Highly efficient strong stability preserving Runge-Kutta methods with Low-Storage Implementations

SIAM J. SCI. COMPUT.
Vol. 30, No. 4, pp. 2113–2136

HIGHLY EFFICIENT STRONG STABILITY-PRESERVING RUNGE–KUTTA METHODS WITH LOW-STORAGE IMPLEMENTATIONS*

DAVID I. KETCHESON†

Abstract. Strong stability-preserving (SSP) Runge–Kutta methods were developed for time integration of semidiscretizations of partial differential equations. SSP methods preserve stability properties satisfied by forward Euler time integration, under a modified time-step restriction. We consider the problem of finding explicit Runge–Kutta methods with optimal SSP time-step restrictions, first for the case of linear autonomous ordinary differential equations and then for nonlinear or nonautonomous equations. By using alternate formulations of the associated optimization problems and introducing a new, more general class of low-storage implementations of Runge–Kutta methods, new optimal low-storage methods and new low-storage implementations of known optimal methods are found. The results include families of low-storage second and third order methods that achieve the maximum theoretically achievable effective SSP coefficient (independent of stage number), as well as low-storage fourth order methods that are more efficient than current full-storage methods. The theoretical properties of these methods are confirmed by numerical experiment.

Key words. method of lines, strong stability-preserving, monotonicity, low-storage, Runge–Kutta methods

AMS subject classifications. Primary, 65M20; Secondary, 65L06

DOI. 10.1137/07070485X

1. Introduction. Strong stability-preserving (SSP) methods are numerical methods for solving ordinary differential equations (ODEs) that preserve monotonicity and contractivity properties of the numerical solution under certain assumptions on the equations and the time step.

Development of SSP methods was originally motivated by the following consideration. Solutions of many important partial differential equations (PDEs)

$$u_t = f(t, u, u_x, u_{xx}, \dots) \tag{1.1}$$

satisfy a *monotonicity* property:

$$||u(t+\tau)|| \le ||u(t)|| \quad \forall \tau \ge 0. \tag{1.2}$$

Here $||\cdot||$ may be a norm or, more generally, any convex functional. For instance, the solutions to scalar hyperbolic conservation laws are monotonic in the total variation seminorm. When employing a discrete approximation to the PDE (1.1), it is often important to satisfy the discrete monotonicity property analogous to (1.2)

$$||\boldsymbol{u}^{n+1}|| \le ||\boldsymbol{u}^n||, \tag{1.3}$$

where the vector $\boldsymbol{u}^n \in \mathbb{R}^N$ is a discrete approximation of $u(t^n)$. However, it is very difficult to find high-order full discretizations that satisfy the discrete monotonicity property (1.3) directly.

*Received by the editors October 8, 2007; accepted for publication (in revised form) January 22, 2008; published electronically May 16, 2008. This work was funded by a U.S. Department of Energy Computational Science Graduate Fellowship under grant DE-FG02-97ER25308.
http://www.siam.org/journals/sisc/30-4/70485.html
†Department of Applied Mathematics, University of Washington, Seattle, WA 98195-2420 (ketch@amath.washington.edu, http://www.amath.washington.edu/∼ketch/).

A common approach to solving PDEs is the method of lines, which involves discretizing in space to obtain a system of ODEs

$$\boldsymbol{u}'(t) = F(t, \boldsymbol{u}) \tag{1.4}$$

and integrating this system by using an ODE solver. Again the vector $\boldsymbol{u} \in \mathbb{R}^N$ is a discrete approximation of the function u, and the function $F : \mathbb{R} \times \mathbb{R}^N \to \mathbb{R}^N$ is a discrete approximation of the function f.

We now assume that the semidiscretization is performed so that the solution of the resulting system of ODEs satisfies the same monotonicity property (1.2) under forward Euler integration, i.e.,

$$||\boldsymbol{u}(t) + \Delta t F(t, \boldsymbol{u}(t))|| \leq ||\boldsymbol{u}(t)|| \ \ \forall \boldsymbol{u}, \ \forall \Delta t \leq \Delta t_{\text{FE}}. \tag{1.5}$$

Here Δt_{FE} represents a fixed maximal time step for which (1.5) holds. In the ODE literature, condition (1.5) has been referred to as a *circle condition* because, in the case that F is linear, (1.5) implies that the eigenvalues of F lie in a circle of radius Δt_{FE} centered at $\lambda = -\Delta t_{\text{FE}}$.

A solution obtained by using an ODE solver other than forward Euler may in general violate the monotonicity property (1.3); however, if the integration is performed by using a strong stability-preserving ODE solver, then monotonicity will be assured if the time step satisfies

$$\Delta t \leq c \Delta t_{\text{FE}}, \tag{1.6}$$

where c is the *SSP coefficient* of the method. This leads to a simpler way of proving monotonicity of the fully discrete scheme, since Δt_{FE} depends only on the spatial discretization and c depends only on the temporal discretization. Just as the method of lines separates the analysis of accuracy for the spatial and temporal discretizations, the SSP approach separates the analysis of (temporal) monotonicity for the spatial and temporal discretizations.

In the numerical solution of systems of ODEs, it is desirable to ensure that errors in the initial conditions or numerical errors at a given step do not grow unduly as they are propagated in subsequent steps. Given two approximate solutions $\boldsymbol{u}^n$ and $\tilde{\boldsymbol{u}}^n$ at time t^n and by letting $\boldsymbol{u}^{n+1}$ and $\tilde{\boldsymbol{u}}^{n+1}$ denote the corresponding numerical solutions at the next time step t^{n+1}, the numerical solution is said to be *contractive* if

$$||\boldsymbol{u}^{n+1} - \tilde{\boldsymbol{u}}^{n+1}|| \leq ||\boldsymbol{u}^n - \tilde{\boldsymbol{u}}^n||. \tag{1.7}$$

By interpreting $\tilde{\boldsymbol{u}}^n$ as a perturbation of $\boldsymbol{u}^n$ due to numerical errors, we see that contractivity implies that these errors do not grow as they are propagated. If the system of ODEs (1.4) is such that the contractivity property (1.7) holds under forward Euler integration subject to some maximal time step Δt_{FE}, then (1.7) will also be satisfied under integration with an SSP method, under the modified time-step restriction (1.6). In the remainder of this work, the discussion will focus on monotonicity rather than contractivity, but the main results apply to both cases (for more details regarding contractivity preservation, see [19]).

In the special case that the relevant convex functional $||\cdot||$ is an inner product norm, then monotonicity and contractivity are preserved by the broader class of circle contractive methods under a less stringent time-step restriction [2, 12].

SSP time discretizations were originally introduced in [27], where they were referred to as TVD methods. Among the Runge–Kutta methods presented there, 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 time step, among all two-stage, second order and three-stage, third order methods, respectively.

When the time step is limited by monotonicity or contractivity, the computational efficiency of a method may be measured by the *effective SSP coefficient*:

$$c_{\text{eff}} = \frac{c}{s}, \tag{1.8}$$

where s is the number of function evaluations per step (the number of stages for Runge–Kutta methods). The work required to solve a problem is inversely proportional to c_{eff}. By definition, the forward Euler method has $c_{\text{eff}} = 1$.

Among the various classes of ODE solvers, explicit Runge–Kutta methods have proven to have the best potential for large effective SSP coefficients. Results for SSP multistep methods are disappointing, in the sense that the SSP coefficients are very small [26, 9, 14, 6] (even for implicit multistep methods, $c \leq 2$ for methods of higher than first order accuracy). By considering a weaker property wherein particular starting procedures are prescribed, Ruuth and Hundsdorfer [23] have developed methods that are competitive in some cases with optimal 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 the present work.

Implicit SSP Runge–Kutta methods of higher than first order accuracy appear to be subject to the bound $c_{\text{eff}} \leq 2$ [16], whereas explicit Runge–Kutta methods can have $c_{\text{eff}} \approx 1$. Typically the cost of the implicit solve more than doubles the work per time step, making explicit SSP Runge–Kutta methods more efficient. Therefore, in this paper we consider only explicit Runge–Kutta methods. However, we note that explicit SSP Runge–Kutta methods cannot be more than fourth order accurate [19] (unless they involve downwinding), while implicit SSP Runge–Kutta methods can have order of accuracy up to six [16].

Extensive efforts have been made to find more efficient explicit SSP Runge–Kutta methods [19, 8, 9, 29, 30, 22]. 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 [8, 9, 22]. Some attention has also been paid to finding optimal SSP methods for linear systems [18, 7].

In all previous searches, it has been found that optimizing either one of efficiency or memory leads to the other being less optimal. Nevertheless, in this work we present methods that achieve the theoretical optimum simultaneously in both properties.

The improved results in the current work are due to two factors. First, the optimization problem is formulated by using the Butcher form and a simplified algebraic characterization of the SSP coefficient, as suggested in [4]. This allows for solution of the optimization problem for much larger numbers of stages. Second, we consider a much more general class of low-storage methods, by using the Shu–Osher form itself to facilitate low-storage implementations.

The paper is organized as follows: in section 2 we discuss strong stability-preserving methods for linear autonomous systems and present an efficient algorithm for computing optimal methods. In section 3 we discuss strong stability-preserving

methods for nonlinear, nonautonomous systems and present optimal methods. In section 5 we verify the properties of the methods by using simple numerical tests.

2. Methods for linear autonomous systems.

2.1. Strong stability preservation for linear systems. 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 $\boldsymbol{b}$ and $\boldsymbol{c}$. The Runge–Kutta method defined by these arrays approximates the solution of the system of ODEs (1.4) by the iteration

$$\begin{aligned} \boldsymbol{y}_i &= \boldsymbol{u}^n + \Delta t \sum_{j=1}^{s} a_{ij} F\left(t_n + c_j \Delta t, \boldsymbol{y}_j\right), \quad 1 \le i \le s, \\ \boldsymbol{u}^{n+1} &= \boldsymbol{u}^n + \Delta t \sum_{j=1}^{s} b_j F\left(t_n + c_j \Delta t, \boldsymbol{y}_j\right). \end{aligned} \tag{2.1}$$

It is convenient to define the $(s+1) \times s$ matrix

$$\mathbf{K} = \begin{pmatrix} \mathbf{A} \\ \boldsymbol{b}^{\mathrm{T}} \end{pmatrix}. \tag{2.2}$$

We will always make the standard assumption that $c_i = \sum_{j=1}^{s} a_{ij}$. For explicit methods (which are the subject of this work), we also have $a_{ij} = 0$ for $j \ge i$.

When applied to a linear autonomous system of ODEs

$$\boldsymbol{u}_t = \mathbf{L}\boldsymbol{u}, \tag{2.3}$$

the Runge–Kutta method (2.1) reduces to the iteration

$$\boldsymbol{u}^{n+1} = \phi(\Delta t \mathbf{L})\boldsymbol{u}^n, \tag{2.4}$$

where ϕ is a rational function called the *stability function* of the Runge–Kutta method [10].

When solving (2.3), the time-step restriction for strong stability preservation depends on the radius of absolute monotonicity of ϕ.

Definition 2.1 (radius of absolute monotonicity (of a function)). *The radius of absolute monotonicity $R(\psi)$ of a function ψ is the largest value of r such that $\psi(x)$ and all of its derivatives exist and are nonnegative for $x \in (-r, 0]$.*

We now consider linear autonomous systems of ODEs that satisfy the monotonicity property (1.3) under a forward Euler step; this leads to the linear version of the circle condition (1.5):

$$||\boldsymbol{u} + \Delta t_{\mathrm{FE}} \mathbf{L}\boldsymbol{u}|| \le ||\boldsymbol{u}||. \tag{2.5}$$

Note that, if $||\cdot||$ represents a norm, this reduces to $||I + \Delta t_{\mathrm{FE}}\mathbf{L}|| \le 1$, where $||\cdot||$ is the induced matrix norm.

Let the class $\mathcal{L}(h)$ denote the set of all pairs $(\mathbf{L}, ||\cdot||)$, where the matrix $\mathbf{L} \in R^{N \times N}$ and convex functional $||\cdot||$ are such that the circle condition (2.5) is satisfied with

$\Delta t_{\mathrm{FE}} = h$. Let $(\mathbf{L}, ||\cdot||) \in \mathcal{L}(\Delta t_{\mathrm{FE}})$, and let ϕ denote the stability function of a consistent Runge–Kutta method. Consider the Taylor expansion of $\phi(x)$ about $x = -r$:

$$\phi(x) = \sum_{i=0}^{s} \gamma_i \left(1 + \frac{x}{r}\right)^i. \tag{2.6}$$

Suppose that ϕ is absolutely monotonic at $-r$ or, in other words, that all of the coefficients are nonnegative: $\gamma_i \geq 0$, and let $\Delta t \leq r\Delta t_{\mathrm{FE}}$. Then

$$||\boldsymbol{u}^{n+1}|| = ||\phi(\Delta t\mathbf{L})\boldsymbol{u}^n|| = \left|\left|\sum_{i=0}^{s} \gamma_i \left(I + \frac{\Delta t}{r}\mathbf{L}\right)^i \boldsymbol{u}^n\right|\right| \tag{2.7}$$

$$\leq \sum_{i=0}^{s} \gamma_i \left|\left|\left(\boldsymbol{u}^n + \frac{\Delta t}{r}\mathbf{L}\boldsymbol{u}^n\right)\right|\right|^i \tag{2.8}$$

$$\leq \sum_{i=0}^{s} \gamma_i ||\boldsymbol{u}^n|| = ||\boldsymbol{u}^n||. \tag{2.9}$$

Here we have used the fact that $\sum_i \gamma_i = 1$ for any consistent method. Hence the numerical solution will be monotonic if $\Delta t \leq R(\phi)\Delta t_{\mathrm{FE}}$. This is a sufficient condition; to state precisely the sense in which it is necessary, we need the following definition.

DEFINITION 2.2 (strong stability preservation for linear systems). *Given a Runge–Kutta method, the linear SSP coefficient of the method is the largest constant $c^{\mathrm{lin}} \geq 0$ such that the numerical solution obtained with the Runge–Kutta method satisfies the monotonicity property $||\boldsymbol{u}^{n+1}|| \leq ||\boldsymbol{u}^n||$ for all $(\mathbf{L}, ||\cdot||) \in \mathcal{L}(\Delta t_{\mathrm{FE}})$ whenever*

$$\Delta t \leq c^{\mathrm{lin}} \Delta t_{\mathrm{FE}}. \tag{2.10}$$

If $c^{\mathrm{lin}} > 0$, the method is said to be strong stability-preserving for linear systems.

The following result is due to Spijker [28].

THEOREM 1. *Let a Runge–Kutta method be given with stability function ϕ. Let c^{lin} denote the linear SSP coefficient of the method (see Definition 2.2). Let $R(\phi)$ denote the radius of absolute monotonicity of ϕ (Definition 2.1). Then*

$$c^{\mathrm{lin}} = R(\phi). \tag{2.11}$$

In other words, the method preserves strong stability for problems in $\mathcal{L}(\Delta t_{\mathrm{FE}})$ under the (maximal) time-step restriction

$$\Delta t \leq R(\phi)\Delta t_{\mathrm{FE}}. \tag{2.12}$$

Remark 1. This time-step restriction is sharp only in the sense of considering all of $\mathcal{L}(\Delta t_{\mathrm{FE}})$. By considering a specific matrix $\mathbf{L}$, convex functional $||\cdot||$, and initial condition $\boldsymbol{u}(0)$, it may be possible to derive a larger time-step restriction that still preserves monotonicity. For instance, when $||\cdot||$ is a norm, the monotonicity condition reduces to $||\phi(\Delta t\mathbf{L})|| \leq 1$. Thus one can directly find the maximal time step such that this condition holds, which may be larger than that given by (2.12).

Remark 2. Under the assumption that $\gamma_i \geq 0$, the Taylor expansion (2.6) may be thought of as rewriting the Runge–Kutta method as a convex combination of (iterated) forward Euler steps of length at most Δt_{FE}. This reasoning was used directly in [9, 7] to find optimal SSP methods for linear systems.

Remark 3. It is interesting to note that in [28] the necessity of the time-step restriction (2.12) was demonstrated by considering the maximum norm and the matrix $\mathbf{L}$ corresponding to a first order upwind differencing approximation of the advection equation. We will return to this point in section 5.1.

2.2. Optimal methods for linear systems. By virtue of Theorem 1, for solution of linear autonomous ODEs, the search for optimal SSP methods reduces to maximization of $R(\phi)$, the radius of absolute monotonicity of the stability function. For explicit Runge–Kutta methods with s stages and order p, $\phi(x)$ is a polynomial of degree s that approximates the exponential function to order p near $x = 0$. Kraaijevanger [18] found optimal methods for many values of s and p, including $1 \leq p \leq s \leq 10$, and $p \in \{1, 2, 3, 4, s-1, s-2, s-3, s-4\}$ for any s. He also provided an algorithm for the computation of the optimal coefficient and method for arbitrary s and 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 [9, 7] 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).

By 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 $s \geq 20$, $p \approx s/2$, the solution would require years of computation. A more efficient method of solution for arbitrary s and p is described below.

Let $\mathbf{R}_{s,p}(\phi)$ denote the maximum of $r(\phi)$ over all methods with s stages and order of accuracy p, and let $\Phi_{s,p}$ denote the corresponding optimal polynomial. By writing the stability function in terms of its Taylor series about $x = 0$:

$$\phi(x) = \sum_{i=0}^{s} \bar{\gamma}_i x^i, \tag{2.13}$$

the problem of finding $\mathbf{R}_{s,p}(\phi)$ can be written

maximize r subject to

$$\bar{\gamma}_i = \frac{1}{i!} \qquad (0 \leq i \leq p), \tag{2.14a}$$

$$\phi^{(k)}(x) = \sum_{i=k}^{s-k} \left(\prod_{j=i-k+1}^{i} j \right) \bar{\gamma}_i x^{i-k} \geq 0 \qquad (-r \leq x \leq 0). \tag{2.14b}$$

The last inequality is required for $0 \leq k \leq s$. This appears to be a difficult problem, because the domain of the inequality constraints depends on the objective function.

Representing ϕ via its Taylor expansion about $x = -r$, as in (2.6) above, leads to the alternate formulation [18]

maximize r subject to

$$\sum_{j=0}^{s} \left(\prod_{k=0}^{i-1} (j-k) \right) \gamma_j = r^i \qquad (0 \leq i \leq p), \tag{2.15a}$$

$$\gamma_j \geq 0 \qquad (0 \leq j \leq s), \tag{2.15b}$$

with the convention $\prod_{k=0}^{-1} = 1$. The advantage of this formulation is that the inequality constraints do not explicitly involve r, yet they imply that ϕ is absolutely monotonic on $(-r, 0]$. By defining

$$\mathbf{B}_{ij} = \sum_{j=0}^{s} \left(\prod_{k=0}^{i-1} (j-k) \right), \tag{2.16a}$$

$$\boldsymbol{d}(r)_i = r^i, \tag{2.16b}$$

we can write (2.15) as

$$\text{maximize } r \text{ subject to}$$

$$\mathbf{B}\boldsymbol{\gamma} = \boldsymbol{d}(r), \tag{2.17a}$$

$$\boldsymbol{\gamma} \geq 0. \tag{2.17b}$$

For a fixed value of r, (2.17) is just a linear programming feasibility problem. We wish to find the largest value of r such that the problem defined by these constraints is feasible. This can be done by using bisection. We have implemented this approach by using MATLAB and the semidefinite programming solver SeDuMi. Unfortunately, even after rescaling to improve conditioning, SeDuMi typically finds solutions to only about five digits of accuracy. We have used the following approach to obtain approximately ten digits of accuracy and also to obtain the optimal polynomials.

It is shown in [19] that, for the optimal polynomial, at most p of the components of $\boldsymbol{\gamma}$ are nonzero. We have found that the zero and nonzero elements of $\boldsymbol{\gamma}$ are well separated in magnitude in the numerical solution given by SeDuMi. By letting R denote the optimal value, we have

$$\hat{\mathbf{B}}\hat{\boldsymbol{\gamma}} = \boldsymbol{d}(R), \tag{2.18}$$

where $\hat{\boldsymbol{\gamma}}$ includes only the nonzero components of $\boldsymbol{\gamma}$ and $\hat{\mathbf{B}}$ includes only the corresponding columns of $\mathbf{B}$. We can find R by forming the matrix $[\hat{\mathbf{B}}\ \boldsymbol{d}(r)]$ and varying r to minimize its smallest singular value (i.e., enforcing the condition that $\boldsymbol{d}(r)$ be in the range of $\hat{\mathbf{B}}$). Finally, to find the optimal polynomial itself, we compute

$$\hat{\boldsymbol{\gamma}} = \hat{\mathbf{B}}^{-1}\boldsymbol{d}(R). \tag{2.19}$$

Then $\Phi_{s,p}$ is given by (2.6).

For large values of s, numerical underflow occurs in the representation of $\mathbf{B}$ because of the large difference in magnitude of its components. This is the limitation on the values s and p that this method is able to solve. Even so, it is capable of quickly solving cases with very large s and modest p (e.g., $s = 10000, p = 3$).

Table 2.1 lists values of $\mathbf{R}_{s,p}(\phi)$ for $s \leq 16, p \leq 30$. For a discussion of many interesting properties of this table, see [19]. Because we are primarily interested in $\mathbf{R}_{s,p}(\phi)$ as an upper bound on the SSP coefficient for nonlinear methods (see the next section), and to save space, we do not give the optimal polynomials here. They may easily be computed with the algorithm just described. A simple MATLAB implementation of this algorithm is available from the author's website.

TABLE 2.1
Optimal SSP coefficients for linear problems: $\mathbf{R}_{s,p}(\phi)$. *Boldface entries represent new results obtained in the present work.*

s \ p	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**

3. Methods for nonlinear systems.

3.1. Strong stability preservation for nonlinear systems. We now consider general (nonlinear, nonautonomous) systems of ODEs. Analogous to $\mathcal{L}(h)$, we define the class $\mathcal{F}(h)$ as the set of all pairs $(F, ||\cdot||)$, where the function F and convex functional $||\cdot||$ are such that the circle condition (1.5) is satisfied with $\Delta t_{\text{FE}} = h$. The following definition is the nonlinear analogue of Definition 2.2.

DEFINITION 3.1 (strong stability preservation). *The SSP coefficient of a Runge–Kutta method* (2.1) *is the largest constant* $c \geq 0$ *such that the numerical solution obtained with the Runge–Kutta method satisfies the monotonicity properties*

$$||\boldsymbol{y}_i|| \leq ||\boldsymbol{u}^n|| \qquad (1 \leq i \leq s), \tag{3.1a}$$

$$||\boldsymbol{u}^{n+1}|| \leq ||\boldsymbol{u}^n|| \tag{3.1b}$$

for all $(F, ||\cdot||) \in \mathcal{F}(\Delta t_{\text{FE}})$ *whenever*

$$\Delta t \leq c \Delta t_{\text{FE}}. \tag{3.2}$$

If $c > 0$*, the method is said to be strong stability-preserving.*

In the case of linear systems, we saw that monotonicity is preserved if the method can be rewritten as a convex combination of forward Euler steps. For the case of nonlinear systems, it is necessary to rewrite each stage as such a combination. To this end, an alternative representation for explicit Runge–Kutta methods was introduced by Shu and Osher[1] [27]:

$$\boldsymbol{y}_0 = \boldsymbol{u}^n, \tag{3.3a}$$

$$\boldsymbol{y}_i = \sum_{j=0}^{i-1} \left(\alpha_{i,j} \boldsymbol{y}_k + \Delta t \beta_{i,j} F(t_n + c_{k-1} \Delta t, \boldsymbol{y}_k) \right), \quad \alpha_{i,j} \geq 0, \quad i = 1, \ldots, s, \tag{3.3b}$$

$$\boldsymbol{u}^{n+1} = \boldsymbol{y}_s. \tag{3.3c}$$

Care should be taken to avoid confusing the SSP coefficient c with the abscissas of the method c_k; the latter will always have a subscripted index. Consistency requires that $\sum_{j=0}^{i-1} \alpha_{i,j} = 1$. Thus, if $\alpha_{i,j} \geq 0$ and $\beta_{i,j} \geq 0$, all of the intermediate stages $\boldsymbol{y}_i$ in (3.3) are simply convex combinations of forward Euler steps, with Δt replaced by $\frac{\beta_{i,j}}{\alpha_{i,j}} \Delta t$. Therefore, the solution obtained with this method will satisfy the monotonicity property (1.3) for all $(F, ||\cdot||) \in \mathcal{F}(\Delta t_{\text{FE}})$ under the time-step restriction $\frac{\beta_{i,j}}{\alpha_{i,j}} \Delta t \leq \Delta t_{\text{FE}}$ or, equivalently,

$$\Delta t \leq \min_{i,j} \frac{\alpha_{i,j}}{\beta_{i,j}} \Delta t_{\text{FE}} = \hat{c}(\boldsymbol{\alpha}, \boldsymbol{\beta}), \tag{3.4}$$

where the minimum is taken over all i, j such that $\beta_{i,j} \neq 0$.

The strategy of rewriting a method as a convex combination of forward Euler steps is analogous to our approach in the case of linear systems (cf. Remark 2). In that case

[1] Note that stage $\boldsymbol{y}_i$ here corresponds to stage $\boldsymbol{y}_{i+1}$ in the Butcher representation, which accounts for the shifted subscript of the abscissa.

we arrived at the same criteria by considering absolute monotonicity of the stability function. In [19], by considering absolute monotonicity of the matrix-valued K-function, and algebraic criterion for contractivity preservation was derived. This criterion involves the *radius of absolute monotonicity* of the Runge–Kutta method $R(\mathbf{K})$, also introduced originally by Kraaijevanger [19]. A more convenient, equivalent definition of $R(\mathbf{K})$ was given in [3, 11], and we repeat it here.

DEFINITION 3.2 (radius of absolute monotonicity (of a Runge–Kutta method)). *The radius of absolute monotonicity $R(\mathbf{K})$ of the Runge–Kutta method defined by Butcher array $\mathbf{K}$ is the largest value of r such that $(I + r\mathbf{A})^{-1}$ exists and*

$$\begin{aligned} \mathbf{K}(I + r\mathbf{A})^{-1} &\geq 0, \\ r\mathbf{K}(I + r\mathbf{A})^{-1}\boldsymbol{e}_s &\leq \boldsymbol{e}_{s+1}. \end{aligned} \tag{3.5}$$

Here the inequalities are componentwise, and $\boldsymbol{e}_s$ denotes the $s \times 1$ vector of ones.

The following result follows from Propositions 2.1, 2.2, and 2.7 of [13].

THEOREM 2. *Consider an explicit Runge–Kutta method with Butcher array $\mathbf{K}$ and a Shu–Osher representation $\boldsymbol{\alpha}, \boldsymbol{\beta}$. Let $\hat{c}(\boldsymbol{\alpha}, \boldsymbol{\beta})$ be defined by* (3.4)*, let $R(\mathbf{K}) < \infty$ denote the radius of absolute monotonicity defined by* (3.5)*, and let c denote the SSP coefficient of Definition* 3.1*. Then $\hat{c}(\boldsymbol{\alpha}, \boldsymbol{\beta}) \leq R(\mathbf{K}) = c$. Furthermore, there exists a Shu–Osher representation $\boldsymbol{\alpha}, \boldsymbol{\beta}$ such that $\hat{c}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = R(\mathbf{K})$. In other words, the method produces a monotonic solution for all $(F, ||\cdot||) \in \mathcal{F}(\Delta t_{\text{FE}})$ under the (maximal) time-step restriction*

$$\Delta t \leq R(\mathbf{K})\Delta t_{\text{FE}}. \tag{3.6}$$

Remark 4. As in the linear case, this time-step restriction is sharp when the entire class $\mathcal{F}(\Delta t_{\text{FE}})$ is considered. However, by considering specific $F, ||\cdot||$, and $\boldsymbol{u}(0)$, it may be possible to derive a larger time-step restriction that preserves monotonicity. For instance, when F is a nonautonomous linear function and $||\cdot||$ is a norm, one may consider the induced matrix norm of the matrix-valued K-function of the method (see section 5.2 below).

3.2. A new class of low-storage Runge–Kutta methods. Efforts to find SSP methods have focused on finding the method with the maximal SSP coefficient for a prescribed order, number of stages, and (sometimes) number of memory registers. However, the number of stages is important only 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.

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 [33, 15]. 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 [33] 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 [15], 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 semidiscretizations of hyperbolic PDEs.

The low-storage methods of van der Houwen type [15] make instead the assumption that it is possible to make assignments of the form

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

again without significant additional storage. This is apparently reasonable for compressible Navier–Stokes calculations 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 third storage register. This assumption implies the assumptions necessary for implementation of Williamson and van der Houwen methods; hence, the class of semidiscretizations 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 mN method. Sometimes it is necessary to retain the value of the solution at the previous time step, 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. Such methods are denoted by mN*.

3.3. Optimal methods for nonlinear systems. Extensive efforts have been made to find optimal explicit SSP Runge–Kutta methods by both analysis and numerical search [19, 8, 9, 29, 30, 25]. Except for [19], all of these efforts formulated the optimization problem by 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 [5] 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. We have applied both analytical and numerical methods to this latter formulation.

The optimization problem is formulated as

$$\text{maximize } r \text{ subject to}$$

$$\mathbf{K}_{ij} = 0 \qquad (j \geq i), \tag{3.7a}$$

$$\tau_k(\mathbf{K}) = 0 \qquad (k \leq p), \tag{3.7b}$$

$$\mathbf{K}(I + r\mathbf{A})^{-1} \geq 0, \tag{3.7c}$$

$$r\mathbf{K}(I + r\mathbf{A})^{-1} e_s \leq e_{s+1}, \tag{3.7d}$$

where τ_k represents the set of order conditions for order k (for an enumeration of these conditions, see, e.g., Appendix B of [15]).

Note that $c \le c^{\text{lin}}$ by definition, so the values of $\mathbf{R}_{s,p}(\phi)$ in Table 2.1 are upper bounds on the value of 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 $c^{\text{lin}} \le s$, it follows that $c_{\text{eff}} \le 1$ (in fact, $c_{\text{eff}} < 1$ 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 c_{eff} as close as possible to 1.

The first order accurate forward Euler method has $c_{\text{eff}} = 1$ and can be implemented in 1N* fashion; hence consideration of first order SSP methods with more stages is superfluous. Since explicit SSP Runge–Kutta methods have order at most four [24], it remains to consider methods of order two through four.

3.3.1. Second order methods.

2N* methods. Optimal second order methods with c_{eff} arbitrarily close to 1 were found in [19] and later independently in [29]. The s-stage method in this family has SSP coefficient $s-1$; hence, $c_{\text{eff}} = \frac{s-1}{s}$. The nonzero entries of the low-storage form for the s-stage method are

$$\beta_{i,i-1} = \begin{cases} \frac{1}{s-1} & 1 \le i \le s-1, \\ \frac{1}{s} & i = s, \end{cases} \tag{3.8a}$$

$$\alpha_{i,i-1} = \begin{cases} 1 & 1 \le i \le s-1, \\ \frac{s-1}{s} & i = s, \end{cases} \tag{3.8b}$$

$$\alpha_{s,0} = \frac{1}{s}. \tag{3.8c}$$

The abscissas are

$$c_i = \frac{i-1}{s-1} \quad (1 \le i \le s). \tag{3.9}$$

Note that these methods do not require a third register even if the previous time step 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 $s > 2$. In fact, in the same paper that announced the family (3.8), other "optimal" low-storage methods of order two (with smaller SSP coefficients) were also given [29].

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. These descriptions are intended

```
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;
```

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

to serve as pseudocode for a truly efficient implementation. The storage registers in the pseudocode are denoted by q_1 and q_2. A low-storage implementation of (3.8) is given in Pseudocode 1.

3.3.2. Third order methods.

2N* methods. The three- and four-stage third order SSP Runge–Kutta methods, originally reported in [27] and [19], respectively, can be implemented with just two memory registers, even if the previous time step 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 ($c_{\text{eff}} = 1/2$).

2N methods. We now consider methods that can be implemented with only two registers if the solution at the previous time step can be discarded. Of course, the previous time step can be retained at the cost of using one more register.

Low-storage third order methods were found in [22]; the best 2N method has $c_{\text{eff}} = 0.297$; the best 3N method has $c_{\text{eff}} = 0.513$.

We have found third order 2N methods with $c = \mathbf{R}_{s,3}(\phi)$ and up to $m = 10$ stages. The most efficient of these has $c_{\text{eff}} = 0.68$. However, these methods are superseded already by the following family of methods, which achieve $c_{\text{eff}} \approx 1$.

THEOREM 3. *Let $n > 2$ be an integer, and let $s = n^2$. Then there exists a third order s-stage method with SSP coefficient $c = \mathbf{R}_{s,3}(\phi) = n^2 - n$. The nonzero coefficients of the method are*

$$\alpha_{i,i-1} = \begin{cases} \frac{n-1}{2n-1} & i = \frac{n(n+1)}{2}, \\ 1 & \textit{otherwise}, \end{cases} \tag{3.10a}$$

$$\alpha_{\frac{n(n+1)}{2},\frac{(n-1)(n-2)}{2}} = \frac{n}{2n-1}, \tag{3.10b}$$

$$\beta_{i,i-1} = \frac{\alpha_{i,i-1}}{n^2-n}. \tag{3.10c}$$

The abscissas of the method are

$$c_i = \frac{i-1}{n^2-n} \qquad \textit{for } 1 \le i \le (n+2)(n-1)/2, \tag{3.11a}$$

$$c_i = \frac{i-n-1}{n^2-n} \qquad \textit{for } (n+2)(n+1)/2 \le i \le n^2. \tag{3.11b}$$

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

```
n=sqrt(s); 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;
```

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

The proof of the theorem is straightforward. The SSP coefficient can be found by using (3.4) and Theorem 2, while the satisfaction of the order conditions can be verified by forming the Butcher array and checking directly. The optimality follows from $c \leq R_{s,p}(\phi)$ and the fact that $R_{n^2,3} = n^2 - n$ [18].

Like the second order family (3.8) above, this family of methods achieves effective SSP coefficients arbitrarily close to one while using only two memory registers. Also, like the family (3.8), and unlike most known optimal third order SSP methods, the coefficients are simple rational numbers. Note that the four-stage 2N* method discussed above is the first member of this family. A low-storage implementation of (3.10) is given in Pseudocode 2.

Remark 5. The family (3.10) was not found by numerical search; we have discovered that, for each value of $s \leq 15$, there exists at least a one-parameter family of third order methods with $c = \mathbf{R}_{s,3}(\phi)$; hence the particular methods (3.10) 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.

3.3.3. Fourth order methods. No explicit fourth order method with four stages has $c > 0$ [19, 8]. The optimal five stage method was found in [19] and again independently in [29]; this method has $c_{\text{eff}} = 0.302$. More efficient methods with up to eight stages were found in [30, 22, 21]; the most efficient (eight-stage) method has $c_{\text{eff}} = 0.518$ and can be implemented in 3N fashion.

Low-storage fourth order SSP methods were found in [22]; no 2N methods are reported, and the best 3N method has $c_{\text{eff}} = 0.106$. Even by allowing downwinding, the best 3N method reported there has $c_{\text{eff}} = 0.187$. 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.

2N methods. No 2N 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. Additionally, this is the only fourth order SSP

```
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;
```

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

method to be analytically proved optimal, because it achieves the optimal bound on ten-stage, fourth order SSP methods for linear problems: $c = \mathbf{R}_{10,4}(\phi) = 6$. Finally, the method has simple rational coefficients. The nonzero coefficients are

$$\beta_{i,i-1} = \begin{cases} \frac{1}{6} & i \in \{1 \dots 4, 6 \dots 9\}, \\ \frac{1}{15} & i = 5, \\ \frac{1}{10} & i = 10, \end{cases}$$

$$\beta_{10,4} = \frac{3}{50},$$

$$\alpha_{i,i-1} = \begin{cases} 1 & i \in \{1 \dots 4, 6 \dots 9\}, \\ \frac{2}{5} & i = 5, \\ \frac{3}{5} & i = 10, \end{cases}$$

$$\alpha_{5,0} = \frac{3}{5},$$

$$\alpha_{10,0} = \frac{1}{25},$$

$$\alpha_{10,4} = \frac{9}{25}.$$

The abscissas are

$$c = \frac{1}{6} \cdot (0, 1, 2, 3, 4, 2, 3, 4, 5, 6)^T. \tag{3.12}$$

Remark 6. 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, fourth order linear SSP Runge–Kutta method is closely related to the optimal s-stage, third order linear SSP Runge–Kutta method [18]. Based on 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 [22]. Furthermore, after extensive analytical and numerical searches, we have been unable to find a method of this type for $n = 4$.

A 2N implementation of the ten-stage method is given in Pseudocode 3.

TABLE 3.1

Properties of some popular and optimal SSP Runge–Kutta methods. Methods and properties in bold indicate new contributions in the present work. An asterisk indicates that the previous time step can be retained without increasing the required number of registers.

Popular method	c_{eff}	Storage	Improved method	c_{eff}	Storage
SSPRK(2,2)	0.500	2N*	SSPRK(s,2)	$1-1/s$	**2N***
SSPRK(3,3)	0.333	**2N***	SSPRK(4,3)	0.500	**2N***
			SSPRK(n^2,3)	$\mathbf{1-1/n}$	**2N**
SSPRK(5,4)	0.377	3N	**SSPRK(10,4)**	**0.600**	**2N**
			SSPRK(26,4)	**0.696**	**3N**

3N methods. We have found many 3N methods with more than ten stages that are more efficient than the 2N ten-stage method above. It appears likely that with three registers it is possible to obtain fourth order methods with c_{eff} arbitrarily close to unity. However, the value of c_{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 $c_{\text{eff}} \approx 0.696$.

The results of this section are summarized in Table 3.1, which lists the effective SSP coefficient and storage requirements for some well-known SSP methods and for improved methods with extra stages.

4. Linear stability and truncation error analysis.

4.1. Linear stability analysis. 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 (2.4), characterized by the stability function ϕ. For the case of a single linear ODE, this is simply $\mathbf{u}^{n+1} = \phi(\lambda\Delta t)\mathbf{u}^n$. The method is said to be absolutely stable for values of z such that $|\phi(z)| < 1$. For the optimal second order SSP family (3.8),

$$\phi(z) = \frac{1}{s} + \frac{s-1}{s}\left(1 + \frac{z}{s-1}\right)^{s-1}. \tag{4.1}$$

These are, of course, the same optimal polynomials found in section 2 for the s-stage second order cases. For the optimal third order SSP methods (3.10),

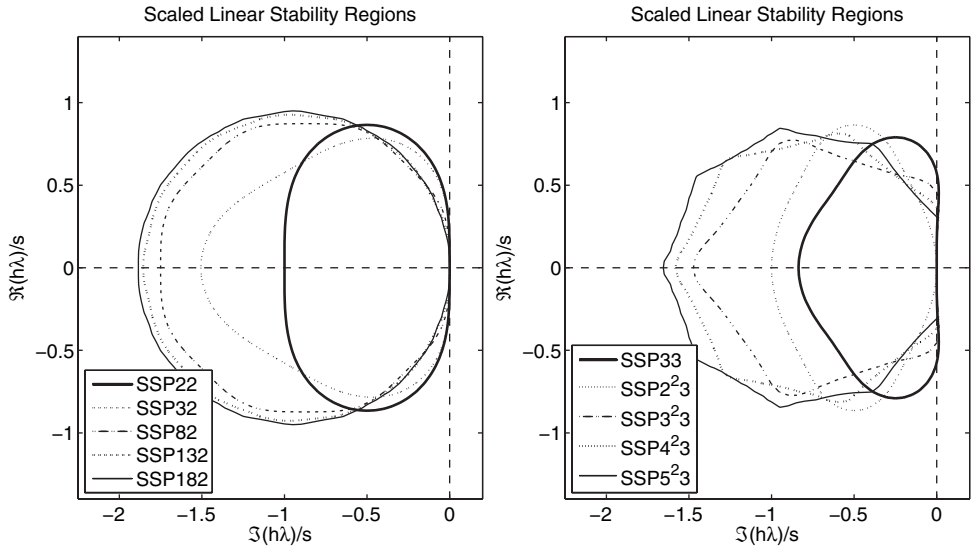
$$\phi(z) = \left(\frac{n}{2n-1}\left(1 + \frac{z}{n^2-n}\right)^{(n-1)^2} + \frac{n-1}{2n-1}\left(1 + \frac{z}{n^2-n}\right)^{n^2}\right), \tag{4.2}$$

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

In Figure 4.1, 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.

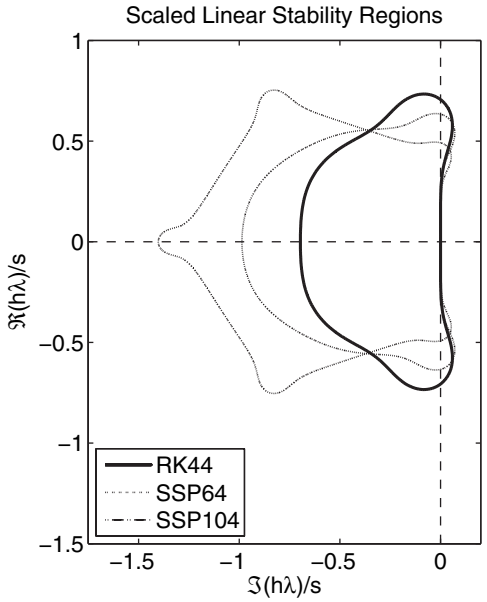
It is interesting to note that for large s the stability functions of the optimal second order methods approach that corresponding to s applications of the forward Euler method. The scaled absolute stability regions of the second order methods therefore tend to that of the forward Euler method.

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.



(a) Second order family of optimal methods; SSPs2 denotes the optimal second order method with s stages

(b) Third order family of optimal methods; SSP$n^2$3 denotes the optimal third order method with n^2 stages



(c) Fourth order methods of optimal methods; SSPs4 denotes the optimal fourth order method with s stages; RK44 denotes the classical fourth order Runge–Kutta method

FIG. 4.1. *Scaled stability regions of some members of the families of optimal second, third, and fourth order methods.*

Consider a Shu–Osher implementation of a Runge–Kutta method including roundoff errors r_i at each stage:

$$\begin{aligned} \tilde{y}_0 &= u^n + r_0, \\ \tilde{y}_i &= \sum_{j=0}^{i-1} (\alpha_{i,j}\tilde{y}_k + \Delta t \beta_{i,j} L(\tilde{y}_k)) + r_i, \quad \alpha_{i,j} \geq 0, \quad i = 1, \ldots, s, \\ \tilde{u}^{n+1} &= \tilde{y}_s. \end{aligned} \tag{4.3}$$

By subtracting the exact method (3.3) from the perturbed method (4.3), one finds that

$$\tilde{u}^{n+1} - u^{n+1} = \phi(\Delta t L) r_0 + \sum_{j=1}^{s} \theta_j(\Delta t L) r_j. \tag{4.4}$$

Here ϕ is again the absolute stability function; the functions θ_j are referred to as *internal stability polynomials* [31]. Assuming the r_j's have magnitude on the order of roundoff ($\epsilon_{machine}$), the method will be internally stable as long as $||\phi(\Delta t L)|| << 1/\epsilon_{machine}$ 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.

4.1.1. Second order methods. Straightforward calculation reveals that, for the optimal family of second order methods (3.8), as implemented above,

$$\theta_j(z) = \left(\frac{s-1}{s} + \frac{z}{s}\right)\left(1 + \frac{z}{s-1}\right)^{s-1-j} \tag{4.5}$$

for $1 \leq j \leq s-1$, while $\theta_s(z) = 1$. 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.

4.1.2. Third order methods. Similarly, for the optimal family of third order methods (3.10), as implemented above, the highest degree internal stability polynomials are given by

$$\theta_j(z) = \frac{1}{2n-1}\left[n\left(1 + \frac{z}{n^2-n}\right)^{(n-1)^2} + (n-1)\left(1 + \frac{z}{n^2-n}\right)^{n^2-j}\right] \tag{4.6}$$

for $1 \leq j \leq \frac{(n-1)(n-2)}{2}$. 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.

4.2. 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

TABLE 4.1
Error constants of new optimal SSP Runge–Kutta methods compared to previously known methods. RK(4,4) is the classical fourth order Runge–Kutta method.

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)	$\frac{1}{18}\cdot\frac{24}{2880}$	$\frac{17}{101}\cdot\frac{101}{2880}$

F and its derivatives, the leading truncation error can be bounded by a constant proportional to ([1, 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 $\gamma(t)$ and $\sigma(t)$ are the density and the symmetry of the tree t, respectively. The reader is referred to [1] 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 and C_L for our optimal methods. The resulting error constants are given in Table 4.1; 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 time step, 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. In general, if we wish to compare two methods of order p, let η_i, s_i, and C_i denote the relative error, number of stages, and error constant, respectively, of method i. Then

$$\frac{\eta_2}{\eta_1} \approx \left(\frac{s_2}{s_1}\right)^p \frac{C_1}{C_2}.$$

5. Numerical tests. In this section we demonstrate the effect of the SSP properties of our optimal methods by numerical examples.

5.1. Constant coefficient advection. We consider the linear system of ODEs

$$\boldsymbol{u}' = \mathbf{L}\boldsymbol{u}, \tag{5.1}$$

with $\boldsymbol{u} \in R^N$, $\mathbf{L} \in R^{N \times N}$, and

$$\mathbf{L} = \frac{1}{N} \begin{pmatrix} -1 & & & \\ 1 & -1 & & \\ & \ddots & \ddots & \\ & & 1 & -1 \end{pmatrix} \tag{5.2}$$

arising from an upwind-differencing approximation of the PDE

$$u_t + u_x = 0 \tag{5.3}$$

on the interval $x = [0, 1]$ with boundary condition $u(0, t) = 0$. We take $N = 20$. For this linear autonomous system, any Runge–Kutta method reduces to the simple iteration (2.4). Clearly monotonicity will be preserved for arbitrary initial conditions iff

$$||\phi(\Delta t \mathbf{L})|| \leq 1 \tag{5.4}$$

(cf. Remark 1). For this problem we have monotonicity for the forward Euler method under the maximal time step $\Delta t_{\text{FE}} \leq \Delta x = \frac{1}{N}$. In order to compare different integration methods, we compute for each method the maximum value c_0 such that (5.4) holds with

$$\Delta t = c_0 \Delta t_{\text{FE}}. \tag{5.5}$$

It can be shown analytically (cf. Remark 3 and [28, section 4.2]) that c_0 is exactly equal to the linear SSP coefficient c^{lin}. In Table 5.1, we list values of c_0, c^{lin}, and $c^{\text{lin}}_{\text{eff}} = c^{\text{lin}}/s$ for various methods. Here we have included the non-SSP methods of Wang and Spiteri [32] (note that they *are* SSP for linear autonomous problems). In

TABLE 5.1
*Monotone time-step coefficients and effective time-step coefficients for some optimal fourth order methods. RK*44 *is the classical fourth order method. NSSP (non-SSP) methods are from* [32].

Method	$c_0 = c^{\text{lin}}$	$c^{\text{lin}}_{\text{eff}}$
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

every case the theory and experiment are in perfect agreement. A clear advantage in efficiency is conferred by the SSP methods with many stages.

5.2. Variable coefficient advection. Previous work on SSP methods has emphasized that the SSP property is most critical when solving semidiscretizations 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 [19].

We solve the IBVP

$$u_t + (a(x,t)u)_x = 0, \tag{5.6a}$$

$$u(0,t) = 0, \qquad u(x,0) = g(x), \tag{5.6b}$$

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

$$a(x,t) = \cos^2(20x + 45t). \tag{5.7}$$

We semidiscretize by 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 (5.6) 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 time step such that these properties hold discretely to within roundoff error ($\approx 10^{-15}$). For the forward Euler method, this maximum time step is found to be $\Delta t = 1.02\Delta x$.

Because this semidiscretization is linear, any Runge–Kutta method applied to it reduces to the iteration

$$\boldsymbol{u}^{n+1} = \mathbf{M}_{\Delta t}(t)\boldsymbol{u}^n, \tag{5.8}$$

where $\mathbf{M}_{\Delta t}(t)$ is the matrix-valued K-function of the method [19]. Thus monotonicity and positivity are equivalent to

$$||\mathbf{M}_{\Delta t}||_1 \leq 1 \quad \text{and} \quad \mathbf{M}_{\Delta t} \geq 0, \tag{5.9}$$

respectively (the second inequality is componentwise).

In Figures 5.1–5.2, we plot the theoretical monotone and positive scaled time step $\Delta t/(s\Delta x)$ (i.e., the effective SSP coefficient) versus the observed maximum scaled monotone and positive time step for the second and third order families, respectively, of optimal methods. In all cases, the theory is borne out by these results; furthermore, the theoretical time-step limit seems to be quite sharp for this problem. For comparison, we also plot the observed maximum monotone and positive time step for the most commonly used second and third order SSP methods.

In Table 5.2 we list, for various methods, the maximum observed monotone and positive time step for this problem, along with effective SSP coefficients for linear and

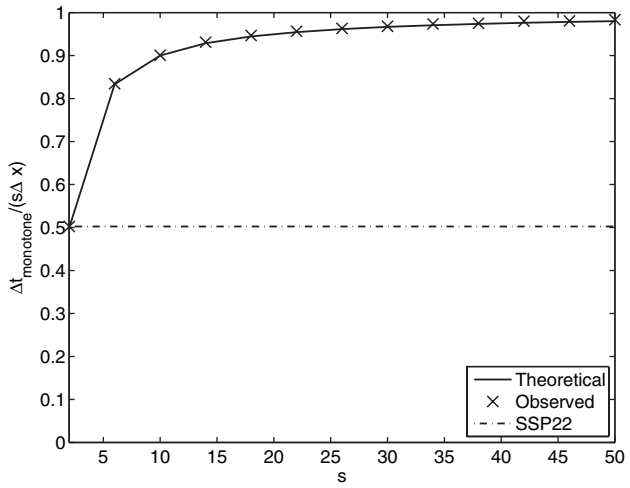


FIG. 5.1. *Theoretical and actual monotone effective time steps for the family of optimal second order methods. The horizontal line shows the actual monotone effective time step of the popular SSP*22 *method for comparison.*

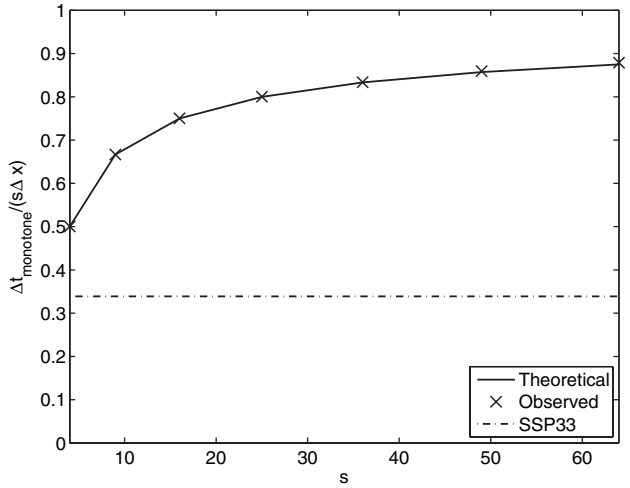


FIG. 5.2. *Theoretical and actual monotone effective time steps for the family of optimal third order methods. The horizontal line shows the actual monotone effective time step of the popular SSP*33 *method for comparison.*

nonlinear problems. Again we see fairly good agreement with theory and a clear advantage conferred by the SSP methods with many stages.

Remark 7. As expected, nearly all of the non-SSP methods fail to produce monotone results even for very small relative time steps (< 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 time-step restrictions are not always sharp for a particular choice of ODEs and time integrator (cf. Remarks 1 and 4).

TABLE 5.2
Theoretical and observed monotone effective time steps for variable coefficient advection. RK44 is the classical fourth order method.

Method	c_{eff}	Monotone effective time step
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

6. Conclusions and future work. We have presented an efficient algorithm for computing optimal strong stability-preserving Runge–Kutta methods for application to linear autonomous ODEs. This enables optimization of such methods with more stages and higher order accuracy than previously possible.

We have presented second, third, and fourth order explicit SSP Runge–Kutta methods for nonlinear or nonautonomous ODEs that achieve theoretically optimal bounds in terms of both the stable time step and memory requirements. These are more efficient than all existing explicit methods (including existing full-storage methods and methods using downwinding). Apparently the only remaining open question in this area is the existence of even more efficient fourth order methods (with even more stages).

It will be important to investigate the practical efficiency of these methods versus traditional methods for particular PDEs and semidiscretizations of interest. Some previous studies using SSP Runge–Kutta methods with a few extra stages have demonstrated their usefulness [17, 20].

The low-storage methods presented here do not fit into either of the established families of low-storage Runge–Kutta methods; consideration of these new low-storage methods leads naturally to a more general theory of low-storage implementations, not just for SSP methods but for general Runge–Kutta methods. This theory is the subject of current research.

Acknowledgments. The author thanks Sameer Agarwal for insightful discussions that led to the efficient algorithm for computing $\mathbf{R}_{s,p}(\phi)$. The author also thanks Sigal Gottlieb and Colin Macdonald for helpful comments on drafts of this paper. Finally, the author thanks the reviewers for providing helpful comments in a very timely manner.

REFERENCES

[1] J. C. BUTCHER, *Numerical Methods for Ordinary Differential Equations*, Wiley, New York, 2003.

[2] G. DAHLQUIST AND R. JELTSCH, *Generalized Disks of Contractivity for Explicit and Implicit Runge-Kutta Methods*, Technical report, Department of Numerical Analysis and Computer Science, Royal Institute of Technology, Stockholm, 1979.

[3] L. FERRACINA AND M. N. SPIJKER, *Stepsize restrictions for the total-variation-diminishing property in general Runge–Kutta methods*, SIAM J. Numer. Anal., 42 (2004), pp. 1073–1093.

[4] L. FERRACINA AND M. N. SPIJKER, *Computing Optimal Monotonicity-Preserving Runge–Kutta Methods*, Technical report MI2005-07, Mathematical Institute, Leiden University, 2005.

[5] L. FERRACINA AND M. N. SPIJKER, *An extension and analysis of the Shu-Osher representation of Runge-Kutta methods*, Math. Comp., 249 (2005), pp. 201–219.

[6] S. Gottlieb, *On high order strong stability preserving Runge-Kutta and multi step time discretizations*, J. Sci. Comput., 25 (2005), pp. 105–127.
[7] S. Gottlieb and L.-Ad J. Gottlieb, *Strong stability preserving properties of Runge-Kutta time discretization methods for linear constant coefficient operators*, J. Sci. Comput., 18 (2003), pp. 83–109.
[8] S. Gottlieb and C.-W. Shu, *Total variation diminishing Runge-Kutta schemes*, Math. Comp., 67 (1998), pp. 73–85.
[9] S. Gottlieb, C.-W. Shu, and E. Tadmor, *Strong stability-preserving high-order time discretization methods*, SIAM Rev., 43 (2001), pp. 89–112.
[10] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations* II: *Stiff and Differential-Algebraic Problems*, Springer Ser. Comput. Math. 14, Springer-Verlag, Berlin, 1991.
[11] I. Higueras, *On strong stability preserving time discretization methods*, J. Sci. Comput., 21 (2004), pp. 193–223.
[12] I. Higueras, *Monotonicity for Runge-Kutta methods: Inner product norms*, J. Sci. Comput., 24 (2005), pp. 97–117.
[13] I. Higueras, *Representations of Runge–Kutta methods and strong stability preserving methods*, SIAM J. Numer. Anal., 43 (2005), pp. 924–948.
[14] W. Hundsdorfer, S. J. Ruuth, and R. J. Spiteri, *Monotonicity-preserving linear multistep methods*, SIAM J. Numer. Anal., 41 (2003), pp. 605–623.
[15] C. A. Kennedy, M. H. Carpenter, and R. M. Lewis, *Low-storage, explicit Runge-Kutta schemes for the compressible Navier-Stokes equations*, Appl. Numer. Math., 35 (2000), pp. 177–219.
[16] D. I. Ketcheson, C. B. Macdonald, and S. Gottlieb, *Optimal implicit strong stability preserving Runge-Kutta methods*, submitted.
[17] D. I. Ketcheson and A. C. Robinson, *On the practical importance of the SSP property for Runge-Kutta time integrators for some common Godunov-type schemes*, Internat. J. Numer. Methods Fluids, 48 (2005), pp. 271–303.
[18] J. F. B. M. Kraaijevanger, *Absolute monotonicity of polynomials occurring in the numerical solution of initial value problems*, Numer. Math., 48 (1986), pp. 303–322.
[19] J. F. B. M. Kraaijevanger, *Contractivity of Runge-Kutta methods*, BIT, 31 (1991), pp. 482–528.
[20] E. J. Kubatko, J. J. Westerink, and C. Dawson, *Semi discrete discontinuous Galerkin methods and stage-exceeding-order, strong-stability-preserving Runge-Kutta time discretizations*, J. Comput. Phys., 222 (2007), pp. 832–848.
[21] C. B. Macdonald, *Constructing High-Order Runge-Kutta Methods with Embedded Strong-Stability-Preserving Pairs*, Master's thesis, Simon Fraser University, Burnaby, BC, 2003.
[22] S. Ruuth, *Global optimization of explicit strong-stability-preserving Runge-Kutta methods*, Math. Comp., 75 (2006), pp. 183–207.
[23] S. J. Ruuth and W. Hundsdorfer, *High-order linear multistep methods with general monotonicity and boundedness properties*, J. Comput. Phys., 209 (2005), pp. 226–248.
[24] S. J. Ruuth and R. J. Spiteri, *Two barriers on strong-stability-preserving time discretization methods*, J. Sci. Comput., 17 (2002), pp. 211–220.
[25] S. J. Ruuth and R. J. Spiteri, *High-order strong-stability-preserving Runge–Kutta methods with downwind-biased spatial discretizations*, SIAM J. Numer. Anal., 42 (2004), pp. 974–996.
[26] C.-W. Shu, *Total-variation-diminishing time discretizations*, SIAM J. Sci. Comput., 9 (1988), pp. 1073–1084.
[27] C.-W. Shu and S. Osher, *Efficient implementation of essentially non-oscillatory shock-capturing schemes*, J. Comput. Phys., 77 (1988), pp. 439–471.
[28] M. N. Spijker, *Contractivity in the numerical solution of initial value problems*, Numer. Math., 42 (1983), pp. 271–290.
[29] R. J. Spiteri and S. J. Ruuth, *A new class of optimal high-order strong-stability-preserving time discretization methods*, SIAM J. Numer. Anal., 40 (2002), pp. 469–491.
[30] R. J. Spiteri and S. J. Ruuth, *Nonlinear evolution using optimal fourth-order strong-stability-preserving Runge-Kutta methods*, Math. Comput. Simulation, 62 (2003), pp. 125–135.
[31] J. G. Verwer, *Explicit Runge-Kutta methods for parabolic partial differential equations*, Appl. Numer. Math., 22 (1996), pp. 359–379.
[32] R. Wang and R. J. Spiteri, *Linear instability of the fifth-order WENO method*, SIAM J. Numer. Anal., 45 (2007), pp. 1871–1901.
[33] J. H. Williamson, *Low-storage Runge-Kutta schemes*, J. Comput. Phys., 35 (1980), pp. 48–56.