Journal of Scientific Computing
https://doi.org/10.1007/s10915-018-0792-9

Preconditioned Smoothers for the Full Approximation Scheme for the RANS Equations

Philipp Birken[1] · **Jonathan Bull**[2] · **Antony Jameson**[3]

Received: 14 October 2017 / Revised: 28 June 2018 / Accepted: 24 July 2018

Abstract
We consider multigrid methods for finite volume discretizations of the Reynolds averaged Navier–Stokes equations for both steady and unsteady flows. We analyze the effect of different smoothers based on pseudo time iterations, such as explicit and additive Runge–Kutta (AERK) methods. Furthermore, by deriving them from Rosenbrock smoothers, we identify some existing schemes as a class of additive W (AW) methods. This gives rise to two classes of preconditioned smoothers, preconditioned AERK and AW, which are implemented the exact same way, but have different parameters and properties. This derivation allows to choose some of these based on results for time integration methods. As preconditioners, we consider SGS preconditioners based on flux vector splitting discretizations with a cutoff function for small eigenvalues. We compare these methods based on a discrete Fourier analysis. Numerical results on pitching and plunging airfoils identify AW3 as the best smoother regarding overall efficiency. Specifically, for the NACA 64A010 airfoil steady-state convergence rates of as low as 0.85 were achieved, or a reduction of 6 orders of magnitude in approximately 25 pseudo-time iterations. Unsteady convergence rates of as low as 0.77 were achieved, or a reduction of 11 orders of magnitude in approximately 70 pseudo-time iterations.

Keywords Unsteady flows · Multigrid · Discrete Fourier analysis · Runge–Kutta smoothers

1 Introduction

We are interested in numerical methods for compressible wall bounded turbulent flows as they appear in many problems in industry. Therefore, both steady and unsteady flows will be considered. Numerically, these are characterized by strong nonlinearities and a large number of unknowns, due to the requirement of resolving the boundary layer. High fidelity

✉ Philipp Birken
philipp.birken@na.lu.se

[1] Centre for the Mathematical Sciences, Numerical Analysis, Lund University, Box 118, Lund, Sweden

[2] Division of Scientific Computing, Department of Information Technology, Uppsala University, Box 337, 75105 Uppsala, Sweden

[3] Department of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305, USA

Published online: 09 August 2018

approaches such as Direct Numerical Simulation (DNS) or Large Eddy Simulation (LES) are slowly getting within reach through improvements in high order discretization methods. Nevertheless, these approaches are, and will remain in the foreseeable future, far too costly to be standard tools in industry.

However, low fidelity turbulence modeling based on the Reynolds Averaged Navier–Stokes (RANS) equations discretized using second order finite volume discretizations is a good choice for many industrial problems where turbulence matters. For steady flows, this comes down to solving one nonlinear system. In the unsteady case, the time discretization has to be at least partially implicit, due to the extremely fine grids in the boundary layer, requiring solving one or more nonlinear systems per time step. The choice for numerical methods for these comes down to Jacobian-free Newton–Krylov (JFNK) methods with appropriate preconditioners or nonlinear multigrid methods (Full Approximation scheme—FAS) with appropriate smoothers, see [3] for an overview.

In this article, we focus on improving the convergence rate of agglomeration multigrid methods [23], which are the standard in the aeronautical industry. For the type of problems considered here, two aspects have been identified that affect solver efficiency. Firstly, the flow is convection dominated. Secondly the grid has high aspect ratio cells. It is important to note that the viscous terms in the RANS equations do not pose problems in themselves. Instead, the problem is that they cause the boundary layer to appear, thus making high aspect ratio grids necessary. These aspects are shared by the Euler equations, meaning that solvers developed for one equation may also be effective for the other.

With regards to convection dominated flows, smoothers such as Jacobi or Gauß–Seidel do not perform well, in particular when the flow is aligned with the grid [24]. One idea has been to adjust multigrid restriction and prolongation by using directional or semi coarsening that respects the flow direction [25]. This approach has the problem of being significantly more complicated to implement than standard agglomeration. Thus, the alternative is to adjust the smoother. As it turned out, symmetric Gauß–Seidel (SGS) is an excellent smoother for the Euler equations even for grid aligned flow [7], simply because it takes into account propagation of information in the flow direction and backwards.

However, when discretizing the Euler equations on high aspect ratio grids suitable for wall bounded viscous flows, this smoother does not perform well. During the last ten years, the idea of preconditioned pseudo time iterations has garnered interest [5,6,14,17–21,27–29,31,32]. This goes back to the additive Runge–Kutta (ARK) smoothers originally introduced in [8] and independently in a multigrid setting in [11]. These exhibit slow convergence, but if they are combined with a preconditioner, methods result that work well for high Reynolds number high aspect ratio RANS simulations.

The preconditioned RK smoother suggested in [28,32] was rederived in [18] by starting with a Rosenbrock method and then approximating the system matrix in the arising linear systems. This is called a W method in the literature on ordinary differential equations. Consequently, we now identify the two classes of preconditioned additive W methods and preconditioned additive explicit Runge–Kutta methods and derive them from time integration methods. This allows us to identify the roles the preconditioners have to play and aids in choosing good parameters. As for preconditioners itself, it turns out that again, SGS is a very good choice, as reported in [18,32].

The specific convergence rate attainable depends on the discretization, in particular the flux function. Here, we consider the Jameson–Schmidt–Turkel (JST) scheme for structured grids, in its latest version [15]. We perform a discrete Fourier analysis of the smoother for the linearized Euler equations on cartesian grids with variable aspect ratios. This is justified, since the core issues of convection and high aspect ratio grids are present in this problem.

A convenient truth is that if we have a fast steady-state solver then it can be used to build a fast unsteady solver via dual timestepping. However, there are subtle differences that affect convergence and stability. In particular, the eigenvalues of the amplification matrix are scaled and shifted in the unsteady case relative to the steady case. For a fuller discussion of these issues we refer to our earlier work [5,6] and to [2]. We compare the analytical behavior and numerical performance of iterative smoothers for steady and unsteady problems.

The article is structured as follows. We first present the governing equations and the discretization, then we describe multigrid methods and at length the smoothers considered. Then we present a Fourier analysis based on the Euler equations and finally numerical results for airfoil test cases.

2 Discretization

We consider the two dimensional compressible (U)RANS equations, where the vector of conservative variables is $(\rho, \rho v_1, \rho v_2, \rho E)^T$ and the convective and viscous fluxes are given by

$$F_i^c = \begin{pmatrix} \rho v_i \\ \rho v_i v_1 + p\delta_{i1} \\ \rho v_i v_1 + p\delta_{i1} \\ \rho v_i H \end{pmatrix}, \quad F_i^v = \begin{pmatrix} 0 \\ \tau_{i1} \\ \tau_{i2} \\ v_j \tau_{ij} + \frac{\mu+\mu_t}{Pr}(C_p \partial_i T) \end{pmatrix}, \quad i = 1, 2,$$

$$\tau_{ij} = (\mu + \mu_t)(\partial_{x_j} v_i + \partial_{x_i} v_j - \frac{2}{3}\delta_{ij}\partial_{x_k} v_k),$$

$$q_j = \left(\frac{\mu}{Pr} + \frac{\mu_t}{Pr_t}\right)\partial_{x_j}\left(H - \frac{1}{2}v_k v_k\right)$$

where we used the Einstein notation.

Here, ρ is the density, v_i the velocity components and E the total energy per unit mass. The enthalpy is given by $H = E + p/\rho$ with $p = (\gamma - 1)\rho(E - 1/2 v_k v_k)$ being the pressure and $\gamma = 1.4$ the adiabatic index for an ideal gas. Furthermore, τ_{ij} is the sum of the viscous and Reynolds stress tensors, q_j the sum of the diffusive and turbulent heat fluxes, μ the dynamic viscosity, μ_t the turbulent viscosity and Pr, Pr_t the dynamic Prandtl and turbulent Prandtl numbers.

As a turbulence model, we use the 0-equation Baldwin–Lomax model [1] for two reasons. Firstly, it performs well for flows around airfoils we use as primary motivation. Secondly, with 1- or 2-equation models more difficulties in implementation, analysis and convergence behavior arise. We believe that these have to be systematically looked at, but separately from this investigation.

The equations are discretized using a finite volume method on a structured mesh and the JST scheme as flux function. There are many variants of this method, see e.g. [15]. Here, we use the following, for simplicity written as if for a one dimensional problem:

$$\mathbf{f}_{j+1/2}^{JST}(\mathbf{u}) = \frac{1}{2}(\mathbf{f}^R(\bar{\mathbf{u}}_j) + \mathbf{f}^R(\bar{\mathbf{u}}_{j+1})) + \mathbf{d}_{j+1/2}(\mathbf{u}).$$

Here, $\mathbf{f}^R(\bar{\mathbf{u}})$ is the difference of the convective and the viscous fluxes, $\mathbf{u} \in \mathbf{R}^m$ is the vector of all discrete unknowns and $\bar{\mathbf{u}} = (\rho, \rho v, \rho E)$ is the vector of conservative variables. The artificial viscosity terms are given by

$$\mathbf{d}_{j+1/2}(\mathbf{u}) = \epsilon_{j+1/2}^{(2)}\Delta\mathbf{w}_j - \epsilon_{j+1/2}^{(4)}(\Delta\mathbf{w}_{j+1} - 2\Delta\mathbf{w}_j + \Delta\mathbf{w}_{j-1})$$

with Δ_j being the forward difference operator and the vector $\mathbf{w}$ being $\bar{\mathbf{u}}$ where in the last component, the energy density has been replaced by the enthalpy density.

The scalar coefficient functions $\epsilon^{(2)}_{j+1/2}$ and $\epsilon^{(4)}_{j+1/2}$ are given by

$$\epsilon^{(2)}_{j+1/2} = s_{j+1/2} r_{j+1/2} \tag{1}$$

and

$$\epsilon^{(4)}_{j+1/2} = \max(0, r_{j+1/2}/32 - 2\epsilon^{(2)}_{j+1/2}). \tag{2}$$

Here, the entropy sensor $s_{j+1/2} = \min(0.25, \max(s_j, s_{j+1}))$ given via

$$s_j = \left| \frac{S_{j+1} - 2S_j + S_{j-1}}{S_{j+1} + 2S_j + S_{j-1} + 0.001} \right|$$

with $S = p/\rho^\gamma$. For the Euler equations, it is suggested to instead use a corresponding pressure sensor.

Furthermore, $r_{i+1/2}$ is the scalar diffusion coefficient, given by

$$r_{j+1/2} = \max(r_j, r_{j+1}).$$

It approximates the spectral radius and is chosen instead of a matrix valued diffusion as in other versions of this scheme. The specific choice of r_j is important with respect to stability and the convergence speed of the multigrid method. Here, we use the locally largest eigenvalue $r_j = |v_{n_j}| + a_j$ as a basis, where a is the speed of sound. In the multidimensional case, this is further modified to be [22]:

$$\begin{aligned} \tilde{r}_i &= r_i(1 + (r_j/r_i)^{2/3}), \\ \tilde{r}_j &= r_j(1 + (r_i/r_j)^{2/3}), \end{aligned} \tag{3}$$

where r_i corresponds to the x direction and r_j to the y direction.

Additionally, to obtain velocity and temperature gradients needed for the viscous fluxes, we exploit that we have a cell centered method on a structured grid and use dual grids around vertices to avoid checker board effects [13, p. 364].

For boundary conditions, we use the no slip condition at fixed wall and far field conditions at outer boundaries. These are implemented using Riemann invariants [13, p. 362].

In time, we use BDF-2 with a fixed time step Δt, resulting at time t_{n+1} in an equation system of the form

$$\mathbf{F}(\mathbf{u}) := \frac{3\mathbf{u} - 4\mathbf{u}^n + \mathbf{u}^{n-1}}{2\Delta t} + \ddot{}^{-1}\mathbf{f}(\mathbf{u}) = \mathbf{0}. \tag{4}$$

Here, $\mathbf{f}(\mathbf{u})$ describes the spatial discretization, whereas $\ddot{}$ is a diagonal matrix with the volumes of the mesh cells as entries. We thus obtain

$$\frac{\partial \mathbf{F}}{\partial \mathbf{u}} = \frac{3}{2\Delta t}\mathbf{I} + \ddot{}^{-1}\frac{\partial \mathbf{f}}{\partial \mathbf{u}}.$$

For a steady state problem, we just have

$$\mathbf{F}(\mathbf{u}) := \ddot{}^{-1}\mathbf{f}(\mathbf{u}) = \mathbf{0}. \tag{5}$$

3 The Full Approximation Scheme

As mentioned in the introduction, we use an agglomeration FAS to solve Eqs. (4) and (5). To employ a multigrid method, we need a hierarchical sequence of grids with the coarsest grid being denoted by level $l = 0$. This is obtained by agglomerating 4 neighboring cells to one, which is straightforward for structured grids. On the coarse grids, the problem is discretized using a first order version of the JST scheme that does not use fourth differences or an entropy sensor.

The iteration is performed as a W-cycle, where on the coarsest grid, one smoothing step is performed. This gives the following pseudo code:

Function FAS-W-cycle($\mathbf{u}_l, \mathbf{s}_l, l$)

- $\mathbf{u}_l = \mathbf{S}_l^{\nu_1}(\mathbf{u}_l, \mathbf{s}_l)$ (Presmoothing)
- if ($l > 0$)
  - $\mathbf{r}_l = \mathbf{s}_l - \mathbf{F}_l(\mathbf{u}_l)$
  - $\tilde{\mathbf{u}}_{l-1} = \mathbf{R}_{l-1,l}\mathbf{u}_l$ (Restriction of solution)
  - $\mathbf{s}_{l-1} = \mathbf{F}_{l-1}(\tilde{\mathbf{u}}_{l-1}) + \mathbf{R}_{l-1,l}\mathbf{r}_l$ (Restriction of residual)
  - For $j = 1, 2$: call FAS-W-cycle($\mathbf{u}_{l-1}, \mathbf{s}_{l-1}, l-1$) (Computation of the coarse grid correction)
  - $\mathbf{u}_l = \mathbf{u}_l + \mathbf{P}_{l,l-1}(\mathbf{u}_{l-1} - \tilde{\mathbf{u}}_{l-1})$ (Correction via Prolongation)
- end if

The restriction $\mathbf{R}_{l-1,l}$ is an agglomeration that weighs components by the volume of their cells and divides by the total volume. As for the prolongation $\mathbf{P}_{l,l-1}$, it uses a bilinear weighting [12].

On the finest level, the smoother is applied to the Eq. (4) resp. (5). On sublevels, it is instead used to solve

$$\mathbf{F} := \mathbf{s}_l - \mathbf{F}_l(\mathbf{u}_l) = \mathbf{0} \tag{6}$$

with

$$\mathbf{s}_l = \mathbf{F}_l(\mathbf{R}_{l,l+1}\mathbf{u}_{l+1}) + \mathbf{R}_{l,l+1}\mathbf{r}_{l+1}.$$

4 Preconditioned Smoothers

All smoothers we use have a pseudo time iteration as a basis. These are iterative methods for the nonlinear equation $\mathbf{F}(\mathbf{u}) = \mathbf{0}$ that are obtained by applying a time integration method to the initial value problem

$$\mathbf{u}_{t^*} = -\mathbf{F}(\mathbf{u}), \quad \mathbf{u}(0) = \mathbf{u}^0.$$

For convenience, we have dropped the subscript l that denotes the multigrid level.

4.1 Preconditioned Additive Runge–Kutta Methods

We start with splitting $\mathbf{F}(\mathbf{u})$ in a convective and diffusive part

$$\mathbf{F}(\mathbf{u}) = \mathbf{f}^c(\mathbf{u}) + \mathbf{f}^v(\mathbf{u}). \tag{7}$$

Hereby, $\mathbf{f}^c$ contains the physical convective fluxes, as well as the discretized time derivative and the multigrid source terms, whereas $\mathbf{f}^v$ contains both the artificial dissipation and the discretized second order terms of the Navier–Stokes equations.

An additive explicit Runge–Kutta (AERK) method is then implemented in the following form:

$$\mathbf{u}^{(0)} = \mathbf{u} \tag{8}$$

$$\mathbf{u}^{(i)} = \mathbf{u} - \alpha_i \Delta t^* (\mathbf{f}^{c,(i-1)} + \mathbf{f}^{v,(i-1)}), \quad i = 1, \ldots, s \tag{9}$$

$$\mathbf{u}^{n+1} = \mathbf{u}^{(s)}, \tag{10}$$

where

$$\mathbf{f}^{c,(i)} = \mathbf{f}^c(\mathbf{u}^{(i)}), \quad i = 0, \ldots, s-1 \tag{11}$$

$$\mathbf{f}^{v,(0)} = \mathbf{f}^v(\mathbf{u}^{(0)}), \tag{12}$$

$$\mathbf{f}^{v,(i)} = \beta_{i+1}\mathbf{f}^v(\mathbf{u}^{(i)}) + (1-\beta_{i+1})\mathbf{f}^{v,(i-1)}, \quad i = 1, \ldots, s-1. \tag{13}$$

The second to last line implies that $\beta_1 = 1$. Here, Δt^* is a local pseudo time step, meaning that it depends on the specific cell and the multigrid level. It is obtained by choosing c^*, a CFL number in pseudo time, and then computing Δt^* based on the local mesh width Δx_{k_l}. On an equidistant mesh, this comes down to:

$$\Delta t^* = c^* \Delta x_{k_l} / \left(|\mathbf{v}_k| + a_k + \frac{16}{\Delta x_{k_l}} \frac{\sqrt{\gamma}}{\rho_k} \frac{Ma}{Re} \left[\gamma \left(\frac{\mu_k}{Pr} + \frac{\mu_t}{Pr_{t_k}} \right) + \frac{\mu_k + \mu_{t_k}}{\sqrt{6}} \right] \right).$$

This implies larger time steps on coarser cells, in particular on coarser grids.

As for the coefficients, several schemes have been designed to have good smoothing properties in a multigrid method for convection dominated model equations. The 3-stage scheme AERK3 has its origins in [35], with the β coefficients being derived in [30]. AERK5J was designed by Jameson using linear advection with a fourth order diffusion term, see [11]. The 5-stage schemes AERK51 and AERK52 are from [35]. AERK52 is employed in [32]. Coefficients for the 3- and 5-stage schemes can be found in Table 1. All of these schemes are first order, except for the last one, which has order two and is therefore denoted as AERK52. In the original publication AERK51 and AERK52 are not additive. When using these within an additive method, we use the β coefficients from AERK5J. For current research into improving these coefficients we refer to [2,4].

Setting $\beta_i = 1$ for all i gives an unsplit low storage explicit Runge–Kutta method that does not treat convection and diffusion differently. We refer to these schemes as ERK methods, e.g. ERK3J or ERK51.

Table 1 Coefficients of explicit and additive explicit Runge–Kutta smoothers, 3- and 5-stage method

	i	1	2	3	4	5
AERK3	α_i	0.1481	2/5	1	–	–
	β_i	1	1/2	1/2	–	–
AERK5J	α_i	1/4	1/6	3/8	1/2	1
	β_i	1	0	0.56	0	0.44
AERK51	α_i	0.0533	0.1263	0.2375	0.4414	1.
AERK52	α_i	0.0695	0.1602	0.2898	0.5060	1.

To precondition this scheme, a preconditioner $\mathbf{P}^{-1} \in \mathbf{R}^{m\times m}$ is applied to the equation system (4) or (5) by multiplying them with it, resulting in an equation

$$\mathbf{P}^{-1}\mathbf{F}(\mathbf{u}) = \mathbf{0}.$$

In a pseudo-time iteration for the new equation, all function evaluations have to be adjusted. In the above algorithm, this is realized by replacing the term $\alpha_i \Delta t^*(\mathbf{f}^{c,(i-1)} + \mathbf{f}^{v,(i-1)})$ with $\alpha_i \Delta t^* \mathbf{P}^{-1}(\mathbf{f}^{c,(i-1)} + \mathbf{f}^{v,(i-1)})$. We discuss the role of the preconditioner in more detail in Sect. 4.3.

4.2 Additive W-Methods

In [28,32] so called RK/implicit methods were suggested with great success. These use in effect the method from above with an approximation of the matrix $\mathbf{I} + \alpha \frac{\partial \mathbf{F}}{\partial \mathbf{u}}$ instead, where α is a parameter. To bring these things together, an alternative way of deriving these methods has been presented by Langer in [17]. He uses the term preconditioned implicit smoothers and derives them from specific singly diagonally implicit RK (SDIRK) methods. SDIRK methods consist of a nonlinear system at each step, which he solves with one Newton step each and then simplifies by always using the Jacobian from the first stage. This is known as a special choice of a Rosenbrock method in the literature on differential equations [10, p. 102]. To arrive at a preconditioned method, Langer then replaces the system matrix with an approximation, for example originating from a preconditioner as known from linear algebra. In fact, this type of method is called a W-method in the ODE community [10, p. 114].

We now extend the framework from [17] to additive Runge–Kutta methods. For clarity we repeat the derivation, but start from the split equation

$$\mathbf{u}_{t^*} + \mathbf{f}^c(\mathbf{u}) + \mathbf{f}^v(\mathbf{u}) = \mathbf{0} \tag{14}$$

as described in (7). To this equation, we apply an additive SDIRK method with coefficients given in Tables 2 and 3:

$$\mathbf{k}_i = -\mathbf{F}\left(\mathbf{u}^n + \Delta t^* \left(\sum_{j=1}^{i-1} (a_{ij}^c \mathbf{k}_j^c + a_{ij}^v \mathbf{k}_j^v) + \eta \mathbf{k}_i\right)\right), \quad i = 1, \ldots, s, \tag{15}$$

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \Delta t^* (\alpha_s \mathbf{k}_s^c + \alpha_s (1 - \beta_{s-1}) \mathbf{k}_{s-1}^v + \alpha_s \beta_s \mathbf{k}_s^v). \tag{16}$$

Hereby, the vectors $\mathbf{k}$ are called stage derivatives and we have $\mathbf{k} = \mathbf{k}^c + \mathbf{k}^v$ according to the splitting (7). Thus, we have to solve s nonlinear equation systems for the stage derivatives $\mathbf{k}_i$.

Table 2 Butcher arrays for additive SDIRK method: convective terms

	η	0	0	0
	α_1	η	0	0
	0	$\ddots$	$\ddots$	0
	0	0	α_{s-1}	η
	0	...	0	α_s

Table 3 Butcher arrays for additive SDIRK method: diffusive terms

η		$\cdots$		0
α_1	η			0
$\alpha_2(1-\beta_1)$	$\alpha_2\beta_2$	$\ddots$		0
0	$\ddots$	$\ddots$	η	0
0	$\cdots$	$\alpha_{s-1}(1-\beta_{s-1})$	$\alpha_{s-1}\beta_s$	η
0	$\cdots$	0	$\alpha_s(1-\beta_{s-1})$	$\alpha_s\beta_s$

To obtain an additive Rosenbrock method, these are solved approximately using one Newton step each with initial guess zero, changing the stage values to

$$\mathbf{k}_i = -(\mathbf{I}+\eta\Delta t^*\mathbf{J}_i)^{-1}\mathbf{F}(\mathbf{u}^n + \Delta t^*\left(\sum_{j=1}^{i-1}(a_{ij}^c\mathbf{k}_j^c + a_{ij}^v\mathbf{k}_j^v)\right), \quad i=1,\ldots,s, \tag{17}$$

where $\mathbf{J}_i = \frac{\partial \mathbf{F}_i(\mathbf{0})}{\partial \mathbf{k}}$, with $\mathbf{F}_i(\mathbf{k}) := \mathbf{F}(\mathbf{u}^n + \Delta t^*\left(\sum_{j=1}^{i-1}(a_{ij}^c\mathbf{k}_j^c + a_{ij}^v\mathbf{k}_j^v) + \eta\mathbf{k}\right)$. Thus, we now have to solve a linear system at each stage. This type of scheme is employed in Swanson et. al. [32]. They refer to the factor η as ϵ and provide a discrete Fourier analysis of this factor.

As a final step, we approximate the system matrices $\mathbf{I}+\eta\Delta t^*\mathbf{J}_i$ by a matrix $\mathbf{W}$. This gives us a new class of schemes, which we call additive W (AW) methods, with stage derivatives given by:

$$\mathbf{k}_i = -\mathbf{W}^{-1}\mathbf{F}(\mathbf{u}^n + \Delta t^*\left(\sum_{j=1}^{i-1}(a_{ij}^c\mathbf{k}_j^c + a_{ij}^v\mathbf{k}_j^v)\right), \quad i=1,\ldots,s, \tag{18}$$

In both additive Rosenbrock and additive W methods, Eq. (16) remains unchanged.

Finally, after some algebraic manipulations, this method can be rewritten in the same form as the low storage preconditioned AERK methods presented earlier:

$$\begin{aligned}
\mathbf{u}^{(0)} &= \mathbf{u}^n \\
\mathbf{u}^{(i)} &= \mathbf{u}^n - \alpha_i\Delta t^*\mathbf{W}^{-1}(\mathbf{f}^{c,(i-1)} + \mathbf{f}^{v,(i-1)}), \quad i=1,\ldots,s \\
\mathbf{u}^{n+1} &= \mathbf{u}^{(s)},
\end{aligned}$$

where

$$\begin{aligned}
\mathbf{f}^{c,(i)} &= \mathbf{f}^c(\mathbf{u}^{(i)}), \quad i=0,\ldots,s-1 \\
\mathbf{f}^{v,(0)} &= \mathbf{f}^v(\mathbf{u}^{(0)}), \\
\mathbf{f}^{v,(i)} &= \beta_{i+1}\mathbf{f}^v(\mathbf{u}^{(i)}) + (1-\beta_{i+1})\mathbf{f}^{v,(i-1)}, \quad i=1,\ldots,s-1.
\end{aligned}$$

As for the explicit methods, one recovers an unsplit scheme for $\beta_i = 1$ for all i and we refer to these methods as SDIRK, Rosenbrock and W methods.

4.3 Comparison

To get a better understanding for the different methods, it is illustrative to consider the linear case. Then, these methods are iterative schemes to solve a linear equation system $(\mathbf{A}+\mathbf{B})\mathbf{x} = \mathbf{b}$ and can be written as

$$\mathbf{x}^{k+1} = \mathbf{M}\mathbf{x}^k + \mathbf{N}\mathbf{b}.$$

The matrix $\mathbf{M}$ is the iteration matrix and for pseudo time iterations, it is given as the stability function S of the time integration method. These are a polynomial P_s of degree s in $\Delta t^*(\mathbf{A}+\mathbf{B})$ for an s stage ERK method and a bivariate polynomial P_s of degree s in $\Delta t^*\mathbf{A}$ and $\Delta t^*\mathbf{B}$ for an s stage AERK method. When preconditioning is added, this results in $P_s(\Delta t^*\mathbf{P}^{-1}(\mathbf{A}+\mathbf{B}))$ and $P_s(\Delta t^*\mathbf{P}^{-1}\mathbf{A}, \Delta t^*\mathbf{P}^{-1}\mathbf{B})$, respectively.

For the implicit schemes, we obtain a rational function of the form $Q_s(\mathbf{I} + \eta\Delta t^*(\mathbf{A} + \mathbf{B}))^{-1} P_s(\Delta t^*(\mathbf{A} + \mathbf{B}))$ for an s stage SDIRK or Rosenbrock method. Here, Q_s is a second polynomial of degree s. Finally, for an s-stage additive W method, we obtain a function of the form $Q_s(\mathbf{W})^{-1} P_s(\Delta t^*\mathbf{A}, \Delta t^*\mathbf{B})$. Due to the specific contruction, the inverse can simply be moved from the left into the argument which gives $P_s(\Delta t^*\mathbf{W}^{-1}\mathbf{A}, \Delta t^*\mathbf{W}^{-1}\mathbf{B})$. Note that this is the same as for the preconditioned AERK method, except for the preconditioner.

The additive W method and the preconditioned AERK method have three main differences. First of all, there is the role of $\mathbf{P}$ in the AERK method versus the matrix $\mathbf{W}$. In the W method $\mathbf{W} \approx (\mathbf{I} + \eta\Delta t^*\mathbf{J}_i)$, whereas in the AERK scheme, $\mathbf{P} \approx \mathbf{J}_i$. Second, the timestep in the one case is that of an explicit ARK method, whereas in the other, that of an implicit method. The latter in its SDIRK or Rosenbrock form is A-stable. However, approximating the Jacobian can cause the stability region to become finite. Finally, the latter method has an additional parameter η that needs to be chosen. However, the large stability region makes the choice of Δt^* easy for the additive W method (very large), whereas it has to be a small value for the preconditioned AERK scheme.

4.4 SGS Preconditioner

The basis of our method is the preconditioner suggested by Swanson et al. in [32] and modified by Jameson in [16]. In effect, this is a choice of a $\mathbf{W}$ matrix in the framework just presented. We now repeat the derivation of this preconditioner in our notation.

The first step is to approximate the Jacobian by using a different first order linearized discretization. It is based on a splitting $\mathbf{A} = \mathbf{A}^+ + \mathbf{A}^-$ of the flux Jacobian. This is evaluated in the average of the values on both sides of the interface, thereby deviating from [32]. The split Jacobians correspond to positive and negative eigenvalues:

$$\mathbf{A}^+ = \frac{1}{2}(\mathbf{A} + |\mathbf{A}|), \quad \mathbf{A}^- = \frac{1}{2}(\mathbf{A} - |\mathbf{A}|).$$

Alternatively, these can be written in terms of the matrix of right eigenvectors $\mathbf{R}$ as

$$\mathbf{A}^+ = \mathbf{R}|\Lambda^+|\mathbf{R}^{-1}, \quad \mathbf{A}^- = \mathbf{R}|\Lambda^-|\mathbf{R}^{-1},$$

where $\Lambda^{\pm}$ are diagonal matrices containing the positive and negative eigenvalues, respectively.

As noted in [16,33], it is now crucial to use a cutoff function for the eigenvalues beforehand, to bound them away from zero. We use a parabolic function which kicks in when the modulus of the eigenvalue λ is smaller or equal to a fraction ad of the speed of sound a with free parameter $d \in [0, 1]$:

$$|\lambda| = \frac{1}{2}\left(ad + \frac{|\lambda|^2}{ad}\right), \quad |\lambda| \leq ad. \tag{19}$$

With this, an upwind discretization is given in cell i by

$$\mathbf{u}_{i_t} = \frac{1}{\Omega_i} \sum_{e_{ij} \in N(i)} |e_{ij}| (\mathbf{A}^+_{\mathbf{n}_{ij}} \mathbf{u}_i + \mathbf{A}^-_{\mathbf{n}_{ij}} \mathbf{u}_j). \tag{20}$$

Here, e_{ij} is the edge between cells i and j, $N(i)$ is the set of cells neighboring i and $\mathbf{n}_{ij}$ the unit normal vector from i to j.

For the unsteady equation (4), we obtain instead

$$\mathbf{u}_{i_t} = \frac{3}{2\Delta t^*}\mathbf{I} + \frac{1}{\Omega_i} \sum_{e_{ij} \in N(i)} |e_{ij}| (\mathbf{A}^+ \mathbf{u}_i + \mathbf{A}^- \mathbf{u}_j). \tag{21}$$

The corresponding approximation of the Jacobian is then used to construct a preconditioner. Specifically, we consider the block SGS preconditioner

$$\mathbf{P}^{-1} = (\mathbf{D} + \mathbf{L})^{-1} \mathbf{D} (\mathbf{D} + \mathbf{U})^{-1}, \tag{22}$$

where $\mathbf{L}$, $\mathbf{D}$ and $\mathbf{U}$ are block matrices with 4×4 blocks. This preconditioner would look different when several SGS steps would be performed. However, a second step increases the cost of the whole method by 50% without giving an appropriate increase in convergence rate. Therefore, it should only be used if one step does not give convergence.

We now have two cases. In the AERK framework, $\mathbf{L} + \mathbf{D} + \mathbf{U} = \mathbf{J}$ and we arrive at

$$\mathbf{L}_{ij} = -\frac{1}{\Omega_i}(\Delta y \mathbf{A}^+_{i-1,j} + \Delta x \mathbf{B}^+_{i,j-1}), \tag{23}$$

$$\mathbf{U}_{ij} = \frac{1}{\Omega_i}(\Delta y \mathbf{A}^-_{i-1,j} + \Delta x \mathbf{B}^-_{i,j-1}), \tag{24}$$

$$\mathbf{D}_{ii} = \frac{1}{\Omega_i}[\Delta y(\mathbf{A}^+_{ii} - \mathbf{A}^-_{ii}) + \Delta x(\mathbf{B}^+_{ii} - \mathbf{B}^-_{ii})], \tag{25}$$

respectively

$$\mathbf{D}_{ii} = \frac{3}{2\Delta t}\mathbf{I} + \frac{1}{\Omega_i}[\Delta y(\mathbf{A}^+_{ii} - \mathbf{A}^-_{ii}) + \Delta x(\mathbf{B}^+_{ii} - \mathbf{B}^-_{ii})], \tag{26}$$

in the unsteady case and we assumed a cartesian grid to simplify notation.

In the additive W framework, $\mathbf{L} + \mathbf{D} + \mathbf{U} = \mathbf{I} + \eta \Delta t^* \mathbf{J}$ and we obtain

$$\mathbf{L}_{ij} = -\frac{\eta \Delta t^*_i}{\Omega_i}(\Delta y \mathbf{A}^+_{i-1,j} + \Delta x \mathbf{B}^+_{i,j-1}), \tag{27}$$

$$\mathbf{U}_{ij} = \frac{\eta \Delta t^*_i}{\Omega_i}(\Delta y \mathbf{A}^-_{i-1,j} + \Delta x \mathbf{B}^-_{i,j-1}), \tag{28}$$

$$\mathbf{D}_{ii} = \mathbf{I} + \frac{\eta \Delta t^*_i}{\Omega_i}\left[\Delta y(\mathbf{A}^+_{ii} - \mathbf{A}^-_{ii}) + \Delta x(\mathbf{B}^+_{ii} - \mathbf{B}^-_{ii})\right]. \tag{29}$$

or in the unsteady case

$$\mathbf{D}_{ii} = \mathbf{I} + \frac{3\eta \Delta t^*}{2\Delta t}\mathbf{I} + \frac{\eta \Delta t^*_i}{\Omega_i}[\Delta y(\mathbf{A}^+_{ii} - \mathbf{A}^-_{ii}) + \Delta x(\mathbf{B}^+_{ii} - \mathbf{B}^-_{ii})]. \tag{30}$$

Applying this preconditioner requires solving small 4×4 systems coming from the diagonal. We use Gaussian elimination for this. A fast implementation is obtained by transforming first to a certain set of symmetrizing variables, see [16].

5 Discrete Fourier Analysis

We now perform a discrete Fourier analysis of the preconditioned AERK method for the two dimensional Euler equations using the JST scheme. For a description of this technique, also called local Fourier analysis (LFA) in the multigrid community, we refer to [9,34]. The rationale for this is that the core convergence problems for multigrid methods for viscous flow problems on high aspect ratio grids are the convective terms and the high aspect ratio grids. The viscous terms are of comparatively minor importance. Here, we do not take into account the coarse grid correction. Thus, our aim is to obtain amplification- and smoothing factors for the smoother. The latter is given by

$$\max_{\lambda_{HF}} |S(\lambda)|, \tag{31}$$

where λ_{HF} denote the high frequency eigenvalues. Since eigenfunctions of first order hyperbolic differential operators involve $e^{i\phi x}$, these are in $[-\pi, -\pi/2]$ and $[\pi/2, \pi]$.

We now consider a linearized version of the underlying equation with periodic boundary conditions on the domain $\Omega = [0, 1]^2$:

$$\frac{d}{dt}\mathbf{u} + (\mathbf{A}\mathbf{u})_x + (\mathbf{B}\mathbf{u})_y = \mathbf{0} \tag{32}$$

with $\mathbf{A} = \frac{\partial \mathbf{f}_1}{\partial \mathbf{u}}$ and $\mathbf{B} = \frac{\partial \mathbf{f}_2}{\partial \mathbf{u}}$ being the Jacobians of the Euler fluxes in a fixed point $\hat{\mathbf{u}}$, to be set later.

5.1 JST Scheme

We discretize (32) on a cartesian mesh with mesh width Δx in x-direction and $\Delta y = \mathrm{AR}\Delta x$ in y-direction (AR = aspect ratio), resulting in an $n_x \times n_y$ mesh. A cell centered finite volume method with the JST flux is employed. We denote the shift operators in x and y direction by E_x and E_y. Cells are indexed the canonical doubly lexicographical way for a cartesian mesh. In cell ij we write the discretization as

$$(\mathbf{H}\mathbf{u})_{ij} = ((\mathbf{H}_c + \mathbf{H}_v)\mathbf{u})_{ij}$$

with

$$\mathbf{H}_c = \frac{1}{2\Delta x \Delta y}(\mathbf{A}(E_x^{+1} - E_x^{-1})\Delta y + \mathbf{B}(E_y^{+1} - E_y^{-1})\Delta x),$$

respectively in the unsteady case,

$$\mathbf{H}_c = \frac{3}{2\Delta t}\mathbf{I} + \frac{1}{2\Delta x \Delta y}(\mathbf{A}(E_x^{+1} - E_x^{-1})\Delta y + \mathbf{B}(E_y^{+1} - E_y^{-1})\Delta x).$$

For $\mathbf{H}_v$, the starting point is that the pressure in conservative variables is

$$p = (\gamma - 1)\left(\rho E - \rho\frac{(\rho v_1)^2 + (\rho v_2)^2}{2\rho^2}\right).$$

In the fraction, all potential shift operators cancel out. Thus, for the second differences in both directions,

$$p_{j+1} - 2p_j + p_{j-1} = (\gamma - 1)[(E^+ - 2 + E^-)\rho E_j - |\mathbf{v}|^2/2(E^+ - 2 + E^-)\rho_j].$$

For the fourth difference, there's a corresponding identity. Furthermore, applying the second or fourth difference to $\rho H_j = \rho E_j + p_j$ results in

$$\rho H_{j+1} - 2\rho H_j + \rho H_{j-1} = \gamma(E^+ - 2 + E^-)\rho E_j - (\gamma - 1)|\mathbf{v}|^2/2(E^+ - 2 + E^-)\rho_j.$$

This gives

$$\begin{aligned}\mathbf{H}_v = \frac{1}{\Delta x \Delta y}\mathbf{M}[&\epsilon^{(2)}((-E_x^{+1} + 2 - E_x^{-1})\Delta y + (-E_y^{+1} + 2 - E_y^{-1})\Delta x)\\ &+\epsilon^{(4)}(E_x^{+2} - 4E_x^{+1} + 6 - 4E_x^{-1} + E_x^{-2})\Delta y\\ &+\epsilon^{(4)}(E_y^{+2} - 4E_y^{+1} + 6 - 4E_y^{-1} + E_y^{-2})\Delta x]\end{aligned}$$

with

$$\mathbf{M} = \begin{pmatrix} 1 & 0\,0\,0 \\ 0 & 1\,0\,0 \\ 0 & 0\,1\,0 \\ -(\gamma-1)|\mathbf{v}|^2/2 & 0\,0\,\gamma \end{pmatrix}.$$

For the coefficient functions $\epsilon^{(2)}$ and $\epsilon^{(4)}$ [see (1) and (2)], we first look at the shock sensor $s_{j+1/2}$. Here, we use the version for the Euler equations based on pressure. Straightforward calculations give

$$p_{j+1} + 2p_j + p_{j-1} = (\gamma - 1)[(E^{+1} + 2 + E^{-1})\rho E_j - 1/2|\mathbf{v}|^2(E^+ + 2 + E^-)\rho_j].$$

Thus

$$s_j = \frac{(\gamma - 1)[(E^{+1} - 2 + E^{-1})\rho E_j - 1/2|\mathbf{v}|^2(E^+ - 2 + E^-)\rho_j]}{(\gamma - 1)[(E^{+1} + 2 + E^{-1})\rho E_j - 1/2|\mathbf{v}|^2(E^+ + 2 + E^-)\rho_j] + 0.001}.$$

For simplicity, we now assume that $\max(s_j, s_{j+1}) = s_j$. Thus,

$$s_{j+1/2} = \min(0.25, s_j).$$

For the spectral radius we note that in the speed of sound $a_j = \sqrt{\gamma p_j/\rho_j}$, possible shift operators cancel out as well, implying that is constant over the mesh. This gives

$$\begin{aligned} r_i &= |v_1| + a, \\ r_j &= |v_2| + a. \end{aligned}$$

Regarding the maxima, we have $r_j = r_{j+1} =: r$ and correspondingly for the y direction with r_i. Thus,

$$\epsilon^{(2)} = r s_{j+1/2}$$

and

$$\epsilon^{(4)} = \max(0, r/32 - 2\epsilon^{(2)}).$$

5.2 Preconditioner

With regards to the SGS preconditioner $\mathbf{P}^{-1} = (\mathbf{D} + \mathbf{L})^{-1}\mathbf{D}(\mathbf{D} + \mathbf{U})^{-1}$, the different discretization based on the flux splitting (20) with cutoff function (19) gives [see (23)–(26)]

$$\mathbf{L} = -\frac{1}{\Omega}[\Delta y \mathbf{A}^+ E_x^{-1} + \Delta x \mathbf{B}^+ E_y^{-1}],$$

$$\mathbf{U} = \frac{1}{\Omega}[\Delta y \mathbf{A}^- E_x^{+1} + \Delta x \mathbf{B}^- E_y^{+1}].$$

We now get two different operators for the diagonal part for the steady and for the unsteady case. We have

$$\mathbf{D}^s = \mathbf{I} + \frac{1}{\Omega}[\Delta y(\mathbf{A}^+ - \mathbf{A}^-) + (\Delta x(\mathbf{B}^+ - \mathbf{B}^-)],$$

for the steady case, whereas for the unsteady case there is

$$\mathbf{D}^u = \frac{3}{2\Delta t}\mathbf{I} + \mathbf{D}^s.$$

With these, the preconditioner (22) is formed. For the W methods, these matrices need to be adjusted slightly, compare (27)–(30).

We now make one simplification in the analysis and that is that we assume the matrices to be evaluated with the value of the respective cell and not the average as in the actual method.

As an example, the application of the 3-stage AERK scheme results in the following operator, where we write $\bar{\mathbf{H}}_c := \mathbf{P}^{-1}\mathbf{H}_c$, $\bar{\mathbf{H}}_v := \mathbf{P}^{-1}\mathbf{H}_v$ and $\bar{\alpha}_i = \Delta t^* \alpha_i$:

$$\begin{aligned}\mathbf{G} = \mathbf{I} &- \bar{\alpha}_3((\bar{\mathbf{H}}_c + \beta_3\bar{\mathbf{H}}_v)(\mathbf{I} - \bar{\alpha}_2((\bar{\mathbf{H}}_c + \beta_2\bar{\mathbf{H}}_v)(\mathbf{I} - \bar{\alpha}_1(\bar{\mathbf{H}}_c + \bar{\mathbf{H}}_v)) + (1-\beta_2)\bar{\mathbf{H}}_v)) \\ &+ (1-\beta_3)(\beta_2\bar{\mathbf{H}}_v(I - \bar{\alpha}_1(\bar{\mathbf{H}}_c + \bar{\mathbf{H}}_v)) + (1-\beta_2)\bar{\mathbf{H}}_v)).\end{aligned}$$

For other smoothers, we have to use other appropriate stability functions, as discussed in Sect. 4.3. As a relation between Δt^* and c^*, we use the relation

$$\Delta t^* = c^* \min(\Delta x, \Delta y)/(|\mathbf{v}| + a).$$

5.3 Amplification and Smoothing Factors

We are now interested in the amplification factor of the corresponding method for different values of Δx and Δy. Working with $\mathbf{G}$ directly would require assembling a large matrix in $\mathbf{R}^{4n_x \times 4n_y}$. Instead, we perform a discrete Fourier transform. In Fourier space, the transformed operator block diagonalizes, allowing to work with the much smaller matrix $\hat{\mathbf{G}} \in \mathbf{R}^{4\times 4}$. Thus, we replace $\mathbf{u}_{ij}$ by its discrete Fourier series

$$\mathbf{u}_{ij} = \sum_{k_x=-n_x/2+1}^{n_x/2} \sum_{k_y=-n_y/2+1}^{n_y/2} \hat{\mathbf{u}}_{k_x,k_y} e^{2\pi i(k_x x_i + k_y y_j)}$$

and analyze

$$\hat{\mathbf{u}}_{k_x,k_y}^{k+1} = \hat{\mathbf{G}}_{k_x,k_y}\hat{\mathbf{u}}_{k_x,k_y}^{k}.$$

When applying a shift operator to one of the exponentials, we obtain

$$E_x e^{2\pi i(k_x x_i + k_y y_j)} = e^{2\pi i(k_x(x_i+1/n_x)+k_y y_j)} = e^{2\pi i k_x/n_x} e^{2\pi i(k_x x_i + k_y y_j)}$$

and similar for E_y. Defining the phase angles

$$\Theta_x = 2\pi k_x/n_x, \quad \Theta_y = 2\pi k_y/n_y,$$

the Fourier transformed shift operators are

$$\hat{E}_x = e^{i\Theta_x}, \quad \hat{E}_y = e^{i\Theta_y}$$

and can replace the dependence on the wave numbers with a dependence on phase angles.

To compute the spectral radius of $\mathbf{G}$, we now just need to look at the maximum of the spectral radius of $\hat{\mathbf{G}}_{\Theta_x,\Theta_y} = \hat{\mathbf{G}}_{k_x,k_y}$ over all phase angles Θ_x and Θ_y between $-\pi$ and π. Furthermore, this allows to compute the smoothing factor (31) as well, by instead taking the maximum over all wave numbers between $-\pi$ and $-\pi/2$, as well as $\pi/2$ and π.

5.4 Results

We evaluate the matrices in the points

$$\hat{\mathbf{u}}_1 = (1, \sqrt{2}/2, \sqrt{2}/2, 3.290)^T \quad (\text{Mach } 0.8, \alpha = 45°)$$

and

$$\hat{\mathbf{u}}_3 = (1, 1, 0, 3.290)^T \quad (\text{Mach } 0.8, \alpha = 0°).$$

We use a $8 \times (8 \cdot AR)$ grid with different aspect ratios (AR), namely AR $= 1$, AR $= 100$ and AR $= 10{,}000$. To determine the physical time step, a CFL number c of 200 is chosen. All results were obtained using a python script, which can be accessed at http://www.maths.lu.se/philipp-birken/rksgs_fourier.zip.

5.4.1 The Explicit Schemes

Results for explicit schemes for different test cases are shown in Table 4. As can be seen, these methods have terrible convergence rates, but are good smoothers for equidistant meshes. For non-equidistant meshes, this is not the case, which demonstrates the poor performance of these methods for viscous flow problems.

5.4.2 Preconditioned AERK

We now consider preconditioned AERK3J with SGS and exact preconditioning. The Mach number is set to 0.8 and the angle of attack to zero degrees, which is the most difficult test case of the ones considered. Even so, it is possible to achieve convergence at all aspect ratios with a large physical CFL $c = 200$. With regards to stability, we show the maximal possible c^* in Table 5. We can see that this is dramatically improved compared to the unpreconditioned method, but it remains finite, as predicted by the theory. We furthermore notice that the choice of d in the cutoff function (19) is important. In particular, the smaller we choose d, meaning the smaller we allow eigenvalues to be, the less stable the method will be. Maximal c^* is approximately proportional to the aspect ratio and to d. The eigenvalues and contours of smoothing factor for $d = 0.5$ are also illustrated in Fig. 1 for aspect ratios 1 and 100, respectively. Clustering of the eigenvalues along the real axis is observed indicating good convergence.

Table 4 Amplification and smoothing factors of ERK3 and AERK3J, $8 \times (8AR)$ grid, $c^* = 0.9$, $d = 0.1$, $M = 0.8$, $\alpha = 0°$

	ERK3			AERK3J		
AR	1	100	10,000	1	100	10,000
$\rho(\mathbf{M})$	0.9933	0.9937	0.9937	0.9933	0.9937	0.9937
Sm. fct.	0.5158	0.9937	0.9937	0.4634	0.9937	0.9937

Table 5 Maximal c^* for AERK3J, $c = 200$, $M = 0.8$, $\alpha = 0^\circ$ for aspect ratios of 1, 100 and 10000 and various values of the cutoff fraction d from (19)

	SGS precond.			Exact precond.		
d AR	1	100	10,000	1	100	10,000
0.0	1	80	10,500	1	80	8500
0.1	6	900	95,000	8	850	80,000
0.25	15	2200	220,000	16	2000	200,000
0.5	30	4400	440,000	27	4000	400,000
1.0	50	6500	800,000	59	6800	790,000

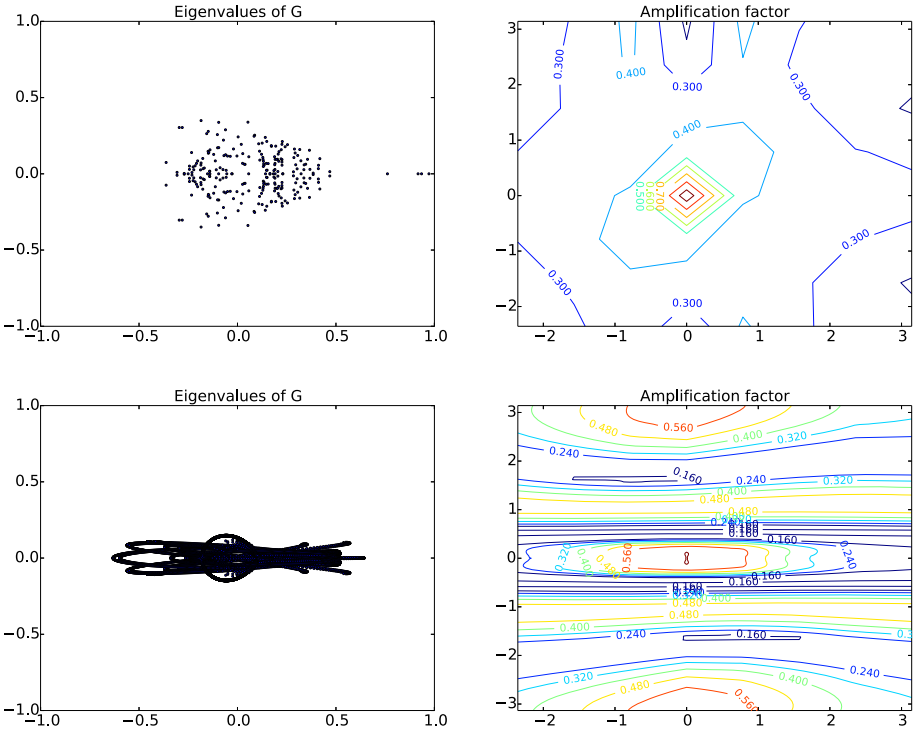

Fig. 1 Spectrum and amplification factors for different wavenumbers, Mach 0.8, $\alpha = 0^\circ$, AERK3J with SGS preconditioner, $c = 200$, $d = 0.5$. Top: AR $= 1$, $c^* = 14$; bottom: AR $= 100$, $c^* = 4300$

For each value of d considered, c^* was optimised (c^* opt) to minimise the smoothing factor (SM fct. opt). The results are shown in Table 6. Optimal smoothing factors improve as d is increased. Preconditioning with the exact inverse affords better smoothing factors than SGS preconditioning. In general, optimal c^* is close to maximal c^*.

5.4.3 Additive W Methods

Results for AW3 with SGS and exact preconditioning are shown in Tables 7 and 8. Again, the Mach number is set to 0.8, the physical CFL $c = 200$ and the angle of attack to zero degrees. We set $\eta = 0.8$. An A means that no bound on c^* was observed. As we can see, as long as d is chosen sufficiently large, the methods are practically A-stable, as suggested by the theory.

Table 6 Amplification and smoothing factors of AERK3J, $c = 200$, $M = 0.8$, $\alpha = 0°$

		SGS precond.			Exact precond.		
d	AR	1	100	10,000	1	100	10,000
0.0	c^* opt	1	80	10,500	1	80	10,500
	$\rho(\mathbf{M})$ opt	0.9974	0.9708	0.9465	0.9489	0.9591	0.9465
	Sm. fct. opt	0.9439	0.9708	0.9465	0.9446	0.9591	0.9465
0.1	c^* opt	5	900	95,000	8	850	85,000
	$\rho(\mathbf{M})$ opt	0.9878	0.7730	0.7739	0.6424	0.6300	0.6300
	Sm. fct. opt	0.7370	0.7730	0.7739	0.6304	0.6300	0.6300
0.25	c^* opt	10	2100	220,000	9	1900	190,000
	$\rho(\mathbf{M})$ opt	0.9774	0.6932	0.6663	0.6051	0.5273	0.4723
	Sm. fct. opt	0.5480	0.6932	0.6663	0.5738	0.5273	0.4723
0.5	c^* opt	13	4300	440,000	13	2200	380,000
	$\rho(\mathbf{M})$ opt	0.9742	0.6440	0.7411	0.4728	0.3601	0.4454
	Sm. fct. opt	0.4119	0.6440	0.7411	0.4421	0.3601	0.4454
1.0	c^* opt	18	6100	700,000	30	6300	680,000
	$\rho(\mathbf{M})$ opt	0.9694	0.7140	0.6729	0.2653	0.5088	0.4548
	Sm. fct. opt	0.2762	0.7140	0.6729	0.2653	0.5088	0.4548

Table 7 Maximal c^* for AW3, $\eta = 0.8$, $c = 200$, $M = 0.8$, $\alpha = 0°$. A implies that no bound was observed

	SGS precond.			Exact precond.		
d AR	1	100	10,000	1	100	10,000
0.0	8	8	8	8	8	8
0.1	11	13	13	13	13	13
0.25	30	2100	A	47	98	98
0.5	A	A	A	A	A	A
1.0	A	A	A	A	A	A

Surprisingly, for d small, stability is worse than for the preconditioned AERK methods. This is also illustrated in Fig. 2 for for Mach 0.5 and aspect ratios 1 and 100, respectively. As with AERK3J, the eigenvalues are clustered along the real axis.

A slightly more complex picture emerges when the optimal smoothing factor is considered. At AR $= 1$, the AW3 scheme attains very low optimal smoothing factors of around 0.3 at all values of d while the AERK3J scheme smoothing factors improved with increasing d. Comparing SGS preconditioning in both schemes, the optimal smoothing factors obtained by AW3 are slightly higher than AERK3J. Using exact preconditioning in both schemes at AR $= 100$ and 10,000, AW3 and AERK3J obtain comparable smoothing factors. Regarding the optimal c^*, it is generally lower than with AERK3J except for $d \geq 0.5$ and AR> 1.

5.4.4 Comparison of AW Schemes and Choice of η

One important question is the optimal choice of the additional parameter η in the W methods. Based on the AW3 results in Table 7 it was decided to focus on two values of d: $d = 0.1$ where limited stability was observed, and $d = 0.5$ where A-stability was observed. Only

Table 8 Amplification and smoothing factors of AW3, $\eta = 0.8$, $c = 200$, $M = 0.8$, $\alpha = 0°$

		SGS precond.			Exact precond.		
d	AR	1	100	10,000	1	100	10,000
0.0	c^* opt	3	8	8	3	8	8
	$\rho(\mathbf{M})$ opt	0.9836	0.9452	0.9441	0.9781	0.9440	0.9440
	Sm. fct. opt	0.3046	0.9452	0.9441	0.3046	0.9440	0.9440
0.1	c^* opt	3	12	13	3	12	12
	$\rho(\mathbf{M})$ opt	0.9837	0.9217	0.9140	0.9781	0.9188	0.9188
	Sm. fct. opt	0.2969	0.9217	0.9140	0.2933	0.9188	0.9188
0.25	c^* opt	3	240	500	3	70	70
	$\rho(\mathbf{M})$ opt	0.9838	0.7157	0.6825	0.9781	0.6843	0.6843
	Sm. fct. opt	0.2818	0.7157	0.6825	0.2787	0.6843	0.6843
0.5	c^* opt	3	$> 1e6$	$> 1e6$	4	$> 1e6$	$> 1e6$
	$\rho(\mathbf{M})$ opt	0.9841	0.7977	0.7916	0.9710	0.3511	0.3511
	Sm. fct. opt	0.2686	0.7977	0.7916	0.2669	0.3511	0.3511
1.0	c^* opt	7	$> 1e6$	$> 1e6$	7	$> 1e6$	$> 1e6$
	$\rho(\mathbf{M})$ opt	0.9765	0.8853	0.8832	0.9506	0.4897	0.4897
	Sm. fct. opt	0.2564	0.8853	0.8832	0.2627	0.4897	0.4897

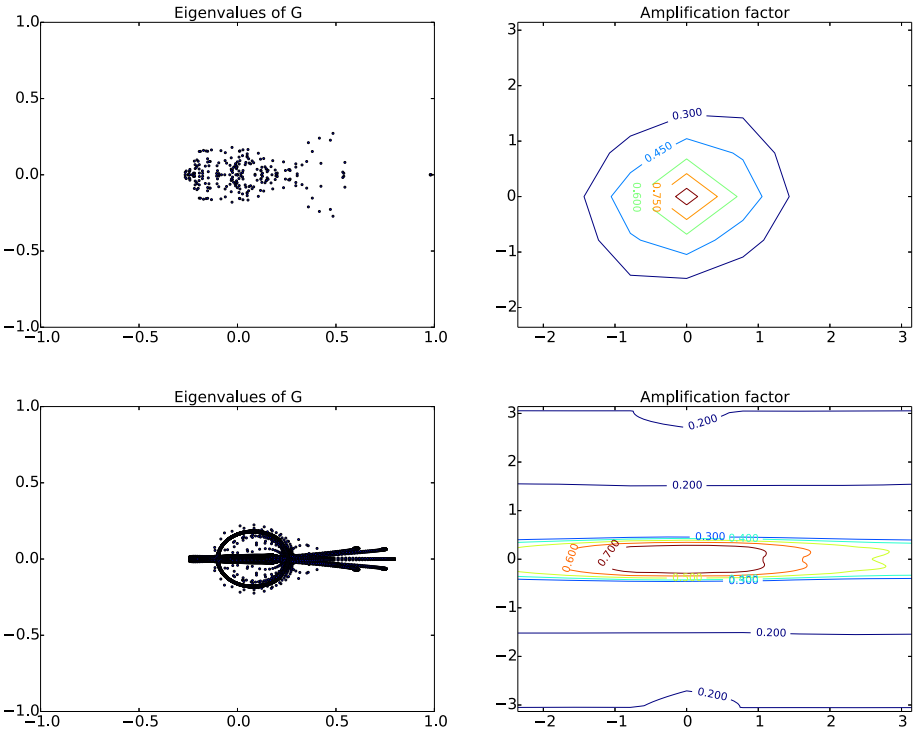

Fig. 2 Spectrum and amplification factors for different wavenumbers, Mach 0.8, $\alpha = 0°$, AW3 with SGS preconditioner, $c = 200$, $d = 0.5$, $\eta = 0.8$. Top: AR $= 1$, $c^* = 3$; bottom: AR $= 100$, $c^* = 1e6$

Table 9 Optimal η, c^*, amplification and smoothing factors of all AW schemes, $c = 200$, $M = 0.8$, $\alpha = 0°$, SGS preconditioning

	Scheme	AW3			AW51		
d	AR	1	100	10,000	1	100	10,000
0.1	c^* opt	3	12	13	3	8	11
	η opt	0.6	0.8	0.8	0.6	0.8	0.9
	$\rho(\mathbf{M})$ opt	0.9823	0.9217	0.9140	0.9823	0.9454	0.9824
	Sm. fct. opt	0.2604	0.9217	0.9140	0.2624	0.9454	0.9824
0.5	c^* opt	3	$> 1e6$	$> 1e6$	4	30	$> 1e6$
	η opt	0.4	0.5	0.5	0.5	0.5	0.7
	$\rho(\mathbf{M})$ opt	0.9809	0.6934	0.6848	0.9776	0.8598	0.7671
	Sm. fct. opt	0.2630	0.6934	0.6848	0.2610	0.8598	0.7671
	Scheme	AW52			AW5J		
d	AR	1	100	10,000	1	100	10,000
0.1	c^* opt	3	8	10	3	9	10
	η opt	0.6	0.8	0.8	0.5	0.7	0.8
	$\rho(\mathbf{M})$ opt	0.9823	0.9456	0.9323	0.9815	0.9389	0.9323
	Sm. fct. opt	0.2256	0.9456	0.9323	0.2064	0.9389	0.9323
0.5	c^* opt	3	$> 1e6$	$> 1e6$	3	$> 1e6$	$> 1e6$
	η opt	0.5	0.5	0.5	0.4	0.5	0.5
	$\rho(\mathbf{M})$ opt	0.9818	0.7031	0.6951	0.9809	0.7007	0.6924
	Sm. fct. opt	0.1762	0.7031	0.6951	0.1721	0.7007	0.6924

SGS preconditioning was used. For each W scheme and value of d, optimal values of c^*, η and amplification and smoothing factors were determined. These are presented in Table 9 for initial conditions Mach=0.8, $\alpha = 0°$ and in Table 10 for initial conditions Mach $= 0.8$, $\alpha = 45°$.

Looking just at Table 9, the optimal value of η is low, either 0.4 or 0.5 (with one case of 0.7), when $d = 0.5$. When $d = 0.1$, the optimal η depends on AR: for $AR = 1$, optimal values of η are 0.5 or 0.6 and for $AR = 100$ and 10,000 the values are higher, mostly 0.8. Looking at Table 10, the optimal value of η is independent of d and the choice of scheme but not of AR. The optimal value of η appears to be somewhat dependent on the initial conditions and other free parameters but independent of the specific W scheme. Furthermore, the optimisation process demonstrated (not all results are shown for brevity) that the W schemes are all stable within a range of about $\eta \in [0.5, 0.9]$, but the maximal c^* varies with η within the range. As shown in Table 8, fixing $\eta = 0.8$ across all tests results in a stable but sub-optimal scheme. Looking at the relative performance of different W schemes in Tables 9 and 10, it is apparent that they all obtain similar optimal smoothing factors at similar c^* values. Therefore, AW3 is the best scheme as it uses only three stages.

The discrete Fourier analysis suggests that the preconditioned AERK3J and additive W schemes should theoretically achieve very good smoothing factors under challenging flow conditions and on high aspect ratio grids. Moreover, in the W schemes the eigenvalue limiting parameter d plays an important role: for $d \geq 0.5$ and $AR > 1$ the allowable c^* is unlimited,

Table 10 Optimal η, c^*, amplification and smoothing factors of all AW schemes, $c = 200$, $M = 0.8, \alpha = 45°$, SGS preconditioning

	Scheme	AW3			AW51		
d	AR	1	100	10,000	1	100	10,000
0.1	c^* opt	3	900	400	3	1200	300
	η opt	0.6	0.8	0.9	0.6	0.8	0.8
	$\rho(\mathbf{M})$ opt	0.9804	0.4481	0.4367	0.9804	0.4545	0.4390
	Sm. fct. opt	0.2642	0.4481	0.4367	0.2654	0.4545	0.4390
0.5	c^* opt	3	$> 1e6$	$> 1e6$	3	$> 1e6$	$> 1e6$
	η opt	0.6	0.8	0.9	0.6	0.8	0.9
	$\rho(\mathbf{M})$ opt	0.9809	0.4441	0.4363	0.9804	0.4484	0.4351
	Sm. fct. opt	0.2642	0.4441	0.4363	0.2654	0.4484	0.4351
	Scheme	AW52			AW5J		
d	AR	1	100	10,000	1	100	10,000
0.1	c^* opt	3	$> 1e6$	500	3	300	200
	η opt	0.5	0.8	0.8	0.5	0.8	0.9
	$\rho(\mathbf{M})$ opt	0.9798	0.4378	0.4107	0.9798	0.5460	0.5220
	Sm. fct. opt	0.1750	0.4378	0.4107	0.1526	0.5460	0.5220
0.5	c^* opt	3	$> 1e6$	$> 1e6$	3	1100	800
	η opt	0.5	0.8	0.9	0.5	0.8	0.9
	$\rho(\mathbf{M})$ opt	0.9799	0.4710	0.3957	0.9799	0.5427	0.4205
	Sm. fct. opt	0.1750	0.4710	0.3957	0.1527	0.5427	0.4205

while for smaller d or $AR = 1$ the optimal c^* is finite and smaller than that found for preconditioned AERK3J.

6 Numerical Results

We now proceed to tests on the RANS equations and use a FAS scheme as the iterative solver. We employ the Fortran code uflo103 to compute flows around pitching airfoils. All computations are run on Ubuntu 16.04 on a single core of an 8-core Intel i7-3770 CPU at 3.40GHz with 8 GB of memory.

C-type grids are employed, where the half of the cells that are closer to the boundary in y-direction get a special boundary layer scaling. To obtain initial conditions for the unsteady simulation, far field values are used from which a steady state is computed. The first unsteady time step does not use BDF-2, but implicit Euler as a startup for the multistep method. From then on, BDF-2 is employed. We look at the startup phase to evaluate the performance of steady state computations and at the second overall timestep, meaning the first BDF-2 step, to evaluate performance for the unsteady case.

As a first test case, we consider the flow around the NACA 64A010 pitching and plunging airfoil at a Mach number 0.796. The grid is illustrated in Fig. 3. For the pitching, we use a frequency of 0.202 and an amplitude of 1.01°. 36 timesteps per cycle (pstep) are chosen. The Reynolds number is 10^6 and the Prandtl number is 0.75. The grid is a C-mesh with 512×64

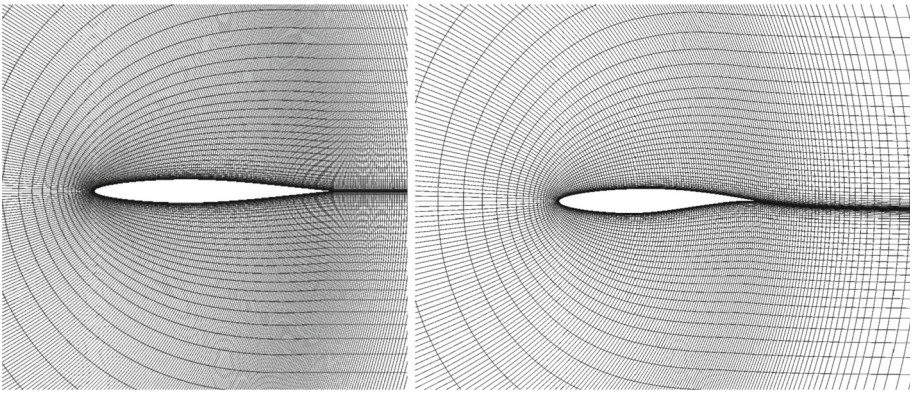

Fig. 3 Zoom of grids around NACA 64A010 and RAE 2822 airfoils

Table 11 Maximal c^* and convergence rates of UFLO103 for NACA 64A010 test case, $d = 0.5$

	Steady		Unsteady	
η	c^*	Conv. rate	c^*	Conv. rate
0.4	10	0.8774	10	0.8770
0.5	10,000	0.8616	10,000	0.8071
0.6	10,000	0.8629	10,000	0.8183
1.0	10,000	0.8759	10,000	0.8546

Table 12 Maximal c^* and convergence rates of UFLO103 for NACA 64A010 test case, $d = 0.1$

	Steady		Unsteady	
η	c^*	Conv. rate	c^*	Conv. rate
0.5	10,000	0.8561	100	0.7517 (90)
0.6	10,000	0.8583	100	0.7732

cells and maximum aspect ratio of $6.31e6$. As a second test case, we look at the pitching RAE 2822 airfoil at a Mach number of 0.75. The grid is illustrated in Fig. 3. For the pitching, we use a frequency of 0.202 and an amplitude of 1.01° and pstep = 36. The grid has 320×64 cells and maximum aspect ratio of $8.22e6$.

The results of the Fourier analysis suggest that the most interesting schemes are SGS preconditioned AERK3J and the various AW schemes. A first thing to note is that due to nonlinear effects, the schemes need to be tweaked from the linear to the nonlinear case. In particular, it is necessary to start with a reduced pseudo CFL number c^*. We restrict it to 20 for the first two iterations.

6.1 Choice of Parameters

In the AW methods, there are now three interdependent parameters to choose: η, d and c^*, the CFL number in pseudo time. We start by fixing d. Choosing $d = 0$ does not cause instability per se, but it leads to a stall in the iteration away from the solution. The convergence rates for $d = 0.05$, $d = 0.1$ and $d = 0.5$ for the NACA and the RAE test case can be seen in Tables 11, 12, 13, 14, 15, 16. The largest c^* tried is 10,000 in all cases. If the number reported

Table 13 Maximal c^* and convergence rates of UFLO103 for NACA 64A010 test case, $d = 0.05$

	Steady		Unsteady	
η	c^*	Conv. rate	c^*	Conv. rate
0.4	10	0.8787	10	0.8828
0.5	10,000	0.8506	100	0.7503 (90)
0.6	10,000	0.8533	100	0.7721

Table 14 Maximal c^* and convergence rates of UFLO103 for RAE 2822 test case, $d = 0.5$

	Steady		Unsteady	
η	c^*	Conv. rate	c^*	Conv. rate
0.4	10,000	0.8567	10,000	0.8228
0.5	10,000	0.8581	10,000	0.8424
0.6	10,000	0.8631	10,000	0.8574
0.7	10,000	0.8691	10,000	0.8688
0.8	10,000	0.8740	10,000	0.8766
0.9	10,000	0.8744	10,000	0.8817
1.0	10,000	0.8804	10,000	0.8866

Table 15 Maximal c^* and convergence rates of UFLO103 for RAE 2822 test case, $d = 0.1$

	Steady		Unsteady	
η	c^*	Conv. rate	c^*	Conv. rate
0.4	400	0.8429	60	0.7996
0.5	10,000	0.8359	70	0.8000
0.8	10,000	0.8471	60	0.8304

Table 16 Maximal c^* and convergence rates of UFLO103 for RAE 2822 test case, $d = 0.05$

	Steady		Unsteady	
η	c^*	Conv. rate	c^*	Conv. rate
0.4	100	0.8470	60	0.7895
0.5	10,000	0.8224	60	0.7972
0.6	10,000	0.8281	60	0.8045
0.7	10,000	0.8326	70	0.8029

is smaller, it implies that it is the largest for which the methods are convergent. A number in parentheses e.g. (90) after the convergence rate means that the rate was calculated for the first 90 iterations, after which convergence stalled. Only stable values of η are reported for brevity. The schemes are stable within a certain range, $0.5 \leq \eta \leq 0.9$, which tallies with the Fourier analysis results.

Qualitatively, we observe the following behavior:

- Increasing d makes the schemes slower to converge and more stable
- This effect is stronger for the unsteady system
- If η is too small, we get instability
- Decreasing η within the stable region will improve the convergence rate

We thus suggest two different modes of operation:

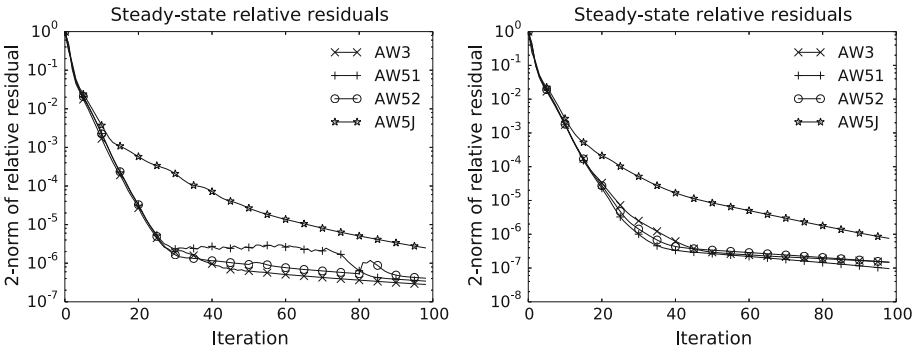

Fig. 4 Convergence behavior for steady flow around NACA64A010 (left) and RAE 2822 (right) airfoil for different AW schemes, 0° angle of attack, $d = 0.05$, $c^* = 100$

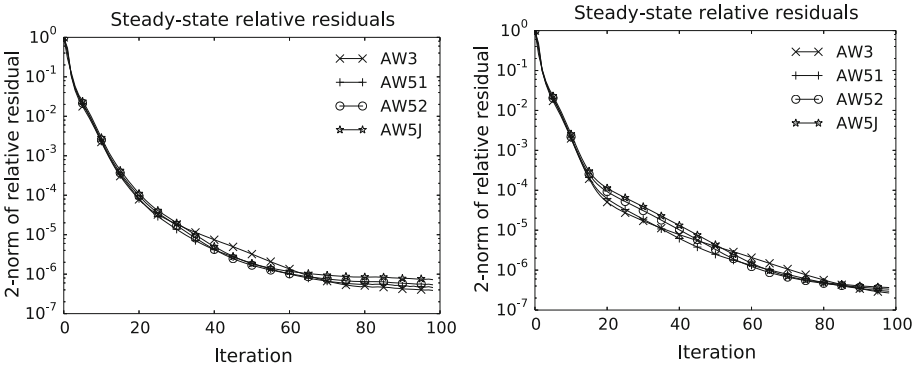

Fig. 5 Convergence behavior for steady flow around NACA64A010 (left) and RAE 2822 (right) airfoil for different AW schemes, 0° angle of attack, $d = 0.5$, $c^* = 10{,}000$

1. The robust mode: Choose $d = 0.5$, $\eta = 0.5$ and c^* very large
2. The fast mode: Choose $d = 0.05$, $\eta = 0.5$ and $c^* = 100$

The robust mode trades some convergence rate for more robustness.

The numerical experiments find somewhat different optimal values of η to those found in the discrete Fourier analysis. Possible reasons for the discrepancies include the linearisations used in the discrete Fourier analysis and the non-cartesian meshes in the numerical experiments.

6.2 Comparison of Schemes

The linear analysis suggests that preconditioned AERK3 is competitive with the preconditioned W methods in terms of smoothing power. However, its application requires choosing c^* within a stability limit whereas the W methods are A-stable for a certain range of d. To test the stability limit of the AERK schemes, we apply preconditioned AERK3 and AERK51 to the pitching NACA airfoil test case. The AERK3 method becomes unstable for $c^* > 1$, whereas AERK51 can be run with $c^* = 3$. However, both methods are completely uncompetitive with convergence rates of 0.999. Hereafter we compare only the AW schemes.

We compare AW3, AW51, AW52 and AW5J for the two airfoils and the two modes of operation: $d = 0.05$ and $c^* = 100$ versus $d = 0.5$ and $c^* = 10{,}000$. The relative residuals

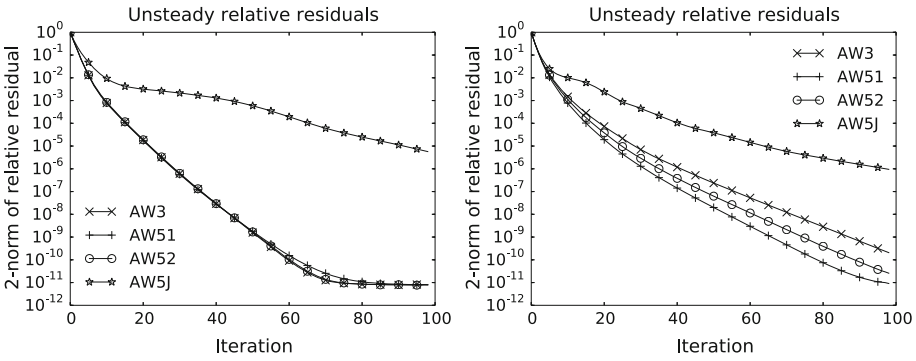

Fig. 6 Convergence behavior for unsteady flow around NACA64A010 (left) and RAE 2822 (right) airfoil for different AW schemes, $d = 0.05$, $c^* = 100$

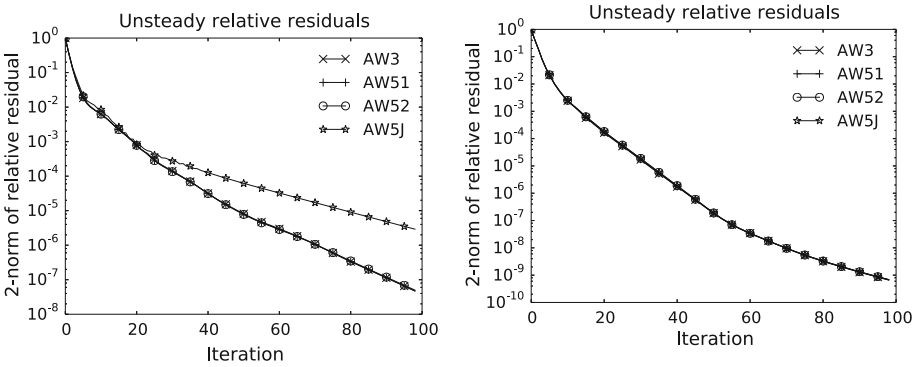

Fig. 7 Convergence behavior for unsteady flow around NACA64A010 (left) and RAE 2822 (right) airfoil for different AW schemes, $d = 0.5$, $c^* = 10{,}000$

for the initial steady state computation are plotted in Fig. 4 for $d = 0.05$ and $c^* = 100$ and in Fig. 5 for $d = 0.5$ and $c^* = 10{,}000$. The residual histories for the same tests, but for the second unsteady timestep can be seen in Figs. 6 and 7.

The convergence rates and CPU times are summarized in Table 17 for the NACA64A010 airfoil and in Table 18 for the RAE 2822 airfoil. The numbers after the scheme names are the values of c^* used in the steady iterations. Faster convergence is obtained with $d = 0.05$ for all schemes except AW5J.

As an immediate conclusion, it can be seen that the different schemes have similar convergence rates. Thus, AW3 performs best in terms of CPU times, since it is a three stage smoother, opposed to the five stage smoothers. With the fast mode, we get a convergence rate for the unsteady case of 0.77 for the NACA profile and 0.8 for the RAE profile. However, for the RAE profile, we have to reduce c^* from 100 for 3 of the 4 schemes to prevent instability. With the convergence rate obtained, 20–30 iterations are sufficient to get a reduction of the residual by five order of magnitude. This is a matter of seconds and is sufficient for most applications. Note however, that for example when assessing the quality of different turbulence models, much lower residuals are desirable. In the robust mode, the convergence rate goes down to 0.8 for the NACA profile and 0.84 for the RAE profile.

In the steady state cases, there is a decline in convergence rate after 20 to 30 iterations. This explains why the convergence rates are significantly slower here than the unsteady ones.

Table 17 Performance of AW3, AW51 and AW52 for the pitching NACA64A010 airfoil, 0° angle of attack, 100 steady/unsteady iterations

		$d = 0.05, c^* = 100$		$d = 0.5, c^* = 10{,}000$	
		CPU[s]	Av. conv. rate	CPU[s]	Av. conv. rate
Steady	AW3	13.6	0.8586	14.5	0.8616
	AW5J	21.3	0.8775	21.4	0.8671
	AW51	21.0	0.8603	21.3	0.8632
	AW52	20.8	0.8618	21.5	0.8645
Unsteady	AW3	19.0	0.7724	18.9	0.8071
BDF-2	AW5J	28.6	0.8844	29.0	0.8074
	AW51	28.7	0.7725	29.6	0.8071
	AW52	28.5	0.7724	29.2	0.8074

Table 18 Performance of AW3, AW51 and AW52 for the RAE 2822 airfoil, 0° angle of attack, 100 steady/unsteady iterations

		$d = 0.05, c^* = 60/100$		$d = 0.5, c^* = 10{,}000$	
		CPU[s]	Av. conv. rate	CPU[s]	Av. conv. rate
Steady	AW3/60	8.6	0.8530	8.9	0.8581
	AW5J/20	12.7	0.8670	13.1	0.8609
	AW51/100	12.6	0.8492	13.1	0.8590
	AW52/80	12.7	0.8530	13.1	0.8601
Unsteady	AW3	11.7	0.7972	12.0	0.8424
BDF-2	AW5J	17.5	0.8687	18.0	0.8786
	AW51	17.5	0.7732	18.1	0.8424
	AW52	17.9	0.7808	18.0	0.8429

In the first phase, a convergence rate of about 0.7 is obtained and the norm of the residual is decreased by about 10^6.

The cause of this decline in convergence must be in nonlinear effects or in boundary conditions. One point of future investigation is if weak boundary conditions would be a better choice [26].

6.3 Different Meshes

To assess the solvers' mesh-dependence, we run the pitching NACA 64010 airfoil with AW3 and $d = 0.5$ on coarse (256×32), medium (512×64) and fine (1024×128) meshes in robust mode. With d set to 0.05 the simulations on the coarse mesh diverged, which is an example where the robust mode is indeed more robust. We use seven multigrid levels for the finest mesh. Table 19 shows the results. As can be seen, the convergence of the preconditioned AW schemes goes down on the finest mesh, caused by a stall in the residual after 50 iterations and a reduction of the residual by five orders of magnitude. To fix this, the solver has to be adjusted. The last line shows a computation with $\eta = 0.8$ and two SGS steps instead of one,

Table 19 Convergence rates for the pitching NACA 64010 airfoil with AW3 and $d = 0.5$ on different meshes. The last row corresponds to $\eta = 0.8$ and two SGS steps

Mesh	Steady	Unsteady
256×32	0.8698	0.8110
512×64	0.8616	0.8071
1024×128	0.8667	0.8917
1024×128	0.8689	0.8103

Table 20 Convergence rate with AW3 smoothing for the pitching NACA 64A010 airfoil at different angles of attack, $c^* = 10{,}000$, $d = 0.5$, pstep=36

	Angle	NACA	RAE
Steady	0	0.8623	0.8625
	1	0.8500	0.8639
	2	0.8597	0.8590
	4	0.8530	0.8564
Unsteady	0	0.8249	0.8846
	1	0.8239	0.9097 (65)
$\Delta t = 0.486822$	2	0.8254	0.8307
	4	0.9055 (30)	0.8290

increasing computational effort, but recovering the convergence rate. On even finer meshes, we observe again the stall after 5 orders of magnitude residual reduction.

6.4 Effect of Flow Angle

In the Fourier analysis it was found that grid-aligned flow could be problematic. We therefore choose angles of attack α of 0, 1, 2 and 4 degrees for the steady state computation or the second time step in an unsteady computation. Table 20 shows the convergence rates in fast mode ($d = 0.05$). Essentially, it is unaffected by the angle of attack. However, for two cases, the iteration stalls after 30, resp. 65 iterations at relative residuals of 10^{-4} and 10^{-5}, respectively.

7 Conclusions

We considered preconditioned pseudo time iterations for agglomeration multigrid schemes for the steady and unsteady RANS equations. As a discretization, the JST scheme was used as a flux function in a finite volume method. Based on previous work of other authors, we derived AW methods, as well as preconditioned AERK methods from time integration schemes. Both are implemented in exactly the same way with the difference being in how the preconditioner is chosen, as well as the pseudo time step size. For the latter, the preconditioner has to approximate the Jacobian $\mathbf{J}$, wheras for the AW method, the preconditioner has to approximate $\mathbf{I} + \eta \Delta t^* \mathbf{J}$. The time integration based derivation allows to conclude that preconditioned AERK has a finite stability region, whereas AW allows for possibly unbounded pseudo time steps. However, we obtain an additional parameter η which currently must be chosen empirically. As a preconditioner, we choose a flux vector splitting with a cutoff of small eigenvalues controlled by the free variable d.

To compare the different methods, we used a discrete Fourier analysis of the linearized Euler equations. Numerical results show that AW3, AW51 and AW52 have similar convergence rates, meaning that AW3 performs best, since it uses two stages less. The free parameter η can be chosen with relative freedom within a stable range (about [0.5, 0.9]) although the optimal value is dependent in some cases on the initial conditions, d and the aspect ratio. Fixing $\eta = 0.8$ is an acceptable simplification in the cases tested. The most significant parameter affecting stability and convergence is the eigenvalue cutoff coefficient d in the numerical flux function. It was found that the W schemes were A-stable for $d \geq 0.5$ and had stability limits lower than preconditioned AERK schemes for $d < 0.5$. Thirdly, the pseudo CFL number c^* was tuned for optimal performance. Different optimal values were obtained for different aspect ratios but as long as c^* was within the stability limit, good convergence was achieved. This is useful since the aspect ratios in practical meshes vary considerably.

Simulations of pitching and plunging NACA 64A010 and RAE2822 airfoils in high Reynolds number flow at Mach 0.796 were performed using the 2D URANS code uflo103. The preconditioned AERK schemes were completely uncompetitive with convergence rates of around 0.999. The additive W schemes, on the other hand, achieved convergence rates of as low as 0.85 for the initial steady-state iteration and 0.77 for the unsteady iterations. Slightly different optimal values of η and c^* were found although the behaviour of the schemes was qualitatively similar to that predicted by the linear analysis. We emphasise two modes of operation for the AW schemes: a fast mode, $d = 0.05$, $\eta = 0.5$ and $c^* = 100$ and a robust mode, $d = 0.5$, $\eta = 0.5$ and $c^* = 10{,}000$. Unsteady convergence rates in the robust mode were higher than the fast mode but still competitive. Steady-state convergence rates for all tests stalled to varying degrees after around 20 iterations but the residuals had already fallen by 6 orders of magnitude.

In summary, the new additive W schemes achieve excellent performance as smoothers in the agglomeration multigrid method applied to 2D URANS simulations of high Reynolds number transonic flows. The stiffness associated with very high aspect ratio grids is counteracted by highly tuned preconditioning. The underlying aim of this paper was to present a complete analysis of the reasons why such preconditioned iterative smoothers are effective, in order that their high performance can be replicated. We encountered two parameters that resisted analysis and had to be tuned empirically: η and d. Nevertheless, this is considered a great improvement. Future work will look at these parameters in more detail. In addition, boundary conditions should have an influence on convergence speed.

Acknowledgements We would like to thank Charlie Swanson for interesting discussions and sharing some code with us.

Funding Funding was provided by Kungliga Fysiografiska Sällskapet i Lund, vetenskapsrådet via grant 2015–04133 and the Air Force Office of Scientific Research via grant FA9550-14-1-0186 under the direction of Fariba Fahroo.

References

1. Baldwin, B.S., Lomax, H.: Thin layer approximation and algebraic model for separated turbulent flows. AIAA Paper 78–257 (1978)

2. Bertaccini, D., Donatelli, M., Durastante, F., Serra-Capizzano, S.: Optimizing a multigrid Runge–Kutta smoother for variable-coefficient convection–diffusion equations. Linear Algebra Appl. **533**, 507–535 (2017)
3. Birken, P.: Numerical methods for the unsteady compressible Navier–Stokes equations. Habilitation Thesis. University of Kassel (2012)
4. Birken, P.: Optimizing Runge–Kutta smoothers for unsteady flow problems. ETNA **39**, 298–312 (2012)
5. Birken, P., Bull, J., Jameson, A.: A note on terminology in multigrid methods. PAMM **16**, 721–722 (2016)
6. Birken, P., Bull, J., Jameson, A.: A study of multigrid smoothers used in compressible CFD based on the convection diffusion equation. In: Papadrakakis, M., Papadopoulos, V., Stefanou, G., Plevris, V. (eds.) ECCOMAS Congress 2016, VII European Congress on Computational Methods in Applied Sciences and Engineering, vol. 2, pp. 2648–2663. Crete Island, Greece (2016)
7. Caughey, D., Jameson, A.: 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)
8. Cooper, G.J., Sayfy, A.: Additive methods for the numerical solution of ordinary differential equations. Math. Comput. **35**(152), 1159–1172 (1980)
9. Gustafsson, B.: High Order Difference Methods for Time Dependent PDE. Springer, Berlin (2008)
10. Hairer, E., Wanner, G.: Solving Ordinary Differential Equations II, 2nd edn. Springer, Berlin (2002)
11. Jameson, A.: Transonic flow calculations for aircraft. In: Brezzi, F. (ed.) Numerical Methods in Fluid Dynamics, Lecture Notes in Mathematics, pp. 156–242. Springer, Berlin (1985)
12. Jameson, A.: Multigrid algorithms for compressible flow calculations. In: W. Hackbusch, U. Trottenberg (eds.) 2nd European Conference on Multigrid Methods, vol. 1228, pp. 166–201. Springer (1986)
13. Jameson, A.: Aerodynamics. In: E. Stein, R. de Borst, T.J.R. Hughes (eds.) Encyclopedia of Computational Mechanics, vol. 3: Fluids, chap. 11, pp. 325–406. Wiley (2004)
14. Jameson, A.: Application of dual time stepping to fully implicit Runge Kutta schemes for unsteady flow calculations. In: 22nd AIAA Computational Fluid Dynamics Conference, 22–26 June 2015, Dallas, TX, AIAA Paper 2015–2753 (2015)
15. Jameson, A.: The origins and further development of the Jameson–Schmidt–Turkel (JST) scheme. In: 33rd AIAA Applied Aerodynamics Conference, 22–26 June 2015, Dallas, TX, AIAA Paper 2015–2718 (2015). https://doi.org/10.2514/6.2015-2718
16. Jameson, A.: Evaluation of fully implicit Runge Kutta schemes for unsteady flow calculations. J. Sci. Comput. (2017). https://doi.org/10.1007/s10745-006-9094-1
17. Langer, S.: Investigation and application of point implicit Runge–Kutta methods to inviscid flow problems. Int. J. Numer. Methods Fluids **69**(2), 332–352 (2012). https://doi.org/10.1002/fld
18. Langer, S.: Application of a line implicit method to fully coupled system of equations for turbulent flow problems. Int. J. CFD **27**(3), 131–150 (2013). https://doi.org/10.1080/10618562.2013.784902
19. Langer, S.: Agglomeration multigrid methods with implicit Runge–Kutta smoothers applied to aerodynamic simulations on unstructured grids. J. Comput. Phys. **277**, 72–100 (2014). https://doi.org/10.1016/j.jcp.2014.07.050
20. Langer, S., Li, D.: Application of point implicit Runge–Kutta methods to inviscid and laminar flow problems using AUSM and AUSM + upwinding. Int. J. CFD **25**(5), 255–269 (2011). https://doi.org/10.1080/10618562.2011.590801
21. Langer, S., Schwöppe, A., Kroll, N.: Investigation and comparison of implicit smoothers applied in agglomeration multigrid. AIAA J. **53**(8), 2080–2096 (2015)
22. Martinelli, L.: Calculations of viscous flows with a multigrid method. Ph.D. thesis, Princeton University (1987)
23. Mavriplis, D.J., Venkatakrishnan, V.: Agglomeration multigrid for two-dimensional viscous flows. Comput. Fluids **24**(5), 553–570 (1995). https://doi.org/10.1016/0045-7930(95)00005-W
24. Mulder, W.A.: A new multigrid approach to convection problems. J. Comput. Phys. **83**, 303–323 (1989)
25. Mulder, W.A.: A high-resolution Euler solver based on multigrid semi-coarsening, and defect correction. J. Comput. Phys. **100**, 91–104 (1992)
26. Nordström, J., Forsberg, K., Adamsson, C., Eliasson, P.: Finite volume methods, unstructured meshes and strict stability for hyperbolic problems. Appl. Numer. Math. **45**(4), 453–473 (2003). https://doi.org/10.1016/S0168-9274(02)00239-8
27. Peles, O., Turkel, E.: Acceleration methods for multi-physics compressible flow. J. Comput. Phys. **358**, 201–234 (2018). https://doi.org/10.1016/j.jcp.2017.10.011
28. Rossow, C.C.: Convergence acceleration for solving the compressible Navier–Stokes equations. AIAA J. **44**, 345–352 (2006)
29. Rossow, C.C.: Efficient computation of compressible and incompressible flows. J. Comput. Phys. **220**(2), 879–899 (2007). https://doi.org/10.1016/j.jcp.2006.05.034

30. Swanson, R.C., Rossow, C.C.: An efficient solver for the RANS equations and a one-equation turbulence model. Comput. Fluids **42**(1), 13–25 (2011). https://doi.org/10.1016/j.compfluid.2010.10.010
31. Swanson, R.C., Turkel, E.: Analysis of a fast iterative method in a dual time algorithm for the Navier–Stokes equations. In: J. Eberhardsteiner (ed.) European Congress on Computational Methods and Applied Sciences and Engineering (ECCOMAS 2012) (2012)
32. Swanson, R.C., Turkel, E., Rossow, C.C.: Convergence acceleration of Runge–Kutta schemes for solving the Navier–Stokes equations. J. Comput. Phys. **224**(1), 365–388 (2007). https://doi.org/10.1016/j.jcp.2007.02.028
33. Swanson, R.C., Turkel, E., Yaniv, S.: Analysis of a RK/implicit smoother for multigrid. In: Kuzmin, A. (ed.) Computational Fluid Dynamics 2010. Springer, Berlin (2011)
34. Trottenberg, U., Oosterlee, C.W., Schüller, S.: Multigrid. Elsevier Academic Press, Cambridge (2001)
35. van Leer, B., Tai, C.H., Powell, K.G.: Design of optimally smoothing multi-stage schemes for the Euler equations. In: AIAA 89-1933-CP, pp. 40–59 (1989)