Optimal Strong-Stability-Preserving Time-Stepping Schemes with Fast Downwind Spatial Discretizations

Sigal Gottlieb* and Steven J. Ruuth†

February 28, 2005

Abstract

In the field of strong-stability-preserving time discretizations, a number of researchers have considered using both upwind and downwind approximations for the same derivative, in order to guarantee that some strong stability condition will be preserved. The cost of computing both the upwind and downwind operator has always been assumed to be double that of computing only one of the two. However, in this paper we show that for the weighted essentially non-oscillatory method it is often possible to compute both these operators at a cost that is far below twice the cost of computing only one. This gives rise to the need for optimal strong-stability-preserving time-stepping schemes which take into account the different possible cost increments. We construct explicit linear multistep schemes up to order six and explicit Runge-Kutta schemes up to order four which are optimal over a range of incremental costs.

Keywords: strong-stability-preserving, total-variation-diminishing, Runge-Kutta methods, linear multistep methods, time discretization.

1 Introduction

When solving a hyperbolic partial differential equation (PDE) of the form $u_t = -f(u)_x$, the spatial domain is discretized, and the method of lines approximation yields the system of ordinary differential equations $U_t = L(U)$ where the components of the vector $U(t)$ approximate the PDE solution at grid points or surrounding cells, $U_i(t) \approx u(x_i, t)$. The approximation $L(U)$ of the spatial derivative $-f(u)_x$ is obtained by a finite difference, finite element, or spectral method. This approximation is carefully chosen to satisfy special nonlinear stability properties, so that coupled with the forward Euler time discretization, the resulting sequence of approximations U_j^n satisfies the strong stability property:

$$||U^{n+1}|| \leq ||U^n + \Delta t L(U^n)||, \tag{1}$$

*Department of Mathematics, UMASS-Dartmouth North Dartmouth, MA (sg@cfm.brown.edu). The work of this author was supported by NSF grant DMS-0106743

†Department of Mathematics, Simon Fraser University, Burnaby, British Columbia, V5A 1S6 Canada (sruuth@sfu.ca). The work of this author was partially supported by a grant from NSERC Canada.

in a given norm $||\cdot||$, under a suitable time-step restriction $\Delta t \leq \Delta t_{FE}$. Strong-stability-preserving (SSP) time discretizations were developed to extend this strong stability property to higher order time discretizations. In the field of explicit SSP Runge-Kutta (SSPRK) time discretizations, a general s-stage, explicit Runge-Kutta method is written in the Shu-Osher form [10]:

$$\begin{aligned} U^{(0)} &= U^n \\ U^{(i)} &= \sum_{k=0}^{i-1} \alpha_{ik} U^{(k)} + \Delta t \beta_{ik} \left\{ \begin{array}{ll} L(U^{(k)}) & \text{if } \beta_{ik} \geq 0 \\ \tilde{L}(U^{(k)}) & \text{otherwise} \end{array} \right. , \qquad i = 1, 2, \ldots, s, \\ U^{n+1} &= U^{(s)}, \end{aligned} \tag{2}$$

where all the $\alpha_{ik} \geq 0$ and $\alpha_{ik} = 0$ only if $\beta_{ik} = 0$ [11]. $L(\cdot)$ approximates $-f(u)_x$ and satisfies the strong stability property described above, and $\tilde{L}(\cdot)$ approximates the same derivative, but satisfies the strong stability property for the backward-in-time Euler method:

$$||U^{n+1}|| \leq ||U^n - \Delta t \tilde{L}(U^n)||. \tag{3}$$

For hyperbolic conservation laws, $-\tilde{L}(\cdot)$ can be obtained by discretizing in space $u_t = f(u)_x$.

This form is a convex combination of explicit Euler steps, which allows us to prove that whenever L and $\tilde{L}$ satisfy, for a given norm $||\cdot||$, the strong stability properties (1) and (3), the Runge-Kutta method above will also satisfy the strong stability property $||U^{n+1}|| \leq ||U^n||$, but under the stepsize restriction $\Delta t \leq C \Delta t_{FE}$, with $C = \min_{i,k} \frac{\alpha_{ik}}{|\beta_{ik}|}$.

Similarly, an explicit linear multistep method is written as a convex combination of explicit Euler steps:

$$U^{n+1} = \sum_{i=1}^{m} \left(a_i U^{n+1-i} + \Delta t b_i \left\{ \begin{array}{ll} L(U^{n+1-i}) & \text{if } b_i \geq 0 \\ \tilde{L}(U^{n+1-i}) & \text{otherwise} \end{array} \right. \right)$$

and can be shown, therefore, to be SSP under the stepsize restriction $\Delta t \leq C \Delta t_{FE}$, with $C = \min_i \frac{a_i}{|b_i|}$.

Clearly, these stepsize restrictions can be prohibitive if C is very small. The efficiency of the Runge-Kutta or multistep method depends on the size of C – a larger C allows for larger Δt, and thus fewer steps are needed to reach the desired time. However, the computational efficiency is not determined by the number of time-steps alone; the amount of computational effort at each time-step must be taken into account as well. In Section 3, we compare the computational efficiencies of linear multistep methods. Section 4 treats the corresponding Runge-Kutta methods. In order to make a fair comparison of the relative efficiencies of these methods and to derive optimal schemes we define:

Definition 1 *The* effective CFL coefficient C_{eff} *of an SSP time-stepping method is* $\frac{C}{w}$ *where* C *is the CFL coefficient of the method, and* w *is the amount of computational work for one step of the method measured relative to an evaluation of* $L(\cdot)$.

Up to now, the computation of L and $\tilde{L}$ has been considered to be equivalent, and the computational cost of both L and $\tilde{L}$ has been estimated as double the cost of computing only L. This work estimate naturally leads to a significant reduction in the effective CFL

coefficient of a method which requires both $L(U^{(k)})$ and $\tilde{L}(U^{(k)})$. We remark that this assumption on computational cost has been central to recent studies for optimal methods involving both operators. See, e.g., [2, 9, 7].

In this paper, we demonstrate that the cost of calculating both L and $\tilde{L}$ is not necessarily double the cost of calculating only L. This is due to the fact that both L and $\tilde{L}$ approximate the same derivative, and there is a significant amount of symmetry which may be exploited to reduce the computational cost. In section 2 we demonstrate that the cost of computing both the upwind and downwind operator can be far less than twice the cost of computing the upwind operator alone. The example in this section suggests the development of new optimal methods, which take into account that the computation of L and $\tilde{L}$ can be performed efficiently. In sections 3 and 4 we present new and optimal linear multistep and Runge-Kutta methods which take advantage of the small incremental cost of computing both operators.

2 Efficient Computation of the Upwind and Downwind Operators using WENO

In this section, we demonstrate that once $L(U)$ is computed, the additional cost of computing $\tilde{L}(U)$ is sometimes far less than the base cost of computing $L(U)$. This is hardly a surprising result: if we consider a flow for which the spatial discretization

$$L(U_j) = \frac{-f(U_{j+1}) + f(U_j)}{\Delta x},$$

corresponds to upwinding, then the corresponding downwind discretization is

$$\tilde{L}(U_j) = \frac{-f(U_j) + f(U_{j-1})}{\Delta x} = L(U_{j-1}),$$

which can be obtained at no extra cost. However, we are interested in more sophisticated and accurate methods, and would like to demonstrate that the cost increment can be made small in a more general context. Therefore, we consider as an example the popular WENO method.

For our spatial discretization, we consider a fifth-order weighted essentially non oscillatory (WENO) approximation because these WENO methods are popular and are frequently used in combination with SSP time-stepping schemes. We emphasize, however, that the multistep and Runge-Kutta schemes derived in sections 3 and 4 are also suitable for other spatially discretized systems. If we compare the relative cost of computing $L(U)$ and $\tilde{L}(U)$ to the cost of computing $L(U)$ alone, we say that the cost of evaluating both operators is $1+\delta$ function evaluations, where $0 \leq \delta \leq 1$. We refer to δ as the *incremental cost*.

In the following, we define the fifth-order WENO algorithm for efficiently computing both L and $\tilde{L}$. This method uses three 3-point stencils to evaluate the numerical flux at each spatial point, it then uses the difference of the numerical fluxes at two neighboring points to evaluate $L(U)$. For simplicity, we shall assume that the spatial domain is $[-1, 1]$.

For the WENO approximation we use an equidistant grid $x_j = -1 + 2\frac{j}{N},\ j = 0, ..., N$. The flux is split into two parts $f = f^+ + f^-$, where $\frac{d}{dU} f^+(U) \geq 0$ and $\frac{d}{dU} f^-(U) \leq 0$. With these preliminaries completed, we proceed to the algorithm

Algorithm 1 WENO to compute $L(\cdot)$ and $\tilde{L}(\cdot)$:

1. *Define the stencils for all grid points $j = -1, ..., N$*

$$\begin{aligned} S_1^{\pm}(j) &= 2f_{j-2}^{\pm} - 7f_{j-1}^{\pm} + 11f_j^{\pm} & S_2^{\pm}(j) &= -f_{j-1}^{\pm} + 5f_j^{\pm} + 2f_{j+1}^{\pm} \\ S_3^{\pm}(j) &= 2f_j^{\pm} + 5f_{j+1}^{\pm} - f_{j+2}^{\pm} & S_4^{\pm}(j) &= 11f_{j+1}^{\pm} - 7f_{j+2}^{\pm} + 2f_{j+3}^{\pm} \end{aligned}$$

2. *Set up the information needed for the smoothness measurements for $k = -2, ..., N+2$:*

$$\begin{aligned} HS_1^{\pm}(k) &= \left(f_{k+1}^{\pm} - 2f_k^{\pm} + f_{k-1}^{\pm}\right)^2 & HS_2^{\pm}(k) &= \left(f_{k+1}^{\pm} - f_{k-1}^{\pm}\right)^2 \\ HS_3^{\pm}(k) &= \left(f_{k+1}^{\pm} - 4f_k^{\pm} + 3f_{k-1}^{\pm}\right)^2 & HS_4^{\pm}(k) &= \left(3f_{k+1}^{\pm} - 4f_k^{\pm} + f_{k-1}^{\pm}\right)^2 \end{aligned}$$

3. *Calculate the smoothness measurements following Jiang and Shu [4].*

	For $IS_i^+(j)$, $j = -1, ..., N+1$		*For $IS_i^-(j)$, $j = -2, ..., N$*
$IS_1 =$	$c_1 HS_1^+(j-1) + c_2 HS_4^+(j-1)$	$IS_1 =$	$c_1 HS_1^-(j+2) + c_2 HS_3^-(j+2)$
$IS_2 =$	$c_1 HS_1^+(j) + c_2 HS_2^+(j)$	$IS_2 =$	$c_1 HS_1^-(j+1) + c_2 HS_2^-(j+1)$
$IS_3 =$	$c_1 HS_1^+(j+1) + c_2 HS_3^+(j+1)$	$IS_3 =$	$c_1 HS_1^-(j) + c_2 HS_4^-(j)$
$IS_i^+(j) =$	$(\epsilon + IS_i)^2 \quad i = 1, 2, 3$	$IS_i^-(j) =$	$(\epsilon + IS_i)^2 \quad i = 1, 2, 3$

Where $c_1 = 13/12$, $c_2 = 1/4$ and the parameter ϵ is a small number chosen to prevent the denominator of $\hat{f}_j^{\pm}$ below from being zero. We typically use $\epsilon = 10^{-13}$.

4. *Construct $f_j^{\pm}$ for $j = -1, ..., N$. The positive and negative fluxes are computed by calculating the weights based on the smoothness measurements as in [4] and finally taking a weighted average of the stencils. These weights are designed so that in the smooth regions, the approximation will be fifth order, while near the discontinuity, the weights approach the ENO weights and thus the approximation will be third order.*

	For f_j^+		*For f_j^-*
$tt_i =$	$IS_i^+(j) \;\; i = 1, 2, 3$	$tt_i =$	$IS_i^-(j) \;\; i = 1, 2, 3$
$w_1 =$	$tt_2 tt_3$	$w_1 =$	$tt_2 tt_3$
$w_2 =$	$6tt_1 tt_3$	$w_2 =$	$6tt_1 tt_3$
$w_3 =$	$3tt_1 tt_2$	$w_3 =$	$3tt_1 tt_2$
$t_0 =$	$6(w_1 + w_2 + w_3)$	$t_0 =$	$6(w_1 + w_2 + w_3)$
$\hat{f}_j^+ =$	$\frac{1}{t_0}(w_1 S_1^+(j) + w_2 S_2^+(j) + w_3 S_3^+(j))$	$\hat{f}_j^- =$	$\frac{1}{t_0}(w_1 S_2^-(j) + w_2 S_3^-(j) + w_3 S_4^-(j))$

5. *Put together the numerical flux $\hat{f}_j = \hat{f}_j^+ + \hat{f}_j^-$, $j = -1, ..., N$, and construct*

$$L(U_j) = -\frac{1}{\Delta x}\left(\hat{f}_j - \hat{f}_{j-1}\right),$$

for $j = 0, ..., N$

6. *To construct the downwind flux, consider the downwind problem where $g(U) = -f(U)$, so that the flux splitting implies that $g^+ = -f^-$ and $g^- = -f^+$. Thus, many of the necessary quantities were computed already. We proceed for $j = -1, ..., N$:*

For g_j^-			*For* g_j^+		
tt_i	=	$IS_{4-i}^+(j+1)\ \ i=1,2,3$	tt_i	=	$IS_{4-i}^-(j-1)\ \ i=1,2,3$
w_1	=	$tt_2 tt_3$	w_1	=	$tt_2 tt_3$
w_2	=	$6tt_1 tt_3$	w_2	=	$6tt_1 tt_3$
w_3	=	$3tt_1 tt_2$	w_3	=	$3tt_1 tt_2$
t_0	=	$6(w_1+w_2+w_3)$	t_0	=	$6(w_1+w_2+w_3)$
$\hat{g}_j^-$	=	$-\frac{1}{t_0}(w_3 S_2^+(j)+w_2 S_3^+(j)+w_1 S_4^+(j))$	$\hat{g}_j^+$	=	$-\frac{1}{t_0}(w_1 S_1^-(j)+w_2 S_2^-(j)+w_3 S_3^-(j))$

7. *Put together the numerical flux* $\hat{g}_j = \hat{g}_j^+ + \hat{g}_j^-,\ j=-1,...,N,$ *and construct*

$$\tilde{L}(U_j) = -\frac{1}{\Delta x}\left(\hat{g}_j - \hat{g}_{j-1}\right),$$

for $j=0,...,N.$

This algorithm takes advantage of the fact that the stencils and the smoothness measurements need only be calculated once for both L and $\tilde{L}$. The WENO method for computing the upwind operator may also be programmed without predetermining the stencils, and this was often more efficient in our numerical tests. The relative efficiencies, in terms of CPU time, of these two programming styles (predetermining the stencils or not) depends on the compiler and machine used. Often, the CPU time involved in pulling the predefined quantities in and out of cache outweighs the benefit of fewer computations. To control for this, a comparison with both the modified algorithm with the $\tilde{L}(\cdot)$ portion removed, and the traditional algorithm is performed. The incremental cost δ is calculated for each of the two cases, and the largest incremental cost is the one used. This computed incremental cost is plotted against the number of grid points for a number of platforms in Figure 1. It is clear that this estimate varies widely depending on the compiler and machine used. Although the incremental cost can be roughly assessed analytically by counting the number of floating point operations, this is not very revealing due to the difficulties in accurately determining the varying cost of different arithmetic operations, as well as accounting for the cost of other operations such as memory access and variable assignment. Thus, we must rely on numerical computations to give us an accurate idea of the cost measurement. However, since computational efficiency is machine and compiler dependent, robust determination of the value of δ may be difficult.

3 Optimal Explicit Multistep Schemes

Our discussion on appropriate time-stepping methods starts with the class of explicit SSP linear multistep schemes. We consider optimal methods for arbitrary starting values (cf. [11, 2]); an analysis that includes starting procedures is also possible using the ideas in [3]. Throughout this section, we allow for the possibility of upwind and downwind operations, and assume that the cost of evaluating both operators is $1+\delta$ general function evaluations where $0 \le \delta \le 1$. Thus, each step of the algorithm will have a cost of $1+\delta$ general function evaluations if any downwind operators are present. This contrasts with the recent efforts in [2, 8] where it is assumed that $\delta = 1$.

CFL coefficients for optimal nonnegative coefficient schemes are provided in [5]; see also [2, 8]. For convenience, these results are provided in Table 1 below. In [2] it was shown that there are no m-step m-th order SSP multistep methods with all nonnegative β's, and thus downwind operators were considered. In [11, 2, 8] optimal multistep SSP methods, including those with downwind operators, were studied. However, up to this point it was assumed that $\delta = 1$; the purpose of this work is to extend these earlier results and consider general $\delta < 1$.

For the case of unrestricted coefficients, guaranteed optimal k-step, order-p schemes are efficiently determined using the deterministic global branch-and-bound software, BARON [13]. To guarantee optimality in BARON, bounds on all the variables are normally needed. Fortunately this is straightforward because all the a_i are bounded by 1 (because $\sum_i a_i = 1$) and all the b_i are bounded by the inverse of the CFL coefficient. Finding globally optimal linear multistep schemes and guaranteeing their optimality is surprisingly efficient. For example, BARON 7.2 finds the optimal seven-step, fifth-order scheme and guarantees its optimality in less than two seconds on a 1.2 GHz Athlon machine. See [6, 7, 8] for further details on the global optimization of SSP time-stepping schemes.

Table 1 provides the CFL coefficients for the optimal schemes, as found by BARON. Some of these results ($k \le 6, 2 \le p \le 6$ and $k \ge 2, p = 2$) were first determined in earlier studies [11, 2, 8] while the remaining results are new. Downwinding often leads to improved effective CFL coefficients. For example, even if $\delta = 1$, downwinding leads to smaller effective CFL coefficients when $(k, p) = (2, 2)$, $(3, 3)$, $(4, 4)$, $(5, 4)$, $(5, 5)$, $(6, 5)$, $(7, 5)$ or if $p = 6$ and $k = 6, ..., 10$. We provide two of these schemes, SSPMS(7,5) and SSPMS(10,6), in Tables 3 and 4 (see [8] for $k \le 6$). On the other hand, downwind operators do not even appear in the optimal schemes when $p = 3$ and $k \ge 6$, so downwinding is not useful there. In the remaining cases, the size of the increment, δ, will determine which scheme is more efficient. See Table 2 for details.

4 Optimal Explicit Runge-Kutta Schemes

We now focus on explicit SSP Runge-Kutta schemes. We allow for the possibility of upwind ($L(U^{(k)})$) and downwind ($\tilde{L}(U^{(k)})$) operations at the same level k, and assume that the cost of evaluating both operators is $1+\delta$ general function evaluations where $0 \le \delta \le 1$. Once again, there has been much work on the derivation of optimal schemes which include downwind operators, e.g. [2, 9, 7]. These results all assume $\delta = 1$. In this section, we proceed to examine the case where $\delta < 1$.

4.1 Second-Order Schemes

We begin by considering two-stage, second-order schemes. There are two cases: either β_{10} and β_{20} have the same sign or they do not. As proven in [1, 9], the former case leads to the classical modified Euler method, SSPRK(2,2), which has an effective CFL coefficient of 0.5. In the latter case, BARON guarantees that the optimal scheme is the SSPRK$_*(2, 2)$ scheme provided in Table 5. This scheme has a CFL coefficient of 1.215 and an effective CFL coefficient of $1.215/(2 + \delta)$. Thus for $\delta \le 0.43$ the optimal scheme is SSPRK$_*(2, 2)$, otherwise it is SSPRK(2,2).

It is known that the SSPRK(3,2) scheme [12] is an optimal second-order method with three general function evaluations [9]. This scheme has a CFL coefficient of 2. If both $L(\cdot)$ and $\tilde{L}(\cdot)$ arise at (precisely) one level then BARON guarantees that the largest CFL coefficient ($C = 2.19$) is attained by the SSPRK$_*(3,2)$ scheme given in Table 6. Larger CFL coefficients cannot arise using only three stages. This result is obtained by applying BARON to the model without any sign restrictions on the β's. By comparing the effective CFL coefficients of the two schemes it is easily seen that for $\delta < 0.28$ the optimal scheme is SSPRK$_*(3,2)$, otherwise it is SSPRK(3,2).

4.2 Third-Order Schemes

For third-order and three general function evaluations, it is known that the Shu-Osher third-order method, SSPRK(3,3), is optimal [1, 9]. If both $L(\cdot)$ and $\tilde{L}(\cdot)$ arise at one level then BARON guarantees that the largest CFL coefficient ($C = 1.30$) is obtained using the SSPRK$_*(3,3)$ scheme given in Table 7. If no sign restrictions are applied to the β's, then BARON guarantees that the largest CFL coefficient is attained by the SSPRK$_{**}(3,3)$ scheme. This scheme has a CFL coefficient of 1.44 and is also provided in Table 7. To obtain the optimal effective CFL coefficient, we choose SSPRK(3,3) for $\delta \geq 0.91$, SSPRK$_*(3,3)$ for $0.35 \leq \delta < 0.91$ and SSPRK$_{**}(3,3)$ for $\delta < 0.35$.

The optimal third-order method with four general function evaluations is the SSPRK(4,3) method [12]; see [9]. This nonnegative coefficient scheme has a CFL coefficient of 2. Downwinding yields little additional benefit: allowing both $L(\cdot)$ and $\tilde{L}(\cdot)$ at one level leads to a maximal CFL coefficient of 2.06 while leaving the sign of all the β's unrestricted leads to a maximal CFL coefficient of 2.12. Due to their limited usefulness we do not provide either of these schemes here.

4.3 Fourth-Order Schemes

It is impossible to construct a fourth-order SSPRK scheme with a positive CFL coefficient using only four general function evaluations [9]. Allowing both upwind and downwind operators at one level leads to the SSPRK$_*(4,4)$ scheme appearing in Table 8. This scheme has a CFL coefficient of 0.98. By removing the sign restrictions on the β's, we can obtain a guarantee (using BARON) that there are no four-stage methods with larger CFL coefficients.

The optimal scheme over the class of fourth-order methods with five general function evaluations is the SSPRK(5,4) scheme [12, 7]. This nonnegative coefficient scheme has a CFL coefficient of 1.51. If both $L(\cdot)$ and $\tilde{L}(\cdot)$ appear at one level then extensive numerical searches suggest that the SSPRK$_*(5,4)$ scheme given in Table 9 is optimal. This scheme has a CFL coefficient of 2.03 which corresponds to a larger effective CFL coefficient than SSPRK(5,4) for any $\delta \leq 1$. We do not report on any other five-stage schemes, since none were found with CFL coefficients exceeding 2.1.

5 Conclusions

In this work, we demonstrate that the incremental cost of computing both the upwind and downwind operators can be less than one, thus challenging the assumptions which underlie the currently known optimal methods. As an example, we show that the popular WENO method (fifth order in smooth regions, third near discontinuities) can be programmed in a way which reduces the incremental cost δ. We observe that, in practice, the value of the incremental cost varies according to the number of points, the compiler and the machine. This example leads to the re-examination of optimal time-stepping methods which include downwinding, and we obtain a variety of multistep and Runge-Kutta methods which are optimal for different ranges of δ. A study the numerical performance of these schemes would also be interesting, and is left to future work.

References

[1] S. Gottlieb and C.W. Shu, *Total variation diminishing Runge-Kutta schemes*, Math. Comp., **67**, No. 221, pp. 73-85, 1998.

[2] S. Gottlieb, C.-W. Shu & E. Tadmor, *Strong stability preserving high-order time discretization methods*. SIAM Review, **42**, pp. 89-112, 2001.

[3] W. Hundsdorfer, S.J. Ruuth and R.J. Spiteri,*Monotonicity-preserving linear multistep methods*. SIAM J. Numer. Anal., **41**, pp. 605-623, 2003.

[4] G.-S. Jiang and C.-W. Shu, *Efficient Implementation of Weighted ENO Schemes*, J. of Comp. Physics, **vol. 126**, 1996, pp.202-228.

[5] H.W.J. Lenferink, *Contractivity preserving explicit linear multistep methods*, Numer. Math. **55**, pp. 213-223, 1989.

[6] C. B. Macdonald, *High-order embedded Runge-Kutta pairs for the time evolution of hyperbolic conservation laws*, Master's thesis, Simon Fraser University, Burnaby, BC, Canada, 2003.

[7] S.J. Ruuth, *Global optimization of explicit strong-stability-preserving Runge-Kutta schemes*, Math. Comp., to appear.

[8] S.J. Ruuth & W. Hundsdorfer, *High-order linear multistep methods with general monotonicity and boundedness properties*, J. Comput. Phys., to appear.

[9] S. J. Ruuth & R. J. Spiteri, *High-order strong-stability-preserving Runge-Kutta methods with downwind-biased spatial discretizations*, SIAM J. Numer. Anal., **42**, No. 3, pp. 974-996, 2004.

[10] C.W. Shu & S. Osher, *Efficient implementation of essentially nonoscillatory shock-capturing schemes*, J. Comput. Phys., **77**, No. 2, pp. 439-471, 1988.

[11] C.W. Shu, *Total-variation-diminishing time discretizations*, SIAM J. Sci. Statist. Comput., **9**, No. 6, pp. 1073-1084, 1988.

[12] R.J. Spiteri and S.J. Ruuth, *A new class of optimal high-order strong-stability-preserving time-stepping schemes*. SIAM J. Numer. Anal., **40**, pp. 469–491, (2002).

[13] M. Tawarmalani & N.V. Sahinidis, *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*. Nonconvex Optimization and Its Applications 65, Kluwer, 2002.

Tables

Table 1: CFL coefficients for optimal k-step, pth-order linear multistep schemes. If any downwind operators are present, the effective CFL coefficient will equal the CFL coefficient divided by $(1+\delta)$; otherwise the effective CFL coefficient will equal the CFL coefficient.

	nonnegative coefficient schemes					unrestricted coefficient schemes				
	$p=2$	$p=3$	$p=4$	$p=5$	$p=6$	$p=2$	$p=3$	$p=4$	$p=5$	$p=6$
$k=2$						1/2				
$k=3$	1/2					2/3	0.2865			
$k=4$	2/3	0.3333				3/4	0.4146	0.1587		
$k=5$	3/4	0.5000	0.0212			4/5	0.5172	0.2371	0.0865	
$k=6$	4/5	0.5828	0.1648			5/6	0.5828	0.2832	0.1313	0.0462
$k=7$	5/6	0.5828	0.2815	0.0381		6/7	0.5828	0.3595	0.1868	0.0809
$k=8$	6/7	0.5828	0.3586	0.1451		7/8	0.5828	0.3945	0.2232	0.1073
$k=9$	7/8	0.5828	0.3925	0.2277		8/9	0.5828	0.4243	0.2610	0.1419
$k=10$	8/9	0.5828	0.4208	0.2822	0.0520	9/10	0.5828	0.4466	0.2989	0.1749

Table 2: Selection of the optimal k-step, order-p linear multistep scheme. Table entries of "1" correspond to cases where the unrestricted coefficient scheme is more efficient (in terms of effective CFL coefficient). Entries of "0" correspond to cases where downwinding does not appear in the optimal schemes. In the remaining entries, the nonnegative coefficient scheme will be more efficient provided the increment δ exceeds the table entry.

	$p=2$	$p=3$	$p=4$	$p=5$	$p=6$
$k=2$	1				
$k=3$	0.3333	1			
$k=4$	0.1250	0.2439	1		
$k=5$	0.0667	0.0344	1	1	
$k=6$	0.0417	0	0.7184	1	1
$k=7$	0.0286	0	0.2771	1	1
$k=8$	0.0208	0	0.1001	0.5382	1
$k=9$	0.0159	0	0.0810	0.1462	1
$k=10$	0.0125	0	0.0613	0.0592	1

Table 3: The nonzero coefficients of the optimal 7-step, 5th-order linear multistep scheme.

SSPMS(7,5)	a_i	b_i	CFL Coefficient
$i = 1$	0.437478073273716	2.341383323503706	0.1868460
$i = 2$	0.177079742280077	-0.947731054044159	
$i = 4$	0.266879475710902	1.428339365991395	
$i = 6$	0.079085404949912	-0.423265209377492	
$i = 7$	0.039477303785393	0.211282590801251	

Table 4: The nonzero coefficients of the optimal 10-step, 6th-order linear multistep scheme.

SSPMS(10,6)	a_i	b_i	CFL Coefficient
$i = 1$	0.421496355190108	2.409253340733589	0.1749490
$i = 2$	0.184871618144855	-1.056717473684455	
$i = 4$	0.261496145095487	1.494699665620621	
$i = 7$	0.030002986393737	-0.171495658990894	
$i = 9$	0.078557623043187	0.449031678275387	
$i = 10$	0.023575272132626	-0.134755146621380	

Tables (cont)

Table 5: The coefficients of the optimal $\text{SSPRK}_*(2,2)$ scheme.

Name	$\text{SSPRK}_*(2,2)$	
α_{ik}	1.000000000000000	
	0.261583187659478	0.738416812340522
β_{ik}	0.822875655532364	
	−0.215250437021539	0.607625218510713
CFL coefficient	1.2152504	

Table 6: The coefficients of the optimal $\mathrm{SSPRK}_*(3,2)$ scheme.

Name	$\mathrm{SSPRK}_*(3,2)$		
α_{ik}	1.000000000000000		
	0.000000000000000	1.000000000000000	
	0.203464834591289	0.000000000000000	0.796535165408711
β_{ik}	0.457427107756303		
	0.000000000000000	0.457427107756303	
	−0.093070330817223	0.000000000000000	0.364356776939073
CFL coefficient	2.1861407		

Table 7: The coefficients of the optimal $\text{SSPRK}_*(3,3)$ and $\text{SSPRK}_{**}(3,3)$ schemes

Name	$\text{SSPRK}_*(3,3)$		
α_{ik}	1.000000000000000		
	0.410802706918667	0.589197293081333	
	0.123062611901395	0.251481201947289	0.625456186151316
β_{ik}	0.767591879243998		
	−0.315328821802221	0.452263057441777	
	−0.041647109531262	0.000000000000000	0.480095089312672
CFL coefficient		1.3027756	

Name	$\text{SSPRK}_{**}(3,3)$		
α_{ik}	1.000000000000000		
	0.352901667695409	0.647098332304591	
	0.049992508960455	0.183215659743209	0.766791831296336
β_{ik}	0.695131544898322		
	−0.245313081462304	0.449818463436018	
	−0.034751369987025	−0.127358984606862	0.533021190304435
CFL coefficient		1.4385766	

Table 8: The coefficients of the optimal $\mathrm{SSPRK}_*(4,4)$ scheme.

Name	$\mathrm{SSPRK}_*(4,4)$			
α_{ik}	1.000000000000000			
	0.447703597093315	0.552296402906685		
	0.174381001639320	0.000000000000000	0.825618998360680	
	0.374455263824577	0.271670479800689	0.081190815217391	0.272683441157343
β_{ik}	0.545797148202810			
	−0.455917323951788	0.562429025981069		
	−0.177580256517037	0.000000000000000	0.840766093415820	
	0.107821590754283	0.276654641489540	0.000000000000000	0.161441275936663
CFL coefficient	0.9819842			

Table 9: The coefficients of the numerically optimal SSPRK$_*(5,4)$ scheme.

Name	SSPRK$_*(5,4)$				
α_{ik}	1.000000000000000				
	0.210186660827794	0.789813339172206			
	0.331062996240662	0.202036516631465	0.466900487127873		
	0.000000000000000	0.000000000000000	0.000000000000000	1.000000000000000	
	0.097315407775058	0.435703937692290	0.000000000000000	0.000000000000000	0.466980654532652
β_{ik}	0.416596471458169				
	−0.103478898431154	0.388840157514713			
	−0.162988621767813	0.000000000000000	0.229864007043460		
	0.000000000000000	0.000000000000000	0.000000000000000	0.492319055945867	
	−0.047910229684804	0.202097732052527	0.000000000000000	0.000000000000000	0.229903474984498
CFL coefficient	2.0312031				

Figure Captions

Figure 1: Incremental cost on various platforms. **A**: 1.5GHz Itanium II; compiled with `ifort -fast` under Linux 2.4.21. **B:** 1.5 GHz G4 Powerbook; compiled with `g77 -O5` under Darwin 7.4.0. **C:** 2.0 GHz AMD Opteron; compiled with `g77 -O5` under Linux 2.4.21. **D:** 3.06 GHz Pentium 4 Xeon; compiled with `g77 -O5` under Linux 2.4.21.

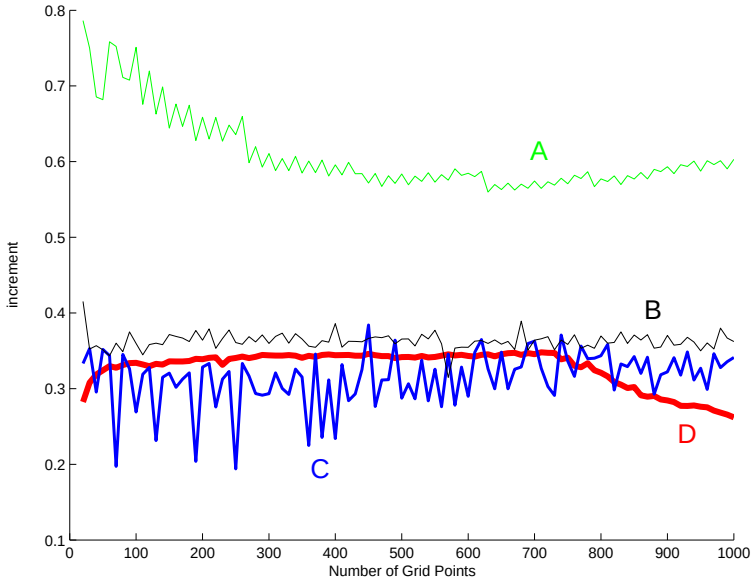
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
increment
A
B
C
D
0
100
200
300
400
500
600
700
800
900
1000
Number of Grid Points

Figure 1: