutta form (3.7):

$$\mathbf{U}^{n+1} = \mathbf{U}^n + \Delta t \mathbf{K}_2, \text{ where } \mathbf{K}_1 = \mathbf{L}(\mathbf{U}^n) \text{ and } \mathbf{K}_2 = \mathbf{L}\left(\mathbf{U}^n + \frac{\Delta t}{2}\mathbf{K}_1\right), \tag{3.11}$$

and Butcher tableau given in Table 3.4. Since the Explicit Midpoint method is not given in Shu-Osher form we cannot find the monotonicity coefficient in a straightforward way (3.8). We can however compute the monotonicity coefficient of the Explicit Midpoint method using the algorithm given in [3], which shows that the Explicit Midpoint method has a monotonicity coefficient of $M = 0$. This also follows as a consequence of Theorem 2.4 in [3] since $b_1 = 0$.

We give an example of the Explicit Midpoint method written in Shu-Osher form to demonstrate that the method has a monotonicity coefficient $M = 0$. If the Explicit Midpoint method is written in the Shu-Osher form

$$\mathbf{U}^{(0)} = \mathbf{U}^{n} \tag{3.12a}$$

$$\mathbf{U}^{(1)} = \mathbf{U}^{(0)} + \Delta t \frac{1}{2}\mathbf{K}_2 = \mathbf{U}^{(0)} + \Delta t \frac{1}{2}\mathbf{L}(\mathbf{U}^{(0)}), \tag{3.12b}$$

$$\mathbf{U}^{(2)} = \mathbf{U}^{(0)} + \Delta t \mathbf{K}_2 = \mathbf{U}^{(0)} + \Delta t \mathbf{L}(\mathbf{U}^{(1)}), \tag{3.12c}$$

then $\alpha_{10} = 1$, $\beta_{10} = \frac{1}{2}$, $\alpha_{20} = 1$, $\alpha_{21} = 0$, $\beta_{20} = 0$, and $\beta_{21} = 1$. Therefore the monotonicity coefficient

$$M = \min \frac{\alpha_{ik}}{\beta_{ik}} = \frac{\alpha_{21}}{\beta_{21}} = \frac{0}{1} = 0.$$

Referring to (3.9), $M = 0$ implies that it is *impossible to guarantee* this method will suppress spurious oscillations, no matter how small we make Δt; i.e., the method is not guaranteed to be SSP for any step size.

0	0	0
1/2	1 /2	0
	0	1

Table 3.4: Butcher tableau for Explicit Midpoint method.

3.2.3 The Gottlieb and Shu Method

The Gottlieb and Shu (GS) method is the two-stage, second-order explicit Runge-Kutta method created in [4] to produce a spurious oscillation. Given in the Shu-Osher form (3.5), this method is written as

$$\mathbf{U}^{(0)} = \mathbf{U}^{n} \tag{3.13a}$$

$$\mathbf{U}^{(1)} = \mathbf{U}^{(0)} - 20\Delta t \mathbf{L}(\mathbf{U}^{(0)}), \tag{3.13b}$$

$$\mathbf{U}^{(2)} = \mathbf{U}^{(0)} + \frac{41}{40}\Delta t \mathbf{L}(\mathbf{U}^{(0)}) - \frac{1}{40}\Delta t \mathbf{L}(\mathbf{U}^{(1)}), \tag{3.13c}$$

$$\mathbf{U}^{n+1} = \mathbf{U}^{(2)}, \tag{3.13d}$$

with the Butcher tableau given in Table 3.5. We have $\alpha_{10} = 1$, $\alpha_{20} = 1$, $\alpha_{21} = 0$, $\beta_{10} = -20$, $\beta_{20} = \frac{41}{40}$, and $\beta_{21} = -\frac{41}{40}$. We will refer to this method as the GS method. We see the monotonicity coefficient given by (3.8) for (3.13) is $M = 0$. As for the Explicit Midpoint method, this means that the method cannot be guaranteed to suppress spurious oscillations, no matter how small we choose Δt; i.e., the method is not guaranteed to be SSP for any step size.

$$\begin{array}{c|cc} 0 & 0 & 0 \\ -20 & -20 & 0 \\ \hline & 41/40 & -1/40 \end{array}$$

Table 3.5: Butcher tableau for Gottlieb and Shu's method.

3.2.4 The SSP(3,2) Runge-Kutta Method

The three-stage, second-order method SSP(3,2) from [15] is written in the Shu-Osher form (3.5):

$$\mathbf{U}^{(0)} = \mathbf{U}^n \tag{3.14a}$$

$$\mathbf{U}^{(1)} = \mathbf{U}^{(0)} + \frac{1}{2}\Delta t \mathbf{L}(\mathbf{U}^{(0)}), \tag{3.14b}$$

$$\mathbf{U}^{(2)} = \mathbf{U}^{(1)} + \frac{1}{2}\Delta t \mathbf{L}(\mathbf{U}^{(1)}), \tag{3.14c}$$

$$\mathbf{U}^{(3)} = \frac{1}{3}\mathbf{U}^{(0)} + \frac{2}{3}\mathbf{U}^{(2)} + \frac{1}{3}\Delta t \mathbf{L}(\mathbf{U}^{(2)}), \tag{3.14d}$$

$$\mathbf{U}^{n+1} = \mathbf{U}^{(3)}. \tag{3.14e}$$

It has $\alpha_{10} = 1$, $\alpha_{20} = 0$, $\alpha_{21} = 1$, $\alpha_{30} = \frac{1}{3}$, $\alpha_{31} = 0$, $\alpha_{32} = \frac{2}{3}$, $\beta_{10} = \frac{1}{2}$, $\beta_{20} = 0$, $\beta_{21} = \frac{1}{2}$, $\beta_{30} = 0$, $\beta_{31} = 0$, and $\beta_{32} = \frac{1}{3}$. Applying the standard transformation it can be shown that this method has the Butcher tableau given in Table 3.6. From (3.8) we see that the monotonicity coefficient of SSP(3,2) is $M = 2$. Referring to (3.9) this implies that the method can take a step twice the size of the forward Euler method while still having the SSP property. As well, because the method is second order it generally delivers better accuracy than the forward Euler method, for a given step size.

$$
\begin{array}{c|ccc}
0 & 0 & 0 & 0 \\
1/2 & 1/2 & 0 & 0 \\
1 & 1/2 & 1/2 & 0 \\
\hline
 & 1/3 & 1/3 & 1/3
\end{array}
$$

Table 3.6: Butcher tableau for SSP(3,2).

3.3 Spurious Oscillations

We now consider an example of a spurious oscillation (similar to the example seen in [4]). We begin by considering the Burgers equation

$$u_t(x,t) + \left(\frac{1}{2}(u(x,t))^2\right)_x = 0, \tag{3.15}$$

with initial data:

$$u(x,0) = \begin{cases} 1, & \text{if } x \leq 0, \\ -0.5, & \text{if } x > 0, \end{cases} \tag{3.16}$$

and Dirichlet boundary conditions

$$
\begin{aligned}
u(-10,t) &= 1, \\
u(10,t) &= -0.5.
\end{aligned}
$$

The MUSCL discretization, which is a TVD spatial discretization, is used on a spatial grid of 200 subintervals, so $\Delta x = 1/10$.

A SSP time discretization method and a non-SSP time discretization method will be used to integrate the problem. For our SSP method we select SSP(2,2) as shown in (3.10) and for our non-SSP method we select the GS method (3.13). For both methods, 50 time steps of fixed size $\Delta t = 1/20$ are taken; this step size restriction satisfies the CFL condition (2.10). Since $\max |U_j^n| = 1$, for the SSP(2,2) method, we pick Δt as large as possible under the monotonicity condition, (3.9), where Δt_{FE} is restricted by (2.10). For the GS method which has $M = 0$, it is impossible to choose a non-zero Δt that satisfies (3.9). The presence of discontinuities (which our example clearly has) in the solution and other non-smooth behaviour can produce spurious oscillations in the numerical solution of the problem [4].

The first experiment uses the GS method; the result is presented in Figure 3.1, where we can see the presence of a spurious oscillation at the discontinuity. We then repeat the experiment using the SSP method SSP(2,2); the results are displayed in Figure 3.2, where we see that by using the SSPRK method we have eliminated the spurious oscillation.

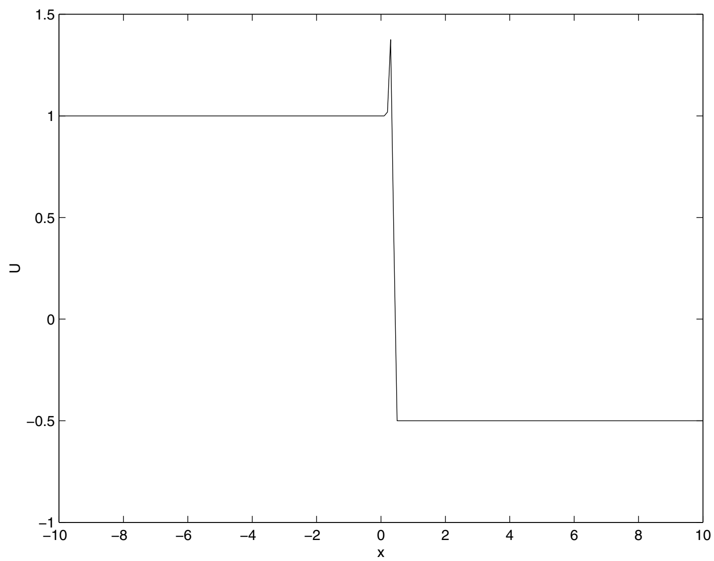

Figure 3.1: An integration by a non-SSP method, (3.13), resulting in a spurious oscillation when applied to (3.15), with initial condition (3.16), with $x \in (-10, 10)$, $\Delta x = 0.1$, taking 50 time steps with $\Delta t = 1/20$.

3.4 Stability of Selected Second-Order Methods

In this section, we consider the time step restriction associated with the SSP property for selected second-order methods. In particular we look at cases where the time step Δt is too large, and the monotonicity condition for the given method is violated, leading to growth in the error. In the following two experiments the range of Δt over which the error is observed is selected to include a time step for which we expect the monotonicity condition to be violated.

We compare the stability of three different second-order methods by comparing the error for some fixed Δx and various Δt. To do this, we consider the error in the numerical approximation versus the *effective monotonicity number*. We define the

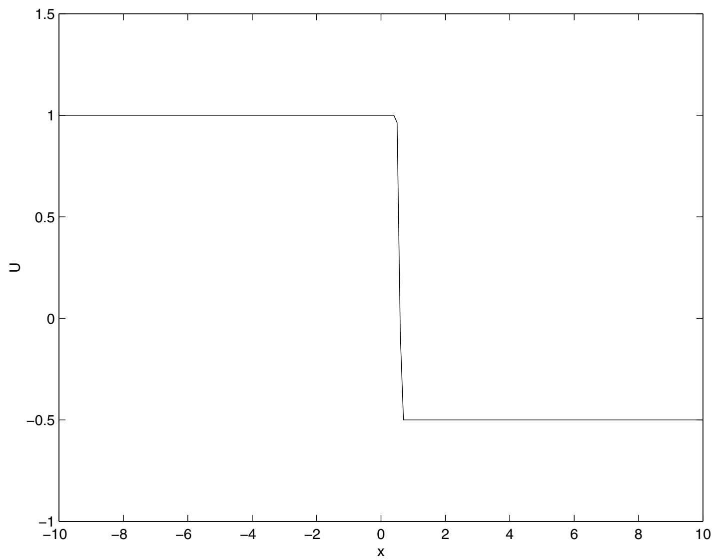

Figure 3.2: An integration by a SSP method, (3.10), resulting in no spurious oscillation when applied to (3.15), with initial condition (3.16), with $x \in (-10, 10)$, $\Delta x = 0.1$, taking 50 time steps with $\Delta t = 1/20$.

effective monotonicity number for a given Δt_{SO}, Δx, and Runge-Kutta method to be

$$\frac{\Delta t_{SO}}{\Delta x}\frac{1}{s}, \tag{3.17}$$

where s is the number of stages of the method, and Δt_{SO} is the maximum time step that satisfies (3.9).

From the experimental results, we can observe for which $\frac{\Delta t}{\Delta x}\frac{1}{s}$ the solution becomes unstable. Once again, we discretize the spatial domain using the second-order MUSCL scheme. We compare the SSP(2,2), Explicit Midpoint, and SSP(3,2) methods ((3.10), (3.11), and (3.14) respectively). We solve two test problems using each of the three methods and a sequence of fixed time step sizes, and we plot the error versus $\frac{\Delta t}{\Delta x}\frac{1}{s}$. The objective of these experiments is to make some observations about when the methods become unstable, and to see how these results compare to those found in [15].

3.4.1 Test 1: Advection equation, smooth initial conditions, periodic boundary conditions.

In this experiment we test the advection equation (2.28) with smooth initial data $u(x,0) = \sin(x)$ and periodic boundary conditions on $[0, 2\pi]$. Since (2.28) fits into the general form (3.3), we can observe from (3.4) that for this problem, $\Delta t_{FE} = \Delta x$. It is useful to note that the exact solution to this problem is $u(x,t) = \sin(x-t)$. Using each of the three methods we evolve the problem to time $t_f = 10$, using time steps of fixed size Δt. We consider a number of experiments, one for each value of Δt to be tested. In our testing we vary Δt from 0.01 and 0.2 in increments of 0.01. At t_f, for any trial, the 1-norm (2.8) of the absolute error,

$$||e||_1 := \frac{1}{N+1} \sum_{i=0}^{N} |U_i^n - u(x_i, t_n)|,$$

for that time step size is recorded, where N is the number of spatial subintervals. The grid that is used in this particular set of experiments contains 100 subintervals of fixed width; i.e., $\Delta x = 2\pi/100$.

The results for Test 1 are represented in Figure 3.3 and Figure 3.4. In Figure 3.3 we see that SSP(2,2) and the Explicit Midpoint method have fairly similar results, and both have errors that increase greatly as the $\frac{\Delta t}{\Delta x}\frac{1}{s}$ approaches $\frac{1}{2}$. This is the result we would expect for SSP(2,2) because the monotonicity number for SSP(2,2) in this problem is 1. Since SSP(2,2) has $s = 2$ the effective monotonicity number for which this two-stage method should exhibit drastically increasing error is $\frac{1}{2}$. We have no expectations for Explicit Midpoint in this case because it has a monotonicity coefficient of 0 and cannot be guaranteed to suppress oscillations for any time step size.

In Figure 3.4 we see that the three-stage, second-order method SSP(3,2) has a larger observed effective monotonicity number than its counterparts SSP(2,2) and the Explicit Midpoint method. When comparing these results with those found in Test 1 of [15], they are found to be in agreement; i.e., the errors are found to increase drastically as $\frac{\Delta t}{\Delta x}\frac{1}{s}$ approaches $\frac{2}{3}$. The results are as we would expect since SSP(3,2) has 3 stages and a monotonicity number of 2; then by (3.17) the effective monotonicity number for which we expect the resulting error to increase drastically is $\frac{2}{3}$; this is what we observe experimentally.

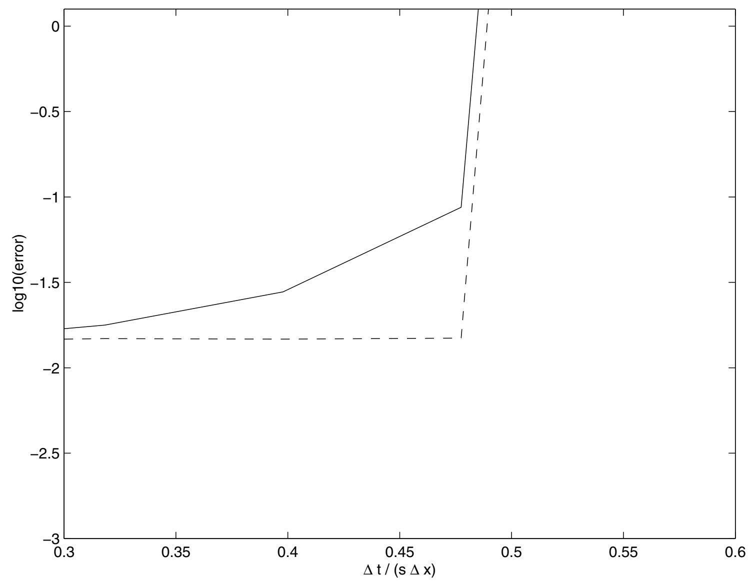

Figure 3.3: Test 1: log of error versus $\frac{\Delta t}{\Delta x}\frac{1}{s}$ for SSP(2,2) in solid line, results for Explicit midpoint in dashed line.

3.4.2 Test 2: Advection equation, discontinuous initial conditions, periodic boundary conditions.

In this experiment we consider the advection equation (2.28) with discontinuous initial data,

$$u(x,0) = \begin{cases} 1 & \text{for} \quad |x| < 1/3, \\ 0 & \text{for} \quad 1/3 < |x| \leq 1, \end{cases}$$

and periodic boundary conditions on $[-1,1]$. In this experiment, we consider a grid of 250 spatial subintervals, each subinterval having a constant width of $\Delta x = 1/125$. We evolve the problem to time $t_f = 2$, at which time the solution has moved exactly once around the spatial domain. The solution of the wave equation has a wave speed of 1, and the spatial domain in this particular problem has a width of 2. Hence, with periodic boundary conditions, a wave has travelled around the domain exactly once at time $\frac{\text{width of spatial domain}}{\text{wave speed}} = \frac{2}{1} = 2$. So, in this problem we have $u(x,2) = u(x,0)$

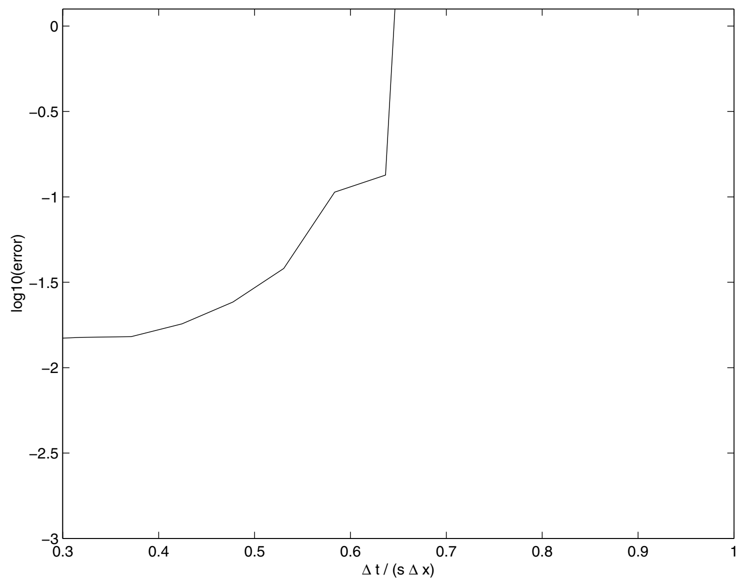

Figure 3.4: Test 1: log of error versus $\frac{\Delta t}{\Delta x}\frac{1}{s}$ for SSP(3,2).

and we can obtain a measure of the error by comparing the solution at $t = 2$ with the initial solution (at $t = 0$), at the spatial grid points.

We are interested in observing experimentally the effective monotonicity number, i.e., the minimum value of $\frac{\Delta t}{\Delta x}\frac{1}{s}$, for which a given method becomes unstable. As in Test 1, the SSP(2,2), Explicit Midpoint, and SSP(3,2) methods are tested for a range of fixed time step sizes. We measure the absolute error. For the Explicit Midpoint method and SSP(2,2), we begin with $\Delta t = 0.003$ and increase the step size in increments of 5×10^{-4} until $\Delta t = 0.013$. When we use SSP(3,2) in this experiment we begin with a step size of $\Delta t = 0.004$ and increase the step size by increments of 5×10^{-4} until $\Delta t = 0.019$.

It is observed in Figure 3.5 that the error increases quickly as $\frac{\Delta t}{\Delta x}\frac{1}{s}$ approaches $\frac{1}{2}$ for both SSP(2,2) and Explicit Midpoint method. This result is what we would expect for SSP(2,2). Because SSP(2,2) has a monotonicity coefficient of 1, by (3.17) we would expect the two-stage method SSP(2,2) to become unstable when the monotonicity number is greater than $\frac{1}{2}$. Likewise for SSP(3,2), a three-stage method with a monotonicity coefficient of 2, by (3.17) we would expect this method to become unstable at a monotonicity number of $\frac{2}{3}$. Again, we have no expectations for the results of the Explicit Midpoint method since the monotonicity coefficient is 0 and can never be guaranteed to suppress spurious oscillations no matter how small the time step size. When we run the experiment using SSP(3,2) we get the results we expect, represented in Figure 3.6. The error increases dramatically when $\frac{\Delta t}{\Delta x}\frac{1}{s}$ approaches $\frac{2}{3}$.

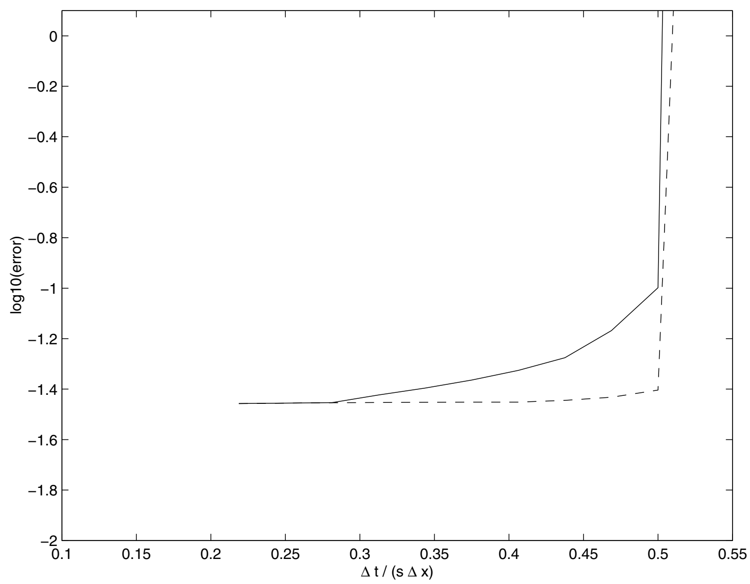

Figure 3.5: Test 2: log of error versus $\frac{\Delta t}{\Delta x}\frac{1}{s}$ for SSP(2,2) in the solid line, and results for Explicit Midpoint in the dashed line.

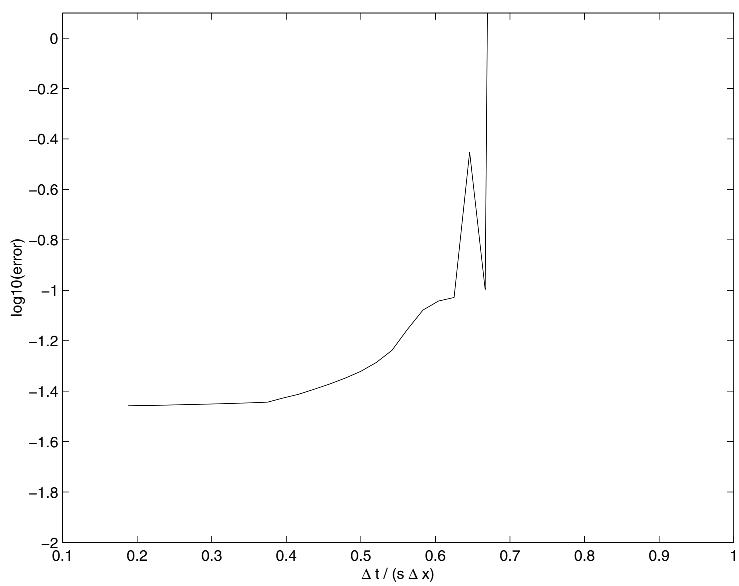

Figure 3.6: Test 2: log of error versus $\frac{\Delta t}{\Delta x}\frac{1}{s}$ for SSP(3,2).

Chapter 4

Some Stability Restrictions for Hyperbolic Conservation Laws

When we have discretized a hyperbolic PDE in the spatial dimension and have chosen a time integrator to integrate the resulting ODEs, we must take into consideration any step size restrictions associated with the method and problem. We must choose an appropriate step size in order to maintain stability in the computation of the numerical solution. In this chapter we examine some stability step size restrictions for the numerical solution of linear and nonlinear hyperbolic PDEs. We consider the step size restriction due to linear stability and the CFL condition for simple hyperbolic conservation laws and test the response of a standard initial-value solver to these conditions. We determine the theoretical and practical monotonicity conditions for a class of second order methods. Finally, we test the performance of an operator designed to maintain the SSP property in a time integrator when $\beta_{ik} < 0$.

4.1 Linear Stability Restrictions and the CFL Condition for Simple Hyperbolic Conservation Laws

Once a linear conservation law has been discretized in space using a simple finite difference scheme, the result is a linear system of initial-value ODEs. In this section we consider several kinds of step size restrictions that become relevant when one attempts to solve this ODE system numerically.

4.1.1 An Analysis of Stability for Simple Hyperbolic Conservation Laws

We first consider the advection equation; it has the form

$$u_t(x,t) + \alpha u_x(x,t) = 0, \quad x \in (a,b), \quad t > 0, \tag{4.1}$$

and we require that $\alpha > 0$, where α is a constant. Associated with this PDE is the boundary condition $u(a,t) = 0$ and the initial condition $u(x,0) = u_0(x)$. We apply a finite difference approximation to $u_x(x,t)$ in (4.1) over the spatial mesh $\{x_i\}_{i=0}^{N}$, and approximate $u(x,t)$ at the discrete points of the spatial mesh using (1.8).

The finite difference approximation to $u_x(x,t)$ that we employ is the first-order backward difference

$$u_x(x_i,t) \approx \frac{u(x_i,t) - u(x_{i-1},t)}{\Delta x_i}, \tag{4.2}$$

where $\Delta x_i = x_i - x_{i-1}$. For simplicity we assume a uniform spatial mesh $\Delta x_i = \Delta x$. By using the boundary condition $u(a,t) = 0$ and applying (4.2) with the given spatial mesh we get the ODE system

$$\dot{\mathbf{U}}(t) = \begin{bmatrix} -\frac{\alpha}{\Delta x} & 0 & \cdots & \cdots & 0 \\ \frac{\alpha}{\Delta x} & -\frac{\alpha}{\Delta x} & & & \vdots \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & \frac{\alpha}{\Delta x} & -\frac{\alpha}{\Delta x} \end{bmatrix} \mathbf{U}(t). \tag{4.3}$$

We next consider solving (4.3) using the simple forward Euler method as the time integrator. Standard theory for initial-value ODEs (see, e.g., [2]) indicates that the linear stability of the forward Euler method requires that

$$\Delta t \leq \frac{2}{-\lambda}, \tag{4.4}$$

where λ is the largest eigenvalue of the coefficient matrix in (4.3). Because the eigenvalues of the system (4.3) are $\lambda = -\frac{\alpha}{\Delta x}$, we see that the linear stability step size restriction for the forward Euler method is

$$\Delta t \leq 2\frac{\Delta x}{\alpha}.$$

We recall from (3.4) that the CFL condition for (4.1) is $\Delta t \leq \frac{\Delta x}{\alpha}$ and from this observe that for (4.1) the CFL condition is more restrictive than the linear stability restriction.

We next extend this analysis to the advection-reaction equation

$$u_t(x,t) + \alpha u_x(x,t) = -\beta u(x,t), \quad x \in (a,b), \quad t > 0, \tag{4.5}$$

where $\alpha > 0$, and both α and β are constants. The boundary condition associated with (4.5) is $u(a,t) = 0$, with initial condition $u(x,0) = u_0(x)$. If we discretize (4.5) in space using the backward difference formula (4.2), then we get the ODE system

$$\dot{\mathbf{U}}(t) = \begin{bmatrix} -\frac{\alpha}{\Delta x} - \beta & 0 & \cdots & \cdots & 0 \\ \frac{\alpha}{\Delta x} & -\frac{\alpha}{\Delta x} - \beta & & & \vdots \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & \frac{\alpha}{\Delta x} & -\frac{\alpha}{\Delta x} - \beta \end{bmatrix} \mathbf{U}(t). \tag{4.6}$$

Note that the eigenvalues of the ODE system (4.6) are

$$\frac{\alpha}{\Delta x} + \beta; \tag{4.7}$$

hence by (4.4) the step size restriction for linear stability using the forward Euler method is

$$\Delta t \leq \frac{2\Delta x}{\alpha + \beta \Delta x}. \tag{4.8}$$

Note from [9] that the CFL condition for (4.5) is

$$\Delta t \leq \frac{\Delta x}{\alpha}. \tag{4.9}$$

We wish to find α, β, such that the linear stability restriction is more severe than the CFL condition for the forward Euler method. From (4.9) and (4.8) we therefore require

$$\frac{2\Delta x}{\alpha + \beta \Delta x} < \frac{\Delta x}{\alpha} \implies 2\alpha < \alpha + \beta \Delta x \implies \alpha < \beta \Delta x.$$

Hence, the step size restriction for linear stability considerations for the forward Euler method is more severe than the step size restriction of the CFL condition when

$$\beta > \frac{\alpha}{\Delta x}. \tag{4.10}$$

4.1.2 Experimental Measurement of Stability Restrictions

We now verify this analysis experimentally by determining the step size restrictions due to linear stability constraints and the CFL condition, first using as the time integrator the forward Euler method, followed by the Dormand-Prince fifth-order Runge-Kutta method having the Butcher tableau seen in Table 2.1. For the following experiments we set $\alpha = 1$ and choose β to make either the linear stability condition or the CFL condition relevant.

When $\alpha = 1$ and $\Delta x = \frac{1}{100}$, (4.10) tells us that if $\beta > 100$, then linear stability is a more restrictive step size constraint than the CFL condition for the forward Euler method. In order to verify this we choose $\beta = 120$ for which we expect linear stability to be the dominant step size restriction. Then (4.8) gives $\Delta t \leq 9 \times 10^{-3}$ while (4.9) gives $\Delta t \leq 1 \times 10^{-2}$. We integrate the ODE system resulting from the discretization of (4.5) (using (4.2)) with $\alpha = 1$, $\beta = 120$, boundary condition $u(0, t) = 0$, and initial condition $u_0(x) = \sin(x)$, to time $t_f = 1$ using time steps, Δt, ranging in size from 6.0×10^{-3} to 9.1×10^{-3} with increments of 6.2×10^{-5} [1]. The 2-norm of the error for each of these step sizes is plotted as well as the maximum step size allowable under linear stability. The results are seen in Figure 4.1 and as we expect the integration becomes unstable as the step size reaches the linear stability restriction.

In the next experiment we choose $\beta = 50$; then by (4.10) we expect the CFL condition (4.9) to be the more severe step size restriction. For this choice of β (4.8) gives $\Delta t = 1.33 \times 10^{-2}$ while (4.9) gives $\Delta t \leq 1 \times 10^{-2}$. We test the CFL condition by integrating the ODE system resulting from the discretization of (4.5) (using (4.2)) with $\alpha = 1$, $\beta = 50$, boundary condition $u(0, t) = 0$, and initial condition $u_0(x) = \sin(x)$ to time $t_f = 1$ using time steps, Δt, ranging in size from 8.5×10^{-3} to 1.05×10^{-2} at increments of 4.0×10^{-5} [2]. The results are presented in Figure 4.2 and, as in the last experiment, the integration becomes unstable when the time step approaches the maximum time step allowed by the more severe step size restriction, which in this case is the CFL condition

[1] Chosen using 50 different time step sizes in the range.

[2] Chosen using 50 different time step sizes in the range.

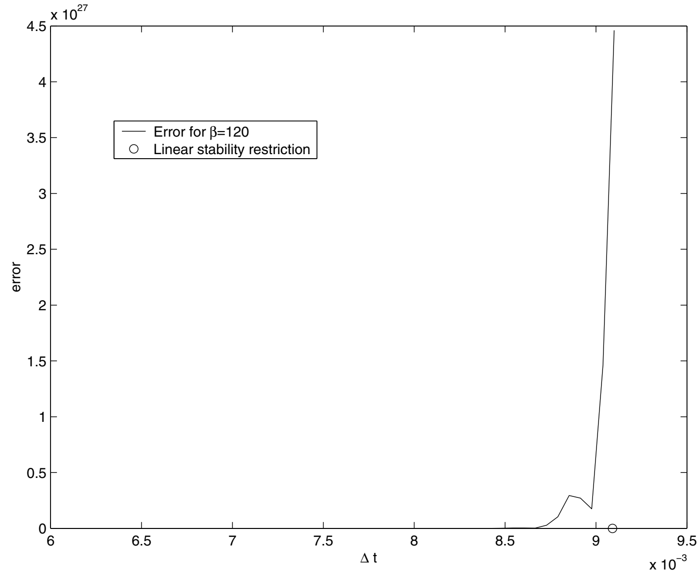

Figure 4.1: Experimentally determined linear stability step size restriction for the time integration of a discretization of the advection-reaction equation using the forward Euler method.

We now consider solving (4.6) using the Dormand-Prince method; the linear stability step size restriction for this method will be different than that for the forward Euler method (4.8). According to [12] the linear stability region for the Dormand-Prince method intersects the real axis at $\Delta t \lambda \approx -3.3$. Assuming λ is a real number, when

$$\Delta t \leq \frac{3.3}{\lambda} \tag{4.11}$$

we expect the integration using the Dormand-Prince Runge-Kutta method is linearly stable [2]. Because the eigenvalues for the ODE system are (4.7), the linear stability restriction for the Dormand-Prince Runge-Kutta method applied to the ODE system (4.6) given by (4.11) is

$$\Delta t \leq \frac{3.3 \Delta x}{\alpha + \beta \Delta x}. \tag{4.12}$$

We do not know how to obtain the CFL restriction for the Dormand-Prince method applied to (4.5). However we can experimentally determine the CFL condition for the advection-reaction equation (4.5) with $\alpha = 1$, $\beta = 150$, boundary

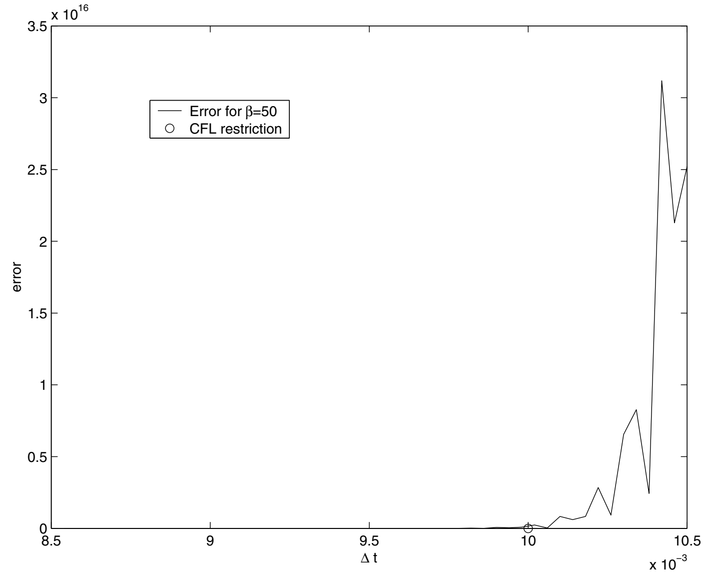

Figure 4.2: Experimentally determined CFL condition for the time integration of a discretization of the advection-reaction equation using the forward Euler method.

condition $u(0,t) = 0$, and initial condition $u_0(x) = \sin(x)$, on $x \in [0,1]$ using the Dormand-Prince method as the time integrator. We integrate the resultant system of ODEs from $t_0 = 0$ to $t_f = 1$ using step sizes, Δt, ranging from 1.9×10^{-2} to 2×10^{-2} at increments of 2×10^{-5}. The 2-norm of the error is plotted versus the step size in Figure 4.3. We observe from Figure 4.3 that the CFL restriction appears to be at $\Delta t = 0.02$. We know that the error growth we see here is due to a violation of the CFL restriction because the linear stability conditions for $\alpha = 1$, $\beta = 50$, and $\Delta x = \frac{1}{100}$ is 2.2×10^{-2} from (4.12).

We now find a choice for β such that the linear stability restriction for the Dormand-Prince is more severe than the CFL condition that we have experimentally determined. Since we see in Figure 4.3 that the CFL condition is violated in the interval $\Delta t \in (1.9 \times 10^{-2}, 2 \times 10^{-2})$, we can expect linear stability to be more restrictive than the CFL condition when

$$\frac{3.3\Delta x}{\alpha + \beta \Delta x} < 1.9 \times 10^{-2}. \tag{4.13}$$

Choosing $\beta = 150$ satisfies the above equation; we get $\Delta t \leq 1.32 \times 10^{-2}$. We check the linear stability restriction experimentally by integrating the ODEs resulting from the discretization of (4.5) (using (4.2)) with $\alpha = 1$, $\beta = 150$, boundary condition $u(0,t) = 0$, and initial condition $u_0(x) = \sin(x)$. In this test we integrate the ODE system from $t_0 = 0$ to $t_f = 1$ using time steps, Δt, ranging in size from 1.2×10^{-2} to 1.32×10^{-2} at increments of 2.0×10^{-5}. The results are presented in Figure 4.4 where we see agreement between the experimental results and (4.12).

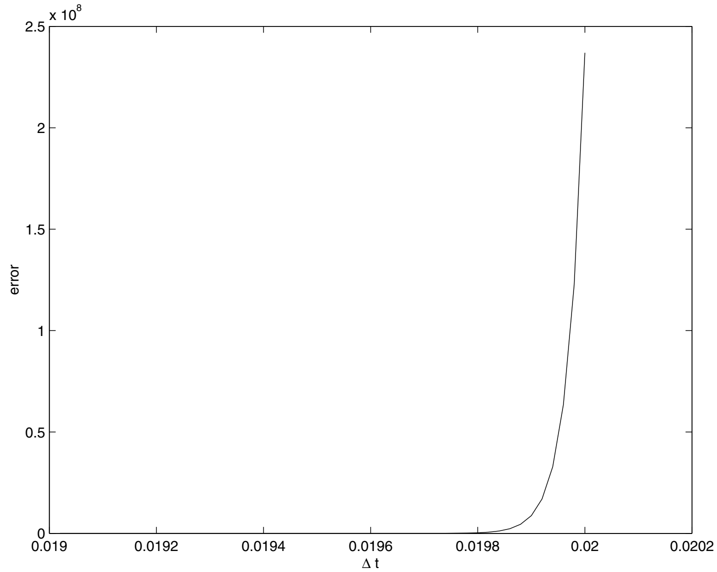

Figure 4.3: Experimentally determined CFL condition for a discretization of the advection-reaction equation using the Dormand-Prince scheme.

We now wish to determine if a standard ODE solver can detect instabilities due to a violation of the CFL condition, as well as instabilities due to a violation of the step size restriction for linear stability. For the ODE solver we use the Matlab function `ode45`, which uses the Dormand-Prince method. For this experiment we solve the ODEs resulting from a discretization of (4.5) (using (4.2)) when $\beta = 150$ and when $\beta = 50$, with $\alpha = 1$ and $\Delta x = \frac{1}{100}$, boundary condition $u(0,t) = 0$, and initial condition $u_0(x) = \sin(x)$ on $x \in [0,1]$, integrating from $t_0 = 0$ to $t_f = 0$. With $\beta = 150$ we have a problem for which the linear stability restriction will restrict the step size. For $\beta = 50$ we have a problem for which the CFL condition will restrict the step size.

By running `ode45` with $\beta = 150$ and $\beta = 50$, we find the average step size, $\overline{\Delta t}$, taken to be 1.278×10^{-2} and 1.968×10^{-2} respectively. From Figures 4.4 and 4.3, we

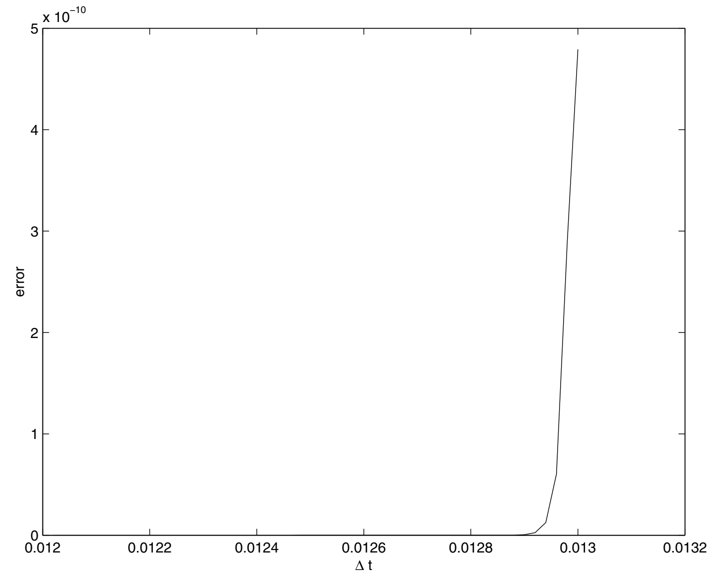

Figure 4.4: Experimentally determined linear stability restriction for a discretization of the advection-reaction equation using the Dormand-Prince scheme.

see that these step sizes correspond to the experimentally determined linear stability step size restriction and the CFL condition, respectively. We expect the ODE solver to detect any linear instability, but what is interesting to notice is that the ODE solver also detects instabilities in the numerical computation due to a violation of the CFL condition and adjusts the step size accordingly.

4.2 Parameterization of the General Two-Stage, Second-Order Explicit Runge-Kutta Method

In the previous section we considered some simple, linear hyperbolic PDEs and made some observations about time step size restrictions. In this section we go further by considering a nonlinear hyperbolic PDE and constructing a parameterization of the two-stage, second-order explicit Runge-Kutta method, allowing us to give a more detailed explanation of the step size restrictions resulting from a discretization of the PDE.

Let us consider the Burgers equation (3.15) with the initial condition

$$u(x,0) = \begin{cases} A, & \text{if } x < 0, \\ B, & \text{if } x \geq 0, \end{cases} \tag{4.14}$$

and having Dirichlet boundary conditions

$$u(a,t) = A, \quad u(b,t) = B,$$

where A and B are constants, and we assume $0 \in [a, b]$.

Recall that the two-stage, second-order explicit Runge-Kutta method can be respresented using the Butcher tableau seen in Table 4.1.

$$\begin{array}{c|cc} 0 & 0 & 0 \\ c_2 & a_{21} & 0 \\ \hline & b_1 & b_2 \end{array}$$

Table 4.1: Butcher tableau representation of the two-stage explicit Runge-Kutta method.

By applying the order conditions for second-order Runge-Kutta methods ([4]), we can parameterize the method as shown in Table 4.2. For the ODE system $\dot{\mathbf{U}}(t) = \mathbf{L}(\mathbf{U}(t))$, the method with the tableau of Table 4.2 is

$$\mathbf{K}_1 = \mathbf{L}(\mathbf{U}^n), \tag{4.15a}$$

$$\mathbf{K}_2 = \mathbf{L}(\mathbf{U}^n + \Delta t \gamma \mathbf{K}_1) \tag{4.15b}$$

$$\mathbf{U}^{n+1} = \mathbf{U}^n + \Delta t \left(\left(1 - \frac{1}{2\gamma} \right) \mathbf{K}_1 + \left(\frac{1}{2\gamma} \right) \mathbf{K}_2 \right). \tag{4.15c}$$

$$\begin{array}{c|cc} 0 & 0 & 0 \\ \gamma & \gamma & 0 \\ \hline & 1 - \frac{1}{2\gamma} & \frac{1}{2\gamma} \end{array}$$

Table 4.2: Parameterized Butcher tableau representation of the two-stage, second-order explicit Runge-Kutta method.

4.3 The Theoretical Monotonicity Condition

In Chapter 3, equation (3.8) we defined the monotonicity coefficient M. This value depends only on the corresponding SSPRK method and gives a lower bound for the step size restriction for a SSPRK method relative to that of the step size restriction for the forward Euler method. Recall that there is an algorithm in [3] that approximates the value of M iteratively. We wish to use this algorithm to determine the monotonicity coefficients of a number of two-stage, second-order explicit Runge-Kutta methods.

From the Butcher Tableau representation of the general two-stage, second-order explicit Runge-Kutta method seen in Tables 4.1 and 4.2 we have $a_{21} = \gamma$. The algorithm for calculating M specifies that the non-zero entries of $\mathbf{A}$ must be positive in order for the algorithm to be used. Hence, we may use the algorithm to calculate the monotonicity coefficient of (4.15) for $\gamma > 0$, but not for $\gamma < 0$. When we use the algorithm to compute M for the two-stage, second-order explicit Runge-Kutta method, for positive γ, we get the results shown in Figure 4.5.

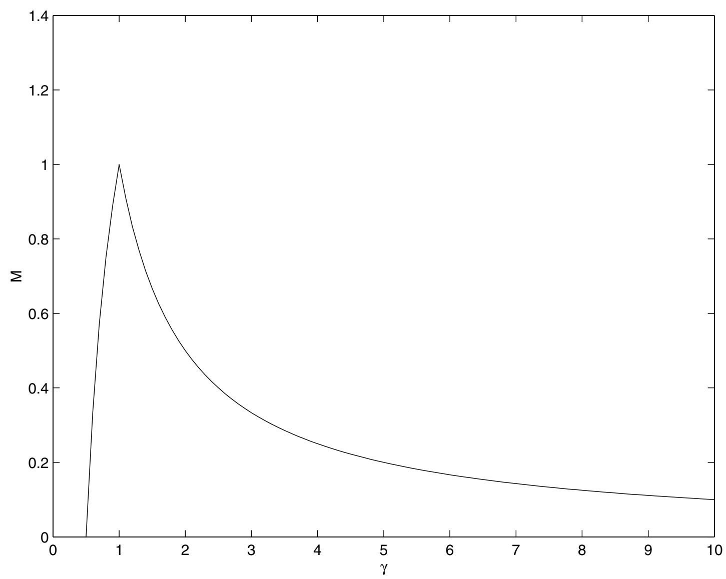

Figure 4.5: Using the algorithm of [3] to calculate M for the two-stage, second-order explicit Runge-Kutta method (4.15) when $\gamma > 0$.

Note that the Runge-Kutta method in standard form (4.15) can be expressed in Shu-Osher form (3.5) as

$$\mathbf{U}^{(1)} = \mathbf{U}^n + \Delta t \gamma \mathbf{L}(U^n), \tag{4.16a}$$

$$\mathbf{U}^{n+1} = (1-\alpha_{21})\mathbf{U}^n + \Delta t \left(\frac{2\gamma - 1 - 2\alpha_{21}\gamma^2}{2\gamma}\right) \mathbf{L}(\mathbf{U}^n) + \alpha_{21}\mathbf{U}^{(1)} + \Delta t \left(\frac{1}{2\gamma}\right) \mathbf{L}(\mathbf{U}^{(1)}), \tag{4.16b}$$

where we have introduced a free parameter $\alpha_{21} \in [0,1]$.

By comparing (4.16) with the general form for a two-stage Runge-Kutta method in Shu-Osher form, we see that $\alpha_{10} = 1$, $\alpha_{20} = 1 - \alpha_{21}$, $\beta_{10} = \gamma$, $\beta_{20} = 1 - \frac{1}{2\gamma} - \alpha_{21}\gamma$, and $\beta_{21} = \frac{1}{2\gamma}$. Using these values for the coefficients and the definition for M given in (3.8), we see that the monotonicity coefficient is given by

$$M = \max_{\alpha_{21}} \min \left\{ \frac{1}{\gamma}, \frac{1-\alpha_{21}}{1 - \frac{1}{2\gamma} - \alpha_{21}\gamma}, 2\gamma\alpha_{21} \right\}. \tag{4.17}$$

With respect to (4.17) we have the additional restrictions that $\gamma > 0$, as mentioned above, and $\alpha_{21} \in [0,1]$, so that $\alpha_{21} \geq 0$, and $\alpha_{20} \geq 0$[3].

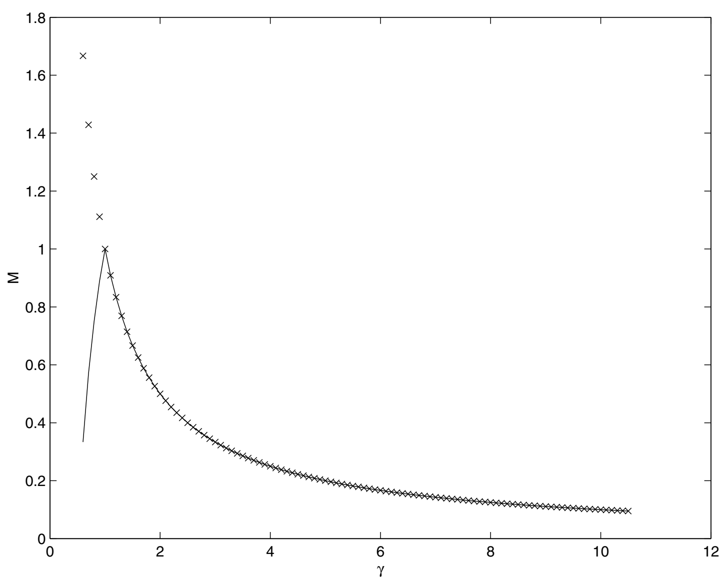

Figure 4.6: Comparing the results of the algorithm of [3] with $\frac{1}{\gamma}$.

We have seen a plot of M in Figure 4.5 as well as an expression for M (4.17). By plotting $\frac{1}{\gamma}$ for $\gamma > 0$ against the results from the algorithm of [3] (shown in Figure 4.5)

[3]Recall we must have $\alpha_{21} \geq 0$ and $\alpha_{20} \geq 0$ so that the SSPRK method is a convex combination of forward Euler steps.

we see in Figure 4.6 that for $\gamma \geq 1$, $M = \frac{1}{\gamma}$. That is, the first term in the minimum (4.17) is relevant. When $0 < \gamma < 1$ it is unknown what the correct expression for M is since the relationship is highly nonlinear.

If we wish to determine an expression for M for the general two-stage, second-order explicit Runge-Kutta method with respect to (4.16) when $\gamma < 0$, then according to [4] we can no longer use (3.8); we must use

$$M = \min_{i,k} \frac{\alpha_{ik}}{|\beta_{ik}|} = \max_{\alpha_{21}} \min \left\{ \frac{1}{|\gamma|}, \frac{1 - \alpha_{21}}{|1 - \frac{1}{2\gamma} - \alpha_{21}\gamma|}, |2\gamma|\alpha_{21} \right\}. \tag{4.18}$$

If we consider (4.18) for $-10 < \gamma < 0$ at increments of 1×10^{-2} and $0 \leq \alpha_{21} \leq 1$ at increments of 1×10^{-2}, then the result is a plot for M versus γ as seen in Figure 4.7. Because α_{21} is a free parameter, we select α_{21} that maximizes the minimum of the three arguments in (4.18) for a given γ in order to approximate M. A graphical comparison shows that

$$M = \frac{1}{|\gamma|}, \text{ for } \gamma < -2. \tag{4.19}$$

However when $-2 < \gamma < 0$ it is unknown what the expression is since the relationship between the second and third arguments of (4.18) is highly nonlinear.

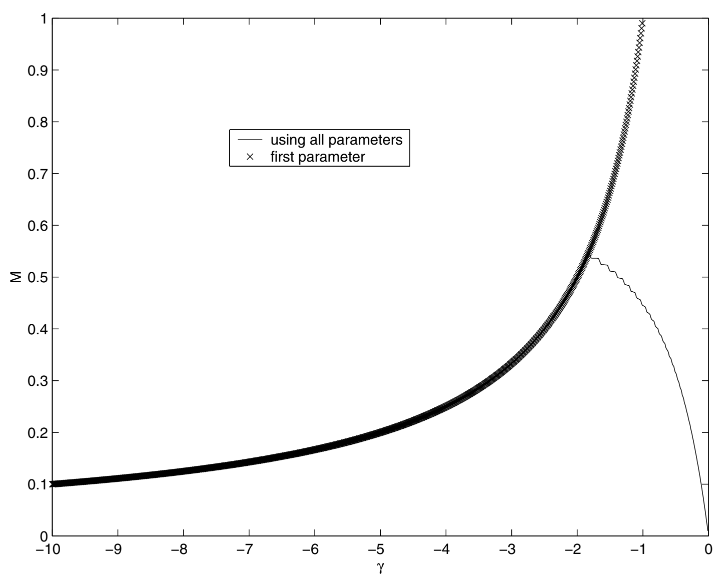

Figure 4.7: Determining the montonicity coefficient for $\gamma < 0$.

4.4 The Practical Monotonicity Coefficient

In the last section we looked at bounds on γ based on expressions which give theoretical bounds for the monotonicity coefficient M (which generates the SSP property for a sufficiently small time step). In this section we determine bounds on the parameter γ defining the two-stage, second-order explicit Runge-Kutta method for which we expect a spurious oscillation. We integrate the ODEs resulting from the MUSCL spatial discretization applied to Burgers' equation (3.15) using this Runge-Kutta method, with the initial condition (4.14). A spurious oscillation is an indicator of a breakdown in monotonicity, so we refer to these bounds as practical monotonicity conditions. We now consider a more detailed analysis for determining more precise monotonicity bounds on the Runge-Kutta method for one particular case. Note that in this analysis the initial discontinuity will fall between two arbitrary points in the grid.

If we consider the case when, in the initial condition (4.14), $A > |B|$, $A > 0$, $B < 0$, and $\gamma < 0$, then by a direct algebraic calculation using (4.15) we find that the two-stage, second-order Runge-Kutta method has the $\mathbf{K}_1$ value

$$\mathbf{K}_1 = L(\mathbf{U}^0) = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \frac{A^2-B^2}{2\Delta x} \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \tag{4.20}$$

where the single non-zero value is located at the grid point immediately to the right of the initial discontinuity. Let $\Omega = \frac{A^2-B^2}{2\Delta x}$. Next we find that $\mathbf{K}_2$ has value

$$\mathbf{K}_2 = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \frac{\frac{A^2}{2} - h(A, B+\Delta t\gamma\Omega)}{\Delta x} \\ \frac{h(A, B+\Delta t\gamma\Omega) - h(B+\Delta t\gamma\Omega, B}{\Delta x} \\ \frac{h(B+\Delta t\gamma\Omega, B) - \frac{B^2}{2}}{\Delta x} \\ 0 \\ \vdots \\ 0 \end{pmatrix}. \tag{4.21}$$

where h is the Godunov flux (2.7). The index of the middle value of the three non-zero values in the vector $\mathbf{K}_2$ (4.21) corresponds to the index of the lone non-zero value in the vector $\mathbf{K}_1$ (4.20).

We are able to observe experimentally that the $s$purious oscillation is located at the grid point located to the immediate left of the initial discontinuity. Thus we infer that the spurious oscillation occurs when the value $\frac{\frac{A^2}{2} - h(A, B+\Delta t\gamma\Omega)}{\Delta x}$ in (4.21) is nonzero (and we determine experimentally for these particular conditions that this contribution will be negative). This is because the numerical solution at the next point in time will be a linear combination of $\mathbf{K}_1$ and $\mathbf{K}_2$, and since only $\mathbf{K}_2$ may contribute to an increase in the solution at this grid point, the term in the vector $\mathbf{K}_2$ corresponding to this grid point must be the source of the oscillation.

Next we investigate to see what selection of parameters (γ, A, B) makes

$$\frac{\frac{A^2}{2} - h(A, B + \Delta t\gamma\Omega)}{\Delta x} < 0. \tag{4.22}$$

By evaluating the function h, the above is true when

$$\frac{\left(B + \Delta t\gamma \frac{A^2 - B^2}{2\Delta x}\right)^2}{2} > \frac{A^2}{2}. \tag{4.23}$$

Simplifying this expression yields the polynomial inequality

$$\frac{1}{2}\left(\frac{\Delta t}{\Delta x}\right)^2 (A^2 - B^2)\gamma^2 + B\left(\frac{\Delta t}{\Delta x}\right)\gamma - 1 > 0. \tag{4.24}$$

Recall that we are concerned with negative γ, and by simply finding the appropriate root of the polynomial in the inequality (4.24), we find that (4.22) holds when

$$\gamma < \frac{-2\Delta x}{(A - B)\Delta t}. \tag{4.25}$$

We now have an expression for $\gamma < 0$, in terms of the initial conditions (4.14), Δt, Δx, when $A > |B|$, $A > 0$, $B < 0$, for which a spurious oscillation will be present after one step using the general two-stage, second-order Runge-Kutta method (4.15), with a MUSCL spatial dicretization. Likewise, for $\gamma > 0$ we find a spurious oscillation occuring when

$$\gamma > \frac{4(\left(\frac{\Delta x}{\Delta t}\right)^2 - \left(\frac{\Delta x}{\Delta t}\right) B)}{A^2 - B^2}.$$

We now consider the remaining three combinations of signs for A and B for which $A > B$. This will give us six bounds on γ. If $A > 0$ and $B > 0$ then a spurious oscillation can be expected when

$$\gamma > \frac{4(\left(\frac{\Delta x}{\Delta t}\right)^2 - \left(\frac{\Delta x}{\Delta t}\right) B)}{A^2 - B^2} \quad \text{or} \quad \gamma < \frac{-2\Delta x}{(A - B)\Delta t}.$$

The next case is when $A < |B|$, $A > 0$, $B < 0$; in this case a spurious oscillation will be produced when either

$$\gamma > \frac{-4(\left(\frac{\Delta x}{\Delta t}\right)^2 + \left(\frac{\Delta x}{\Delta t}\right) A)}{A^2 - B^2} \quad \text{or} \quad \gamma < \frac{2\Delta x}{(A + B)\Delta t}.$$

The last case is when the initial conditions are $A < 0$, $A > B$. In this case we observe spurious oscillations in the numerical solution when

$$\gamma > \frac{-4(\left(\frac{\Delta x}{\Delta t}\right)^2 + \left(\frac{\Delta x}{\Delta t}\right) A)}{A^2 - B^2} \quad \text{or} \quad \gamma < \frac{-2\Delta x}{(A - B)\Delta t}.$$

4.5 The $\tilde{\mathbf{L}}$ Operator

When all the coefficients, α_{ik}, β_{ik}, are positive then the Runge-Kutta method written in Shu-Osher form (3.5) is a convex combination of forward Euler steps. However, when $\beta_{ik} < 0$ we must take a different approach than simply using the spatial operator $\mathbf{L}(\mathbf{U}^n)$. In order to maintain the SSP property we must correct the upwinding of the method in the time dimension. See [4] for further details.

When a Runge-Kutta method is in Shu-Osher form and there exists a $\beta_{ik} < 0$, we must use another operator, which we will call $\tilde{\mathbf{L}}$, that approximates the same spatial derivative as $\mathbf{L}$ but is SSP for the forward Euler method backward in time [4]; i.e., we have

$$\mathbf{U}^{n+1} = \mathbf{U}^n - \Delta t \tilde{\mathbf{L}}(\mathbf{U}^n).$$

This can be done by solving

$$u_t(x,t) - f(u(x,t))_x = 0$$

rather than (1.1). Applying the $\tilde{\mathbf{L}}$ spatial operator is effectively done in the MUSCL discretization by replacing (2.7) with

$$h(\tilde{u}_1, \tilde{u}_2) = \begin{cases} \min_{\tilde{u}_1 \le U \le \tilde{u}_2} -f(U), & \text{if } \tilde{u}_1 \le \tilde{u}_2, \\ \max_{\tilde{u}_1 \le U \le \tilde{u}_2} -f(U), & \text{if } \tilde{u}_1 > \tilde{u}_2, \end{cases} \tag{4.26}$$

whenever the corresponding β_{ik} in the Shu-Osher form is negative. If we do not do this, then the SSPRK method (with a negative β_{ik} coefficient) is not guaranteed to suppress spurious oscillations. We now demonstrate an example of how the use of $\tilde{\mathbf{L}}$ can suppress a spurious oscillation.

Recall the experimental results shown in Figure 3.1. In that experiment we used the GS method (3.13) which had negative β_{ik} coefficients in Shu-Osher form. From our discussion above we know that we must use $\tilde{\mathbf{L}}$ for the spatial operator where $\beta_{ik} < 0$. Also, since γ for this method is -20 (recall, $\gamma = a_{21}$, and from Table 3.5 we note that a_{21} for the GS method is -20), by (4.19) and (3.9) we have the step size restriction

$$\Delta t \leq \frac{1}{20} \Delta t_{FE}. \tag{4.27}$$

In the experiment demonstrated in Figure 3.1, we had solved Burgers' equation (3.15) on $x \in [-10, 10]$ with $\Delta x = 0.1$, initial conditions (3.16), and Dirichlet boundary conditions

$$u(-10, t) = 1 \text{ and } u(10, t) = -0.5.$$

In that experiment we took 50 time steps of $\Delta t = \frac{1}{20}$, to time $t_f = 2.5$. In this demonstration we integrate to the same time using the same conditions; however, when we apply $\tilde{\mathbf{L}}$, and an appropriate time step under (4.27), the spurious oscillation noted in Figure 3.1 disappears as shown in Figure 4.8.

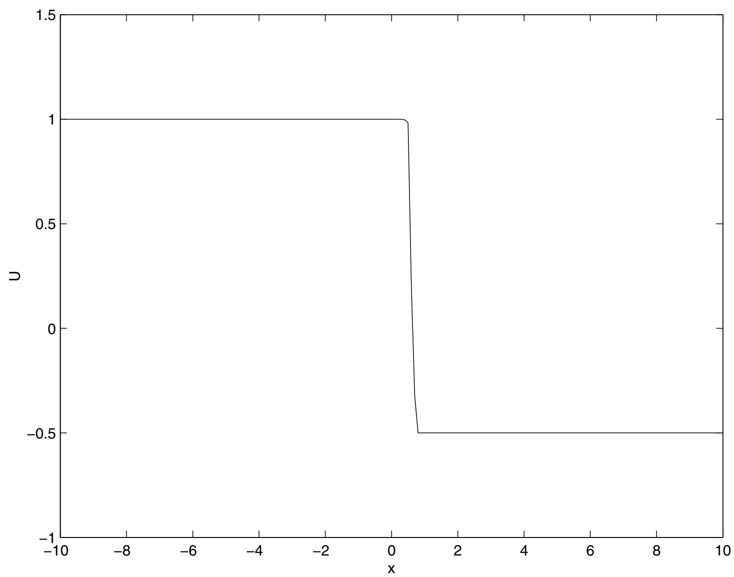

Figure 4.8: In this figure we note the absence of a spurious oscillation when $\tilde{\mathbf{L}}$ during the time step.

Up to this point we have discussed three types of step size restrictions, the CFL condition, the theoretical monotonicity condition, and the practical monotonicity condition. In this next two experiments we wish to compare the stability of the computation using simply $\mathbf{L}$ for the spatial operator rather than both $\mathbf{L}$ and $\tilde{\mathbf{L}}$. In the next two experiments we use slightly different experimental conditions from the previous experiment. We use the initial condition

$$u(x,0) = \begin{cases} 1.5, & \text{if } x \leq 0, \\ 0.5, & \text{if } x > 0, \end{cases} \tag{4.28}$$

Dirichlet boundary conditions

$$u(-10,t) = 1.5, u(10,t) = 0.5,$$

and we measure the relative total variation,

$$\frac{|TV(\mathbf{U}^n) - TV(\mathbf{u}^n)|}{TV(\mathbf{u}^n)},$$

where

$$\mathbf{u}^n = \begin{bmatrix} u(x_0,t_n) \\ u(x_1,t_n) \\ \vdots \\ u(x_N,t_n) \end{bmatrix},$$

for various values of Δt at $t_f = 1$.

Recall that the theoretical monotonicity condition implies (3.9). The practical monotonicity condition is calculated using $\gamma = -20$, $A = 1.5$, and $B = 0.5$; we find the practical monotonicity condition in this case to be $\Delta t \leq 0.01$. The CFL condition is calculated by first using the equation for shock speed for Burgers' equation from [9] giving us

$$s_{shock} = \frac{1}{2}(A+B) = 1.$$

Thus the shock moves along the line $\Delta x = s\Delta t$ and in order for this line to fall within the domain of dependence of the numerical scheme we require $\Delta t \leq 0.1$, which we will call our CFL condition.

In the next experiment we integrate using only the operator $\mathbf{L}$, with step sizes, Δt, ranging between 2.0×10^{-3} and 1.002×10^{-1}, at increments of 5×10^{-3}. In Figure 4.9 we see that by using only the operator $\mathbf{L}$ the relative total variation increases dramatically before reaching the CFL condition, beginning just beyond the practical monotonicity condition.

Finally, we run the experiment using both $\mathbf{L}$ and $\tilde{\mathbf{L}}$ with step sizes ranging from 2.0×10^{-3} and 1.002×10^{-1}, at increments of 5×10^{-3}. We observe in Figure 4.10 that, when using the additional operator, there is no increase in the relative total variation until the time step reaches a point between the practical monotonicity condition and the CFL condition (the discontinuity represents a drastic increase in the error). In this example there is an additional cost of calculating $\tilde{\mathbf{L}}$, however, we are able to safely take a step that is nearly twice the size of the one taken when only using $\mathbf{L}$.

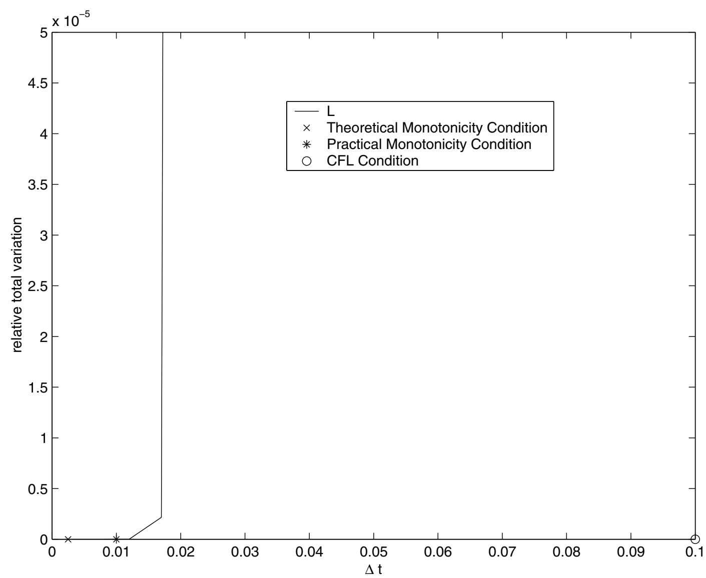

Figure 4.9: Relative total variation when applying only $\mathbf{L}$ and not $\tilde{\mathbf{L}}$, using the GS method on Burgers' equation and a discontinuous initial condition.

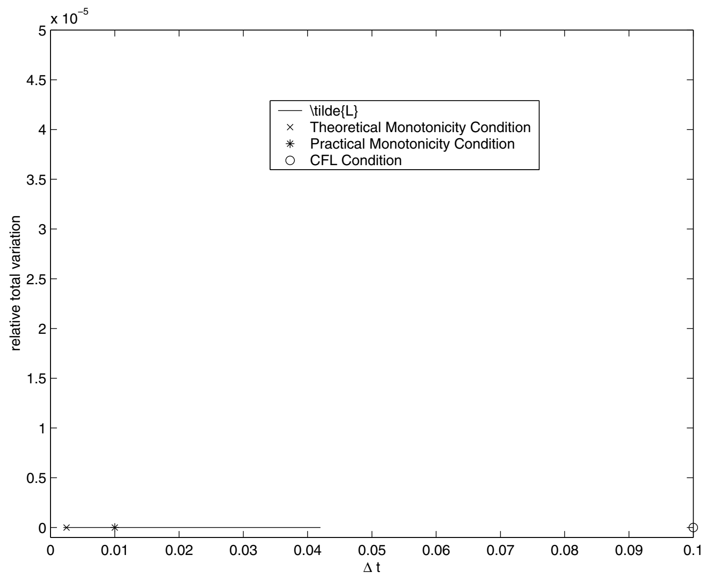

Figure 4.10: Relative total variation when applying both $\mathbf{L}$ and $\tilde{\mathbf{L}}$, using the GS method on Burgers' equation and a discontinuous initial condition.

Chapter 5

Conclusions and Future Work

In this thesis we have discussed some of the issues surrounding the numerical solution of hyperbolic conservation laws having the general form (1.1), with either periodic (1.3) or Dirichlet (1.4) boundary conditions, and initial condition (1.2). We then considered some numerical methods designed to address these issues, including spatial discretizations and time integrators. We have demonstrated the benefits of ENO and WENO spatial discretizations, and SSPRK methods, in order to more accurately compute the numerical solution of a hyperbolic conservation law.

We reformulated (1.1) using the finite volume formulation (2.3), in which we wrote (1.1) in terms of the cell averages. Once this was done we used either the ENO or WENO spatial discretizations to provide solution approximations at the cell boundaries (i.e., mesh points) of the grid. These solution approximations were used to construct the ODEs making up the finite volume formulation. These ODEs were then integrated in time using SSPRK methods. Assuming that the spatial discretization had not introduced spurious oscillations, SSPRK methods ensured that no spurious oscillations were introduced during the time integration.

In Chapter 2, we were able to confirm that the MUSCL spatial discretization was second order (Table 2.2). As well, we were able to confirm that ENO schemes are $O(\Delta x^k)$ (Table 2.3, Table 2.4, and Table 2.5), and that WENO schemes were $O(\Delta x^{2k-1})$ (Table 2.6), where Δx was the uniform grid spacing and k was the number of cells in the stencil being used.

In Chapter 3 we implemented selected SSPRK methods and confirmed their effective CFL numbers. We were also able to show an example in which using a non-SSP method results in a spurious oscillation in the numerical solution (Figure 3.1), whereas using an SSPRK method does not (Figure 3.2), demonstrating the need for such methods.

In Chapter 4 we were able to demonstrate that an initial-value ODE solver was able to detect instabilities in the numerical solution due to a violation of the CFL condition. As well we were able to determine the theoretical monotonicity condition for the general two-stage, second-order, explicit Runge-Kutta method. Practical monotonicity bounds were also determined for the general two-stage, second-order, explicit Runge-Kutta method under certain problem conditions. From this we were able to show that in the theoretical monotonicity condition is not a strict condition in practice and that it is often possible to take a larger step size. Also examined in this Chapter was an operator, $\tilde{\mathbf{L}}$, for when $\beta_{ik} < 0$ in the Shu-Osher representation, used to make sure that a SSPRK method retains its SSP property.

In the future we wish to incorporate these new methods into a general software package for the numerical solution of hyperbolic partial differential equations. The methods discussed in this thesis will be implemented using the initial-value software package `odeToJava`. As well a comparison will be made between the performance of SSPRK methods and implicit-explicit (IMEX) Runge-Kutta methods in the numerical solution of hyperbolic conservation laws.

Bibliography

[1] William F. Ames. *Numerical Methods for Partial Differential Equations.* Academic Press, New York, 1977.

[2] Uri M. Ascher and Linda R. Petzold. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations.* Society for Industrial & Applied Mathematics, 1998.

[3] Luca Ferracina and Marc N. Spijker. Stepsize restrictions for the total-variation-diminishing property in general Runge-Kutta methods. Technical Report Report No. 2002-21, Mathematical Institute, Leiden Univeriversity, Netherlands, 2002.

[4] Sigal Gottlieb and Chi-Wang Shu. Total variation diminishing Runge-Kutta schemes. *Math. Comp.*, 67(221):73–85, 1998.

[5] Ami Harten. High resolution schemes for hyperbolic conservation laws. *J. Comput. Phys.*, 135(2):259–278, 1997. With an introduction by Peter Lax, Commemoration of the 30th anniversary {of J. Comput. Phys.}.

[6] Guang-Shan Jiang. *Performance Analysis of Unconditionally Synchronizing Distributed Computer Programs Using The Geometric Concurrency Model.* PhD thesis, Brown University, August 1986. TR-1696.

[7] Guang-Shan Jiang and Chi-Wang Shu. Efficient implementation of weighted ENO schemes. *J. Comput. Phys.*, 126(1):202–228, 1996.

[8] Culbert B. Laney. *Computational Gasdynamics.* Cambridge University Press, Cambridge, U.K., 1998.

[9] Randall J. LeVeque. *Finite volume methods for hyperbolic problems.* Cambridge Texts in applied mathematics. Cambridge University Press, 2002.

[10] Xu-Dong Liu, Stanley Osher, and Tony Chan. Weighted essentially non-oscillatory schemes. *J. Comput. Phys.*, 115(1):200–212, 1994.

[11] K. William Morton and David F. Mayers. *Numerical solution of partial differential equations.* Cambridge University Press, 1993.

[12] S. Nassos Papakostas and Grigoris Papageorgiou. A family of fifth-order Runge-Kutta pairs. *Math. Comp.*, 65(215):1165–1181, 1996.

[13] William E. Schiesser. *The Numerical Method of Lines: Integration of Partial Differential Equations.* Academic Press, New York, 1991.

[14] Chi-Wang Shu. Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws. Technical Report NASA CR-97-206253 ICASE Report No. 97-65, Institute for Computer Applications in Science and Engineering, November 1997.

[15] Raymond J. Spiteri and Steven J. Ruuth. A new class of optimal high-order strong-stability-preserving time discretization methods. *SIAM J. Numer. Anal.*, 40(2):469–491 (electronic), 2002.

[16] Bram van Leer. Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method [J. Comput. Phys. **32** (1979), no. 1, 101–136]. *J. Comput. Phys.*, 135(2):227–248, 1997. With an introduction by Ch. Hirsch, Commemoration of the 30th anniversary {of J. Comput. Phys.}.