Electronic Transactions on Numerical Analysis.
Volume 39, pp. 298-312, 2012.
Copyright © 2012, Kent State University.
ISSN 1068-9613.

ETNA
Kent State University
http://etna.math.kent.edu

OPTIMIZING RUNGE-KUTTA SMOOTHERS FOR UNSTEADY FLOW PROBLEMS[*]

PHILIPP BIRKEN[†]

Abstract. We consider the solution of unsteady viscous flow problems by multigrid methods employing Runge-Kutta smoothers. Optimal coefficients for the smoothers are found by considering the unsteady linear advection equation and performing a Fourier analysis, respectively, looking at the spectral radius of the multigrid iteration matrix. The new schemes are compared to using a classic dual time stepping approach, where the scheme for the steady state equation is reused, meaning that the smoother coefficients are obtained by an optimization based on the steady state equations, showing significant improvements.

Key words. Multigrid, unsteady flows, finite volume methods, explicit Runge-Kutta methods, linear advection equation

AMS subject classifications. 35L04, 35Q35, 65F10, 65L06, 65M08

1. Introduction.

During the last decade, numerical methods for unsteady flows have garnered increasing attention. In a way, this can be attributed to a certain maturity reached by methods for steady flows. As was shown by Caughey and Jameson [4], the solution of steady Euler flows is possible in three to five multigrid steps. Thus, steady two dimensional flows around airfoils can be solved on a PC in a matter of seconds. The solution of the steady Reynolds-averaged Navier-Stokes equations (RANS equations) is more difficult and takes between 50 and 200 steps with a fast solver [12], which means that adequate methods for steady flows exist.

In this article, we will consider the unsteady case. Using dual time stepping, the above mentioned multigrid method can be used for unsteady flows at almost no additional implementation cost [6]. However, it turns out that the convergence rate deteriorates significantly. Nevertheless, we still obtain a reasonably fast method for unsteady Euler flows. If we deviate further from steady Euler flows and consider the unsteady Navier-Stokes equations, the dual time stepping multigrid method was observed to be very slow, in particular for turbulent flows on high aspect ratio grids [7]. Note that this has been demonstrated in the context of discontinuous Galerkin methods as well [11].

As mentioned, the multigrid method was fine tuned for the steady Euler equations. We believe that this is the reason for the deterioration of the convergence rate for unsteady flows: the method was designed for a different problem and multigrid is known to be sensitive to this. To redesign the method, it is possible to change restriction and prolongation or the smoother. The latter is easier to do in an existing code and therefore, we will taylor the smoother in the multigrid method to unsteady flow problems to see if convergence speed can be improved.

The specific class of smoothers considered here are explicit Runge-Kutta methods (RK methods), which have low storage demands and scale well in parallel. Several parameters can be tuned to obtain a good smoother. This was first considered by Jameson for the slightly more general class of additive RK methods [7]. Thereby, a steady fourth order equation was used as a design equation with the goal of obtaining large stability regions and fast damping of fine grid modes. These schemes are widely used, also for the steady Euler equations. Van Leer et. al. considered the steady linear advection equation [13] and came up with coefficients,

[*]Received November 8, 2011. Accepted for publication July 23, 2012. Published online on October 1, 2012. Recommended by T. Manteuffel. Work supported by the German Research Foundation (DFG) as part of the SFB-TR TRR 30, project C2

[†]Institute of Mathematics, University of Kassel, Heinrich-Plett-Str. 40, D-34132 Kassel, Germany, email: birken@mathematik.uni-kassel.de

which also turned out to work well for the steady Euler equations. Their methodology was to consider the eigenvalues and eigenvectors of the discrete and continuous design system, giving rise to the notion of smooth and nonsmooth error components in this context. Then, the coefficients of the RK method and the CFL number are optimized, such that the smooth error components are damped the fastest.

Here, we use the unsteady linear advection equation as a model equation and discretize this with a finite volume scheme and the implicit Euler method. This leads to a linear system, which is scaled and shifted compared to the one for steady state computations. The aggregation multigrid scheme is applied in a dual time stepping manner. To obtain optimal coefficients, we pursue two strategies. First of all, we apply the technique of van Leer et. al., but to the scaled and shifted system, leading to different results than for the non dual time stepping case. However, this does not take the interactions between the different components of the multigrid method into account. Therefore, we suggest as a second method to optimize the spectral radius of the multigrid iteration matrix. This has the potential of leading to truly optimal schemes, but is significantly more costly than the first approach. To make the results easily reproducible, the code can be downloaded from http://www.mathematik.uni-kassel.de/~birken/mglinadvgl.zip.

The resulting schemes are compared to the methods of van Leer et. al., showing an increase in convergence speed of about 50 percent, just by changing the coefficients of the RK scheme. Thus, it does pay off to redesign the multigrid scheme for the unsteady case, instead of simply reusing the scheme for the steady case.

The relevance of this approach regarding unsteady flow phenomema stems from the fact that for a lot of applications, the interesting features are not on the scale of the fast acoustic eigenvalues, but on the scale of the convective eigenvalues. This makes A-stable implicit schemes for time integration much more interesting than explicit schemes, which are severely restrained by stability conditions. In this context, the implicit Euler method is a prototype that gives important insight for BDF or DIRK methods as well. The applicability of implicit schemes in turn is determined by the availability of fast solvers for the arising large nonlinear equation systems.

If we consider as target application three dimensional unsteady compressible viscous flows, it becomes apparent that a fast solver must have strong parallel scalability and that memory requirements must be low. The above mentioned multigrid method scales reasonably well and has low storage requirements.

The alternative to multigrid is to use Newton's method, which requires the solution of large sparse linear equation systems, usually by preconditioned Krylov subspace methods like GMRES or BiCGSTAB. In particular Jacobian-free methods that circumvent computation and storage of the Jacobian are an attractive option [10]. However, the preconditioner should be chosen appropriately Jacobian-free as well. Now, a lot of methods that work well on sequential machines like ILU or SGS do not scale well in parallel, resulting in multigrid appearing as an attractive preconditioner. This leads us to the same conundrum in that we cannot expect multigrid to be a good preconditioner, if it is a slow method.

To improve the steady Euler multigrid method when applying it to different equations or discretizations in the dual time stepping approach, a few approaches have been tried. Jameson and Hsu suggest to use one step of the ADI method, followed by a few multigrid steps for the dual time problem [5], which is similar to using one Newton step, followed by dual time stepping. Bijl and Carpenter on the other hand use several multigrid steps up front, followed by several steps of Newton's methods [2]. Both report an improvement in comparison to the base pure dual time stepping scheme. Recently, Birken and Jameson proved that using multigrid as a nonlinear left preconditioner leads to a stall in Newton convergence [3].

Finally for steady flows, in the case of a discontinuous Galerkin (DG) method, Bassi et. al. [1] used the same technique as van Leer et. al. to come up with optimal coefficients for that discretization, again for the steady linear advection equation and demonstrated that these work reasonably well for the steady Euler equations. Kannan and Wang used the specific class of SSP RK methods as smoothers in a multi-p method in a spectral volume solver and determined the maximal possible time step for the heat equation [8]. Finally, Klaij et. al. analyzed the convergence of a multigrid method for a space-time DG method with specific explicit Runge-Kutta smoothers for the convection-diffusion equation [9].

2. Governing equations and discretization.

We consider the linear advection equation

$$u_t + au_x = 0. \tag{2.1}$$

with $a > 0$ on the interval $x \in [0, 2]$ with periodic boundary conditions.

An equidistant FV discretization for (2.1) with mesh width Δx leads to the evolution equation for the cell average u_i in one cell i:

$$u_{i_t} + \frac{a}{\Delta x}(u_i - u_{i-1}) = 0.$$

Using the vector $\mathbf{u} = (u_1, ..., u_m)^T$ and

$$\mathbf{B} = \begin{bmatrix} 1 & & & & -1 \\ -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \end{bmatrix},$$

we obtain the system of ODEs

$$\mathbf{u}_t + \frac{a}{\Delta x}\mathbf{B}\mathbf{u}(t) = \mathbf{0}. \tag{2.2}$$

Here, we discretize this using implicit Euler with time step size Δt, which is also a building block for the more general diagonally implicit Runge-Kutta (DIRK) methods. Thus, in each time step, a linear system has to be solved. Using the notation $\mathbf{u}^n \approx \mathbf{u}(t_n)$, this can be written as

$$\mathbf{u}^{n+1} - \mathbf{u}^n + \frac{a\Delta t}{\Delta x}\mathbf{B}\mathbf{u}^{n+1} = 0$$

$$\Leftrightarrow \mathbf{u}^n - \mathbf{A}\mathbf{u}^{n+1} = 0 \tag{2.3}$$

where

$$\mathbf{A} = \mathbf{I} + \frac{\nu}{\Delta x}\mathbf{B}, \tag{2.4}$$

with $\nu = a\Delta t$. Here, $CFL := a\Delta t/\Delta x$ corresponds to the CFL number in the implicit Euler method. If we consider nonperiodic boundary conditions, the entry in the upper right corner of $\mathbf{B}$ becomes zero. Otherwise, additional terms appear on the right hand side, but this does not affect multigrid convergence.

ETNA
Kent State University
http://etna.math.kent.edu

3. Basic multigrid method.

To solve equation (2.3), an agglomeration multigrid method is used, which corresponds best to finite volume discretizations. Thus, the restriction and prolongation correspond to joining and dividing neighboring cells and are given by

$$\mathbf{R} = \frac{1}{2}\begin{bmatrix} 1 & 1 & & & & & \\ & & 1 & 1 & & & \\ & & & \ddots & \ddots & & \\ & & & & & 1 & 1 \end{bmatrix} \text{ and } \mathbf{P} = 2\mathbf{R}^T = \begin{bmatrix} 1 & & & \\ 1 & & & \\ & 1 & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \\ & & & 1 \end{bmatrix}.$$

Note that this implicitely restricts us to cases with an even number of cells.

The coarse grid matrix is obtained by discretizing the problem on that grid. On the coarsest level, the smoother is applied instead of the usual direct solver, since this better corresponds to the Full Approximation Scheme used for the nonlinear equations. We use a V-cycle and presmoothing only. Thus we obtain the scheme:

```
Function MG($\mathbf{x}_l, \mathbf{b}_l, l$)
  • $\mathbf{x}_l = \mathbf{S}_l^{\nu_1}(\mathbf{x}_l, \mathbf{b}_l)$ (Presmoothing)
  • if $(l > 0)$
      – $\mathbf{r}_{l-1} = \mathbf{R}_{l-1,l}(\mathbf{b}_l - \mathbf{A}_l\mathbf{x}_l)$ (Restriction)
      – $\mathbf{v}_{l-1} = 0$
      – Call $MG(\mathbf{v}_{l-1}, \mathbf{r}_{l-1}, l-1)$ (Computation of the coarse grid correction)
      – $\mathbf{x}_l = \mathbf{x}_l + \mathbf{P}_{l,l-1}\mathbf{v}_{l-1}$ (Correction via Prolongation)
  • end if
```

As smoothers, we consider s-stage low-storage explicit Runge-Kutta schemes. These approximate the solution of an initial value problem

$$\mathbf{u}_t = \mathbf{f}(\mathbf{u}), \quad \mathbf{u}_n = \mathbf{u}(t_n),$$

and are of the form

$$\begin{aligned} \mathbf{u}_0 &= \mathbf{u}_n \\ \mathbf{u}_j &= \mathbf{u}_n + \alpha_j \Delta t^* \mathbf{f}(\mathbf{u}_{j-1}),\ j = 1, ..., s-1 \\ \mathbf{u}_{n+1} &= \mathbf{u}_n + \Delta t^* \mathbf{f}(\mathbf{u}_{s-1}), \end{aligned}$$

where the α_j and Δt^* are free parameters and we make the consistency requirement that $\alpha_j \in [0, 1]$. The differential equation resulting from a dual time stepping approach to (2.3) is a hyperbolic equation with source terms in pseudo time t^*:

$$\mathbf{u}_{t^*} = \mathbf{u}^n - \mathbf{A}\mathbf{u}(t^*), \quad \mathbf{u}(t_0^*) = \mathbf{u}^n. \tag{3.1}$$

One step of the RK smoother thus consists of performing one step of the RK scheme for the solution of equation (3.1).

Note that if we were interested in the computation of steady states and wanted to use time marching to compute these, the relevant equation would be (2.2). Here, for unsteady problems, we have to look at the system (3.1) instead. The difference is an identity shift and a factor of Δt in front of the matrix $\mathbf{B}$.

For a linear problem with $\mathbf{f}(\mathbf{u}) = \mathbf{A}\mathbf{u}$, one step of an explicit s-stage RK smoother can be described by its stability polynomial P_s of degree s as

$$\mathbf{u}_{n+1} = P_s(\Delta t^* \mathbf{A})\mathbf{u}_n,$$

with Δt^* the time step in pseudotime. For example, an explicit 2-stage RK scheme is represented by

$$P_2(z) = 1 + z + \alpha_1 z^2, \tag{3.2}$$

whereas for a 3-stage scheme we have two free parameters α_1 and α_2 and the polynomial

$$P_3(z) = 1 + z + \alpha_2 z^2 + \alpha_1\alpha_2 z^3. \tag{3.3}$$

Finally, we also consider 4-stage schemes with parameters α_1, α_2 and α_3 with polynomial

$$P_4(z) = 1 + z + \alpha_3 z^2 + \alpha_3\alpha_2 z^3 + \alpha_1\alpha_2\alpha_3 z^4. \tag{3.4}$$

4. Optimizing the smoother.

4. Optimizing the smoother. We will consider two different ways of finding good smoothers, both involving an optimization process. First of all, we can try to optimize the smoother alone, such that it removes nonsmooth error components fast. This approach was followed in [13] and later in [1]. Additionally, we suggest to compute the iteration matrix of the multigrid scheme and minimize its spectral radius in a discrete way.

The difference between the two approaches is that the latter one is theoretically able to provide truly optimal schemes, whereas the first one does not. However, the second approach is much more costly, meaning that we are not able to compute the global optimum in a reasonable time. Therefore, both approaches are discussed here.

4.1. Optimizing the smoothing properties. For the first approach, the eigenvectors of the matrix $\mathbf{A}$ from (2.4) are discrete forms of the functions $e^{ix\Theta}$ for various Θ and the eigenvalues are given by

$$\lambda(\Theta) = -1 - \frac{\nu}{\Delta x}(1 - e^{-i\Theta}).$$

If nonperiodic boundary conditions are used, the matrix becomes lower triangular and all eigenvalues are equal to $-1 - \frac{\nu}{\Delta x}$. In the steady case, the eigenvalues would be scaled and shifted, resulting in $\lambda(\Theta) = -\frac{a}{\Delta x}(1 - e^{-i\Theta})$. Now, on the coarse grid, we can represent functions with $\Theta \in [-\pi/2, \pi/2]$. Thus, the smoother has to take care of error components with $|\Theta| \in [\pi/2, \pi]$.

Due to the linearity, it is sufficient to look at $P_s(\Delta t^* \lambda(\Theta))$ with

$$\Delta t^* \lambda(\Theta) = -\Delta t^* - \frac{\nu \Delta t^*}{\Delta x}(1 - e^{-i\Theta}).$$

Possible parameters of the smoother are the pseudo time step size Δt^* and the coefficients of the RK method. Now, $\nu = a\Delta t$ is fixed during the multigrid iteration, but Δx not. Furthermore, the pseudo time step is restricted by a CFL condition based on ν. Thus, instead of optimizing for Δt^*, we define the pseudo time step on each grid level as

$$\Delta t_l^* = c\Delta x_l / \nu$$

and optimize for $c := \nu \Delta t_l^* / \Delta x_l$. Now we have

$$z(\Theta, c; \nu, \Delta x_l) := \Delta t^* \lambda(\Theta) = -c\Delta x_l/\nu - c + ce^{-i\Theta}, \tag{4.1}$$

where we see that z does not depend on ν and Δx_l separately, but only on $\Delta x_l/\nu = 1/CFL$. Thus, with $e^{-i\Theta} = \cos(\Theta) - i\sin(\Theta)$ we obtain

$$z(\Theta, c; CFL) = -c/CFL - c + c\cos(\Theta) - ic\sin(\Theta).$$

In the end, given CFL, we have to solve an optimization problem where we look at the modulus of the maximal value of the smoother for $|\Theta| \in [\pi/2, \pi]$ and then minimize that over the parameters α_j and c. Using symmetry of P_s and equivalenty looking at the square of the modulus, we obtain

$$\min_{c,P_s} \max_{|\Theta|\in[\pi/2,\pi]} |P_s(z(\Theta, c; CFL))|^2. \tag{4.2}$$

Due to the dependence of the optimal coefficients on CFL, there is no unique optimal smoother for all problems.

For the 2-stage scheme (3.2), we have:

$$\begin{aligned}|P_2(z)|^2 &= |1 + z + \alpha_1 z^2|^2,\\ &= (1 + \text{Re}z + \alpha_1\text{Re}z^2 - \alpha_1\text{Im}z^2)^2 + (2\alpha_1\text{Re}z\text{Im}z + \text{Im}z)^2.\end{aligned}$$

Similar computations for the 3-stage scheme (3.3) lead to

$$\begin{aligned}|P_3(z)|^2 = &(1 + \text{Re}z + \alpha_2\text{Re}z^2 - \alpha_2\text{Im}z^2 + \alpha_1\alpha_2\text{Re}z^3 - 3\alpha_1\alpha_2\text{Re}z\text{Im}z^2)^2\\ &+(\text{Im}z + 2\alpha_2\text{Re}z\text{Im}z - \alpha_1\alpha_2\text{Im}z^3 + 3\alpha_1\alpha_2\text{Re}z^2\text{Im}z)^2\end{aligned}$$

and for the 4-stage scheme we obtain

$$\begin{aligned}|P_4(z)|^2 = &\big(1 + \text{Re}z + \alpha_1\text{Re}z^2 - \alpha_1\text{Im}z^2 + \alpha_1\alpha_2\text{Re}z^3 - 3\alpha_1\alpha_2\text{Re}z\text{Im}z^2\\ &\quad + \alpha_1\alpha_2\alpha_3\text{Re}z^4 - 6\alpha_1\alpha_2\alpha_3\text{Re}z^2\text{Im}z^2 + \alpha_1\alpha_2\alpha_3\text{Im}z^4\big)^2\\ &+ \big(\text{Im}z + 2\alpha_1\text{Re}z\text{Im}z + 3\alpha_1\alpha_2\text{Re}z^2\text{Im}z - \alpha_1\alpha_2\text{Im}z^3\\ &\quad + 4\alpha_1\alpha_2\alpha_3\text{Re}z^3\text{Im}z - 4\alpha_1\alpha_2\alpha_3\text{Re}z\text{Im}z^3\big)^2.\end{aligned}$$

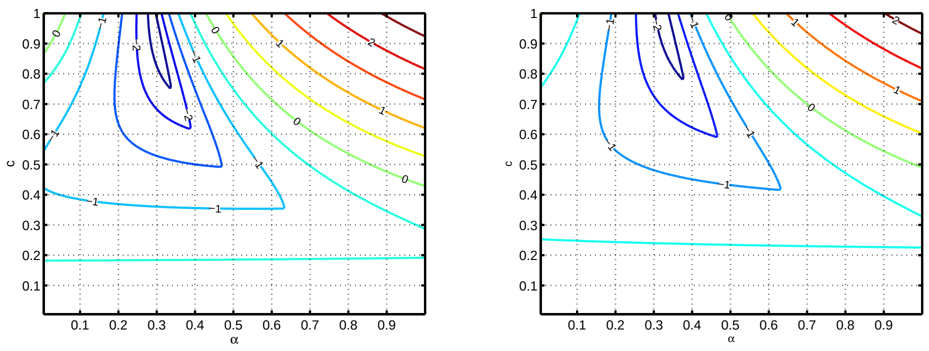

FIG. 4.1. *Contourplots of functions* $\log_{10}\max_{|\Theta|\in[\pi/2,\pi]}|P_2(z(\Theta, c; 3))|^2$ *(left) and* $\log_{10}\max_{|\Theta|\in[\pi/2,\pi]}|P_2(z(\Theta, c; 24))|^2$ *(right).*

It turns out that for these functions, the final form of (4.2) is too difficult to solve exactly, in particular due to the min-max-formulation. Therefore, we discretize the parameter space

TABLE 4.1
Results of optimization of smoothing properties for 2-stage scheme.

CFL	α	c	Opt-value
1	0.275	0.745	0.01923
3	0.3	0.93	0.05888
6	0.315	0.96	0.08011
9	0.32	0.975	0.08954
12	0.325	0.97	0.09453
24	0.33	0.98	0.10257

TABLE 4.2
Results of optimization of smoothing properties for 3-stage scheme.

CFL	α_1	α_2	c	Opt-value
1	0.12	0.35	1.14	0.001615
3	0.135	0.375	1.37	0.007773
6	0.14	0.385	1.445	0.01233
9	0.14	0.39	1.45	0.01486
12	0.145	0.395	1.44	0.01588
24	0.145	0.395	1.495	0.01772

and compute an approximate solution. This requires a bounded region, which is already the case for Θ and the α_j, which are between 0 and 1. As for c, we know that any explicit RK scheme has a bounded stability region, therefore we chose an upper bound for c, such that the optimal value for c is not on the boundary. As an example, the function

$$f(\alpha, c) := \log_{10} \max_{|\Theta|\in[\pi/2,\pi]} |P_2(z(\Theta, c; CFL))|^2$$

is shown in Figure 4.1 for $CFL = 3$ and $CFL = 24$. Note that the optimal c is not on the boundary, meaning that the choice $c \in [0, 1]$ here is reasonable. Furthermore, we can see that for $c = 0$, we obtain a method with a value of 1. This is correct, since this is a method with time step zero, meaning that the resulting smoother is the identity. For $\alpha = 0$, we obtain the explicit Euler method. This is also a possible smoother, but as can be seen it is less powerful. Furthermore, we can see the finite stability regions of the methods.

We now compute the optimal value for $CFL = 1, 3, 6, 9, 12, 24$, for all schemes using a MATLAB/C++ code. For the 2-stage scheme, we choose a grid of $200 \times 200 \times 200$ for the parameter space $\alpha_1 \times c \times t$. The optimization gives results presented in Table 4.1. As can be seen, $CFL = 1$ is a special case, otherwise the parameter α_1 does not depend on CFL, whereas there is a slight increase of c with CFL.

Table 4.2 shows the results for the 3-stage scheme. There we have one additional parameter, leading to the parameter space $\alpha_1 \times \alpha_2 \times c \times t$ and the grid $200 \times 200 \times (2 \cdot 200) \times 200$. As a restriction for c, we put $c \in [0, 2]$. Again, $CFL = 1$ is a special case in that a significantly smaller value for c comes out. Otherwise, the coefficients α_1 and α_2 have only a weak dependence on CFL. However, the optimal value is decreased by a factor of about 500, suggesting that these schemes are significantly better than the 2-stage methods.

Table 4.3 shows the results for the 4-stage scheme, where we chose a grid of the size $150 \times 150 \times 150 \times (2 \cdot 150) \times 100$ with $c \in [0, 2]$. A finer grid was not possible due to storage restrictions. As for the 3-stage case, a significantly smaller value for c is obtained for $CFL = 1$, otherwise there is only a weak dependence of the parameters on CFL. The optimal value is decreased by a factor of four compared to the 3-stage schemes.

TABLE 4.3
Results of optimization of smoothing properties for 4-stage scheme.

CFL	α_1	α_2	α_3	c	Opt-value
1	0.0733	0.1867	0.4	1.3267	0.0005302
3	0.0733	0.1867	0.4	1.96	0.002118
6	0.08	0.2	0.4133	1.907	0.00297
9	0.08	0.2	0.4133	1.98	0.003372
12	0.08	0.2	0.42	1.907	0.003972
24	0.0867	0.2133	0.433	1.84	0.004313

An important difference between the schemes obtained is the size of c, which is about 0.9 for the 2-stage case, 1.4 for the 3-stage case and 1.9 for the 4-stage case, which suggests that one effect of allowing more freedom in the design of the smoother is that the stability region is increased. The stability regions of the optimal methods obtained for $CFL = 3$ and $CFL = 24$ are shown in Figure 4.2, emphasizing this point.

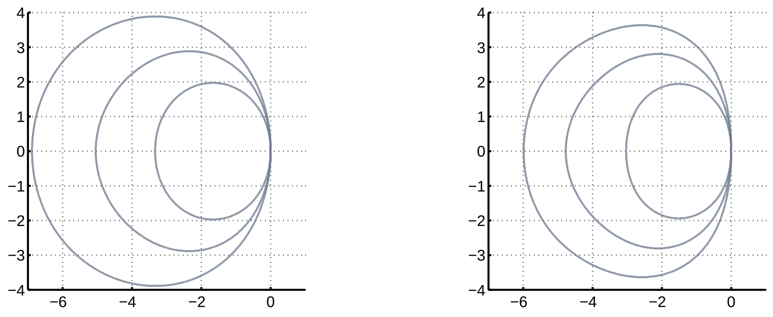

FIG. 4.2. *Stability regions of optimally smoothing methods for $CFL = 3$ (left) and $CFL = 24$ (right). The larger the stability region, the higher the number of stages.*

4.2. Optimizing the spectral radius. The optimization just considered aims at improving the smoother on its own, without taking into account the interaction with the coarse grid correction or the multigrid structure. This has the benefit that the optimization is fast, even for the 4-stage case, where we run into memory problems. An alternative approach is obtained by remembering that a linear multigrid scheme can be written as a linear iterative scheme

$$\mathbf{u}^{(k+1)} = \mathbf{M}\mathbf{u}^{(k)} + \mathbf{N}\mathbf{b}.$$

The optimal scheme is then obtained by optimizing the spectral radius of the iteration matrix $\mathbf{M}$ as a function of the smoother, which in turn is a function of α and c, with c defined as above:

$$\min_{\alpha,c} \rho(\mathbf{M}(\alpha, c; \nu, \Delta x)). \tag{4.3}$$

Regarding the iteration matrix, it is important to note that the smoother is furthermore a function of a right hand side and an approximate solution.

Thus, when applying the s-stage Runge-Kutta smoother on level l to an ODE with right hand side $\mathbf{f}_l(\mathbf{u}) = \mathbf{b} - \mathbf{A}_l\mathbf{u}$, there is an additional term on top of the stability polynomial, depending on the vector $\mathbf{b}$:

$$\mathbf{S}_{s,l}(\mathbf{b},\mathbf{u}) = \mathbf{S}^u_{s,l}\mathbf{u} + \mathbf{S}^b_{s,l}\mathbf{b}.$$

Here, $\mathbf{S}^u_{s,l} = P_s(-\Delta t^* \mathbf{A}_l)$ is the matrix coming out of the stability polynomial, whereas the second matrix corresponds to a different polynomial. For the 2-, 3- and 4-stage smoother, we have

$$\mathbf{S}^b_{s,l} = \Delta t \mathbf{I}_l + \alpha_1 \Delta t^2 \mathbf{A}_l, \tag{4.4}$$

$$\mathbf{S}^b_{3,l} = \Delta t \mathbf{I}_l + \alpha_1 \Delta t^2 \mathbf{A}_l + \alpha_1\alpha_2 \Delta t^3 \mathbf{A}_l^2, \tag{4.5}$$

$$\mathbf{S}^b_{4,l} = \Delta t \mathbf{I}_l + \alpha_1 \Delta t^2 \mathbf{A}_l + \alpha_1\alpha_2 \Delta t^3 \mathbf{A}_l^2 + \alpha_1\alpha_2\alpha_3 \Delta t^4 \mathbf{A}_l^3. \tag{4.6}$$

When first calling the multigrid function, this is done with the actual right hand side $\mathbf{b}$ and the current approximation $\mathbf{u}^{(k)}$. However, on lower levels the right hand side is the restricted residual $\mathbf{R}_{l-1,l}\mathbf{r}_l$ and the current approximation is the zero vector. For a three-level scheme, we have

$$\begin{aligned} \mathbf{u}^{(k+1)} &= \mathbf{S}^u_{s,2}\mathbf{u}^{(k)} + \mathbf{S}^b_{s,2}\mathbf{b} + \mathbf{P}_{2,1}\mathbf{MG}(\mathbf{0}, \mathbf{r}_1, 1) \\ &= \mathbf{S}^u_{s,2}\mathbf{u}^{(k)} + \mathbf{S}^b_{s,2}\mathbf{b} + \mathbf{P}_{2,1}(\mathbf{S}^b_{s,1}\mathbf{r}_1 + \mathbf{P}_{1,0}\mathbf{S}^b_{s,0}\mathbf{r}_0), \end{aligned} \tag{4.7}$$

where

$$\mathbf{r}_0 = \mathbf{R}_{0,1}(\mathbf{I}_1 - \mathbf{A}_1\mathbf{S}^b_{s,1})\mathbf{r}_1, \tag{4.8}$$

and

$$\begin{aligned} \mathbf{r}_1 &= \mathbf{R}_{1,2}(\mathbf{b} - \mathbf{A}_2(\mathbf{S}^b_{s,2}\mathbf{b} + \mathbf{S}^u_{s,2}\mathbf{u}^{(k)})) \\ &= \mathbf{R}_{1,2}(\mathbf{I}_2 - \mathbf{A}_2\mathbf{S}^b_{s,2})\mathbf{b} - \mathbf{R}_{1,2}\mathbf{A}_2\mathbf{S}^u_{s,2}\mathbf{u}^{(k)}. \end{aligned} \tag{4.9}$$

Inserting (4.8) and (4.9) into (4.7), we obtain

$$\begin{aligned} \mathbf{u}^{(k+1)} &= \mathbf{S}^u_{s,2}\mathbf{u}^{(k)} + \mathbf{S}^b_{s,2}\mathbf{b} + \mathbf{P}_{2,1}(\mathbf{S}^b_{s,1} + \mathbf{P}_{1,0}\mathbf{S}^b_{s,0}\mathbf{R}_{0,1}(\mathbf{I}_1 - \mathbf{A}_1\mathbf{S}^b_{s,1}))\mathbf{r}_1 \\ &= (\mathbf{S}^b_{s,2} + \mathbf{P}_{2,1}(\mathbf{S}^b_{s,1} + \mathbf{P}_{1,0}\mathbf{S}^b_{s,0}\mathbf{R}_{0,1}(\mathbf{I}_1 - \mathbf{A}_1\mathbf{S}^b_{s,1}))\mathbf{R}_{1,2}(\mathbf{I}_2 - \mathbf{A}_2\mathbf{S}^b_{s,2}))\mathbf{b} \\ &\quad + (\mathbf{S}^u_{s,2} - \mathbf{P}_{2,1}(\mathbf{S}^b_{s,1} + \mathbf{P}_{1,0}\mathbf{S}^b_{s,0}\mathbf{R}_{0,1}(\mathbf{I}_1 - \mathbf{A}_1\mathbf{S}^b_{s,1}))\mathbf{R}_{1,2}\mathbf{A}_2\mathbf{S}^u_{s,2})\mathbf{u}^{(k)}. \end{aligned}$$

Thus, the iteration matrix of the three level scheme is given by

$$\mathbf{M} = \mathbf{S}^u_{s,2} - \mathbf{P}_{2,1}(\mathbf{S}^b_{s,1} + \mathbf{P}_{1,0}\mathbf{S}^b_{s,0}\mathbf{R}_{0,1}(\mathbf{I}_1 - \mathbf{A}_1\mathbf{S}^b_{s,1}))\mathbf{R}_{1,2}\mathbf{A}_2\mathbf{S}^u_{s,2}.$$

To solve the optimization problem (4.3), we again compute a discrete optimum, this time using a MATLAB code and the eig function to obtain the spectral radius. Two alternatives were considered, firstly using eigs(M,1), which was slower than eig, probably because the matrices considered are so small and secondly the functions of the optimization tool box. To this end, problem (4.3) was solved using the functions fminsearch and fminunc. This was significantly faster, but the solutions found had larger function values than the discrete search, in particular for the larger parameter spaces.

For the 2-stage case, we use a grid of size $200 \times (2 \cdot 200)$ with $c \in [0, 2]$. Here, both ν and Δx have to be varied, where we chose ν according to the same CFL numbers as for the last strategy and $\Delta x = 1/24, 1/12, 1/6$. The results are shown in Table 4.4. The contour lines of the function $\rho(\mathbf{M}(\alpha, c; \nu, \Delta x))$ are illustrated in Figure (4.3). As can be seen, the results are qualitatively similar to the ones from the other strategy.

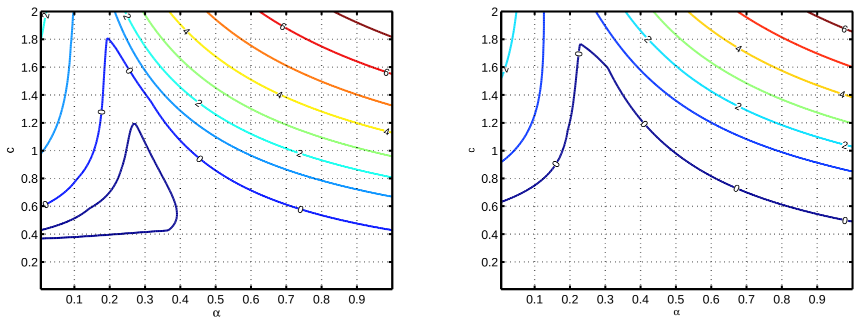

FIG. 4.3. *2-stage method: Contourplots of* $\log(\rho(\mathbf{M}(\alpha, c)))$ *for* $\Delta x = 1/24$ *and* $\nu = 0.125$ *(left) and* $\nu = 1.0$ *(right).*

TABLE 4.4
Results of optimization of $\rho(\mathbf{M})$ *for 2-stage scheme.*

	$\Delta x = 1/24$				$\Delta x = 1/12$			
CFL	ν	α	c	$\rho(\mathbf{M})$	ν	α	c	$\rho(\mathbf{M})$
1	1/24	0.25	0.545	0.0689	1/12	0.25	0.56	0.0681
3	0.125	0.21	0.615	0.2072	0.25	0.21	0.615	0.2072
6	0.25	0.305	0.94	0.3007	0.5	0.315	0.96	0.2954
9	0.375	0.295	1.145	0.3824	0.75	0.3	1.145	0.3819
12	0.5	0.295	1.255	0.4584	1.0	0.3	1.26	0.4575
24	1.0	0.295	1.46	0.6425	2.0	0.29	1.485	0.6371
	$\Delta x = 1/6$							
1	1/6	0.245	0.555	0.0673				
3	0.5	0.27	0.77	0.1851				
6	1.0	0.295	1.0	0.2734				
9	1.5	0.29	1.175	0.3694				
12	2.0	0.29	1.29	0.4473				
24	4.0	0.28	1.51	0.6315				

Tables 4.5 and 4.6 show the results for the 3-stage and 4-stage case, respectively, where we used grids of the sizes $100 \times 100 \times (2 \cdot 100)$ and $50 \times 50 \times 50 \times (2 \cdot 50)$. The decrease in mesh width is due to the polynomially growing computational cost. One optimization on the fine grid in the 2-stage case takes a couple of minutes, whereas the 4-stage case needs more than ten hours, despite the coarser grid.

Comparing the results for the different stages, we see that there is no dependence of the optimal solution on Δx in the sense that for a fixed CFL number and accordingly chosen ν, the results are almost identical. In this sense, the multigrid method obtained has a convergence speed which is independent of the mesh width. Otherwise, an increase in CFL number leads to an increase in the spectral radius and thus a decrease of the convergence speed of the methods. Regarding the coefficients, there is a weak dependence of α on the CFL number, but c increases significantly with CFL. Regarding the size of the spectral radius, going from two to three stages leads to a huge decrease of the optimal value for small and moderate CFL number, but not for large CFL numbers. As for adding a fourth stage, this actually leads to a decrease in spectral radius. This can be explained by the much coarser grid used for the optimization of the 4-stage method. Consequently, the solution found is too far away from

TABLE 4.5
Results of optimization of $\rho(\mathbf{M})$ for 3-stage scheme.

	$\Delta x = 1/24$					$\Delta x = 1/12$				
CFL	ν	α_1	α_2	c	$\rho(\mathbf{M})$	ν	α_1	α_2	c	$\rho(\mathbf{M})$
1	1/24	0.11	0.33	0.69	0.0402	1/12	0.11	0.33	0.69	0.0402
3	0.125	0.14	0.39	1.42	0.0819	0.25	0.14	0.39	1.43	0.0799
6	0.25	0.14	0.4	1.58	0.1444	0.5	0.14	0.4	1.6	0.1397
9	0.375	0.12	0.37	1.75	0.2317	0.75	0.12	0.36	1.77	0.2237
12	0.5	0.13	0.39	1.91	0.3124	1.0	0.12	0.36	1.95	0.2954
24	1.0	0.12	0.38	2.14	0.5252	2.0	0.10	0.31	2.42	0.4720
	$\Delta x = 1/6$									
1	1/6	0.11	0.33	0.68	0.0381					
3	0.5	0.14	0.39	1.43	0.0799					
6	1.0	0.13	0.38	1.67	0.1375					
9	1.5	0.12	0.36	1.78	0.2230					
12	2.0	0.11	0.34	1.93	0.2948					
24	4.0	0.09	0.28	2.59	0.4427					

TABLE 4.6
Results of optimization of $\rho(\mathbf{M})$ for 4-stage scheme.

	$\Delta x = 1/24$						$\Delta x = 1/12$					
CFL	ν	α_1	α_2	α_3	c	$\rho(\mathbf{M})$	ν	α_1	α_2	α_3	c	$\rho(\mathbf{M})$
1	1/24	0.02	0.08	0.3	0.62	0.0525	1/12	0.02	0.08	0.3	0.62	0.0525
3	0.125	0.02	0.14	0.38	1.42	0.1138	0.25	0.02	0.14	0.38	1.42	0.1138
6	0.25	0.02	0.14	0.38	1.52	0.1783	0.5	0.02	0.14	0.38	1.52	0.1783
9	0.375	0.02	0.14	0.4	1.7	0.2501	0.75	0.02	0.12	0.36	1.78	0.2365
12	0.5	0.02	0.12	0.36	1.9	0.3053	1.0	0.02	0.12	0.34	1.88	0.3040
24	1.0	0.02	0.12	0.34	2.16	0.5173	2.0	0.02	0.1	0.30	2.18	0.5094
	$\Delta x = 1/6$											
1	1/6	0.02	0.08	0.3	0.62	0.0492						
3	0.5	0.02	0.14	0.38	1.42	0.1138						
6	1.0	0.02	0.14	0.38	1.52	0.1783						
9	1.5	0.02	0.12	0.36	1.78	0.2236						
12	2.0	0.02	0.12	0.34	1.88	0.3040						
24	4.0	0.02	0.10	0.32	2.34	0.4858						

the actual optimum to beat the 3-stage method.

In Figure 4.4, the stability region of the optimal methods for $\Delta x = 1/24$ and $\nu = 0.125$, as well as $\nu = 1.0$ are shown. Again, an increase in the number of stages leads to a larger stability region.

Comparing the optimal solutions found with the schemes obtained by the previous method, we see that the coefficients are similar, as is the value of c and the same goes for the stability regions.

4.3. Comparison of costs. We now compare the costs of the two approaches. Denoting by n_c, n_α and n_Θ the number of points used for the respective parameter space, the first approach needs $n_c n_\alpha n_\Theta$ evaluations of the function $|P_s(z(\Theta, c; CFL))|^2$. On top of that $\max_{|\Theta| \in [\pi/2, \pi]} |P_s(z(\Theta, c; CFL))|^2$ has to be computed $n_c n_\alpha$ times, followed by a minimum over $n_c n_\alpha$ values. The second approach needs $n_c n_\alpha$ computations of $\rho(\mathbf{M}(\alpha, c; \nu, \Delta x))$, followed by one mininum over $n_c n_\alpha$ values. Since in MATLAB, the computation of the spec-

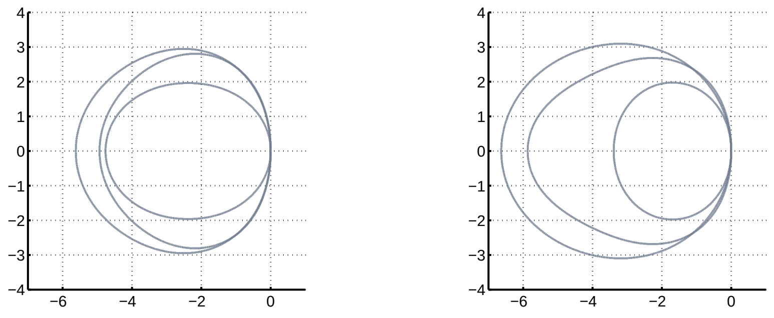

FIG. 4.4. *Stability region of optimal 2-, 3- and 4-stage method for* $\Delta x = 1/24$ *and* $\nu = 0.125$ *(left) and* $\nu = 1.0$ *(right). The larger the number of stages, the larger the stability region.*

tral radius is about a factor of 1,000 more expensive than evaluating the above function, the second approach is more costly.

A speedup could be obtained by implementing the method in C++, thus avoiding MATLABs problems with loops.

5. Numerical results.

We test the smoothers on two problems with $\Delta x = 1/24$ on the finest level and $a = 2.0$. As initial conditions, we use a step function with values 5 and 1, as well as the function $\sin(\pi x)$. We then perform one time step with $\Delta t = 1/16$, respectively $\Delta t = 0.5$, meaning that $\nu = 0.125$ and $CFL = 3$ (a problem with a medium CFL number), respectively $\nu = 1.0$ and $CFL = 24$ (a problem with a large CFL number). The resulting linear equation system is solved with 80 steps of the different multigrid methods. As a reference, the optimal 2- and 3-stage methods derived by van Leer et. al. [13] for steady state problems are used. The 2-stage method is given by $\alpha = 1/3$ and a $c = 1$, whereas the 3-stage method does not actually fall into the framework considered here. It consists of one step of the explicit Euler method with $c = 0.5$, followed by a step of a 2-stage method with $\alpha = 0.4$ and $c = 1$. All computations are performed using MATLAB.

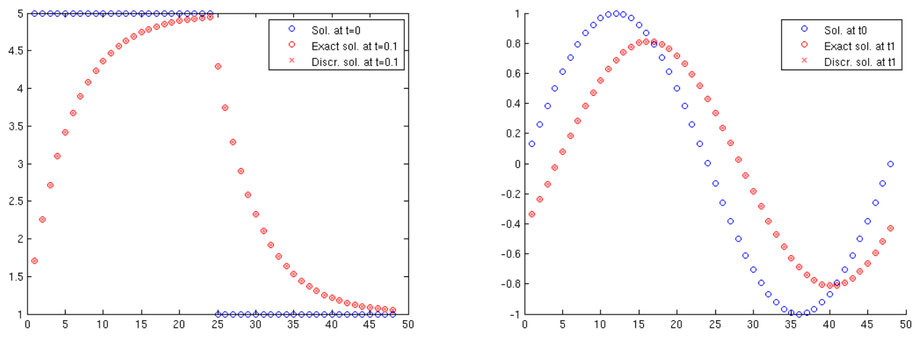

FIG. 5.1. *Initial solution and discrete and numerical solution after one time step for step function (left) and sine initial data (right).*

In Figure 5.1, the computed solutions for the linear advection equation are shown. The initial data is always shown in blue, whereas the exact solution is in red, as well as the numerical one. Since the space discretization is of first order and the time integration method is implicit Euler, the results are very diffusive. However, the multigrid method computes the correct solution.

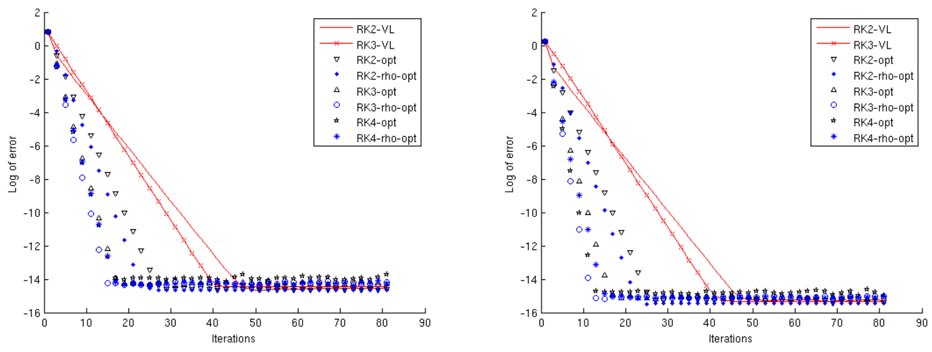

FIG. 5.2. *Convergence plots for different methods and $CFL = 3$: step function (left) and sine initial data (right).*

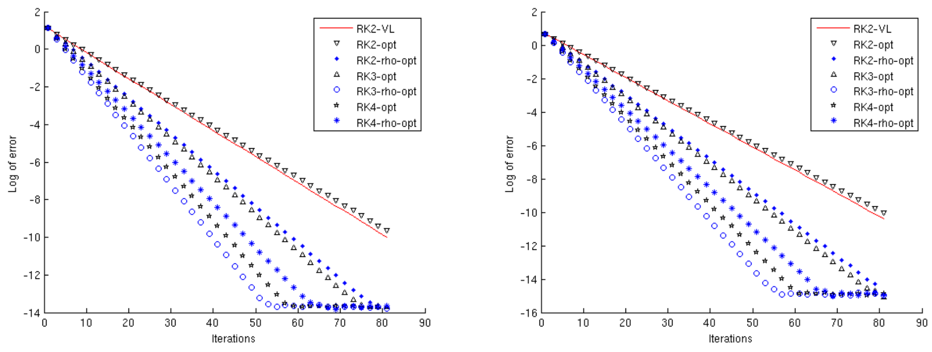

FIG. 5.3. *Convergence plots for different methods and $CFL = 24$: step function (left) and sine initial data (right).*

Regarding computational effort, the main work of the multigrid method consists of matrix vector multiplications in the smoother. Thus, the 3-stage schemes are conservatively 50% more costly than the 2-stage schemes, whereas the 4-stage schemes are less than twice as expensive as the RK-2 smoother.

We now look at the convergence speed of the different methods, where we call the methods obtained by the second optimization ρ-optimized schemes. In Figure 5.2 and 5.3, $\log_{10}$ of the error in the 2-norm is plotted over multigrid iterations, where the first shows the results for $CFL = 3$ for both test cases and the latter the results for $CFL = 24$ for both test cases. The 3-stage method of van Leer diverges for $CFL = 24$ and is only shown in the first figure, where it is barely faster than the 2-stage method of van Leer. Otherwise we can see, that the ρ-optimized schemes behave as expected in that the 3-stage scheme is the fastest, then the 4-stage scheme and then the 2-stage scheme with the 3-stage scheme being roughly twice as fast as the 2-stage scheme. For the schemes coming out of the first optimization, there the 4-stage scheme is faster than the 3-stage scheme, which is faster than the 2-stage scheme. Furthermore, the ρ-optimized schemes are able to beat their counterparts with the exception of the 4-stage scheme. Thus, the more costly optimization is generally worthwhile.

Generally, the 3-stage ρ-optimized scheme is the fastest, in particular it is almost twice as fast as the 2-stage ρ-optimized scheme, making it more efficient. Compared to the reference method of van Leer, it is between two and four times faster, making it between 70% and 270% more efficient. Thus, just by changing the coefficients of the RK smoother, we can expect to gain more than a factor of two in multigrid efficiency.

Finally, we consider the step function case with nonperiodic boundary, to see if the different eigenvalues respectively different matrices lead to problems. As can be seen in Figure 5.4, this is not the case for $CFL = 3$, as the convergence rate for all methods is almost unchanged, but not so for $CFL = 24$, where the new methods have the same convergence speed, but van Leers 2-stage method becomes as fast as the ρ-optimized 2-stage method.

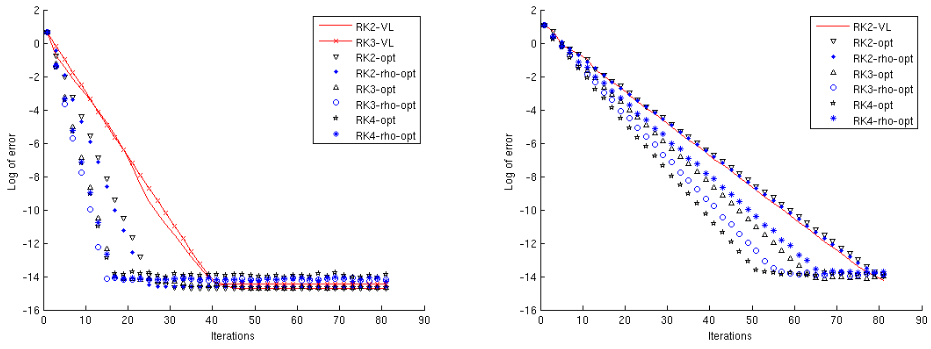

FIG. 5.4. *Convergence plots for different methods for step function initial data with nonperiodic boundary conditions: CFL 3 (left) and CFL 24 (right).*

6. Conclusions and outlook. We developed optimal explicit 2-, 3- and 4-stage Runge-Kutta smoothers for the unsteady linear advection equation. These were demonstrated to improve convergence speed by a factor of two or more, compared to using a method designed for steady state. This shows that it does pay to optimize for unsteady flows instead of reusing the method for steady flows. The optimal smoother does depend on the problem parameters, but only weakly. The optimization was done in two different ways, one considering the effect of the smoother on fine grid modes only, whereas the other optimized the spectral radius of the complete 3-level iteration matrix, being more accurate, but more costly. The best method was the 3-stage method from the second approach, showing that the more costly approach pays off. To find methods that perform well for unsteady Euler or even Navier-Stokes equations, we will pursue the use of more complex model equations for future research.

REFERENCES

[1] F. BASSI, A. GHIDONI, AND S. REBAY, *Optimal Runge-Kutta smoothers for the p-multigrid discontinuous Galerkin solution of the 1D Euler equations*, J. Comp. Phys., 11 (2011), pp. 4153–4175.
[2] H. BIJL AND M. H. CARPENTER, *Iterative solution techniques for unsteady flow computations using higher order time integration schemes*, Int. J. Num. Meth. in Fluids, 47 (2005), pp. 857–862.
[3] P. BIRKEN AND A. JAMESON, *On nonlinear preconditioners in Newton-Krylov methods for unsteady flows*, Int. J. Num. Meth. Fluids, 62 (2010), pp. 565–573.
[4] D. A. CAUGHEY AND A. JAMESON, *How many steps are required to solve the Euler equations of steady compressible flow: In search of a fast solution algorithm*, AIAA Paper 2001-2673 (2001).
[5] J. HSU AND A. JAMESON, *An implicit-explicit hybrid scheme for calculating complex unsteady flows*, AIAA Paper 2002-0714 (2002).
[6] A. JAMESON, *Time dependent calculations using multigrid, with applications to unsteady flows past airfoils and wings*, AIAA Paper 91-1596 (1991).

[7] ———, *Aerodynamics*, in Encyclopedia of Computational Mechanics, E. Stein, R. de Borst, and T. J. R. Hughes, eds., vol. 3, John Wiley & Sons, 2004, pp. 325–406.

[8] R. KANNAN AND Z. J. WANG, *A study of viscous flux formulations for a p-multigrid spectral volume Navier Stokes solver*, J. Sci. Comput., 41 (2009), pp. 165–199.

[9] C. M. KLAIJ, M. H. VAN RAALTE, J. J. W. VAN DER VEGT, AND H. VAN DER VEN, *h-Multigrid for space-time discontinuous Galerkin discretizations of the compressible Navier-Stokes equations*, J. Comp. Phys., 227 (2007), pp. 1024–1045.

[10] D. A. KNOLL AND D. E. KEYES, *Jacobian-free Newton-Krylov methods: a survey of approaches and applications*, J. Comp. Phys., 193 (2004), pp. 357–397.

[11] H. LUO, J. D. BAUM, AND R. LÖHNER, *A p-multigrid discontinuous Galerkin method for the Euler equations on unstructured grids*, J. Comp. Phys., 211 (2006), pp. 767–783.

[12] D. J. MAVRIPLIS, *An assessment of linear versus nonlinear multigrid methods for unstructured mesh solvers*, J. Comp. Phys., 175 (2002), pp. 302–325.

[13] B. VAN LEER, C.-H. TAI, AND K. G. POWELL, *Design of optimally smoothing multi-stage schemes for the Euler equations*, in AIAA 89-1933-CP, 1989, pp. 40–59.