High Order Strong Stability Preserving Time Integrators and Numerical Wave Propagation Methods for Hyperbolic PDEs

David I. Ketcheson

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

Doctor of Philosophy

University of Washington

2009

Program Authorized to Offer Degree: Applied Mathematics

University of Washington
Graduate School

This is to certify that I have examined this copy of a doctoral dissertation by

David I. Ketcheson

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Chair of the Supervisory Committee:

_____

Randall J. LeVeque

Reading Committee:

_____

Randall J. LeVeque

_____

Bernard Deconinck

_____

Kenneth Bube

Date: _____

In presenting this dissertation in partial fulfillment of the requirements for the doctoral degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of this dissertation is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for copying or reproduction of this dissertation may be referred to Proquest Information and Learning, 300 North Zeeb Road, Ann Arbor, MI 48106-1346, 1-800-521-0600, to whom the author has granted "the right to reproduce and sell (a) copies of the manuscript in microform and/or (b) printed copies of the manuscript made from microform."

Signature_____

Date_____

. .

University of Washington

Abstract

High Order Strong Stability Preserving Time Integrators and Numerical Wave Propagation Methods for Hyperbolic PDEs

David I. Ketcheson

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

Hyperbolic PDEs describe the great variety of physical phenomena governed by wave behavior. Numerical methods are necessary to provide approximate solutions to real wave propagation problems. High order accurate methods are often essential to achieve accurate solutions on computationally feasible grids. However, maintaining numerical stability and satisfying physical constraints becomes increasingly difficult as higher order methods are employed. In this thesis, high order numerical tools for hyperbolic PDEs are developed in a method of lines framework. Optimal high order accurate strong stability preserving (SSP) time integrators, for both linear and nonlinear systems, are developed. Improved SSP methods are found as a result of rewriting the relevant optimization problems in a form more amenable to efficient solution. A new, very general class of low-storage Runge-Kutta methods is proposed (based on the form of some optimal SSP methods) and shown to include methods with properties that cannot be achieved by existing classes of low-storage methods. A high order accurate semi-discrete wave propagation method is developed in one and two dimensions using wave propagation Riemann solvers and high order weighted essentially non-oscillatory (WENO) reconstruction. The space and time discretizations are combined to achieve a high order accurate method for general hyperbolic PDEs. This method is applied to model solitary waves in a nonlinear, non-dispersive periodic heterogeneous medium.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I wish to thank my advisor, Randy LeVeque, not only for providing continual support and guidance, but especially for allowing (and even enthusiastically encouraging) me to pursue research that I found fascinating but that is only indirectly related to his own research program.

I will always be grateful to my other, "unofficial" advisor, Sigal Gottlieb, who has been my mentor, colleague, and friend. My mathematical writing in general has benefited greatly from her guidance and example, and several parts of this text have profited directly from her editing.

I am grateful to Chi-Wang Shu and to Colin Macdonald, in collaboration with whom some parts of this work were performed. Some parts of the text (and my laTeX skills in general) have benefited from Colin's work.

Thanks also go to the other members of my committee, Bernard Deconinck, Ken Bube, and Tom Quinn. I am grateful to my fellow students in the UW Applied Math program, who have made my time here enjoyable, and in particular to Kyle Mandli, who also proofread an early draft of this work. I am grateful to Allen Robinson and Jeff Favorite, who have mentored me and supported me with advice, encouragement, and recommendation letters.

Last, but certainly not least, I thank my daughters Elena and Victoria, and most of all my wife, Belky, for their support and inspiration.

I am grateful for generous funding during my graduate studies, provided principally by a US Dept. of Energy Computational Science Graduate Fellowship (2006-2009) under

grant DE-FG02-97ER25308, and by a US Dept. of Homeland Security Graduate Fellowship (2004-2006). Some support was also provided by AFOSR under grant number FA9550-06-1-0255. I also acknowledge significant support for travel to many conferences to present parts of this work, from various DOE and NSF grants, including a VIGRE grant.

DEDICATION

To my father.

Chapter 1

Introduction

The aim of this thesis is the development of high order numerical methods[1] for hyperbolic PDEs. One of the principal difficulties in solving hyperbolic PDEs is the handling of discontinuities, which tend to lead to spurious oscillations and numerical instability. This thesis is largely concerned with methods developed to avoid such oscillations.

In this chapter we give background information and motivation for the numerical methods developed in this thesis. We also provide an outline of the remainder of the thesis.

1.1 Motivation

1.1.1 High Order Numerical Methods for Hyperbolic Conservation Laws

Many important physical systems may be described by hyperbolic systems of conservation laws. The main difficulty in solving such systems numerically results from the tendency of the solutions to form discontinuities, even starting from smooth initial data. Numerical approximations to such discontinuities tend to develop spurious oscillations.

In contrast, exact solutions to hyperbolic conservation laws in the scalar one-dimensional case have the property that their total variation is non-increasing in time. Total variation diminishing (TVD) numerical approximations are appealing because they preserve this property of the true solution and because the TVD property is an important step

[1] The term *high order method* in this thesis generally refers to any method of greater than second order accuracy.

in proving convergence of the numerical solution. Most modern numerical methods for hyperbolic conservation laws are based on Godunov's method, which yields a TVD solution even in the presence of discontinuities. Godunov's method uses the solution to the Riemann problem – the solution to the PDE in the case of a single discontinuity.

Godunov's method gives only a first order accurate approximation, and thus is inadequate for most purposes. Godunov himself proved that any higher order accurate *linear* method must give rise to spurious oscillations. Modern second order improvements to Godunov's method get around this difficulty by employing *nonlinear* limiters that avoid oscillations and ensure the TVD property.

However, strictly TVD methods can be accurate to at most second order (first order in multi-dimensions). Various approaches have been employed in developing methods of higher order accuracy, but most make use of the general approach known as the method of lines, in which a system of PDEs is first discretized in space to yield a system of ODEs. The semi-discrete system is then integrated using a numerical ODE solver. Perhaps the most prevalent higher order discretizations for hyperbolic PDEs are discontinuous Galerkin (DG) methods and weighted essentially non-oscillatory (WENO) methods. Both DG and WENO methods are typically integrated in time using strong stability preserving Runge-Kutta methods.

1.1.2 Strong Stability Preserving Time Integration

When a high order nonlinear semi-discretization (such as a DG or WENO method) is combined with a high order ODE solver, the behavior of the resulting full discretization is usually very difficult to analyze. While it may be possible to perform a linear stability analysis of some linearization of the scheme, the results typically apply only to problems with smooth solutions, and provide little or no information about properties like total variation or positivity.

Instead, the total variation behavior is often analyzed for the exact solution to the semi-discrete system of ODEs or for the fully discrete system obtained by using forward Euler time integration. Unfortunately, this does not directly provide any information about the total variation behavior under a higher order time discretization.

Strong stability preserving (SSP) time integrators preserve properties like TVD, whenever the same property is satisfied under forward Euler integration. Just as the method of lines separates the analysis of accuracy for the spatial and temporal discretizations, the SSP approach separates the analysis of the TVD property for the spatial and temporal discretizations.

Because they preserve any convex functional bound on the solution or on differences

between two solutions, SSP methods are also referred to as TVD, monotonicity preserving, contractive, contractivity preserving, or positivity preserving methods. SSP methods may find application far beyond time integration of hyperbolic PDEs, since they are useful whenever a system of ODEs must be integrated subject to some convex bound or positivity-like constraint.

Existing high order SSP methods are either computationally inefficient, requiring several function evaluations to advance in time by the amount that Euler's method would in a single evalution, or they require unusually large amounts of memory (or, frequently, both). A goal of this thesis is the development of high order SSP methods that are optimally efficient and use the smallest possible amount of memory.

1.1.3 Flux Differencing and Wave Propagation

Written in integral form, a conservation law states that the rate of change of the solution in any region is given by the net flux through the boundary of that region. Godunov's method is based on using the solution of local Riemann problems to determine fluxes between computational cells. Most numerical methods for conservation laws follow this approach, known as *flux-differencing*.

Wave propagation methods use the solution of the Riemann problem in a different way. They compute the waves generated at each cell interface and update the solution based on the net effect of the waves. When applied to conservation laws, and if an appropriate Riemann solver is used, wave propagation methods are equivalent to flux-differencing methods.

However, many important problems are naturally modelled by hyperbolic PDEs that are not in conservation form. In this case, the flux-differencing approach cannot be used, since the system is not written in terms of a flux function. However, if the Riemann solution for the system can be computed (or approximated), wave propagation methods *can* be applied.

Wave propagation finite volume methods have proven capable of robustly handling other difficulties that are problematic for traditional flux-differencing methods. For instance, they are easily adapted to problems involving a balance between convective and source terms, or problems with spatially varying coefficients [3]. They can be applied to a variety of useful geometries by the use of mapped grids [14].

This thesis develops a high order accurate method based on wave propagation, allowing for highly accurate numerical solutions of general hyperbolic PDEs.

1.1.4 Waves in Heterogeneous Media

As mentioned above, finite volume wave propagation methods are well suited to model problems with spatially varying coefficients. An important example of such problems involves the propagation of acoustic and elastic waves in heterogeneous materials. In this case, the coefficients of the PDE are piecewise constant, but discontinuous at material interfaces. Many interesting and remarkable phenomena that do not occur in homogeneous materials may be observed in wave propagation in heterogeneous materials. These include bandgaps, spatial and temporal focusing, and solitary waves.

Solitary waves are generally understood to arise through a balance between dispersion and nonlinearity. However, first-order hyperbolic PDEs are non-dispersive. Nevertheless, solitary wave solutions to hyperbolic systems can arise in the presence of spatially varying coefficients. This was first observed computationally in [82, 84].

1.2 Outline

The first and largest part of the thesis, consisting of chapters 2 - 6, is concerned with SSP methods. The goal of this work is to determine the best possible SSP methods, in terms of accuracy, computational efficiency, and memory usage. This work led serendipitously to a new general class of memory-efficient time integrators, which form the topic of Chapter 7.

The second part of the thesis, comprising chapters 8, 9, and 10, is concerned with a new spatial discretization for hyperbolic PDEs. The goal of this work is to provide a high order accurate method (extendable to arbitrary order accuracy in principle) that is applicable to general hyperbolic systems. This is accomplished by combining wave propagation Riemann solvers with high order non-oscillatory reconstruction.

Chapter 2 introduces background material on numerical methods for initial value problems. We review the method of lines and then discuss the principal classes of methods; namely, linear multistep methods and Runge-Kutta methods. We recall the theory describing stability and accuracy of these methods. Finally, we discuss two generalizations of these methods: general linear methods and additive methods.

Chapter 3 reviews the theory of strong stability preservation in the context of Runge-Kutta methods, beginning with their development as TVD time integrators for hyperbolic PDEs. The emphasis of the chapter is on the threshold factor R and the SSP coefficient $\mathcal{C}$, which control the relative size of the strong stability preserving timestep in the case of linear and nonlinear problems, respectively. These quantities are shown to be related to the radius of absolute monotonicity of the stability function and of the Runge-Kutta method, respectively. New, simpler derivations and proofs of some of the main results

are given.

Chapter 4 deals with threshold factors for explicit methods applied to linear problems. An upper bound on the threshold factor is obtained. The problem of finding optimal threshold factors and methods is recast as a sequence of linear programming (LP) problems, and optimal factors are found for many classes of methods.

Chapter 5 deals with strong stability preserving linear multistep methods. The same solution algorithm of Chapter 4, using linear programming, is applied to find optimal explicit and implicit methods both with and without downwinding. Most of Chapters 4 and 5 corresponds to the paper [70].

Chapter 6 deals with strong stability preserving Runge-Kutta methods. Important bounds on the SSP coefficient for RK methods are reviewed and further developed. The optimization problem for finding optimal SSP RK methods is cast in a new form that allows the first investigation of implicit methods, as well as new and improved optimal explicit methods. Several good properties of these methods are analyzed, and they are applied to some simple numerical tests. This chapter corresponds mostly to the papers [69, 73].

Chapter 7 represents an intermission of sorts, as it is related to but separate from the material in the rest of the thesis. It deals with a new class of low-storage Runge-Kutta methods inspired by the nice low-storage properties of the optimal SSP methods found in Chapter 6. The new class of methods is compared with existing classes, and shown to allow for even more storage savings over these classes.

Chapter 8 presents a high order spatial discretization for hyperbolic PDEs based on the wave propagation solution to the Riemann problem. Concepts of wave propagation methods are reviewed and extended to a method of lines approach. TVD and WENO reconstruction methods are reviewed and adapted for use with the semi-discrete wave propagation scheme.

In Chapter 9, the numerical methods of Chapter 8 are applied to some numerical tests, including compressible fluid dynamics and acoustics in heterogeneous media.

In Chapter 10, these numerical methods as well as analytical techniques are applied to study the behavior of solitary elastic waves in periodic nonlinear, non-dispersive media.

In Chapter 11, we review the main contributions of this thesis and discuss interesting directions for future work.

Chapter 2

Numerical Methods for the Initial Value Problem

2.1 The Method of Lines

High order accurate numerical methods for PDEs, including finite element, finite volume, finite difference, and spectral methods, are usually based on a numerical approach known as the method of lines [81]. Given an evolution PDE

$$\frac{\partial U(\boldsymbol{x},t)}{\partial t} = \mathcal{F}(U(\boldsymbol{x},t),t) \qquad U(\boldsymbol{x},0) = U_0(\boldsymbol{x}), \tag{2.1}$$

the method of lines involves first discretizing in $\boldsymbol{x}$ and forming a discrete operator F that approximates the differential operator $\mathcal{F}$. This is known as semi-discretization and results in a system of ODEs (the *semi-discrete scheme*):

$$u'(t) = F(u,t) \qquad u(0) = u_0. \tag{2.2}$$

Here $u \in \Re^N$, where N is the number of degrees of freedom in the spatial discretization and may be very large. System (2.2) is then solved using a numerical ODE solver. For convenience, we will often omit the explicit time dependence of F in our notation.

The method of lines has some advantages over direct full discretization of the PDE:

- If the spatial discretization and temporal integration are each of order of accuracy p, then the full discretization is typically also accurate to order p. This makes con-

struction of high order schemes relatively simple.

- Different pairings of spatial and temporal discretizations can be used with ease, as the two are decoupled. This makes method-of-lines approaches useful for testing numerical methods.

- Similarly, method-of-lines discretizations are often straightforward to analyze because the space and time discretizations are decoupled.

We will frequently consider the special case of (2.2) in which F is linear and autonomous:

$$u'(t) = \mathbf{L}u \qquad u(0) = u_0. \tag{2.3}$$

Here $\mathbf{L} \in \Re^{N\times N}$ is a constant matrix.

We next review important classes of numerical methods for the initial value problems (2.2) and (2.3). For further information, see e.g. [12, 47].

2.2 Linear Multistep Methods

A linear multistep method (LMM) approximates the solution of (2.2) at successive timesteps by using information from previous timesteps. A k-step LMM has the form

$$u^n - \Delta t\beta_k F(u^n) = \sum_{j=0}^{k-1} \alpha_j u^{n-k+j} + \Delta t\beta_j F(u^{n-k+j}). \tag{2.4}$$

The method is explicit if $\beta_k = 0$. When applied to the linear autonomous IVP (2.3), the method (2.4) reduces to the iteration

$$u^n = (1-\beta_k z)^{-1} \sum_{j=0}^{k-1} \left(\alpha_j + \beta_j z\right) u^{n-k+j}, \tag{2.5}$$

where $z = \Delta t\mathbf{L}$. Since the exact solution of (2.3) satisfies

$$u(n\Delta t) = e^{nz}u_0, \tag{2.6}$$

the method (2.4) approximates the solution of (2.3) to order p if

$$(1-\beta_k z)e^{kz} = (\alpha_{k-1}+\beta_{k-1}z)e^{(k-1)z} + (\alpha_{k-2}+\beta_{k-2}z)e^{(k-2)z} + \cdots + (\alpha_0+\beta_0 z) + \mathcal{O}(z^{p+1}) \text{ for } z \to 0. \tag{2.7}$$

Expanding (2.5) and (2.7) in powers of z and equating coefficients, we obtain the conditions for order p:

$$\sum_{j=0}^{k-1} \alpha_j j^i + \sum_{j=0}^{k} \beta_j i j^{i-1} = k^i \qquad (0 \le i \le p). \tag{2.8}$$

The method is said to be consistent if (2.8) holds for $i = 0$. Although conditions (2.8) were derived for the linear IVP (2.3), they are valid also for the solution of the nonlinear IVP (2.2).

2.3 Runge-Kutta Methods

In contrast to multistep methods, Runge-Kutta methods (RKMs) retain only the most recent timestep solution. They achieve high order approximation by repeated evaluation of F.

An s-stage Runge-Kutta method is usually represented by its Butcher array, consisting of an $s \times s$ matrix $\mathbf{A}$ and two $s \times 1$ vectors $\boldsymbol{b}, \boldsymbol{c}$. The Runge-Kutta method defined by these arrays approximates the solution of the IVP (2.2) by the iteration

$$y_i = u^{n-1} + \Delta t \sum_{j=1}^{s} a_{ij} F\left(t^{n-1} + c_j \Delta t, y_j\right), \quad 1 \le i \le s \tag{2.9a}$$

$$u^n = u^{n-1} + \Delta t \sum_{j=1}^{s} b_j F\left(t^{n-1} + c_j \Delta t, y_j\right). \tag{2.9b}$$

We will assume the abscissae are determined by $c_i = \sum_{j=1}^{s} a_{ij}$.

In general, (2.9a) represents a coupled system of nonlinear equations that may be expensive to solve. Under certain conditions on $\mathbf{A}$, the system may be solved more easily. These conditions correspond to the following important subclasses of Runge-Kutta methods:

- Diagonally implicit: $a_{ij} = 0$ for $j > i$ (i.e., $\mathbf{A}$ lower triangular)
- Singly diagonally implicit: $a_{ij} = 0$ for $j > i$ and $a_{ii} = \gamma$ for $1 \le i \le s$
- Explicit: $a_{ij} = 0$ for $j \ge i$ (i.e., $\mathbf{A}$ strictly lower triangular)

When applied to the linear IVP (2.3), the Runge-Kutta method (2.9) reduces to the iteration

$$u^n = \psi(z) u^{n-1}, \tag{2.10}$$

where ψ is the *stability function* of the method. The stability function is given by

$$\psi(z) = \frac{\det(\mathbf{I} - z(\mathbf{A} - \boldsymbol{e}\boldsymbol{b}^{\mathrm{T}})}{\det(\mathbf{I} - z\mathbf{A})} = 1 + z\boldsymbol{b}^{\mathrm{T}}(\mathbf{I} - z\mathbf{A})^{-1}\boldsymbol{e}, \tag{2.11}$$

where the last equality holds whenever $(\mathbf{I} - z\mathbf{A})^{-1}$ exists. Here and throughout this thesis, $\boldsymbol{e}$ represents a vector of appropriate dimension with all entries equal to unity. We see that the solution given by the Runge-Kutta method is accurate to order p, where p is the largest integer such that

$$\psi(z) = \exp(z) + \mathcal{O}(z^{p+1}) \tag{2.12}$$

Expanding the right hand side of (2.11) in powers of z and equating coefficients in (2.12), we find that the order conditions are

$$\boldsymbol{b}^{\mathrm{T}}\mathbf{A}^{i}\boldsymbol{e} = \frac{1}{i!} \qquad\qquad 0 \leq i \leq p. \tag{2.13}$$

When solving (2.2), additional order conditions are necessary [12, 47]. We do not enter into details here, except to mention that the order conditions can be explained nicely in terms of rooted trees. Figures 2.1-2.3 illustrate the rooted trees of order 1 to 6, along with NumPy code to evaluate the corresponding order condition for each tree. These figures were generated automatically using a Python software package developed as part of this thesis. Note that conditions (2.13) correspond to the so-called tall trees (those without multiple branches). For this reason we will sometimes refer to the conditions that are relevant for linear problems as 'tall-tree' order conditions.

The forward Euler method plays a special role in the theory of strong stability preserving methods, as described in Chapter 3. It can be viewed as the simplest Runge-Kutta or multistep method:

$$u^n = u^n + \Delta t F(u^{n-1}). \tag{2.14}$$

Its stability function is $\psi(z) = 1 + z$, and it is accurate to order $p = 1$.

2.4 General Linear Methods

Runge-Kutta methods use multiple stages (function evaluations) but only the most recent timestep solution. Linear multistep methods use multiple previous timesteps but only a single stage. More general methods can be constructed using multiple stages *and* multiple steps. Such methods are referred to as general linear methods (GLMs).

When applied to the linear equation (2.3), a k-step general linear method takes the

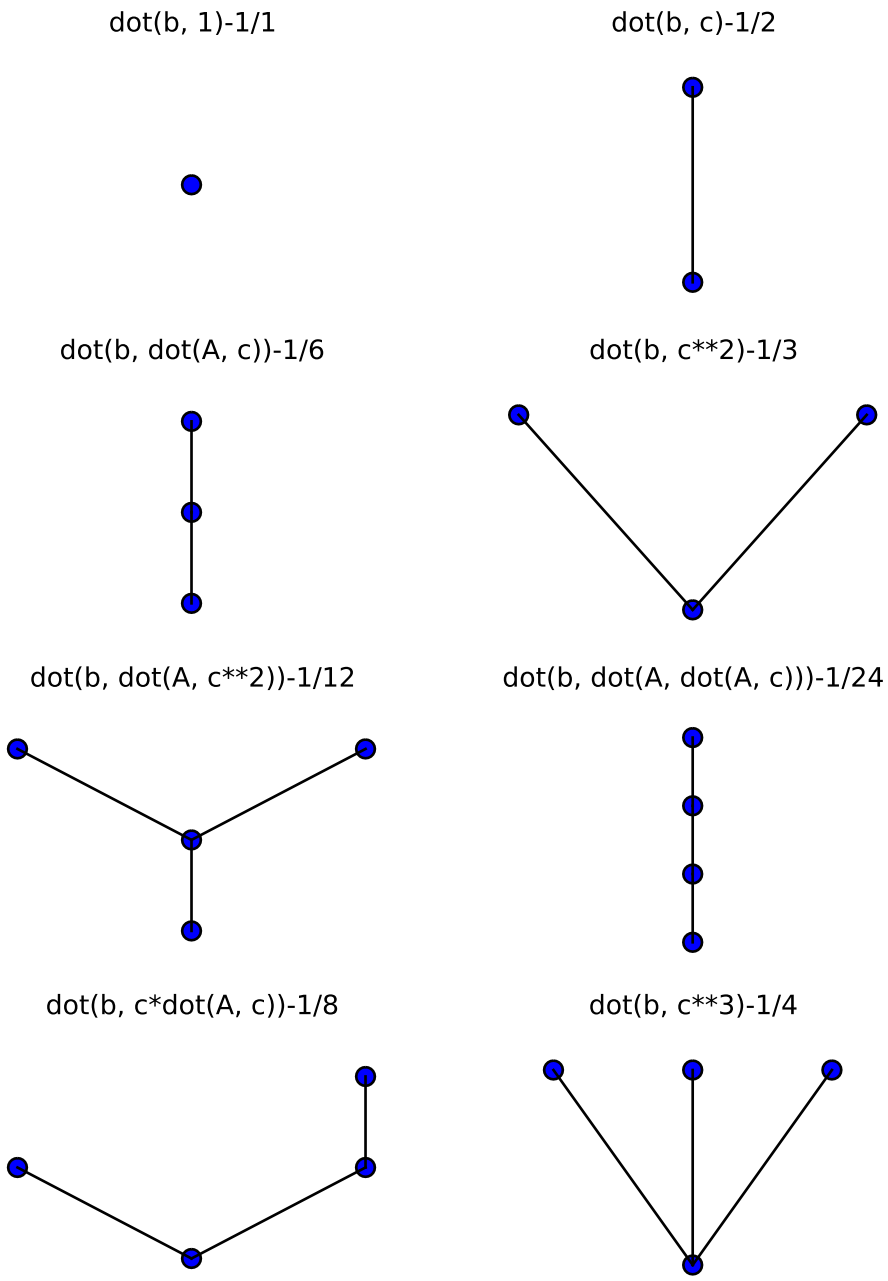


Figure 2.1: Rooted trees and corresponding Runge-Kutta order conditions of order 1 to 4

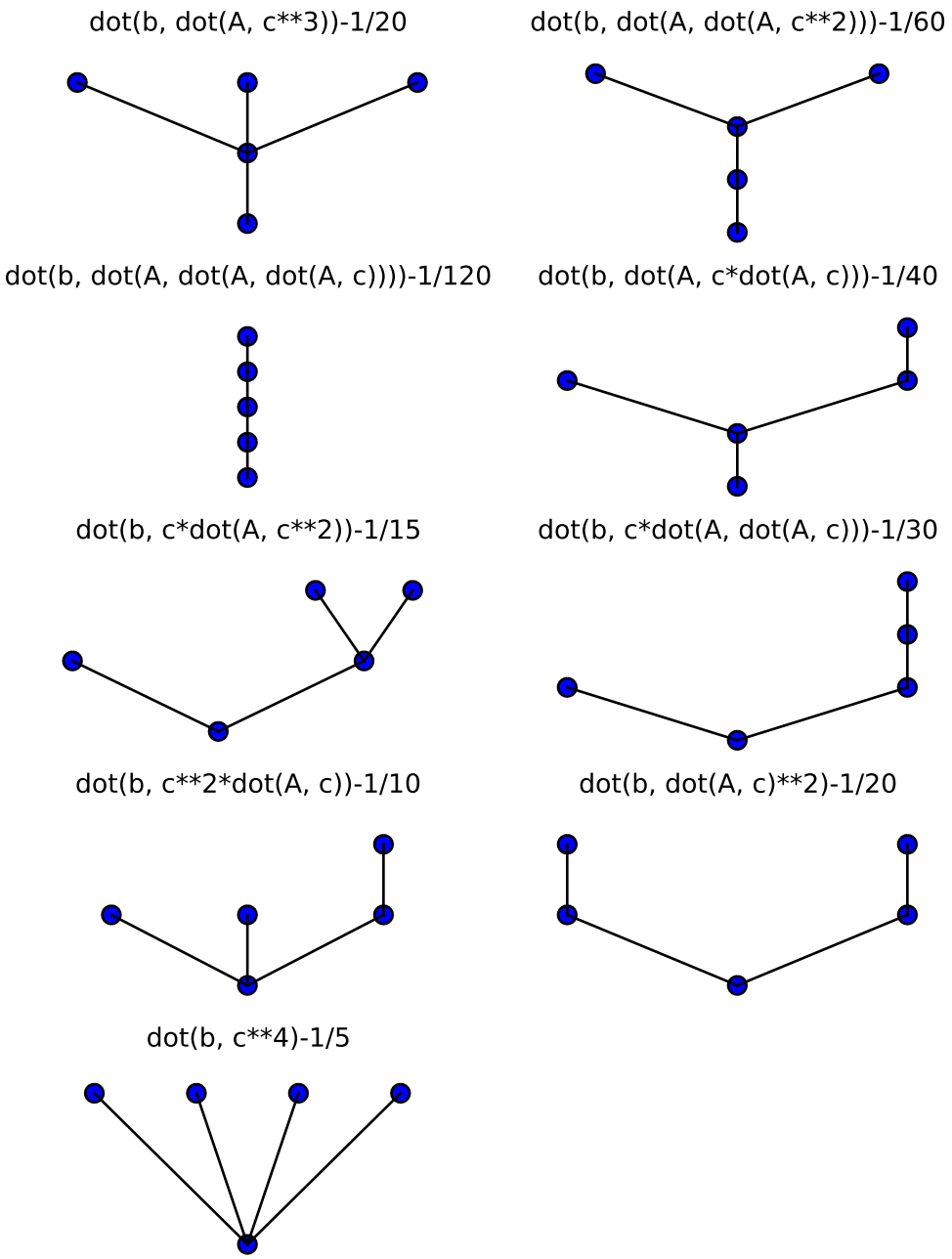


Figure 2.2: Fifth order rooted trees and corresponding Runge-Kutta order conditions

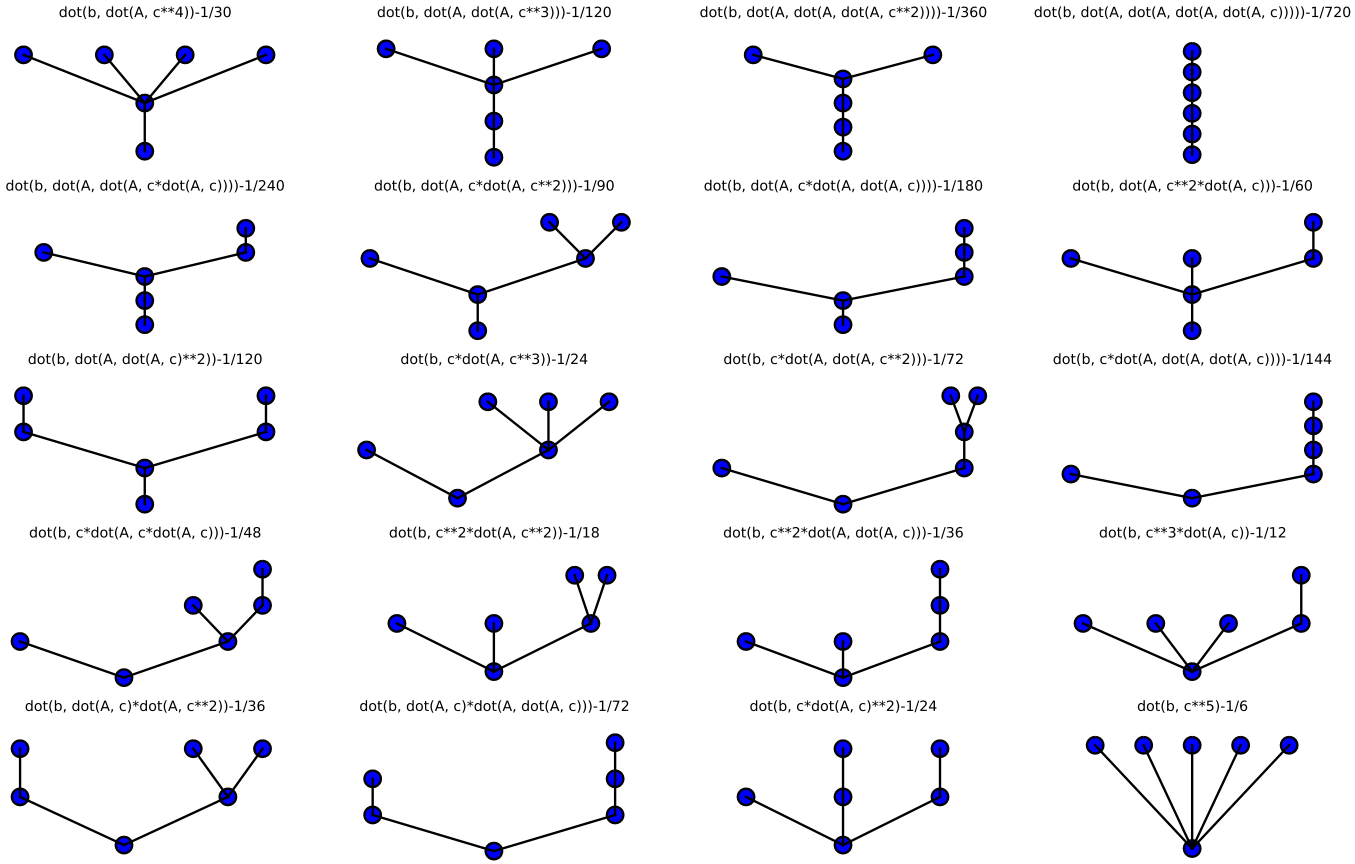


Figure 2.3: Sixth order rooted trees and corresponding Runge-Kutta order conditions

form

$$u^n = \psi_1(z)u^{n-1} + \psi_2(z)u^{n-2} + \cdots + \psi_k(z)u^{n-k}. \tag{2.15}$$

In this work, we deal only with explicit general linear methods applied to the linear IVP (2.3). Hence, we will refer to the iteration (2.15) itself as a general linear method (see, e.g., [12, 47] for a fuller description of general linear methods and their application to nonlinear IVPs). For explicit methods, each ψ_i is a polynomial whose degree is at most the number of stages of the method, s:

$$\psi_i = \sum_{j=0}^{s} a_{ij} z^j \qquad 1 \le i \le k. \tag{2.16}$$

The method (2.15) approximates the solution of (2.3) to order p if

$$e^{kz} = \psi_1(z)e^{(k-1)z} + \psi_2(z)e^{(k-2)z} + \cdots + \psi_k(z) + \mathcal{O}(z^{p+1}) \text{ for } z \to 0. \tag{2.17}$$

Combining (2.15) and (2.16), we can write a general linear method as

$$\boldsymbol{u}^n = \sum_{i=1}^{k} \sum_{j=0}^{s} a_{ij} z^j \boldsymbol{u}^{n-i}. \tag{2.18}$$

Writing the exponential functions in (2.17) as Taylor series and equating coefficients of powers of z, we find the order conditions for order p in terms of the coefficients a_{ij}:

$$\sum_{i=1}^{k} \sum_{j=0}^{q} a_{ij} \frac{(k-i)^{q-j}}{(q-j)!} = \frac{k^q}{q!} \qquad 1 \le q \le p. \tag{2.19}$$

2.5 Additive Methods

In Section 3.5 we will consider a situation in which it is useful to semi-discretize a PDE in two different ways, resulting in two right hand sides $F, \tilde{F}$, with the property that the eigenvalues $\tilde{\lambda}$ of $\tilde{F}$ are related to the eigenvalues λ of F by $\tilde{\lambda} = -\lambda$. Numerical methods that incorporate both F and $\tilde{F}$ can be understood as additive methods. Additive Runge-Kutta methods have the form

$$y_i = u^{n-1} + \Delta t \sum_{j=1}^{s} a_{ij} F\left(t^{n-1} + c_j \Delta t, y_j\right) + \Delta t \sum_{j=1}^{s} \tilde{a}_{ij} \tilde{F}\left(t^{n-1} + c_j \Delta t, y_j\right), \quad 1 \le i \le s \tag{2.20a}$$

$$u^n = u^{n-1} + \Delta t \sum_{j=1}^{s} b_j F\left(t^{n-1} + c_j \Delta t, y_j\right) + \Delta t \sum_{j=1}^{s} \tilde{b}_j \tilde{F}\left(t^{n-1} + c_j \Delta t, y_j\right). \tag{2.20b}$$

General linear methods of this type have stability function of the form

$$u_n = \psi_1(z,\tilde{z})u_{n-1} + \psi_2(z,\tilde{z})u_{n-2} + \cdots + \psi_k(z,\tilde{z})u_{n-k}, \tag{2.21}$$

where $\tilde{z} = \tilde{\lambda}\Delta t$ and the ψ_i are now bivariate functions. For explicit methods, they are polynomials with combined degree s:

$$\psi_i = \sum_{j=0}^{s}\sum_{l=0}^{j} a_{ijl} z^{j-l}\tilde{z}^l \qquad\qquad 1 \le i \le k. \tag{2.22}$$

Since $\tilde{\lambda} = -\lambda$, the method (2.21) approximates the solution of (2.3) to order p if

$$e^{kz} = \psi_1(z,-z)e^{(k-1)z} + \psi_2(z,-z)e^{(k-2)z} + \cdots + \psi_k(z,-z) + \mathcal{O}(z^{p+1}), \tag{2.23}$$

Using (2.22) and equating coefficients in (2.23) gives the order conditions:

$$\sum_{i=1}^{k}\sum_{j=0}^{q}\sum_{l=0}^{j}(-1)^l a_{ijl}\frac{(k-i)^{p-j}}{(p-j)!} = \frac{k^p}{p!} \qquad 1 \le q \le p. \tag{2.24}$$

In Section 5.4, we will consider additive linear multistep methods. These take the form

$$u_n - \Delta t\beta_k F(u_n) - \Delta t\tilde{\beta}_k\tilde{F}(u_n) = \sum_{j=0}^{k-1}\alpha_j u_{n-k+j} + \Delta t\beta_j F(u_{n-k+j}) + \Delta t\tilde{\beta}_j\tilde{F}(u_{n-k+j}). \tag{2.25}$$

Chapter 3

Strong Stability Preserving Methods and Absolute Monotonicity

Strong stability preserving (SSP) methods are numerical methods for the initial value problem that preserve convex boundedness properties of a solution, such as positivity or a total variation bound. SSP methods are widely used in the solution of hyperbolic PDEs. They have been employed in a variety of application areas, including compressible flow [122], incompressible flow [92], viscous flow [114], two-phase flow [13, 4], relativistic flow [28, 1, 125], cosmological hydrodynamics [30], magnetohydrodynamics [2], radiation hydrodynamics [89], two-species plasma flow [77], atmospheric transport [22], large-eddy simulation [91], Maxwell's equations [23], semiconductor devices [19], lithotripsy [115], geometrical optics [24], and Schrodinger equations [21, 65]. They are combined with a range of spatial discretizations, including discontinuous Galerkin methods [23], level set methods [93, 13, 29, 21, 24, 65], ENO methods [13, 28, 1], WENO methods [4, 19, 115, 30, 77, 2, 125, 91], spectral finite volume methods [114, 22], and spectral difference methods [122, 123]. This list of references is inevitably only a small sample.

Development of SSP methods was historically motivated in two ways, and developed by two groups: one focusing on ordinary differential equations, the other focusing on hyperbolic partial differential equations. Many terms have been used to describe what we refer to as strong stability preservation; here we stick mostly to this term for clarity.

Among the ODE community, work on this topic began with investigations of positivity by Bolley & Crouzeix [10] and of contractivity (or monotonicity) by Spijker [109], for linear

systems of ODEs. In these works it was noted that such properties cannot be preserved unconditionally by general linear methods of higher than first order. Conditional strong stability preservation was shown to be related to the radius of absolute monotonicity for methods satisfying a circle condition. Optimal Runge-Kutta methods for linear systems, including implicit and explicit methods, were investigated in [75, 118].

Conditions for strong stability preserving linear multistep methods in the context of nonlinear equations were given in [102], and optimal linear multistep methods for linear and nonlinear equations were investigated by Lenferink [78, 79].

The rich theory of absolute monotonicity of Runge-Kutta methods, and its relation to contractivity for nonlinear equations was developed by Kraaijevanger [76]. In addition, Kraaijevanger's work provided important results such as the order barriers for SSP Runge-Kutta methods and several optimal methods. The relation of this theory to positivity preservation was later developed by Horvath [55, 56].

Meanwhile, the idea of strong stability preservation in the context of TVD methods for hyperbolic conservation laws had been proposed by Shu & Osher [105, 107], and some of the optimal low order methods proposed by Kraaijevanger had already been proposed in this work. Shu & Osher also proposed the idea of using downwind-biased discretizations in order to preserve strong stability. The Shu-Osher approach was further developed by Gottlieb and co-authors [45, 46, 42], who proved the optimality of several methods in this context and also considered strong stability preservation for linear systems, as well as considering for the first time low-storage SSP Runge-Kutta methods and independently proving that unconditionally SSP Runge-Kutta and multistep methods cannot exist.

Ruuth & Spiteri used the Shu-Osher theory and numerical optimization to develop optimal methods over many classes, including downwind methods [111, 96, 97] (see also the work by Gottlieb & Ruuth [44]). They also proved in this (different) context some of the barriers given previously by Kraaijevanger [99].

The equivalence of the Shu-Osher theory and the theory of absolute monotonicity, both of which had been well developed for nearly fifteen years, was discovered independently and almost simultaneously by both Ferracina & Spijker [31, 33] and by Higueras [50, 51]. This connection was also independently discovered by the present author [68]. The unification of the two theories has provided a theoretical framework that is more elegant, complete, and useful than either of its predecessors.

Recently SSP theory has been extended in several important ways. Higueras has extended the theory of absolute monotonicity to include methods with downwind-biased operators [51] and, more generally, additive Runge-Kutta methods [53, 52]. A theory of SSP has been developed also for diagonally split Runge-Kutta methods, which lie outside

the class of general linear methods and are capable of being unconditionally SSP and higher than first order accurate [5, 58, 6, 55, 87]. Hundsdorfer & Ruuth have developed a class of linear multistep methods that satisfy a more general (weaker) condition than the SSP condition, but allow much larger timesteps [63, 62, 98]. First attempts to characterize the practical sharpness of SSP theory have been made in [74, 43]. New approaches to finding optimal methods have yielded new optimal methods in several classes (see [69, 73, 70] and this thesis). A general SSP theory for multistage methods applied to nonlinear equations has been developed by Spijker [110], and optimal SSP general linear methods have been investigated for certain classes [60, 25].

In this chapter we review both theories and their relationship to each other. The theory is most interesting in the context of Runge-Kutta methods, and in this chapter we focus on them exclusively. Extensions of the theory to multistep methods will be discussed in Chapter 4 and Chapter 5. In Section 3.1, we review the motivation for SSP methods as a way to obtain total variation diminishing discretizations of hyperbolic PDEs. We also review development of explicit SSP methods as convex combinations of forward Euler steps. In Section 3.2, we discuss absolutely monotonic functions, and their connection to strong stability preservation for linear IVPs. In Section 3.3, we discuss absolute monotonicity of Runge-Kutta methods, and its connection to strong stability preservation for general IVPs. In Section 3.5, we discuss SSP methods for hyperbolic PDEs that use downwind-biased spatial discretizations. Finally, in Section 3.6, we discuss the problem of finding optimal strong stability preserving methods.

As the intent of this chapter is to provide an introduction, we will focus on presenting the essential ideas in a straightforward manner, rather than providing a complete discussion of all the details. While most of the material in this chapter is a review, the construction in Section 3.3 of an example for which the SSP timestep restriction is sharp is an original contribution inspired by the proof of [110, Theorem 2.4].

3.1 Strong Stability Preservation

3.1.1 TVD Time Integration

Solutions to scalar hyperbolic conservation laws in one dimension

$$U_t + f(U)_x = 0 \tag{3.1}$$

possess the property that their total variation, defined by

$$||U||_{\mathrm{TV}} = \limsup_{\epsilon \to 0} \frac{1}{\epsilon} \int_{-\infty}^{\infty} |U(x) - U(x-\epsilon)| dx, \tag{3.2}$$

is nonincreasing in time:

$$||U(t+\Delta t)||_{\mathrm{TV}} \leq ||U(t)||_{\mathrm{TV}} \qquad \text{for } \Delta t \geq 0. \tag{3.3}$$

Certain spatial discretizations of (3.1) satisfy the same property discretely, under forward Euler time integration (2.14), subject to some maximal timestep restriction:

$$||u(t) + \Delta t F(u(t))||_{\mathrm{TV}} \leq ||u(t)||_{\mathrm{TV}} \qquad \text{for } 0 \leq \Delta t \leq \Delta t_{\mathrm{FE}}. \tag{3.4}$$

However, this does not guarantee that the numerical solution will be total variation diminishing (TVD):

$$||u^n||_{\mathrm{TV}} \leq ||u^{n-1}||_{\mathrm{TV}} \tag{3.5}$$

when u^n is computed using some other (higher order accurate) integration method. More precisely, property (3.4) by itself provides no indication of the timestep size under which (3.5) may hold for other integration schemes.

The discrete TVD property (3.5) is important for at least three reasons:

- It is satisfied by the exact solution.

- It implies that the solutions lie in a compact space, which is an important step in proving convergence for nonlinear problems with discontinuous solutions.

- Reasonable timestep sizes (i.e. CFL numbers) for high order discretizations of hyperbolic PDEs are often determined empirically by trial and error. If one can determine by analysis a maximum timestep such that (3.5) holds, this can serve as a useful guide.

It is worth mentioning that, for the latter two points, a guarantee that the solution is total variation bounded might serve just as well.

3.1.2 The Shu-Osher Form

In order to investigate explicit Runge-Kutta methods that guarantee the TVD property (3.5), Shu and Osher introduced the following representation [105]:

$$y_1 = u^{n-1} \tag{3.6a}$$

$$y_i = \sum_{j=1}^{i-1} \left(\alpha_{ij} y_j + \beta_{ij} \Delta t F(y_j)\right) \qquad 2 \le i \le s+1 \tag{3.6b}$$

$$u^n = y_{s+1}. \tag{3.6c}$$

For convenience we have shifted the indexing of α, β relative to the usual Shu-Osher form in order to make the stage indices agree with those of the Butcher form (2.9). Whereas the Butcher form is unique for an irreducible method, a given method can be represented in many ways using (3.6) (see Example 3.1.1 below).

Since consistency requires that $\sum_{j=1}^{i-1} \alpha_{ij} = 1$, then if all the coefficients α_{ij}, β_{ij} are non-negative, the form (3.6) consists of convex combinations of forward Euler steps, with a modified timestep:

$$y_i = \sum_{j=1}^{i-1} \alpha_{ij} \left(y_j + \Delta t \frac{\beta_{ij}}{\alpha_{ij}} F(y_j) \right)$$

Thus if $\Delta t \frac{\beta_{ij}}{\alpha_{ij}} \le \Delta t_{FE}$, then by 3.4

$$\begin{aligned} \|y_i\|_{\mathrm{TV}} &= \left\| \sum_{j=1}^{i-1} \alpha_{ij} \left(y_j + \Delta t \frac{\beta_{ij}}{\alpha_{ij}} F(y_j) \right) \right\|_{\mathrm{TV}} \\ &\le \sum_{j=1}^{i-1} \alpha_{ij} \left\| y_j + \Delta t \frac{\beta_{ij}}{\alpha_{ij}} F(y_j) \right\|_{\mathrm{TV}} \\ &\le \sum_{j=1}^{i-1} \alpha_{ij} \left\| y_j \right\|_{\mathrm{TV}} \le \max_{1 \le j \le i-1} \|y_j\|_{\mathrm{TV}}. \end{aligned}$$

Hence

$$\|u^n\|_{\mathrm{TV}} = \|y_{s+1}\|_{\mathrm{TV}} \le \|y_s\|_{\mathrm{TV}} \le \cdots \le \|y_1\|_{\mathrm{TV}} = \|u^{n-1}\|_{\mathrm{TV}}. \tag{3.7}$$

We see that the TVD property is preserved under the timestep restriction

$$0 \le \Delta t \le \mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta}) \Delta t_{\mathrm{FE}} \tag{3.8}$$

where

$$\mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \begin{cases} \min_{i,j} \alpha_{ij}/\beta_{ij} & \text{if } \alpha_{ij}, \beta_{ij} \geq 0 \text{ for } 1 \leq i, j \leq s \\ 0 & \text{otherwise.} \end{cases} \tag{3.9}$$

3.1.3 Strong Stability Properties

Monotonicity

The argument above is not specific to the total variation semi-norm; indeed, the only property of $||\cdot||_{\text{TV}}$ that is used is convexity. Suppose $||\cdot||$ represents any convex functional and that F satisfies

$$||u + \Delta t F(u)|| \leq ||u|| \qquad \text{for } u \in \Re^N, 0 \leq \Delta t \leq \Delta t_{\text{FE}}. \tag{3.10}$$

Then the solution obtained by the Runge-Kutta method (3.6) satisfies the *monotonicity* property

$$||u^n|| \leq ||u^{n-1}|| \tag{3.11}$$

under the timestep restriction (3.8).

Contractivity

Whereas monotonicity is concerned with the growth of the solution itself, contractivity deals with the growth of the difference between two solutions. Given two approximate solutions $u^{n-1}, \tilde{u}^{n-1}$ at time t^{n-1}, and letting $u^n, \tilde{u}^n$ denote the corresponding numerical solutions at the next timestep t^n, the numerical solution is said to be *contractive* if

$$||u^n - \tilde{u}^n|| \leq ||u^{n-1} - \tilde{u}^{n-1}||. \tag{3.12}$$

Interpreting $\tilde{u}^{n-1}$ as a perturbation of u^{n-1} due to errors, we see that contractivity implies that these errors do not grow as they are propagated.

Suppose that the solution of (2.2) is contractive under forward Euler integration:

$$||u + \Delta t F(u) - (\tilde{u} + \Delta t F(\tilde{u}))|| \leq ||u - \tilde{u}|| \qquad \text{for } u, \tilde{u} \in \Re^N, 0 \leq \Delta t \leq \Delta t_{\text{FE}}. \tag{3.13}$$

Using the convexity argument above, it is straightforward to show that (3.12) is then obtained under the timestep restriction (3.8).

Positivity

Often u represents a physical quantity, such as density, concentration, etc., that must be non-negative. In this case, it is desirable that the numerical method be *positivity preserving*:

$$u^{n-1} \geq 0 \implies u^n \geq 0 \tag{3.14}$$

It turns out that positivity is preserved under the timestep restriction (3.8), but with Δt_{FE} equal to the positivity preserving forward Euler timestep. That is, given initial data $u_0 \geq 0$, and assuming that

$$u + \Delta t F(u) \geq 0 \quad \text{for } u \geq 0, \quad 0 \leq \Delta t \leq \Delta t_{\text{FE}}, \tag{3.15}$$

then the solution obtained with the Runge-Kutta method (3.6) satisfies (3.14) under the timestep restriction (3.8).

Observe that the positivity condition (3.14) can be written as a monotonicity condition (3.11) by defining the convex functional $\|x\| = \max(\min_i(-x_i), 0)$. However, the forward Euler condition is required only for the positive orthant $u \geq 0$.

Strong stability preservation

The discussion above is summarized in the following general theorem:

Theorem 3.1.1. *Suppose that the monotonicity property* (3.11)*, the contractivity property* (3.12)*, or the positivity property* (3.14) *holds in the numerical solution of the IVP* (2.2) *when using the forward Euler method with timestep* Δt_{FE}*. Then the same property holds when using any explicit Runge-Kutta method under the timestep restriction* (3.8)*.*

We use the term *strong stability property* to refer generally to monotonicity, contractivity, and positivity properties. In the remainder of this work, results will usually be formulated in the context of monotonicity. However, the results apply equally well to contractivity and positivity. Methods that can be represented in form (3.6) with $\mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta}) > 0$ are referred to as *strong stability preserving* methods.

Note that the stable timestep is the product of only two factors, the forward Euler timestep (Δt_{FE}), which depends only on the spatial discretization, and the coefficient $\mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta})$, which depends only on the time discretization. However, different values of $\mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta})$ may be obtained for a given method, depending on the particular Shu-Osher representation chosen. This is illustrated in the following example.

Example 3.1.1. Consider the second order Runge-Kutta method, based on the trapezoidal rule, with Butcher form

$$\mathbf{A} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \qquad \boldsymbol{b} = \begin{pmatrix} 1/2 \\ 1/2 \end{pmatrix}. \tag{3.16}$$

Writing out (2.9) for this method gives

$$y_1 = u^{n-1} \tag{3.17a}$$
$$y_2 = y_1 + \Delta t F(y_1) \tag{3.17b}$$
$$u^n = y_1 + \frac{1}{2}\Delta t F(y_1) + \frac{1}{2}\Delta t F(y_2). \tag{3.17c}$$

Note that this is a Shu-Osher form (3.6) and yields $\mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = 0$. However, using the equation for y_2, we can rewrite the equation for u^n to obtain a better result. For instance,

$$u^n = \frac{3}{4}y_1 + \frac{1}{4}\Delta t F(y_1) + \frac{1}{4}y_2 + \frac{1}{2}\Delta t F(y_2) \tag{3.18}$$

yields $\mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = 1/2$. This is still not optimal; rewriting (3.17c) as

$$u^n = \frac{1}{2}y_1 + \frac{1}{2}y_2 + \frac{1}{2}\Delta t F(y_2) \tag{3.19}$$

yields $\mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = 1$. In fact, this is the optimal value of $\mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta})$ for this method, as we will be able to verify after studying the radius of absolute monotonicity in Section 3.3.

Given the dependence of $\mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta})$ on the choice of representation, we are interested in the maximal value

$$\mathcal{C} = \max_{\boldsymbol{\alpha}, \boldsymbol{\beta}} \mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta}) \tag{3.20}$$

where the maximum is taken over all representations $\boldsymbol{\alpha}, \boldsymbol{\beta}$ corresponding to a given method. We refer to $\mathcal{C}$ as the *strong stability preserving coefficient* of the method.[1]

We note here three deficiencies in the foregoing theory. First, Theorem 3.1.1 gives sufficient conditions for strong stability preservation, but makes no claims about their necessity. Second, it does not tell us how to find the value $\mathcal{C}$ in (3.20) for a given method. Finally, the theory presented here applies only to explicit methods. All of these deficiencies will be addressed in Section 3.3.

[1] In the literature, $\mathcal{C}$ has been referred to as a *CFL coefficient*. However, the CFL condition prescribes a relation between the time step and the spatial grid size, whereas the SSP coefficient describes the ratio of the strong stability preserving timestep to the strongly stable forward Euler time step.

3.2 *Absolute Monotonicity and Strong Stability Preservation for Linear IVPs*

In this section, we study strong stability preservation in the context of the linear, autonomous IVP (2.3). In this case, the forward Euler monotonicity condition (3.10) reduces to the *circle condition*

$$||\mathbf{I} + \Delta t \mathbf{L}|| \leq 1 \text{ for } 0 < \Delta t \leq \Delta t_{\text{FE}}. \tag{3.21}$$

When numerically solving the linear IVP (2.3), the maximum timestep for strong stability preservation depends on the radius of absolute monotonicity of the stability function of the numerical method.

Definition 3.2.1. Radius of absolute monotonicity *The radius of absolute monotonicity $R(\psi)$ of a function $\psi : \mathbb{R} \to \mathbb{R}$ is the largest value of r such that $\psi(z)$ and all of its derivatives exist and are nonnegative for $z \in (-r, 0]$.*

The following lemma, whose easy proof is omitted, can be seen as a special case of [76, Lemma 3.1]:

Lemma 3.2.1. *A polynomial ψ has radius of absolute monotonicity $R(\psi) \geq \xi$ if and only if $\psi(z)$ is absolutely monotonic at $z = -\xi < 0$.*

Lemma 3.2.1 indicates that the radius of absolute monotonicity can be verified simply by checking absolute monotonicity at the left endpoint.

When considering absolute monotonicity, it is often helpful to write the stability function in the form

$$\psi(z) = \sum_j \gamma_j \left(1 + \frac{z}{r}\right)^j \qquad \text{with } \gamma_j = \frac{r^j}{j!}\psi^{(j)}(-r). \tag{3.22}$$

In this form, absolute monotonicity of ψ at $z = -r$ is equivalent to non-negativity of the coefficients γ_j. By Lemma 3.2.1,

$$\gamma_j \geq 0 \iff r \leq R(\psi). \tag{3.23}$$

Additionally,when ψ corresponds to a consistent Runge-Kutta method, we have

$$\sum_j \gamma_j = 1. \tag{3.24}$$

The following result may be viewed as a special case of [109, Theorem 3.5], generalized to convex functionals.

Theorem 3.2.2. *Let the matrix* $\mathbf{L}$ *and convex functional* $||\cdot||$ *be such that the circle condition* (3.21) *is satisfied. Then the monotonicity property* (3.11) *holds for the solution of the linear autonomous IVP (2.3) by a consistent Runge-Kutta method (2.9) if the timestep satisfies*

$$0 \leq \Delta t \leq R\Delta t_{\text{FE}}, \tag{3.25}$$

where the threshold factor $R = R(\psi)$.

Proof. Taking $\Delta t = r\Delta t_{\text{FE}}$ with $r \leq R(\psi)$, and using (3.22), we have

$$\begin{aligned} ||u^n|| &= \left|\left|\sum_i \psi(\Delta t\mathbf{L})u^{n-1}\right|\right| \\ &= \left|\left|\sum_j \gamma_j \left(\mathbf{I} + \frac{\Delta t}{r}\mathbf{L}\right)^j u^{n-1}\right|\right| \\ &\leq \sum_j \gamma_j \, ||\mathbf{I} + \Delta t_{\text{FE}}\mathbf{L}||^j \, ||u^{n-1}|| \\ &\leq \sum_j \gamma_j ||u^{n-1}|| \leq ||u^{n-1}||. \end{aligned}$$

The first inequality follows from (3.23), (3.24), and convexity of $||\cdot||$, while the second follows from the circle condition (3.21). □

The essence of the proof is the observation that form (3.22) expresses the method as a convex combination of iterated forward Euler steps, in perfect analogy to the proof of the SSP property in section Section 3.1.1. Observe that the timestep restriction (3.25), like (3.8), involves two factors: Δt_{FE}, which depends only on $\mathbf{L}$ (i.e., on the particular system of ODEs), and on R, which depends only on the numerical method.

We now give an example that demonstrates the sense in which absolute monotonicity of ψ is a *necessary* condition for strong stability preservation for linear systems. Consider the 1D advection equation:

$$U_t = U_x \tag{3.26}$$

The exact solution is monotonic in the maximum norm $\|\cdot\|_\infty$. A first order upwind finite difference discretization of (3.26) yields the linear system (2.3) with

$$\mathbf{L} = \frac{1}{\Delta x}\begin{pmatrix} 1 & & & \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{pmatrix} \tag{3.27}$$

Straightforward computation reveals that for this $\mathbf{L}$ and the maximum norm, the circle condition (3.21) is satisfied for $\Delta t \leq \Delta x = \Delta t_{\text{FE}}$.

We now show that the timestep restriction (3.25) is strictly necessary in this case. That is, Theorem 3.2.3 tells us that if a RKM with stability function ψ is applied to this problem and timestep restriction (3.25) is violated, then there exists an initial condition u_0 such that $\|u^1\|_\infty > \|u_0\|_\infty$.

Theorem 3.2.3. *For any function ψ and for $\mathbf{L}$ given by* (3.27)*, we have*

$$\|\psi(\Delta t\mathbf{L})\|_\infty \leq 1 \qquad \textit{iff } \Delta t \leq R(\psi)\Delta t_{\text{FE}}.$$

Hence, for a Runge-Kutta method with stability function ψ, monotonicity in the maximum norm is guaranteed for the solution of the IVP (2.3) *iff the timestep satisfies* (3.25)*.*

Proof. The 'if' part follows from Theorem 3.2.2. To show the 'only if' part, assume $\|\psi(\Delta t\mathbf{L})\|_\infty \leq 1$. Then taking $\Delta t = r\Delta x$, we have

$$\Delta t\mathbf{L} = \mathbf{Z} = r\mathbf{I} + r\mathbf{E},$$

where

$$\mathbf{E} = \begin{pmatrix} 0 & & & \\ -1 & 0 & & \\ & \ddots & \ddots & \\ & & -1 & 0 \end{pmatrix}.$$

So, expanding ψ about $-r\mathbf{I}$,

$$\psi(\mathbf{Z}) = \sum_{j=0}^{\infty} \frac{(\mathbf{Z} - r\mathbf{I})^j}{j!}\psi^{(j)}(-r) = \sum_{j=0}^{\infty} \frac{r^j}{j!}\psi^{(j)}(-r)\mathbf{E}^j = \sum_{j=0}^{\infty} \gamma_j \mathbf{E}^j,$$

where γ_j is defined in (3.22). Since

$$\sum_{j=0}^{\infty} \gamma_j \mathbf{E}^j = \begin{pmatrix} \gamma_0 & & & \\ -\gamma_1 & \gamma_0 & & \\ \vdots & \ddots & \ddots & \\ (-1)^{N-1}\gamma_{N-1} & \cdots & -\gamma_1 & \gamma_0 \end{pmatrix}.$$

then

$$\sum_{j=0}^{N-1} |\gamma_j| = \|\psi(\mathbf{Z})\|_\infty \leq 1 = \sum_{j=0}^{\infty} \gamma_j,$$

where the last equality follows by (3.24). Since this holds for any positive integer N, we have

$$\sum_{j=0}^{\infty} |\gamma_j| \leq \sum_{j=0}^{\infty} \gamma_j,$$

so $\gamma_j \geq 0$. Thus ψ is absolutely monotonic at $-r$, so $\Delta t = r\Delta x \leq R(\psi)\Delta t_{\text{FE}}$. □

3.3 Absolute Monotonicity of Runge-Kutta Methods

In Section 3.1 we dealt with strong stability preservation for explicit Runge-Kutta methods. We found that the apparent SSP coefficient $\mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta})$ for a method depends on the particular Shu-Osher representation chosen. In this section we present a unique form for general (implicit or explicit) Runge-Kutta methods that makes apparent the true SSP coefficient (3.20). This coefficient turns out to be closely related to the concept of absolute monotonicity of Runge-Kutta methods [76]. The material is adapted primarily from [110]. For more details, the interested reader is referred also to [76, 31, 51].

A modification of the Shu-Osher form

In the investigation of absolute monotonicity of the stability function, we saw that it was useful to write the stability function in the form (3.22). Similarly, for investigating absolute monotonicity of the Runge-Kutta method, it is useful to consider, in place of the Butcher coefficients $\mathbf{A}, \boldsymbol{b}$, another form of the Runge-Kutta method. We first define

$$\mathbf{K} = \begin{pmatrix} \mathbf{A} & 0 \\ \boldsymbol{b}^{\mathrm{T}} & 0 \end{pmatrix} \tag{3.28a}$$

$$\boldsymbol{y} = (y_1, y_2, \ldots, y_s, y_{s+1})^{\mathrm{T}} \tag{3.28b}$$

$$\boldsymbol{f} = (F(y_1), F(y_2), \ldots, F(y_s), 0)^{\mathrm{T}}. \tag{3.28c}$$

This allows us to write the Runge-Kutta method compactly as follows:

$$\begin{aligned} \boldsymbol{y} &= u^{n-1}\boldsymbol{e} + \Delta t \mathbf{K} \boldsymbol{f} \\ u^n &= y_{s+1} \end{aligned} \tag{3.29}$$

Using the notation (3.28), and assuming $\mathbf{I} + r\mathbf{K}$ is invertible, we can rewrite the Runge-

Kutta method as follows:

$$\begin{aligned} \boldsymbol{y} &= u^{n-1}\boldsymbol{e} + \Delta t \mathbf{K}\boldsymbol{f} \\ (\mathbf{I} + r\mathbf{K})\boldsymbol{y} &= u^{n-1}\boldsymbol{e} + r\mathbf{K}\left(\boldsymbol{y} + \frac{\Delta t}{r}\boldsymbol{f}\right) \\ \boldsymbol{y} &= u^{n-1}(\mathbf{I} + r\mathbf{K})^{-1}\boldsymbol{e} + r(\mathbf{I} + r\mathbf{K})^{-1}\mathbf{K}\left(\boldsymbol{y} + \frac{\Delta t}{r}\boldsymbol{f}\right). \end{aligned} \tag{3.30}$$

Defining

$$\mathbf{P} = r(\mathbf{I} + r\mathbf{K})^{-1}\mathbf{K}, \qquad \boldsymbol{d} = (\mathbf{I} - \mathbf{P})\boldsymbol{e} = (\mathbf{I} + r\mathbf{K})^{-1}\boldsymbol{e}, \tag{3.31}$$

we can write (3.30) compactly:

$$\boldsymbol{y} = u^{n-1}\boldsymbol{d} + \mathbf{P}\left(\boldsymbol{y} + \frac{\Delta t}{r}\boldsymbol{f}\right). \tag{3.32}$$

Writing out equation (3.32), we see that it is similar to the Shu-Osher form:

$$y_i = d_i u^{n-1} + \sum_{j=1}^{s} p_{ij}\left(y_j + \frac{1}{r}\Delta t F(y_j)\right) \qquad 1 \le i \le s+1 \tag{3.33a}$$

$$u^n = y_{s+1}. \tag{3.33b}$$

For explicit methods, since $y_1 = u^{n-1}$, (3.33) is just the Shu-Osher form (3.6) with

$$\alpha_{ij} = \begin{cases} d_i + p_{ij} & j = 1 \\ p_{ij} & j > 1 \end{cases} \qquad \beta_{ij} = \frac{1}{r}p_{ij} \tag{3.34}$$

and if all elements of $\mathbf{P}$ and $\boldsymbol{d}$ are non-negative, the method has $\mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = r$.

Example 3.3.1. Consider again the explicit trapezoidal rule method, from Example 3.1.1. The method has

$$\mathbf{K} = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1/2 & 1/2 & 0 \end{pmatrix}. \tag{3.35}$$

Taking $r = 1/2$, we find

$$\boldsymbol{d} = \begin{pmatrix} 1 \\ 1/2 \\ 5/8 \end{pmatrix}, \qquad \mathbf{P} = \begin{pmatrix} 0 & 0 & 0 \\ 1/2 & 0 & 0 \\ 1/8 & 1/4 & 0 \end{pmatrix}, \tag{3.36}$$

which is the form (3.18), with $\mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = 1/2$. Taking $r = 1$ gives

$$\boldsymbol{d} = \begin{pmatrix} 1 \\ 0 \\ 1/2 \end{pmatrix}, \qquad \mathbf{P} = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1/2 & 0 \end{pmatrix} \tag{3.37}$$

which corresponds to the form (3.19), with $\mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = 1$. On the other hand, if we take $r > 1$, we find that $p_{31} + d_3 < 0$ so that $\mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = 0$.

Absolute Monotonicity of Runge-Kutta Methods

The form (3.32) bears a strong resemblance to the form (3.22) used to study absolute monotonicity of the stability function, with the coefficient matrices $\mathbf{P}, \boldsymbol{d}$ playing the role of the coefficients γ_j. Thus we will be concerned with the following condition (the positivity of $\mathbf{P}, \boldsymbol{d}$):

$$(\mathbf{I} + r\mathbf{K})^{-1} \text{ exists and } p_{ij}, d_j \geq 0 \qquad\qquad 1 \leq i, j \leq s+1 \tag{3.38}$$

Definition 3.3.1. *The radius of absolute monotonicity of a Runge-Kutta method, denoted $\mathcal{C}(\mathbf{K})$, is (with $\mathbf{P}, \boldsymbol{d}$ defined in* (3.31)*):*

$$\mathcal{C}(\mathbf{K}) = \sup\{r | (3.38) \textit{ is satisfied}\}.$$

If (3.38) *is not satisfied even for $r = 0$, we define $\mathcal{C}(\mathbf{K}) = 0$.*

It turns out that $\mathcal{C}(\mathbf{K})$ generalizes $\mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta})$, providing a timestep criterion for monotonicity for implicit as well as explicit RK methods. In order to show this, we need the following lemma, which we will not prove here. It is a special case of [110, Theorem 2.2(ii)], and is analogous to Lemma 3.2.1.

Lemma 3.3.1. *For all $0 \leq r \leq \mathcal{C}(\mathbf{K})$,* (3.38) *holds.*

Since $\boldsymbol{d} = (\mathbf{I} - \mathbf{P})\boldsymbol{e}$, we have

$$d_i + \sum_j p_{ij} = 1 \qquad \text{for } 1 \leq i \leq s+1. \tag{3.39}$$

Thus we have the equivalence

$$r \leq \mathcal{C}(\mathbf{K}) \iff (\mathbf{I} + r\mathbf{K})^{-1} \text{ exists and } \|[\boldsymbol{d} \;\; \mathbf{P}]\|_\infty \leq 1, \tag{3.40}$$

where $[\boldsymbol{d} \;\; \mathbf{P}]$ is the matrix formed by adjoining $\boldsymbol{d}$ and $\mathbf{P}$.

We now show the relevance of $\mathcal{C}(\mathbf{K})$ to strong stability preservation. Take $0 \leq r \leq \mathcal{C}(\mathbf{K})$. Applying $\|\cdot\|$ to both sides of (3.32), and letting $[\|x_i\|]$ denote the vector whose ith component is $\|x_i\|$, we have

$$\begin{aligned}
[\|y_i\|] &\leq (\mathbf{I}-\mathbf{P})\boldsymbol{e}\|u^{n-1}\| + \mathbf{P}[\|y_i + \frac{\Delta t}{r}F(y_i)\|] \\
[\|y_i\|] &\leq (\mathbf{I}-\mathbf{P})\boldsymbol{e}\|u^{n-1}\| + \mathbf{P}[\|y_i\|] \\
(\mathbf{I}-\mathbf{P})[\|y_i\|] &\leq (\mathbf{I}-\mathbf{P})\boldsymbol{e}\|u^{n-1}\|.
\end{aligned}$$

Since $(\mathbf{I}-\mathbf{P})^{-1} = \mathbf{I} + r\mathbf{K}$ is non-negative and invertible, we have $\|y_i\| \leq \|u^{n-1}\|$ for $1 \leq i \leq s+1$. In particular

$$\|y_{s+1}\| = \|u^n\| \leq \|u^{n-1}\|.$$

Thus monotonicity is preserved for timesteps

$$0 \leq \Delta t \leq \mathcal{C}(\mathbf{K})\Delta t_{\text{FE}}. \tag{3.41}$$

Necessity of the SSP Timestep Restriction

Before proceeding, we must first introduce the concept of reducibility. It is possible to write down a method in Butcher form (2.9) with s stages that is equivalent to a method with fewer than s stages. Such methods are said to be *reducible*.

Although multiple types of reducible methods have been defined in the literature, in this thesis we will only need to be concerned with one type. If $p_{ij} \neq 0$, we say that stage j influences stage i directly. If there are indices $i_1, i_2, ..., i_m$ such that i_n influences i_{n+1} for $1 \leq n \leq m-1$, we say that stage i_1 influences stage i_m. Clearly, if some stage j does not influence stage $s+1$, then u^n may be computed without computing y_j, so the method can be written equivalently by removing the jth row and jth column of $\mathbf{P}$ and the jth component of $\boldsymbol{d}$. If such a superfluous stage exists, we say the method is reducible; otherwise it is irreducible.

We now show the sense in which the timestep restriction (3.41) is necessary for strong stability preservation. Whereas most of this chapter is a review, the proof of Theorem 3.3.2 is an original contribution. We think it provides the clearest available explanation of the sense in which the SSP timestep restriction is necessary. Our proof is most closely related to, and was inspired by, the proof of [110, Theorem 2.4]. Compare also [76, Theorem 5.4], [57, Theorem 8], and [33, Theorem 3.4].

Theorem 3.3.2. *Given any irreducible method with radius of absolute monotonicity $\mathcal{C}(\mathbf{K})$ and any timestep $\Delta t > \mathcal{C}(\mathbf{K})\Delta t_{\text{FE}}$, there exists an IVP* (2.2) *such that the forward Euler condition*

(3.10) *holds with respect to the maximum norm, but monotonicity is violated by the Runge-Kutta solution (i.e.,* $||u^1||_\infty > ||u^0||_\infty$*).*

Proof. The proof is by construction.

For now, assume that $\mathbf{I} + r\mathbf{K}$ is invertible and let $r > \mathcal{C}(\mathbf{K})$. Let $\Delta t = r\Delta t_{\text{FE}}$ and let $\mathbf{P}, \boldsymbol{d}$ be defined by (3.31). Then (3.40) implies $\|[\boldsymbol{d} \quad \mathbf{P}]\|_\infty > 1$. The idea of the proof is to construct an IVP with $\|u_0\|_\infty = 1$ such that $\|y_j\|_\infty \geq \|[\boldsymbol{d} \quad \mathbf{P}]\|_\infty$.

Define $\widehat{\mathbf{P}}, \widehat{\boldsymbol{d}}$ by $\hat{p}_{ij} = \operatorname{sgn}(p_{ij}), \hat{d}_j = \operatorname{sgn}(d_j)$, and let $\hat{\boldsymbol{p}}_j$ denote the jth column of $\widehat{\mathbf{P}}$. We will construct an IVP with $N = s + 1$ equations, such that the Runge-Kutta stages are given by

$$y_j = d_j \widehat{\boldsymbol{d}} + \sum_k p_{jk} \hat{\boldsymbol{p}}_k. \tag{3.42}$$

For the moment, assume that the resulting stages y_j are distinct. Then we can take

$$u_0 = \widehat{\boldsymbol{d}}, \qquad F(u,t) = \begin{cases} \frac{1}{\Delta t_{\text{FE}}}(\hat{\boldsymbol{p}}_j - y_j) & \text{if } u = y_j \text{ for some } j \\ 0 & \text{otherwise.} \end{cases} \tag{3.43}$$

It is straightforward to check that (3.32) is then satisfied, so the y_j are indeed the stages of the Runge-Kutta solution. The forward Euler condition (3.10) holds, since $F(u) = 0$ if $u \neq y_j$, whereas for $u = y_j$ we have

$$\|u + \Delta t F(u)\|_\infty = \|(1-r)y_j + r\hat{\boldsymbol{p}}_j\|_\infty \leq (1-r)\|y_j\| + r\|\hat{\boldsymbol{p}}_j\| \leq \|u\|_\infty \tag{3.44}$$

since $\|\hat{\boldsymbol{p}}_j\|_\infty \leq 1 \leq \|y_j\|_\infty$.

The key property of this construction is that

$$\|y_j\|_\infty \geq (y_j)_j = d_j \operatorname{sgn}(d_j) + \sum_k p_{jk} \operatorname{sgn}(p_{jk}) = |d_j| + \sum_k |p_{jk}|. \tag{3.45}$$

Hence

$$\max_j \|y_j\|_\infty \geq \|[\boldsymbol{d} \quad \mathbf{P}]\|_\infty > 1.$$

Thus monotonicity is violated by one of the stages of the method.

This is essentially the construction used in [110] to prove the necessity of the timestep restriction (3.41). If $p_{s+1,j} \geq 0$ for all j, then for this example the monotonicity condition (3.11) is still satisfied. However, the example can be modified so that the monotonicity condition is violated by the solution itself (rather than just by the Runge-Kutta stage(s)). To do so, we assume that the method is irreducible, so that every stage influences stage

y_{s+1}. Then choose some j such that $\|y_j\|_\infty > 1$. Replace the jth column of $\widehat{\mathbf{P}}$ by $\hat{p}_{ij} = \|y_j\|_\infty \operatorname{sgn}(p_{ij})$. Then (3.42) and (3.43) are still consistent, and the forward Euler condition (3.10) still holds by (3.44). But now for any stage y_k that is directly influenced by y_j, we have (note the strict inequality, in place of (3.45))

$$\|y_m\|_\infty \geq (y_m)_m = |d_i| + \sum_k |p_{jk}| \geq 1. \tag{3.46}$$

Hence, we can modify the mth column of $\widehat{\mathbf{P}}$ by multiplying it by $\|y_m\|_\infty$. Then every stage influenced directly by y_m has maximum norm greater than 1. Proceeding in this manner, since the method is irreducible, we eventually obtain $\|u^n\|_\infty > 1$.

For the special cases in which $\mathbf{I} + r\mathbf{K}$ is singular or $y_i = y_j$ for some $i \neq j$, see pp. 1242-44 of [110]. □

Example 3.3.2. Consider the classical 4-stage, 4th order Runge-Kutta method, which has

$$\mathbf{A} = \begin{pmatrix} 0 & & & \\ \frac{1}{2} & 0 & & \\ 0 & \frac{1}{2} & 0 & \\ 0 & 0 & 1 & 0 \end{pmatrix}, \quad \boldsymbol{b} = \left(\frac{1}{6}, \frac{1}{3}, \frac{1}{3}, \frac{1}{6}\right)^{\mathrm{T}},$$

and $\mathcal{C}(\mathbf{K}) = 0$. Taking $r = 1$, we find

$$\boldsymbol{d} = \left(1, \frac{1}{2}, \frac{3}{4}, \frac{1}{4}, \frac{3}{8}\right)^{\mathrm{T}}, \qquad \mathbf{P} = \begin{pmatrix} 0 & & & & \\ \frac{1}{2} & 0 & & & \\ -\frac{1}{4} & \frac{1}{2} & 0 & & \\ \frac{1}{4} & -\frac{1}{2} & 1 & 0 & \\ \frac{1}{24} & \frac{1}{4} & \frac{1}{6} & \frac{1}{6} & 0 \end{pmatrix}.$$

Thus we set

$$\widehat{\boldsymbol{d}} = (1,1,1,1,1)^{\mathrm{T}}, \qquad \widehat{\mathbf{P}} = \begin{pmatrix} 0 & & & & \\ 1 & 0 & & & \\ -1 & 1 & 0 & & \\ 1 & -1 & 1 & 0 & \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}.$$

Then $(y_3)_3 = \frac{3}{4}\hat{d}_3 - \frac{1}{4}\hat{p}_{31} + \frac{1}{2}\hat{p}_{32} = 3/2$ and we have $\|y_3\|_\infty = 3/2$. So we set

$$\widehat{\boldsymbol{d}} = (1,1,1,1,1)^{\mathrm{T}}, \qquad \widehat{\mathbf{P}} = \begin{pmatrix} 0 & & & & \\ 1 & 0 & & & \\ -1 & 1 & 0 & & \\ 1 & -1 & 3/2 & 0 & \\ 1 & 1 & 3/2 & 1 & 0 \end{pmatrix}.$$

Then we find

$$(u^1)_5 = (y_5)_5 = \frac{3}{8}\hat{d}_5 + \frac{1}{24}\hat{p}_{51} + \frac{1}{4}\hat{p}_{52} + \frac{1}{6}\hat{p}_{53} + \frac{1}{6}\hat{p}_{54} = \frac{13}{12}$$

so $\|u^1\|_\infty \geq 13/12 > 1 = \|u^0\|_\infty$.

3.3.1 The SSP Coefficient

By (3.34) we see that for explicit methods there exists a Shu-Osher representation such that $\mathcal{C}(\mathbf{K}) \leq \mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta})$. Since (3.41) is a necessary condition for strong stability preservation, while (3.8) is a sufficient condition, it follows that $\mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta}) \leq \mathcal{C}(\mathbf{K})$ for any Shu-Osher representation. Thus

$$\mathcal{C} = \max_{\boldsymbol{\alpha},\boldsymbol{\beta}} \mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \mathcal{C}(\mathbf{K}), \tag{3.47}$$

i.e., the SSP coefficient is equal to the radius of absolute monotonicity.

For a given Runge-Kutta method, finding the optimal value of $\mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta})$ using the Shu-Osher formulation requires solving a nonlinear optimization problem. The theory of absolute monotonicity, on the other hand, provides a purely algebraic characterization of the SSP coefficient, making calculation of $\mathcal{C}$ trivial. This leads to simplification of the problem of finding optimal methods [34]. The investigations of optimal implicit and explicit Runge-Kutta methods with many stages presented in Chapter 6 would not be possible without this simplification.

An additional benefit provided by the theory of absolute monotonicity is that, for many optimal SSP methods, the form (3.32) with $r = \mathcal{C}$ results in a very sparse matrix $\mathbf{P}$ so that the method can be implemented in this form using very little memory. This will be examined in detail in Chapter 6.

For more details regarding the relationship between SSP, absolute monotonicity, and the Shu-Osher form, see [34, 33, 50, 51].

3.4 Unconditional Strong Stability Preservation

In classical numerical analysis, stability restrictions on the timestep can often be avoided by use of an appropriate implicit method. For instance, A-stable Runge-Kutta methods are stable in L^2 under arbitrary timesteps when applied to linear problems involving dissipative normal matrices.

It is natural to hope that the same can be accomplished in the context of strong stability preservation. Indeed, it is easy to show that the Backward Euler method

$$u^n = u^{n-1} + \Delta t F(u^n) \tag{3.48}$$

is unconditionally strong stability preserving [76, 50]. However, for higher order methods, unconditional strong stability preservation is not possible. The result below follows from [109, Theorem 1.3] (see also [46]).

Theorem 3.4.1. *If $\psi(z) - \exp(z) = \mathcal{O}(z^3)$ for $z \to 0$, then*

$$R(\psi) < \infty. \tag{3.49}$$

Since $\mathcal{C}(\mathbf{K}) \leq R(\psi)$, then any Runge–Kutta method of order $p > 1$ has a finite SSP coefficient:

$$p > 1 \implies \mathcal{C}(\mathbf{K}) < \infty. \tag{3.50}$$

In fact, this result holds for all general linear methods. However, this does not indicate how restrictive the step-size condition is; it may still be worthwhile to consider implicit methods if the radius of absolute monotonicity is large enough to offset the additional work involved in an implicit solver.

3.5 Negative Coefficients and Downwinding

For many Runge-Kutta methods, including the classical fourth order method, it turns out that $\mathcal{C} = 0$, so that the method is not SSP under any positive timestep. This is because, whenever the method is written in a (generalized) Shu-Osher form, some of the coefficients are necessarily negative.

However, in the solution of hyperbolic conservation laws, the SSP property can be guaranteed also for such methods, provided that we use a modified spatial discretization for these instances. The semi-discretization F of a hyperbolic system typically involves some form of upwinding. To obtain a semi-discretization that preserves strong stability in the presence of negative coefficients, we introduce an alternative semi-discretization $\tilde{F}$

that uses downwinding. For instance, in the case of linear finite difference discretizations, the matrix representing $\tilde{F}$ is just the negative transpose of that representing F. Then $\tilde{F}$ satisfies the forward Euler condition

$$\|u + \Delta t \tilde{F}(u)\| \leq \|u\|. \tag{3.51}$$

Using the discretizations $F, \tilde{F}$, we can apply additive Runge-Kutta methods (see (2.20)). Introducing

$$\tilde{\boldsymbol{f}} = (\tilde{F}(y_1), \tilde{F}(y_2), \ldots, \tilde{F}(y_s), 0)^{\mathrm{T}} \tag{3.52}$$

we can write the method as

$$\begin{aligned} \boldsymbol{y} &= u^{n-1}\boldsymbol{e} + \Delta t \mathbf{K}\boldsymbol{f} + \Delta t \tilde{\mathbf{K}}\tilde{\boldsymbol{f}}, \\ u^n &= y_{s+1} \end{aligned} \tag{3.53}$$

This can be rewritten as

$$\begin{aligned} \boldsymbol{y} &= u^{n-1}\boldsymbol{d} + \mathbf{P}\left(\boldsymbol{y} + \frac{\Delta t}{r}\boldsymbol{f}\right) + \tilde{\mathbf{P}}\left(\boldsymbol{y} + \frac{\Delta t}{r}\tilde{\boldsymbol{f}}\right), \\ u^n &= y_{s+1} \end{aligned} \tag{3.54}$$

where

$$\mathbf{P} = r(\mathbf{I} + r\mathbf{K} + r\tilde{\mathbf{K}})^{-1}\mathbf{K}, \tag{3.55}$$

$$\tilde{\mathbf{P}} = r(\mathbf{I} + r\mathbf{K} + r\tilde{\mathbf{K}})^{-1}\tilde{\mathbf{K}}, \tag{3.56}$$

$$\boldsymbol{d} = (\mathbf{I} - \mathbf{P} - \tilde{\mathbf{P}})\boldsymbol{e} = (\mathbf{I} + r\mathbf{K} + r\tilde{\mathbf{K}})^{-1}\boldsymbol{e}. \tag{3.57}$$

Following arguments similar to those of Section 3.3, we find that the method is SSP under the timestep restriction

$$\Delta t \leq \tilde{\mathcal{C}} \Delta t_{\mathrm{FE}}, \tag{3.58}$$

where

$$\tilde{\mathcal{C}} = \sup\{r | (\mathbf{I} + r\mathbf{K} + r\tilde{\mathbf{K}})^{-1} \text{ exists and } \mathbf{P}, \tilde{\mathbf{P}}, \boldsymbol{d} \geq 0\}. \tag{3.59}$$

It would seem that if both $F(y_j)$ and $\tilde{F}(y_j)$ must be computed for the same j, the computational cost as well as storage requirement for this j is doubled. For this reason, negative coefficients were avoided whenever possible in [45, 46, 42, 99, 112]. However, since, as shown in Proposition 3.3 of [45] and Theorem 4.1 in [99], it is not always possible

to avoid negative coefficients, recent studies (e.g. [96, 97, 44]) have considered efficient ways of implementing downwind discretizations. Inclusion of negative coefficients, even when not absolutely necessary, may raise the SSP coefficient enough to compensate for the additional computational cost incurred by $\tilde{F}$. Since $\tilde{F}$ is, numerically, the downwind version of F, it is sometimes possible to compute both F and $\tilde{F}$ without doubling the computational cost [44]. If F and $\tilde{F}$ do not appear for the same j, then neither the computational cost nor the storage requirement is increased.

We will discuss downwind methods further in Section 4.6 and Section 5.4. For more details on SSP methods with downwinding, see [32, 46, 41, 44, 53, 96, 97, 105, 107, 110].

3.6 Optimal SSP Methods

In solving the linear autonomous IVP (2.3) or the nonlinear, nonautonomous IVP (2.2), the maximum allowable timestep that preserves strong stability bounds is proportional to the threshold factor R or the SSP coefficient $\mathcal{C}$, respectively. Hence it is advantageous to use a method with the largest possible value of this coefficient. The next three chapters are devoted to finding such methods. Specifically, given a prescribed number of stages s, number of steps k, and order of accuracy p, we determine the largest possible value of $R(\psi)$ or $\mathcal{C}$ among s-stage, k-step, order p methods. In general, this leads to a nonlinear optimization problem subject to both equality and inequality constraints. The equality constraints are the order conditions, while the inequality constraints are the absolute monotonicity conditions.

In order to facilitate discussion of optimal methods, we introduce the following notation. The coefficients $\mathcal{C}_{s,k,p}$ and $R_{s,k,p}$ denote the maximal value of $\mathcal{C}$ or R over all explicit general linear methods with s stages, k steps, and order p. Thus, for instance, $R_{1,2,3}$ denotes the optimal threshold factor over all third order linear multistep methods with two steps. Similarly, $\tilde{\mathcal{C}}_{s,k,p}$ and $\tilde{R}_{s,k,p}$ denote the same quantities but where the maximum is taken over methods that use downwinding as well. Sometimes we will discuss implicit methods or other special classes; the optimal coefficients over these classes will be denoted using superscripts; e.g. $\mathcal{C}^{\mathrm{I}}_{s,k,p}$ for implicit Runge-Kutta methods.

3.6.1 Efficiency

The underlying assumption motivating development of these methods, is that the timestep is actually limited in practice by a requirement of monotonicity, contractivity, or positivity. We also assume that the computational effort in each timestep is dominated by the cost of evaluating F. When this is the case, the computational efficiency of a method may be

measured by the *effective SSP coefficient*:

$$\mathcal{C}_{\text{eff}} = \frac{\mathcal{C}}{s}, \tag{3.60}$$

The work required to solve a problem is inversely proportional to $\mathcal{C}_{\text{eff}}$. By definition, the forward Euler method has $\mathcal{C}_{\text{eff}} = 1$. We may analogously define the *scaled threshold factor*:

$$R_{\text{eff}} = \frac{R}{s}. \tag{3.61}$$

Clearly, $\mathcal{C}_{\text{eff}}, R_{\text{eff}}$ are more useful measures of the efficiency of a method than $\mathcal{C}, R(\psi)$. Thus, the final objective of the studies in the following chapters is to find methods with optimal values of $\mathcal{C}_{\text{eff}}, R_{\text{eff}}$.

3.6.2 The Relation Between R and C

Since the timestep restriction (3.41) guarantees strong stability preservation for a wider class of problems than the restriction (3.25), it follows that

$$\mathcal{C} \leq R \tag{3.62}$$

for any method, hence also the optimal values of $\mathcal{C}_{s,k,p}$ and $R_{s,k,p}$ over any class satisfy this inequality.

For some classes of methods, it can be shown *a priori* that $\mathcal{C} = R$. This is because the order conditions and absolute monotonicity conditions are equivalent for these methods, whether they are applied to linear or nonlinear IVPs. This is true for the following classes:

- Explicit linear multistep methods
- Explicit Runge-Kutta methods of order $p \leq 2$
- Explicit general linear methods of order $p \leq 2$

In addition, it turns out that $\mathcal{C} = R$ for the optimal methods in some other classes. It is often easier to find the optimal value of R than the optimal value of $\mathcal{C}$ over a given class of methods. If one can find the optimal R and then find a method that has SSP coefficient $\mathcal{C}$ equal to this value, it follows from (3.62) that this method is optimal. This reasoning can be used to demonstrate the optimality of many SSP methods, including explicit 3rd-order Runge-Kutta methods and one explicit 4th order Runge-Kutta method, as we will see in

Chapter 6. It is also relevant for many classes of implicit multistep methods [79, Theorem 4.2].

Chapter 4

Optimal Threshold Factors for Linear Initial Value Problems

> Clearly, the larger R, the better is the general contractivity behaviour...
>
> *M.N. Spijker (1981)*

In this chapter, we investigate the problem of finding optimal threshold factors for linear autonomous problems. Important examples of the linear IVP (2.3) include semi-discretizations of the partial differential equations describing acoustics, linear elasticity, and Maxwell's equations. Optimally contractive methods may be used to integrate such semi-discretizations, including those with time-dependent source terms [42, 20]. For instance, they have been used for integration of Maxwell's equations [20, 86] and geometrical optics [24], using discontinuous Galerkin semi-discretizations. They are also useful for providing strong stability bounds when applied to spectral semi-discretizations [40, 46, 42]. In the present work, the optimal threshold factors found in this chapter will be useful as upper bounds for the optimal SSP coefficients to be found in later chapters.

Previous studies have investigated optimal threshold factors for explicit Runge-Kutta methods [75, 118, 42] and for linear multistep methods [78, 79]. In this chapter we generalize these studies by investigating optimal threshold factors for general linear methods. By suitable reformulation of the associated optimization problem, we are also able to find optimal threshold factors for much broader classes of Runge-Kutta methods than was previously possible.

In Section 3.2, we reviewed the theory of strong stability preservation for Runge-Kutta methods applied to the linear autonomous IVP. We saw that the relative timestep for strong stability preservation is given by the radius of absolute monotonicity of the stability function of the method, also referred to as the threshold factor. We also saw that the same holds for general linear methods. Hence the problem of finding optimal explicit strong stability preserving general linear methods translates to finding functions with the largest radius of absolute monotonicity, subject to appropriate order conditions.

In Section 4.1, we define the concept of optimal threshold factor. In Section 4.2, we derive 'tall tree' order conditions for explicit general linear methods. In Section 4.3, we derive a general upper bound on the radius of absolute monotonicity for stability functions of explicit general linear methods. In Section 4.4, we show that the problem of finding optimal threshold factors can be approximately solved by solving a sequence of linear programming problems (LPs). Section 4.5 describes optimal methods obtained in this manner. Finally, in Section 4.6, we present some results on absolute monotonicity of stability functions of additive Runge-Kutta methods.

The contents of this chapter correspond to the paper [70].

4.1 Threshold Factors for General Linear Methods

In this section, we discuss conditions for general linear methods (introduced in Section 2.4) to preserve strong stability when applied to the linear autonomous IVP (2.3). A k-step method computes $\boldsymbol{u}^n$ in terms of $\boldsymbol{u}^{n-k} \dots \boldsymbol{u}^{n-1}$. Thus we replace the monotonicity condition (3.11) with

$$||\boldsymbol{u}^n|| \leq \min_{1 \leq i \leq k} ||\boldsymbol{u}^{n-i}||. \tag{4.1}$$

The theory of Section 3.2 extends in a straightforward way to general linear methods:

Theorem 4.1.1. *Let the matrix* $\mathbf{L}$ *and convex functional* $||\cdot||$ *be such that the circle condition* (3.21) *is satisfied. Then the monotonicity property* (4.1) *holds for the solution of the linear autonomous IVP (2.3) by a consistent general linear method with stability function (2.15) if the timestep satisfies*

$$\Delta t \leq \Delta t_{FE} R, \tag{4.2}$$

where the threshold factor R *is given by*

$$R = \min_{1 \leq i \leq k} R(\psi_i). \tag{4.3}$$

The proof is analogous to the proof of Theorem 3.2.2.

4.2 'Tall Tree' Order Conditions for Explicit General Linear Methods

In this section we derive order conditions for explicit GLMs applied to the linear IVP (2.3). As mentioned in Chapter 2, these correspond to the tall trees in the theory of rooted trees.

Let $\Pi_{s,k,p}$ denote the set of all ordered sets of k polynomials $(\psi_1, ..., \psi_k)$ of degree at most s satisfying the order conditions (2.19) up to order p.

Definition 4.2.1. *For given integers s, k, p, the* optimal threshold factor $R_{s,k,p}$ *is the largest threshold factor R among all explicit k-step, s-stage general linear methods of order p:*

$$R_{s,k,p} = \sup \left\{ \min_i R(\psi_i) \mid (\psi_1, ..., \psi_k) \in \Pi_{s,k,p} \right\}. \tag{4.4}$$

In order to investigate $R_{s,k,p}$, it is helpful to rewrite each of the functions ψ_i in the form used in Section 3.2:

$$\psi_i(z) = \sum_j \gamma_{ij} \left(1 + \frac{z}{r}\right)^j \qquad \text{with } \gamma_{ij} = \frac{r^j}{j!}\psi_i^{(j)}(-r). \tag{4.5}$$

Recall that, in this form, absolute monotonicity of ψ_i is equivalent to non-negativity of the coefficients γ_{ij}, i.e.

$$\gamma_{ij} \geq 0 \iff r \leq R(\psi_i). \tag{4.6}$$

Equating the right hand sides of Equations (2.16) and (4.5) gives the following relation between the coefficients a_{il} and γ_{ij}:

$$a_{il} = \frac{1}{l! r^l} \sum_{j=0}^{s} \gamma_{ij} \prod_{n=0}^{l-1} (j - n). \tag{4.7}$$

Using Equations (2.19) and (4.7), the conditions for the method to be accurate to order p can be written as

$$\sum_{i=1}^{k} \sum_{j=0}^{s} \gamma_{ij} \sum_{l=0}^{q} \binom{q}{l} \frac{(k-i)^{q-l}}{r^l} \prod_{n=0}^{l-1} (j - n) = k^q \qquad 1 \leq q \leq p. \tag{4.8}$$

4.3 An Upper Bound

Theorem 4.3.1. *For any $s, k, p > 0$, the optimal threshold factor for explicit s-stage, k-step, order p general linear methods is at most equal to the number of stages; i.e., $R_{s,k,p} \leq s$.*

Proof. Take any $(\psi_1, ..., \psi_k) \in \Pi_{s,k,p}$, and let R be the threshold factor of this method. Writing out explicitly the first two order conditions (i.e., (4.8) for $p = 0, 1$, with $r = R$)

gives

$$\sum_{i=1}^{k}\sum_{j=0}^{s}\gamma_{ij} = 1, \tag{4.9a}$$

$$\sum_{i=1}^{k}\sum_{j=0}^{s}\gamma_{ij}(j + R(k-i)) = kR. \tag{4.9b}$$

Subtracting ks times (4.9a) from (4.9b) gives

$$\sum_{i=1}^{k}\sum_{j=0}^{s}\gamma_{ij}\left(j + R(k-i) - ks\right) = k(R-s). \tag{4.10}$$

Since (for $1 \le i \le k$, $0 \le j \le s$)

$$j + R(k-i) - ks = (j-s) + R(1-i) + (R-s)(k-1) \le (R-s)(k-1), \tag{4.11}$$

then

$$\begin{aligned} k(R-s) &= \sum_{i=1}^{k}\sum_{j=0}^{s}\gamma_{ij}\left(j + R(k-i) - ks\right) \\ &\le (k-1)(R-s)\sum_{i=1}^{k}\sum_{j=0}^{s}\gamma_{ij} \\ &= (k-1)(R-s), \end{aligned}$$

which implies that $R \le s$. □

An alternate proof is as follows. Let R denote the threshold factor of the method given by $(\psi_1, \ldots, \psi_k) \in \Pi_{s,k,p}$. Consider the disk

$$D_R = \{z \in \mathbb{C} \mid |z + R| \le R\}. \tag{4.12}$$

For $z \in D_R$, $|1 + z/R| \le 1$, so using (4.5), (4.6), and (4.9), we find $|\sum_i \psi(z)| \le 1$. Hence the region of absolute stability for this method contains D_R. Then [64, Theorem 3.1] asserts that $R \le s$.

By using higher order conditions or restricting to special cases, tighter bounds can be obtained. However, we discard that approach in favor of a general method for computing $R_{s,k,p}$, given in the next section.

4.4 Solution Algorithm

We now present an algorithm for finding optimal threshold factors $R_{s,k,p}$ and corresponding methods $(\psi_1, ..., \psi_k)$ for a given number of stages s, steps k, and order p. The next two results justify our approach, which relies on bisection in r.

Lemma 3.2.1, combined with Definition 4.2.1, leads immediately to the following result, which will be useful in constructing our solution algorithm.

Corollary 4.4.1. *Let $r > 0$. Then $R_{s,k,p} \geq r$ if and only if there exists $(\psi_1, ..., \psi_k) \in \Pi_{s,k,p}$ such that each ψ_i is absolutely monotonic at $-r$.*

Corollary 4.4.1 indicates that $R_{s,k,p}$ can be found by bisection, as follows:

Optimization Algorithm
Inputs: Positive integers s, k, p and real numbers ϵ and $r_{\max}$ such that $R_{s,k,p} \leq r_{\max}$
Output: r satisfying $R_{s,k,p} - \epsilon \leq r \leq R_{s,k,p}$
1. $r_{\min} := 0$.
2. Set $r := (r_{\max} + r_{\min})/2$.
3. Determine whether there exists $(\psi_1, ..., \psi_k) \in \Pi_{s,k,p}$ such that each ψ_i is absolutely monotonic at $-r$. If so, set $r_{\min} := r$; otherwise set $r_{\max} := r$.
4. If $r_{\max} - r_{\min} < \epsilon$, set $r := r_{\min}$ and stop. Otherwise, return to step 2.

Two ingredients are necessary for the execution of this approach: a value for $r_{\max}$ and a method to solve the feasibility problem in step 3. Theorem 4.3.1 provides the bound $r_{\max}$. We now show that the feasibility problem in step 3 of our algorithm above is a linear programming problem (LP).

Using (4.6), the feasibility problem can be stated very simply: Determine whether there exist $\gamma_{ij} \geq 0$ satisfying (4.8). Since (4.8) is linear in γ_{ij}, for a fixed value of r this is a linear programming feasibility problem. Set

$$\boldsymbol{\gamma} = (\gamma_{10}, \ldots, \gamma_{1s}, \gamma_{20}, \ldots, \gamma_{2s}, \ldots, \ldots, \gamma_{k0}, \ldots, \gamma_{ks})^T \tag{4.13a}$$

$$\mathbf{k} = (1, k, k^2, \ldots, k^p)^T \tag{4.13b}$$

$$[\mathbf{B}(r)]_{q,m} = \sum_{l=0}^{q} \binom{q}{l} \frac{(k-i)^{q-l}}{r^l} \prod_{n=0}^{l-1} (j-n). \tag{4.13c}$$

where $1 \le i \le k$, $0 \le j \le s$, $1 \le q \le p$, and $m = s(i-1) + i + j$. Then the problem is given by the standard form feasibility LP (for given $s, k \ge 1$, $r > 0$):

LP 1. *Determine whether there exists* $\boldsymbol{\gamma} \ge 0$ *such that* $\mathbf{B}(r)\boldsymbol{\gamma} = \mathbf{k}$.

Modern and highly efficient LP solvers can be applied to LP 1. These solvers also return the coefficients describing $(\psi_1, ..., \psi_k)$. In the present work, the LP solvers included in MATLAB and Maple have been used. Scripts are available from the SSP website [72].

4.5 Optimal Threshold Factors for Explicit Methods

In this section we present optimal threshold factors and methods obtained using the algorithm of the previous section.

4.5.1 One-step methods

Significant results have been found previously for the case $k = 1$. Kraaijevanger [75] found optimal methods for $1 \le p \le s \le 10$, and for $p \in \{1, 2, 3, 4, s-1, s-2, s-3, s-4\}$ for any s. He also provided an algorithm for the computation of the optimal coefficient and method for arbitrary s, p. Unfortunately, the computational cost of his algorithm grows exponentially in s and p. A different but related approach was used by Gottlieb and others in [46, 42] to find results for the cases $s \in \{1, 2, p-1, p\}$ and arbitrary values of p (these results were also found by Kraaijevanger). Implementing Kraaijevanger's algorithm in Maple, we found that on a 2.5 Ghz G5 processor, for $s = 16, p = 8$, the solution requires days, and for $s \ge 20$, $p \approx s/2$, the solution would require years of computation.

Table 4.1 lists values of $R_{s,1,p}$ for $1 \le s \le 16, 1 \le p \le 30$. For a discussion of many interesting properties of this table, see [76]. Because we are primarily interested in $R_{s,k,p}$ as an upper bound on the SSP coefficient for nonlinear methods (see Chapter 6), and to save space, we do not give the optimal polynomials here. They may easily be computed with the code `Rsp.m`used to produce Table 4.1, which is available from the author's website [67].

4.5.2 One-stage multistep methods

We next consider the case $s = 1$, corresponding to linear multistep methods. Because $R_{1,k,p} = C_{1,k,p}$ the optimal threshold factors and methods are also optimal SSP coefficients and methods. Because of this, we will discuss these methods in detail in Chapter 5, when we investigate optimal SSP linear multistep methods.

Table 4.1: Optimal threshold factors $R_{s,1,p}$ for 1-step methods. Boldface entries represent new results obtained in the present work.

	p															
s	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1.00															
2	2.00	1.00														
3	3.00	2.00	1.00													
4	4.00	3.00	2.00	1.00												
5	5.00	4.00	2.65	2.00	1.00											
6	6.00	5.00	3.52	2.65	2.00	1.00										
7	7.00	6.00	4.29	3.52	2.65	2.00	1.00									
8	8.00	7.00	5.11	4.29	3.37	2.65	2.00	1.00								
9	9.00	8.00	6.00	5.11	4.10	3.37	2.65	2.00	1.00							
10	10.00	9.00	6.79	6.00	4.83	4.10	3.37	2.65	2.00	1.00						
11	11.00	10.00	7.63	6.79	**5.52**	**4.83**	4.10	3.37	2.65	2.00	1.00					
12	12.00	11.00	8.52	7.63	**6.35**	**5.52**	**4.69**	4.10	3.37	2.65	2.00	1.00				
13	13.00	12.00	9.36	8.52	**7.05**	**6.35**	**5.35**	**4.69**	4.10	3.37	2.65	2.00	1.00			
14	14.00	13.00	10.21	9.36	**7.83**	**7.05**	**6.09**	**5.35**	**4.69**	4.10	3.37	2.65	2.00	1.00		
15	15.00	14.00	11.09	10.21	**8.58**	**7.83**	**6.80**	**6.09**	**5.34**	**4.69**	4.10	3.37	2.65	2.00	1.00	
16	16.00	15.00	12.00	11.09	**9.39**	**8.58**	**7.46**	**6.80**	**5.93**	**5.34**	**4.69**	4.10	3.37	2.65	2.00	1.00
17	17.00	16.00	12.85	12.00	**10.18**	**9.39**	**8.18**	**7.46**	**6.63**	**5.93**	**5.34**	**4.69**	4.10	3.37	2.65	2.00
18	18.00	17.00	13.72	12.85	**10.95**	**10.18**	**8.89**	**8.18**	**7.32**	**6.63**	**5.93**	**5.34**	**4.69**	4.10	3.37	2.65
19	19.00	18.00	14.61	13.72	**11.76**	**10.95**	**9.64**	**8.89**	**7.93**	**7.32**	**6.60**	**5.93**	**5.34**	**4.69**	4.10	3.37
20	20.00	19.00	15.52	14.61	**12.55**	**11.76**	**10.40**	**9.64**	**8.62**	**7.93**	**7.20**	**6.60**	**5.93**	**5.34**	**4.69**	4.10
21	21.00	20.00	16.40	15.52	**13.39**	**12.55**	**11.13**	**10.40**	**9.32**	**8.62**	**7.81**	**7.20**	**6.60**	**5.93**	**5.34**	**4.69**
22	22.00	21.00	17.28	16.40	**14.20**	**13.39**	**11.84**	**11.13**	**9.99**	**9.32**	**8.49**	**7.81**	**7.20**	**6.60**	**5.93**	**5.34**
23	23.00	22.00	18.17	17.28	**15.00**	**14.20**	**12.63**	**11.84**	**10.74**	**9.99**	**9.12**	**8.49**	**7.78**	**7.20**	**6.60**	**5.93**
24	24.00	23.00	19.08	18.17	**15.84**	**15.00**	**13.36**	**12.63**	**11.42**	**10.74**	**9.76**	**9.12**	**8.36**	**7.78**	**7.20**	**6.60**
25	25.00	24.00	20.00	19.08	**16.64**	**15.84**	**14.15**	**13.36**	**12.12**	**11.42**	**10.44**	**9.76**	**9.00**	**8.35**	**7.78**	**7.20**
26	26.00	25.00	20.88	20.00	**17.48**	**16.64**	**14.93**	**14.15**	**12.83**	**12.12**	**11.15**	**10.44**	**9.65**	**9.00**	**8.35**	**7.78**
27	27.00	26.00	21.78	20.88	**18.35**	**17.48**	**15.70**	**14.93**	**13.56**	**12.83**	**11.78**	**11.15**	**10.25**	**9.65**	**8.96**	**8.35**
28	28.00	27.00	22.68	21.78	**19.16**	**18.35**	**16.45**	**15.70**	**14.29**	**13.56**	**12.49**	**11.78**	**10.92**	**10.25**	**9.54**	**8.96**
29	29.00	28.00	23.59	22.68	**19.98**	**19.16**	**17.23**	**16.45**	**15.01**	**14.29**	**13.14**	**12.49**	**11.59**	**10.92**	**10.14**	**9.54**
30	30.00	29.00	24.52	23.59	**20.84**	**19.98**	**18.04**	**17.23**	**15.80**	**15.01**	**13.86**	**13.14**	**12.21**	**11.59**	**10.82**	**10.14**

4.5.3 *Multistage multistep methods*

Table 4.2 gives optimal threshold factors for two-, three- and four-step methods with up to ten stages.

Table 4.3 gives optimal threshold factors for two-, three- and four-stage methods with up to ten steps. In general, less is gained from additional steps than from additional stages. For instance, for first order methods, the optimal threshold factor is the same regardless of the number of steps, but increases linearly with the number of stages.

Some of the particularly simple methods are described here. The optimal 2nd order 2-step, s-stage method is

$$\boldsymbol{u}_n = \frac{2(s+R)-2}{2(s+R)-1}\left(I + \frac{\Delta t \mathbf{L}}{R}\right)^s \boldsymbol{u}_{n-1} + \frac{1}{2(s+R)-1}\boldsymbol{u}_{n-2}$$

where $R = R_{s,2,2} = \sqrt{s(s-1)}$. The optimal 2nd order k-step, 2-stage method is

$$\boldsymbol{u}_n = \frac{kR}{2+R(k-1)}\left(I + \frac{\Delta t \mathbf{L}}{R}\right)^2 \boldsymbol{u}_{n-k+1} + \frac{2-R}{2+R(k-1)}\boldsymbol{u}_{n-k}$$

where $R = R_{2,k,2} = 2/(\sqrt{(k-1)^2+1} - k + 2)$. The optimal 3rd order 2-step, 8-stage method is

$$\boldsymbol{u}_n = \frac{2}{3}\left(I + \frac{\Delta t \mathbf{L}}{R}\right)^8 \boldsymbol{u}_{n-1} + \frac{1}{3}\left(I + \frac{\Delta t \mathbf{L}}{R}\right)^8 \boldsymbol{u}_{n-2}$$

where $R = R_{8,2,3} = 6$. The optimal 3rd order 3-step, 3-stage method is

$$\boldsymbol{u}_n = \frac{3}{4}\left(I + \frac{\Delta t \mathbf{L}}{R}\right)^3 \boldsymbol{u}_{n-1} + \frac{1}{4}\left(I + \frac{\Delta t \mathbf{L}}{R}\right)^3 \boldsymbol{u}_{n-3}$$

where $R = R_{3,3,3} = 2$.

The methods are presented here in a form corresponding to (4.5). This form is useful for implementation because it can be shown (see [78, Lemma 2.4(ii)]) that any optimal method will have at most p nonzero coefficients γ_{ij}. Typically they have exactly p nonzero coefficients; the last two methods above are remarkable in that they have only $p-1$ nonzero coefficients. They can be implemented very efficiently by storing only the quantities $\left(I + \frac{\Delta t \mathbf{L}}{R}\right)^s \boldsymbol{u}$ for each solution vector $\boldsymbol{u}$.

The second order methods are of particular interest because they are also optimal second order SSP methods in their respective classes. For instance, the optimal second order 2-stage methods have been proposed as SSP methods in [60], while the optimal second order 2-step methods have been proposed in [71]. The present results imply that

Table 4.2: Threshold factors $R_{s,k,p}$ of optimal 2-, 3- and 4-step general linear methods.

		p									
	s	1	2	3	4	5	6	7	8	9	10
k=2	1	1.0									
	2	2.0	1.414	0.732							
	3	3.0	2.449	1.651	1.284	0.654					
	4	4.0	3.464	2.507	2.118	1.620	1.217				
	5	5.0	4.472	3.385	2.929	2.355	1.977	1.447			
	6	6.0	5.477	4.229	3.775	3.071	2.614	2.027	1.465		
	7	7.0	6.481	5.093	4.662	3.649	3.137	2.567	2.099	1.501	
	8	8.0	7.483	6.0	5.433	4.274	3.737	3.078	2.641	2.180	1.562
	9	9.0	8.485	6.674	6.114	4.837	4.366	3.621	3.180	2.777	2.363
	10	10.0	9.487	7.352	6.797	5.461	4.979	4.239	3.796	3.335	3.000
k=3	1	1.0	0.500								
	2	2.0	1.618	1.113	0.822						
	3	3.0	2.637	2.0	1.586	1.123	0.466	0.051			
	4	4.0	3.646	2.617	2.241	1.728	1.466	1.083	0.716		
	5	5.0	4.651	3.393	3.060	2.489	2.215	1.879	1.655	1.342	
	6	6.0	5.653	4.229	3.897	3.284	2.994	2.582	2.291	2.011	1.668
	7	7.0	6.655	5.093	4.777	4.016	3.701	3.135	2.846	2.412	2.093
	8	8.0	7.657	6.0	5.624	4.633	4.307	3.708	3.393	2.898	2.515
	9	9.0	8.658	6.674	6.303	5.251	4.911	4.267	3.891	3.350	3.019
	10	10.0	9.659	7.352	6.985	5.884	5.557	4.794	4.446	3.822	3.513
k=4	1	1.0	0.667	0.333							
	2	2.0	1.721	1.243	0.934	0.542					
	3	3.0	2.732	2.0	1.684	1.223	0.928	0.555			
	4	4.0	3.737	2.617	2.331	1.883	1.664	1.406	1.188	0.602	0.325
	5	5.0	4.740	3.393	3.143	2.639	2.400	2.035	1.751	1.525	1.235
	6	6.0	5.742	4.229	3.975	3.403	3.124	2.676	2.437	2.123	1.903
	7	7.0	6.743	5.093	4.847	4.039	3.770	3.230	2.996	2.640	2.443
	8	8.0	7.744	6.0	5.723	4.633	4.384	3.801	3.572	3.197	2.969
	9	9.0	8.745	6.674	6.400	5.251	4.986	4.383	4.148	3.719	3.478
	10	10.0	9.745	7.352	7.081	5.884	5.628	4.987	4.742	4.237	4.005

Table 4.3: Threshold factors $R_{s,k,p}$ of optimal 2-,3-, and 4-stage general linear methods.

		p									
	k	1	2	3	4	5	6	7	8	9	10
s=2	1	2.0	1.0								
	2	2.0	1.414	0.732							
	3	2.0	1.618	1.113	0.823						
	4	2.0	1.721	1.243	0.934	0.542					
	5	2.0	1.781	1.243	0.984	0.674	0.434				
	6	2.0	1.820	1.243	1.028	0.796	0.575	0.263			
	7	2.0	1.847	1.243	1.063	0.833	0.691	0.413	0.174		
	8	2.0	1.867	1.243	1.089	0.876	0.750	0.484	0.308	0.010	
	9	2.0	1.883	1.243	1.109	0.905	0.765	0.573	0.395	0.186	0.021
	10	2.0	1.895	1.243	1.124	0.905	0.779	0.614	0.481	0.266	0.115
s=3	1	3.0	2.0	1.0							
	2	3.0	2.449	1.651	1.284	0.655	0.000				
	3	3.0	2.637	2.0	1.586	1.123	0.466	0.052			
	4	3.0	2.732	2.0	1.684	1.223	0.928	0.555	0.000		
	5	3.0	2.788	2.0	1.752	1.384	1.143	0.945	0.388	0.131	
	6	3.0	2.825	2.0	1.798	1.450	1.272	1.006	0.755	0.427	0.000
	7	3.0	2.851	2.0	1.831	1.467	1.301	1.018	0.851	0.656	0.257
	8	3.0	2.870	2.0	1.855	1.467	1.325	1.075	0.947	0.780	0.520
	9	3.0	2.885	2.0	1.873	1.467	1.339	1.121	0.990	0.870	0.645
	10	3.0	2.897	2.0	1.887	1.467	1.351	1.150	1.052	0.873	0.706
s=4	1	4.0	3.0	2.0	1.0						
	2	4.0	3.464	2.507	2.118	1.620	1.217	0.000			
	3	4.0	3.646	2.617	2.241	1.728	1.466	1.083	0.716		
	4	4.0	3.737	2.617	2.331	1.883	1.664	1.406	1.188	0.602	0.325
	5	4.0	3.791	2.617	2.390	1.979	1.802	1.465	1.271	1.025	0.815
	6	4.0	3.827	2.617	2.430	2.002	1.827	1.533	1.336	1.180	0.975
	7	4.0	3.852	2.617	2.459	2.002	1.848	1.603	1.448	1.260	1.130
	8	4.0	3.871	2.617	2.480	2.002	1.866	1.643	1.525	1.297	1.150
	9	4.0	3.886	2.617	2.496	2.002	1.881	1.658	1.537	1.309	1.179
	10	4.0	3.898	2.617	2.509	2.002	1.893	1.658	1.548	1.340	1.242

these methods, which in some cases were obtained by nonlinear optimization, are indeed optimal, even among much larger classes of methods than those considered in [60, 71]. These results also imply the optimality of the third order SSP general linear methods with up to three stages or three steps in [60, 71].

The threshold factors and methods in this section were obtained using the code `Rskp.m`, which is available from [67].

The numerical results obtained here suggest the following conjectures, which we leave as open problems.

Conjecture 4.5.1. *For any number of stages s and fixed* odd *order $p = 2n + 1$, there exists a k_0 beyond which all optimal methods have the same R; i.e. $R_{s,k,2n+1} = R_{s,k_0,2n+1}$ for all $k \geq k_0$.*

Conjecture 4.5.2. $R_{2,k_0,2n+1} = \max_k \mathcal{C}^{\mathrm{I}}_{1,k,p-1}$ *(see Section 5.3).*

Conjecture 4.5.3. *For any number of stages s and fixed* even *order p, $R_{s,k,p}$ is a strictly increasing function of k.*

4.6 Threshold Factors for Methods with Downwinding

In this section we investigate threshold factors for methods with downwinding, as introduced in Section 2.5 and Section 3.5. Hence, in addition to $\mathbf{L}$ we will refer to a second right hand side matrix $\tilde{\mathbf{L}}$, which would typically be given by $\tilde{\mathbf{L}} = -\mathbf{L}^{\mathrm{T}}$, and is assumed to satisfy the forward Euler condition under the same maximal timestep as $\mathbf{L}$:

$$||I + \Delta t \tilde{\mathbf{L}}|| \leq 1 \text{ for } 0 \leq \Delta t \leq \Delta t_{FE}. \tag{4.14}$$

4.6.1 Threshold Factors for Downwinded Explicit GLMs

Once again, we find that the maximal contractivity preserving timestep for these methods is related to the radius of absolute monotonicity. The definition of the radius of absolute monotonicity extends naturally to functions $\psi(z, \tilde{z})$ of two variables.

Definition 4.6.1. *The radius of absolute monotonicity $\tilde{R}(\psi)$ of $\psi : \Re \to \Re^2$ is the largest value $r \geq 0$ such that all partial derivatives of $\psi(z, \tilde{z})$ are non-negative for $z, \tilde{z} \in (-r, 0]$.*

In order to study the absolute monotonicity of a bivariate polynomial $\psi(z, \tilde{z})$, it is helpful to write it in the following form (analogous to (3.22)):

$$\psi_i(z, \tilde{z}) = \sum_{j=0}^{s} \sum_{l=0}^{j} \gamma_{ijl} \left(1 + \frac{z}{r}\right)^{j-l} \left(1 + \frac{\tilde{z}}{r}\right)^{l} \qquad \text{with } \gamma_{ijl} = \frac{r^j}{j!} \frac{\partial^j \psi_i}{\partial z^{j-l} \partial \tilde{z}^l} \tag{4.15}$$

The following lemma is analogous to Lemma 3.2.1.

Lemma 4.6.1. *A bivariate polynomial $\psi(z,\tilde{z})$ is absolutely monotonic at $-\xi < 0$ if and only if ψ has radius of absolute monotonicity $R(\psi) \geq \xi$.*

Proof. Suppose $R(\psi) \geq \xi$; then ψ is absolutely monotonic at $-\xi$ by continuity. On the other hand, suppose ψ is absolutely monotonic at $-\xi$. Write ψ in the form (4.15). Then term-by-term differentiation shows that ψ is absolutely monotonic on $(-\xi, 0]$. □

Thus we have

$$r \leq \tilde{R}(\psi_i) \iff \gamma_{ijl} \geq 0 \quad \text{for all } 0 \leq j \leq i \leq s \tag{4.16}$$

Theorem 4.6.2. *Let $||\cdot||$ be any convex functional and suppose $\mathbf{L}, \tilde{\mathbf{L}}$ satisfy the circle conditions* (3.21) *and* (4.14)*. Then the monotonicity condition* (4.1) *holds for the solution of the linear autonomous IVP* (2.3) *by method* (2.21) *if the timestep satisfies*

$$\Delta t \leq \tilde{R}\Delta t_{FE} \tag{4.17}$$

where the threshold factor $\tilde{R} = \min_i R(\psi_i(z,\tilde{z}))$.

Proof. Using (4.15),

$$\begin{aligned}
||\boldsymbol{u}^n|| &= ||\sum_i \psi_i(\Delta t\mathbf{L}, \Delta t\tilde{\mathbf{L}})\boldsymbol{u}_{n-i}|| \\
&\leq \left\| \sum_i \sum_{j=0}^{s} \sum_{l=0}^{j} \gamma_{ijl} \left(1 + \frac{\Delta t}{r}\mathbf{L}\right)^{j-l} \left(1 + \frac{\Delta t}{r}\tilde{\mathbf{L}}\right)^{l} \right\| \\
&\leq \sum_{i,j,l} \gamma_{ijl} \left\| \left(I + \frac{\Delta t}{r}\mathbf{L}\right) \right\|^{j-l} \left\| \left(I + \frac{\Delta t}{r}\tilde{\mathbf{L}}\right) \right\|^{l} ||\boldsymbol{u}_{n-i}|| \\
&\leq \sum_{i,j,l} \gamma_{ijl} ||\boldsymbol{u}_{n-i}|| \leq \max_l ||\boldsymbol{u}_{n-i}||.
\end{aligned}$$

Here we have used the fact that $\sum_{i,j,l} \gamma_{ijl} = 1$, which holds for any consistent method. □

For a more general consideration of absolutely monotonic functions that arise from considering downwind-biased methods, see [51, 53].

Let $\tilde{\Pi}_{s,k,p}$ denote the set of all ordered sets of k bivariate polynomials $\{\psi_1, ..., \psi_k\}$ of degree at most s satisfying the order conditions (2.24) up to order p.

Definition 4.6.2. *For given integers s, k, p, the* optimal downwind threshold factor $\tilde{R}_{s,k,p}$ *is the largest threshold factor among all k-step, s-stage, order p accurate general linear methods with*

downwinding:

$$\tilde{R}_{s,k,p} = \sup\left\{\min_i \tilde{R}(\psi_i) \mid \{\psi_1, ..., \psi_k\} \in \tilde{\Pi}_{s,k,p}\right\}. \tag{4.18}$$

Below, we will determine values of $\tilde{R}_{s,k,p}$ and corresponding optimal methods. By virtue of Theorem 4.6.2 these methods are optimal in terms of the maximum timestep for contractivity.

The following corollary follows from Lemma 4.6.1 and Definition 4.6.2.

Corollary 4.6.3. *Let $r > 0$. Then $\tilde{R}_{s,k,p} \geq r$ if and only if there exists $\{\psi_1, ..., \psi_k\} \in \tilde{\Pi}_{s,k,p}$ such that each $\psi_i(z, \tilde{z})$ is absolutely monotonic at $-r$.*

The next theorem provides an upper bound on $\tilde{R}_{s,k,p}$. We omit its proof, which is very similar to that of Theorem 4.3.1.

Theorem 4.6.4. *For any $s, k, p > 0$, the optimal downwind threshold factor for explicit s-stage, k-step, order p general linear methods is at most equal to the number of stages; i.e., $\tilde{R}_{s,k,p} \leq s$.*

4.6.2 Optimal One-step Methods with Downwinding

From Corollary 4.6.3, it follows that we can use the bisection algorithm from Section 4.4 to find $\tilde{R}_{s,k,p}$, by simply replacing $\Pi_{s,k,p}$ with $\tilde{\Pi}_{s,k,p}$. The effectiveness of this approach will be demonstrated by application to additive one-step methods. Thus we consider the special case $k = 1$ of the foregoing analysis. Note that these methods represent a generalization of the downwind-biased Runge-Kutta methods considered in [46, 96, 97, 44].

It would be straightforward (though more tedious) to find optimal threshold factors and methods with arbitrary numbers of stages and steps. This is left for future work.

We will use the notation of (4.15), but drop the first subscript of γ since $k = 1$. Thus (4.15) becomes

$$\psi_i(z, \tilde{z}) = \sum_{j=0}^{s} \sum_{l=0}^{j} \gamma_{ijl} \left(1 + \frac{z}{r}\right)^{j-l} \left(1 + \frac{\tilde{z}}{r}\right)^{l} \qquad \text{with } \gamma_{ijl} = \frac{r^j}{j!} \frac{\partial^j \psi_i}{\partial z^{j-l} \partial \tilde{z}^l} \tag{4.19}$$

Note that not all functions of the form (4.19) can be realized as the stability function of an s-stage additive Runge-Kutta method (2.20). This is because the form (2.20) results in certain necessary relations between the coefficients γ. Nevertheless, the results we will obtain provide upper bounds for the threshold factors of additive Runge-Kutta methods, and the methods below may be of independent interest for the integration of linear systems.

After considerable manipulation we can write $\psi(z,-z) = \sum_{i=0}^{s} C_i z^i$ where

$$C_i(r,\boldsymbol{\gamma}) = \sum_{j=i}^{s}\sum_{l=0}^{j}\gamma_{jl}\sum_{m=\max(0,i-l)}^{\min(i,j-l)}\binom{j-l}{m}\binom{l}{i-m}\frac{(-1)^{i-m}}{r^i} \tag{4.20}$$

Hence the optimal method of order at least p with at most s stages is given by

Given r find $\boldsymbol{\gamma}$ such that

$$\gamma_{jl} \geq 0 \qquad 0 \leq l \leq j \leq s \tag{4.21a}$$

$$C_i(r,\boldsymbol{\gamma}) = \frac{1}{i!} \qquad 1 \leq i \leq p. \tag{4.21b}$$

Since (4.21b) is a system of linear equations (in $\boldsymbol{\gamma}$) then for any given value of r (4.21) represents a linear programming feasibility problem. Hence we can apply the strategy of using bisection and an LP solver to find the optimal values $\tilde{R}_{s,1,p}$.

Table 4.4 gives optimal linear SSP coefficients for Runge-Kutta methods with downwinding. The optimal first order methods are simply repeated forward Euler steps. The optimal second order method of s stages is

$$\psi = \frac{2(s+\tilde{R})-1}{2(s+\tilde{R})}\left(I+\frac{\Delta t L}{\tilde{R}}\right)^s + \frac{1}{2(s+\tilde{R})}\left(I+\frac{\Delta t\tilde{\mathbf{L}}}{\tilde{R}}\right)^s.$$

where $R = \tilde{R}_{s,1,2} = \sqrt{s(s-1)}$. We found that some of the other methods have rational coefficients. The six-stage, third order method is

$$\psi = \frac{7}{12}\left(I+\frac{\Delta t L}{\tilde{R}}\right)^6 + \frac{1}{4}\left(I+\frac{\Delta t L}{\tilde{R}}\right)^4\left(I+\frac{\Delta t\tilde{\mathbf{L}}}{\tilde{R}}\right)^2 + \frac{1}{6}\left(I+\frac{\Delta t L}{\tilde{R}}\right)^3\left(I+\frac{\Delta t\tilde{\mathbf{L}}}{\tilde{R}}\right)^3.$$

where $R = \tilde{R}_{6,1,3} = 4$. The four-stage, fourth order method is

$$\psi = \frac{1}{3}\left(I+\frac{\Delta t L}{\tilde{R}}\right)^2 + \frac{17}{48}\left(I+\frac{\Delta t L}{\tilde{R}}\right)^4 + \frac{14}{48}\left(I+\frac{\Delta t L}{\tilde{R}}\right)^2\left(I+\frac{\Delta t\tilde{\mathbf{L}}}{\tilde{R}}\right)^2 + \frac{1}{48}\left(I+\frac{\Delta t\tilde{\mathbf{L}}}{\tilde{R}}\right)^4.$$

where $R = \tilde{R}_{4,1,4} = 2$.

Table 4.4: Optimal downwind threshold factors $\tilde{R}_{s,1,p}$ for one-step methods with downwinding.

s p	1	2	3	4	5	6	7	8	9	10
1	1.00									
2	2.00	1.41								
3	3.00	2.45	1.60							
4	4.00	3.46	2.49	2.00						
5	5.00	4.47	3.20	2.94	2.18					
6	6.00	5.48	4.00	3.65	3.11	2.58				
7	7.00	6.48	4.86	4.45	3.88	3.55	2.76			
8	8.00	7.48	5.77	5.31	4.57	4.32	3.72	3.15		
9	9.00	8.49	6.62	6.22	5.24	5.02	4.52	4.14	3.33	
10	10.00	9.49	7.42	7.09	5.95	5.70	5.25	4.96	4.32	3.73

Chapter 5

Optimal SSP Linear Multistep Methods

In the previous chapter, we investigated general linear methods with optimal values of the threshold factor R, which governs the maximum timestep for contractivity in the solution of the linear autonomous problem (2.3). As a special case ($s = 1$), the algorithm of the last chapter can be used to find optimally contractive explicit linear multistep methods. Since order conditions and absolute monotonicity conditions for explicit LMMs are the same in the case of linear or nonlinear problems, these methods are also optimal explicit SSP LMMs.

It turns out that the problem of finding optimal implicit SSP LMMs can also be solved using linear programming and bisection. Furthermore, optimal explicit and implicit SSP LMMs using downwinding can also be found in this way. In each case, the appropriate LP is obtained by combining the order conditions and the inequalities that are necessarily satisfied by the SSP coefficient.

In this chapter, we solve each of these problems. In Section 5.1, we generalize the algorithm from Chapter 4. In Section 5.3, we apply the algorithm to find optimal SSP linear multistep methods with positive coefficients. In Section 5.4, we consider methods that include negative coefficients. We prove a helpful result about the coefficients of optimal SSP LMMs with downwinding. This result implies that previously-found optimal methods are optimal over a larger class of methods than was previously considered. Finally, we apply the algorithm to find optimal methods.

The SSP coefficients of high order multistep methods are rather small. By considering a weaker property wherein particular starting procedures are prescribed, Ruuth and Hundsdorfer [98] have developed methods that are competitive in some cases with op-

timal Runge-Kutta methods; however these methods require more memory and in some cases were observed to violate the SSP property. Also, they are surpassed in efficiency by the optimal Runge-Kutta methods of Chapter 6.

For multistage (i.e., Runge-Kutta) methods, the problem of finding optimal SSP methods involves highly nonlinear order conditions, and cannot be solved using the current approach. Thus we defer that problem to Chapter 6.

5.1 General Solution Algorithm

In each section of this chapter, we pose an optimization problem that includes two sets of constraints. The first set of constraints are inequalities imposed by the requirement of absolute monotonicity (of either the stability function or the method). These can be written as non-negativity of a vector of coefficients: $\boldsymbol{x} \geq 0$. The second set of constraints are equalities arising from the order conditions. The key to our approach is that these equalities are linear, except for the dependence on r. That is, they take the form $\mathbf{A}(r)\boldsymbol{x} = \boldsymbol{b}$. Thus, as in the previous chapter, for a fixed value of r the constraints lead to a linear programming feasibility problem. Therefore, we can apply the algorithm of the previous chapter, using bisection in r and repeatedly solving the necessary LP.

5.2 Bounds on the SSP Coefficient

5.2.1 Explicit Methods

The existence of explicit SSP LMMs of arbitrarily high order was proven by Sand [102] (see also [78, Theorem 2.3(ii)]). From Theorem 4.3.1 with $s = 1$, we have that $R_{1,k,p} \leq 1$, so any explicit linear multistep method has $\mathcal{C} \leq 1$. The optimal first order method is forward Euler, which achieves this exactly.

A tighter bound for higher order methods was proven by Lenferink:

Theorem 5.2.1. *[78, Theorem 2.2] For $k \geq p > 1$,*

$$R_{1,k,p} \leq \frac{k-p}{k-1}. \tag{5.1}$$

For further interesting properties of $R_{1,k,p}$, the reader is referred to [78].

From Theorem 4.6.4, it follows that $\tilde{R}_{1,k,p} \leq 1$; thus the overall upper bound on the threshold factor for methods with downwinding is the same as that for methods without downwinding. However, it is interesting to note that (5.1) does not hold for $\tilde{R}_{1,k,p}$ (cf. Table 5.3).

5.2.2 *Implicit Methods*

Lenferink also showed that $\mathcal{C} \leq 2$ for implicit SSP methods of order $p > 1$[79]. This bound was shown to hold in an even more general sense in [62]. By similar means, it can be shown that this bound holds also for implicit methods with downwinding. For further discussion of the behavior of $\mathcal{C}^I_{s,k,p}$ and $R^I_{s,k,p}$ for implicit LMMs using many steps, the reader is referred to [79].

5.3 *Optimal Methods without Downwinding*

Optimal SSP linear multistep methods have been studied previously (see [102, 105, 78, 45] for explicit methods and [79] for implicit methods). Known results include optimal explicit methods of up to seventh order and twenty steps, as well as an algorithm for finding the optimal methods of arbitrary order with arbitrary number of steps. Using the present (more efficient) algorithm we have computed optimal methods of up to fortieth order.

For simplicity, we use here the traditional notation (2.4) for multistep methods, rather than the notation of the previous chapter. If $\alpha_i, \beta_i \geq 0$, the linear multistep method (2.4) can be written as a convex combination of forward Euler steps:

$$\boldsymbol{u}^n - \Delta t \beta_k F(\boldsymbol{u}^n) = \sum_{j=0}^{k-1} \alpha_j \left(\boldsymbol{u}^{n-k+j} + \Delta t \frac{\beta_j}{\alpha_j} F(\boldsymbol{u}^{n-k+j}) \right). \tag{5.2}$$

Thus the SSP coefficient is

$$\mathcal{C} = \begin{cases} \min_i \alpha_i / \beta_i & \text{if } \alpha_i, \beta_i \geq 0 \text{ for } 0 \leq i \leq k-1 \\ 0 & \text{otherwise.} \end{cases} \tag{5.3}$$

In the case of explicit linear multistep methods, since

$$\psi_i(z) = \alpha_i + \beta_i z,$$

we have

$$R(\psi_i) = \begin{cases} \alpha_i / \beta_i & \text{if } \alpha_i, \beta_i \geq 0 \\ 0 & \text{otherwise.} \end{cases}$$

Hence the threshold factor is equal to the SSP coefficient. Thus, the theories presented in Section 3.1, Section 3.2, and Section 3.3 are immediately equivalent when applied to explicit linear multistep methods, and $R = \mathcal{C}$ for these methods. For implicit linear multistep methods, the difference between $\mathcal{C}^I_{1,k,p}$ and $R^I_{1,k,p}$ is generally small (see Thm.

4.3 and Cor. 4.4 of [79]).

Rewriting (5.3), the method (5.2) has SSP coefficient $\mathcal{C} \geq r$ if

$$\beta_j \geq 0, \qquad \alpha_j - r\beta_j \geq 0 \qquad ((0 \leq j \leq k). \tag{5.4}$$

If (5.4) is satisfied for some $r > 0$, then clearly it is satisfied for any smaller positive value of r. This implies that bisection can be used to find $\mathcal{C}$.

Combining this with the order conditions (2.8), and introducing $\delta_j = \alpha_j - r\beta_j$, the feasibility problem in this case (equivalent to LP 1 with $s = 1$) takes the form (for given positive integers k, p with $p \leq k$)

LP 2. *Given r, determine whether there exist β_j, δ_j such that*

$$\begin{aligned} \beta_j, \delta_j &\geq 0 && (0 \leq j \leq k-1) \\ \sum_{j=0}^{k-1} \left((\delta_j + r\beta_j) j^i + \beta_j i j^{i-1} \right) &= k^i && (0 \leq i \leq p). \end{aligned}$$

Thus optimal methods may be found by applying the algorithm of Section 4.4, where the feasibility problem in step (3) is replaced by LP 2. Computed optimal values of the SSP coefficient for explicit methods with $1 \leq k \leq 50, 1 \leq p \leq 15$ are shown in Table 5.1. These were computed using the code `Rkp.m`, available from the author's website [67]. Note that, since $\mathcal{C} = R$ for explicit linear multistep methods, the methods given here are optimal in terms of both threshold factor and SSP coefficient. In the notation of the previous chapter, the optimal coefficients are values of $R_{1,k,p}$.

SSP coefficients of optimal implicit methods for $1 \leq k \leq 20, 1 \leq p \leq 8$ were computed in [79]. SSP coefficients of optimal methods for $1 \leq k \leq 50, 1 \leq p \leq 15$, computed using `Rkp_imp.m`, are listed in Table 5.2.

5.4 Optimal Methods with Downwinding

We now consider LMMs with downwinding; these take the form

$$\boldsymbol{u}_n - \Delta t \beta_k F(\boldsymbol{u}_n) - \Delta t \tilde{\beta}_k \tilde{F}(\boldsymbol{u}_n) = \sum_{j=0}^{k-1} \alpha_j \boldsymbol{u}_{n-k+j} + \Delta t \beta_j F(\boldsymbol{u}_{n-k+j}) + \Delta t \tilde{\beta}_j \tilde{F}(\boldsymbol{u}_{n-k+j}). \tag{5.5}$$

The method is accurate to order p if

$$\sum_{j=0}^{k-1} \alpha_j j^i + \sum_{j=0}^{k} (\beta_j - \tilde{\beta}_j) i j^{i-1} = k^i \qquad (0 \leq i \leq p) \tag{5.6}$$

Table 5.1: SSP coefficients of optimal explicit linear multistep methods (note that these are also the optimal threshold factors $R_{1,k,p}$).

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1.000														
2	1.000														
3	1.000	0.500													
4	1.000	0.667	0.333												
5	1.000	0.750	0.500	0.021											
6	1.000	0.800	0.583	0.165											
7	1.000	0.833	0.583	0.282	0.038										
8	1.000	0.857	0.583	0.359	0.145										
9	1.000	0.875	0.583	0.393	0.228										
10	1.000	0.889	0.583	0.421	0.282	0.052									
11	1.000	0.900	0.583	0.443	0.317	0.115									
12	1.000	0.909	0.583	0.460	0.345	0.175	0.018								
13	1.000	0.917	0.583	0.474	0.370	0.210	0.077								
14	1.000	0.923	0.583	0.484	0.390	0.236	0.116								
15	1.000	0.929	0.583	0.493	0.406	0.259	0.154	0.012							
16	1.000	0.933	0.583	0.501	0.411	0.276	0.177	0.044							
17	1.000	0.938	0.583	0.507	0.411	0.291	0.198	0.075							
18	1.000	0.941	0.583	0.513	0.411	0.304	0.217	0.106	0.003						
19	1.000	0.944	0.583	0.517	0.411	0.314	0.232	0.128	0.035						
20	1.000	0.947	0.583	0.521	0.411	0.322	0.246	0.148	0.063						
21	1.000	0.950	0.583	0.525	0.411	0.330	0.259	0.163	0.082						
22	1.000	0.952	0.583	0.528	0.411	0.337	0.269	0.177	0.100	0.010					
23	1.000	0.955	0.583	0.531	0.411	0.342	0.278	0.190	0.116	0.031					
24	1.000	0.957	0.583	0.534	0.411	0.347	0.286	0.201	0.131	0.048					
25	1.000	0.958	0.583	0.536	0.411	0.351	0.294	0.210	0.145	0.063					
26	1.000	0.960	0.583	0.538	0.411	0.354	0.301	0.220	0.155	0.078	0.012				
27	1.000	0.962	0.583	0.540	0.411	0.358	0.307	0.228	0.165	0.091	0.027				
28	1.000	0.963	0.583	0.542	0.411	0.360	0.312	0.234	0.174	0.105	0.042				
29	1.000	0.964	0.583	0.543	0.411	0.363	0.317	0.240	0.184	0.116	0.055				
30	1.000	0.966	0.583	0.545	0.411	0.365	0.319	0.246	0.191	0.125	0.066	0.002			
31	1.000	0.967	0.583	0.546	0.411	0.368	0.319	0.250	0.199	0.134	0.079	0.014			
32	1.000	0.968	0.583	0.548	0.411	0.370	0.319	0.255	0.205	0.142	0.089	0.026			
33	1.000	0.969	0.583	0.549	0.411	0.371	0.319	0.259	0.211	0.150	0.097	0.036			
34	1.000	0.970	0.583	0.550	0.411	0.373	0.319	0.262	0.216	0.157	0.106	0.047			
35	1.000	0.971	0.583	0.551	0.411	0.375	0.319	0.266	0.221	0.164	0.114	0.057	0.005		
36	1.000	0.971	0.583	0.552	0.411	0.376	0.319	0.269	0.225	0.169	0.121	0.066	0.016		
37	1.000	0.972	0.583	0.553	0.411	0.377	0.319	0.271	0.229	0.175	0.128	0.074	0.027		
38	1.000	0.973	0.583	0.554	0.411	0.378	0.319	0.274	0.233	0.180	0.134	0.082	0.036		
39	1.000	0.974	0.583	0.555	0.411	0.379	0.319	0.276	0.237	0.185	0.140	0.089	0.044		
40	1.000	0.974	0.583	0.555	0.411	0.380	0.319	0.278	0.240	0.189	0.146	0.096	0.052	0.003	

Table 5.2: Optimal SSP coefficients $\mathcal{C}^I_{1,k,p}$ for implicit linear multistep methods

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	∞	2.000													
2	∞	2.000	1.000												
3	∞	2.000	1.500	1.000											
4	∞	2.000	1.667	1.243	0.667										
5	∞	2.000	1.750	1.243	0.796	0.500									
6	∞	2.000	1.800	1.243	0.929	0.660	0.300								
7	∞	2.000	1.833	1.243	1.006	0.784	0.468	0.197							
8	∞	2.000	1.857	1.243	1.052	0.868	0.550	0.345	0.006						
9	∞	2.000	1.875	1.243	1.084	0.905	0.642	0.443	0.206	0.024					
10	∞	2.000	1.889	1.243	1.106	0.905	0.690	0.533	0.295	0.127					
11	∞	2.000	1.900	1.243	1.123	0.905	0.733	0.580	0.393	0.203					
12	∞	2.000	1.909	1.243	1.136	0.905	0.764	0.625	0.444	0.295	0.093				
13	∞	2.000	1.917	1.243	1.147	0.905	0.781	0.662	0.485	0.353	0.171	0.042			
14	∞	2.000	1.923	1.243	1.155	0.905	0.795	0.692	0.526	0.402	0.238	0.103			
15	∞	2.000	1.929	1.243	1.162	0.905	0.806	0.714	0.551	0.438	0.303	0.163			
16	∞	2.000	1.933	1.243	1.168	0.905	0.815	0.719	0.578	0.468	0.342	0.221	0.076		
17	∞	2.000	1.938	1.243	1.174	0.905	0.823	0.719	0.593	0.493	0.368	0.273	0.128	0.028	
18	∞	2.000	1.941	1.243	1.178	0.905	0.829	0.719	0.609	0.518	0.397	0.299	0.173	0.072	
19	∞	2.000	1.944	1.243	1.182	0.905	0.835	0.719	0.623	0.535	0.420	0.328	0.223	0.111	
20	∞	2.000	1.947	1.243	1.186	0.905	0.839	0.719	0.631	0.550	0.443	0.356	0.253	0.159	0.034
21	∞	2.000	1.950	1.243	1.189	0.905	0.844	0.719	0.639	0.564	0.459	0.374	0.279	0.191	0.081
22	∞	2.000	1.952	1.243	1.192	0.905	0.847	0.719	0.646	0.577	0.475	0.394	0.302	0.221	0.120
23	∞	2.000	1.955	1.243	1.194	0.905	0.851	0.719	0.651	0.587	0.487	0.410	0.320	0.242	0.148
24	∞	2.000	1.957	1.243	1.197	0.905	0.853	0.719	0.656	0.595	0.497	0.426	0.336	0.264	0.178
25	∞	2.000	1.958	1.243	1.199	0.905	0.856	0.719	0.661	0.596	0.506	0.439	0.353	0.283	0.201
26	∞	2.000	1.960	1.243	1.201	0.905	0.858	0.719	0.665	0.596	0.514	0.450	0.368	0.300	0.221
27	∞	2.000	1.962	1.243	1.202	0.905	0.861	0.719	0.668	0.596	0.520	0.459	0.383	0.314	0.241
28	∞	2.000	1.963	1.243	1.204	0.905	0.862	0.719	0.671	0.596	0.527	0.468	0.392	0.327	0.257
29	∞	2.000	1.964	1.243	1.205	0.905	0.864	0.719	0.674	0.596	0.532	0.476	0.402	0.341	0.270
30	∞	2.000	1.966	1.243	1.207	0.905	0.866	0.719	0.676	0.596	0.536	0.482	0.413	0.352	0.283
31	∞	2.000	1.967	1.243	1.208	0.905	0.867	0.719	0.678	0.596	0.540	0.489	0.420	0.362	0.294
32	∞	2.000	1.968	1.243	1.209	0.905	0.869	0.719	0.679	0.596	0.543	0.495	0.426	0.372	0.305
33	∞	2.000	1.969	1.243	1.210	0.905	0.870	0.719	0.681	0.596	0.547	0.500	0.433	0.380	0.316
34	∞	2.000	1.970	1.243	1.211	0.905	0.871	0.719	0.682	0.596	0.549	0.505	0.438	0.388	0.327
35	∞	2.000	1.971	1.243	1.212	0.905	0.872	0.719	0.684	0.596	0.552	0.508	0.443	0.394	0.335
36	∞	2.000	1.971	1.243	1.213	0.905	0.873	0.719	0.685	0.596	0.554	0.508	0.448	0.400	0.343
37	∞	2.000	1.972	1.243	1.214	0.905	0.874	0.719	0.686	0.596	0.556	0.508	0.451	0.406	0.349
38	∞	2.000	1.973	1.243	1.215	0.905	0.875	0.719	0.687	0.596	0.557	0.508	0.455	0.411	0.356
39	∞	2.000	1.974	1.243	1.216	0.905	0.876	0.719	0.688	0.596	0.559	0.508	0.458	0.416	0.363
40	∞	2.000	1.974	1.243	1.217	0.905	0.877	0.719	0.689	0.596	0.560	0.508	0.461	0.420	0.368

and has SSP coefficient $\tilde{\mathcal{C}} \geq r$ if

$$\beta_j, \tilde{\beta}_j \geq 0 \qquad (0 \leq j \leq k-1) \tag{5.7a}$$

$$\alpha_j - r(\beta_j + \tilde{\beta}_j) \geq 0 \qquad (0 \leq j \leq k-1). \tag{5.7b}$$

If (5.7) is satisfied for some positive value of $\tilde{\mathcal{C}}$, then it holds for any smaller positive value. Hence we are again justified in using bisection to find optimal methods. The feasibility problem to be solved at each step is (with $\delta_j = \alpha_j - r(\beta_j + \tilde{\beta}_j)$):

LP 3. *Given r, determine whether there exist $\beta_j, \tilde{\beta}_j, \delta_j$ such that*

$$\beta_j, \tilde{\beta}_j, \delta_j \geq 0 \qquad (0 \leq j \leq k-1)$$

$$\sum_{j=0}^{k-1} \left(\delta_j + r(\beta_j + \tilde{\beta}_j)\right) j^i + \sum_{j=0}^{k} (\beta_j - \tilde{\beta}_j) i j^{i-1} = k^i \qquad (0 \leq i \leq p)$$

Optimal SSP *explicit* linear multistep methods with downwinding have been studied previously [105, 46, 98, 44]. Previous searches for methods in this class were restricted to methods satisfying $\beta_j \tilde{\beta}_j = 0$. The following lemma shows that this restriction is always satisfied by optimal methods.

Lemma 5.4.1. *Any optimal SSP method of the form* (5.5) *has the property that $\beta_j \tilde{\beta}_j = 0$ for each j.*

Proof. Note that the order conditions (5.6) depend only on the difference $\beta_j - \tilde{\beta}_j$, while the inequality constraint (5.7a) can be written as (setting $r = \tilde{\mathcal{C}}$)

$$\tilde{\mathcal{C}} \leq \alpha_j / (\beta_j + \tilde{\beta}_j).$$

Suppose that an optimal method has $\beta_j > \tilde{\beta}_j > 0$ for $j \in J_1 \subset (0, 1, \ldots, k-1)$ and $\tilde{\beta}_j > \beta_j > 0$ for $j \in J_2 \subset (0, 1, \ldots, k-1)$. Then for $j \in J_1$ define $\beta_j^* = \beta_j - \tilde{\beta}_j$ and $\tilde{\beta}_j^* = 0$; for $j \in J_2$ define $\tilde{\beta}_j^* = \tilde{\beta}_j - \beta_j$ and $\beta_j^* = 0$. Then (α, β^*) satisfies (5.7) and (5.6) with a larger value of $\tilde{\mathcal{C}}$, which is a contradiction. □

Lemma 5.4.1 could be used to write the optimization problem in terms of fewer variables, obtaining the formulation used in [98, 44]. This results in an integer programming problem with 2^k possibilities, each of which is solved by nonlinear programming in [98, 44]. Although the NLP subproblems are solved very quickly, this approach is unreasonable for very large k because of the exponential growth of the number of subproblems. By retaining all of the $\beta_j, \tilde{\beta}_j$, we are able to solve the problem using linear programming;

furthermore, the number of linear programming solves is independent of k. Only the size of the LPs grows with k, and only at a linear rate.

Optimal coefficients of explicit methods are known for methods with up to $k = 10$ steps and order $p = 6$ [44]. Optimal SSP coefficients for $1 \leq k \leq 50, 1 \leq p \leq 15$ are given in Table 5.3. These were computed using the code `Rkp_dw.m`, available from the author's website [67]. For $p \leq 6, k \leq 10$, these values agree with those reported in [44]. The remaining values are new. Note that, for large values of k, there is little or no difference between the SSP coefficient for the optimal methods with and without downwinding.

Coefficients of optimal implicit methods with $1 \leq k \leq 40, 1 \leq p \leq 15$, computed using `Rkp_imp_dw.m`, are given in Table 5.4. This is the first investigation of implicit SSP methods with downwinding.

Table 5.3: Optimal SSP coefficients $\tilde{\mathcal{C}}_{1,k,p}$ for explicit linear multistep methods with downwinding

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1.000														
2	1.000	0.500													
3	1.000	0.667	0.287												
4	1.000	0.750	0.415	0.159											
5	1.000	0.800	0.517	0.237	0.087										
6	1.000	0.833	0.583	0.283	0.131	0.046									
7	1.000	0.857	0.583	0.360	0.187	0.081	0.024								
8	1.000	0.875	0.583	0.394	0.223	0.107	0.044	0.013							
9	1.000	0.889	0.583	0.424	0.261	0.142	0.068	0.025	0.007						
10	1.000	0.900	0.583	0.447	0.299	0.175	0.086	0.037	0.013	0.003					
11	1.000	0.909	0.583	0.464	0.326	0.199	0.104	0.056	0.022	0.007	0.002				
12	1.000	0.917	0.583	0.477	0.351	0.215	0.131	0.069	0.030	0.012	0.004	0.000			
13	1.000	0.923	0.583	0.487	0.372	0.242	0.154	0.083	0.044	0.019	0.007	0.002	0.000		
14	1.000	0.929	0.583	0.496	0.391	0.266	0.171	0.096	0.055	0.024	0.010	0.004	0.001	0.000	
15	1.000	0.933	0.583	0.503	0.407	0.280	0.187	0.112	0.066	0.034	0.016	0.006	0.002	0.001	0.000
16	1.000	0.938	0.583	0.509	0.411	0.295	0.199	0.133	0.075	0.043	0.020	0.008	0.003	0.001	0.000
17	1.000	0.941	0.583	0.514	0.411	0.307	0.216	0.147	0.086	0.051	0.025	0.013	0.005	0.002	0.001
18	1.000	0.944	0.583	0.519	0.411	0.317	0.232	0.161	0.095	0.060	0.032	0.016	0.007	0.003	0.001
19	1.000	0.947	0.583	0.523	0.411	0.326	0.243	0.173	0.111	0.067	0.040	0.020	0.010	0.004	0.002
20	1.000	0.950	0.583	0.526	0.411	0.334	0.254	0.182	0.125	0.077	0.046	0.024	0.013	0.005	0.002
21	1.000	0.952	0.583	0.529	0.411	0.339	0.263	0.189	0.135	0.084	0.053	0.030	0.015	0.007	0.004
22	1.000	0.955	0.583	0.532	0.411	0.344	0.272	0.201	0.146	0.090	0.059	0.036	0.018	0.010	0.004
23	1.000	0.957	0.583	0.535	0.411	0.349	0.280	0.212	0.157	0.103	0.066	0.040	0.022	0.012	0.006
24	1.000	0.958	0.583	0.537	0.411	0.353	0.288	0.220	0.164	0.113	0.073	0.047	0.026	0.015	0.007
25	1.000	0.960	0.583	0.539	0.411	0.356	0.295	0.228	0.171	0.122	0.079	0.052	0.031	0.017	0.009
26	1.000	0.962	0.583	0.541	0.411	0.359	0.302	0.235	0.177	0.132	0.085	0.057	0.036	0.020	0.011
27	1.000	0.963	0.583	0.543	0.411	0.362	0.307	0.241	0.183	0.140	0.092	0.063	0.040	0.023	0.013
28	1.000	0.964	0.583	0.544	0.411	0.365	0.312	0.246	0.189	0.147	0.102	0.069	0.045	0.027	0.015
29	1.000	0.966	0.583	0.546	0.411	0.367	0.317	0.250	0.196	0.154	0.109	0.074	0.050	0.031	0.018
30	1.000	0.967	0.583	0.547	0.411	0.369	0.319	0.254	0.202	0.159	0.117	0.079	0.054	0.034	0.020
31	1.000	0.968	0.583	0.548	0.411	0.371	0.319	0.257	0.209	0.164	0.124	0.084	0.058	0.038	0.023
32	1.000	0.969	0.583	0.549	0.411	0.373	0.319	0.261	0.214	0.169	0.131	0.089	0.064	0.043	0.026
33	1.000	0.970	0.583	0.550	0.411	0.374	0.319	0.265	0.219	0.173	0.137	0.096	0.068	0.047	0.029
34	1.000	0.971	0.583	0.551	0.411	0.376	0.319	0.268	0.224	0.178	0.142	0.102	0.072	0.050	0.033
35	1.000	0.971	0.583	0.552	0.411	0.377	0.319	0.271	0.228	0.182	0.146	0.109	0.076	0.054	0.036
36	1.000	0.972	0.583	0.553	0.411	0.378	0.319	0.273	0.231	0.185	0.151	0.115	0.081	0.058	0.039
37	1.000	0.973	0.583	0.554	0.411	0.379	0.319	0.276	0.235	0.190	0.155	0.120	0.085	0.062	0.043
38	1.000	0.974	0.583	0.555	0.411	0.380	0.319	0.278	0.238	0.194	0.159	0.126	0.089	0.066	0.046
39	1.000	0.974	0.583	0.556	0.411	0.381	0.319	0.279	0.240	0.198	0.162	0.130	0.095	0.070	0.049
40	1.000	0.975	0.583	0.556	0.411	0.382	0.319	0.281	0.243	0.202	0.166	0.134	0.100	0.073	0.052

Table 5.4: Optimal SSP coefficients $\tilde{\mathcal{C}}^I_{1,k,p}$ for implicit linear multistep methods with downwinding

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	∞	2.000													
2	∞	2.000	1.303												
3	∞	2.000	1.591	1.000											
4	∞	2.000	1.710	1.243	0.744										
5	∞	2.000	1.776	1.243	0.871	0.544									
6	∞	2.000	1.817	1.243	0.976	0.672	0.384								
7	∞	2.000	1.845	1.243	1.035	0.788	0.535	0.261							
8	∞	2.000	1.866	1.243	1.073	0.870	0.619	0.410	0.171						
9	∞	2.000	1.882	1.243	1.093	0.905	0.668	0.474	0.280	0.107					
10	∞	2.000	1.894	1.243	1.117	0.905	0.718	0.542	0.369	0.191	0.065				
11	∞	2.000	1.905	1.243	1.132	0.905	0.755	0.585	0.433	0.280	0.119	0.038			
12	∞	2.000	1.913	1.243	1.143	0.905	0.776	0.629	0.485	0.304	0.195	0.078	0.021		
13	∞	2.000	1.920	1.243	1.153	0.905	0.791	0.664	0.512	0.376	0.234	0.121	0.047	0.013	
14	∞	2.000	1.926	1.243	1.160	0.905	0.803	0.693	0.540	0.424	0.296	0.175	0.077	0.028	0.007
15	∞	2.000	1.931	1.243	1.167	0.905	0.813	0.714	0.570	0.455	0.342	0.227	0.116	0.050	0.014
16	∞	2.000	1.935	1.243	1.172	0.905	0.821	0.719	0.588	0.479	0.376	0.240	0.157	0.074	0.029
17	∞	2.000	1.939	1.243	1.177	0.905	0.828	0.719	0.605	0.497	0.404	0.266	0.199	0.103	0.047
18	∞	2.000	1.943	1.243	1.181	0.905	0.834	0.719	0.620	0.520	0.430	0.330	0.228	0.140	0.067
19	∞	2.000	1.946	1.243	1.185	0.905	0.839	0.719	0.630	0.536	0.445	0.354	0.263	0.170	0.089
20	∞	2.000	1.949	1.243	1.188	0.905	0.843	0.719	0.638	0.551	0.457	0.379	0.292	0.201	0.124
21	∞	2.000	1.951	1.243	1.191	0.905	0.847	0.719	0.645	0.565	0.472	0.396	0.313	0.227	0.148
22	∞	2.000	1.953	1.243	1.194	0.905	0.850	0.719	0.651	0.577	0.484	0.413	0.335	0.256	0.172
23	∞	2.000	1.956	1.243	1.196	0.905	0.853	0.719	0.656	0.587	0.495	0.424	0.352	0.276	0.196
24	∞	2.000	1.957	1.243	1.198	0.905	0.856	0.719	0.660	0.595	0.504	0.435	0.369	0.293	0.220
25	∞	2.000	1.959	1.243	1.200	0.905	0.858	0.719	0.664	0.596	0.513	0.443	0.383	0.308	0.240
26	∞	2.000	1.961	1.243	1.202	0.905	0.860	0.719	0.668	0.596	0.520	0.453	0.392	0.325	0.258
27	∞	2.000	1.962	1.243	1.203	0.905	0.862	0.719	0.671	0.596	0.526	0.460	0.402	0.339	0.274
28	∞	2.000	1.964	1.243	1.205	0.905	0.864	0.719	0.674	0.596	0.532	0.469	0.410	0.352	0.289
29	∞	2.000	1.965	1.243	1.206	0.905	0.866	0.719	0.676	0.596	0.536	0.477	0.417	0.362	0.301
30	∞	2.000	1.966	1.243	1.208	0.905	0.867	0.719	0.678	0.596	0.540	0.483	0.423	0.370	0.314
31	∞	2.000	1.967	1.243	1.209	0.905	0.869	0.719	0.679	0.596	0.543	0.489	0.430	0.379	0.326
32	∞	2.000	1.968	1.243	1.210	0.905	0.870	0.719	0.681	0.596	0.547	0.495	0.435	0.386	0.336
33	∞	2.000	1.969	1.243	1.211	0.905	0.871	0.719	0.682	0.596	0.549	0.500	0.439	0.392	0.343
34	∞	2.000	1.970	1.243	1.212	0.905	0.872	0.719	0.684	0.596	0.552	0.505	0.444	0.398	0.350
35	∞	2.000	1.971	1.243	1.213	0.905	0.873	0.719	0.685	0.596	0.554	0.508	0.447	0.403	0.356
36	∞	2.000	1.972	1.243	1.214	0.905	0.874	0.719	0.686	0.596	0.556	0.508	0.451	0.408	0.363
37	∞	2.000	1.973	1.243	1.215	0.905	0.875	0.719	0.687	0.596	0.558	0.508	0.455	0.413	0.369
38	∞	2.000	1.973	1.243	1.216	0.905	0.876	0.719	0.688	0.596	0.559	0.508	0.458	0.417	0.374
39	∞	2.000	1.974	1.243	1.216	0.905	0.877	0.719	0.689	0.596	0.560	0.508	0.461	0.421	0.378
40	∞	2.000	1.975	1.243	1.217	0.905	0.878	0.719	0.690	0.596	0.562	0.508	0.463	0.425	0.382

Chapter 6

SSP Runge-Kutta Methods

> I am currently working on a survey of implicit RK schemes which are SSP... I don't anticipate your part of the project will take a lot of your time, maybe a month or so.
>
> *S. Gottlieb to the author (Jan. 2007)*

In this chapter we present new optimal SSP Runge-Kutta methods of both explicit and implicit type. Among the various classes of ODE solvers, explicit Runge-Kutta methods have proven to have the best potential for large effective SSP coefficients. As we saw in Chapter 5, results for SSP multistep methods are disappointing, in the sense that the SSP coefficients are very small [107, 46, 63, 41] (even for implicit multistep methods with downwinding, $\mathcal{C} \leq 2$ if $p > 1$).

In Section 6.1, we review many known results from the literature that provide bounds on $\mathcal{C}$ for the classes of RK methods we will consider (explicit, implicit, diagonally implicit, and singly diagonally implicit). We also draw some simple new conclusions providing further bounds.

In Section 6.2, we describe the formulation of the optimization problem that is used in our numerical searches. The optimization problem associated with finding optimal methods is formulated using the theory of absolutely monotonic methods described in Chapter 3, which turns out to be advantageous relative to formulations that were used previously. This formulation uses the Butcher form and a simplified algebraic charac-

terization of the SSP coefficient, as suggested in [32]. This allows for solution of the optimization problem for higher order and larger numbers of stages.

In Section 6.3, we investigate numerically optimal implicit SSP Runge-Kutta methods. Our results provide the first implicit RK methods with large SSP coefficients, as well as the first such methods of order six. The methods have many good properties: they are diagonally implicit and possess small error coefficients and useful low-storage implementations. We find that the ratio of SSP coefficient for optimal implicit versus explicit methods is rather small, however.

In Section 6.4, we investigate numerically optimal explicit SSP Runge-Kutta methods. Our new explicit methods are superior to known methods both in terms of computational efficiency (larger $\mathcal{C}_{\text{eff}}$) and memory usage. They also have small error coefficients and good internal stability.

6.1 Bounds on the SSP Coefficient for Runge-Kutta Methods

In this section, we will discuss bounds on $\mathcal{C}$ and $R(\psi)$ for various classes of Runge-Kutta methods. The relationships between these classes are illustrated in Figure 6.1.

The SSP property is a very strong requirement, and imposes severe restrictions on other properties of a Runge–Kutta method. We now review these results and draw a few additional conclusions that will guide our search for optimal methods in the next section.

Some results in this and the next section will deal with the optimal value of $\mathcal{C}$ when $\mathbf{K}$ ranges over some class of methods. Recall that $\mathcal{C}_{s,k,p}$ (resp., $\mathcal{C}^I_{s,k,p}$) denotes the optimal value of $\mathcal{C}$ over all explicit (resp., implicit) methods of order at least p with at most s stages and k steps. Since Runge-Kutta methods are 1-step methods, optimal values of $\mathcal{C}$ for explicit (resp., implicit) Runge-Kutta methods will be denoted by $\mathcal{C}_{s,1,p}$ (resp., $\mathcal{C}^{\text{I}}_{s,1,p}$) when $\mathbf{K}$ is permitted to be any explicit (resp., implicit) Runge–Kutta method with at most s stages and at least order p.

An s-stage Runge–Kutta method applied to a system of N ODEs typically requires the solution of a system of sN equations. When the system results from the semi-discretization of a system of nonlinear PDEs, N is typically very large and the system of ODEs is nonlinear, making the solution of this system very expensive. Using a transformation involving the Jordan form of $\mathbf{A}$, the amount of work can be reduced [11]. This is especially efficient for *singly implicit* (SIRK) methods (those methods for which $\mathbf{A}$ has only one distinct eigenvalue), because the necessary matrix factorizations can be reused. On the other hand, *diagonally implicit* (DIRK) methods, for which $\mathbf{A}$ is lower triangular, can be implemented efficiently without transforming to the Jordan form of $\mathbf{A}$. The class of *singly diagonally implicit* (SDIRK) methods, which are both singly implicit and diagonally

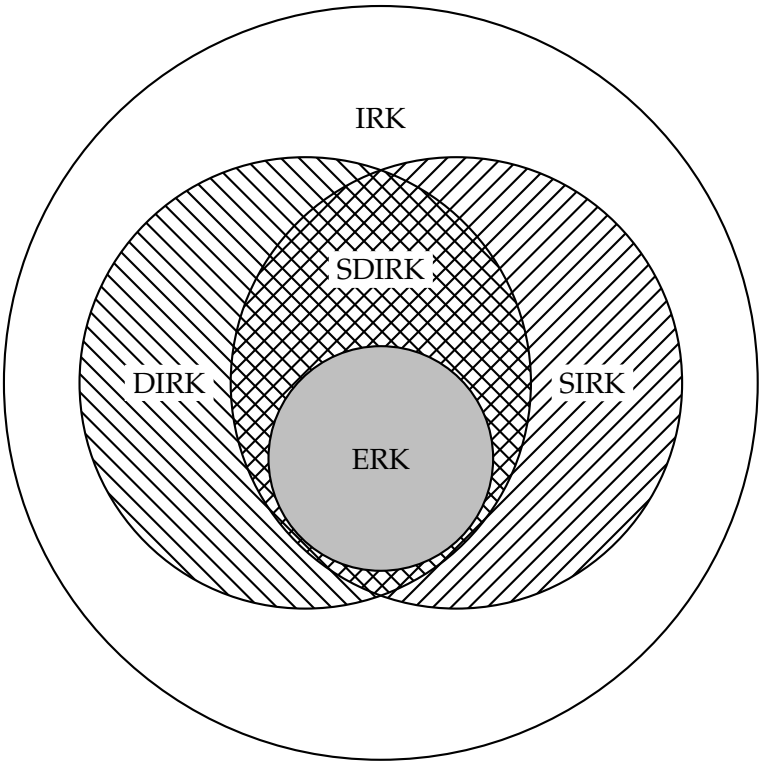


Figure 6.1: Diagram of important classes of Runge-Kutta methods: IRK=Implicit; DIRK=Diagonally Implicit; SIRK=Singly Implicit; SDIRK=Singly Diagonally Implicit; ERK=Explicit.

implicit (i.e., $\mathbf{A}$ is lower triangular with all diagonal entries identical), incorporate both of these advantages. Note that in the literature the term diagonally implicit has sometimes been used to mean singly diagonally implicit. We use $\mathcal{C}^{\mathrm{DI}}_{s,1,p}$, $\mathcal{C}^{\mathrm{SI}}_{s,1,p}$, and $\mathcal{C}^{\mathrm{SDI}}_{s,1,p}$ to denote the optimal value of $\mathcal{C}$ over each of the respective classes of DIRK, SIRK, and SDIRK methods. Note that for a given s and p, these three quantities are each bounded by $R^{\mathrm{I}}_{s,1,p}$. For details on efficient implementation of implicit Runge–Kutta methods see, e.g., [27].

We recall from Theorem 3.4.1 that $\mathcal{C}(\mathbf{K})$ is finite for all methods of higher than first order.

The following result, from [76, Theorem 4.2], provides lower bounds for the coefficients in our numerical searches.

Result 6.1.1. *Any irreducible Runge–Kutta method with positive radius of absolute monotonicity $\mathcal{C} > 0$, must have all non-negative coefficients $\mathbf{A} \geq 0$ and positive weights $\boldsymbol{b} > 0$.*

Result 6.1.1 implies restrictions on the order and stage order of SSP Runge-Kutta methods. The stage order is a lower bound on the order of convergence when a method is applied to arbitrarily stiff problems. Thus low stage order may lead to slow convergence (i.e., *order reduction*) when computing solutions of stiff ODEs. The stage order is given by the largest integer $\tilde{p}$ such that the simplifying assumptions $B(\tilde{p}), C(\tilde{p})$ hold, where [27]:

$$B(\xi): \quad \sum_{j=1}^{s} b_j c_j^{k-1} = \frac{1}{k}, \qquad (1 \leq k \leq \xi), \tag{6.1a}$$

$$C(\xi): \quad \sum_{j=1}^{s} a_{ij} c_j^{k-1} = \frac{1}{k} c_i^k, \qquad (1 \leq k \leq \xi). \tag{6.1b}$$

Result 6.1.2. *[76, Lemma 8.6] A Runge–Kutta method with weights $\boldsymbol{b} > 0$ must have stage order $\tilde{p} \geq \left\lfloor \frac{p-1}{2} \right\rfloor$.*

Combining Results 6.1.1 and 6.1.2, we have that

$$\mathcal{C} > 0 \implies p \leq 2\tilde{p} + 2. \tag{6.2}$$

We next review restrictions on the stage order of certain classes of Runge-Kutta methods.

Result 6.1.3. *[76, Theorem 8.5] A Runge–Kutta method with non-negative coefficients $\mathbf{A} \geq 0$ must have stage order $\tilde{p} \leq 2$. If $\tilde{p} = 2$, then $\mathbf{A}$ must have a zero row.*

When dealing with singly diagonally implicit methods or explicit methods, stage order is limited whether or not one requires non-negative coefficients [76, 26, 27]:

Result 6.1.4. *The stage order of a singly diagonally implicit (or explicit) Runge–Kutta method is at most* $\tilde{p} = 1$.

For SSP methods, the stage order restriction leads to restrictions on the classical order as well. Combining Results 6.1.1, 6.1.3, 6.1.2, and 6.1.4, we obtain:

Result 6.1.5. *(see also [76, Corollary 8.7]) Any irreducible Runge–Kutta method with* $\mathcal{C} > 0$ *has order* $p \leq 6$ ($p \leq 4$ *if it is explicit or singly diagonally implicit). Furthermore, if* $p \geq 5$, *then* $\mathbf{A}$ *has a zero row.*

We next give a general result on the order of DIRK and SIRK methods.

Result 6.1.6. *The order of an* s*-stage SIRK or DIRK method is at most* $s + 1$.

Proof. For a given s-stage SIRK or DIRK method, let ψ denote the stability function. For both classes of methods, ψ is a rational function with numerator of degree s and only real poles. Such a function approximates the exponential function to order at most $s + 1$ [27, Theorem 3.5.11]. □

In the following result, $\Pi_{s,p}$ denotes the set of all polynomials ψ of degree less than or equal to s satisfying $\psi(x) = \exp(x) + \mathcal{O}(x^{p+1})$ as $x \to 0$.

Corollary 6.1.7. *For SIRK methods with* $p \geq 5$,

$$\mathcal{C}^{\mathrm{SI}}_{s,1,p} \leq R_{s,1,p}.$$

Proof. For SIRK methods with $p \geq 5$, Result 6.1.5 implies that all eigenvalues of $\mathbf{A}$ must be zero, hence the stability function ψ must be a polynomial. Combined with (3.62), this proves the inequality. □

Corollary 6.1.7 implies that for s-stage SIRK methods of order $p \geq 5$, $\mathcal{C}$ is bounded by the optimal threshold factor of s-stage explicit Runge–Kutta methods of the same order (see Chapter 4 for values of these optimal coefficients).

6.2 Formulation of the Optimization Problem

Extensive efforts have been made to find optimal explicit SSP Runge-Kutta methods both by analysis and numerical search [76, 45, 46, 111, 112, 96]. Except for [76], all of these efforts formulated the optimization problem using the Shu-Osher form. While this allows the inequality constraints to be written as linear constraints, it leads to a large number of decision variables. It has been pointed out in [33] that by using the conditions for

absolute monotonicity, the problem can be formulated in terms of the Butcher array only, reducing the number of variables by half and simplifying dramatically the form of the order conditions. We adopt this latter formulation, which can be applied to implicit methods as well:

$$\begin{aligned}
&\text{maximize } r \text{ subject to} \\
&\mathbf{K}(I + r\mathbf{A})^{-1} \geq 0 && && \text{(6.3a)} \\
&\|r\mathbf{K}(I + r\mathbf{A})^{-1}\|_\infty \leq 1 && && \text{(6.3b)} \\
&\tau_k(\mathbf{K}) = 0 && (k \leq p) && \text{(6.3c)}
\end{aligned}$$

where the matrix inequality is understood componentwise and τ_k represents the set of order conditions for order k (see Section 2.3). Additional constraints are added in order to investigate various subclasses of methods. For explicit methods, we impose

$$\mathbf{K}_{ij} = 0 \qquad (j \geq i). \tag{6.4}$$

For diagonally implicit methods, we impose

$$\mathbf{K}_{ij} = 0 \qquad (j > i). \tag{6.5}$$

For singly diagonally implicit methods, we impose

$$\mathbf{K}_{ij} = 0 \qquad (j > i) \tag{6.6}$$

$$\mathbf{K}_{jj} = \mathbf{K}_{11} \qquad (j > 1). \tag{6.7}$$

This formulation, implemented in MATLAB using a sequential quadratic programming approach (fmincon in the optimization toolbox), was used to find the methods given below.

The above problem can be reformulated (using a standard approach for converting rational constraints to polynomial constraints) as

$$\max_{\mathbf{K},\mathbf{P}} r \tag{6.8a}$$

$$\text{subject to} \quad \begin{cases} \mathbf{P} \geq 0 \\ \|\mathbf{P}\|_\infty \leq 1 \\ r\mathbf{K} = \mathbf{P}(\mathbf{I} + r\mathbf{K}), \\ \Phi_p(\mathbf{K}) = 0. \end{cases} \tag{6.8b}$$

This optimization problem has only polynomial constraints and thus is appropriate for

the BARON optimization software which requires such constraints to be able to guarantee global optimality [100].

6.3 Implicit Runge-Kutta Methods

> For implicit methods it is not known whether $p = 6$ can be achieved...
>
> *J.F.B.M. Kraaijevanger (1991)*

In this section we present numerically optimal implicit SSP Runge–Kutta methods for nonlinear systems of ODEs. These methods were found via numerical search, and in general we have no analytic proof of their optimality. In a few cases, we have employed BARON, an optimization software package that provides a numerical certificate of global optimality [100]. BARON was used to find optimal explicit SSP Runge–Kutta methods in [88, 97]. However, this process is computationally expensive and was not practical in most cases.

We applied our optimization approach to finding optimal SSP Runge–Kutta methods over other classes for which results are already known, and successfully found a solution at least as good as the previously best known solution in every case. Because our approach was able to find these previously known methods (and some improvements), we expect that many of the new methods are globally optimal (to within numerical precision).

The optimization problem for general (implicit) Runge–Kutta methods involves approximately twice as many decision variables (dimensions) as the explicit or singly diagonally implicit cases, which have previously been investigated [45, 46, 111, 112, 97, 35]. Despite the larger number of decision variables, we have been able to find numerically optimal methods even for large numbers of stages. We attribute this success to the reformulation of the optimization problem in terms of the Butcher coefficients rather than the Shu–Osher coefficients.

Because in most cases we cannot prove the optimality of the resulting methods, we use hats to denote the best value found by numerical search, e.g. $\hat{\mathcal{C}}^{\mathrm{I}}_{s,1,p}$, etc.

In comparing methods with different numbers of stages, one is usually interested in the relative time advancement per computational cost. For diagonally implicit methods, the computational cost per time-step is proportional to the number of stages; hence in this case the *effective SSP coefficient* is relevant. However, for non-DIRK methods of various s, it is much less obvious how to compare computation cost.

Since $\mathcal{C} < \infty$ for all general linear methods, one cannot hope to avoid SSP time restrictions completely by using an implicit method. Nevertheless, implicit methods may

be expected to allow larger SSP timesteps than explicit methods.

6.3.1 Optimal Methods

For implicit methods, the matrix $\mathbf{P}$ defined in (3.31) has non-zero diagonal elements. Thus it is not convenient for implementation, since y_j appears on the right hand side of the equation for y_j in this form. We thus present the coefficients in modified Shu-Osher form (see [73]):

$$y_i = \left(1 - \sum_{j=1}^{s} \lambda_{ij}\right) u^n + \sum_{j=1}^{s} \lambda_{ij} y_j + \Delta t \mu_{ij} F(t_n + c_j \Delta t, y_j), \quad (1 \leq i \leq s+1), \tag{6.9a}$$

$$u^{n+1} = y_{s+1}. \tag{6.9b}$$

We choose the coefficients so that the diagonal elements of λ_{ii} are zero. This form is a simple rearrangement and involves no loss of generality.

Second-order Methods

Optimizing over the class of all $(s \leq 11)$-stage second-order implicit Runge–Kutta methods we found that the numerically optimal methods are, remarkably, identical to the numerically optimal SDIRK methods found in [32, 35]. This result stresses the importance of the second-order SDIRK methods found in [32, 35]: they appear to be optimal not only among SDIRK methods, but also among the much larger class of all implicit Runge–Kutta methods.

These methods are most advantageously implemented in a certain modified Shu–Osher form. This is because these arrays (if chosen carefully) are more sparse. In fact, for these methods there exist modified Shu–Osher arrays that are bidiagonal. We give the general formulae here.

The numerically optimal second-order method with s stages has $\mathcal{C} = 2s$ and coefficients

$$\boldsymbol{\lambda} = \begin{bmatrix} 0 & & & \\ 1 & 0 & & \\ & 1 & \ddots & \\ & & \ddots & 0 \\ & & & 1 \end{bmatrix}, \quad \boldsymbol{\mu} = \begin{bmatrix} \frac{1}{2s} & & & \\ \frac{1}{2s} & \frac{1}{2s} & & \\ & \frac{1}{2s} & \ddots & \\ & & \ddots & \frac{1}{2s} \\ & & & \frac{1}{2s} \end{bmatrix}. \tag{6.10}$$

The one-stage method of this class is the implicit midpoint rule, while the s-stage

method is equivalent to s successive applications of the implicit midpoint rule (as was observed in [32]). Thus these methods inherit the desirable properties of the implicit midpoint rule, such as algebraic stability and A-stability [48]. Of course, since they all have the same effective SSP coefficient $\mathcal{C}/s = 2$, they are all essentially equivalent.

The one-stage method is the unique method with $s = 1, p = 2$ and hence is optimal. The two-stage method achieves the maximum radius of absolute monotonicity for rational functions that approximate the exponential to second order with numerator and denominator of degree at most two, hence it is optimal to within numerical precision [118, 68, 35]. In addition to duplicating these optimality results, BARON was used to numerically prove that the $s = 3$ scheme is globally optimal, verifying [35, Conjecture 3.1] for the case $s = 3$. The $s = 1$ and $s = 2$ cases required only several seconds but the $s = 3$ case took much longer: approximately 11 hours of CPU time on an Athlon MP 2800+ processor.

While the remaining second order methods have not been proven optimal, it appears likely that they may be (see Conjecture 6.5.1).

Third-order Methods

The numerically optimal third-order implicit Runge–Kutta methods with $s \geq 2$ stages are also singly diagonally implicit and identical to the numerically optimal SDIRK methods found in [32, 35], which have $\mathcal{C} = s - 1 + \sqrt{s^2 - 1}$. Once again, these results indicate that the methods found in [32, 35] are likely optimal over the entire class of implicit Runge–Kutta methods.

These methods may also be implemented using bidiagonal modified Shu–Osher arrays. For $p = 3$ and $s \geq 2$ the numerically optimal methods have coefficients

$$\boldsymbol{\lambda} = \begin{bmatrix} 0 & & & \\ 1 & \ddots & & \\ & \ddots & 0 & \\ & & 1 & 0 \\ & & & \lambda_{s+1,s} \end{bmatrix}, \quad \boldsymbol{\mu} = \begin{bmatrix} \mu_{11} & & & \\ \mu_{21} & \ddots & & \\ & \ddots & \mu_{11} & \\ & & \mu_{21} & \mu_{11} \\ & & & \mu_{s+1,s} \end{bmatrix}, \tag{6.11a}$$

where

$$\mu_{11} = \frac{1}{2}\left(1 - \sqrt{\frac{s-1}{s+1}}\right), \qquad \mu_{21} = \frac{1}{2}\left(\sqrt{\frac{s+1}{s-1}} - 1\right), \tag{6.11b}$$

$$\mu_{s+1,s} = \frac{s+1}{s(s+1+\sqrt{s^2-1})}, \qquad \lambda_{s+1,s} = \frac{(s+1)(s-1+\sqrt{s^2-1})}{s(s+1+\sqrt{s^2-1})}. \tag{6.11c}$$

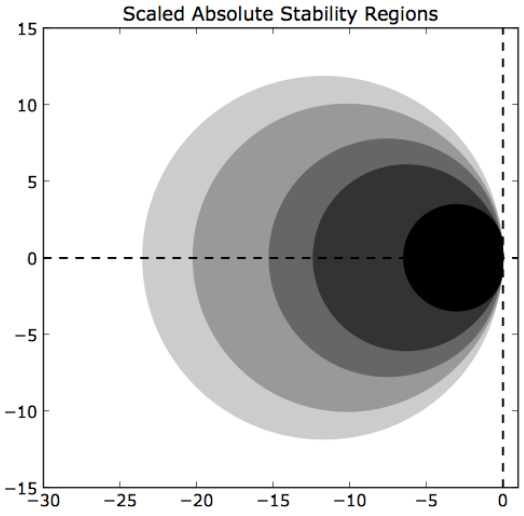


Figure 6.2: Scaled absolute stability regions of optimal third-order implicit SSP Runge-Kutta methods with two to six stages. The larger scaled stability regions correspond to the methods with more stages.

The two-stage method in this family achieves the maximum value of $R(\psi)$ found in [118] for ψ in the set of third-order rational approximations to the exponential with numerator and denominator of degree at most 2. Since the corresponding one-parameter optimization problem is easy to solve, then (since $C \leq R$), the method is clearly optimal to within numerical precision. BARON was used to numerically prove global optimality for the three-stage method (6.11), requiring about 12 hours of CPU time on an Athlon MP 2800+ processor. Note that this verifies [35, Conjecture 3.2] for the case $s = 3$.

The scaled absolute stability regions (scaled by the number of stages) of the methods with two to six stages are plotted in Figure 6.2.

While the remaining third order methods (those with $s \geq 4$) have not been proven optimal, we are again led to suspect that they may be, because of the nature of the optimal methods and the convergent behavior of the optimization algorithm for these cases (see Conjecture 6.5.2).

Fourth-order Methods

Based on the above results, one might suspect that all optimal implicit SSP methods are singly diagonally implicit. In fact, this cannot hold for $p \geq 5$ since in that case $\mathbf{A}$ must

Table 6.1: SSP coefficients and effective SSP coefficients of numerically optimal fourth-order implicit Runge–Kutta methods and SDIRK methods.

s	$\mathcal{C}^{\mathrm{I}}_{s,1,4}$	$\mathcal{C}^{\mathrm{SDI}}_{s,1,4}$	$\mathcal{C}^{\mathrm{I}}_{s,1,4}/s$	$\mathcal{C}^{\mathrm{SDI}}_{s,1,4}/s$
3	2.05	1.76	0.68	0.59
4	4.42	4.21	1.11	1.05
5	6.04	5.75	1.21	1.15
6	7.80	7.55	1.30	1.26
7	9.19	8.67	1.31	1.24
8	10.67	10.27	1.33	1.28
9	12.04		1.34	
10	13.64		1.36	
11	15.18		1.38	

have a zero row (see Result 6 above). The numerically optimal methods of fourth-order are not singly diagonally implicit either; however, all numerically optimal fourth-order methods we have found are diagonally implicit.

The unique two-stage fourth-order Runge–Kutta method has a negative coefficient and so is not SSP. Thus we begin our search with three-stage methods. We list the SSP coefficients and effective SSP coefficients of the numerically optimal methods in Table 6.1. For comparison, the table also lists the effective SSP coefficients of the numerically optimal SDIRK methods found in [35]. Our numerically optimal DIRK methods have larger SSP coefficients in every case. Furthermore, they have representations that allow for very efficient implementation in terms of storage. However, SDIRK methods may be implemented in a potentially more efficient (in terms of computation) manner than DIRK methods. An exact evaluation of the relative efficiencies of these methods is beyond the scope of this work. The coefficients of the methods are given in Appendix A.

BARON was run on the three-stage fourth-order case but was unable to prove the global optimality of the resulting method using 14 days of CPU time on an Athlon MP 2800+ processor. However, during that time BARON did establish an upper bound $\mathcal{C}^{\mathrm{I}}_{3,1,4} \leq 3.234$. BARON was not run on any other fourth-order cases, nor was it used for $p = 5$ or $p = 6$.

Although none of the fourth-order methods are proven optimal, it appears that they may be optimal. This is again because the optimization algorithm is able to converge to these methods from a range of random initial guesses, and because very many of the inequality constraints are satisfied exactly for these methods. Additionally, we were able to recover all of the optimal fourth-order SDIRK methods of [35] by restricting our search to the space of SDIRK methods.

Fifth- and Sixth-order Methods

We have found fifth- and sixth-order SSP methods with up to eleven stages. Two sets of numerical searches were conducted, corresponding to optimization over the full class of implicit Runge–Kutta methods and optimization over the subclass of diagonally implicit Runge–Kutta methods. More CPU time was devoted to the first set of searches; however, in most cases the best methods we were able to find resulted from the searches restricted to DIRK methods. Furthermore, when searching over fully implicit methods, in every case for which the optimization algorithm successfully converged to a (local) optimum, the resulting method was diagonally implicit. Thus all of the numerically optimal methods found are diagonally implicit.

Because better results were obtained in many cases by searching over a strictly smaller class of methods, it seems likely that the methods found are not globally optimal. This is not surprising because the optimization problems involved are highly nonlinear with many variables, many constraints, and multiple local optima. The application of more sophisticated software to this problem is an area of future research. Nevertheless, the observation that all converged solutions correspond to DIRK methods leads us to believe that the globally optimal methods are likely to be DIRK methods.

Typically, an optimization algorithm may be expected to fail for sufficiently large problems (in our case, sufficiently large values of s). However, we found that the cases of relatively small s and large p (i.e., $p = 5$ and $s < 6$ or $p = 6$ and $s < 9$) also posed great difficulty. This may be because the feasible set in these cases is extremely small. The methods found in these cases were found indirectly by searching for methods with more stages and observing that the optimization algorithm converged to a reducible method. Due to the high nonlinearity of the problem for $p \geq 5$, we found it helpful to explicitly limit the step sizes used by fmincon in the final steps of optimization.

Fifth-order Methods

Three stages Using the W transformation [27] we find the one parameter family of three-stage, fifth-order methods

$$\mathbf{A} = \begin{bmatrix} \frac{5}{36} + \frac{2}{9}\gamma & \frac{5}{36} + \frac{1}{24}\sqrt{15} - \frac{5}{18}\gamma & \frac{5}{36} + \frac{1}{30}\sqrt{15} + \frac{2}{9}\gamma \\ \frac{2}{9} - \frac{1}{15}\sqrt{15} - \frac{4}{9}\gamma & \frac{2}{9} + \frac{5}{9}\gamma & \frac{2}{9} + \frac{1}{15}\sqrt{15} - \frac{4}{9}\gamma \\ \frac{5}{36} - \frac{1}{30}\sqrt{15} + \frac{2}{9}\gamma & \frac{5}{36} - \frac{1}{24}\sqrt{15} - \frac{5}{18}\gamma & \frac{5}{36} + \frac{2}{9}\gamma \end{bmatrix}.$$

It is impossible to choose γ so that a_{21} and a_{31} are simultaneously nonnegative, so there are no SSP methods in this class.

Table 6.2: Comparison of SSP coefficients of numerically optimal fifth-order IRK methods with theoretical upper bounds on SSP coefficients of fifth-order SIRK methods.

s	$\mathcal{C}^{\mathrm{I}}_{s,1,5}$	$\mathcal{C}^{\mathrm{SI}}_{s,1,5}$ (upper bound)	$\mathcal{C}^{\mathrm{I}}_{s,1,5}/s$	$\mathcal{C}^{\mathrm{SI}}_{s,1,5}/s$ (upper bound)
4	1.14		0.29	
5	3.19	1.00	0.64	0.20
6	4.97	2.00	0.83	0.33
7	6.21	2.65	0.89	0.38
8	7.56	3.37	0.94	0.42
9	8.90	4.10	0.99	0.46
10	10.13	4.83	1.01	0.48
11	11.33	5.52	1.03	0.50

Four to Eleven stages We list the time-step coefficients and effective SSP coefficients of the numerically optimal fifth order implicit Runge–Kutta methods for $4 \leq s \leq 11$ in Table 6.2. It turns out that all of these methods are diagonally implicit.

For comparison, we also list the upper bounds on effective SSP coefficients of SIRK methods in these classes implied by combining Corollary 6.1.7 with [69, Table 2.1]. Our numerically optimal IRK methods have larger effective SSP coefficients in every case. The coefficients of the methods are given in Appendix A.

Sixth-order Methods

Kraaijevanger [76] proved the bound $p \leq 6$ (see Result 6.1.5 above) and presented a single fifth-order method, leaving the existence of sixth-order methods as an open problem. The sixth-order methods we have found settle this problem, demonstrating that the order barrier $p \leq 6$ for implicit SSP/contractive methods is sharp.

The non-existence of three-stage SSP Runge–Kutta methods of fifth-order, proved above, implies that sixth-order SSP Runge–Kutta methods must have at least four stages. Result 6.1.6 implies that sixth-order SSP DIRK methods must have at least five stages, and Corollary 6.1.7 shows that sixth-order SSP SIRK methods require at least six stages. We were unable to find sixth-order SSP Runge–Kutta methods with fewer than six stages.

The SSP coefficients and effective SSP coefficients of the numerically optimal methods for $6 \leq s \leq 11$ are listed in Table 6.3. All of these methods are diagonally implicit. The coefficients of the methods are given in Appendix A. We were unable to find an eleven-stage method with larger *effective* SSP coefficient than that of the ten-stage method (although we did find a method with larger $\mathcal{C}$).

Table 6.3: SSP coefficients and effective SSP coefficients for numerically optimal sixth-order implicit RK methods.

s	$\mathcal{C}^{\mathrm{I}}_{s,1,6}$	$\mathcal{C}^{\mathrm{I}}_{s,1,6}/s$
6	0.18	0.030
7	0.26	0.038
8	2.25	0.28
9	5.80	0.63
10	8.10	0.81
11	8.85	0.80

6.3.2 Numerical Experiments

We begin our numerical examples with a convergence study on a linear advection problem with smooth initial conditions. We then proceed to show the importance of the threshold factor for this linear advection problem with discontinuous initial condition. Finally, the effect of the SSP coefficient is demonstrated on the nonlinear Burgers' and Buckley–Leverett equations.

The computations in Section 6.3.2 were performed with MATLAB version 7.1 on a Mac G5; those in Sections 6.3.2 and 6.3.2 were performed with MATLAB version 7.3 on x86-64 architecture. All calculations were performed in double precision. For the implicit solution of linear problems we used MATLAB's backslash operator, while for the nonlinear implicit solves we used the fsolve function with very small tolerances.

We refer to the numerically optimal methods as SSPsp where s, p are the number of stages and order, respectively. For instance, the numerically optimal eight-stage method of order five is SSP85.

Linear Advection

The prototypical hyperbolic PDE is the linear advection equation,

$$u_t + a u_x = 0, \qquad 0 \le x \le 2\pi. \tag{6.12}$$

We consider (6.12) with $a = -2\pi$, periodic boundary conditions and various initial conditions. We use a method-of-lines approach, discretizing the interval $(0, 2\pi]$ into m points $x_j = j\Delta x$, $j = 1, \ldots, m$, and then discretizing $-au_x$ with first-order upwind finite differences. We solve the resulting system using our timestepping schemes. To isolate the effect of the time-discretization error, we exclude the effect of the error associated with the spa-

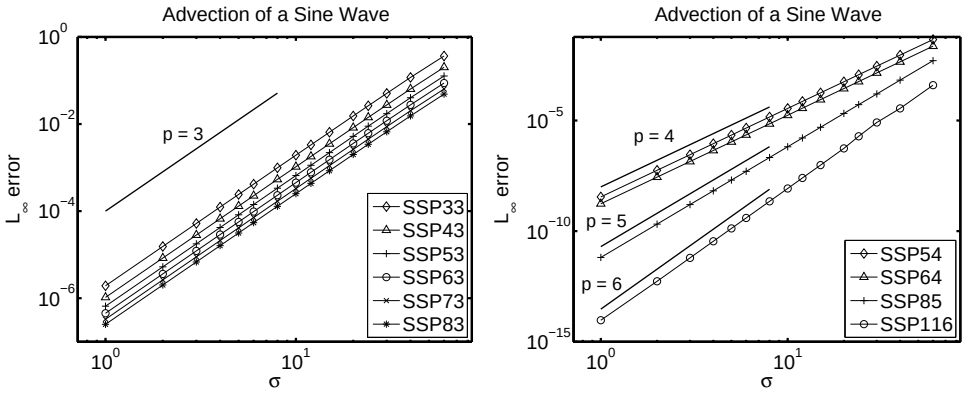


Figure 6.3: Convergence of optimal SSP IRK methods for the sine wave advection problem. All methods achieve their designed rate of convergence.

tial discretization by comparing the numerical solution to the exact solution of the ODE system, rather than to the exact solution of the PDE (6.12). In lieu of the exact solution we use a very accurate numerical solution obtained using MATLAB's ode45 solver with minimal tolerances (AbsTol $= 1 \times 10^{-14}$, RelTol $= 1 \times 10^{-13}$).

Figure 6.3 shows a convergence study for various numerically optimal schemes for the problem (6.12) with $m = 120$ points in space and smooth initial data

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

advected until final time $t_f = 1$. Here σ indicates the relative size of the timestep: $\Delta t = \sigma \Delta t_{\text{FE}}$. The results show that all the methods achieve their design order.

Now consider the advection equation with discontinuous initial data

$$u(x, 0) = \begin{cases} 1 & \text{if } \frac{\pi}{2} \leq x \leq \frac{3\pi}{2}, \\ 0 & \text{otherwise.} \end{cases} \tag{6.13}$$

Figure 6.4 shows a convergence study for the third-order methods with $s = 3$ to $s = 8$ stages, for $t_f = 1$ using $m = 64$ points and the first-order upwinding spatial discretization. Again, the results show that all the methods achieve their design order. Finally, we note that the higher-stage methods give a smaller error for the same timestep; that is, as s increases, the error constant of the method decreases.

Figure 6.5 shows the result of solving the discontinuous advection example using the two-stage third-order method over a single timestep with $m = 200$. For this lin-

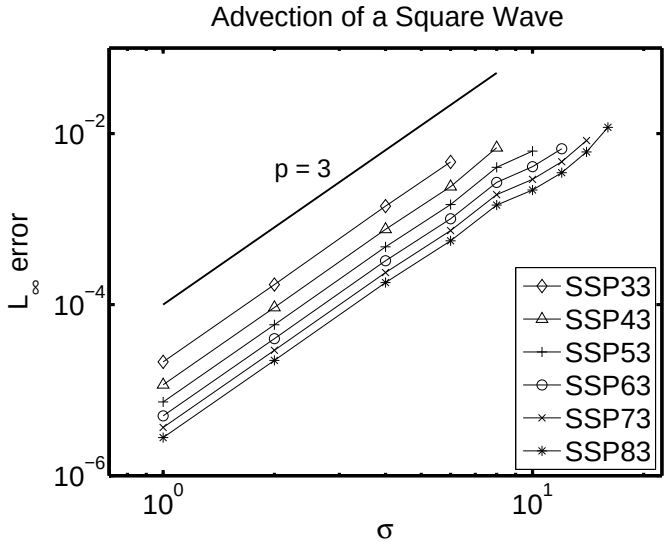


Figure 6.4: Convergence of optimal third-order SSP IRK methods for the square wave advection problem.

ear autonomous system, the theoretical monotonicity-preserving timestep bound is $\sigma \leq R(\psi) = 2.732$. We see that as the timestep is increased, the line steepens and forms a small step, which becomes an oscillation as the stability limit is exceeded, and worsens as the timestep is raised further.

Burgers' Equation

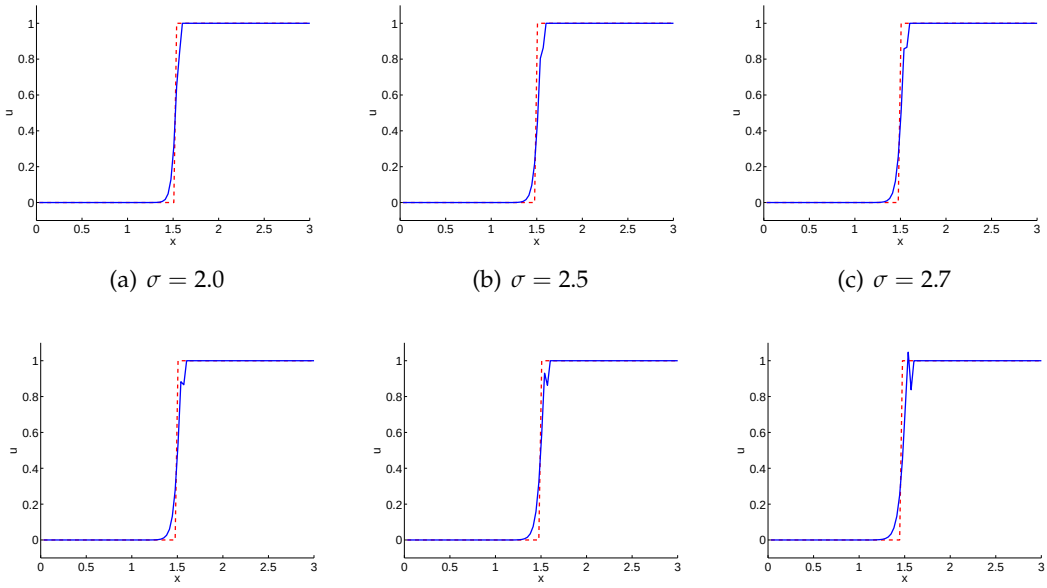
In this section we consider the inviscid Burgers' equation, which consists of the conservation law

$$u_t + f(u)_x = 0 \tag{6.14}$$

with flux function $f(u) = \frac{1}{2}u^2$. We take initial conditions $u(0,x) = \frac{1}{2} - \frac{1}{4}\sin(\pi x)$ on the periodic domain $x \in [0,2)$. The solution is right-travelling and over time develops a shock. We discretize $-f(u)_x$ using the conservative upwind approximation

$$-f(u)_x \approx -\frac{1}{\Delta x}\left(f(u_i) - f(u_{i-1})\right). \tag{6.15}$$

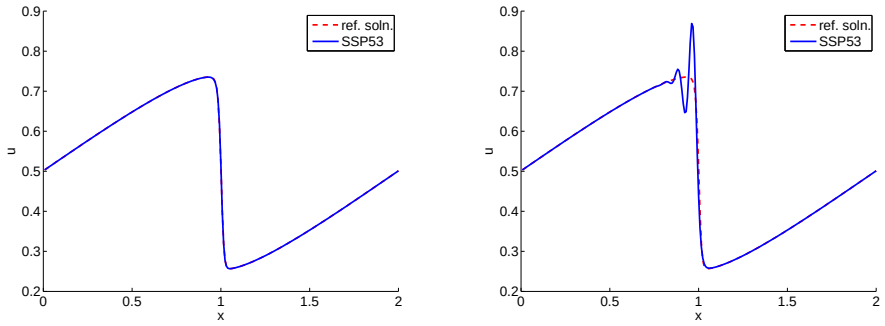
with $m = 256$ points in space and integrate to time $t_f = 2$. The convergence study in Figure 6.7 shows that the fourth-, fifth- and sixth-order s-stage methods achieve their respective orders of convergence when compared to a temporally very refined solution of



(a) $\sigma = 2.0$ (b) $\sigma = 2.5$ (c) $\sigma = 2.7$

(d) $\sigma = 2.8$ (e) $\sigma = 3.0$ (f) $\sigma = 3.6$

Figure 6.5: Comparison of square wave advection using a range of CFL numbers. The solution obtained with the optimal implicit two-stage third-order method ($R(\psi) = 2.732$) is plotted after one timestep.



(a) $\sigma = 8$ (b) $\sigma = 32$

Figure 6.6: Comparison of Burgers evolution of a sine wave for CFL numbers below and above the SSP limit. The solution is obtained with the optimal five-stage third-order implicit Runge-Kutta method ($\mathcal{C} = 8.90$).

the discretized system.

Figure 6.6 shows that when the timestep is below the stability limit no oscillations appear, but when the stability limit is violated, oscillations are observed.

Buckley–Leverett Equation

The Buckley–Leverett equation is a model for two-phase flow through porous media [82] and consists of the conservation law (6.14) with flux function

$$f(u) = \frac{u^2}{u^2 + a(1-u)^2}.$$

We take $a = \frac{1}{3}$ and initial conditions

$$u(x,0) = \begin{cases} 1 & \text{if } x \leq \frac{1}{2}, \\ 0 & \text{otherwise,} \end{cases}$$

on $x \in [0,1)$ with periodic boundary conditions. Our spatial discretization uses $m = 100$ points and we use the conservative scheme with Koren limiter used in [35] and [61, Section III.1]. The nonlinear system of equations for each stage of the Runge–Kutta method is solved with MATLAB's fsolve, with the Jacobian approximated [61] by that of the first-order upwind discretization (6.15). We compute the solution for $n = \left\lceil \frac{1}{8} \frac{1}{\Delta t} \right\rceil$ timesteps.

For this problem, as in [35], we find that the forward Euler solution is total variation

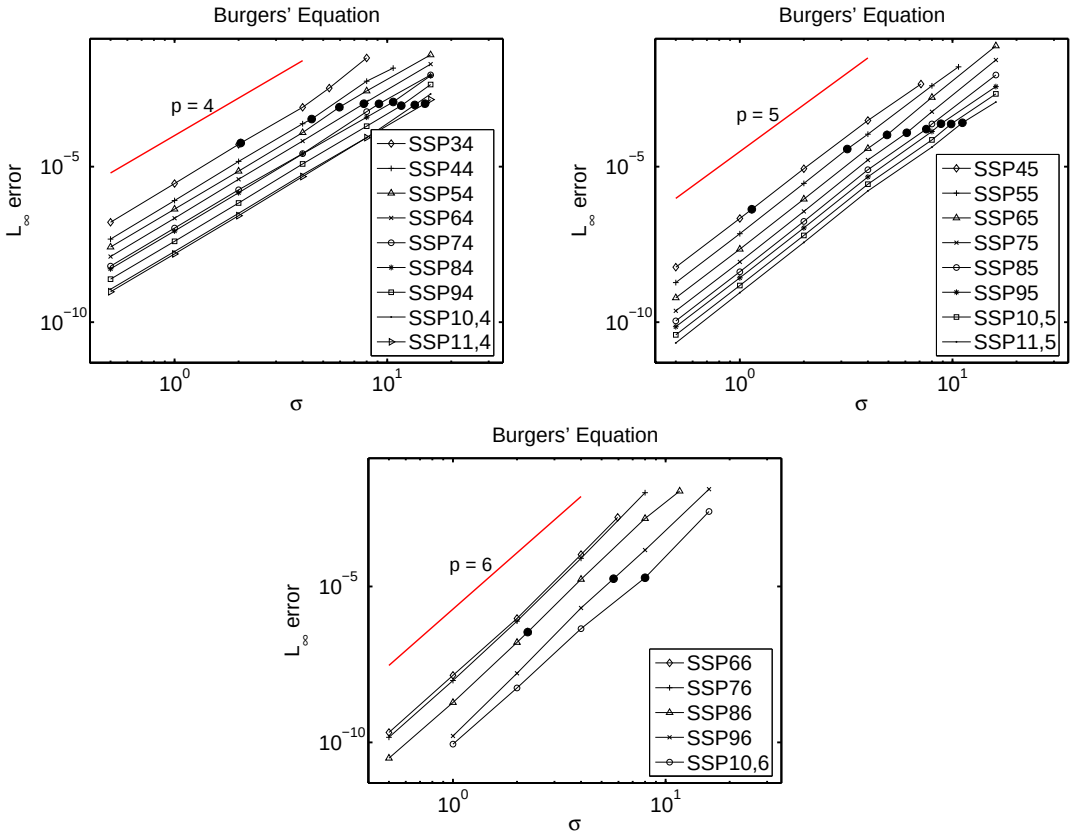
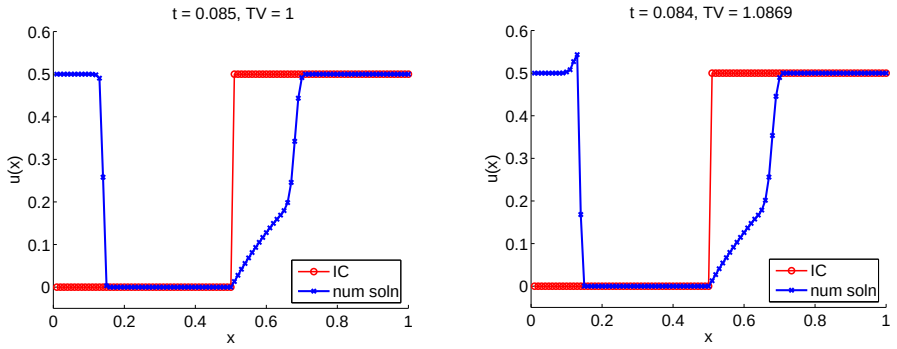


Figure 6.7: Convergence of optimal implicit SSP RK methods for the Burgers' sine wave problem. The solid circles indicate $\sigma = \mathcal{C}$ for each scheme.



(a) $\sigma = 2$ (b) $\sigma = 2.4$

Figure 6.8: Comparison of solutions of the Buckley-Leverett equation for CFL numbers below and above the SSP limit. The optimal second-order, one-stage implicit SSP RK method is used ($\mathcal{C} = 2$).

diminishing (TVD) for $\Delta t \leq \Delta t_{\text{FE}} = 0.0025$. Figure 6.8 shows typical solutions for the SSP(1,2) scheme with timestep $\Delta t = \sigma \Delta t_{\text{FE}}$. Table 6.4 compares the SSP coefficient $\mathcal{C}$ with $\sigma_{\text{BL}} = \Delta t_{\text{RK}} / \Delta t_{\text{FE}}$, where Δt_{RK} is the largest *observed* timestep for which the numerical solution obtained with the Runge–Kutta method is TVD. We note that, for each method, the value of σ_{BL} is greater than the SSP coefficient. In fact, at least for either lower order p or high number of stages s, the values are in good correspondence. For $p = 2$ and $p = 3$, our results agree with those of [35].

6.4 Explicit Runge-Kutta Methods

Among the Runge-Kutta methods originally introduced in [105], two have been very widely implemented: the two stage, second order method of Heun, and a three stage, third order method originally proposed by Fehlberg (though not in the context of SSP methods). These methods achieve the largest SSP coefficient, or relative nonlinearly stable timestep, among all two-stage, second order and three-stage, third order methods, respectively. The non-existence of four-stage, fourth order methods was suggested in [105] and proved in [45] (cf. [76]).

Extensive efforts have been made to find more efficient explicit SSP Runge-Kutta methods by allowing more stages [76, 45, 46, 111, 112, 97]. Although increasingly efficient methods have been found, most require increased storage and have not been widely used. A few studies have considered optimal low-storage SSP methods [45, 46, 97].

Efforts to find SSP methods have focused on finding the method with maximal SSP

Table 6.4: Comparison of the SSP coefficient $\mathcal{C}$ and the maximum TVD timestep σ_{BL} for the Buckley-Leverett example for optimal implicit SSP RK methods.

	$\mathcal{C}$					σ_{BL}				
s \ p	2	3	4	5	6	2	3	4	5	6
1	2	-	-	-	-	2.03	-	-	-	-
2	4	2.73	-	-	-	4.08	3.68	-	-	-
3	6	4.83	2.05	-	-	6.11	5.39	4.01	-	-
4	8	6.87	4.42	1.14		8.17	7.13	5.59	4.04	
5	10	8.90	6.04	3.21		10.25	9.06	6.46	4.91	
6	12	10.92	7.80	4.97	0.18	12.33	11.18	7.98	6.92	4.83
7	14	12.93	9.19	6.21	0.26	14.43	13.33	9.31	9.15	5.14
8	16	14.94	10.67	7.56	2.25	16.53	15.36	11.42	8.81	5.66
9	18	16.94	12.04	8.90	5.80	18.60	17.52	15.01	11.04	7.91
10	20	18.95	13.64	10.13	8.10	20.66	19.65	13.84	12.65	10.80
11	22	20.95	15.18	11.33	8.85	22.77	21.44	15.95	14.08	11.82

coefficient for a prescribed order, number of stages, and (sometimes) number of memory registers. However, the number of stages is only important as it affects the computational efficiency and memory requirements of the method. We therefore focus on methods that are optimal over all stages in terms of efficiency and memory. We will see that in some cases the optimal method is obtained as the limit of a family of methods, parameterized by stage number.

6.4.1 Memory Considerations

We now briefly discuss some concepts regarding memory usage for explicit Runge-Kutta methods. These will be explored in much greater detail in Chapter 7.

A naive implementation of an s-stage Runge-Kutta method requires $s + 1$ memory registers. However, if certain algebraic relations between the coefficients are satisfied, the method may be implemented with fewer registers. Two such types of relations have been exploited in the literature [124, 66]. The resulting two types of low-storage methods make different important assumptions on the manner in which F is evaluated.

Consider two storage registers, q1 and q2, each of size N, where N denotes the number of ODEs to be integrated. The low-storage methods of Williamson [124] assume that it is possible to make assignments of the form

```
q1 := q1 + F(q2),
```

without allocating (much) additional storage for the evaluation of `F(q2)`. As noted in [66], this requires that the evaluation be done in 'piecemeal fashion'. This is natural, for

instance, if F corresponds to a spatial discretization of a PDE where the spatial stencil is localized, which is usually the case for semi-discretizations of hyperbolic PDEs.

The low-storage methods of van der Houwen type [66] make instead the assumption that it is possible to make assignments of the form

```
q1 := F(q1),
```

again without significant additional storage. This is reasonable for compressible Navier-Stokes calculations [66], and also when F corresponds to a spatial discretization of a PDE where the spatial stencil is localized.

In the present work we give a new class of low-storage methods; the low-storage methods presented here require the assumption that it is possible to make assignments of the form

```
q1 := q1 + F(q1),
```

without employing a second storage register. This assumption implies the assumptions necessary for implementation of Williamson and van der Houwen methods; hence, the class of semi-discretizations to which it is applicable is smaller. However, it is still reasonable for spatial discretizations with local stencils. While it requires careful programming, especially for problems in two or three dimensions, when memory considerations are important this may be worth the effort.

In the following, a method requiring m storage registers of length N is referred to as an mS method. Sometimes it is necessary to retain the value of the solution at the previous timestep, usually in order to restart the step if some accuracy or stability requirement is violated. While most low-storage methods will require an additional register in this case, some will not. Methods that do not are denoted by mS*.

The first order accurate forward Euler method has $\mathcal{C}_{\text{eff}} = 1$ and can be implemented in 1S* fashion; hence consideration of first order SSP methods with more stages is superfluous. Since explicit SSP Runge-Kutta methods have order at most four [99], it remains to consider methods of order two through four.

6.4.2 Optimal Methods

Note that $\mathcal{C} \leq R$ by definition, so the values of $\mathbf{R}_{s,1,p}(\psi)$ in Table 4.1 are upper bounds on the value of $\mathcal{C}$ for methods with a given order p and number of stages s. We will see that, in many cases, this bound can be achieved. Surprisingly, it is even possible to find methods that achieve this bound and that can be implemented in low-storage form.

Since $R \leq s$, it follows that $\mathcal{C}_{\text{eff}} \leq 1$ (in fact, $\mathcal{C}_{\text{eff}} < 1$ for methods of greater than first order). Also, any method that uses only a single memory register must consist simply

of repeated forward Euler steps and therefore cannot be more than first order accurate. Thus an ideal higher order method would have $m = 2$ and $\mathcal{C}_{\text{eff}}$ as close as possible to 1.

Second Order Methods

2S* Methods. Optimal second order methods with $\mathcal{C}_{\text{eff}}$ arbitrarily close to 1 were found in [76], and later independently in [111]. The s-stage method in this family has SSP coefficient $s - 1$, hence $\mathcal{C}_{\text{eff}} = \frac{s-1}{s}$. The nonzero entries of the low-storage form for the s-stage method are

$$\beta_{i,i-1} = \begin{cases} \frac{1}{s-1} & 2 \le i \le s \\ \frac{1}{s} & i = s+1 \end{cases} \tag{6.16a}$$

$$\alpha_{i,i-1} = \begin{cases} 1 & 2 \le i \le s \\ \frac{s-1}{s} & i = s+1 \end{cases} \tag{6.16b}$$

$$\alpha_{s+1,0} = \frac{1}{s}. \tag{6.16c}$$

The abscissas are

$$c_i = \frac{i-1}{s-1} \quad (1 \le i \le s). \tag{6.17}$$

Note that these methods do not require a third register even if the previous timestep must be retained. As far as we know, no low-storage implementations of this type have been proposed before, for any Runge-Kutta method. Because the storage costs do not increase with s while the effective SSP coefficient does, there seems to be little drawback to using these methods with large values of s. However, we are not aware of any implementation of these methods for $s > 4$. This may be because the low-storage property of these methods has not been pointed out previously, probably because they cannot be written in Williamson or van der Houwen form, for $s > 2$.

Since second order discretizations are often considered to be the most efficient for compressible flow problems involving shocks (the same problems that originally motivated development of SSP methods), these methods should prove to be very useful. Note that the large s members of this family are approximately twice as efficient as the two-stage method, which is the most commonly used.

Here and below, low-storage implementations are given in MATLAB code; if implemented exactly in this form in MATLAB, an additional memory register will be used for temporary storage during evaluation of F. To avoid this, the code should be implemented in a manner such that the argument of F is passed by reference, rather than by copy. The storage registers in the pseudo-code are denoted by q_1 and q_2. A low-storage implemen-

tation of (6.16) is given in Pseudo-code 1.

```
q1 = u; q2=u;
for i=1:s-1
  q1 = q1 + dt*F(q1)/(s-1);
end
q1 = ( (s-1)*q1 + q2 + dt*F(q1) )/s;
u=q1;
```

Pseudo-code 1: Low-storage implementation of the optimal second order methods.

Third Order Methods

2S* Methods. The three- and four-stage third order SSP Runge-Kutta methods, originally reported in [105] and [76], respectively, can be implemented with just two memory registers, even if the previous timestep must be retained. This is possible because, like the second order methods above, the only nonzero coefficients in the Shu-Osher arrays of these methods are in the first column of $\boldsymbol{\alpha}$ and the first subdiagonal of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ Since the four-stage method is 50% more efficient and requires the same amount of memory, it seems always preferable to the three-stage method. By allowing more than four stages, we found methods of this type with larger SSP coefficient; however, the number of stages required resulted in every case in an effective SSP coefficient smaller than that of the four-stage method ($\mathcal{C}_{\text{eff}} = 1/2$).

2S Methods. We now consider methods that can be implemented with only two registers, assuming the solution at the previous timestep can be discarded. Of course, they may be implemented with three registers if the previous timestep solution needs to be retained.

Low-storage 3rd order methods were found in [97]; the best 2S method has $\mathcal{C}_{\text{eff}} = 0.297$; the best 3S method has $\mathcal{C}_{\text{eff}} = 0.513$. In the limit of $s \to \infty$, the following family of methods achieves the ideal $\mathcal{C}_{\text{eff}} \to 1$.

Theorem 6.4.1. *Let $n > 2$ be an integer and let $s = n^2$. Then there exists a third order s-stage*

method with SSP coefficient $\mathcal{C} = R_{s,1,3} = n^2 - n$. *The non-zero coefficients of the method are*

$$\alpha_{i,i-1} = \begin{cases} \frac{n-1}{2n-1} & i = \frac{n(n+1)}{2} + 1 \\ 1 & \textit{otherwise} \end{cases} \tag{6.18a}$$

$$\alpha_{\frac{n(n+1)}{2}+1,\frac{(n-1)(n-2)}{2}+1} = \frac{n}{2n-1} \tag{6.18b}$$

$$\beta_{i,i-1} = \frac{\alpha_{i,i-1}}{n^2 - n}. \tag{6.18c}$$

The abscissas of the method are

$$c_i = \frac{i-1}{n^2-n}, \qquad \textit{for } 1 \le i \le (n^2+n)/2 \tag{6.19a}$$

$$c_i = \frac{i-n-1}{n^2-n} \qquad \textit{for } (n^2+n+2)/2 \le i \le n^2. \tag{6.19b}$$

Furthermore, no third order s-stage method exists with a larger SSP coefficient.

The proof of the theorem is straightforward. The SSP coefficient is evident from the coefficients, while the satisfaction of the order conditions can be verified by forming the Butcher array and checking directly. The optimality follows from (3.62) and the fact that $R_{n^2,1,3} = n^2 - n$ [75].

Like the second order family (6.16) above, this family of methods achieves effective SSP coefficients arbitrarily close to one while using only two memory registers. Also, like the family (6.16), and unlike most known optimal third order SSP methods, the coefficients are simple rational numbers. Note that the four-stage 2S* method discussed above is the first member of this family. A low-storage implementation of (6.18) is given in Pseudo-code 2.

Remark 6.4.1. *The family (6.18) was not found by numerical search; we have discovered that, for each value of* $s \le 15$, *there exists at least a one-parameter family of third order methods with* $\mathcal{C} = R_{s,1,3}$; *hence the particular methods (6.18) are unlikely to be found by numerical search (much less the low-storage implementations). We were led to these methods by the discovery of the remarkable ten-stage method of order four, discussed in the next section.*

Fourth Order Methods

No explicit fourth order method with four stages has $c > 0$ [76, 45]. The optimal five stage method was found in [76] and again independently in [111]; this method has $\mathcal{C}_{\text{eff}} = 0.302$, More efficient methods with up to eight stages were found in [112, 97, 88]; the most efficient (eight-stage) method has $\mathcal{C}_{\text{eff}} = 0.518$ and can be implemented in 3S fashion.

```
s=n^2; r=s-n; q1=u;
for i=1:(n-1)*(n-2)/2
  q1 = q1 + dt*F(q1)/r;
end
q2=q1;
for i=(n-1)*(n-2)/2+1:n*(n+1)/2-1
  q1 = q1 + dt*F(q1)/r;
end
q1 = ( n*q2 + (n-1)*(q1 + dt*F(q1)/r) ) / (2*n-1);
for i=n*(n+1)/2+1:s
  q1 = q1 + dt*F(q1)/r;
end
u=q1;
```

Pseudo-code 2: Low-storage implementation of the optimal third order methods.

Low-storage fourth order SSP methods were found in [97]; no 2S methods are reported and the best 3S method has $\mathcal{C}_{\text{eff}} = 0.106$. Even allowing downwinding, the best 3S method reported there has $\mathcal{C}_{\text{eff}} = 0.187$. Therefore these are inferior to the eight-stage method mentioned above.

Our search recovered the same optimal methods for up to eight stages. However, these methods are superseded in terms of both efficiency and storage by the optimal ten stage method below.

2S Methods. No 2S fourth order SSP methods were previously known. By numerical search, we found a ten stage fourth order method implementable with two registers and with an effective SSP coefficient greater than any previously known fourth order full-storage method. The numerically determined coefficients approximate, to within machine precision, simple rational numbers. The method given below, using these rational number coefficients, is the only fourth order SSP method to be analytically proved optimal, because it achieves the optimal bound on ten-stage, fourth order SSP methods for linear

problems: $\mathcal{C} = R_{10,1,4}(\psi) = 6$. The nonzero coefficients are

$$\begin{aligned}
\beta_{i,i-1} &= \begin{cases} \frac{1}{6} & i \in \{1..4, 6..9\} \\ \frac{1}{15} & i = 5 \\ \frac{1}{10} & i = 10 \end{cases} \\
\beta_{10,4} &= \frac{3}{50} \\
\alpha_{i,i-1} &= \begin{cases} 1 & i \in \{1..4, 6..9\} \\ \frac{2}{5} & i = 5 \\ \frac{3}{5} & i = 10 \end{cases} \\
\alpha_{5,0} &= \frac{3}{5} \\
\alpha_{10,0} &= \frac{1}{25} \\
\alpha_{10,4} &= \frac{9}{25}.
\end{aligned}$$

The abscissas are

$$c = \frac{1}{6} \cdot (0, 1, 2, 3, 4, 2, 3, 4, 5, 6)^T. \tag{6.20}$$

Remark 6.4.2. *This method bears a remarkable similarity to the nine-stage third order method of the previous section; this is not altogether surprising because both of these methods achieve the linear stability limit and the optimal $s+1$-stage, 4th order linear SSP Runge-Kutta method is closely related to the optimal s-stage, 3rd order linear SSP Runge-Kutta method [75]. Generalizing this, one is led to hope for a family of fourth order methods similar to the third order family above - i.e., a family with n^2+1 stages and SSP coefficient $n^2 - n$. However, for the case $n = 2$, no such method exists [97]. Furthermore, after extensive analytical and numerical searches, we have been unable to find a method of this type for $n = 4$.*

A 2S implementation of the ten-stage method is given in Pseudo-code 3.

3S Methods. We have found many fourth order 3S methods with more than ten stages that are more efficient than the 2S ten-stage method above. It appears likely that with three registers it is possible to obtain fourth order methods with $\mathcal{C}_{\text{eff}}$ arbitrarily close to unity. However, the value of $\mathcal{C}_{\text{eff}}$ increases very slowly with the stage number and the optimization problem becomes increasingly difficult. We do not discuss these methods further here, except to say that the most efficient found so far has 26 stages and $\mathcal{C}_{\text{eff}} \approx 0.696$.

```
q1 = u; q2=u;
for i=1:5
  q1 = q1 + dt*F(q1)/6;
end
q2 = 1/25*q2 + 9/25*q1;
q1 = 15*q2-5*q1;
for i=6:9
  q1 = q1 + dt*F(q1)/6;
end
q1 = q2 + 3/5*q1 + 1/10*dt*F(q1);
u=q1;
```

Pseudo-code 3: Low-storage implementation of the ten-stage fourth order method.

Popular Method	$\mathcal{C}_{\text{eff}}$	Storage	Improved Method	$\mathcal{C}_{\text{eff}}$	Storage
SSPRK(2,2)	0.500	2S*	SSPRK(s,2)	$1-1/s$	**2S***
SSPRK(3,3)	0.333	**2S***	SSPRK(4,3)	0.500	**2S***
			SSPRK(n^2,3)	$\mathbf{1-1/n}$	**2S**
SSPRK(5,4)	0.377	3S	**SSPRK(10,4)**	**0.600**	**2S**
			SSPRK(26,4)	**0.696**	**3S**

Table 6.5: Properties of popular and of optimal explicit SSP Runge-Kutta methods. Methods and properties in bold indicate new contributions in the present work. An asterisk indicates that the previous timestep can be retained without increasing the required number of registers.

6.4.3 *Absolute Stability Regions*

As discussed in section 2, when a Runge-Kutta method is applied to a linear autonomous system of ODEs (2.3), it reduces to the iteration (2.10), characterized by the stability function ψ. For the case of a single linear ODE, this is simply $\mathbf{u}^{n+1} = \psi(\lambda\Delta t)\mathbf{u}^n$. The method is said to be absolutely stable for values of z such that $|\psi(z)| < 1$. For the optimal second order SSP family (6.16),

$$\psi(z) = \frac{1}{s} + \frac{s-1}{s}\left(1 + \frac{z}{s-1}\right)^{s-1}. \tag{6.21}$$

These are, of course, the same optimal polynomials found in section 2 for the s-stage 2nd order cases. For the optimal third order SSP methods (6.18),

$$\psi(z) = \left(\frac{n}{2n-1}\left(1 + \frac{z}{n^2-n}\right)^{(n-1)^2} + \frac{n-1}{2n-1}\left(1 + \frac{z}{n^2-n}\right)^{n^2}\right) \tag{6.22}$$

where again $n = \sqrt{s}$.

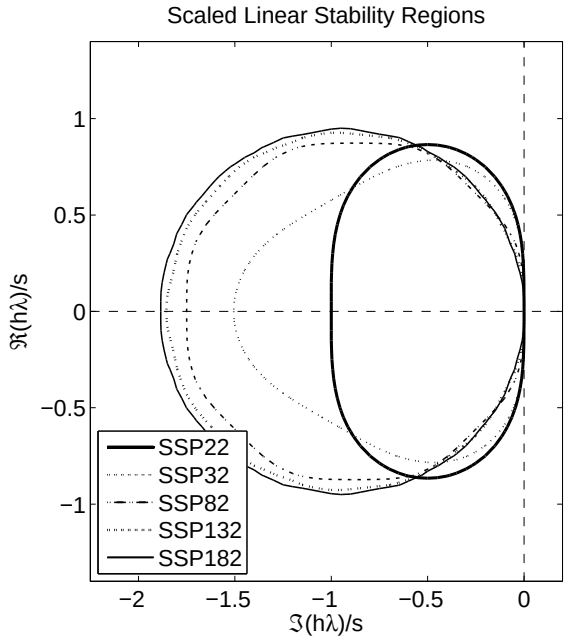
In Figure 6.9, we plot the corresponding absolute stability regions for some of these methods and some of the optimal fourth-order methods. The plots have been rescaled by dividing $h\lambda$ by the number of stages s, in order to give a fair comparison of relative computational efficiency. Note that, despite the increase in stage number, the SSP methods generally have larger scaled stability regions.

It is interesting to note that for large s, the stability functions of the optimal 2nd order methods approach that corresponding to s applications of the forward Euler method. The scaled absolute stability regions of the second order methods therefore tend to that of the forward Euler method.

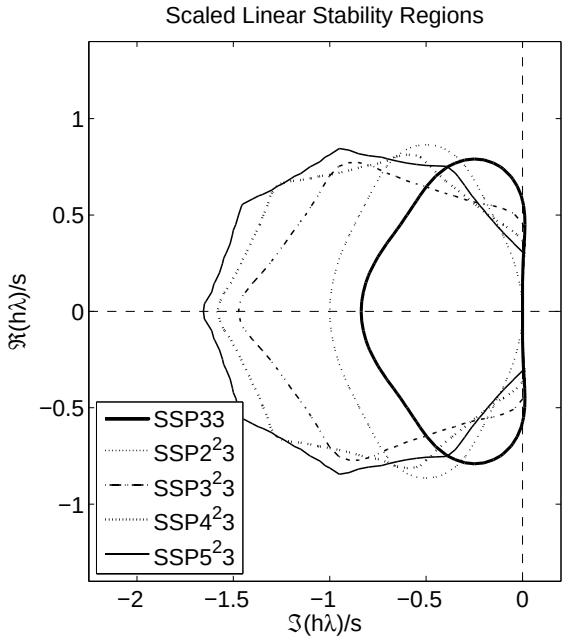
6.4.4 *Internal Stability*

The stability function may be thought of as modeling the amplification of errors in the initial stage. For Runge-Kutta methods with many stages, it is important also to consider amplification of roundoff errors occurring in the intermediate stages.

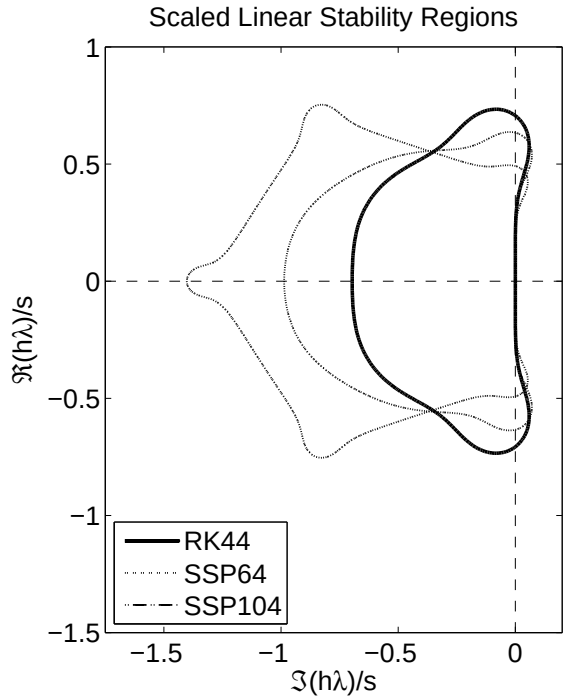
Consider a Shu-Osher implementation of a Runge-Kutta method including roundoff



(a) 2nd order methods



(b) 3rd order methods



(c) 4th order methods of optimal methods; RK44 denotes the classical 4th order Runge-Kutta method

Figure 6.9: Scaled stability regions of optimal explicit SSP methods.

errors r_i at each stage, applied to the test equation $u' = \lambda u$:

$$\begin{aligned} \tilde{y}_1 &= u^n + r_1, \\ \tilde{y}_i &= \sum_{j=1}^{i-1} \left(\alpha_{i,j} \tilde{y}_k + \Delta t \lambda \beta_{i,j} \tilde{y}_k \right) + r_i, \quad \alpha_{i,j} \geq 0, \quad i = 2, \ldots, s+1, \\ \tilde{u}^{n+1} &= \tilde{y}_{s+1}. \end{aligned} \tag{6.23}$$

By subtracting the true method (3.6) from the perturbed method (6.23), one finds that

$$\tilde{u}^{n+1} - u^{n+1} = \psi(\Delta t \lambda) r_1 + \sum_{j=2}^{s+1} \theta_j(\Delta t \lambda) r_j. \tag{6.24}$$

Here, ψ is again the absolute stability function; the functions θ_j are referred to as *internal stability polynomials* [120]. Assuming the r_j have magnitude on the order of roundoff ($\epsilon_{machine}$), the method will be internally stable as long as $||\psi(\Delta t \lambda)|| << 1/\epsilon_{machine}$ in the appropriate region of the complex plane. It is important to note that, in contrast to the stability function of a method, the internal stability polynomials depend on the particular manner in which the method is implemented.

2nd Order Methods

Straightforward calculation reveals that, for the optimal family of second order methods (6.16), as implemented above,

$$\theta_j(z) = \left(\frac{s-1}{s} + \frac{z}{s} \right) \left(1 + \frac{z}{s-1} \right)^{s-j} \tag{6.25}$$

for $2 \leq j \leq s$, while $\theta_{s+1}(z) = 1$. It can be shown that the region for which $|\theta_j(z)| < 1$ contains the absolute stability region of the method, so that for any linearly stable calculation, internal stability is never a concern for these methods.

3rd Order Methods

Similarly, for the optimal family of third order methods (6.18), as implemented above, the highest degree internal stability polynomials are given by

$$\theta_j(z) = \frac{1}{2n-1} \left[n \left(1 + \frac{z}{n^2-n} \right)^{(n-1)^2} + (n-1) \left(1 + \frac{z}{n^2-n} \right)^{n^2-j+1} \right] \tag{6.26}$$

for $2 \leq j \leq \frac{(n-1)(n-2)}{2} + 1$. Again, it can be shown that the region for which $|\theta_j(z)| < 1$ contains the absolute stability region of the method, so that for any linearly stable calculation, internal stability is never a concern for these methods.

Similar analysis shows that the new ten-stage fourth order method with the low-storage implementation presented here is internally stable. We omit the details here.

6.4.5 Truncation Error Analysis

By considering the Taylor series of the true solution and comparing the terms appearing in a Runge-Kutta method, bounds on the relative size of the leading terms of the local truncation error can be found. Similar to the derivation of order conditions, this analysis is simplified by using the theory of rooted trees. By assuming that F is sufficiently smooth and assuming bounds on F and its derivatives, the leading truncation error can be bounded by a constant proportional to ([12], p. 152)

$$C = \sum_{r(t)=p+1} \frac{1}{\sigma(t)} \left| \Phi(t) - \frac{1}{\gamma(t)} \right|.$$

Here the sum is over all rooted trees of order $p+1$, $\Phi(t)$ are the elementary differentials, and $\gamma(t), \sigma(t)$ are the density and the symmetry of the tree t, respectively. The reader is referred to [12] for further details. In the case of a linear autonomous ODE, only tall trees are important, so the above reduces to

$$C_L = \frac{1}{\sigma(T)} \left| \Phi(T) - \frac{1}{\gamma(T)} \right|$$

where T is the tall tree of order $p+1$.

It is straightforward to calculate the values of C, C_L for our optimal methods. The resulting error constants are given in Table 6.6; the error constants of some previously known methods are provided for comparison. Note that the error constants of the new methods are smaller in all cases, and decrease as the stage number increases. Thus, for a given timestep, the new methods are more accurate. If we compare the accuracy while holding the amount of computational work constant, we find that the size of the error increases very slowly with the number of stages. For instance, for linear problems SSP(10,4) gives an error about twice as large as RK(4,4) if the amount of work is held constant.

Table 6.6: Error constants of optimal explicit SSP RK methods. RK(4,4) is the classical 4th order Runge-Kutta method (for comparison).

Method	C_L	C
SSP(2,2)	$\frac{1}{6}$	$\frac{1}{4}$
SSP(s,2)	$\frac{1}{6(s-1)}$	$\frac{1}{4(s-1)}$
SSP(3,3)	$\frac{1}{24}$	$\frac{1}{8}$
SSP(n^2,3)	$\frac{((n-2)!)^2}{12(n!)^2}$	$\frac{(n^2-n+1)((n-2)!)^2}{12(n!)^2}$
RK(4,4)	$\frac{24}{2880}$	$\frac{101}{2880}$
SSP(10,4)	$\frac{1}{18} \cdot \frac{24}{2880}$	$\frac{17}{101} \cdot \frac{101}{2880}$

6.4.6 Embedding optimal SSP methods

Embedded Runge-Kutta methods provide an estimate of the local truncation error that can be computed at little cost. Embedded methods will be discussed further in Chapter 7; see also [47, 12].

When using an SSP method to enforce an important constraint, it is very desirable to have an embedded method that is also SSP under the same (or larger) timestep restriction, since violation of the constraint in the computation of the error estimate might lead to adverse effects. It turns out that it is possible to create embedded pairs from some of the optimal SSP methods.

The SSP72 method can be used as an embedded method with SSP93 for error control as follows:

$$u^{(0)} = u^n \tag{6.27a}$$

$$u^{(i)} = u^{(i-1)} + \frac{\Delta t}{6} F(u^{(i-1)} \qquad 1 \le i \le 6 \tag{6.27b}$$

$$u_2^{n+1} = \frac{1}{7}\left(u^n + 6u^{(6)} + \frac{\Delta t}{6} F(u^{(6)}\right) \tag{6.27c}$$

$$u^{(6)*} = \frac{3}{5}u^{(1)} + \frac{2}{5}u^{(6)} \tag{6.27d}$$

$$u^{(i)*} = u^{(i-1)*} + \frac{\Delta t}{6} F(u^{(i-1)*} \qquad 7 \le i \le 9 \tag{6.27e}$$

$$u_3^{n+1} = u^{(9)*} \tag{6.27f}$$

Here u_2^{n+1}, u_3^{n+1} are the second and third-order approximations corresponding to SSP72, SSP93. Note that one extra function evaluation is required over what is necessary for SSP93 alone. The same can be done for SSP32 with SSP43.

6.4.7 Numerical Experiments

In this section we demonstrate the effect of the SSP properties of our optimal methods through example problems.

Constant Coefficient Advection

We consider again the linear system of ODEs given by (2.3) with right-hand-side matrix (3.27), arising from first order upwind differencing of the linear advection equation (3.26). We recall that the exact solution of the PDE, as well as the exact solution of the semi-discretization, is monotonic in the maximum norm.

For this linear autonomous system, any Runge-Kutta method reduces to the simple iteration (2.10). Clearly monotonicity will be preserved for arbitrary initial conditions iff

$$||\psi(\Delta t\mathbf{L})|| \leq 1. \tag{6.28}$$

For this problem we have monotonicity for the forward Euler method under the maximal timestep $\Delta t_{\text{FE}} \leq \Delta x = \frac{1}{N}$. In order to compare different integration methods, we compute for each method the maximum value c_0 such that (6.28) holds with

$$\Delta t = c_0 \Delta t_{\text{FE}}. \tag{6.29}$$

From Theorem 3.2.3, we know that c_0 is exactly equal to the threshold factor $R(\psi)$. In Table 6.7, we list values of c_0, $R(\psi)$, and $R_{\text{eff}} = R(\psi)/s$ for various methods. Here we have included the non-SSP methods of Wang & Spiteri [1] [121] (note that they *are* SSP for linear autonomous problems). In every case the theory and experiment are in perfect agreement. A clear advantage in efficiency is conferred by the SSP methods with many stages.

---

[1] Note that in [121], the term 'strong stability preserving' was mistakenly used to refer to full discretizations of PDEs in many places, whereas the term actually refers to a property of the ODE solver. It appears that the term intended in these cases was 'total variation diminishing'.

Table 6.7: Threshold factors and effective threshold factors for some optimal explicit SSP RK methods. RK44 is the classical 4th order method (for comparison). The NSSP (Non-SSP) methods are from [121].

Method	$c_0 = R(\psi)$	R_{eff}
NSSP(2,1)	0.67	0.33
NSSP(3,2)	1	0.33
SSP(2,2)	1	0.50
SSP(10,2)	9	0.90
NSSP(3,3)	1	0.33
NSSP(5,3)	1.4	0.28
SSP(3,3)	1	0.33
SSP(4,3)	2	0.50
SSP(9,3)	6	0.67
SSP(25,3)	20	0.80
RK(4,4)	1	0.25
SSP(5,4)	1.86	0.37
SSP(10,4)	6	0.60

Variable Coefficient Advection

Previous work on SSP methods has emphasized that the SSP property is most critical when solving semi-discretizations of nonlinear PDEs with discontinuous solutions. The following example shows that SSP methods are relevant also for linear nonautonomous problems with smooth solutions but rapidly varying coefficients. A similar test problem was used in [76].

We solve the IBVP

$$u_t + (a(x,t)u)_x = 0 \tag{6.30a}$$

$$u(0,t) = 0 \qquad u(x,0) = g(x) \tag{6.30b}$$

on the interval $x \in [0,1]$ with

$$a(x,t) = \cos^2(20x + 45t). \tag{6.31}$$

We semi-discretize using upwind differencing and $N = 20$ points.

This rapidly oscillating velocity field is designed to demonstrate the effect of the SSP

property; it might be considered as a simple model of an underresolved turbulent flow. The low-storage property of our methods makes them appealing choices for direct numerical simulation of turbulent flows.

The exact solution to (6.30) is monotonic in the L^1 norm and is nonnegative for all time if $g(x) \geq 0$. For a given integration method, we are interested in the maximum timestep such that these properties hold discretely to within roundoff error ($\approx 10^{-15}$). For the forward Euler method, this maximum timestep is found to be $\Delta t = 1.02\Delta x$.

Because this semi-discretization is linear, any Runge-Kutta method applied to it reduces to the iteration

$$\boldsymbol{u}^{n+1} = \mathbf{M}_{\Delta t}(t)\boldsymbol{u}^n \tag{6.32}$$

where $\mathbf{M}_{\Delta t}(t)$ is the matrix-valued K-function of the method [76]. Thus monotonicity and positivity are equivalent to

$$||\mathbf{M}_{\Delta t}||_1 \leq 1, \quad \text{and} \quad \mathbf{M}_{\Delta t} \geq 0,$$

respectively (the second inequality is interpreted componentwise).

In figures 6.10-6.11, we plot the theoretical monotone and positive scaled timestep $\Delta t/(s\Delta x)$ (i.e., the effective SSP coefficient) versus the observed maximum scaled monotone and positive timestep for the second and third order families of optimal methods. In all cases, the theory is borne out by these results; furthermore, the theoretical timestep limit seems to be quite sharp for this problem. For comparison, we also plot the observed maximum monotone and positive timestep for the most commonly used second and third order SSP methods.

In Table 6.8 we list, for various methods, the maximum observed monotone and positive timestep for this problem, along with effective SSP coefficients for linear and nonlinear problems. Again we see fairly good agreement with theory, and a clear advantage conferred by the SSP methods with many stages.

Remark 6.4.3. *As expected, nearly all the non-SSP methods fail to produce monotone results even for very small relative timesteps (< 0.1) for this problem. On the other hand, the classical fourth order method performs reasonably well despite being non-SSP for nonautonomous/nonlinear problems. This demonstrates that SSP timestep restrictions are not always very sharp for a particular choice of ODE and time integrator.*

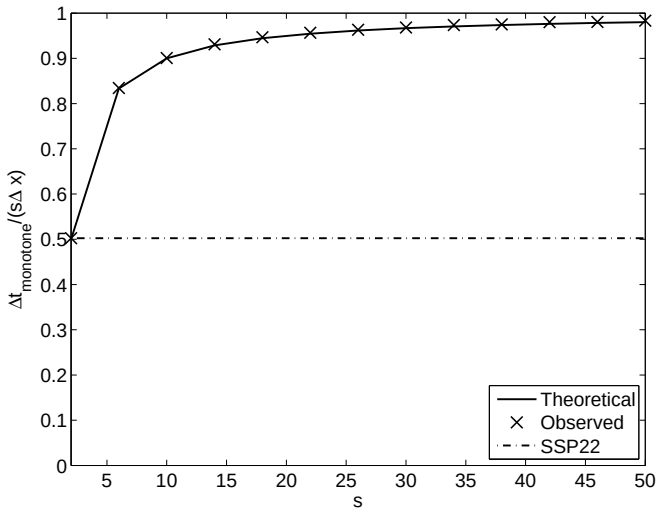


Figure 6.10: Theoretical and actual monotone effective timesteps for the family of optimal 2nd order methods. The horizontal line shows the actual monotone effective timestep of the popular SSP22 method, for comparison.

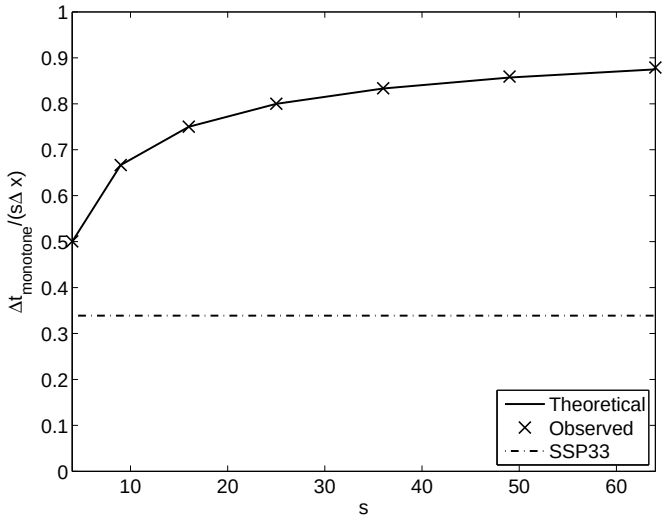


Figure 6.11: Theoretical and actual monotone effective timesteps for the family of optimal 3rd order methods. The horizontal line shows the actual monotone effective timestep of the popular SSP33 method, for comparison.

Method	$\mathcal{C}_{\text{eff}}$	Monotone effective timestep
NSSP(2,1)	0	0.033
NSSP(3,2)	0	0.037
NSSP(3,3)	0	0.004
NSSP(5,3)	0	0.017
RK(4,4)	0	0.287
SSP(5,4)	0.302	0.416
SSP(10,4)	0.600	0.602

Table 6.8: Theoretical and observed monotone effective timesteps for variable coefficient advection. RK44 is the classical 4th order method.

6.5 Summary and Conjectures

The effective SSP coefficients of all known optimal SSP Runge-Kutta methods are summarized in Table 6.9. For integration of systems of ODEs that satisfy the forward Euler condition (3.10), these coefficients are the bottom line in terms of computational efficiency of a method. Note that this should be used only to compare methods within a class – explicit or (diagonally) implicit. Comparison of efficiency of explicit versus implicit methods is much more complicated, although it seems clear that the explicit methods will be more efficient when the timestep is controlled exclusively by SSP considerations. The implicit methods may be useful for stiff problems that also have a mild SSP timestep constraint since the scaled absolute stability regions of the implicit methods are much larger than those of optimal explicit SSP methods.

The optimality of most of the explicit methods is ascertained by virtue of their achieving the theoretical upper bound $\mathcal{C}_{s,1,p} = R_{s,1,p}$. On the other hand, this holds for only two of the implicit methods.

However, we have reason to believe that at least the second and third order implicit methods are optimal. For these cases, using multiple random initial guesses, the optimization algorithm consistently converges to the same method, or to a reducible method corresponding to one of the numerically optimal methods with a smaller number of stages. Also, many of the inequality constraints are satisfied exactly for these methods. Furthermore, the methods all have a similar form, depending only on the stage number. These observations suggest the following two conjectures.

Table 6.9: Effective SSP coefficients of best known methods. A dash indicates that SSP methods of this type cannot exist. A blank space indicates that no SSP methods of this type were found.

s \ p	Implicit Methods					Explicit Methods		
	2	3	4	5	6	2	3	4
1	2	-	-	-	-	-	-	-
2	2	1.37	-	-	-	0.5	-	-
3	2	1.61	0.68	-	-	0.67	0.33	-
4	2	1.72	1.11	0.29		0.75	0.5	-
5	2	1.78	1.21	0.64		0.8	0.53	0.30
6	2	1.82	1.30	0.83	0.030	0.83	0.59	0.38
7	2	1.85	1.31	0.89	0.038	0.86	0.61	0.47
8	2	1.87	1.33	0.94	0.28	0.88	0.64	0.52
9	2	1.89	1.34	0.99	0.63	0.89	0.67	0.54
10	2	1.90	1.36	1.01	0.81	0.9	0.68	0.60
11	2	1.91	1.38	1.03	0.80	0.91	0.69	0.59

Conjecture 6.5.1. *(An extension of [35, Conjecture 3.1]) The optimal second-order s-stage implicit SSP method is given by the SDIRK method* (6.10) *and hence* $\mathcal{C}^{\mathrm{I}}_{s,1,2} = \mathcal{C}^{\mathrm{SDI}}_{s,1,2} = 2s$.

Conjecture 6.5.2. *(An extension of [35, Conjecture 3.2]) For* $s \geq 2$, *the optimal third-order s-stage implicit Runge–Kutta SSP method is given by the SDIRK method* (6.11) *and hence* $R^{\mathrm{I}}_{s,1,3} = s - 1 + \sqrt{s^2 - 1}$.

Conjecture 6.5.1 would imply that the effective SSP coefficient of any Runge–Kutta method of order greater than one is at most equal to two. In fact, van de Griend and Kraaijevanger [118] showed that the optimal $R(\psi) \geq 2m$ for second order methods with m stages, and conjectured that $R(\psi) = 2m$. They proved the conjecture only in the one- and two-stage cases. In fact it does not hold for the three-stage case, as we demonstrate with the following counterexample:

$$\psi = \frac{1 + \frac{7969150767159903}{18014398509481984}x + \frac{4716995547632067}{72057594037927936}x^2 + \frac{1867769670100979}{576460752303423488}x^3}{1 - \frac{313913991947565}{562949953421312}x + \frac{8869189497956419}{72057594037927936}x^2 - \frac{1762527965732417}{144115188075855872}x^3} \tag{6.34}$$

Since the numerator and denominator have degree three, this corresponds to the stability function of an implicit RK method with $m = 3$ stages. This function was found by numerical search; using the algorithm in [118] we have verified that $R(\psi) \geq 6.77 > 2s$.

Thus our conjecture cannot be proved by analyzing $R(\psi)$.

Proving the implication that $C_{\text{eff}} \leq 2$ for Runge-Kutta methods, and its extension to general linear methods, is the most significant open problem on SSP methods. For this reason, we restate the problem here in a concise algebraic form:

Conjecture 6.5.3. *(algebraic reformulation of conjecture 6.5.1) Let $r > 0$, $\boldsymbol{b} \in \Re^s$, $\mathbf{A} \in \Re^{s\times s}$, and let $\mathbf{K}$ be given by* (3.28)*. Suppose that $\mathbf{I} + r\mathbf{K}$ is invertible and that the following hold:*

$$\boldsymbol{b}^{\mathrm{T}}\boldsymbol{e} = 1 \tag{6.35a}$$

$$\boldsymbol{b}^{\mathrm{T}}\mathbf{A}\boldsymbol{e} = \frac{1}{2} \tag{6.35b}$$

$$\mathbf{K}(\mathbf{I} + r\mathbf{K})^{-1} \geq 0 \tag{6.35c}$$

$$r\mathbf{K}(\mathbf{I} + r\mathbf{K})^{-1}\boldsymbol{e} \leq 1. \tag{6.35d}$$

where, as usual, vector and matrix inequalities are understood componentwise. Then $r \leq 2s$.

Chapter 7

Low-Storage Runge-Kutta Methods

> Storage management is an aspect of codes for the solution of ODEs which has received little attention...
>
> *L.F. Shampine (1979)*
>
> ...the storage requirement becomes crucial whenever the systems are very large. For instance, in the simulation of plasmas, there may be 10,000 or even 1,000,000 particles...
>
> *J.H. Williamson (1980)*

7.1 Introduction

This chapter deals with new classes of Runge-Kutta methods that were inspired by low-storage properties of the optimal explicit SSP methods of Chapter 6. In this chapter, we disregard the SSP property and focus instead on the issue of storage requirements.

A straightforward implementation of the Runge-Kutta method (2.9a) requires $m + 1$ memory registers of length N, where m is the number of stages and N is the number of ordinary differential equations (note that we break from the notation of previous chapters, in this chapter only, by using m in place of s to represent the number of stages). In early computers, only a very small amount of memory was available for storing both the program and intermediate results of the computation, which led Gill to devise a four-stage

fourth-order Runge-Kutta method that could be implemented using only three memory registers [39]. The method relied on a particular algebraic relation between the coefficients, such that only certain combinations of previous stages (rather than all of the stages themselves) needed to be stored. This is the basic idea underlying all low-storage methods, including those of the present work. Blum later provided a three-register implementation of the classical Runge-Kutta method (with rational coefficients) [8]. Fyfe showed that all four-stage methods are capable of three-register implementation [38]. Shampine devised a variety of techniques for reducing the storage requirements of methods with many stages [104]. Williamson devised a two-register algorithm [124]; he showed that "all second-order, many third-order, and a few fourth-order" methods can be implemented in this fashion. One of his third-order methods is among the most popular low-storage methods; however, his fourth-order methods are not generally useful because they apply only to the special case in which $F(u)$ is bounded as $u \to \infty$.

On modern computers, storage space for programs is no longer a concern; however, when integrating very large numbers of ODEs, fast memory for temporary storage during a computation is often the limiting factor. This is typically the case in method of lines discretizations of PDEs, and modern efforts have focused on finding low-storage methods that are also optimized for stability and or accuracy relative to particular semi-discretizations of PDEs. Exploiting Williamson's technique, Carpenter and Kennedy [18] developed four-stage, third-order, two-register methods with embedded second order methods for error control. They also derived five-stage fourth order two-register methods [17]. In perhaps the most thorough work on low-storage methods, Kennedy et. al. [66] generalized a type of low-storage implementation originally due to van der Houwen [119]. They provide many methods of various orders, optimized for a variety of accuracy and stability properties. Further development of low-storage Runge-Kutta methods in the last decade has come from the computational aeroacoustics community [59, 113, 15, 16, 9, 7, 117].

All of the two-register methods in the literature use one of two algorithms (referred to below as 2N and 2R). These algorithms rely on particular assumptions regarding the evaluation and storage of F. Recently, in [69], a new type of low-storage algorithm was proposed, based on a different assumption on F. The aim of this chapter is to present a general algorithm based on this assumption, which includes the 2N and 2R algorithms as special cases, but allows additional degrees of freedom in designing the methods.

It is often important, when solving an ODE numerically, to have an estimate of the local truncation error. Methods that provide such error estimates are known as embedded methods. Existing low-storage embedded methods always require an extra memory

register to provide the error estimate. If a desired error tolerance is exceeded in a given step, it may be necessary to restart that step. In this case, another additional register is necessary for storing the previous step solution. In some applications where no error estimate is used, restarting may still be required based on some other condition (such as a CFL condition) that is checked after each step. In this case, again, existing low-storage methods require the use of an extra register.

In this chapter work we present improved low-storage methods that use the theoretical minimum number of registers in each of these situations, i.e. two registers if an error estimate or the ability to restart is required, and three registers if both are needed. In each case these methods use one register fewer than any known methods.

In Section 7.2, we review existing low-storage methods and explicitly define the assumptions required for their implementation. In Section 7.3, we observe that these methods have sparse Shu-Osher forms. Based on this observation, we introduce a new, more general class of low-storage methods in Section 7.4. In Section 7.5, we explain how the low-storage methods can be implemented for integrating an important class of PDE semi-discretizations. In Section 7.6, we present new low-storage methods.

7.2 Two-register Methods

Before proceeding, it is helpful to define precisely what is meant by the number of registers required by a method. Let N denote the number of ODEs to be integrated (typically the number of PDEs multiplied by the number of gridpoints). Then we say a method requires M registers if each step can be calculated using $MN + o(N)$ memory locations.

Let $\mathtt{S}_1, \mathtt{S}_2$ represent two N-word registers in memory. Then it is always assumed that we can make the assignments

$$\mathtt{S}_1 := \mathtt{F}(\mathtt{S}_2)$$

and

$$\mathtt{S}_1 := \mathtt{c}_1\mathtt{S}_1 + \mathtt{c}_2\mathtt{S}_2$$

without using additional memory beyond these two registers. Here $\mathtt{a} := \mathtt{b}$ means 'the value of $\mathtt{b}$ is stored in $\mathtt{a}$'. Using these only these two types of assignments, it is straightforward to implement an m-stage method using $m + 1$ registers.

Various Runge-Kutta algorithms have been proposed that require only two registers. Each requires some additional type of assignment, and takes one of the following two forms.

7.2.1 Williamson (2N) methods

Williamson methods [124] require 2 registers and take the following form:

Algorithm 1: Williamson (2N)

(y_1) $\mathtt{S_1} := \mathtt{u^n}$
$\mathtt{for\ i} = \mathtt{2 : m} + \mathtt{1\ do}$
$\quad \mathtt{S_2} := \mathtt{A_i S_2} + \Delta\mathtt{tF}(\mathtt{S_1})$
(y_i) $\quad \mathtt{S_1} := \mathtt{S_1} + \mathtt{B_i S_2}$
$\mathtt{end}$
$u^{n+1} = \mathtt{S_1}$

with $A_2 = 0$. The methods proposed in [18, 17, 113, 7, 54] are also of Williamson type. Observe that an m-stage method has $2m - 1$ free parameters. The coefficients above are related to the Butcher coefficients as follows:

$$\begin{aligned} B_i &= a_{i+1,i} && i < m \\ B_m &= b_m \\ A_i &= \frac{b_{i-1} - a_{i,i-1}}{b_i} && b_i \neq 0 \\ A_i &= \frac{a_{i+1,i-1} - c_i}{a_{i+1,i}} && b_i = 0. \end{aligned}$$

In order to implement these methods with just two registers, the following assumption is required:

Assumption 7.2.1 (Williamson)**.** *Assignments of the form*

$$\mathtt{S_2} := \mathtt{S_2} + \mathtt{F}(\mathtt{S_1}) \tag{7.2}$$

can be made with only $2N + o(N)$ *memory.*

Williamson notes that (converting some notation to the present)

> No advantage can be gained by trying to generalize [Algorithm 1] by including a term [proportional to $\mathtt{S_1}$] in the expression for [$\mathtt{S_2}$]...this is because the additional equations determining the new parameters turn out to be linearly dependent on those for A_i and B_i.

Indeed, it appears that the algorithm above is the most general possible using only two registers and Assumption 7.2.1.

7.2.2 van der Houwen (2R) methods

A different low-storage algorithm was developed by van der Houwen [119] and Wray [66]. It is similar to, but more aggressive than the approach used by Gill [39]. The implementation is given as Algorithm 2.

Algorithm 2: van der Houwen (2R)

$\mathtt{S}_2 := \mathtt{u}^{\mathtt{n}}$

`for i = 1 : m do`

(y_i) $\quad \mathtt{S}_1 := \mathtt{S}_2 + (\mathtt{a}_{\mathtt{i},\mathtt{i}-1} - \mathtt{b}_{\mathtt{i}-1})\Delta\mathtt{t}\mathtt{S}_1$

$\quad \mathtt{S}_1 := \mathtt{F}(\mathtt{S}_1)$

$\quad \mathtt{S}_2 := \mathtt{S}_2 + \mathtt{b}_{\mathtt{i}}\Delta\mathtt{t}\mathtt{S}_1$

`end`

$u^{n+1} = \mathtt{S}_2.$

The coefficients a_{ij}, b_j are the Butcher coefficients, and we define $a_{10} = b_0 = 0$. Again, an m-stage method has $2m - 1$ free parameters. The class of methods proposed in [15, 16] is equivalent. In [66], an implementation is given that requires swapping the roles of the two registers at each stage. Here we have followed the implementation of [16] as it is less complicated in that the roles of the two registers need not be swapped at each stage.

Methods of van der Houwen type have also been proposed in [117]. The low-storage methods of [59] can be viewed as a subclass of van der Houwen methods with especially simple structure.

In order to implement these methods with just two registers, the following assumption is required:

Assumption 7.2.2 (van der Houwen)**.** *Assignments of the form*

$$\mathtt{S}_1 := \mathtt{F}(\mathtt{S}_1)$$

may be made with only $N + o(N)$ memory.

Again, it seems that the above algorithm is the most general possible using only two registers and Assumption 7.2.2.

7.3 Low-Storage Methods Have Sparse Shu-Osher Forms

The low-storage methods above can be better understood by considering the Shu-Osher form (3.6) discussed in Chapter 3. As discussed there, the Shu-Osher form for a given method is not unique. By writing (3.6) as a homogeneous linear system, it follows that the method is invariant under the transformation (for any t and $i, j > 1$)

$$\alpha_{ij} \implies \alpha_{ij} - t \tag{7.3a}$$

$$\alpha_{ik} \implies \alpha_{ik} + t\alpha_{jk} \qquad k \neq j \tag{7.3b}$$

$$\beta_{ik} \implies \beta_{ik} + t\beta_{jk}. \tag{7.3c}$$

It is convenient to define the matrices:

$$(\boldsymbol{\alpha})_{ij} = \begin{cases} 0 & i = 1 \\ \alpha_{ij} & i > 1 \end{cases} \tag{7.4}$$

$$(\boldsymbol{\beta})_{ij} = \begin{cases} 0 & i = 1 \\ \beta_{ij} & i > 1. \end{cases} \tag{7.5}$$

Defining further $\boldsymbol{\alpha}_0, \boldsymbol{\beta}_0$ to be the upper $m \times m$ parts of $\boldsymbol{\alpha}, \boldsymbol{\beta}$, and $\boldsymbol{\alpha}_1, \boldsymbol{\beta}_1$ to be the remaining last rows, the relation between the Butcher array and the Shu-Osher form is

$$A = (I - \boldsymbol{\alpha}_0)^{-1}\boldsymbol{\beta}_0 = \left(\sum_{i=0}^{m-1} \boldsymbol{\alpha}_0^i \right) \boldsymbol{\beta}_0 \tag{7.6a}$$

$$b^{\mathrm{T}} = \boldsymbol{\beta}_1 + \boldsymbol{\alpha}_1 A. \tag{7.6b}$$

It turns out that Williamson and van der Houwen methods possess a Shu-Osher form in which the matrices $\boldsymbol{\alpha}, \boldsymbol{\beta}$ are very sparse. This is not surprising, since low-storage algorithms rely on partial linear dependencies between the stages, and such dependencies can be exploited using the transformation (7.3) to introduce zeros into these matrices.

7.3.1 2N Methods

By straightforward algebraic manipulation, Williamson methods can be written in Shu-Osher form with

$$y_i = \alpha_{i,i-2} y_{i-2} + (1 - \alpha_{i,i-2}) y_{i-1} + \beta_{i,i-1} \Delta t F(y_{i-1}) \qquad 1 < i \leq m + 1 \tag{7.7}$$

where $\alpha_{i,i-2} = -\frac{B_i A_i}{B_{i-1}}$ and $\beta_{i,i-1} = B_i$. Here and elsewhere, any coefficients with nonpositive indices are taken to be zero. Notice that $\boldsymbol{\alpha}$ is bidiagonal and $\boldsymbol{\beta}$ is diagonal.

7.3.2 2R Methods

Similarly, van der Houwen methods can be written in Shu-Osher form with

$$y_i = y_{i-1} + \beta_{i,i-2}\Delta t F(y_{i-2}) + \beta_{i,i-1}\Delta t F(y_{i-1}) \qquad 1 < i \leq m+1 \tag{7.8}$$

where $\beta_{i,i-2} = b_{i-1} - a_{i-1,i-2}$ and $\beta_{i,i-1} = a_{i,i-1}$. Notice that $\boldsymbol{\alpha}$ is diagonal and $\boldsymbol{\beta}$ is bidiagonal.

7.4 2S Methods

Based on the Shu-Osher forms presented above for 2N and 2R methods, it is natural to ask whether it is possible to implement a method with just two registers if $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ have other types of sparse structure. Perhaps the most obvious generalization is to allow both matrices to be bidiagonal, i.e.

$$y_i = \alpha_{i,i-2} y_{i-2} + (1 - \alpha_{i,i-2}) y_{i-1} + \beta_{i,i-2}\Delta t F(y_{i-2}) + \beta_{i,i-1}\Delta t F(y_{i-1}). \tag{7.9}$$

It turns out that this is possible, under the following assumption, which was introduced in [69]:

Assumption 7.4.1. *Assignments of the form*

$$\mathtt{S}_1 := \mathtt{S}_1 + \mathtt{F}(\mathtt{S}_1)$$

can be made with only $N + o(N)$ memory.

Examining the Shu-Osher form (7.9), it is clear that 2S methods may be implemented (under Assumption 7.4.1) using two registers if one is willing to evaluate $F(y_i)$ twice for each stage i. With a little care, however, this doubling of the number of function evaluations can be avoided. The resulting algorithm is given as Algorithm 3.

The value of δ_m makes no essential difference (any change can be compensated by changing $\gamma_{m+1,1}, \gamma_{m+1,2}$), so we set it to zero. Consistency requires that $\delta_1 = 1, \gamma_{22} = 1$, and

$$\gamma_{i,1} = 1 - \gamma_{i,2} \sum_{j=1}^{i-1} \delta_j \qquad 2 \leq i \leq m+1,$$

Algorithm 3: 2S

$\mathtt{S}_2 := 0$

(y_1) $\mathtt{S}_1 := \mathtt{u}^{\mathtt{n}}$

$\mathtt{for\ i} = 2 : \mathtt{m} + 1\ \mathtt{do}$

$\quad \mathtt{S}_2 := \mathtt{S}_2 + \mathtt{¡}_{\mathtt{i}-1}\mathtt{S}_1$

(y_i) $\quad \mathtt{S}_1 := \mathtt{'}_{\mathtt{i1}}\mathtt{S}_1 + \mathtt{'}_{\mathtt{i2}}\mathtt{S}_2 + \mathtt{↓}_{\mathtt{i},\mathtt{i}-1}\Delta\mathtt{tF}(\mathtt{S}_1)$

$\mathtt{end}$

$u^{n+1} = \mathtt{S}_1$

leaving $3s - 3$ free parameters – significantly more than for the 2N or 2R methods. We refer to these methods as 2S methods. Clearly this class includes the 2N and 2R methods as well as new methods.

While the Butcher coefficients are not needed for implementation, they are useful for analyzing the properties of the methods. They can be obtained from the low-storage coefficients as follows. The coefficients $\beta_{i,i-1}$ appearing in Algorithm 3 are Shu-Osher coefficients. In terms of the low-storage coefficients, the remaining nonzero Shu-Osher coefficients are

$$\begin{aligned}
\beta_{i+1,i-1} &= -\frac{\gamma_{i+1,2}}{\gamma_{i,2}}\beta_{i,i-1} && 2 \le i \le m \\
\alpha_{i+1,i-1} &= -\frac{\gamma_{i+1,2}}{\gamma_{i,2}}\gamma_{i,1} && 2 \le i \le m \\
\alpha_{i+1,i} &= 1 - \alpha_{i+1,i-1} && 2 \le i \le m.
\end{aligned}$$

The Butcher coefficients are obtained by substituting the above values into (7.6).

If $\gamma_{i2} = 0$ for some i, the low-storage method cannot be written in the bidiagonal Shu-Osher form; however, it will still possess a sparse (in fact, even more sparse) Shu-Osher form, and can be implemented using two registers in a slightly different way. We do not investigate such methods in detail, since they have a smaller number of free parameters than the general case. However, note that some of the methods of [69] are of this type.

7.4.1 2S Methods*

It is common to check some accuracy or stability condition after each step, and to reject the step if the condition is violated. In this case, the solution from the last timestep,

u^n, must be retained during the computation of u^{n+1}. For 2R/2N/2S methods, this will require an additional register. On the other hand, in [69], methods were proposed that can be implemented using only two registers, with one register retaining the previous solution. We refer to these as 2S* methods. These methods have Shu-Osher form

$$y_i = \alpha_{i,1}u^n + \alpha_{i,i-1}y_{i-1} + \beta_{i,i-1}\Delta t F(y_{i-1}) \tag{7.11}$$

Here we give a general algorithm for such methods (Algorithm 4), which is straightforward given the sparse Shu-Osher form. It is equivalent to the usual 2S algorithm with $\gamma_{i2} = \alpha_{i,1}$ and $\delta_i = 0$ except $\delta_1 = 1$. Remarkably, these methods have as many free parameters (2m-1) as 2N/2R methods.

Algorithm 4: 2S*

(y_1) $\mathtt{S_1} := \mathtt{u^n} \quad \mathtt{S_2} := \mathtt{u^n}$

`for i = 2 : m + 1 do`

(y_i) $\quad \mathtt{S_1} := (1 - \uparrow_{\mathtt{i,1}})\mathtt{S_1} + \uparrow_{\mathtt{i,1}}\mathtt{S_2} + \downarrow_{\mathtt{i,i-1}}\Delta\mathtt{tF}(\mathtt{S_1})$

`end`

$u^{n+1} = \mathtt{S_1}$

7.4.2 2S Embedded Pairs

It is often desirable to compute an estimate of the local error at each step. The most common way of doing this is to use an embedded method, i.e. a second Runge-Kutta method that shares the same matrix A (hence the same stages) but a different vector of weights $\hat{b}$ in place of b. The methods are designed to have different orders, so that their difference gives an estimate of the error in the lower order result. As it is common to advance the higher order solution ('local extrapolation'), we refer to the higher order method as the principal method, and the lower order method as the embedded method. Typically the embedded method has order one less than the principal method.

In [66], many 2R embedded pairs are given; however, a third storage register is required for the error estimate. The 2S algorithm can be modified to include an embedded method while still using only two storage registers. The implementation is given as Algorithm 5.

Here $\hat{u}^{n+1}$ is the embedded solution. Note that there are two additional free parameters, δ_m, δ_{m+1}, effectively determining the weights $\hat{b}$. Since the conditions for first and

Algorithm 5: 2S embedded

$\mathtt{S}_2 := 0$

(y_1) $\mathtt{S}_1 := \mathtt{u}^{\mathtt{n}}$

for $\mathtt{i} = 2 : \mathtt{m} + 1$ do

  $\mathtt{S}_2 := \mathtt{S}_2 + {}_{\mathtt{i}-1}\mathtt{S}_1$

(y_i) $\mathtt{S}_1 := {}'_{\mathtt{i}1}\mathtt{S}_1 + {}'_{\mathtt{i}2}\mathtt{S}_2 + {}_{\downarrow\mathtt{i},\mathtt{i}-1}\Delta\mathtt{t}\mathtt{F}(\mathtt{S}_1)$

end

$u^{n+1} = \mathtt{S}_1$

$(\hat{u}^{n+1})$ $\mathtt{S}_2 := \dfrac{1}{\sum_{\mathtt{i}=2}^{\mathtt{m}+1} {}_{\mathtt{i}}\mathtt{i}} \left(\mathtt{S}_2 + {}_{\mathtt{i}\mathtt{m}+1}\mathtt{S}_1\right)$

second order are (for fixed abscissas c) a pair of linear equations for the weights, it would appear that if the embedded method is at least second order, then necessarily $\hat{b} = b$. However, by including additional stages, the added degrees of freedom can be used to achieve independence of $b, \hat{b}$.

The relation between the coefficients in Algorithm 5 and the Shu-Osher coefficients for the principal method is again given by (7.10), and the Butcher coefficients can then be obtained using (7.6). The embedded method has the same Butcher arrays A, c, with the weights $\hat{b}$ given by

$$\hat{b}_j = \frac{1}{\sum_k \delta_k} \left(\delta_{m+1} b_j + \sum_i \delta_i a_{ij} \right) \tag{7.12}$$

where b_j, a_{ij} are the Butcher coefficients of the principal method.

7.4.3 3S Methods*

Our experience indicates that the class of 2S* methods above typically have relatively unfavorable error constants (though they are not so large as to be unusable for practical applications). Hence we are led to consider methods with Shu-Osher form

$$y_i = \alpha_{i,1} u^n + \alpha_{i,i-1} y_{i-1} + \alpha_{i,i-2} y_{i-2} + \beta_{i,i-2} \Delta t F(y_{i-2}) + \beta_{i,i-1} \Delta t F(y_{i-1}) \tag{7.13}$$

These methods can be implemented using three registers, while retaining the previous solution. Hence we refer to them as 3S* methods. Because they allow more free parameters than 2S* methods, 3S* methods can be found with much smaller error constants. Furthermore, it is possible to design embedded pairs within this framework. The corresponding

algorithm is given as Algorithm 6. Note that these are the first methods to provide both error control and the ability to restart a step with only three memory registers. No further improvement is possible, since the new solution, the previous solution, and an error estimate must be available simultaneously.

Algorithm 6: 3S* embedded

$$
\begin{aligned}
& \mathtt{S}_3 := \mathtt{u}^{\mathtt{n}} \\
(y_1) \quad & \mathtt{S}_1 := \mathtt{u}^{\mathtt{n}} \\
& \texttt{for i} = 2 : \mathtt{m}+1 \texttt{ do} \\
& \quad \mathtt{S}_2 := \mathtt{S}_2 + {}_{\mathtt{i}-1}\mathtt{S}_1 \\
(y_i) \quad & \quad \mathtt{S}_1 := {}'_{\mathtt{i}1}\mathtt{S}_1 + {}'_{\mathtt{i}2}\mathtt{S}_2 + {}'_{\mathtt{i}3}\mathtt{S}_3 + {}_{\downarrow\mathtt{i},\mathtt{i}-1}\Delta\mathtt{t}\mathtt{F}(\mathtt{S}_1) \\
& \texttt{end} \\
(\hat{u}^{n+1}) \quad & \mathtt{S}_2 := \frac{1}{\sum_{\mathtt{j}=1}^{\mathtt{m}+2} {}_{\mathtt{i}}\mathtt{j}} \left(\mathtt{S}_2 + {}_{\mathtt{i}\mathtt{m}+1}\mathtt{S}_1 + {}_{\mathtt{i}\mathtt{m}+2}\mathtt{S}_3\right) \\
& u^{n+1} = \mathtt{S}_1
\end{aligned}
$$

Note that including terms in $\mathtt{S}_2$ proportional to $\mathtt{S}_3$ is superfluous. Consistency requires $\delta_1 = 1$ and we take $\gamma_{22} = 1, \gamma_{21} = \gamma_{23} = \gamma_{33} = \gamma_{43} = 0$ to eliminate additional spurious degrees of freedom. Thus the primary method has $4m - 6$ free parameters, with 3 more available for the embedded method. Once again, to avoid having $\hat{b} = b$, additional stages are necessary.

Again, the coefficients $\beta_{i,i-1}$ in Algorithm 6 are just the corresponding Shu-Osher coefficients. The remaining Shu-Osher coefficients are

$$
\begin{aligned}
\beta_{i+1,i-1} &= -\frac{\gamma_{i+1,2}}{\gamma_{i,2}}\beta_{i,i-1} && 2 \le i \le m \\
\alpha_{i+1,1} &= \gamma_{i+1,3} - \frac{\gamma_{i+1,2}}{\gamma_{i,2}}\gamma_{i,3} && 2 \le i \le m \\
\alpha_{i+1,i-1} &= -\frac{\gamma_{i+1,2}}{\gamma_{i,2}}\gamma_{i,1} && 2 \le i \le m \\
\alpha_{i+1,i} &= 1 - \alpha_{i+1,i-1} - \alpha_{i+1,1} && 2 \le i \le m.
\end{aligned}
$$

The Butcher array for the principal method can then be obtained using (7.6). The embedded method is again identical except for the weights, which are given by (7.12).

7.5 Feasibility of Low-storage Assumptions

In this section we discuss the feasibility of the various low-storage assumptions. We will assume the method is applied to integrate a semi-discretization of a PDE on a structured grid where the stencil is local; in 1D this means that the Jacobian is sparse with narrow bandwidth; i.e., the formula for updating a given cell depends on a local stencil whose size is independent of the overall grid size. This is typical of many finite difference, finite volume, and discontinuous Galerkin methods. For semi-discretizations with dense jacobian, it appears that an additional 'working space' memory register will always be necessary. In one dimension, we write the semi-discretization as:

$$\frac{\partial u_{ij}}{\partial t} = F(u_{i-r}, \ldots, u_{i+r}) \tag{7.15}$$

for some (small) fixed integer r.

7.5.1 2N Methods

Recall that 2N methods require Assumption 7.2.1, which involves assignments of the form

$$\mathtt{S}_1 := \mathtt{S}_1 + \mathtt{F}(\mathtt{S}_2). \tag{7.16}$$

For systems of the form (7.15), implementation of 2N methods is completely straightforward, since the register from which F is being calculated is different from the register to which it is being written. The algorithm simply marches along the grid, calculating F at each point.

7.5.2 2R Methods

Recall that 2R methods require Assumption 7.2.2, which involves assignments of the form

$$\mathtt{S}_1 := \mathtt{F}(\mathtt{S}_1). \tag{7.17}$$

A naive implementation like that prescribed for 2N methods above will overwrite solution values that are needed for subsequent computations of F. It is thus necessary to maintain a small buffer with old solution values that are still needed. In one dimension, the buffer need only be the size of the computational stencil (i.e., 2r+1). The algorithm looks roughly

like this (letting S denote the memory register and w the buffer)

$$\begin{aligned}
&\texttt{w}[\texttt{1} : \texttt{2r}] := \texttt{w}[\texttt{2} : \texttt{2r} + \texttt{1}] \\
&\texttt{w}[\texttt{2r} + \texttt{1}] := \texttt{S}[\texttt{i} + \texttt{r}] \\
&\texttt{S}[\texttt{i}] := \texttt{F}(\texttt{w}).
\end{aligned}$$

In higher dimensions a similar strategy can be used, depending on whether the stencil is one-dimensional or multi-dimensional. If it is 1D, then one can simply apply the algorithm above along each slice. If it is d-dimensional then a buffer containing $2r + 1$ slices of dimension $d - 1$ is required. In either case, the buffer size is much smaller than a full register.

7.5.3 2S Methods

For the type of spatial discretizations under consideration here, implementation of 2S methods is no more difficult than implementation of 2R methods. The algorithm in 1D is identical except that one assigns

$$\texttt{s}[\texttt{i}] := \texttt{s}[\texttt{i}] + \texttt{F}(\texttt{w})$$

at each point. The extension to multiple dimensions follows the same pattern.

7.6 Improved low-storage methods

In this section we present some new low-storage methods of the types developed in the previous section. This is only intended to demonstrate what is possible; a thorough investigation of 2S methods optimized for various properties, like that done in [66] for 2R methods, is left for future work.

Similarly to [66], we refer to a method as RK-$p(\hat{p})m[X]$, where m is the number of stages, p is the order of accuracy, $\hat{p}$ is the order of accuracy of the embedded method (if any), and X indicates the type of method (2R, 2S, 2S*, etc.).

By writing a three-stage Runge-Kutta method in Shu-Osher form (3.6) and using transformations of the form (7.3) to write the method in the form (7.9), it is seen that any three-stage method may be implemented in 2S form except in the special case that $\beta_{31} = \alpha_{31}\beta_{21}$. In this case the method may be implemented with two registers using a slight modification of the 2S algorithm. Hence all three-stage Runge-Kutta methods can be implemented using two registers.

Method	$A^{(p+1)}$	S_I	S_R
Classical RK4	1.45e-02	0.71	0.70
RK4(3)5[2R+]C	5.12e-03	0.66	0.96
RK4()4[2S]	2.81e-02	0.71	0.70
RK4()6[2S]	4.17e-03	0.60	1.60
RK4()5[2S*]	1.49e-02	0.62	0.67
RK4(3)5[2S]	1.25e-02	0.57	0.56
RK4(3)5[3S*]	5.52e-03	0.67	0.93

Table 7.1: Properties of low-storage methods

Throughout the development of low-storage Runge-Kutta methods, fourth-order methods have been of particular interest [39, 8, 38, 124]. In this section we present several examples of minimum storage fourth order methods. We have also found 2S (and 2S*, embedded, etc.) methods of fifth and sixth orders. As far as we know, these are the first two-register methods of sixth order.

It is known that no generally applicable four-stage fourth order two-register methods of 2N or 2R type exist [124, 66]. This is not surprising, since four-stage methods in those classes have seven free parameters, whereas there are 8 conditions for fourth order accuracy. Four-stage 2S methods, on the other hand, have 9 free parameters, and fourth order methods of this kind exist. An example of such a method is given in Table A.23 as RK4()4[2S].

By allowing additional stages, methods with improved accuracy or stability are possible. These methods can have very good properties compared to 2R or 2N methods because of the additional degrees of freedom available. As an example, we include in Table A.24 a six-stage, fourth order method RK4()6[2S] with improved real-axis stability. In table A.25 we present a 2S* method of fourth order, using five stages. In Table A.26 we present a 4(3)5 2S pair, and in Table A.27 a 4(3)5 3S* pair.

Table 7.1 summarizes the accuracy and stability properties of these methods. The quantities S_I, S_R are the size of the largest interval included in the region of absolute stability along the imaginary and real axes, respectively, scaled (divided) by the number of stages of the method. The quantity $A^{(p+1)}$ is the L_2 principal error norm (i.e., the norm of the vector of leading-order truncation error coefficients). The classical fourth-order RK method and a recommended 2R method from [66] are included for comparison. The new methods all have reasonably good properties.

7.7 Conclusions

We have proposed a new class of low-storage Runge-Kutta methods and given examples of high order methods in this class requiring fewer stages than existing low-storage methods. The methods include embedded pairs using only two memory registers, as well as embedded pairs that retain the previous solution value and use only three memory registers. Such methods were not previously available. A thorough investigation of 2S methods optimized for various properties, like that done in [66] for 2R methods, would be of great utility.

Chapter 8

Numerical Wave Propagation

In this chapter, we introduce a high order accurate semi-discretization for hyperbolic PDEs, based on wave propagation Riemann solvers and high order reconstruction. In Section 8.1, we briefly review exact solution of linear hyperbolic systems and Riemann problems for such systems. In Section 8.2, we present Godunov's method for linear hyperbolic PDEs in wave propagation form. This method is extended to high order in Section 8.3 by introducing a high order reconstruction based on cell averages. The method is generalized to variable-coefficient and nonlinear hyperbolic systems in Section 8.4 and Section 8.5. In Section 8.6, we discuss several approaches to reconstruction of a vector function from componentwise cell averages.

For more information regarding the wave propagation methods on which the approach here is based, see [80].

8.1 Linear Hyperbolic Systems

Our starting point is the one-dimensional linear system

$$q_t + \mathbf{A}q_x = 0. \tag{8.1}$$

Here $q \in \Re^m$ and $\mathbf{A} \in \Re^{m \times m}$. System (8.1) is said to be hyperbolic if $\mathbf{A}$ is diagonalizable with real eigenvalues; we will henceforth assume this to be the case.

Let the eigendecomposition of $\mathbf{A}$ be given by

$$\mathbf{A} = \mathbf{R}\Lambda\mathbf{R}^{-1} \tag{8.2}$$

where Λ is a diagonal matrix. We order the eigenvalues so that $\lambda^1 \leq \lambda^2 \leq \cdots \leq \lambda^m$, and let r^p (l^p) denote the right (left) eigenvector of $\mathbf{A}$ corresponding to λ^p. Multiplying (8.1) on the left by $\mathbf{R}^{-1}$ gives

$$w_t + \Lambda w_x = 0 \tag{8.3}$$

where $w = \mathbf{R}^{-1}q$. System (8.3) is a set of uncoupled advection equations; each characteristic field w^p simply translates with velocity equal to the corresponding eigenvalue λ^p. Thus if we decompose the initial data as

$$q(x,0) = \sum_p w_0^p(x) r^p,$$

the solution at time t is given by

$$q(x,t) = \sum_p w_0^p(x - \lambda^p t) r^p. \tag{8.4}$$

As a special case, consider the *Riemann problem* consisting of (8.1) together with initial data

$$q(x,0) = \begin{cases} q_l & x < 0 \\ q_r & x > 0 \end{cases} \tag{8.5}$$

In this case, the solution may be obtained by decomposing just the difference $q_r - q_l$ in terms of the eigenvectors $\mathbf{A}$:

$$q_r - q_l = \sum_p \alpha^p r^p = \sum_p \mathcal{W}^p. \tag{8.6}$$

We refer to the vectors $\mathcal{W}^p$ as waves. Each wave is a jump discontinuity along the ray $x = \lambda^p t$ in phase space. The solution is pictured in Figure 8.1.

8.2 The Semi-discrete Wave-Propagation form of Godunov's Method

We now describe the well-known numerical method due to Godunov, which is based on the solution to the Riemann problem. Taking a finite volume approach, we define the cell averages

$$Q_i(t) = \frac{1}{\Delta x} \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} q(x,t)dx. \tag{8.7}$$

To solve (8.1), we initially approximate the solution q by these cell averages; that is, we

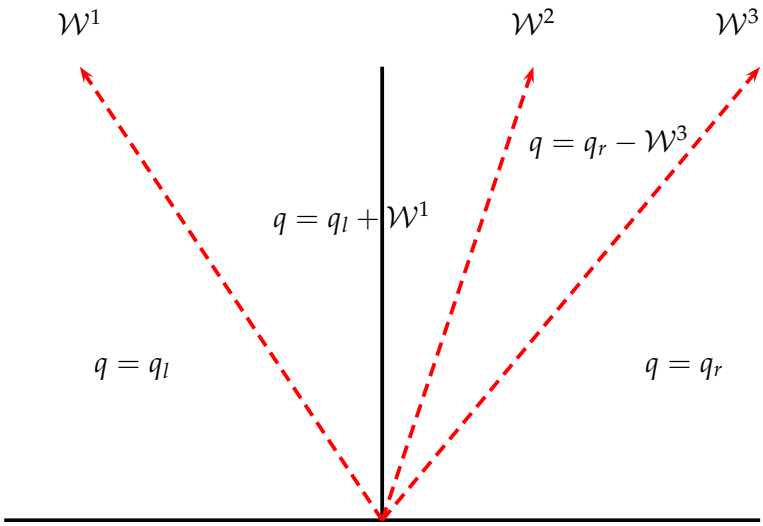


Figure 8.1: The wave propagation solution of the Riemann problem.

define the piecewise function

$$\tilde{q}(x) = \tilde{q}_i(x) \text{ for } x \in (x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}) \tag{8.8}$$

with

$$\tilde{q}_i(x) = Q_i. \tag{8.9}$$

Clearly, (8.1) with initial data $\tilde{q}$ consists of a series of Riemann problems, with a jump at each interface $x_{i-\frac{1}{2}}$. Let $\tilde{q}(x, \Delta t)$ denote the exact evolution of $\tilde{q}$ after a time increment Δt. If we take Δt small enough that the waves from adjacent interfaces do not interact, then we can integrate (8.1) over $[x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}] \times [0, \Delta t]$, and divide by Δx, to obtain

$$Q_i(t + \Delta t) - Q_i(t) = -\frac{1}{\Delta x} \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \mathbf{A}\tilde{q}_x(x, \Delta t) dx \tag{8.10}$$

We can split the integral into three parts, representing the Riemann fans from the two

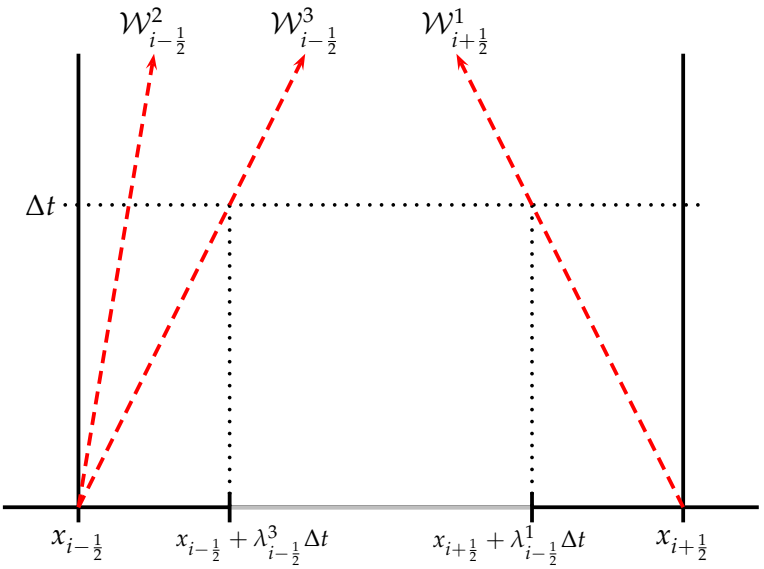


Figure 8.2: Time evolution of the reconstructed solution $\tilde{q}$ in cell i.

interfaces, and the remaining piece:

$$\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \mathbf{A}\tilde{q}_x dx = \int_{x_{i-\frac{1}{2}}}^{x_{i-\frac{1}{2}}+\lambda^r \Delta t} \mathbf{A}\tilde{q}_x dx + \int_{x_{i+\frac{1}{2}}+\lambda^l \Delta t}^{x_{i+\frac{1}{2}}} \mathbf{A}\tilde{q}_x dx + \int_{x_{i-\frac{1}{2}}+\lambda^r \Delta t}^{x_{i+\frac{1}{2}}+\lambda^l \Delta t} \mathbf{A}\tilde{q}_x dx \tag{8.11}$$

$$= \Delta t \sum_{p=1}^{m} \left(\lambda^p_{i-\frac{1}{2}}\right)^+ \mathcal{W}^p_{i-\frac{1}{2}} + \Delta t \sum_{p=1}^{m} \left(\lambda^p_{i+\frac{1}{2}}\right)^- \mathcal{W}^p_{i+\frac{1}{2}}. \tag{8.12}$$

This is illustrated in Figure 8.2. Here we have defined $\lambda^l = \min(\lambda^1_{i+\frac{1}{2}}, 0)$ and $\lambda^r = \max(\lambda^m_{i-\frac{1}{2}}, 0)$, and $(x)\pm$ denotes the positive or negative part of x:

$$(x)^- = \min(x, 0) \qquad (x)^+ = \max(x, 0).$$

The waves $\mathcal{W}_{i-\frac{1}{2}}$ and speeds $\lambda_{i-\frac{1}{2}}$ are those resulting from the Riemann problem with left and right states Q_{i-1} and Q_i, respectively. The third piece vanishes because $\tilde{q}(x, \Delta t)$ is

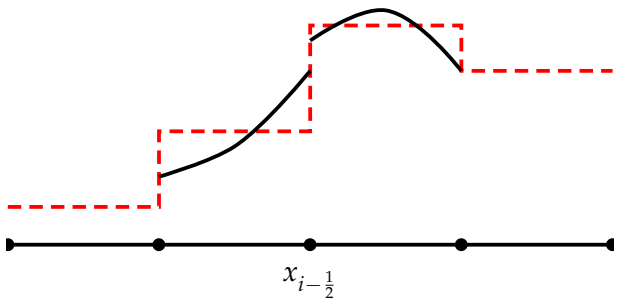


Figure 8.3: Illustration of piecewise polynomial reconstruction from cell averages.

constant outside the Riemann fans. Defining the fluctuations

$$\mathcal{A}^+ \Delta q_{i-\frac{1}{2}} = \sum_{p=1}^{m} \left(\lambda^p_{i-\frac{1}{2}}\right)^+ \mathcal{W}^p_{i-\frac{1}{2}} \tag{8.13}$$

$$\mathcal{A}^- \Delta q_{i+\frac{1}{2}} = \sum_{p=1}^{m} \left(\lambda^p_{i+\frac{1}{2}}\right)^- \mathcal{W}^p_{i+\frac{1}{2}}, \tag{8.14}$$

we have

$$Q_i(t+\Delta t) - Q_i(t) = -\frac{\Delta t}{\Delta x}\left(\mathcal{A}^- \Delta q_{i+\frac{1}{2}} + \mathcal{A}^+ \Delta q_{i-\frac{1}{2}}\right). \tag{8.15}$$

Dividing by Δt and taking the limit as Δt approaches zero, we obtain the semi-discrete scheme

$$\frac{\partial Q_i}{\partial t} = -\frac{1}{\Delta x}\left(\mathcal{A}^- \Delta q_{i+\frac{1}{2}} + \mathcal{A}^+ \Delta q_{i-\frac{1}{2}}\right). \tag{8.16}$$

Equation (8.16) constitues a linear system of ODEs that may be integrated, for instance, with any of the methods discussed in the first part of this thesis.

8.3 Extension to Higher Order

The method of the previous section is only first order accurate in space. In order to improve the spatial accuracy, we replace the piecewise-constant approximation (8.9) by a piecewise-polynomial approximation that is accurate to order p in regions where the solution is smooth:

$$\tilde{q}_i(x) = q(x,t) + \mathcal{O}(\Delta x^{p+1}). \tag{8.17}$$

This reconstruction is illustrated in Figure 8.3.

We again integrate 8.1 over $[x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}] \times [0, \Delta t]$, divide by $\Delta t \Delta x$, and take the limit as

Δt approaches zero. We now find that the third integral in (8.11) contributes, since $\tilde{q}$ is not constant outside the Riemann fans: Defining

$$q^+_{i-\frac{1}{2}} = \tilde{q}_i(x_{i-\frac{1}{2}}) \qquad\qquad q^-_{i+\frac{1}{2}} = \tilde{q}_i(x_{i+\frac{1}{2}}), \tag{8.18}$$

the contribution from this term is

$$\lim_{\Delta t \to 0} \int_{x_{i-\frac{1}{2}}+\lambda^l \Delta t}^{x_{i+\frac{1}{2}}+\lambda^r \Delta t} \mathbf{A}\tilde{q}_x = \mathbf{A}(q^-_{i+\frac{1}{2}} - q^+_{i-\frac{1}{2}}). \tag{8.19}$$

The resulting scheme is thus

$$\frac{\partial Q_i}{\partial t} = -\frac{1}{\Delta x}\left(\mathcal{A}^- \Delta q_{i+\frac{1}{2}} + \mathcal{A}^+ \Delta q_{i-\frac{1}{2}} + \mathbf{A}(q^-_{i+\frac{1}{2}} - q^+_{i-\frac{1}{2}})\right). \tag{8.20}$$

Note that, for instance, the fluctuation $\mathcal{A}^+ \Delta q_{i-\frac{1}{2}}$ corresponds to the effect of right-going waves from the Riemann problem with left state $q^-_{i-\frac{1}{2}}$ and right state $q^+_{i-\frac{1}{2}}$.

8.4 Variable Coefficient Linear Systems

We now generalize the method to solve linear hyperbolic systems with variable coefficients:

$$q_t + \mathbf{A}(x) q_x = 0. \tag{8.21}$$

We assume that the system is hyperbolic for all x and that $\mathbf{A}(x)$ is piecewise-constant, with points of discontinuity aligned with grid interfaces. Thus $\mathbf{A}(x)$ is given by a constant matrix $\mathbf{A}_i$ within grid cell i. The Riemann problem at $x_{i-\frac{1}{2}}$ is now given by (8.21) together with

$$q(x,0) = \begin{cases} q^-_{i-\frac{1}{2}} & x < x_{i-\frac{1}{2}} \\ q^+_{i-\frac{1}{2}} & x > x_{i-\frac{1}{2}} \end{cases} \qquad \mathbf{A}(x) = \begin{cases} \mathbf{A}_{i-1} & x < x_{i-\frac{1}{2}} \\ \mathbf{A}_i & x > x_{i-\frac{1}{2}} \end{cases} \tag{8.22}$$

As in the constant coefficient case, the Riemann solution consists of waves, but now the left-going waves are multiples of the eigenvectors r_{i-1} of $\mathbf{A}_{i-1}$ while the right-going waves are multiples of the eigenvectors r_i of $\mathbf{A}_i$. Thus the semi-discrete scheme is again (8.20), but with fluctuations corresponding to these waves and with $\mathbf{A} = \mathbf{A}_i$.

8.5 Nonlinear Systems

Next we generalize the method to solve general nonlinear hyperbolic systems:

$$q_t + \mathbf{A}(q,x)q_x = 0. \tag{8.23}$$

We again assume that $\mathbf{A}$ is a constant function of x within each cell, so we can write $\mathbf{A}(q,x) = \mathbf{A}_i(q)$. In the special case that $\mathbf{A}$ is the Jacobian matrix of some function f, (8.23) corresponds to a conservation law:

$$q_t + f(q,x)_x = 0. \tag{8.24}$$

Our method can be applied to the general system (8.23) only if a meaningful solution to the Riemann problem can be given. In that case, the scheme is given by

$$\frac{\partial Q_i}{\partial t} = -\frac{1}{\Delta x}\left(\mathcal{A}^-\Delta q_{i+\frac{1}{2}} + \mathcal{A}^+\Delta q_{i-\frac{1}{2}} + \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \mathbf{A}_i(\tilde{q}(x)dx)\right). \tag{8.25}$$

The fluctuations may be computed using a suitable (exact or approximate) Riemann solver. In general, the integral must be evaluated by quadrature; however, in the case of (8.24), the integral can be evaluated exactly, and is given by

$$\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \mathbf{A}_i(\tilde{q}(x)dx) = f(q^-_{i+\frac{1}{2}}) - f(q^+_{i-\frac{1}{2}}). \tag{8.26}$$

Noting that the sum of fluctuations from a Riemann solution are equal to the flux difference between the states involved, we can write the above flux difference as the sum of both fluctuations resulting from the Riemann problem

$$q(x,0) = \begin{cases} q^+_{i-\frac{1}{2}} & x < x_i \\ q^-_{i+\frac{1}{2}} & x > x_i \end{cases} \qquad \mathbf{A}(x) = \mathbf{A}_i. \tag{8.27}$$

Then we can write both (8.20) and (8.25) as

$$\frac{\partial Q_i}{\partial t} = -\frac{1}{\Delta x}\left(\mathcal{A}^-\Delta q_{i+\frac{1}{2}} + \mathcal{A}^+\Delta q_{i-\frac{1}{2}} + \mathcal{A}\Delta q_i\right), \tag{8.28}$$

where $\mathcal{A}\Delta q_i$ is the sum of both fluctuations in the solution of the Riemann problem (8.27).

Note that, in the case of (8.24), if the fluctuations are equal to flux differences

$$\mathcal{A}^- \Delta q_{i-\frac{1}{2}} = \hat{f}_{i-\frac{1}{2}} - f(q^-_{i-\frac{1}{2}}) \tag{8.29}$$

$$\mathcal{A}^+ \Delta q_{i-\frac{1}{2}} = f(q^+_{i-\frac{1}{2}}) - \hat{f}_{i-\frac{1}{2}}, \tag{8.30}$$

(where $\hat{f}_{i-\frac{1}{2}}$ is the numerical flux at $x_{i-\frac{1}{2}}$), then (8.28) is equivalent to the traditional flux-differencing method

$$\frac{\partial Q_i}{\partial t} = -\frac{1}{\Delta x}\left(\hat{f}_{i+\frac{1}{2}} - \hat{f}_{i-\frac{1}{2}}\right). \tag{8.31}$$

In particular, the scheme is conservative in this case.

8.6 High Order Non-oscillatory Reconstruction of Scalar Functions

In this section we discuss the problem of reconstruction of a scalar-valued function from cell averages. We will focus on methods that are able to reconstruct functions with discontinuities without introducing spurious oscillations. For simplicity, we will assume an equispaced grid.

8.6.1 Linear (Non-limited) Reconstruction

Given a stencil of k cells $i-r, \ldots, i-r+k$ and the average values $Q_{i-r}, \ldots, Q_{i-r+k}$, there exists a unique reconstructed polynomial $P_i(x)$ possessing exactly these cell averages. We will be interested mainly in the values of the reconstructed function at the cell interfaces. For a centered stencil and a uniform grid, these values can be written as

$$q^+_{i-\frac{1}{2}} = P_i(x_{i-\frac{1}{2}}) = \sum_{j=-s}^{s} c_j Q_{i+j}, \qquad q^-_{i+\frac{1}{2}} = P_i(x_{i+\frac{1}{2}}) = \sum_{j=-s}^{s} c_j Q_{i-j}. \tag{8.32}$$

Here c_j are fixed weights. For instance, a 3rd order reconstruction using the stencil $\{i-1, i, i+1\}$ is

$$q^+_{i-\frac{1}{2}} = \frac{1}{3}Q_{i-1} + \frac{5}{6}Q_i - \frac{1}{6}Q_{i+1} \tag{8.33a}$$

$$q^-_{i+\frac{1}{2}} = \frac{1}{3}Q_{i+1} + \frac{5}{6}Q_i - \frac{1}{6}Q_{i-1}. \tag{8.33b}$$

We refer to this as a *linear* reconstruction because the reconstructed values are linear combinations of the cell averages.

If a linear reconstruction is used for a stencil that contains a discontinuity, the reconstructed solution will exhibit spurious oscillations. For this reason, when reconstructing

solutions of hyperbolic PDEs, typically some kind of nonlinear *limiting* is performed. In the following sections we discuss limited reconstructions that are designed to avoid oscillations.

8.6.2 TVD Reconstruction

The simplest improvement of the piecewise-constant reconstruction (8.9) is to reconstruct a linear function $\tilde{q}$ in each cell. If we force the resulting reconstruction to have the property that reconstructed values in cell i lie between the cell averages Q_{i-1} and Q_{i+1}, we are led to the well-known class of total variation diminishing (TVD) reconstructions. We will discuss two types of reconstructions; the first is *cell-centered* reconstruction:

$$q^+_{i-\frac{1}{2}} = Q_i - \frac{1}{2}\delta_i \qquad q^-_{i+\frac{1}{2}} = Q_i + \frac{1}{2}\delta_i \tag{8.34}$$

where δ_i is a local approximation to $\frac{\partial q}{\partial x}\Delta x$. For purposes of accuracy, a finite difference approximation could be used. In order to satisfy the TVD property, this approximation is 'limited' in a nonlinear fashion. Many limiters can be written in the form

$$\delta_i = \phi(\theta_i)\Delta Q_{i-\frac{1}{2}} \tag{8.35}$$

where

$$\theta_i = \frac{\Delta Q_{i+\frac{1}{2}}}{\Delta Q_{i-\frac{1}{2}}}. \tag{8.36}$$

For instance, the harmonic van Leer limiter is given by

$$\phi(\theta) = \frac{\theta + |\theta|}{1 + |\theta|} \tag{8.37}$$

.

Interface-centered TVD reconstructions can be written as

$$q^+_{i-\frac{1}{2}} = Q_i - \delta_{i-\frac{1}{2}} \qquad q^-_{i+\frac{1}{2}} = Q_i + \delta_{i+\frac{1}{2}} \tag{8.38}$$

where

$$\delta_{i-\frac{1}{2}} = \phi(\theta_{i-\frac{1}{2}})\Delta Q_{i-\frac{1}{2}} \tag{8.39}$$

with

$$\theta_{i-\frac{1}{2}} = \frac{\Delta Q_{I-\frac{1}{2}}}{\Delta Q_{i-\frac{1}{2}}}. \tag{8.40}$$

Here $I - \frac{1}{2}$ refers to the interface upwind of $i - \frac{1}{2}$.

8.6.3 *Weighted Essentially Non-Oscillatory Reconstruction*

TVD reconstructions are generally at most second order accurate (first order accurate near extrema). We now discuss WENO reconstruction, which achieves higher order accuracy while still generally avoiding spurious oscillations when reconstructing functions with discontinuities or large gradients. For further details on WENO reconstruction, see the recent review paper [108].

For a given integer k, the WENO reconstruction of order $2k-1$ in cell i can be described as follows. First, component reconstructions $p_j(x)$ are formed using linear reconstruction based on each of the k-cell stencils containing cell i. Observe that there are k such stencils, and that the total number of cells in all of the component stencils is $2k-1$. For each component reconstruction, a measure of its smoothness is computed; we will denote the smoothness indicator of the jth component reconstruction by β_j. The smoothness indicator is given by a scaled sum of the squared L^2 norms of all of the derivatives of the reconstructed function:

$$\beta_j = \sum_{l=1}^{k} \Delta x^{2l-1} \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \left(\frac{d^l}{dx^l} p_j(x) \right)^2 dx.$$

Discretely, the smoothness indicators are quadratic functions of the cell averages.

For any interface location $x_{i-\frac{1}{2}}$, there exists a linear combination of these k component reconstructions p_j that is accurate to order $2k-1$:

$$\sum_{l=1}^{k} \gamma_j p_j(x_{i-\frac{1}{2}}) = q^{\text{exact}}(x_{i-\frac{1}{2}}) + \mathcal{O}(\Delta x^{2k-1}).$$

WENO reconstruction uses a different linear combination of the component reconstructions to find the reconstructed value $q^-_{i-\frac{1}{2}}$. The weights are given by

$$\omega_j = \frac{\tilde{\omega}_j}{\sum_{l=1}^{k} \tilde{\omega}_l} \qquad \text{with } \tilde{\omega}_j = \frac{\gamma_j}{(\epsilon + \beta_j)^2}. \tag{8.41}$$

Here ϵ is a small number used only to avoid division by zero. Although a value of 10^{-6} is generally advocated in the literature, we have found that this can have an adverse effect on accuracy in some well-resolved convergence tests. In all numerical tests, we use a value of 10^{-36}.

In order to implement WENO in the context of the vector reconstruction approaches

below, we rewrite the WENO reconstruction in a form analogous to that of TVD reconstruction methods:

$$q^+_{i-\frac{1}{2}} = Q_i - \phi(\theta_{i-\frac{1}{2},2-k}, \ldots, \theta_{i-\frac{1}{2},k-1})\Delta Q_{i-\frac{1}{2}} \tag{8.42}$$

where

$$\theta_{i-\frac{1}{2},j} = \frac{\Delta Q_{i-\frac{1}{2}+j}}{\Delta Q_{i-\frac{1}{2}}}. \tag{8.43}$$

The reconstruction methods we have considered are all symmetric in the sense that

$$q^-_{i+\frac{1}{2}} = Q_i - \phi(\theta_{i+\frac{1}{2},1-k}, \ldots, \theta_{i+\frac{1}{2},k-2})\Delta Q_{i-\frac{1}{2}} \tag{8.44}$$

8.7 Reconstruction of Vector-valued Functions

In this section, we discuss how the reconstruction techniques of the previous section may be extended for systems of equations. In this case q is a vector-valued function. The simplest approach, which we refer to as *component-wise reconstruction*, is to simply apply the scalar reconstruction to each component of q. However, this is generally less accurate or less stable than other approaches that take into account the characteristic structure of $\mathbf{A}$.

8.7.1 Reconstruction of Eigencomponent Coefficients

In the approach above, the nonlinear limiting is applied directly to the components of q. This approach has worked well for many finite volume methods for conservative systems and a range of problems; however, in other cases it is insufficient. In particular, it appears to become successively less satisfactory as the order of accuracy of the reconstruction is increased. See [94] for a detailed discussion with respect to central WENO schemes, for instance.

A somewhat more sophisticated approach, suitable for linear systems (8.21), requires the decomposition of Q_i into eigenvectors of A_i:

$$Q_i = \sum_p w_i^p r_i^p. \tag{8.45}$$

We then form the "wave strengths"

$$\alpha^p_{i-\frac{1}{2}} = w_i^p - w_{i-1}^p. \tag{8.46}$$

Note that these are the strengths of waves that would be obtained in Godunov's method for the Riemann problem at $x_{i-\frac{1}{2}}$ only in the case of a linear system with uniform coefficients. The limiter function ϕ is computed for each characteristic field

$$\phi^p_{i-\frac{1}{2}} = \phi(\theta^p_{i-\frac{1}{2},-s+1}, \ldots, \theta^p_{i-\frac{1}{2},s}), \tag{8.47}$$

using the ratios

$$\theta^p_{i-\frac{1}{2},j} = \frac{\alpha^p_{i-\frac{1}{2}+j}}{\alpha^p_{i-\frac{1}{2}}}. \tag{8.48}$$

The reconstructed values are given by

$$q^+_{i-\frac{1}{2}} = Q_i - \sum_p \phi^p_{i-\frac{1}{2}} \alpha^p_{i-\frac{1}{2}} r^p_i \qquad q^-_{i+\frac{1}{2}} = Q_i + \sum_p \phi^p_{i+\frac{1}{2}} \alpha^p_{i-\frac{1}{2}} r^p_i. \tag{8.49}$$

This approach is ideally suited to linear systems of equations with uniform coefficients, since it corresponds to performing the nonlinear limiting on the decoupled characteristic fields.

For problems with variable coefficients, this method is potentially inaccurate for two reasons. First, it is not always clear how to normalize the eigenvectors in a consistently meaninful way among cells with differing coefficients $\mathbf{A}_i$. Second, the eigenvectors in such cells will not be parallel in phase space, so that comparing their magnitudes may not give a useful indicator of oscillations. However, this method is widely used in finite volume schemes and was used in the original ENO schemes [49].

8.7.2 Characteristic-wise Reconstruction

We now present a method that addresses the two problems mentioned in the previous section. This is similar to the characteristic-wise approach used in Shu's finite volume WENO scheme of [106]. The limiter is computed for each characteristic field using the ratios (8.48), but the wave strengths are calculated differently. For each interface $x_{i-\frac{1}{2}}$, an interface Jacobian $\mathbf{A}_{i-\frac{1}{2}}$ is defined and the jumps $\Delta Q_{j-\frac{1}{2}}$ for $i-k+2 \le j \le i+k-1$ are all decomposed in terms of the eigenvectors of $\mathbf{A}_{i-\frac{1}{2}}$:

$$\Delta Q_{j-\frac{1}{2}} = \sum_p \alpha^p_{j-\frac{1}{2}} r^p_{i-\frac{1}{2}}. \tag{8.50}$$

The reconstructed values are again given by (8.49). The interface Jacobian $\mathbf{A}_{i-\frac{1}{2}}$ may be chosen as a simple average of $\mathbf{A}(\mathbf{q}_i, x_i)$ and $\mathbf{A}(\mathbf{q}_{i-1}, x_{i-1})$, or as something more sophisticated. For systems such as the Euler equations, the Roe average seems to be a good

choice. For linear acoustics, we choose the Jacobian with first eigenvector equal to the first eigenvector of $\mathbf{A}_{i-1}$ and second eigenvector equal to that of $\mathbf{A}_i$. This results in a reconstruction similar in spirit to that of Lax & Liu [85] or Fogarty [37].

8.7.3 Wave-slope Reconstruction

The following method accounts for spatial variation in the coefficients and can conveniently be performed using the existing Riemann solvers in Clawpack.

At each interface $x_{i-\frac{1}{2}}$, the jump $\Delta Q_{i-\frac{1}{2}}$ is decomposed in terms of the eigenvectors of $\mathbf{A}_{i-\frac{1}{2}}$:

$$\Delta Q_{i-\frac{1}{2}} = \sum_p \alpha^p_{i-\frac{1}{2}} r^p_{i-\frac{1}{2}} = \sum_p \mathcal{W}^p_{i-\frac{1}{2}}. \tag{8.51}$$

The reconstructed values are given by

$$q^-_{i-\frac{1}{2}} = Q_{i-1} + \sum_p \phi^p_{i-\frac{1}{2}} \mathcal{W}^p_{i-\frac{1}{2}} \qquad q^+_{i-\frac{1}{2}} = Q_i - \sum_p \phi^p_{i-\frac{1}{2}} \mathcal{W}^p_{i-\frac{1}{2}}. \tag{8.52}$$

with

$$\theta^p_{i-\frac{1}{2}} = \frac{\mathcal{W}^p_{I-\frac{1}{2}} \cdot \mathcal{W}^p_{i-\frac{1}{2}}}{\mathcal{W}^p_{i-\frac{1}{2}} \cdot \mathcal{W}^p_{i-\frac{1}{2}}}. \tag{8.53}$$

This approach is intended to be similar to that used in Clawpack [80]. It was found that this limiting method does not yield consistent improvement over component-wise limiting in practice; for this reason, we do not use wave-slope reconstruction in any of the numerical examples in Chapter 9.

8.8 Extension to Two Dimensions

In this section, we extend the numerical wave propagation method to two dimensions using a simple dimension-by-dimension approach. The method is applicable to systems of the form

$$q_t + \mathbf{A}(q,x,y)q_x + \mathbf{B}(q,x,y)q_y = 0 \tag{8.54}$$

on uniform Cartesian grids.

The 2D analog of the semi-discrete scheme (8.28) is

$$\frac{\partial Q_{ij}}{\partial t} = -\frac{1}{\Delta x \Delta y}\left(\mathcal{A}^-\Delta q_{i+\frac{1}{2},j} + \mathcal{A}^+\Delta q_{i-\frac{1}{2},j} + \mathcal{A}\Delta q_{i,j} + \mathcal{B}^-\Delta q_{i,j+\frac{1}{2}} + \mathcal{B}^+\Delta q_{i,j-\frac{1}{2}} + \mathcal{B}\Delta q_{i,j}\right). \tag{8.55}$$

For the method to be high order accurate, the fluctuation terms like $\mathcal{A}^-\Delta q_{i+\frac{1}{2},j}$ should in-

volve integrals over cell edges, while the total fluctuation terms like $\mathcal{A}\Delta q_{i,j}$ should involve integrals over cell areas. This can be achieved by forming a genuinely multidimensional reconstruction of q and using, e.g., Gauss quadrature. An implementation following this approach was undertaken and exists in the SharpClaw software, but has been found to be extremely inefficient, as it typically yields only a small improvement in accuracy over the dimension-by-dimension scheme given below, but has a much greater computational cost. A careful comparison of the two approaches is left for future work.

We now describe the dimension-by-dimension scheme for a single Runge-Kutta stage. We first reconstruct piecewise-polynomial functions $\tilde{q}_j(x)$ along each row of the grid and $\tilde{q}_i(y)$ along each column, by applying a 1D reconstruction procedure to each slice. We thus obtain reconstructed values

$$\tilde{q}_j^+(x_{i-\frac{1}{2}}) \approx q(x_{i-\frac{1}{2}}, y_j) \tag{8.56a}$$

$$\tilde{q}_j^-(x_{i+\frac{1}{2}}) \approx q(x_{i+\frac{1}{2}}, y_j) \tag{8.56b}$$

$$\tilde{q}_i^+(y_{i-\frac{1}{2}}) \approx q(x_i, y_{i-\frac{1}{2}}) \tag{8.56c}$$

$$\tilde{q}_i^-(y_{i+\frac{1}{2}}) \approx q(x_i, y_{i+\frac{1}{2}}). \tag{8.56d}$$

for each cell i, j. The fluctuation terms in (8.55) are determined by solving Riemann problems between the appropriate reconstructed values; for instance $\mathcal{B}^-\Delta q_{i,j+\frac{1}{2}}$ is determined from the Riemann problem with

$$q_l = q^-_{i,j+\frac{1}{2}} \quad q_r = q^+_{i,j+\frac{1}{2}}.$$

Similarly, in the case of conservative systems or piecewise constant coefficients, the total fluctuation terms $\mathcal{A}\Delta q_{i,j}$ and $\mathcal{B}\Delta q_{i,j}$ can be determined by summing the left- and right-going fluctuations of an appropriate Riemann problem. Thus, for instance, $\mathcal{B}\Delta q_{i,j}$ is determined by the fluctuations resulting from the Rieman problem with

$$q_l = q^+_{i,j-\frac{1}{2}} \quad q_r = q^-_{i,j+\frac{1}{2}}.$$

Chapter 9

Numerical Tests

In this chapter we present results of numerical tests using the wave propagation method of Chapter 8 and the explicit SSP Runge-Kutta methods of Chapter 6 to solve hyperbolic PDEs. The high order wave propagation method will be compared with the well-known TVD wave propagation code Clawpack (see [80]). In Section 9.1, we explain the methods that will be compared. In Section 9.2, we present results for the variable coefficient linear acoustics equations. In Section 9.3, we present results for the Euler equations of compresible fluid flow.

9.1 Methods

In this chapter we will compare numerical results obtained using the following methods:

Clawpack. Wave propagation method of LeVeque [80]. In all tests, we use a CFL number of 0.9 and the monotonized centered limiter.

UC3, UC7. High order wave propagation with un-limited centered reconstruction of third or seventh order accuracy.

TVD2. High order wave propagation with component-wise TVD (monotonized centered) reconstruction.

WENO5. High order wave propagation with component-wise fifth-order WENO reconstruction.

WENO5 Char. High order wave propagation with characteristic-wise fifth-order WENO reconstruction.

All of the high order methods are integrated using SSPRK(10,4) with a CFL number of 2.85.

9.2 Acoustics

In this section, we apply the high-order wave propagation methods of Chapter 8 to linear acoustics in piecewise homogeneous materials. The acoustics equations in one dimension are

$$p_t + K(x)u_x = 0 \tag{9.1a}$$
$$u_t + \frac{1}{\rho(x)}p_x = 0 \tag{9.1b}$$

where p, u are pressure and velocity perturbations (respectively) relative to some ambient state. This system is of the form (8.21), with

$$q = \begin{pmatrix} p \\ u \end{pmatrix}, \qquad \mathbf{A}(x) = \begin{pmatrix} 0 & K(x) \\ 1/\rho(x) & 0 \end{pmatrix} \tag{9.2}$$

The eigenvectors of the matrix $\mathbf{A}_i$ in this case are

$$r_i^1 = \begin{pmatrix} -Z_i \\ 1 \end{pmatrix}, \qquad r_i^2 = \begin{pmatrix} Z_i \\ 1 \end{pmatrix} \tag{9.3}$$

and the eigenvalues are

$$\lambda_i^1 = -c_i, \qquad \lambda_i^2 = c_i. \tag{9.4}$$

Here $Z_i = \sqrt{K_i\rho_i}$ is the impedance and $c_i = \sqrt{K_i/\rho_i}$ is the sound speed. For the solution of the Riemann problem, and also for characteristic-wise decomposition, we will make use of the matrix $\mathbf{A}_{i-\frac{1}{2}}$ with eigenvectors

$$r_{i-\frac{1}{2}}^1 = \begin{pmatrix} -Z_{i-1} \\ 1 \end{pmatrix}, \qquad r_{i-\frac{1}{2}}^2 = \begin{pmatrix} Z_i \\ 1 \end{pmatrix} \tag{9.5}$$

and eigenvalues

$$\lambda_{i-\frac{1}{2}}^1 = -c_{i-1}, \qquad \lambda_{i-\frac{1}{2}}^2 = c_i. \tag{9.6}$$

Thus the solution to the Riemann problem at $x_{i-\frac{1}{2}}$ consists of a left-moving wave $\alpha^1_{i-\frac{1}{2}} r^1_{i-1}$ with velocity $-c_{i-1}$ and a right-moving wave $\alpha^2_{i-\frac{1}{2}} r^2_i$ with velocity c_i. The wave strengths are found to be

$$\alpha^1_{i-\frac{1}{2}} = \frac{-\Delta p_{i-\frac{1}{2}} + Z_i \Delta u_{i-\frac{1}{2}}}{Z_i + Z_{i-1}} \tag{9.7a}$$

$$\alpha^2_{i-\frac{1}{2}} = \frac{\Delta p_{i-\frac{1}{2}} + Z_{i-1} \Delta u_{i-\frac{1}{2}}}{Z_i + Z_{i-1}}. \tag{9.7b}$$

9.2.1 Single Material Interface

In this section, we study a test problem involving a single interface between two materials. Thus we solve (9.1) with

$$\rho, c = \begin{cases} \rho_l, c_l & x < 0 \\ \rho_r, c_r & x > 0 \end{cases} \tag{9.8}$$

We will measure the convergence rate of the solution in order to determine a practical order of accuracy for smooth solutions. We thus require an initial condition that is p-times differentiable, where p is greater than the expected order of accuracy. Additionally, we require that the initial condition have compact support so that after some time the solution will again be sufficiently smooth. We therefore use as initial condition the Newton interpolating polynomial

$$p(x,0) = \frac{((x-x_0)-a)^6((x-x_0)+a)^6}{a^{12}} \xi(x-x_0) \tag{9.9}$$

$$u(x,0) = p(x,0)/Z(x) \tag{9.10}$$

where

$$\xi(x-x_0) = \begin{cases} 0 & (x - x_0 < -a) \\ 1 & (-a \le x - x_0 \le a) \\ 0 & (x - x_0 > a) \end{cases} \tag{9.11}$$

with $x_0 = -4$ and $a = 1$. This function is everywhere six times differentiable and is identically zero at the interface. Since the solution is a multiple of $r^2(x)$ at every value of x, initially the wave is purely right-going. The evolution of the exact solution for two sets of material parameters is illustrated in Figure 9.1. When the wave reaches the interface, part is reflected and part transmitted. Although the solution is not differentiable during the interaction with the interface, at later times it is once again six-times differentiable. Ideally, we would like to recover a high order of accuracy with our numerical scheme,

even after the wave has passed through the interface.

As a first verification of the implementation, we consider a homogeneous medium by taking $\rho_l = c_l = \rho_r = c_r = 1$. Table 9.1 shows results for three different reconstruction methods and Clawpack. In each case, the order of convergence is approximately equal to the theoretical order of accuracy of the reconstruction, except for UC7. In this case the spatial error is small enough that the error of the time discretization dominates.

Table 9.1: Errors for homogeneous problem

	TVD2		WENO5		UC7		Clawpack	
mx	Error	Order	Error	Order	Error	Order	Error	Order
200	1.69e-01	nan	3.60e-02	nan	6.41e-03	nan	4.10e-02	nan
400	5.95e-02	1.50	3.65e-03	3.30	2.83e-04	4.50	1.30e-02	1.66
800	2.22e-02	1.42	1.85e-04	4.31	1.63e-05	4.11	3.61e-03	1.85
1600	6.67e-03	1.74	7.35e-06	4.65	1.01e-06	4.02	8.94e-04	2.01
3200	1.83e-03	1.87	2.72e-07	4.76	6.25e-08	4.01	2.19e-04	2.03

For the next test we take

$$\rho_l = c_l = 1 \quad \rho_r = 4 \quad c_r = 1/2,$$

yielding an impedance ratio of two. Results are shown in Table 9.2. In this case the methods UC7 and WENO5 show convergence rates well below their formal order, even though the initial and final solutions are smooth. To investigate this further, we repeat the same test with a wider pulse by taking $a = 4$. Results are shown in table 9.3.

For the latter test, we observe a convergence rate of approximately two for all methods. The results can be understood as follows. For this problem, the errors can be divided into two sources. First, the truncation error that occurs as the pulse is propagated in a homogeneous medium; this error is of the order given by the design order of each method. This error is large for a narrow pulse that is not well resolved on the grid.

The second source of error arises in the reconstruction step, when the solution is reconstructed using stencils that cross the material interface. The high-order reconstruction is based on an assumption of smoothness of the solution, which does not hold at the interface. Since the jump in the first derivative of the solution at the interface is $\mathcal{O}(1)$, the error in the reconstructed values in cells whose stencil overlaps the interface is $\mathcal{O}(\Delta x)$. Since the total area of all such cells is $\mathcal{O}(\Delta x)$, the resulting global error is $\mathcal{O}(\Delta x^2)$. This

error dominates when the pulse is well-resolved on the grid, so that the first type of error is small.

Table 9.2: Errors for interface 1 problem

	TVD2		WENO5		UC7		Clawpack	
mx	Error	Order	Error	Order	Error	Order	Error	Order
200	2.73e-01	nan	2.10e-01	nan	6.76e-02	nan	1.98e-01	nan
400	9.70e-02	1.50	5.98e-02	1.81	5.05e-03	3.74	7.26e-02	1.45
800	3.56e-02	1.45	1.25e-02	2.26	1.01e-03	2.31	2.21e-02	1.71
1600	1.17e-02	1.61	1.17e-03	3.42	2.51e-04	2.01	7.86e-03	1.49
3200	3.35e-03	1.80	1.39e-04	3.07	6.28e-05	2.00	3.18e-03	1.31

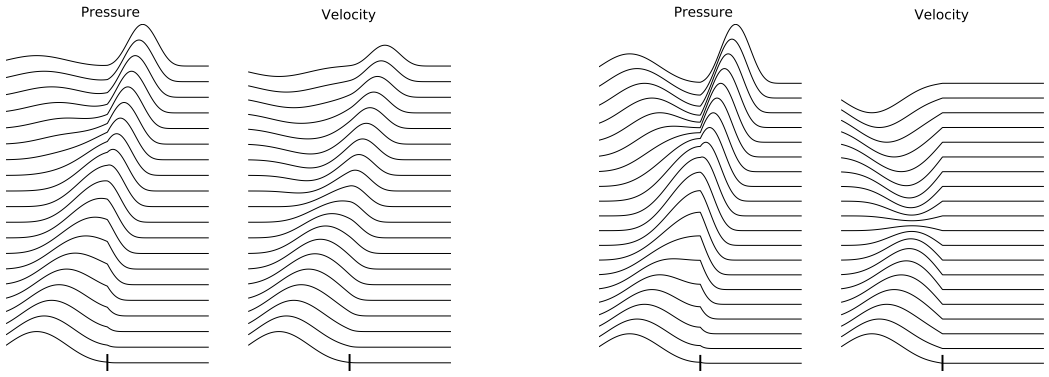
Table 9.3: Errors for interface 1 problem with wide pulse (a=4)

	TVD2		WENO5		UC7		Clawpack	
mx	Error	Order	Error	Order	Error	Order	Error	Order
200	4.70e-02	nan	9.67e-03	nan	4.04e-03	nan	5.23e-02	nan
400	1.30e-02	1.85	2.01e-03	2.27	1.01e-03	2.01	2.32e-02	1.17
800	3.46e-03	1.91	4.89e-04	2.04	2.51e-04	2.00	1.09e-02	1.09
1600	8.91e-04	1.96	1.22e-04	2.00	6.28e-05	2.00	5.26e-03	1.05
3200	2.26e-04	1.98	3.04e-05	2.00	1.57e-05	2.00	2.58e-03	1.02

For the next test we take

$$\rho_l = c_l = 1 \quad \rho_r = 4000 \quad c_r = 1/2,$$

yielding an impedance ratio of 2000. This is a much more difficult problem. The solution has a larger jump in the first derivative while the pulse is passing through the interface (see Figure 9.1). The linear reconstruction methods UC3 and UC7 fail completely (are unstable) in this case. Again we use two different initial conditions to illustrate the two different types of errors. Results are shown in Tables 9.4 and 9.5. We see that the WENO5 methods are significantly more accurate than the 2nd order methods, and that using characteristic-wise decomposition improves the accuracy further. Between the 2nd order



(a) $\rho_r = 4, c_r = 1/2$ (b) $\rho_r = 4000, c_r = 1/2$

Figure 9.1: Evolution of an acoustic pulse at an interface, for two different values of density of the right-side material. In both cases, $\rho_l = c_l = 1$. The exact solution is plotted for the interval $x \in [-1, 1]$. Note the more significant lack of smoothness at $x = 0$ in the case of large jump in parameters.

methods, Clawpack is more accurate when the traditional truncation error dominates, while TVD2 is more accurate when the interface error dominates.

Table 9.4: Errors for interface 2 problem

	TVD2		WENO5		WENO5 Char		Clawpack	
mx	Error	Order	Error	Order	Error	Order	Error	Order
200	5.32e-01	nan	3.60e-01	nan	2.58e-01	nan	3.18e-01	nan
400	1.94e-01	1.45	9.72e-02	1.89	2.50e-02	3.37	1.16e-01	1.45
800	6.75e-02	1.52	1.97e-02	2.30	3.20e-03	2.97	3.52e-02	1.72
1600	2.11e-02	1.68	1.96e-03	3.33	3.56e-04	3.17	1.22e-02	1.53
3200	5.93e-03	1.83	2.72e-04	2.85	7.37e-05	2.27	4.85e-03	1.33

Table 9.5: Errors for interface 2 problem with wide pulse

	TVD2		WENO5		WENO5 Char		Clawpack	
mx	Error	Order	Error	Order	Error	Order	Error	Order
200	9.73e-02	nan	1.75e-02	nan	6.60e-03	nan	8.23e-02	nan
400	2.53e-02	1.94	4.02e-03	2.12	1.18e-03	2.49	3.55e-02	1.22
800	6.42e-03	1.98	9.91e-04	2.02	2.92e-04	2.01	1.65e-02	1.11
1600	1.61e-03	2.00	2.47e-04	2.00	7.34e-05	1.99	7.91e-03	1.06
3200	4.07e-04	1.98	6.17e-05	2.00	1.84e-05	2.00	3.88e-03	1.03

9.2.2 Several Interfaces

We now consider a medium with periodic material parameters. We use the material considered in [103], for which

$$(\rho(x), K(x)) = \begin{cases} (1,1) & 0 < x \bmod L < \theta L \\ (3,3) & \theta L < x \bmod L < L \end{cases}, \tag{9.12}$$

and we take $\theta = 1/2, L = 4$. As above we consider the domain $[-10, 10]$ and initial condition (9.9), now with $x_0 = -1$, $a = 1$. The exact solution is as smooth as the initial condition at integer times; we compare computed and exact solutions at $t = 8$. Results are shown in Table 9.6. We see that the interface error dominates the convergence rate now even though the pulse is not so wide. The errors for most of the methods are similar to but larger than those for a single interface (compare with Tables 9.2 and 9.4). However, the WENO5 Char method converges much more slowly on this problem; in fact, it is less accurate than the component-wise WENO5 method on the finer grids. This problem (with a different domain and initial condition) was also considered in [37].

9.2.3 A Sonic Crystal

In this section we model sound propagation in a sonic crystal. A sonic crystal is a periodic structure composed of materials with different sounds speeds and impedances. The periodic inhomogeneity can give rise to *bandgaps* – frequency bands that are completely reflected by the crystal. This phenomenon is widely utilized in photonics, but its significance for acoustics has only recently been considered. Photonic crystals can be analyzed quite accurately using analytic techniques, since they are essentially infinite size struc-

Table 9.6: Errors for periodic problem

	TVD2		WENO5		WENO5 Char		Clawpack	
mx	Error	Order	Error	Order	Error	Order	Error	Order
500	6.11e-01	nan	3.26e-01	nan	3.02e-01	nan	1.51e-01	nan
400	2.14e-01	1.51	6.61e-02	2.30	4.82e-02	2.65	5.89e-02	1.36
800	6.57e-02	1.71	9.20e-03	2.84	1.21e-02	1.99	2.52e-02	1.22
1600	1.84e-02	1.84	1.86e-03	2.31	2.57e-03	2.24	1.16e-02	1.13
3200	4.92e-03	1.90	4.29e-04	2.12	5.71e-04	2.17	5.55e-03	1.06

tures relative to the wavelength of the waves of interest. In contrast, sonic crystals are typically only a few wavelengths in size, so that the effects of their finite size cannot be neglected. For more information on sonic crystals, see for instance the review paper [90].

We consider a square array of square rods in air with a plane wave disturbance incident parallel to one of the axes of symmetry. The array is infinitely wide but only five periods deep. The lattice spacing is 10 cm and the rods have a cross-sectional side length of 4 cm, so that the filling fraction is 0.16. This crystal is similar to one studied in [101], and it is expected that sound waves in the 1200-1800 Hz range will experience severe attenuation in passing through it, while longer wavelengths will not be significantly attenuated.

A numerical instability very similar to that observed in 1D simulations in [36, 37] was observed when the standard Clawpack method was applied to this problem. The WENO5 Char method with characteristic- wise limiting showed no such instability.

Figure 9.2 shows the RMS pressure for a plane wave with $k = 15$ incident from the left. This wave has a frequency of about 800 Hz, well below the partial band gap. As expected, the wave passes through the crystal without significant attenuation. In Figure 9.3, the pressure is plotted along a slice in the x-direction approximately midway between rows of rods.

Figure 9.4 shows the RMS pressure for an incident plane wave with with frequency 1600 Hz, inside the partial bandgap. Notice that the wave is almost entirely reflected, resulting in a standing wave in front of the crystal. Figure 9.5 shows the rms pressure along a slice in the x-direction.

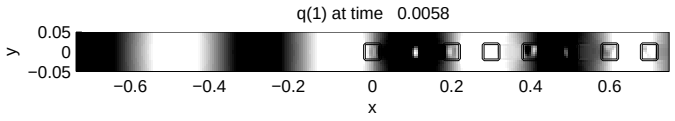


Figure 9.2: Pressure in the sonic crystal for a long wavelength plane wave incident from the left.

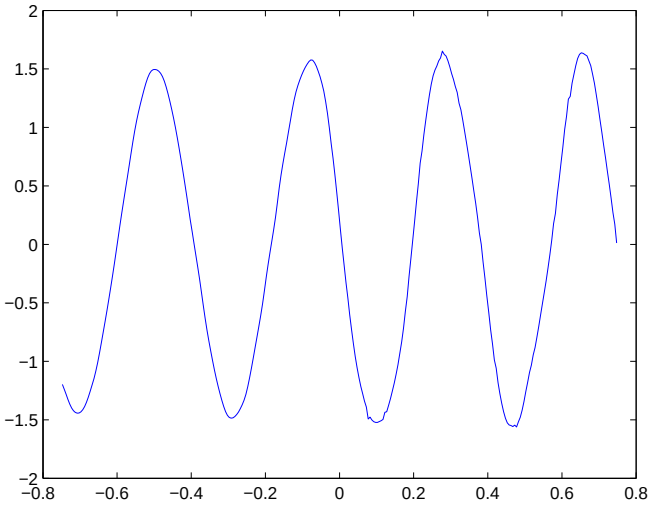


Figure 9.3: Pressure in the sonic crystal for a long wavelength plane wave incident from the left.

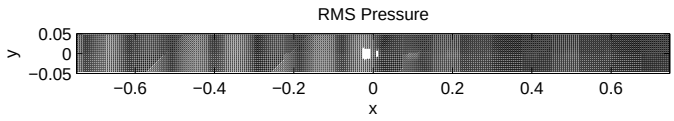


Figure 9.4: RMS pressure in the sonic crystal for a plane wave incident from the left.

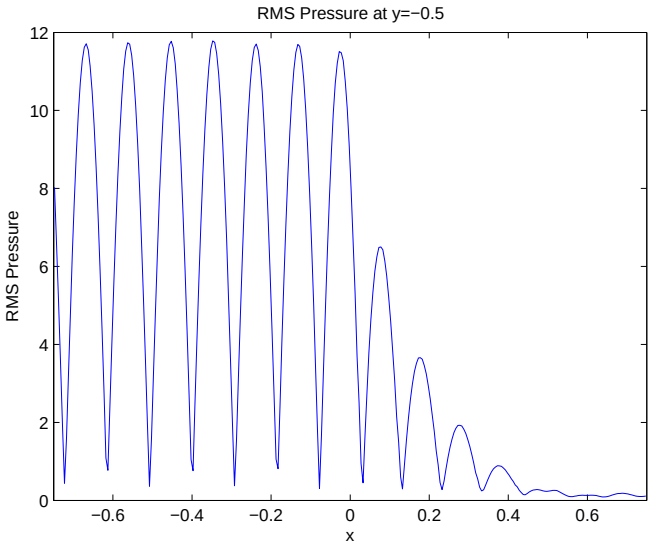


Figure 9.5: RMS pressure in the sonic crystal along a slice at y=-0.05.

9.3 Fluid Dynamics

We now consider the one-dimensional Euler equations of fluid dynamics:

$$\rho_t + (\rho u)_x = 0 \tag{9.13}$$

$$(\rho u)_t + (\rho u^2 + p)_x = 0 \tag{9.14}$$

$$E_t + ((E + p)u)_x = 0 \tag{9.15}$$

where ρ, u, p, and E are the density, velocity, pressure, and total energy, respectively. It is important that the solution satisfy the physical constraints that the density, pressure, and internal energy remain positive. This can be very difficult to maintain numerically. Often, WENO spatial discretization and SSP Runge-Kutta time discretization are applied to this system with the intent of better preserving positivity. However since the WENO semi-discretization of (9.13) is not provably TVD, the theory of strong stability preservation does not strictly apply. Nevertheless, good results are generally observed.

In this section we apply our wave propagation method to investigate the usefulness of SSP methods for integrating fifth-order WENO discretizations of (9.13). We discretize using component-wise fifth order WENO reconstruction and the two-wave HLL Riemann solver. We consider a Riemann problem similar to Test 2 of [116, Section 8.5], with pressure and density equal to 1 everywhere and $u_r = 3.1, u_l = -3.1$. The solution consists of two rarefaction waves and a near-vacuum state in the middle. It is thus a very challenging problem in terms of maintaining positivity.

We solve the problem using several time integrators and determine the largest CFL number that maintains positive density and pressure. Results are given in Table 9.7. The NSSP methods are so-called non-SSP methods proposed in [121]. Method RK(4,4) is the classical fourth order method and RK(6,5) is a fifth-order method of Butcher [12]. We see that, in general, the SSP methods allow a significantly larger timestep.

Table 9.7: Largest positivity-preserving timestep for double-rarefaction problem.

Method	$\Delta t/\Delta x$	$\Delta t/s\Delta x$
NSSP(2,1)	0.14	0.070
NSSP(3,2)	0.40	0.133
NSSP(3,3)	0.26	0.087
NSSP(5,3)	0.50	0.100
RK(4,4)	0.77	0.193
RK(6,5)	0.60	0.100
SSP(3,3)	0.77	0.257
SSP(4,3)	1.13	0.283
SSP(9,3)	2.33	0.259
SSP(5,4)	0.95	0.190
SSP(10,4)	2.70	0.270

Chapter 10

Stegotons

In this chapter we present numerical and analytical investigations of a class of solitary waves known as stegotons. These waves, originally discovered in [82] and studied in [84], arise from the interaction of nonlinearity and an effective dispersion due to material interfaces in layered media. In Section 10.1, we review previous work on stegotons. In Section 10.2, we investigate the stegotons through analysis of homogenized equations, and relate those equations to other interesting PDEs. In Section 10.4, we investigate solitary waves in smoothly varying periodic elastic media.

10.1 Previous Work

10.1.1 Nonlinear Elasticity in 1D

Elastic compression waves in one dimension are governed by the equations

$$\epsilon_t(x,t) - u_x(x,t) = 0 \tag{10.1a}$$

$$(\rho(x)u(x,t))_t - \sigma(\epsilon(x,t),x)_x = 0. \tag{10.1b}$$

where ϵ is the strain, u the velocity, ρ the density, and σ the stress. This is a conservation law of the form (8.24), with

$$q(x,t) = \begin{pmatrix} \epsilon \\ \rho(x)u \end{pmatrix} \qquad f(q,x) = \begin{pmatrix} -u \\ -\sigma(\epsilon,x) \end{pmatrix}. \tag{10.2}$$

Note that the density and the stress-strain relationship vary in x. We will also refer to the sound speed $c(x)$, impedance $Z(x)$, and linearized bulk modulus $K(x)$, given by

$$c(x) = \sqrt{\sigma(\epsilon, x)_\epsilon / \rho(x)} \tag{10.3}$$

$$Z(x) = \rho(x)c(x) \tag{10.4}$$

$$K(x) = \sigma(\epsilon, x)_\epsilon|_{\epsilon=0}. \tag{10.5}$$

The Jacobian of the flux function is

$$f'(q) = \begin{pmatrix} 0 & -1/\rho(x) \\ -\sigma(\epsilon, x)_\epsilon & 0 \end{pmatrix}, \tag{10.6}$$

with eigenvectors

$$r^1 = \begin{pmatrix} 1 \\ -Z(q, x) \end{pmatrix}, \qquad r^2 = \begin{pmatrix} 1 \\ Z(q, x) \end{pmatrix} \tag{10.7}$$

In the case of the linear stress-strain relation $\sigma(x) = K(x)\epsilon(x)$, (10.1) is just the one-dimensional wave equation, and is equivalent to the acoustics equations studied in the last chapter. In this chapter we are interested in studying phenomena that arise in the presence of a nonlinear stress-strain relationship and a periodically varying medium.

10.1.2 An F-wave Riemann Solver

To apply the wave propagation method of Chapter 8 to (10.1), we need to define a Riemann solver for this system. We will use the f-wave solver used in [82]. We assume that ρ, σ are independent of x in each cell, so they can be written as $\rho(x) = \rho_i, \sigma(\epsilon, x) = \sigma_i(\epsilon)$. Given a Riemann problem at $x_{i-\frac{1}{2}}$ we use the approximate wave speeds

$$s^1_{i-\frac{1}{2}} = -\sqrt{\frac{\sigma'_{i-1}(\epsilon^-_{i-\frac{1}{2}})}{\rho^-_{i-\frac{1}{2}}}} \qquad s^2_{i-\frac{1}{2}} = \sqrt{\frac{\sigma'_i(\epsilon^+_{i-\frac{1}{2}})}{\rho^+_{i-\frac{1}{2}}}}. \tag{10.8}$$

and the eigenvectors

$$r^1_{i-\frac{1}{2}} = \begin{pmatrix} 1 \\ -\sqrt{\rho^-_{i-\frac{1}{2}}\sigma'_{i-1}(\epsilon^-_{i-\frac{1}{2}})} \end{pmatrix}, \qquad r^2_{i-\frac{1}{2}} = \begin{pmatrix} 1 \\ \sqrt{\rho^+_{i-\frac{1}{2}}\sigma'_i(\epsilon^+_{i-\frac{1}{2}})} \end{pmatrix}. \tag{10.9}$$

We decompose the flux difference as

$$f(q^+_{i-\frac{1}{2}}) - f(q^-_{i-\frac{1}{2}}) = \beta^1_{i-\frac{1}{2}} r^1_{i-\frac{1}{2}} + \beta^2_{i-\frac{1}{2}} r^2_{i-\frac{1}{2}}. \tag{10.10}$$

Then the fluctuations are simply

$$\mathcal{A}^- \Delta q_{i-\frac{1}{2}} = \beta^1_{i-\frac{1}{2}} r^1_{i-\frac{1}{2}} \qquad \mathcal{A}^+ \Delta q_{i-\frac{1}{2}} = \beta^2_{i-\frac{1}{2}} r^2_{i-\frac{1}{2}} \tag{10.11}$$

To begin, we consider the piecewise constant medium studied in [82, 84], with exponential stress-strain relation

$$\sigma(\epsilon, x) = \exp(K(x)\epsilon) - 1 \tag{10.12}$$

and

$$(\rho(x), K(x)) = \begin{cases} (\rho_A, K_A) & \text{if } j\delta < x < (j+\alpha)\delta \text{ for some integer } j \\ (\rho_B, K_B) & \text{otherwise.} \end{cases} \tag{10.13}$$

We take $\delta = 1, \alpha = 1/2$, and

$$\rho_A = 4 \qquad K_A = 4 \tag{10.14}$$

$$\rho_B = 1 \qquad K_B = 1. \tag{10.15}$$

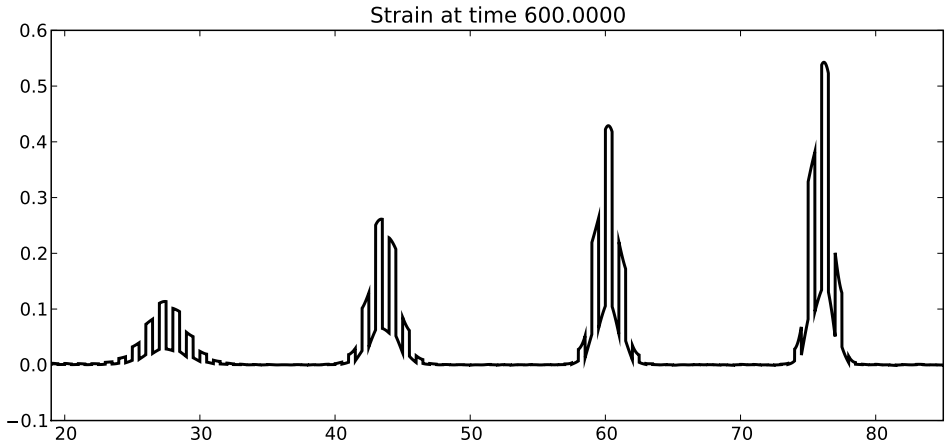
The initial condition is uniformly zero, and the boundary condition at the left generates a half-cosine pulse.

As discussed extensively in [82, 83, 84], the initial pulse breaks up into a train of solitary waves. These waves all have similar shape (under an appropriate rescaling), and appear to interact like solitons, though it is not clear whether they are solitons in the strict mathematical sense. Figure 10.1 shows the result of a very highly resolved simulation (192 cells per layer) of this phenomenon.

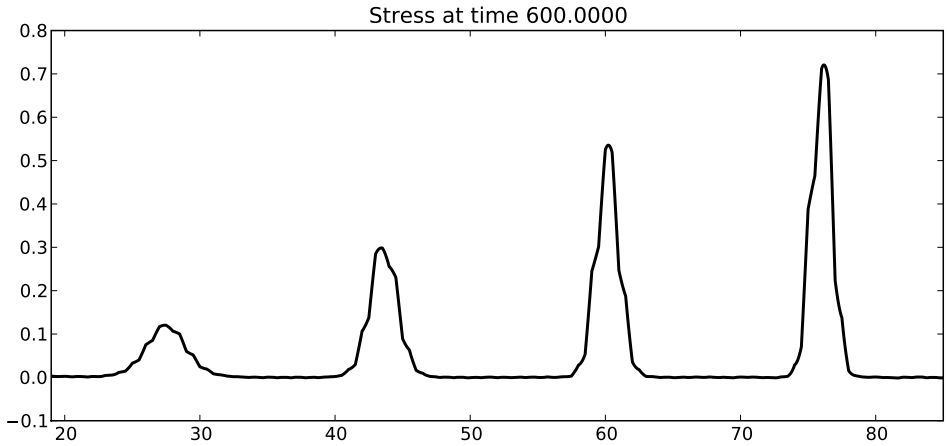
Figure 10.2 shows a comparison of results using Clawpack and our high order wave propagation method (WENO5 of Chapter 9) on this problem, with only 24 cells per layer. The WENO5 results are significantly more accurate.

10.1.3 Homogenized Equations

These solitary waves can be understood by examining the behavior of long-wavelength waves in the periodic medium. To first order, the effect of the layering is an effective dispersion. When this balances with the steepening due to nonlinearity, solitary waves

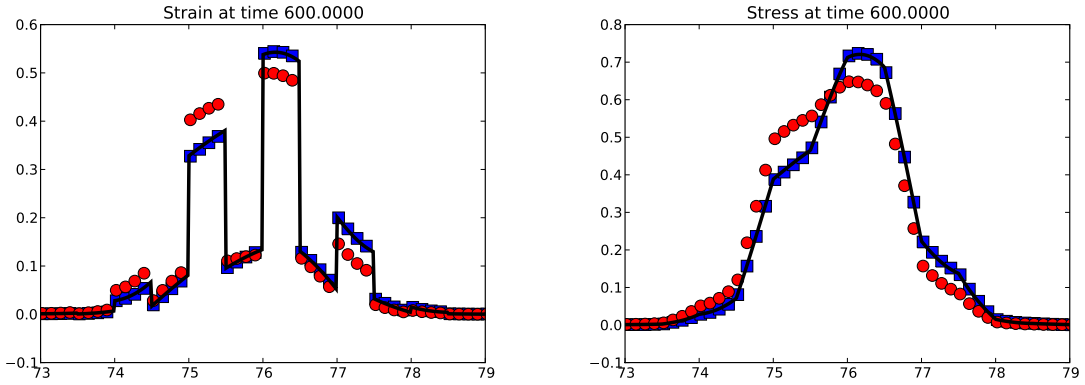


(a) Stress



(b) Stress

Figure 10.1: Stegotons



(a) Strain (b) Stress

Figure 10.2: Comparison of Clawpack (red circles) and WENO5 (blue squares) solution of the stegoton problem using 24 cells per layer. For clarity, only every third solution point is plotted. The black line represents a very highly resolved solution.

arise. The remarkable aspect of these solitary waves is that they satisfy an equation with no explicit dispersion; rather, the dispersion is an effect of the spatially varying coefficients.

To understand this further, LeVeque & Yong derived a system of effective or homogenized equations that describe the evolution of stegotons. This is done by first changing variables to write (10.1) as a pair of evolution equations for σ and u, which are continuous even across material interfaces:

$$\sigma_t - K(x)G(\sigma)u_x = 0 \tag{10.16}$$

$$\rho(x)u_t - \sigma_x = 0. \tag{10.17}$$

Here $G(\sigma) = \sigma_\epsilon(\epsilon, x)/K(x)$. In the case of the exponential stress-strain relationship (10.12), $G(\sigma) = \sigma + 1$. The homogenized equations are written in terms of σ and u, since these variables are continuous across material interfaces (whereas $\epsilon, \rho u$ are not). Introducing a fine spatial scale δ and performing an asymptotic expansion of the solution in powers of δ, one obtains a system of equations with homogeneous coefficients that describes the evolution of long-wavelength waves in this medium. To order δ^2, the resulting

equations are [84]

$$\sigma_t = \hat{K}\left(G(\sigma)u_x - \delta C_{11}G(\sigma)u_{xx} - \delta^2 C_{12}G(\sigma)u_{xxx} - \delta^2 C_{13}\left(G'(\sigma)\sigma_x u_{xx} + \frac{1}{2}G''(\sigma)\sigma_x^2 u_x\right)\right) + \mathcal{O}(\delta^3), \tag{10.18a}$$

$$\rho_t = \frac{1}{\hat{\rho}}\left(\sigma_x - \delta C_{21}\sigma_{xx} - \delta^2 C_{22}\sigma_{xxx}\right)\mathcal{O}(\delta^3) \tag{10.18b}$$

For the case of a piecewise-constant bilayered medium, the coefficients in (10.18) are given by (note that the equation for C_{13} contains a small typo in [84]):

$$C_{11} = C_{21} = 0 \tag{10.19a}$$

$$C_{12} = -\frac{1}{12}\alpha^2(1-\alpha)^2\frac{(\rho_A - \rho_B)(Z_A^2 - Z_B^2)}{K_A K_B \hat{K}^{-1}\hat{\rho}^2} \tag{10.19b}$$

$$C_{22} = -\frac{1}{12}\alpha^2(1-\alpha)^2\frac{(K_A - K_B)(Z_A^2 - Z_B^2)}{K_A^2 K_B^2 (\hat{K}^{-1})^2\hat{\rho}} \tag{10.19c}$$

$$C_{13} = -\frac{1}{12}\alpha^2(1-\alpha)^2\frac{\hat{\rho}^2(K_A - K_B)^2 + (Z_A^2 - Z_B^2)^2}{K_A^2 K_B^2 (\hat{K}^{-1})^2\hat{\rho}^2}. \tag{10.19d}$$

10.2 Analysis of the Homogenized Equations

In this section we investigate two simplifications of the homogenized equations (10.18). These simplifications are obtained by choosing special values for the coefficients C_{ij}.

10.2.1 Reduced Equations and Phase-Plane Analysis

In this section we introduce a simplified system of equations based on (10.18). This system produces the same qualitative behavior, but is more amenable to analysis. To this end, consider (10.18) with $C_{11} = C_{21} = C_{22} = C_{13} = 0$:

$$\sigma_t - (\sigma+1)\hat{K}u_x = -(\sigma+1)\hat{K}C_{12}u_{xxx} \tag{10.20a}$$

$$\hat{\rho}u_t - \sigma_x = 0. \tag{10.20b}$$

Note that the terms on the left correspond to the original equations with homogenized coefficients. By adding the single nonlinear dispersive term on the right, solitary wave solutions arise. To see this, we use a traveling wave ansatz:

$$\sigma(x,t) + 1 = W(x - Vt) \tag{10.21a}$$

$$u(x,t) = U(x - Vt). \tag{10.21b}$$

Here V is a constant velocity. Substituting these into (10.20) gives

$$-VW' = \hat{K}(WU' - C_{12}WU''') \tag{10.22a}$$

$$-\hat{\rho}VU' = W'. \tag{10.22b}$$

Using the second equation we can eliminate U in the first to obtain

$$W' = \frac{\hat{K}}{\hat{\rho}V^2}(WW' - C_{12}WW''') \tag{10.23a}$$

Since

$$WW' = \left(\frac{1}{2}W^2\right)' \tag{10.24a}$$

$$WW''' = (WW'')' - W'W'' = \left(WW'' - \frac{1}{2}W'^2\right)', \tag{10.24b}$$

we can integrate by parts to obtain (defining the mach number $M = \sqrt{\frac{\hat{\rho}}{\hat{K}}}V = \frac{V}{\hat{c}}$)

$$W = \frac{1}{M^2}\left(\frac{1}{2}W^2 - C_{12}\left(WW'' - \frac{1}{2}W'^2\right)\right) + \gamma$$

The constant of integration γ is determined by the physical boundary conditions at ∞: $W \to 1, W' \to 0$. This yields

$$\gamma = 1 - \frac{1}{2M^2}.$$

Rearranging (and assuming $W \neq 0$), we have

$$W'' = \frac{W'^2}{2W} + \frac{W}{2C_{12}} - \frac{M^2}{C_{12}}\left(1 - \frac{2 - M^{-2}}{2W}\right). \tag{10.25}$$

This can be rewritten as a first order system by setting $w_1 = W, w_2 = W'$:

$$w_1' = w_2 \tag{10.26a}$$

$$w_2' = \frac{w_2^2}{2w_1} + \frac{w_1}{2C_{12}} - \frac{M^2}{C_{12}}\left(1 - \frac{2 - M^{-2}}{2w_1}\right). \tag{10.26b}$$

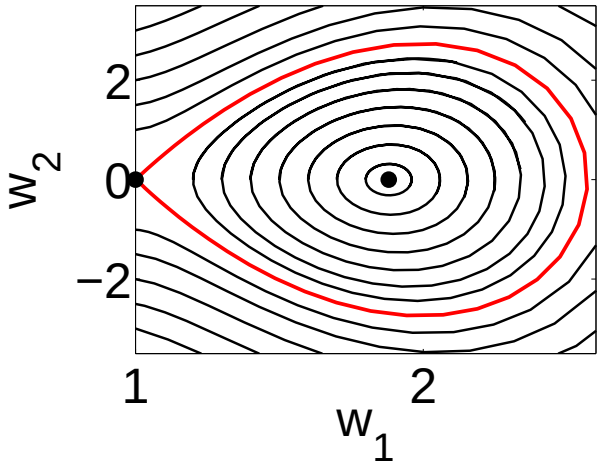


Figure 10.3: Phase plane topology for solitary wave solutions of (10.20).

Solving $w_1' = w_2' = 0$, we find that the equilibria of this system occur at $(1,0)$ and $(2M^2-1,0)$. The Jacobian of system (10.26) is

$$J = \begin{pmatrix} 0 & 1 \\ -\frac{w_2^2}{2w_1^2} + \frac{1}{2C_{12}} - \frac{2M^2-1}{2C_{12}w_1^2} & \frac{w_2}{w_1} \end{pmatrix}. \tag{10.27}$$

We find that the eigenvalues for the equilibrium at (1,0) are

$$\lambda = \pm\sqrt{\frac{1-M^2}{C_{12}}}.$$

For the layered medium given by (10.13)-(10.14), we have $C_{12} < 0$. Thus this equilibrium is a saddle if $M^2 > 1$ and it is a center if $M^2 < 1$. Meanwhile, the eigenvalues for the equilibrium at $(2M^2-1,0)$ are $\lambda = \pm\sqrt{\frac{M^2-1}{C_{12}(2M^2-1)}}$. Hence this equilibrium is a saddle for $\frac{1}{\sqrt{2}} < M^2 < 1$ and is a center otherwise.

Since the physical boundary conditions for $|x| \to \infty$ correspond to the (1,0) equilibrium, we expect the stegotons to correspond to a homoclinic connection for this node. We see that this can occur only in the case $|M| > 1$. This agrees with what has been observed in simulations: contrary to intuition based on their name, stegotons are supersonic. Figure 10.3 shows the phase plane for $M = 1.2$; the homoclinic connection is shown in red.

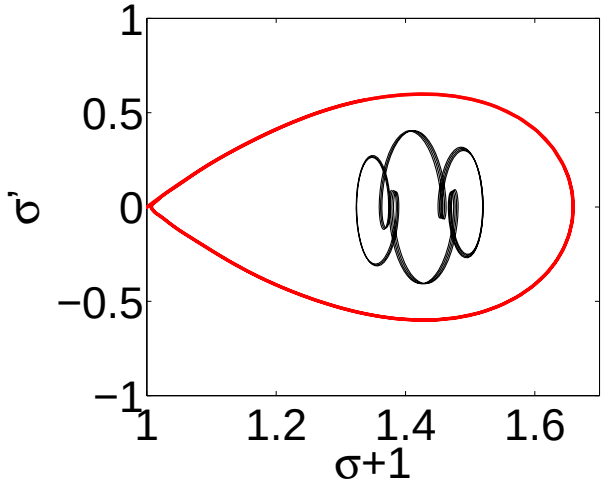


Figure 10.4: Phase plane topology for solitary wave solutions of (10.18).

The convenience of the simple system (10.20) is that it can be reduced to the single second order ODE (10.25), allowing for 2D phase plane analysis. Applying the stationary solution ansatz (10.21) to the homogenized equations (10.18), we obtain a pair of 3rd-order ODEs that may be integrated numerically. By integrating an appropriate trajectory and projecting onto the σ, σ'-plane, we obtain the homoclinic connection shown in Figure 10.4, with the same qualitative behavior. Figure 10.4 also shows an example of a nearly-periodic solution.

10.2.2 Riemann Invariants

LeVeque & Yong [83] observed that the stegotons appear to be related to the Riemann invariants of the lowest-order homogenized system

$$u_t - \frac{1}{\hat{\rho}}\sigma_x = 0 \tag{10.28a}$$

$$\sigma_t - \hat{K}\exp(\hat{K}\epsilon)u_x = 0. \tag{10.28b}$$

This system cannot give rise to solitary waves, since it includes no dispersive effects. However, the Riemann invariants for this system are

$$w^1 = \rho u - \frac{2}{\hat{c}}\sqrt{\sigma+1} \tag{10.29a}$$

$$w^2 = \rho u + \frac{2}{\hat{c}}\sqrt{\sigma+1}. \tag{10.29b}$$

It is observed in [83] that w^1 is essentially constant for right-going stegotons, while w^2 is essentially constant for left-going stegotons. In other words, these stegotons are essentially simple wave solutions to the homogenized first-order system:

$$\epsilon_t - u_x = 0 \tag{10.30a}$$

$$\hat{\rho} u_t - \exp(\hat{K}\epsilon)_x = 0 \tag{10.30b}$$

For stegotons, u and σ vanish as $|x| \to \infty$, so if w^2 or w^1 is constant (for a right-going stegoton), then

$$u = \pm\frac{2}{\hat{Z}}(1 - \sqrt{\sigma+1}), \tag{10.31}$$

where the plus (minus) sign corresponds to right- (left-) going stegotons. If we substitute the relation (10.31) into the homogenized equations (10.28), we get (to third order)

$$\frac{1}{\hat{c}}u_t = \sqrt{\sigma+1}u_x - \delta C_{21}\left(\sqrt{\sigma+1}u_{xx} + \frac{\hat{Z}}{2}u_x^2\right) - \delta^2 C_{22}\left(\sqrt{\sigma+1}u_{xxx} + \frac{3}{2}\hat{Z}u_x u_{xx}\right), \tag{10.32a}$$

$$\frac{1}{\hat{c}}u_t = \sqrt{\sigma+1}u_x - \delta C_{11}\sqrt{\sigma+1}u_{xx} - \delta^2\left(C_{12}\sqrt{\sigma+1}u_{xxx} + C_{13}\hat{Z}u_x u_{xx}\right). \tag{10.32b}$$

Observe that (10.32a) and (10.32b) are equivalent if

$$C_{11} = C_{21} = 0 \tag{10.33a}$$

$$C_{12} = C_{22} \tag{10.33b}$$

$$C_{22} = \frac{2}{3}C_{13}. \tag{10.33c}$$

The first condition is fulfilled for any piecewise constant medium, as well as for the sinusoidal media considered below. For a piecewise-constant, two-material medium, the condition $C_{12} = C_{22}$ is always satisfied if we take $\alpha = 1/2$ (that is, if the half-layers of material A and material B have the same width). Remarkably, these conditions correspond precisely to the case considered in detail by LeVeque & Yong. The final condition, $C_{22} =$

$\frac{2}{3}C_{13}$, turns out to be impossible to satisfy for a piecewise-constant two-layer periodic medium. Nevertheless, we proceed to consider (10.32a) for the stegoton medium. In this case, (10.32a) can be written

$$\frac{1}{\hat{c}}u_t = \left(1 \pm \frac{\hat{Z}}{2}u\right) u_x - \delta^2 C_{22}\left(\left(1 \pm \frac{\hat{Z}}{2}u\right) u_{xxx} + \frac{3}{2}\hat{Z}u_x u_{xx}\right). \tag{10.34}$$

Taking the plus sign and setting $v = 1 + u\hat{Z}/2$, this reduces to

$$\frac{1}{\hat{c}}v_t = vv_x - \delta^2 C_{22}\left(vv_{xxx} + \frac{3}{2}\hat{Z}v_x v_{xx}\right). \tag{10.35}$$

The case studied by LeVeque & Yong has $\hat{Z} = 2$. This turns out to be a very special value; in this case, the quantity in parentheses in (10.35) is a total derivative and (10.35) can be written as

$$v_t = \frac{\hat{c}}{2}\left(v^2\right)_x - \frac{\hat{c}}{2}\delta^2 C_{22}\left(v^2\right)_{xxx}, \tag{10.36}$$

which is the K(2,2) compacton equation, first studied by Rosenau & Hyman [95]. However, note that the boundary conditions here are $v \to 1$ as $|x| \to \infty$, which precludes compacton solutions. Instead, the equation has 'shelf soliton' solutions, as noted in [95]. To see this, we look for traveling wave solutions of the form $v(x,t) = V(x - \lambda t)$. This gives (setting $b = -\delta^2 C_{22}$)

$$-\lambda V' = \frac{\hat{c}}{2}(V^2)' + b\frac{\hat{c}}{2}(V^2)'''. \tag{10.37}$$

Integrate once to find

$$-\lambda V = \frac{\hat{c}}{2}V^2 + b\frac{\hat{c}}{2}(V^2)'' + P_1. \tag{10.38}$$

The conditions $V(\pm\infty) = 1, V'(\pm\infty) = 0$ yields $P_1 = -\lambda - \frac{\hat{c}}{2}$, giving

$$-\lambda V = \frac{\hat{c}}{2}V^2 + b\frac{\hat{c}}{2}(V^2)'' - \lambda - \frac{\hat{c}}{2}. \tag{10.39}$$

Now we multiply by VV':

$$-\lambda V^2 V' = \frac{\hat{c}}{2}V^3 V' + b\frac{\hat{c}}{2}(V^2)'' VV' - \left(\lambda - \frac{\hat{c}}{2}\right) VV', \tag{10.40}$$

and integrate again to obtain

$$-\frac{\lambda}{3}V^3 = \frac{\hat{c}}{8}V^4 + b\frac{\hat{c}}{2}(VV')^2 - \left(\lambda - \frac{\hat{c}}{2}\right)\frac{1}{2}V^2 + P_0. \tag{10.41}$$

Applying the boundary conditions at infinity yields $P_0 = -\frac{3}{8}\hat{c} + \frac{1}{6}\lambda$. Dividing by V^2, we have

$$-\frac{\lambda}{3}V = \frac{\hat{c}}{8}V^2 + b\frac{\hat{c}}{2}V'^2 - \frac{1}{2}\left(\lambda - \frac{\hat{c}}{2}\right) + V^{-2}\left(-\frac{3}{8}\hat{c} + \frac{1}{6}\lambda\right) \tag{10.42}$$

which simplifies to

$$V'^2 = \frac{1}{b}\left(\frac{\lambda}{\hat{c}} - \frac{1}{2}\right) - \frac{2}{3}\frac{\lambda}{b\hat{c}}V - \frac{1}{4b}V^2 - \frac{1}{b}\frac{1}{V^2}\left(\frac{\lambda}{3\hat{c}} - \frac{3}{4}\right). \tag{10.43}$$

The solutions of (10.43) are elliptic functions.

10.3 Time Reversal

Because system (10.1) is invariant under the transformation

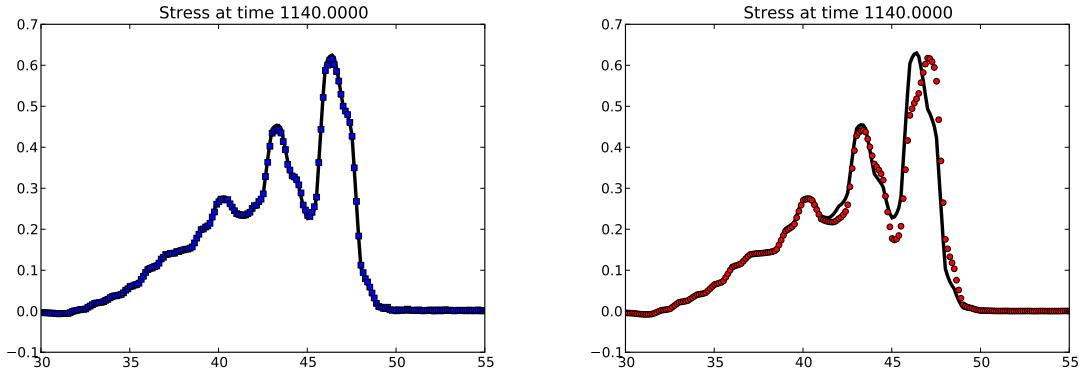
$$u \to -u, \tag{10.44a}$$

$$x \to -x, \tag{10.44b}$$

its solutions are time-reversible up to when shocks form. The dispersion induced by material inhomogeneities can be used to delay the onset of shocks and allow time-reversible nonlinear wave propagation over longer distances than otherwise possible. In the case of the stegoton medium, since shocks apparently do not form at all, the solution is time-reversible over any time interval.

This provides a useful numerical test. Namely, one may solve the stegoton problem numerically up to time T, then negate the velocity and continue solving to time $2T$. The solution at any time $2T - t_0$, with $t_0 \leq T$, should be exactly equal to the solution at t_0. As a numerical test, we take $T = 600$ and $t_0 = 60$. In Figure 10.5(a) we plot the solution obtained using WENO5 on a grid with 24 cells per layer. The $t = 1140$ solution (blue squares) is in excellent agreement with the $t = 60$ solution (black line). In fact, the maximum pointwise difference has magnitude less than 2×10^{-2}. Using a grid twice as fine, with 48 cells per layer, reduces the pointwise difference to 1×10^{-3}. The Clawpack solution, computed on the same grid (24 cells per layer), is shown in Figure 10.5(b). This lower order accurate solution shows significant numerical errors.

It is interesting to consider the time-reversibility of the stegotons in terms of the homogenized equations. In [84], it was noted that when deriving homogenized equations, for the special case of a piecewise constant medium, all of the terms with even numbers of spatial derivatives vanish. Thus, the resulting equations are invariant under the transformation In other words, the homogenized equations are exactly time-reversible. However,



(a) WENO5 (b) Clawpack

Figure 10.5: Comparison of forward solution (black line) and time-reversed solution (symbols).

it is not true that waves that obey the 1D elasticity equations (10.1) are time-reversible in any piecewise constant medium. If we modify the stegoton medium by decreasing the impedance contrast between layers (by taking a homogeneous medium, for instance), shocks form and the solution is not time-reversible.

10.4 Smoothly Varying Media

We now consider the case of a smoothly varying medium, with

$$\rho(x) = a + b \sin\left(2\pi \frac{x}{\delta}\right) \tag{10.45a}$$

$$K(x) = a + b \sin\left(2\pi\left(x + \theta\right)\right). \tag{10.45b}$$

To achieve true high order accuracy in our numerical simulations, we would need to use quadrature to evaluate certain terms, as discussed in Chapter 9. For simplicity, we instead approximate the medium by a piecewise constant medium that is uniform in each computational cell. This easily provides sufficient accuracy to determine the correct qualitative behavior in the following examples.

We first consider the case $\theta = 0, \delta = 0$, so that $K(x) = \rho(x)$ and thus the linearized sound speed is constant, just as in the medium of LeVeque & Yong. As for that medium, it can be shown that the coefficients of the dissipative terms in the homogenized equations vanish in this case. Taking $a = 5/2$, $b = 3/2$ yields a smoothly varying medium that

approximates the original stegoton medium, as ρ and K vary between a maximum of 4 and minimum of 1. We observe solitary waves similar to the piecewise constant case, as shown in Figure 10.6. Since the medium is smoothly varying, the strain is a continuous function (in contrast to the stegotons). The peaks occur at the local minima of $\rho(x)$ and $K(x)$. However, when plotted as a function of time for a given point in space, the solitary waves are smooth functions.

Next we take the same parameters except for $\theta = 1/2$, so that the fluctuations in the density and the bulk modulus are precisely out of phase (and thus the linearized sound speed varies dramatically). Again, it can be shown that the coefficients of the dissipative terms in the homogenized equations vanish in this case. However, the observed behavior (shown in Figure 10.7) is quite different from the previous example. Now the solution evolves in a manner similar to Burgers equation, with the nonlinearity dominating. The dispersion, rather than leading to solitary waves, appears only to generate noise-like oscillations.

Now we consider the same parameter values except we take $\theta = 1/4$. In this case, the coefficients of the dissipative homogenized terms are nonzero. Localized structures are observed, but there is a lot of incoherence between and behind them.

Finally, consider the case $\theta = 0$ but with the period of the density variation doubled (while the bulk modulus variation remains the same). In this case, dispersion seems to dominate and little coherent structure remains after even a short time.

10.5 $1\frac{1}{2}$*D Stegotons*

In this section we briefly investigate a simple 2D generalization of system (10.1):

$$\epsilon_t - u_x - v_y = 0 \tag{10.46a}$$

$$(\rho(x)u)_t - \sigma(\epsilon, x)_x = 0 \tag{10.46b}$$

$$(\rho(x)v)_t - \sigma(\epsilon, x)_y = 0. \tag{10.46c}$$

Here v denotes the velocity in the y-coordinate direction. This system is intended to be the simplest possible generalization, rather than to model a particular physical system. In particular, observe that (10.46) reduces to (10.1) if all partial derivatives with respect to y vanish. Consider a 2D analog of the stegoton medium obtained by simply extending uniformly in y. Clearly the 1D stegotons (again extended uniformly in y) are solutions of this system. The question we investigate is whether they are still globally attracting, stable solutions, or whether they are subject to transverse instabilities like those that arise in other important systems.

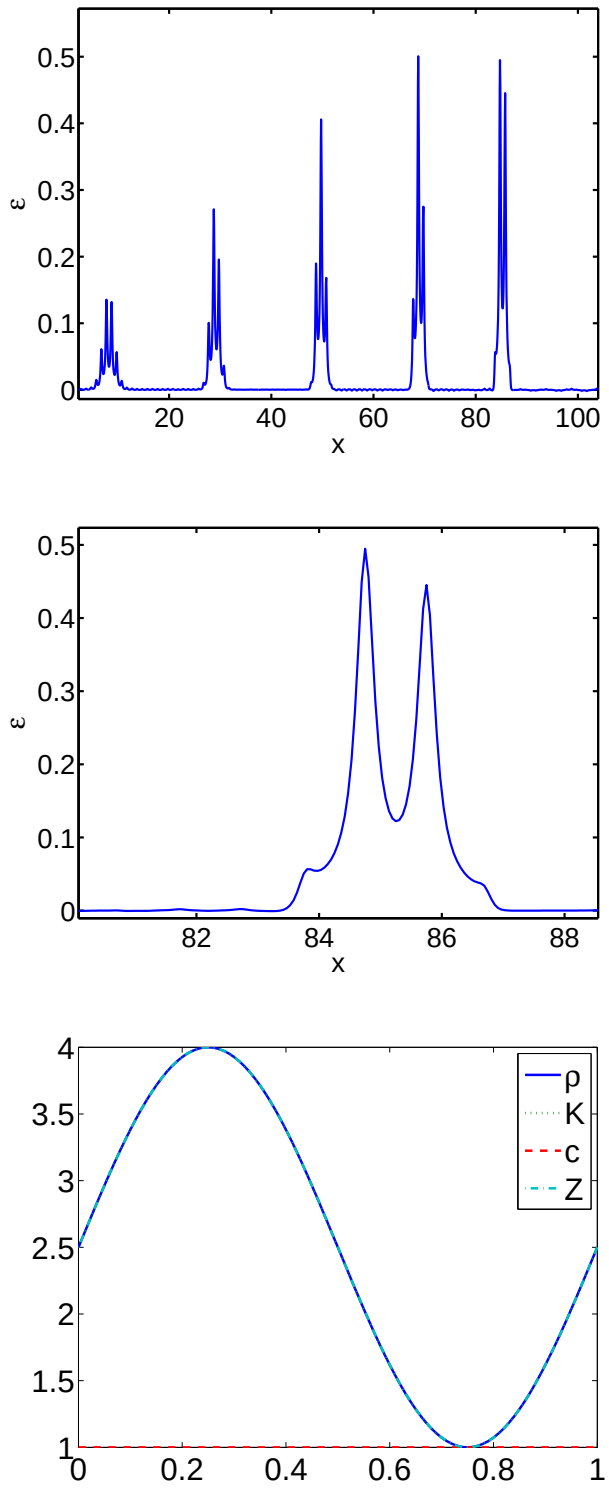


Figure 10.6: Results for the medium (10.45) with $\theta = 0$.

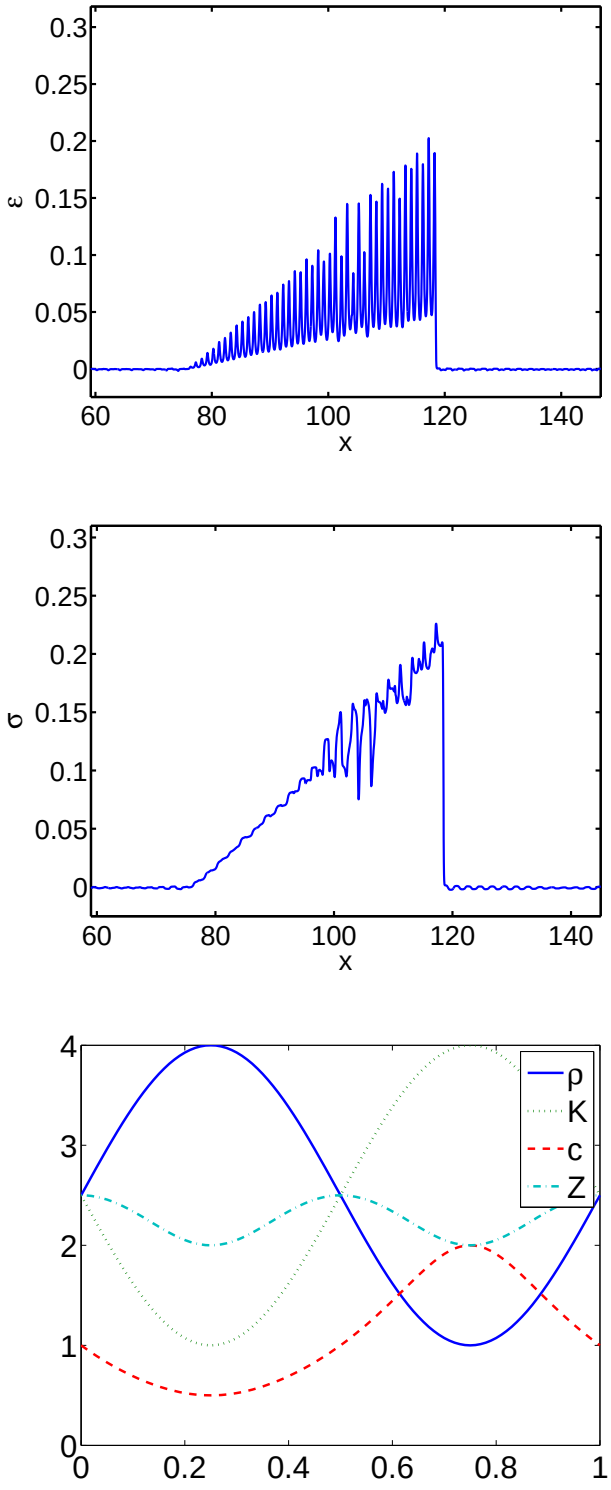


Figure 10.7: Strain (left) and stress (right) for the medium (10.45) with $\theta = 1/2$.

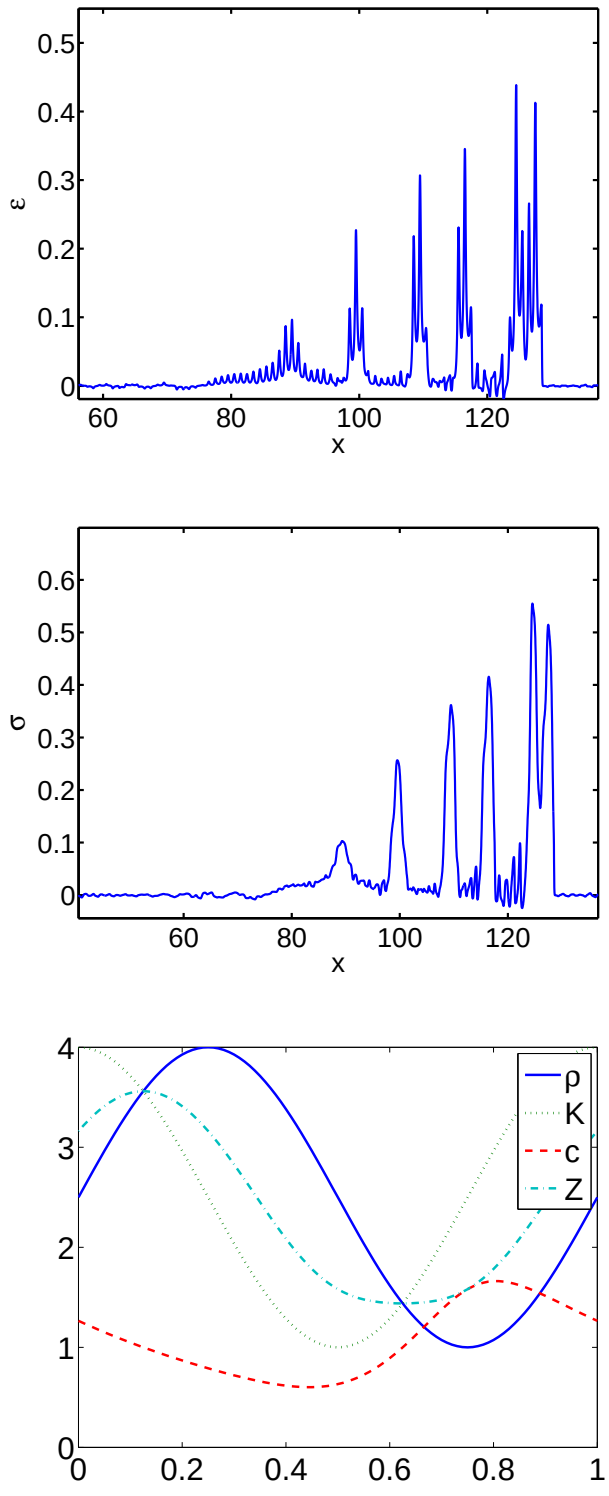


Figure 10.8: Strain (left) and stress (right) for the medium (10.45) with $\theta = 1/4$.

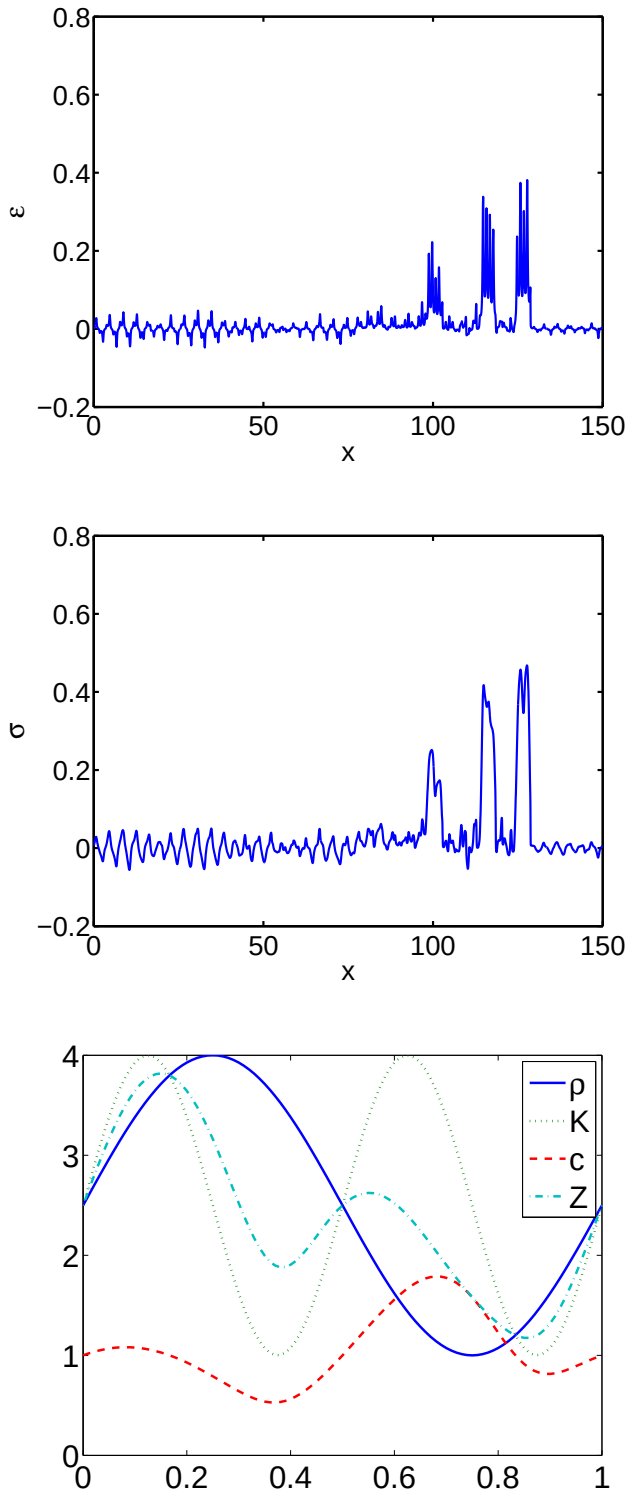


Figure 10.9: Strain (left) and stress (right) for the medium (10.45) with $\theta = 0$ and the period of the density variation equal to twice the period of the bulk modulus variation.

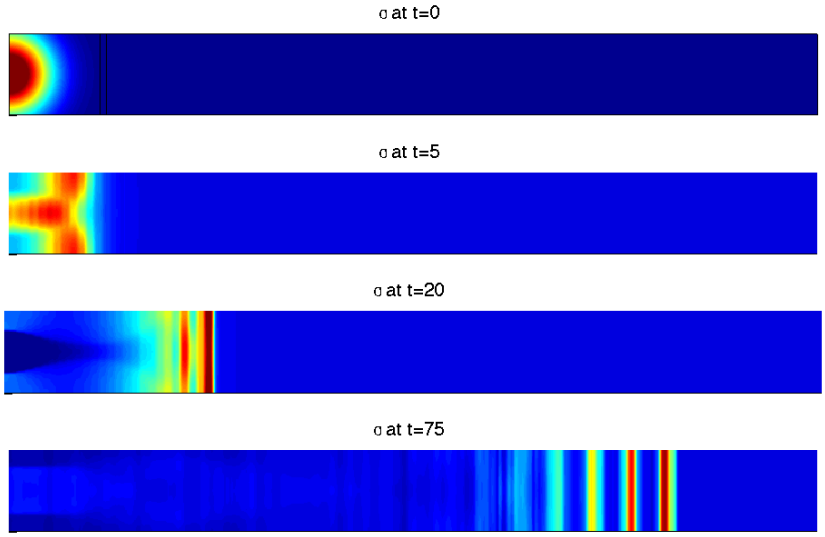


Figure 10.10: Time evolution of $1\frac{1}{2}$D Stegotons

We consider a long thin domain, with periodic boundary conditions in the y-direction and reflecting boundaries in the x-direction. The initial condition consists of a circular gaussian stress distribution centered at $(0, 0)$ and zero velocity. Figure 10.10 shows the time evolution. We observe that stegotons arise, suggesting that they are stable, globally attracting solutions. Figure 10.11 shows two 1D slices of the solution, corresponding to y-values at the middle and edge of the domain.

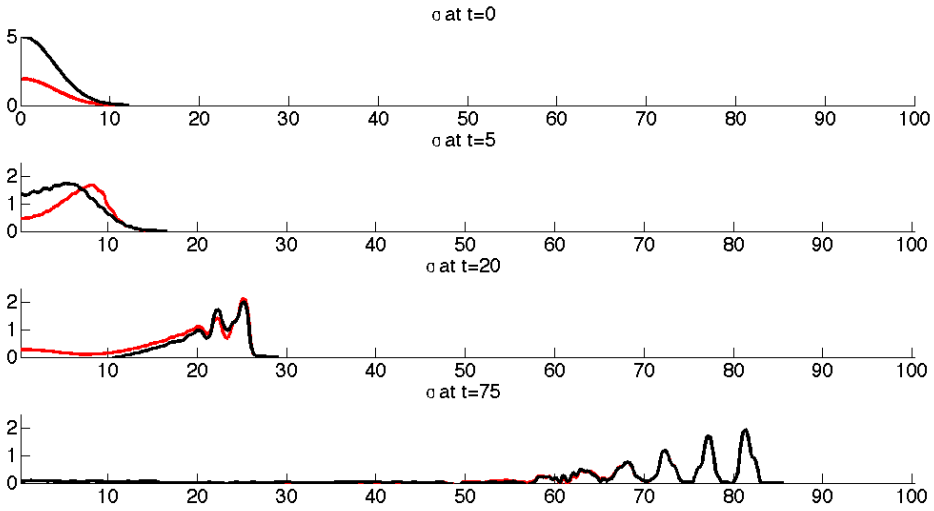


Figure 10.11: Time evolution of $1\frac{1}{2}$D Stegotons: two slices in the x-direction

Chapter 11

Conclusions and Future Directions

In this chapter we review the main contributions of this thesis and outline avenues for ongoing and future work.

11.1 SSP Theory and Methods

This thesis extends the theory of strong stability preservation in several important ways. The principal contribution is the determination of optimal threshold factors R and optimal SSP coefficients $\mathcal{C}$, as well as corresponding optimal methods, for many important classes of methods. The determination of optimal explicit Runge-Kutta and linear multistep methods has been the main focus of SSP and contractivity research over the past decade, and the open questions in this area were answered rather completely, for methods of high order with many stages and/or steps, in Chapters 4, 5, and 6. Furthermore, optimal methods and coefficients were found for classes of methods that had not been investigated previously, including:

- Threshold factors for general linear methods (Section 4.5)
- Threshold factors for multistage methods with downwinding (Section 4.6)
- SSP coefficients for implicit linear multistep methods, including methods with downwinding (Chapter 5)
- SSP coefficients for implicit Runge-Kutta methods (Section 6.3)

The most important results from all of this are:

- The new explicit Runge-Kutta methods that are simultaneously optimal in terms of both time-stepping efficiency and storage. The existence of such methods is both surprising and fortunate.

- The conjecture that $\mathcal{C} \leq 2s$ for implicit Runge-Kutta methods. Ongoing work suggests that this conjecture generalizes to general linear methods, which would be even more remarkable.

While many of the important open questions on SSP methods have been answered in this thesis, many others have been created or remain open. The following are areas in which we are currently conducting ongoing research:

- Despite the many improvements in SSP methods that are presented in this thesis, no very efficient methods of order greater than four have been given. Some such methods do exist (see [97, 98]), but they have large storage requirements or use downwinding. We are currently investigating SSP multistep Runge-Kutta methods of fifth and higher order, and this class seems to hold promise for efficient low-storage methods.

- For RKDG methods, the timestep restriction imposed by linear stability analysis is smaller than the SSP timestep restriction. We are investigating methods that are optimized in terms of the minimum of these two timestep restrictions.

- Many important systems have a dominant hyperbolic part along with a parabolic part. In such cases, traditional SSP methods are inefficient because of the stiffness of the parabolic operator. We are developing SSP methods of Chebyshev-Runge-Kutta type for application to such systems.

Many other important questions await future efforts. Among them we mention the following:

- Although Conjecture 6.5.3 is in a sense a negative result, its proof would provide a very satisfying theoretical support for the results on implicit SSP Runge-Kutta methods in Chapter 6.

- As has been noted in the literature, the SSP timestep restriction is not generally sharp for a given problem and method[74, 43]. It would be helpful to quantify the

importance of the SSP timestep restriction and the advantage conferred by using SSP methods for classes of PDEs and semi-discretizations of interest.

- The optimization formulation using absolute monotonicity can also be applied to search for optimal SSP Runge-Kutta methods with downwinding, as suggested in [32]. This would probably lead to improved downwind methods and allow development of downwind methods for higher order and more stages than has so far been undertaken.

- An important theoretical question regarding downwind methods, was posed in [51]: given a Runge-Kutta method, how may one best split the matrix $\mathbf{K}$ in order to maximize $\tilde{C}$? We have found some promising heuristic approaches, but a rigorously justified general answer to this question is still lacking.

11.2 Low-Storage Time Integrators

The new class of low-storage Runge-Kutta methods introduced in Chapter 7 is more economical than any existing methods. It is expected that these methods will become increasingly important in the future since computational power is expected to grow much more quickly than available (fast) memory.

The following are interesting avenues of research regarding low-storage methods:

- A thorough search for 2S, 2S embedded, etc. methods that are optimized for various stability and accuracy properties, or designed to be paired with specific semi-discretizations (similar to [66]). This will require the application of more sophisticated numerical optimization methods than were employed in Chapter 7.

- We are currently investigating a rigorous theoretical framework for low-storage methods, in order to be able to guarantee optimality of methods under given constraints on the number of memory registers and function evaluations.

- The various classes of low-storage Runge-Kutta methods could be extended to consider low-storage general linear methods. These would generally require more than two registers, since they would have multiple input vectors.

- Many more low-storage algorithms are possible with the addition of a third register (and even more with a fourth, etc.). In this case it is much more difficult to find all possible low-storage algorithms by hand. We hope eventually to use the rigorous

theoretical framework mentioned above to systematically analyze these, and then develop optimal methods.

11.3 High Order Numerical Wave Propagation

The high order wave propagation method of Chapter 8 is one of the first high order (higher than 3rd order) numerical methods applicable to general hyperbolic systems. It shares many of the advantages of Clawpack over traditional flux-differencing systems; for instance, it can easily be applied to problems not written in conservation form (provided that a meaningful Riemann solution can be obtained). Also, it can easily handle problems in which convective and source terms are nearly balanced, by use of the f-wave formulation.

The wave propagation methods discussed in this thesis are implemented in a software package named SharpClaw, which is available from the Clawpack website (`www.clawpack.org`). We would like to extend the numerical wave propagation method and its implementation in several ways:

- Generalize the implementation of WENO reconstruction to handle mapped logically quadrilateral grids
- Incorporate the adaptive mesh refinement algorithms available in AMRClaw
- Provide parallel implementations for both shared and distributed memory machines

11.4 Stegotons

The analysis and experiments in Chapter 10 further our understanding of these interesting waves in several ways. The analysis of the homogenized equations indicates that the stegotons correspond to a homoclinic connection in phase space and suggests that they may be related to other non-variable-coefficient systems that exhibit solitary waves. The numerical experiments indicate that the solitary waves arise for a range of materials, provided that the impediance variation generates a sufficient degree of effective dispersion.

The analysis and experiments raise many new and interesting questions. Among them are the following:

- Is it possible to predict, in general, when nonlinear first-order hyperbolic systems with spatially varying coefficients will give rise to solitary waves?

- Are these waves true solitons – i.e., are the equations integrable for some particular choice of nonlinearity and coefficients?

- Can we better explain the time-reversibility of the stegotons and predict for what parameters this will persist?

BIBLIOGRAPHY

[1] Luca Baiotti, Ian Hawke, Pedro J. Montero, Frank Loffler, Luciano Rezzolla, Nikolaos Stergioulas, José A. Font, and Ed Seidel. Three-dimensional relativistic simulations of rotating neutron-star collapse to a Kerr black hole. *Physical Review D*, 71, 2005.

[2] Jorge Balbás and Eitan Tadmor. A central differencing simulation of the Orszag-Tang Vortex system. *IEEE Transactions on Plasma Science*, 33(2):470–471, April 2005.

[3] Derek S. Bale, Randall J. LeVeque, Sorin Mitran, and James A. Rossmanith. A wave propagation method for conservation laws and balance laws with spatially varying flux functions. *SIAM Journal of Scientific Computing*, 24(3):955–978, 2002.

[4] Edmondo Bassano. Numerical simulation of thermo-solutal-capillary migration of a dissolving drop in a cavity. *IJNMF*, 41:765–788, 2003.

[5] A. Bellen, Z. Jackiewicz, and M. Zennaro. Contractivity of waveform relaxation Runge-Kutta iterations and related limit methods for dissipative systems in the maximum norm. *SIAM Journal of Numerical Analysis*, 31:499–523, 1994.

[6] A. Bellen and L. Torelli. Unconditional contractivity in the maximum norm of diagonally split Runge-Kutta methods. *SIAM Journal of Numerical Analysis*, 34:528–543, 1997.

[7] J. Berland, C. Bogey, and C. Bailly. Low-dissipation and low-dispersion fourth-order Runge-Kutta algorithm. *Computers & Fluids*, 35:1459–1463, 2006.

[8] E.K. Blum. A modification of the Runge-Kutta fourth-order method. *Mathematics of Computation*, 16:176–187, 1962.

[9] C. Bogey and C. Bailly. A family of low dispersive and low dissipative explicit schemes for flow and noise computations. *Journal of Computational Physics*, 194:194–214, 2004.

[10] C. Bolley and M. Crouzeix. Conservation de la positivité lors de la discrétisation des problémes d'évolution paraboliques. *R.A.I.R.O. Analyse Numérique*, 12:237–245, 1978.

[11] J. C. Butcher. On the implementation of implicit Runge–Kutta methods. *BIT*, 17:375–378, 1976.

[12] J.C. Butcher. *Numerical Methods for Ordinary Differential Equations*. Wiley, 2003.

[13] Rachel Caiden, Ronald P. Fedkiw, and Chris Anderson. A numerical method for two-phase flow consisting of separate compressible and incompressible regions. *Journal of Computational Physics*, 166:1–27, 2001.

[14] D.A. Calhoun, C. Helzel, and R.J. LeVeque. Logically rectangular grids and finite volume methods for pdes in circular and spherical domains. *SIAM Review*, 50(4):723–752, 2008.

[15] M. Calvo, J.M. Franco, and L. Rández. Minimum storage Runge-Kutta schemes for computational acoustics. *Computers and Mathematics with Applications*, 45:535–545, 2003.

[16] M. Calvo, J.M. Franco, and L. Rández. A new minimum storage Runge-Kutta scheme for computational acoustics. *Journal of Computational Physics*, 201:1–12, 2004.

[17] Mark H. Carpenter and Christopher A. Kennedy. Fourth-order 2N-storage Runge-Kutta schemes. Technical Report TM 109112, NASA Langley Research Center, June 1994.

[18] Mark H. Carpenter and Christopher A. Kennedy. Third-order 2N-storage Runge-Kutta schemes with error control. Technical report, NASA, 1994.

[19] José Carrillo, Irene M. Gamba, Armando Majorana, and Chi-Wang Shu. A weno-solver for the transients of boltzmann-poisson system for semiconductor devices: performance and comparisons with monte carlo methods. *Journal of Computational Physics*, 184:498–525, 2003.

[20] M.-H. Chen, B. Cockburn, and F. Reitich. High-order RKDG methods for computational electromagnetics. *Journal of Scientific Computing*, 22-23:205–226, 2005.

[21] Li-Tien Cheng, Hailiang Liu, and Stanley Osher. Computational high-frequency wave propagation using the level set method, with applications to the semi-classical limit of Schrodinger equations. *Comm. Math. Sci.*, 1(3):593–621, 2003.

[22] Vani Cheruvu, Ramachandran D. Nair, and Henry M. Turfo. A spectral finite volume transport scheme on the cubed-sphere. *Applied Numerical Mathematics*, 57:1021–1032, 2007.

[23] Bernardo Cockburn, Fengyan Li, and Chi-Wang Shu. Locally divergence-free discontinuous Galerkin methods for the Maxwell equations. *Journal of Computational Physics*, 194:588–610, 2004.

[24] Bernardo Cockburn, Jianliang Qian, Fernando Reitich, and Jing Wang. An accurate spectral/discontinuous finite-element formulation of a phase-space-based level set approach to geometrical optics. *Journal of Computational Physics*, 208:175–195, 2005.

[25] E.M. Constantinescu. Optimal explicit strong-stability-preserving general linear methods. submitted, 2009.

[26] G. Dahlquist and R. Jeltsch. Generalized disks of contractivity for explicit and implicit Runge-Kutta methods. Technical report, Department of Numerical Analysis and Computational Science, Royal Institute of Technology, Stockholm, 1979.

[27] K. Dekker and J. G. Verwer. *Stability of Runge-Kutta methods for stiff nonlinear differential equations*, volume 2 of *CWI Monographs*. North-Holland Publishing Co., Amsterdam, 1984.

[28] L. Del Zanna and N. Bucciantini. An efficient shock-capturing central-type scheme for multidimensional relativistic flows: I. hydrodynamics. *Astronomy and Astrophysics*, 390:1177–1186, 2002.

[29] Douglas Enright, Ronald Fedkiw, Joel Ferziger, and Ian Mitchell. A hybrid particle level set method for improved interface capturing. *Journal of Computational Physics*, 183:83–116, 2002.

[30] Long-Long Feng, Chi-Wang Shu, and Mengping Zhang. A hybrid cosmological hydrodynamic/n-body code based on a weighted essentially nonoscillatory scheme. *The Astrophysical Journal*, 612:1–13, 2004.

[31] L. Ferracina and M. N. Spijker. Stepsize restrictions for the total-variation-diminishing property in general Runge-Kutta methods. *SIAM Journal of Numerical Analysis*, 42:1073–1093, 2004.

[32] L. Ferracina and M. N. Spijker. Computing optimal monotonicity-preserving Runge-Kutta methods. Technical Report MI2005-07, Mathematical Institute, Leiden University, 2005.

[33] L. Ferracina and M. N. Spijker. An extension and analysis of the Shu-Osher representation of Runge-Kutta methods. *Mathematics of Computation*, 249:201–219, 2005.

[34] L Ferracina and MN Spijker. Stepsize restrictions for total-variation-boundedness in general runge-kutta procedures. *Applied Numerical Mathematics*, 53:265–279, 2005.

[35] Luca Ferracina and Marc Spjker. Strong stability of singly-diagonally-implicit Runge-Kutta methods. *Applied Numerical Mathematics*, 2008. doi:10.1016/j.apnum.2007.10.004.

[36] Tiernan R. Fogarty. *High-Resolution Finite Volume Methods for Acoustics in a Rapidly-Varying Heterogeneous Medium*. PhD thesis, University of Washington, 1998.

[37] Tiernan R. Fogarty and Randall J. LeVeque. High-resolution finite-volume methods for acoustic waves in periodic and random media. *Journal of the Acoustical Society of America*, 106:17–28, 1999.

[38] David J. Fyfe. Economical evaluation of Runge-Kutta formulae. *Mathematics of Computation*, 20(95):392–398, 1966.

[39] S. Gill. A process for the step-by-step integration of differential equations in an automatic digital computing machine. *Proceedings of the Cambridge Philosophical Society*, 47:96–108, 1950.

[40] D. Gottlieb and E. Tadmor. The CFL condition for spectral approximations to hyperbolic initial-boundary value problems. *Mathematics of Computation*, 56:565–588, 1991.

[41] Sigal Gottlieb. On high order strong stability preserving Runge-Kutta and multi step time discretizations. *Journal of Scientific Computing*, 25:105–127, 2005.

[42] Sigal Gottlieb and Lee-Ad J. Gottlieb. Strong stability preserving properties of Runge-Kutta time discretization methods for linear constant coefficient operators. *Journal of Scientific Computing*, 18:83–109, 2003.

[43] Sigal Gottlieb, David I. Ketcheson, and Chi-Wang Shu. High order strong stability preserving time discretizations. *Journal of Scientific Computing*, 38(3):251, 2009.

[44] Sigal Gottlieb and Steven J. Ruuth. Optimal strong-stability-preserving time-stepping schemes with fast downwind spatial discretizations. *Journal of Scientific Computing*, 27:289–303, 2006.

[45] Sigal Gottlieb and Chi-Wang Shu. Total variation diminishing Runge-Kutta schemes. *Mathematics of Computation*, 67:73–85, 1998.

[46] Sigal Gottlieb, Chi-Wang Shu, and Eitan Tadmor. Strong stability preserving high-order time discretization methods. *SIAM Review*, 43:89–112, 2001.

[47] E. Hairer, S.P. Norsett, and G. Wanner. *Solving ordinary differential equations I: Nonstiff Problems*. Springer Series in Computational Mathematics. Springer, Berlin, 1993.

[48] E. Hairer and G. Wanner. *Solving ordinary differential equations II: Stiff and differential-algebraic problems*, volume 14 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 1991.

[49] Ami Harten, Bjorn Engquist, Stanley Osher, and Sukumar R. Chakravarthy. Uniformly high order accurate essentially non-oscillatory schemes, iii. *Journal of Computational Physics*, 71:231–303, 1987.

[50] I Higueras. On strong stability preserving time discretization methods. *Journal of Scientific Computing*, 21:193–223, 2004.

[51] I Higueras. Representations of Runge-Kutta methods and strong stability preserving methods. *Siam Journal On Numerical Analysis*, 43:924–948, 2005.

[52] I. Higueras. Characterizing strong stability preserving additive runge-kutta methods. *Journal of Scientific Computing*, 39(1):115–128, 2009.

[53] Inmaculada Higueras. Strong stability for additive Runge-Kutta methods. *SIAM J. Numer. Anal.*, 44:1735–1758, 2006.

[54] R. Hixon, V. Allampalli, M. Nallasamy, and S.D. Sawyer. High-accuracy large-step explicit Runge-Kutta (HALE-RK) schemes for computational aeroacoustics. AIAA paper 2006-797, AIAA, 2006.

[55] Zoltan Horvath. Positivity of Runge-Kutta and diagonally split Runge-Kutta methods. *Applied Numerical Mathematics*, 28:309–326, 1998.

[56] Zoltan Horvath. On the positivity of matrix-vector products. *Linear Algebra and its Applications*, 393:253–258, 2004.

[57] Zoltan Horvath. On the positivity step size threshold of Runge-Kutta methods. *Applied Numerical Mathematics*, 53:341–356, 2005.

[58] KJI Hout. A note on unconditional maximum norm contractivity of diagonally split runge-kutta methods. *Siam Journal On Numerical Analysis*, 33:1125–1134, 1996.

[59] F.Q. Hu, M.Y. Hussaini, and J.L. Manthey. Low-dissipation and low-dispersion Runge-Kutta schemes for computational acoustics. *Journal of Computational Physics*, 124:177–191, 1996.

[60] C. Huang. Strong stability preserving hybrid methods. *Applied Numerical Mathematics*, 2008. doi: 10.1016/j.apnum.2008.03.030.

[61] W.H. Hundsdorfer and J.G. Verwer. *Numerical solution of time-dependent advection-diffusion-reaction equations*, volume 33 of *Springer Series in Computational Mathematics*. Springer, 2003.

[62] Willem Hundsdorfer and Steven J. Ruuth. On monotonicity and boundedness properties of linear multistep methods. *Mathematics of Computation*, 75(254):655–672, 2005.

[63] Willem Hundsdorfer, Steven J. Ruuth, and Raymond J. Spiteri. Monotonicity-preserving linear multistep methods. *SIAM Journal of Numerical Analysis*, 41:605–623, 2003.

[64] Rolf Jeltsch and Olavi Nevanlinna. Stability of explicit time discretizations for solving initial value problems. *Numerische Mathematik*, 37:61–91, 1981.

[65] Shi Jin, Hailiang Liu, Stanley Osher, and Yen-Hsi Richard Tsai. Computing multivalued physical observables for the semiclassical limit of the Schrodinger equation. *Journal of Computational Physics*, 205:222–241, 2005.

[66] Christopher A. Kennedy, Mark H. Carpenter, and R. Michael Lewis. Low-storage, explicit Runge-Kutta schemes for the compressible Navier-Stokes equations. *Applied Numerical Mathematics*, 35:177–219, 2000.

[67] David I. Ketcheson. Personal webpage. http://www.amath.washington.edu/ ketch/.

[68] David I. Ketcheson. An algebraic characterization of strong stability preserving Runge-Kutta schemes, 2004. BS Thesis.

[69] David I. Ketcheson. Highly efficient strong stability preserving Runge-Kutta methods with low-storage implementations. *SIAM Journal on Scientific Computing*, 30(4):2113–2136, 2008.

[70] David I. Ketcheson. Computation of optimal monotonicity preserving general linear methods. *Mathematics of Computation*, 2009. to appear.

[71] David I. Ketcheson, Sigal Gottlieb, and Colin B. Macdonald. Strong stability preserving two-step Runge-Kutta methods. 2008. in preparation.

[72] David I. Ketcheson, Colin B. Macdonald, and Sigal Gottlieb. SSP website. http://www.cfm.brown.edu/people/sg/ssp.html.

[73] David I. Ketcheson, Colin B. Macdonald, and Sigal Gottlieb. Optimal implicit strong stability preserving Runge-Kutta methods. *Applied Numerical Mathematics*, 52(2):373, 2009.

[74] David I. Ketcheson and Allen C. Robinson. On the practical importance of the SSP property for Runge-Kutta time integrators for some common Godunov-type schemes. *International Journal for Numerical Methods in Fluids*, 48:271–303, 2005.

[75] J. F. B. M. Kraaijevanger. Absolute monotonicity of polynomials occurring in the numerical solution of initial value problems. *Numerische Mathematik*, 48:303–322, 1986.

[76] J. F. B. M. Kraaijevanger. Contractivity of Runge-Kutta methods. *BIT*, 31:482–528, 1991.

[77] Simon Labrunie, José Carrillo, and Pierre Bertrand. Numerical study on hydrodynamic and quasi-neutral approximations for collisionless two-species plasmas. *Journal of Computational Physics*, 200:267–298, 2004.

[78] H W. J. Lenferink. Contractivity-preserving explicit linear multistep methods. *Numerische Mathematik*, 55:213–223, 1989.

[79] H W. J. Lenferink. Contractivity-preserving implicit linear multistep methods. *Math. Comp.*, 56:177–199, 1991.

[80] Randall J. LeVeque. *Finite Volume Methods for Hyperbolic Problems*. Cambridge University Press, 2002.

[81] Randall J. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations*. SIAM, 2007.

[82] R.J. LeVeque. Finite-volume methods for non-linear elasticity in heterogeneous media. *IJNMF*, 40:93–104, 2002.

[83] R.J. LeVeque and D.H. Yong. Phase plane behavior of solitary waves in nonlinear layered media. In T. Hou and E. Tadmor, editors, *Proeedings of the 9'th Intl. Conf. on Hyperbolic Problems: Theory, Numerics, Applications*, pages 43–51. Springer, 2002.

[84] R.J. LeVeque and D.H. Yong. Solitary waves in layered nonlinear media. *SIAM Journal of Applied Mathematics*, 63:1539–1560, 2003.

[85] X. D. Liu and P. D. Lax. Positive schemes for solving multi-dimensional hyperbolic systems of conservation laws. *Computational Fluid Dynamics Journal*, 5:133–156, 1996.

[86] Tiao Lu, Wei Cai, and Pingwen Zhang. Discontinuous Galerkin time-domain method for GPR simulation in dispersive media. *IEEE Transactions on Geoscience and Remote Sensing*, 43(1):72–80, 2005.

[87] Colin MacDonald, Sigal Gottlieb, and Steven Ruuth. A numerical study of diagonally split Runge-Kutta methods for PDEs with discontinuities. *Journal of Scientific Computing*, 2008. doi:10.1007/s10915-007-9180-6.

[88] Colin B. Macdonald. Constructing high-order Runge-Kutta methods with embedded strong-stability-preserving pairs. Master's thesis, Simon Fraser University, August 2003.

[89] A. Mignone. The dynamics of radiative shock waves: linear and nonlinear evolution. *The Astrophysical Journal*, 626:373–388, June 2005.

[90] T. Miyashita. Sonic crystals and sonic wave-guides. *Meas. Sci. Technol*, 16:R47–R63, 2005.

[91] C. Pantano, R. Deiterding, D.J. Hill, and D.I. Pullin. A low numerical dissipation patch-based adaptive mesh refinement method for large-eddy simulation of compressible flows. *Journal of Computational Physics*, 221:63–87, 2007.

[92] S. Patel and D. Drikakis. Effects of preconditioning on the accuracy and efficiency of incompressible flows. *IJNMF*, 47:963–970, 2005.

[93] Danping Peng, Barry Merriman, Stanley Osher, Hongkai Zhao, and Myungjoo Kang. A PDE-based fast local level set method. *Journal of Computational Physics*, 155:410–438, 1999.

[94] Jianxian Qiu and Chi-Wang Shu. On the construction, comparison, and local characteristic decomposition for high-order central WENO schemes. *Journal of Computational Physics*, 183:187–209, 2002.

[95] Philip Rosenau and James M. Hyman. Compactons: Solitons with finite wavelength. *Physical Review Letters*, 70(5):564–567, 1993.

[96] S. J. Ruuth and R. J. Spiteri. High-order strong-stability-preserving Runge-Kutta methods with downwind-biased spatial discretizations. *SIAM Journal of Numerical Analysis*, 42:974–996, 2004.

[97] Steven Ruuth. Global optimization of explicit strong-stability-preserving Runge-Kutta methods. *Math. Comp.*, 75:183–207, 2006.

[98] Steven J. Ruuth and Willem Hundsdorfer. High-order linear multistep methods with general monotonicity and boundedness properties. *Journal of Computational Physics*, 209:226–248, 2005.

[99] Steven J. Ruuth and Raymond J. Spiteri. Two barriers on strong-stability-preserving time discretization methods. *Journal of Scientific Computation*, 17:211–220, 2002.

[100] N. V. Sahinidis and M. Tawarmalani. *BARON 7.2: Global Optimization of Mixed-Integer Nonlinear Programs,* User's Manual, 2004. Available at `http://www.gams.com/dd/docs/solvers/baron.pdf`.

[101] L. Sanchis, F. Cervera, J. Sanchez-Dehesa, J. V. Sanchez-Perez, C. Rubio, and R. Martinez-Sala. Reflectance properties of two-dimensional sonic band-gap crystals. *J. Acoust. Soc. Am.*, 109:2598–2605, 2001.

[102] J. Sand. Circle contractive linear multistep methods. *BIT*, 26:114–122, 1986.

[103] F. Santosa and W. Symes. A dispersive effective medium for wave propagation in periodic composites. *SIAM Journal of Applied Mathematics*, 51:984–1005, 1991.

[104] L.F. Shampine. Storage reduction for runge-kutta codes. *ACM Transactions on Mathematical Software*, 5(3):245–250, 1979.

[105] C.-W. Shu and S. Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes. *Journal of Computational Physics*, 77:439–471, 1988.

[106] Chi-Wang Shu. ENO and WENO schemes for hyperbolic conservation laws. preprint.

[107] Chi-Wang Shu. Total-variation diminishing time discretizations. *SIAM J. Sci. Stat. Comp.*, 9:1073–1084, 1988.

[108] C.W. Shu. High order weighted essentially nonoscillatory schemes for convection dominated problems. *SIAM Review*, 51:82, 2009.

[109] M. N. Spijker. Contractivity in the numerical solution of initial value problems. *Numerische Mathematik*, 42:271–290, 1983.

[110] M. N. Spijker. Stepsize conditions for general monotonicity in numerical initial value problems. *Siam Journal On Numerical Analysis*, 45:1226–1245, 2007.

[111] Raymond J. Spiteri and Steven J. Ruuth. A new class of optimal high-order strong-stability-preserving time discretization methods. *SIAM Journal of Numerical Analysis*, 40:469–491, 2002.

[112] Raymond J. Spiteri and Steven J. Ruuth. Nonlinear evolution using optimal fourth-order strong-stability-preserving Runge-Kutta methods. *Mathematics and Computers in Simulation*, 62:125–135, 2003.

[113] D. Stanescu and W.G. Habashi. 2N-storage low dissipation and dispersion Runge-Kutta schemes for computational acoustics. *Journal of Computational Physics*, 143:674–681, 1998.

[114] Yuzhi Sun, Z.J. Wang, and Yen Liu. Spectral (finite) volume method for conservation laws on unstructured grids VI: Extension to viscous flow. *Journal of Computational Physics*, 215:41–58, 2006.

[115] Michel Tanguay and Tim Colonius. Progress in modeling and simulation of shock wave lithotripsy (swl). In *Fifth International Symposium on cavitation (CAV2003)*, number OS-2-1-010, 2003.

[116] E.F. Toro. Riemann solvers and numerical methods for fluid dynamics- a practical introduction(book). *Berlin: Springer-Verlag, 1997.*, 1997.

[117] K. Tselios and T.E. Simos. Optimized Runge-Kutta methods with minimal dispersion and dissipation for problems arising from computational acoustics. *Physics Letters A*, 363:38–47, 2007.

[118] J. A. van de Griend and J. F. B. M. Kraaijevanger. Absolute monotonicity of rational functions occurring in the numerical solution of initial value problems. *Numerische Mathematik*, 49:413–424, 1986.

[119] P.J. van der Houwen. Explicit Runge-Kutta formulas with increased stability boundaries. *Numerische Mathematik*, 20:149–164, 1972.

[120] J. G. Verwer. Explicit Runge-Kutta methods for parabolic partial differential equations. *Applied Numerical Mathematics*, 22:359–379, 1996.

[121] Rong Wang and Raymond J. Spiteri. Linear instability of the fifth-order weno method. *SIAM Journal on Numerical Analysis*, 45(5):1871–1901, 2007.

[122] Z.J. Wang and Yen Liu. The spectral difference method for the 2D Euler equations on unstructured grids. In *17th AIAA Computational Fluid Dynamics Conference*. AIAA, 2005.

[123] Z.J. Wang, Yen Liu, Georg May, and Antony Jameson. Spectral difference method for unstructured grids II: Extension to the Euler equations. *Journal of Scientific Computing*, 32(1):45–71, 2007.

[124] J. H. Williamson. Low-storage Runge-Kutta schemes. *Journal of Computational Physics*, 35:48–56, 1980.

[125] Weiqun Zhang and Andrew I. MacFayden. RAM: A relativistic adaptive mesh refinement hydrodynamics code. *The Astrophysical Journal Supplement Series*, 164:255–279, April 2006.

APPENDIX A

Coefficients of Runge-Kutta Methods

The coefficients of all optimal SSP methods in this thesis are available from the SSP website [72].

A.1 Optimal Implicit SSP RK Methods

A.1.1 Fourth-order Methods

Table A.1: Coefficients of the optimal 3-stage implicit SSP RK method of order 4.

$\mu_{11} = 0.157330905682085$	$\mu_{41} = 0.081822264233578$	$\lambda_{41} = 0.168078141811591$
$\mu_{21} = 0.342491639470766$	$\mu_{42} = 0.079106848361263$	$\lambda_{42} = 0.162500172803529$
$\mu_{22} = 0.047573123554705$	$\mu_{43} = 0.267698531248384$	$\lambda_{43} = 0.549902549377947$
$\mu_{32} = 0.338136048168635$	$\lambda_{21} = 0.703541497995214$	
$\mu_{33} = 0.157021682372699$	$\lambda_{32} = 0.694594303739345$	

Table A.2: Coefficients of the optimal 4-stage implicit SSP RK method of order 4.

$\mu_{11} = 0.119309657880174$	$\mu_{44} = 0.119309875536981$	$\lambda_{43} = 0.939878564212065$
$\mu_{21} = 0.226141632153728$	$\mu_{51} = 0.010888081702583$	$\lambda_{51} = 0.048147179264990$
$\mu_{22} = 0.070605579799433$	$\mu_{52} = 0.034154109552284$	$\lambda_{52} = 0.151029729585865$
$\mu_{32} = 0.180764254304414$	$\mu_{54} = 0.181099440898861$	$\lambda_{54} = 0.800823091149145$
$\mu_{33} = 0.070606483961727$	$\lambda_{21} = 1$	
$\mu_{43} = 0.212545672537219$	$\lambda_{32} = 0.799340893504885$	

Table A.3: Coefficients of the optimal 5-stage implicit SSP RK method of order 4.

$\mu_{11} = 0.072154507748981$	$\mu_{54} = 0.158089969701175$	$\lambda_{43} = 0.934991917505507$
$\mu_{21} = 0.165562779595956$	$\mu_{55} = 0.106426690493882$	$\lambda_{54} = 0.954864191619538$
$\mu_{22} = 0.071232036614272$	$\mu_{65} = 0.148091381629243$	$\lambda_{65} = 0.894472670673021$
$\mu_{32} = 0.130035287184462$	$\mu_{52} = 0.007472809894781$	$\lambda_{52} = 0.045135808380468$
$\mu_{33} = 0.063186062090477$	$\mu_{62} = 0.017471397966712$	$\lambda_{62} = 0.105527329326976$
$\mu_{43} = 0.154799860761964$	$\lambda_{21} = 1$	
$\mu_{44} = 0.077017601068238$	$\lambda_{32} = 0.785413771753555$	

Table A.4: Coefficients of the optimal 6-stage implicit SSP RK method of order 4.

$\mu_{11} = 0.077219435861458$	$\mu_{55} = 0.064105484788524$	$\lambda_{43} = 0.805203213502341$
$\mu_{21} = 0.128204308556198$	$\mu_{63} = 0.008043763906343$	$\lambda_{54} = 1$
$\mu_{22} = 0.063842903854499$	$\mu_{65} = 0.120160544649854$	$\lambda_{63} = 0.062741759593964$
$\mu_{32} = 0.128204308556197$	$\mu_{66} = 0.077016336936138$	$\lambda_{65} = 0.937258240406037$
$\mu_{33} = 0.058359965096908$	$\mu_{73} = 0.013804194371285$	$\lambda_{73} = 0.107673404480272$
$\mu_{41} = 0.008458154338733$	$\mu_{76} = 0.114400114184912$	$\lambda_{76} = 0.892326595519728$
$\mu_{43} = 0.103230521234296$	$\lambda_{21} = 1$	
$\mu_{44} = 0.058105933032597$	$\lambda_{32} = 1$	
$\mu_{54} = 0.128204308556197$	$\lambda_{41} = 0.065974025631326$	

Table A.5: Coefficients of the optimal 7-stage implicit SSP RK method of order 4.

$\mu_{11} = 0.081324471088377$	$\mu_{65} = 0.108801609187400$	$\lambda_{43} = 0.865661994183934$
$\mu_{21} = 0.108801609187400$	$\mu_{66} = 0.061352000212100$	$\lambda_{54} = 1$
$\mu_{22} = 0.051065224656204$	$\mu_{73} = 0.020631403945188$	$\lambda_{65} = 1$
$\mu_{32} = 0.108801609187400$	$\mu_{76} = 0.088170205242212$	$\lambda_{73} = 0.189624069894518$
$\mu_{33} = 0.036491713577701$	$\mu_{77} = 0.080145231879588$	$\lambda_{76} = 0.810375930105481$
$\mu_{43} = 0.094185417979586$	$\mu_{83} = 0.001561606596621$	$\lambda_{83} = 0.014352789524754$
$\mu_{44} = 0.037028821732794$	$\mu_{87} = 0.107240002590779$	$\lambda_{87} = 0.985647210475246$
$\mu_{54} = 0.108801609187400$	$\lambda_{21} = 1$	
$\mu_{55} = 0.040474271914787$	$\lambda_{32} = 1$	

Table A.6: Coefficients of the optimal 8-stage implicit SSP RK method of order 4.

$\mu_{11} = 0.080355939553359$	$\mu_{65} = 0.093742212796061$	$\lambda_{43} = 1$
$\mu_{21} = 0.093742212796061$	$\mu_{66} = 0.038334326442344$	$\lambda_{51} = 0.047220157287989$
$\mu_{22} = 0.054617345411549$	$\mu_{76} = 0.093742212796061$	$\lambda_{54} = 0.887270992114641$
$\mu_{32} = 0.093742212796061$	$\mu_{77} = 0.058861620081910$	$\lambda_{65} = 1$
$\mu_{33} = 0.039438131644116$	$\mu_{84} = 0.021977226754808$	$\lambda_{76} = 1$
$\mu_{43} = 0.093742212796061$	$\mu_{87} = 0.071764986041253$	$\lambda_{84} = 0.234443225728203$
$\mu_{44} = 0.032427875074076$	$\mu_{88} = 0.055606577879005$	$\lambda_{87} = 0.765556774271797$
$\mu_{51} = 0.004426522032754$	$\mu_{98} = 0.093742212796061$	$\lambda_{98} = 1$
$\mu_{54} = 0.083174746150582$	$\lambda_{21} = 1$	
$\mu_{55} = 0.030116385482588$	$\lambda_{32} = 1$	

Table A.7: Coefficients of the optimal 9-stage implicit SSP RK method of order 4.

$\mu_{11} = 0.068605696784244$	$\mu_{66} = 0.029699905991308$	$\lambda_{54} = 1$
$\mu_{21} = 0.082269487560004$	$\mu_{76} = 0.083046524401968$	$\lambda_{61} = 0.105338196876962$
$\mu_{22} = 0.048685583036902$	$\mu_{77} = 0.035642110881905$	$\lambda_{62} = 0.015973817828813$
$\mu_{32} = 0.077774790319743$	$\mu_{87} = 0.083046524401969$	$\lambda_{65} = 0.878687985294225$
$\mu_{33} = 0.039925150083662$	$\mu_{88} = 0.050978240433952$	$\lambda_{76} = 1$
$\mu_{43} = 0.083046524401968$	$\mu_{95} = 0.017775897980583$	$\lambda_{87} = 1$
$\mu_{44} = 0.031928917146492$	$\mu_{98} = 0.065270626421385$	$\lambda_{95} = 0.214047464461523$
$\mu_{54} = 0.083046524401968$	$\mu_{99} = 0.057552171403649$	$\lambda_{98} = 0.785952535538477$
$\mu_{55} = 0.029618614941264$	$\mu_{10,9} = 0.083046524401968$	$\lambda_{10,9} = 1$
$\mu_{61} = 0.008747971137402$	$\lambda_{21} = 0.990643355064403$	
$\mu_{62} = 0.001326570052113$	$\lambda_{32} = 0.936520713898770$	
$\mu_{65} = 0.072971983212453$	$\lambda_{43} = 1$	

Table A.8: Coefficients of the optimal 10-stage implicit SSP RK method of order 4.

$\mu_{11} = 0.053637857412307$	$\mu_{76} = 0.064406146499568$	$\lambda_{43} = 0.990280128291965$
$\mu_{21} = 0.073302847899924$	$\mu_{77} = 0.033369849008191$	$\lambda_{54} = 1$
$\mu_{22} = 0.042472343576273$	$\mu_{87} = 0.073302847899924$	$\lambda_{65} = 1$
$\mu_{32} = 0.063734820131903$	$\mu_{88} = 0.037227578299133$	$\lambda_{72} = 0.121369109867354$
$\mu_{33} = 0.039816143518898$	$\mu_{98} = 0.073302847899924$	$\lambda_{76} = 0.878630890132646$
$\mu_{43} = 0.072590353622503$	$\mu_{99} = 0.046126339053885$	$\lambda_{87} = 1$
$\mu_{44} = 0.034233821696022$	$\mu_{10,6} = 0.012892211367605$	$\lambda_{98} = 1$
$\mu_{54} = 0.073302847899924$	$\mu_{10,9} = 0.060410636532319$	$\lambda_{10,6} = 0.175875995775857$
$\mu_{55} = 0.030626774272464$	$\mu_{10,10} = 0.053275700719583$	$\lambda_{10,9} = 0.824124004224143$
$\mu_{65} = 0.073302847899924$	$\mu_{11,10} = 0.073302847899924$	$\lambda_{11,10} = 1$
$\mu_{66} = 0.029485772863308$	$\lambda_{21} = 1$	
$\mu_{72} = 0.008896701400356$	$\lambda_{32} = 0.869472632481021$	

Table A.9: Coefficients of the optimal 11-stage implicit SSP RK method of order 4.

$\mu_{11} = 0.056977945207836$	$\mu_{83} = 0.009935800759662$	$\lambda_{43} = 0.929154313811668$
$\mu_{21} = 0.065880156369595$	$\mu_{87} = 0.055944355609932$	$\lambda_{54} = 1$
$\mu_{22} = 0.043484869703481$	$\mu_{88} = 0.027887296332663$	$\lambda_{65} = 1$
$\mu_{32} = 0.065880156369595$	$\mu_{98} = 0.065880156369595$	$\lambda_{76} = 1$
$\mu_{33} = 0.035790792116714$	$\mu_{99} = 0.033340440672342$	$\lambda_{83} = 0.150816289869158$
$\mu_{41} = 0.000026595081404$	$\mu_{10,9} = 0.065880156369595$	$\lambda_{87} = 0.849183710130842$
$\mu_{43} = 0.061212831485396$	$\mu_{10,10} = 0.042024506703707$	$\lambda_{98} = 1$
$\mu_{44} = 0.029306212740362$	$\mu_{11,7} = 0.012021727578515$	$\lambda_{10,9} = 1$
$\mu_{54} = 0.065880156369595$	$\mu_{11,10} = 0.053858428791080$	$\lambda_{11,7} = 0.182478734735714$
$\mu_{55} = 0.028274789742965$	$\mu_{11,11} = 0.045164424313434$	$\lambda_{11,10} = 0.817521265264286$
$\mu_{65} = 0.065880156369595$	$\mu_{12,11} = 0.065880156369595$	$\lambda_{12,11} = 1$
$\mu_{66} = 0.025442782369057$	$\lambda_{21} = 1$	
$\mu_{76} = 0.065880156369595$	$\lambda_{32} = 1$	
$\mu_{77} = 0.029602951078198$	$\lambda_{41} = 0.000403688802047$	

A.1.2 Fifth-order Methods

Table A.10: Coefficients of the optimal 4-stage implicit SSP RK method of order 5.

$\mu_{21} = 0.125534208080981$	$\mu_{51} = 0.022869941925234$	$\lambda_{43} = 0.462033126016285$
$\mu_{22} = 0.125534208080983$	$\mu_{52} = 0.138100556728488$	$\lambda_{51} = 0.026143376902960$
$\mu_{32} = 0.350653119567098$	$\mu_{53} = 0.157510964003014$	$\lambda_{52} = 0.157867252871240$
$\mu_{33} = 0.048181647388277$	$\mu_{54} = 0.277310825799681$	$\lambda_{53} = 0.180055922824003$
$\mu_{41} = 0.097766579224131$	$\lambda_{21} = 0.143502249669229$	$\lambda_{54} = 0.317003054133379$
$\mu_{42} = 0.000000005345013$	$\lambda_{32} = 0.400843023432714$	
$\mu_{43} = 0.404181556145118$	$\lambda_{41} = 0.111760167014216$	
$\mu_{44} = 0.133639210602434$	$\lambda_{42} = 0.000000006110058$	

Table A.11: Coefficients of the optimal 5-stage implicit SSP RK method of order 5.

$\mu_{21} = 0.107733237609082$	$\mu_{53} = 0.024232322953809$	$\lambda_{43} = 0.786247596634378$
$\mu_{22} = 0.107733237609079$	$\mu_{54} = 0.220980752503271$	$\lambda_{51} = 0.128913001605754$
$\mu_{31} = 0.000009733684024$	$\mu_{55} = 0.098999612937858$	$\lambda_{52} = 0.036331447472278$
$\mu_{32} = 0.205965878618791$	$\mu_{63} = 0.079788022937926$	$\lambda_{53} = 0.077524819660326$
$\mu_{33} = 0.041505157180052$	$\mu_{64} = 0.023678103998428$	$\lambda_{54} = 0.706968664080396$
$\mu_{41} = 0.010993335656900$	$\mu_{65} = 0.194911604040485$	$\lambda_{63} = 0.255260385110718$
$\mu_{42} = 0.000000031322743$	$\lambda_{21} = 0.344663606249694$	$\lambda_{64} = 0.075751744720289$
$\mu_{43} = 0.245761367350216$	$\lambda_{31} = 0.000031140312055$	$\lambda_{65} = 0.623567413728619$
$\mu_{44} = 0.079032059834967$	$\lambda_{32} = 0.658932601159987$	
$\mu_{51} = 0.040294985548405$	$\lambda_{41} = 0.035170229692428$	
$\mu_{52} = 0.011356303341111$	$\lambda_{42} = 0.000000100208717$	

Table A.12: Coefficients of the optimal 6-stage implicit SSP RK method of order 5.

$\mu_{21} = 0.084842972180459$	$\mu_{63} = 0.026804592504486$	$\lambda_{54} = 0.861728690085026$
$\mu_{22} = 0.084842972180464$	$\mu_{65} = 0.159145416202648$	$\lambda_{62} = 0.072495338903420$
$\mu_{32} = 0.149945333907731$	$\mu_{66} = 0.085074359110886$	$\lambda_{63} = 0.133329934574294$
$\mu_{33} = 0.063973483119994$	$\mu_{73} = 0.004848530454093$	$\lambda_{65} = 0.791612404723054$
$\mu_{43} = 0.175767531234932$	$\mu_{74} = 0.042600565019890$	$\lambda_{73} = 0.024117294382203$
$\mu_{44} = 0.055745328618053$	$\mu_{76} = 0.151355691945479$	$\lambda_{74} = 0.211901395105308$
$\mu_{51} = 0.024709139041008$	$\lambda_{21} = 0.422021261021445$	$\lambda_{76} = 0.752865185365536$
$\mu_{54} = 0.173241563951140$	$\lambda_{32} = 0.745849859731775$	
$\mu_{55} = 0.054767418942828$	$\lambda_{43} = 0.874293218071360$	
$\mu_{62} = 0.014574431645716$	$\lambda_{51} = 0.122906844831659$	

Table A.13: Coefficients of the optimal 7-stage implicit SSP RK method of order 5.

$\mu_{21} = 0.077756487471956$	$\mu_{66} = 0.037306165750735$	$\lambda_{54} = 0.900717090387559$
$\mu_{22} = 0.077756487471823$	$\mu_{73} = 0.020177924440034$	$\lambda_{62} = 0.071128941372444$
$\mu_{32} = 0.126469010941083$	$\mu_{76} = 0.140855998083160$	$\lambda_{63} = 0.168525096484428$
$\mu_{33} = 0.058945597921853$	$\mu_{77} = 0.077972159279168$	$\lambda_{65} = 0.760345962143127$
$\mu_{43} = 0.143639250502198$	$\mu_{84} = 0.009653207936821$	$\lambda_{73} = 0.125302322168346$
$\mu_{44} = 0.044443238891736$	$\mu_{85} = 0.025430639631870$	$\lambda_{76} = 0.874697677831654$
$\mu_{51} = 0.011999093244164$	$\mu_{86} = 0.000177781270869$	$\lambda_{84} = 0.059945182887979$
$\mu_{54} = 0.145046006148787$	$\mu_{87} = 0.124996366168017$	$\lambda_{85} = 0.157921009644458$
$\mu_{55} = 0.047108760907057$	$\lambda_{21} = 0.482857811904546$	$\lambda_{86} = 0.001103998884730$
$\mu_{62} = 0.011454172434127$	$\lambda_{32} = 0.785356333370487$	$\lambda_{87} = 0.776211398253764$
$\mu_{63} = 0.027138257330487$	$\lambda_{43} = 0.891981318293413$	
$\mu_{65} = 0.122441492758580$	$\lambda_{51} = 0.074512829695468$	

Table A.14: Coefficients of the optimal 8-stage implicit SSP RK method of order 5.

$\mu_{21} = 0.068228425119547$	$\mu_{76} = 0.116977452926909$	$\lambda_{62} = 0.056144626483417$
$\mu_{22} = 0.068228425081188$	$\mu_{77} = 0.050447703819928$	$\lambda_{63} = 0.148913610539984$
$\mu_{32} = 0.105785458668142$	$\mu_{84} = 0.011255581082016$	$\lambda_{65} = 0.794939486396848$
$\mu_{33} = 0.049168429086829$	$\mu_{85} = 0.006541409424671$	$\lambda_{73} = 0.115904148048060$
$\mu_{43} = 0.119135238085849$	$\mu_{87} = 0.114515518273119$	$\lambda_{76} = 0.884095226988328$
$\mu_{44} = 0.040919294063196$	$\mu_{88} = 0.060382824328534$	$\lambda_{84} = 0.085067722561958$
$\mu_{51} = 0.009164078944895$	$\mu_{95} = 0.002607774587593$	$\lambda_{85} = 0.049438833770315$
$\mu_{54} = 0.120257079939301$	$\mu_{96} = 0.024666705635997$	$\lambda_{87} = 0.865488353423280$
$\mu_{55} = 0.039406904101415$	$\mu_{98} = 0.104666894951906$	$\lambda_{95} = 0.019709106398420$
$\mu_{62} = 0.007428674198294$	$\lambda_{21} = 0.515658560550227$	$\lambda_{96} = 0.186426667470161$
$\mu_{63} = 0.019703233696280$	$\lambda_{32} = 0.799508082567950$	$\lambda_{98} = 0.791054172708715$
$\mu_{65} = 0.105180973170163$	$\lambda_{43} = 0.900403391614526$	
$\mu_{66} = 0.045239659320409$	$\lambda_{51} = 0.069260513476804$	
$\mu_{73} = 0.015335646668415$	$\lambda_{54} = 0.908882077064212$	

Table A.15: Coefficients of the optimal 9-stage implicit SSP RK method of order 5.

$\mu_{21} = 0.057541273792734$	$\mu_{77} = 0.042925976445877$	$\lambda_{62} = 0.100295062538531$
$\mu_{22} = 0.057541282875429$	$\mu_{84} = 0.011070977346914$	$\lambda_{65} = 0.899704937426848$
$\mu_{32} = 0.089687860942851$	$\mu_{87} = 0.101327254746568$	$\lambda_{73} = 0.032083982209117$
$\mu_{33} = 0.041684970395150$	$\mu_{88} = 0.046669302312152$	$\lambda_{74} = 0.161972606843345$
$\mu_{43} = 0.101622955619526$	$\mu_{95} = 0.010281040119047$	$\lambda_{76} = 0.805943410735452$
$\mu_{44} = 0.040743690263377$	$\mu_{98} = 0.102117191974435$	$\lambda_{84} = 0.098497788983963$
$\mu_{51} = 0.009276188714858$	$\mu_{99} = 0.050500143250113$	$\lambda_{87} = 0.901502211016037$
$\mu_{54} = 0.101958242208571$	$\mu_{10,6} = 0.000157554758807$	$\lambda_{95} = 0.091469767162319$
$\mu_{55} = 0.040815264589441$	$\mu_{10,7} = 0.023607648002010$	$\lambda_{98} = 0.908530232837680$
$\mu_{62} = 0.011272987717036$	$\mu_{10,9} = 0.088454624345414$	$\lambda_{10,6} = 0.001401754777391$
$\mu_{65} = 0.101125244372555$	$\lambda_{21} = 0.511941093031398$	$\lambda_{10,7} = 0.210035759124536$
$\mu_{66} = 0.040395338505384$	$\lambda_{32} = 0.797947256574797$	$\lambda_{10,9} = 0.786975228149903$
$\mu_{73} = 0.003606182878823$	$\lambda_{43} = 0.904133043080300$	
$\mu_{74} = 0.018205434656765$	$\lambda_{51} = 0.082529667434119$	
$\mu_{76} = 0.090586614534056$	$\lambda_{54} = 0.907116066770269$	

Table A.16: Coefficients of the optimal 10-stage implicit SSP RK method of order 5.

$\mu_{21} = 0.052445615058994$	$\mu_{87} = 0.089103469454345$	$\lambda_{62} = 0.094135396158718$
$\mu_{22} = 0.052445635165954$	$\mu_{88} = 0.040785658461768$	$\lambda_{65} = 0.905864193215084$
$\mu_{32} = 0.079936220395519$	$\mu_{95} = 0.009201462517982$	$\lambda_{73} = 0.033130514796271$
$\mu_{33} = 0.038724845476313$	$\mu_{98} = 0.089540979697808$	$\lambda_{74} = 0.154496709294644$
$\mu_{43} = 0.089893189589075$	$\mu_{99} = 0.042414168555682$	$\lambda_{76} = 0.812371189661489$
$\mu_{44} = 0.037676214671832$	$\mu_{10,6} = 0.005634796609556$	$\lambda_{84} = 0.097617319434729$
$\mu_{51} = 0.007606429497294$	$\mu_{10,7} = 0.006560464576444$	$\lambda_{87} = 0.902382678155958$
$\mu_{54} = 0.090180506502554$	$\mu_{10,9} = 0.086547180546464$	$\lambda_{95} = 0.093186499255038$
$\mu_{55} = 0.035536573874530$	$\mu_{10,10} = 0.043749770437420$	$\lambda_{98} = 0.906813500744962$
$\mu_{62} = 0.009295158915663$	$\mu_{11,7} = 0.001872759401284$	$\lambda_{10,6} = 0.057065598977612$
$\mu_{65} = 0.089447242753894$	$\mu_{11,8} = 0.017616881402665$	$\lambda_{10,7} = 0.066440169285130$
$\mu_{66} = 0.036490114423762$	$\mu_{11,10} = 0.079160150775900$	$\lambda_{10,9} = 0.876494226842443$
$\mu_{73} = 0.003271387942850$	$\lambda_{21} = 0.531135486241871$	$\lambda_{11,7} = 0.018966103726616$
$\mu_{74} = 0.015255382390056$	$\lambda_{32} = 0.809542670828687$	$\lambda_{11,8} = 0.178412453726484$
$\mu_{76} = 0.080215515252923$	$\lambda_{43} = 0.910380456183399$	$\lambda_{11,10} = 0.801683136446066$
$\mu_{77} = 0.035768398609662$	$\lambda_{51} = 0.077033029836054$	
$\mu_{84} = 0.009638972523544$	$\lambda_{54} = 0.913290217244921$	

Table A.17: Coefficients of the optimal 11-stage implicit SSP RK method of order 5.

$\mu_{21} = 0.048856948431570$	$\mu_{95} = 0.008095394925904$	$\lambda_{63} = 0.008158028526592$
$\mu_{22} = 0.048856861697775$	$\mu_{98} = 0.080142391870059$	$\lambda_{65} = 0.905811942678904$
$\mu_{32} = 0.072383163641108$	$\mu_{99} = 0.036372965664654$	$\lambda_{73} = 0.034327672500586$
$\mu_{33} = 0.035920513887793$	$\mu_{10,6} = 0.005907318148947$	$\lambda_{74} = 0.138178156365216$
$\mu_{43} = 0.080721632683704$	$\mu_{10,7} = 0.005394911565057$	$\lambda_{76} = 0.827494171134198$
$\mu_{44} = 0.034009594943671$	$\mu_{10,9} = 0.076935557118137$	$\lambda_{84} = 0.093508818968334$
$\mu_{51} = 0.006438090160799$	$\mu_{10,10} = 0.032282094274356$	$\lambda_{87} = 0.906491181031666$
$\mu_{54} = 0.081035022899306$	$\mu_{11,7} = 0.003571080721480$	$\lambda_{95} = 0.091745217287743$
$\mu_{55} = 0.032672027896742$	$\mu_{11,8} = 0.008920593887617$	$\lambda_{98} = 0.908254782302260$
$\mu_{62} = 0.007591099341932$	$\mu_{11,10} = 0.075746112223043$	$\lambda_{10,6} = 0.066947714363965$
$\mu_{63} = 0.000719846382100$	$\mu_{11,11} = 0.042478561828713$	$\lambda_{10,7} = 0.061140603801867$
$\mu_{65} = 0.079926841108108$	$\mu_{12,8} = 0.004170617993886$	$\lambda_{10,9} = 0.871911681834169$
$\mu_{66} = 0.033437798720082$	$\mu_{12,9} = 0.011637432775226$	$\lambda_{11,7} = 0.040471104837131$
$\mu_{73} = 0.003028997848550$	$\mu_{12,11} = 0.072377330912325$	$\lambda_{11,8} = 0.101097207986272$
$\mu_{74} = 0.012192534706212$	$\lambda_{21} = 0.553696439876870$	$\lambda_{11,10} = 0.858431687176596$
$\mu_{76} = 0.073016254277378$	$\lambda_{32} = 0.820319346617409$	$\lambda_{12,8} = 0.047265668639449$
$\mu_{77} = 0.033377699686911$	$\lambda_{43} = 0.914819326070196$	$\lambda_{12,9} = 0.131887178872293$
$\mu_{84} = 0.008251011235053$	$\lambda_{51} = 0.072962960562995$	$\lambda_{12,11} = 0.820253244225314$
$\mu_{87} = 0.079986775597087$	$\lambda_{54} = 0.918370981510030$	
$\mu_{88} = 0.035640440183022$	$\lambda_{62} = 0.086030028794504$	

A.1.3 Sixth-order Methods

Table A.18: Coefficients of the optimal 6-stage implicit SSP RK method of order 6.

$\mu_{21} = 0.306709397198437$	$\mu_{63} = 0.395057247524893$	$\lambda_{51} = 0.018587746937629$
$\mu_{22} = 0.306709397198281$	$\mu_{64} = 0.014536993458566$	$\lambda_{52} = 0.000004866574675$
$\mu_{31} = 0.100402778173265$	$\mu_{65} = 0.421912313467517$	$\lambda_{53} = 0.024929494718837$
$\mu_{32} = 0.000000014622272$	$\mu_{66} = 0.049194928995335$	$\lambda_{54} = 0.060412325234826$
$\mu_{33} = 0.100402700098726$	$\mu_{71} = 0.054129307323559$	$\lambda_{61} = 0.000006020335333$
$\mu_{41} = 0.000015431349319$	$\mu_{72} = 0.002083586568620$	$\lambda_{62} = 0.000000003205153$
$\mu_{42} = 0.000708584139276$	$\mu_{73} = 0.233976271277479$	$\lambda_{63} = 0.072039142196788$
$\mu_{43} = 0.383195003696784$	$\mu_{74} = 0.184897163424393$	$\lambda_{64} = 0.002650837430364$
$\mu_{44} = 0.028228318307509$	$\mu_{75} = 0.303060566272042$	$\lambda_{65} = 0.076936194272824$
$\mu_{51} = 0.101933808745384$	$\mu_{76} = 0.135975816243004$	$\lambda_{71} = 0.009870541274021$
$\mu_{52} = 0.000026687930165$	$\lambda_{21} = 0.055928810359256$	$\lambda_{72} = 0.000379944400556$
$\mu_{53} = 0.136711477475771$	$\lambda_{31} = 0.018308561756789$	$\lambda_{73} = 0.042665841426363$
$\mu_{54} = 0.331296656179688$	$\lambda_{32} = 0.000000002666388$	$\lambda_{74} = 0.033716209818106$
$\mu_{55} = 0.107322255666019$	$\lambda_{41} = 0.000002813924247$	$\lambda_{75} = 0.055263441854804$
$\mu_{61} = 0.000033015066992$	$\lambda_{42} = 0.000129211130507$	$\lambda_{76} = 0.024795346049276$
$\mu_{62} = 0.000000017576816$	$\lambda_{43} = 0.069876048429340$	

Table A.19: Coefficients of the optimal 7-stage implicit SSP RK method of order 6.

$\mu_{21} = 0.090485932570398$	$\mu_{73} = 0.069025907032937$	$\lambda_{61} = 0.000000155574348$
$\mu_{22} = 0.090485932570397$	$\mu_{74} = 0.373360315300742$	$\lambda_{62} = 0.102670355321862$
$\mu_{32} = 0.346199513509666$	$\mu_{75} = 0.007542750523234$	$\lambda_{63} = 0.000000129323288$
$\mu_{33} = 0.056955495796615$	$\mu_{76} = 0.005465714557738$	$\lambda_{64} = 0.086906235023916$
$\mu_{41} = 0.089183260058590$	$\mu_{77} = 0.063240270982556$	$\lambda_{65} = 0.001948095974350$
$\mu_{42} = 0.122181527536711$	$\mu_{81} = 0.044161355044152$	$\lambda_{71} = 0.000000005742021$
$\mu_{43} = 0.340520235772773$	$\mu_{82} = 0.204837996136028$	$\lambda_{72} = 0.085547570527144$
$\mu_{44} = 0.086699362107543$	$\mu_{83} = 0.191269829083813$	$\lambda_{73} = 0.018145676643359$
$\mu_{51} = 0.214371998459638$	$\mu_{84} = 0.255834644704751$	$\lambda_{74} = 0.098149750494075$
$\mu_{52} = 0.046209156887254$	$\mu_{85} = 0.015984178241749$	$\lambda_{75} = 0.001982854233713$
$\mu_{53} = 0.215162143673919$	$\mu_{86} = 0.016124165979879$	$\lambda_{76} = 0.001436838619770$
$\mu_{54} = 0.000000362542364$	$\mu_{87} = 0.151145768228502$	$\lambda_{81} = 0.011609230551384$
$\mu_{55} = 0.209813410800754$	$\lambda_{21} = 0.023787133610744$	$\lambda_{82} = 0.053848246287940$
$\mu_{61} = 0.000000591802702$	$\lambda_{32} = 0.091009661390427$	$\lambda_{83} = 0.050281417794762$
$\mu_{62} = 0.390556634551239$	$\lambda_{41} = 0.023444684301672$	$\lambda_{84} = 0.067254353278777$
$\mu_{63} = 0.000000491944026$	$\lambda_{42} = 0.032119338749362$	$\lambda_{85} = 0.004201954631994$
$\mu_{64} = 0.330590135449081$	$\lambda_{43} = 0.089516680829776$	$\lambda_{86} = 0.004238754905099$
$\mu_{65} = 0.007410530577593$	$\lambda_{51} = 0.056354565012571$	$\lambda_{87} = 0.039733519691061$
$\mu_{66} = 0.070407008959133$	$\lambda_{52} = 0.012147561037311$	
$\mu_{71} = 0.000000021842570$	$\lambda_{53} = 0.056562280060094$	
$\mu_{72} = 0.325421794191472$	$\lambda_{54} = 0.000000095305905$	

Table A.20: Coefficients of the optimal 8-stage implicit SSP RK method of order 6.

$\mu_{21} = 0.078064586430339$	$\mu_{83} = 0.008253954430873$	$\lambda_{65} = 0.530062554633790$
$\mu_{22} = 0.078064586430334$	$\mu_{84} = 0.230190271515289$	$\lambda_{72} = 0.000000021253185$
$\mu_{31} = 0.000000000128683$	$\mu_{85} = 0.046429529676480$	$\lambda_{73} = 0.254322947692795$
$\mu_{32} = 0.207887720440412$	$\mu_{86} = 0.017457063072040$	$\lambda_{74} = 0.004502630688369$
$\mu_{33} = 0.051491724905522$	$\mu_{87} = 0.017932893410781$	$\lambda_{75} = 0.400665465691124$
$\mu_{41} = 0.039407945831803$	$\mu_{88} = 0.322331010725841$	$\lambda_{76} = 0.223929973789109$
$\mu_{43} = 0.256652317630585$	$\mu_{91} = 0.011069087473717$	$\lambda_{81} = 0.275406645480353$
$\mu_{44} = 0.062490509654886$	$\mu_{92} = 0.010971589676607$	$\lambda_{82} = 0.001937467969363$
$\mu_{51} = 0.009678931461971$	$\mu_{93} = 0.068827453812950$	$\lambda_{83} = 0.018605123379003$
$\mu_{52} = 0.113739188386853$	$\mu_{94} = 0.048864283062331$	$\lambda_{84} = 0.518868675379274$
$\mu_{54} = 0.227795405648863$	$\mu_{95} = 0.137398274895655$	$\lambda_{85} = 0.104656154246370$
$\mu_{55} = 0.076375614721986$	$\mu_{96} = 0.090347431612516$	$\lambda_{86} = 0.039349722004217$
$\mu_{62} = 0.010220279377975$	$\mu_{97} = 0.029504401738350$	$\lambda_{87} = 0.040422284523661$
$\mu_{63} = 0.135083590682973$	$\mu_{98} = 0.000167109498102$	$\lambda_{91} = 0.024950675444873$
$\mu_{65} = 0.235156310567507$	$\lambda_{21} = 0.175964293749273$	$\lambda_{92} = 0.024730907022402$
$\mu_{66} = 0.033370798931382$	$\lambda_{31} = 0.000000000290062$	$\lambda_{93} = 0.155143002154553$
$\mu_{72} = 0.000000009428737$	$\lambda_{32} = 0.468596806556916$	$\lambda_{94} = 0.110144297841125$
$\mu_{73} = 0.112827524882246$	$\lambda_{41} = 0.088828900190110$	$\lambda_{95} = 0.309707532056893$
$\mu_{74} = 0.001997541632150$	$\lambda_{43} = 0.578516403866171$	$\lambda_{96} = 0.203650883489192$
$\mu_{75} = 0.177750742549303$	$\lambda_{51} = 0.021817144198582$	$\lambda_{97} = 0.066505459796630$
$\mu_{76} = 0.099344022703332$	$\lambda_{52} = 0.256377915663045$	$\lambda_{98} = 0.000376679185235$
$\mu_{77} = 0.025183595544641$	$\lambda_{54} = 0.513470441684846$	
$\mu_{81} = 0.122181071065616$	$\lambda_{62} = 0.023037388973687$	
$\mu_{82} = 0.000859535946343$	$\lambda_{63} = 0.304490034708070$	

Table A.21: Coefficients of the optimal 9-stage implicit SSP RK method of order 6.

$\mu_{21} = 0.060383920365295$	$\mu_{88} = 0.056749019092783$	$\lambda_{65} = 0.519973146034093$
$\mu_{22} = 0.060383920365140$	$\mu_{91} = 0.000000072610411$	$\lambda_{71} = 0.000035341304071$
$\mu_{31} = 0.000000016362287$	$\mu_{92} = 0.000000387168511$	$\lambda_{72} = 0.108248004479122$
$\mu_{32} = 0.119393671070984$	$\mu_{93} = 0.000400376164405$	$\lambda_{73} = 0.150643488255346$
$\mu_{33} = 0.047601859039825$	$\mu_{94} = 0.000109472445726$	$\lambda_{74} = 0.001299063147749$
$\mu_{42} = 0.000000124502898$	$\mu_{95} = 0.012817181286633$	$\lambda_{75} = 0.000727575773504$
$\mu_{43} = 0.144150297305350$	$\mu_{96} = 0.011531979169562$	$\lambda_{76} = 0.727853067743022$
$\mu_{44} = 0.016490678866732$	$\mu_{97} = 0.000028859233948$	$\lambda_{81} = 0.000000864398917$
$\mu_{51} = 0.014942049029658$	$\mu_{98} = 0.143963789161172$	$\lambda_{82} = 0.000000092581509$
$\mu_{52} = 0.033143125204828$	$\mu_{99} = 0.060174596046625$	$\lambda_{83} = 0.198483904509141$
$\mu_{53} = 0.020040368468312$	$\mu_{10,1} = 0.001577092080021$	$\lambda_{84} = 0.099500236576982$
$\mu_{54} = 0.095855615754989$	$\mu_{10,2} = 0.000008909587678$	$\lambda_{85} = 0.000000002211499$
$\mu_{55} = 0.053193337903908$	$\mu_{10,3} = 0.000003226074427$	$\lambda_{86} = 0.007174780797111$
$\mu_{61} = 0.000006536159050$	$\mu_{10,4} = 0.000000062166910$	$\lambda_{87} = 0.694839938634174$
$\mu_{62} = 0.000805531139166$	$\mu_{10,5} = 0.009112668630420$	$\lambda_{91} = 0.000000420876394$
$\mu_{63} = 0.015191136635430$	$\mu_{10,6} = 0.008694079174358$	$\lambda_{92} = 0.000002244169749$
$\mu_{64} = 0.054834245267704$	$\mu_{10,7} = 0.017872872156132$	$\lambda_{93} = 0.002320726117116$
$\mu_{65} = 0.089706774214904$	$\mu_{10,8} = 0.027432316305282$	$\lambda_{94} = 0.000634542179300$
$\mu_{71} = 0.000006097150226$	$\mu_{10,9} = 0.107685980331284$	$\lambda_{95} = 0.074293052394615$
$\mu_{72} = 0.018675155382709$	$\lambda_{21} = 0.350007201986739$	$\lambda_{96} = 0.066843552689032$
$\mu_{73} = 0.025989306353490$	$\lambda_{31} = 0.000000094841777$	$\lambda_{97} = 0.000167278634186$
$\mu_{74} = 0.000224116890218$	$\lambda_{32} = 0.692049215977999$	$\lambda_{98} = 0.834466572009306$
$\mu_{75} = 0.000125522781582$	$\lambda_{42} = 0.000000721664155$	$\lambda_{10,1} = 0.009141400274516$
$\mu_{76} = 0.125570620920810$	$\lambda_{43} = 0.835547641163090$	$\lambda_{10,2} = 0.000051643216195$
$\mu_{77} = 0.019840674620006$	$\lambda_{51} = 0.086609559981880$	$\lambda_{10,3} = 0.000018699502726$
$\mu_{81} = 0.000000149127775$	$\lambda_{52} = 0.192109628653810$	$\lambda_{10,4} = 0.000000360342058$
$\mu_{82} = 0.000000015972341$	$\lambda_{53} = 0.116161276908552$	$\lambda_{10,5} = 0.052820347381733$
$\mu_{83} = 0.034242827620807$	$\lambda_{54} = 0.555614071795216$	$\lambda_{10,6} = 0.050394050390558$
$\mu_{84} = 0.017165973521939$	$\lambda_{61} = 0.000037885959162$	$\lambda_{10,7} = 0.103597678603687$
$\mu_{85} = 0.000000000381532$	$\lambda_{62} = 0.004669151960107$	$\lambda_{10,8} = 0.159007699664781$
$\mu_{86} = 0.001237807078917$	$\lambda_{63} = 0.088053362494510$	$\lambda_{10,9} = 0.624187175011814$
$\mu_{87} = 0.119875131948576$	$\lambda_{64} = 0.317839263219390$	

Table A.22: Coefficients of the optimal 10-stage implicit SSP RK method of order 6.

$\mu_{21} = 0.054638144097621$	$\mu_{95} = 0.013484714992727$	$\lambda_{65} = 0.755424009901960$
$\mu_{22} = 0.054638144097609$	$\mu_{96} = 0.012301077330264$	$\lambda_{73} = 0.189047812082446$
$\mu_{32} = 0.094708145223810$	$\mu_{98} = 0.097178530400423$	$\lambda_{74} = 0.000003741673193$
$\mu_{33} = 0.044846931722606$	$\mu_{99} = 0.039273658398104$	$\lambda_{76} = 0.810948446244362$
$\mu_{43} = 0.108958403164940$	$\mu_{10,1} = 0.000987065715240$	$\lambda_{84} = 0.169503368254511$
$\mu_{44} = 0.031071352647397$	$\mu_{10,2} = 0.000000347467847$	$\lambda_{85} = 0.060663661331375$
$\mu_{51} = 0.004498251069701$	$\mu_{10,6} = 0.004337021151393$	$\lambda_{86} = 0.000000038106595$
$\mu_{52} = 0.005530448043688$	$\mu_{10,7} = 0.011460261685365$	$\lambda_{87} = 0.768392593572726$
$\mu_{54} = 0.107851443619437$	$\mu_{10,8} = 0.002121689510807$	$\lambda_{94} = 0.000000003546047$
$\mu_{55} = 0.018486380725450$	$\mu_{10,9} = 0.104338127248348$	$\lambda_{95} = 0.109198714839684$
$\mu_{62} = 0.015328210231111$	$\mu_{10,10} = 0.042268075457472$	$\lambda_{96} = 0.099613661566658$
$\mu_{63} = 0.014873940010974$	$\mu_{11,3} = 0.000656941338471$	$\lambda_{98} = 0.786948084216732$
$\mu_{64} = 0.000000013999299$	$\mu_{11,7} = 0.015039465910057$	$\lambda_{10,1} = 0.007993221037648$
$\mu_{65} = 0.093285690103096$	$\mu_{11,8} = 0.004816543620956$	$\lambda_{10,2} = 0.000002813781560$
$\mu_{66} = 0.031019852663844$	$\mu_{11,9} = 0.031302441038151$	$\lambda_{10,6} = 0.035121034164983$
$\mu_{73} = 0.023345108682580$	$\mu_{11,10} = 0.071672462436845$	$\lambda_{10,7} = 0.092804768098049$
$\mu_{74} = 0.000000462051194$	$\lambda_{21} = 0.442457635916190$	$\lambda_{10,8} = 0.017181361859997$
$\mu_{76} = 0.100142283610706$	$\lambda_{32} = 0.766942997969774$	$\lambda_{10,9} = 0.844926230212794$
$\mu_{77} = 0.037191650574052$	$\lambda_{43} = 0.882341050812911$	$\lambda_{11,3} = 0.005319886250823$
$\mu_{84} = 0.020931607249912$	$\lambda_{51} = 0.036426667979449$	$\lambda_{11,7} = 0.121789029292733$
$\mu_{85} = 0.007491225374492$	$\lambda_{52} = 0.044785360253007$	$\lambda_{11,8} = 0.039004189088262$
$\mu_{86} = 0.000000004705702$	$\lambda_{54} = 0.873376934047102$	$\lambda_{11,9} = 0.253485990215933$
$\mu_{87} = 0.094887152674486$	$\lambda_{62} = 0.124127269944714$	$\lambda_{11,10} = 0.580400905152248$
$\mu_{88} = 0.041052752299292$	$\lambda_{63} = 0.120448606787528$	
$\mu_{94} = 0.000000000437894$	$\lambda_{64} = 0.000000113365798$	

A.2 Low-Storage Methods

Table A.23: Coefficients for the low-storage method RK44[2S]

i	γ_{i1}	γ_{i2}	$\beta_{i,i-1}$	δ_i
1	0.000000000000000	0.000000000000000	0.000000000000000	1.000000000000000
2	0.000000000000000	1.000000000000000	1.193743905974738	0.217683334308543
3	0.121098479554482	0.721781678111411	0.099279895495783	1.065841341361089
4	-3.843833699660025	2.121209265338722	1.131678018054042	0.000000000000000
5	0.546370891121863	0.198653035682705	0.310665766509336	

Table A.24: Coefficients for the low-storage method RK4()6[2S]

i	γ_{i1}	γ_{i2}	$\beta_{i,i-1}$	δ_i
1	0.000000000000000	0.000000000000000	0.000000000000000	1.000000000000000
2	0.000000000000000	1.000000000000000	0.238829375897678	0.564427596596565
3	0.344088773828091	0.419265952351424	0.467431873315953	1.906950911013704
4	-0.655389499112535	0.476868049820393	0.215210792473781	0.617263698427868
5	0.698092532461612	0.073840520232494	0.205665392762124	0.534245263673355
6	-0.463842390383811	0.316651097387661	0.803800094404076	0.000000000000000
8	0.730367815757090	0.058325491591457	0.076403799554118	

Table A.25: Coefficients for the low-storage method RK45[2S*]

i	γ_{i1}	γ_{i2}	$\beta_{i,i-1}$
1	0.000000000000000	0.000000000000000	0.000000000000000
2	0.000000000000000	1.000000000000000	0.357534921136978
3	-3.666545952121251	4.666545952121251	2.364680399061355
4	0.035802535958088	0.964197464041912	0.016239790859612
5	4.398279365655791	-3.398279365655790	0.498173799587251
6	0.770411587328417	0.229588412671583	0.433334235669763

Table A.26: Coefficients for the low-storage method RK4(3)6[2S]

i	γ_{i1}	γ_{i2}	$\beta_{i,i-1}$	δ_i
1	0.000000000000000	0.000000000000000	0.000000000000000	1.000000000000000
2	0.000000000000000	1.000000000000000	0.653858677151052	-1.662080444041546
3	1.587969352283926	0.888063312510453	0.258675602947738	1.024831293149243
4	1.345849277346560	-0.953407216543495	0.802263873737920	1.000354140638651
5	-0.088819115511932	0.798778614781935	0.104618887237994	0.093878239568257
6	0.206532710491623	0.544596034836750	0.199273700611894	1.695359582053809
7	-3.422331114067989	1.402871254395165	0.318145532666168	0.392860285418747

Table A.27: Coefficients for the low-storage method RK4(3)5[3S*]

i	γ_{i1}	γ_{i2}	γ_{i3}	$\beta_{i,i-1}$	δ_i
1	0.000000000000000	0.000000000000000	0.000000000000000	0.000000000000000	1.000000000000000
2	0.000000000000000	1.000000000000000	0.000000000000000	0.075152045700771	0.081252332929194
3	-0.497531095840104	1.384996869124138	0.000000000000000	0.211361016946069	-1.083849060586449
4	1.010070514199942	3.878155713328178	0.000000000000000	1.100713347634329	-1.096110881845602
5	-3.196559004608766	-2.324512951813145	1.642598936063715	0.728537814675568	2.859440022030827
6	1.717835630267259	-0.514633322274467	0.188295940828347	0.393172889823198	-0.655568367959557
7					-0.194421504490852

VITA

David Ketcheson is husband to Belky Ketcheson and father to two adorable daughters, Elena and Victoria. David earned the degree of Bachelor of Science, with majors in Mathematics and Physics & Astronomy, from Brigham Young University in 2004. He received the degrees of Master of Science and Doctor of Philosophy in Applied Mathematics from the University of Washington in 2008 and 2009, respectively. He has been appointed Assistant Professor of Applied Mathematics at King Abdullah University of Science and Technology.