A comparison of Rosenbrock and ESDIRK methods combined with iterative solvers for unsteady compressible flows

Blom, David S.; Birken, Philipp; Bijl, Hester; Kessels, Fleur; Meister, Andreas; van Zuijlen, Alexander H.

DOI
10.1007/s10444-016-9468-x

Publication date
2016

Document Version
Final published version

Published in
Advances in Computational Mathematics

Adv Comput Math (2016) 42:1401–1426
DOI 10.1007/s10444-016-9468-x

A comparison of Rosenbrock and ESDIRK methods combined with iterative solvers for unsteady compressible flows

David S. Blom[1] · Philipp Birken[2] · Hester Bijl[1] · Fleur Kessels[1] · Andreas Meister[3] · Alexander H. van Zuijlen[1]

Received: 5 June 2015 / Accepted: 23 June 2016 /
Published online: 6 July 2016

Abstract In this article, we endeavour to find a fast solver for finite volume discretizations for compressible unsteady viscous flows. Thereby, we concentrate on comparing the efficiency of important classes of time integration schemes, namely time adaptive Rosenbrock, singly diagonally implicit (SDIRK) and explicit first stage singly diagonally implicit Runge-Kutta (ESDIRK) methods. To make the comparison fair, efficient equation system solvers need to be chosen and a smart choice of tolerances is needed. This is determined from the tolerance TOL that steers time adaptivity. For implicit Runge-Kutta methods, the solver is given by preconditioned inexact Jacobian-free Newton-Krylov (JFNK) and for Rosenbrock, it is preconditioned Jacobian-free GMRES. To specify the tolerances in there, we suggest a simple strategy of using TOL/100 that is a good compromise between stability and computational effort. Numerical experiments for different test cases show that the fourth

Communicated by: Silas Alben

✉ David S. Blom
davidsblom@gmail.com

Philipp Birken
philipp.birken@na.lu.se

Andreas Meister
meister@mathematik.uni-kassel.de

[1] Faculty of Aerospace Engineering, Delft University of Technology, P.O. Box 5058, 2600 GB Delft, The Netherlands

[2] Centre for the Mathematical Sciences, Numerical Analysis, Lunds University, Box 118, 22100 Lund, Sweden

[3] Institute of Mathematics, University of Kassel, Heinrich-Plett-Str. 40, 34132 Kassel, Germany

order Rosenbrock method RODASP and the fourth order ESDIRK method ESDIRK4 are best for fine tolerances, with RODASP being the most robust scheme.

Keywords Rosenbrock methods · Navier-Stokes equations · ESDIRK · Jacobian-free Newton-Krylov · Unsteady flows · Time adaptivity

Mathematics Subject Classification (2010) 76N99

1 Introduction

In many engineering and scientific problems, unsteady compressible fluid dynamics play a key role. Examples would be simulation of tunnel fires [3], flow around wind turbines [49], fluid-structure-interaction like flutter [10], aeroacoustics [6], turbomachinery, flows inside nuclear reactors [32], wildfires [31], hurricanes and unsteady weather phenomenas [30], gas quenching [20] and many others. Simulation of such unsteady phenomena is extremely computationally expensive due to the vast amount of time steps that need to be taken. An efficient (low computational time for a given accuracy) time integration scheme is therefore of the utmost importance.

In wall bounded flows, boundary layers are present, which need a high resolution to be resolved [24]. This in turn causes the time step of explicit methods to be restricted due to stability considerations, making implicit or linear implicit time integration necessary. Furthermore, the high aspect ratio cells introduced in the boundary layer contribute to an increased stiffness of the problem. The wishlist for time integration scheme is thus that to deal with stiff and oscillatory problems they should be A- or L-stable, time adaptivity should be easy and of course, they should be efficient [4].

The scheme mainly used in CFD is the backward differentiation formula BDF-2, a multistep method for which a time adaptive implementation exists with SUNDIALS [12]. An alternative are singly diagonally implicit Runge-Kutta (SDIRK) methods [4]. In the autonomous case, these consist of a sequence of implicit Euler steps, meaning that in every stage, a nonlinear system has to be solved. Another variant are explicit first stage singly diagonally implicit Runge-Kutta methods (ESDIRK), where the first step is explicit. A specific example are ESDIRK3 and ESDIRK4, which were designed in [1, 17, 19, 36, 48]. These are a four stage method of third order with an embedded method of second order and a six stage method of fourth order with an embedded method of third order. The use of these schemes in the context of compressible Navier-Stokes equations was analyzed in [2] where they were demonstrated to be more efficient than implicit Euler (BDF-1) and BDF-2 methods for moderately accurate solutions. This result about the comparative efficiency of BDF-2 and ESDIRK4 was later confirmed for unsteady Euler flow by [15] and for a discontinuous Galerkin discretization by [44]. Another interesting alternative is the second order SDIRK method SDIRK2, which was demonstrated to have good performance in [4], whereas SDIRK3 is not competitive.

An alternative to implicit Runge-Kutta schemes and BDF schemes that has not been considered much for compressible flows are Rosenbrock schemes, which are also referred to as linearly implicit or semi-implicit [42]. The idea is to linearize an

s-stage DIRK scheme, thus sacrificing some stability properties as well as accuracy, but hopefully reducing the computational effort in that per time step. s linear equation systems with the same system matrix and different right hand sides have to be solved.

Numerous Rosenbrock schemes have been proposed in literature. Rang presents new third and fourth order Rosenbrock schemes in [26–29] which satisfy extra order conditions to avoid order reduction [11]. The test cases we consider here are autonomous problems, and order reduction for these cases is much less a problem. Previously, these schemes have been considered by St-Cyr et. al. in the context of discontinuous Galerkin methods [40], as well as in [14] for the incompressible Navier-Stokes equations. In the latter paper, a large number of different ESDIRK and Rosenbrock methods is compared to each other in a time adaptive setting. The linear systems are solved using direct solvers.

The Rosenbrock methods we choose are thus the third order method ROS34PW2 from [29] and the fourth order method RODASP [41]. These methods are A- and L-stable, furthermore ROS34PW2 is a W-method, meaning that approximations to the Jacobian can be used. The latter is beneficial for the JFNK schemes we have in mind for the solver, see later.

Here, we compare time adaptive SDIRK, ESDIRK and Rosenbrock methods in the context of finite volume discretizations of the compressible Navier-Stokes equations based on their work error ratio for realistic problems. The comparison between Rosenbrock and DIRK schemes is nontrivial even for a fixed time step method, since the errors are different and solving linear systems is not necessarily faster than solving nonlinear systems. In the case of time adaptivity, a comparison becomes even more intricate, since the time step size influences the speed of the solvers in a nonlinear way. Furthermore, the amount of work depends on both the code and the test cases chosen.

Regarding solvers, the alternatives are Newton methods and Multigrid methods. We consider preconditioned inexact Jacobian-free Newton-GMRES [18] schemes with a state of the art choice of the tolerances [9, 38] in all iterations to be superior to current multigrid methods [5]. In Rosenbrock schemes, there is no Newton scheme, but the implementation can nevertheless be done in a Jacobian-free manner exactly as before. Furthermore, it has been justified in a series of papers on so called Krylov-ROW methods [33, 46, 47] that solving the linear systems inexactly can be done in Rosenbrock methods without loss of order. However, it is not clear how to choose the tolerance for GMRES and thus we will develop a strategy here.

For the iterative solution strategy using a GMRES approach, a good preconditioner is essential for obtaining a high computational efficiency, especially when solving stiff, ill conditioned system, e.g. as a result of high aspect ratio cells in the boundary layer. The preconditioner is always a trade off between accuracy (effectiveness) of the preconditioner and computational effort to build the preconditioner. In this respect the Rosenbrock schemes are expected to have an additional benefit over ESDIRK schemes as the linear system to solve for Rosenbrock schemes is constant for all stages within a time step, whereas for the ESDIRK schemes the linear system to solve changes with every stage and even every Newton step. In this paper we therefore also investigate the effect of the preconditioner on the required number of GMRES iterations with increased condition numbers for the system to solve.

As for realistic test cases, it is important to note that for complex 3D flows we have grids with extreme aspect ratios and that furthermore, we do not know the error. Thus, we will first work with problems where we have an exact or reference solution. In particular, we consider a 2D nonlinear convection-diffusion problem with variable non-linearity and grid stretching to demonstrate how the schemes react to changes in these.

Finally, we move to viscous flow problems and use two different codes and several test cases. To obtain a fair comparison here, we use both reference solutions and the concept of tolerance scaling [38] to obtain a reasonable relation between the tolerance in the time adaptive scheme and the error. The test cases are chosen to represent wall bounded laminar flows. Thus, there is a boundary layer, causing the need for high resolution in the vicinity of the walls, but there are no additional issues from turbulence.

The paper is organized as follows. Section 2 discusses the time integration schemes and time adaptivity. The methods used to solve nonlinear and linear systems of equations are the subject of Section 3. We then present results for the nonlinear convection-diffusion equation and Navier-Stokes simulations in Sections 4 and 5.

2 Time integration

Here, we use the method of lines paradigm, where a partial differential equation is first discretized in space and then in time. We restrict ourselves to autonomous problems, obtaining an initial value problem of the form

$$\frac{d}{dt}\mathbf{u}(t) = \mathbf{f}(\mathbf{u}(t)), \quad \mathbf{u}(0) = \mathbf{u}^0, \quad t \in [t_0, t_{end}], \tag{1}$$

where $\mathbf{u} \in \mathbb{R}^m$ and $\mathbf{f} : \mathbb{R}^m \to \mathbb{R}^m$. As time integration schemes, we consider SDIRK, ESDIRK and Rosenbrock schemes [11].

2.1 SDIRK and ESDIRK schemes

An SDIRK or ESDIRK scheme with s stages is of the form

$$\mathbf{U}_i = \mathbf{u}^n + \Delta t^n \sum_{j=1}^{i} a_{ij}\, \mathbf{f}\left(\mathbf{U}_j\right),\ i = 1, \ldots, s \tag{2}$$

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \Delta t^n \sum_{i=1}^{s} b_i\, \mathbf{f}\left(\mathbf{U}_i\right). \tag{3}$$

The Butcher tableau of a SDIRK method is illustrated in Table 1, wherein the diagonal coefficient a_{ii} is constant, which is a property of SDIRK schemes. For an ESDIRK scheme, the first stage is explicit, i.e. $a_{11} = 0$.

The schemes considered here are stiffly accurate, i.e. the last row of the Butcher tableau is identical to $\mathbf{b}^T$ which is advantageous in solving stiff problems [11]. This means that the solution at the next time step is obtained with $\mathbf{u}^{n+1} = \mathbf{U}_s$.

Table 1 Butcher tableau of a SDIRK method

c_1	a_{11}	0	0	0
c_2	a_{21}	a_{22}	0	0
$\vdots$	$\vdots$	$\ddots$	$\ddots$	0
c_s	a_{s1}	$\dots$	a_{ss-1}	a_{ss}
	a_{s1}	$\dots$	a_{ss-1}	a_{ss}

Thus, s or $s-1$ nonlinear equation systems have to be solved for SDIRK and ESDIRK, respectively. The advantage of DIRK schemes is that the computation of the stage vectors is decoupled. Instead of solving one nonlinear system with sm unknowns, s nonlinear systems (2) with m unknowns have to be solved. With the starting vectors

$$\mathbf{s}_i = \mathbf{u}^n + \Delta t^n \sum_{j=1}^{i-1} a_{ij}\, \mathbf{f}\left(\mathbf{U}_j\right), \tag{4}$$

equation (2) corresponds to one step of the implicit Euler method with starting vector $\mathbf{s}_i$ and time step $a_{ii}\,\Delta t^n$. Thus the solution at each implicit stage can be written as:

$$\frac{\mathbf{U}_i - \mathbf{u}^n}{a_{ii}\Delta t^n} = \mathbf{f}(\mathbf{U}_i) + \frac{1}{a_{ii}} \sum_{j=1}^{i-1} a_{ij}\mathbf{f}\left(\mathbf{U}_j\right). \tag{5}$$

The stage derivative $\mathbf{f}(\mathbf{U}_i)$ is later needed for time adaptivity. It is obtained via

$$\mathbf{f}(\mathbf{U}_i) = \frac{1}{a_{ii}\,\Delta t^n}(\mathbf{U}_i - \mathbf{s}_i),$$

which avoids a costly and for stiff problems error prone evaluation of the right hand side [34].

2.2 Rosenbrock schemes

To circumvent the solution of nonlinear equation systems, Rosenbrock methods, also called Rosenbrock-Wanner, ROW or linearly implicit methods, can be used. The idea is to linearize a DIRK scheme, thus sacrificing some stability properties, as well as accuracy, but obtaining a method that has to solve s linear equation systems with the same system matrix and different right hand sides per time step.

To derive these schemes, we start by linearizing $\mathbf{f}(\mathbf{u})$ around $\mathbf{s}_i$ (4) to obtain

$$\mathbf{k}_i \approx \mathbf{f}(\mathbf{s}_i) + \Delta t^n a_{ii}\,\frac{\partial \mathbf{f}(\mathbf{s}_i)}{\partial \mathbf{u}}\mathbf{k}_i,$$

introducing the notation $\mathbf{k}_i = \mathbf{f}(\mathbf{U}_i)$. To avoid a re-computation of the Jacobian, we replace $\frac{\partial \mathbf{f}(\mathbf{s}_i)}{\partial \mathbf{u}}$ by $\mathbf{J} = \frac{\partial \mathbf{f}(\mathbf{u}^n)}{\partial \mathbf{u}}$. Finally, to gain more freedom in the definition of the method, linear combinations of $\Delta t^n\,\mathbf{J}\,\mathbf{k}_i$ are added to the last term. Note that the linearization procedure can be interpreted as performing one Newton step at every stage of the DIRK method instead of a Newton loop. If instead of the exact Jacobian, an approximation $\mathbf{W} \approx \mathbf{J}$ is used, we obtain so called W-methods, which have additional order conditions [11].

We thus obtain an s-stage Rosenbrock method with coefficients a_{ij}, γ_{ij} and b_i in the form

$$
\begin{aligned}
\big(\mathbf{I} - \gamma_{ii}\, \Delta t^n\, \mathbf{W}\big)\, \mathbf{k}_i &= \mathbf{f}(\mathbf{s}_i) + \Delta t^n \mathbf{W} \sum_{j=1}^{i-1} \gamma_{ij}\, \mathbf{k}_j, \quad i = 1, \ldots, s \\
\mathbf{s}_i &= \mathbf{u}^n + \Delta t^n \sum_{j=1}^{i-1} a_{ij}\, \mathbf{k}_j, \quad i = 1, \ldots, s \\
\mathbf{u}^{n+1} &= \mathbf{u}^n + \Delta t^n \sum_{i=1}^{s} b_i\, \mathbf{k}_i.
\end{aligned} \tag{6}
$$

Here, the coefficients a_{ij} and b_i correspond to those of the DIRK method and the γ_{ii} are the diagonal coefficients of that, whereas the off-diagonal γ_{ij} are additional coefficients. Note that in the case of a non-autonomous equation, an additional term $\Delta t^n\, \gamma_i\, \partial_t\, \mathbf{f}\,(t_n, \mathbf{u}_n)$ would appear on the right hand side of (6), with $\gamma_i = \sum_j \gamma_{ij}$.

An efficient implementation of the Rosenbrock methods is used in order to circumvent the matrix-vector multiplication in (6). Further details can be found in [11]. The used coefficients for the different time integration methods can be found in the appendix.

2.3 Time adaptivity

An adaptive time stepping scheme is employed in order to be able to control the accuracy and to enhance the efficiency of the simulations. To this end, the user supplies a tolerance TOL and based on an estimate of the time integration error, a time step is chosen that is supposed to keep the time integration error below TOL. Compared to a fixed time step scheme this gives an estimate of the overall time integration error in the first place, as well as allowing to increase, respectively decrease the time step based on what happens in the flow.

Here, the H211PI controller as introduced by [37] is used to determine the next time step. An error estimate of the solution for the current time step is readily available with the embedded scheme of lower order of the various methods:

$$
\hat{\mathbf{l}} = \Delta t^n \sum_{j=1}^{s} (b_i - \hat{b}_i)\, \mathbf{k}_i. \tag{7}
$$

The next time step Δt^{n+1} is determined with

$$
\Delta t^{n+1} = \hat{\rho}^n\, \Delta t^n, \tag{8}
$$

with $\hat{\rho}^n$ given by the smooth limiter

$$
\hat{\rho}^n = 1 + \kappa\, \arctan\left(\frac{\rho^n - 1}{\kappa}\right). \tag{9}
$$

Choosing $\kappa = 2$ implies that the step size can be increased by a factor 4, and reduced by a factor 13. In [38] it is noted that the actual value of κ is not crucial. A larger value of κ may allow the step size to increase quick after the transients have decayed.

ρ^n is determined with the H211PI controller:

$$\rho^n = \|\mathbf{d}^n./\hat{\mathbf{l}}^n\|^{\beta_1}\|\mathbf{d}^{n-1}./\hat{\mathbf{l}}^{n-1}\|^{\beta_2}\left(\rho^{n-1}\right)^{-\zeta}, \tag{10}$$

where. denotes a point-wise division operator. The coefficients β_1, β_2 and ζ are set to $\beta_1 = \beta_2 = \frac{1}{4\hat{p}}$ and $\zeta = \frac{1}{4}$ due to the use of the H211PI controller. $\hat{p}$ represents the order of the embedded scheme of the Rosenbrock and (E)SDIRK methods. The vector $\mathbf{d}$ is defined by the fixed resolution test as discussed in [39]:

$$d_i = RTOL|u_i^n| + ATOL, \tag{11}$$

with $RTOL$ and $ATOL$ being user defined tolerances. Here, $RTOL$ is set to $RTOL = ATOL$, such that only one input parameter is required.

Step size rejections occur in case the inequality

$$\|\hat{\mathbf{l}}./\mathbf{d}\|_2 \leq 1, \tag{12}$$

is not met. The time step is repeated with the smaller time step as suggested by the H211PI controller.

The H211PI controller needs to be started with either a different controller or with equidistant time steps in case the first time step is not rejected. Here, the procedure is started with the classical controller:

$$\Delta t^{n+1} = \Delta t^n \|\hat{\mathbf{l}}./\mathbf{d}\|^{-1/\hat{p}}. \tag{13}$$

In order to compare the computational efficiency of different time integration schemes, it is desirable to scale and calibrate the tolerance of the adaptive time step size control algorithm. The control algorithm should run in a tolerance proportional mode: when the tolerance is changed by one order of magnitude, then the error of the solution should change by one order of magnitude. Also, the different time integration schemes should deliver the same accuracy for the same tolerance setting [38].

The scaling transformation

$$TOL' = \beta\, TOL_0^{\frac{\xi-1}{\xi}}\; TOL^{\frac{1}{\xi}} = \kappa\, TOL^{\frac{1}{\xi}} \tag{14}$$

with $\kappa = \beta\, TOL_0^{\frac{\xi-1}{\xi}}$, can be used to compare the computational efficiencies of the different integration schemes [38]. TOL' represents the tolerance parameter used by the adaptive step size control algorithm, TOL is the parameter specified by the user, TOL_0 is the equivalence point determined during the calibration, β is a constant to equally calibrate the different time integration schemes, and ξ is the measured order of the adaptive step size control algorithm of the reference computations for the calibration.

3 Solving nonlinear and linear equation systems

The SDIRK and ESDIRK schemes lead to nonlinear systems as shown in Eq. 2. To solve this equation, iterative methods are needed. See the textbooks [4, 16] for an overview of methods and theory.

3.1 Newton methods

In order to solve Eq. 2, we use Newton's method. This solves the root problem

$$\mathbf{F}(\mathbf{u}) = \mathbf{0} \tag{15}$$

for a differentiable function $\mathbf{F}(\mathbf{u})$. Here, we have $\mathbf{F}(\mathbf{u}) = \mathbf{u} - \mathbf{s}_i - \Delta t^n\, a_{ii}\, \mathbf{f}\,(\mathbf{u})$. The iteration is then

$$\begin{aligned} \left.\frac{\partial \mathbf{F}(\mathbf{u})}{\partial \mathbf{u}}\right|_{\mathbf{u}^{(k)}} \Delta \mathbf{u} &= -\mathbf{F}(\mathbf{u}^{(k)}) \\ \mathbf{u}^{(k+1)} &= \mathbf{u}^{(k)} + \Delta \mathbf{u}, \quad k = 0, 1, \ldots \end{aligned} \tag{16}$$

with some starting value $\mathbf{u}^{(0)}$. Here, we always choose $\mathbf{u}^{(0)} = \mathbf{s}_i$ from Eq. 4. More effective formulas for the starting values can be found in [1, 11, 17] where higher order approximations are used to form the initial guess. As termination criteria, we always use relative ones, where the residual serves as an estimate of the iteration error:

$$\|\mathbf{F}(\mathbf{u}^{(k+1)})\| \leq \tau \cdot \|\mathbf{F}(\mathbf{u}^{(0)})\|. \tag{17}$$

Here the tolerance τ needs to be chosen such that the error from the Newton iteration does not interfere with the error estimate in the adaptive time step. To this end, it is chosen 5 times more accurate than TOL, as suggested in [38], which avoids over solving while giving reliable time integration error estimates.

If the linear equation systems (16) are solved exactly, the method is locally second order convergent. Since an exact Jacobian is rarely available and the scheme is computationally expensive, other variants approximate terms in (16) and solve the linear systems only approximately. Here, the linear equation systems are solved by an iterative scheme. These schemes are called inexact Newton methods and have been analyzed in [8], where the inner solver is terminated if the relative residual of the linear system is below a certain threshold. This type of scheme can be written as:

$$\begin{aligned} \left\| \frac{\partial \mathbf{F}(\mathbf{u})}{\partial \mathbf{u}} |_{\mathbf{u}^{(k)}} \Delta \mathbf{u} + \mathbf{F}(\mathbf{u}^{(k)}) \right\| &\leq \eta_k \,\|\mathbf{F}(\mathbf{u}^{(k)})\| \\ \mathbf{u}^{(k+1)} &= \mathbf{u}^{(k)} + \Delta \mathbf{u}, \quad k = 0, 1, \ldots . \end{aligned} \tag{18}$$

The $\eta_k \in \mathbb{R}$ are called forcing terms. In [9], the choice for this sequence is discussed and it is proved that this scheme converges locally quadratic, if the forcing terms go to zero fast enough in a certain sense. However, it is not necessary to solve the first few linear systems very accurately. This is in line with the intuition that while we are far away from the solution, we do not need the optimal search direction for

Newton's method, just a reasonable one to get us in the generally right direction. A way of achieving this is the following:

$$\eta_k^A = \gamma \frac{\|\mathbf{F}(\mathbf{u}^{(k)})\|^2}{\|\mathbf{F}(\mathbf{u}^{(k-1)})\|^2}$$

with a parameter $\gamma \in (0, 1]$. We set $\eta_0 = \eta_{\max}$ for some $\eta_{\max} < 1$ and for $k > 0$:

$$\eta_k^B = \min(\eta_{\max}, \eta_k^A).$$

Furthermore, Eisenstat and Walker [9] suggest safeguards to avoid volatile decreases in η_k. To this end, $\gamma\eta_{k-1}^2 > 0.1$ is used as a condition to determine if η_{k-1} is rather large and thus the definition of η_k is refined to

$$\eta_k^C = \begin{cases} \eta_{\max}, & n = 0, \\ \min(\eta_{\max}, \eta_k^A), & n > 0, \gamma\eta_{k-1}^2 \leq 0.1 \\ \min(\eta_{\max}, \max(\eta_k^A, \gamma\eta_{k-1}^2)) & n > 0, \gamma\eta_{k-1}^2 > 0.1 \end{cases}$$

Finally, to avoid over solving in the final stages, Eisenstat and Walker suggest

$$\eta_k = \min(\eta_{\max}, \max(\eta_k^C, 0.5\tau/\|\mathbf{F}(\mathbf{u}^{(k)})\|)), \tag{19}$$

where τ is the tolerance at which the Newton iteration would terminate, see Eq. 17.

3.2 Jacobian-free GMRES

To solve the linear systems arising in the Newton scheme, and also the linear system present in the Rosenbrock scheme, we use GMRES(m), meaning restarted GMRES. There, the system matrix appears only in matrix vector products. Thus it is possible to formulate a Jacobian free version of Newton's method [18]. To this end, the matrix vector products $\mathbf{Ax}$ are replaced by a difference quotient via

$$\mathbf{Ax} = \frac{\partial \mathbf{F}(\mathbf{u})}{\partial \mathbf{u}}\mathbf{x} \approx \frac{\mathbf{F}(\mathbf{u} + \epsilon\mathbf{x}) - \mathbf{F}(\mathbf{u})}{\epsilon}. \tag{20}$$

A second order difference scheme could also be applied, but this comes at the expense of an extra function evaluation compared to Eq. 20.

If the parameter ϵ is chosen very small, the approximation becomes better, however, cancellation errors become a major problem. A simple choice for the parameter that avoids cancellation but still is moderately small is given by [25] as

$$\epsilon = \frac{\sqrt{eps}}{\|\mathbf{x}\|_2},$$

where *eps* is the machine accuracy.

As initial guess, we use the zero vector. The method terminates based on a relative criterion, namely

$$\left|\left|\mathbf{Ax}^k - \mathbf{b}\right|\right|_2 \leq \eta_k \, ||\mathbf{b}||_2 \,,$$

where $\mathbf{b}$ is the right hand side vector of the linear system. In the Newton case, the η_k are the forcing terms as just described, whereas in the Rosenbrock case, no theory is available that tells how to best choose the tolerance. We will discuss this in Section 5.1.1.

3.3 Preconditioning strategy

Two different preconditioning strategies are employed. For the first strategy, ILU(0) is employed as a preconditioner, and is refactored periodically after 30 time steps. Thereby, the factorization is based on the Jacobian corresponding to the first order discretization.

The second preconditioning strategy consists of a measure to automatically determine whether it is preferred to compute a new preconditioner [35]. The preconditioner update strategy is based on the principle that the accuracy of the preconditioner influences the number of iterations of the Krylov subspace solver. A new preconditioner is computed in case the total time spent on GMRES iterations is greater than the computational time necessary for the evaluation of the preconditioner.

The preconditioner freeze strategy in [35] is modified in the sense that computational times for rejected time steps are ignored by the algorithm. Also, the number of stages per time step differs per time integration scheme. Therefore, only the computational time needed for the first stage is used to determine whether it is necessary to update the preconditioner.

3.4 Summary of methodology

We now summarize the numerical method to demonstrate the interplay between all components, both the solver and the time integration method. Given an error tolerance TOL, a time t_n and time step size Δt^n one time step of an s-stage SDIRK method results in:

- For $i = 1, \dots, s$
  - For $k = 0, 1, \dots$ until termination criterion (17) with tolerance $\tau = TOL/5$ is satisfied or MAX_NEWTON_ITER has been reached
    - Solve linear system (16) using GMRES up to tolerance given by Eq. 19
  - If MAX_NEWTON_ITER has been reached, but Eq. 17 is not satisfied, repeat time step with $\Delta t^n = \Delta t^n/4$.
  - If the norm of any right hand side encountered is NaN, repeat time step with $\Delta t^n = \Delta t^n/4$.
- Estimate local error using Eq. 7 and compute new time step size Δt^{n+1} with Eq. 8
- $t^{n+1} = t_n + \Delta t^n$

For an ESDIRK method, the first stage becomes explicit, whereas for a ROW method, the nonlinear solve is replaced by a linear solve only. When repeating a time step, the division by four is heuristic. In our experiments a division by two often resulted in a situation where once this problem was encountered, every third time step was rejected for this reason. When dividing by four however, time integration continues smoothly. Thus, there is a feedback loop between the nonlinear iterations and the time step size, which in a way creates an upper bound on the time step. If a minimum time step is reached, the computation is aborted with an error message.

4 A nonlinear convection-diffusion equation

As a first test case, we consider a generalized nonlinear convection diffusion equation for two purposes: (1) investigate the effect of non-linearity on the accuracy of Rosenbrock schemes versus ESDIRK schemes, and (2) investigate the effect of a large condition number on the efficiency of the preconditioner. As the Rosenbrock schemes are linearly implicit, it is expected that their accuracy is less compared to ESDIRK schemes when solving a nonlinear problem. Also, by refining the time step the condition number of the linear system decreases which leads to better convergence of the linear solver. Which of these effects weighs the most and whether increased computational efficiency can be observed for Rosenbrock schemes is investigated for an academic problem.

4.1 Model problem

The governing equation for this problem is given by

$$\mathbf{u}_t = \boldsymbol{\beta}\mathbf{u}^{k_c} \cdot \nabla\mathbf{u} + \nabla \cdot (\mathbf{u}^{k_d}\nabla\mathbf{u}), \quad \mathbf{x} \in \Omega := (0, 1) \times (0, 1) \tag{21}$$

with Dirichlet boundary conditions $u = 1$ on the outer boundary $\partial\Omega$ and initial condition $u(x, y, 0) = u_0(x, y)$.

Here,

$$\boldsymbol{\beta} = \tilde{\beta}\begin{pmatrix}\sin\gamma\\ \cos\gamma\end{pmatrix}$$

with $\tilde{\beta} = 200$ the magnitude and $\gamma = 0.35\pi$ the angle of the direction of forced convection. Finally, the coefficients $k_c, k_d \in \mathbb{N}$ determine the degree of non-linearity. With $k_c = k_d = 0$, we obtain the linear convection diffusion equation, with $k_c = 1, k_d = 0$ the nonlinear convection diffusion equation often used as a model for the Navier-Stokes equations. For larger values, the strength of the non-linearity increases. As initial data we use the function that is one everywhere, except on the square $[0.2, 0.3] \times [0.2, 0.3]$, where the initial value is $1+\Delta u$, with the initial jump $\Delta u = 0.1$ for the baseline case, see Fig. 1.

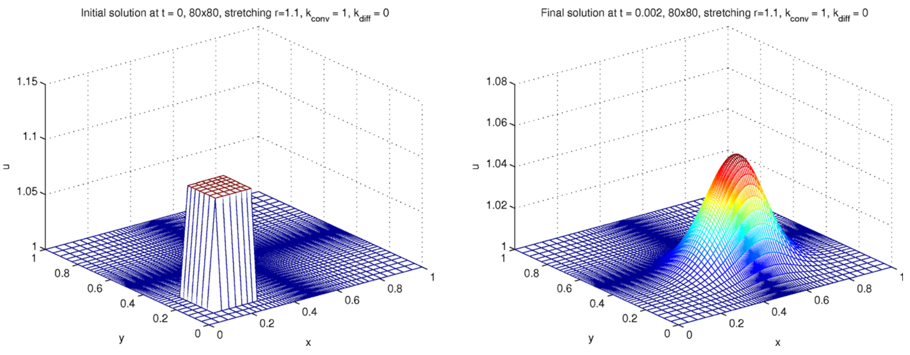


Fig. 1 Initial solution (*left*) and reference solution for $k_c = 1, k_d = 0$ at $t = 0.002$, obtained with ESDIRK5 and $\Delta t = 0.002/2^8$ (*right*), both on a 80×80 grid with stretching ratio $SR = 1.1$

We discretize this problem using finite differences, where we use first order upwind for the convective part and second order central differences for the diffusive part. The computational domain is discretized by $N \times N$ points with a stretching ratio SR to define the amount of stretching in the mesh. In the following test cases the stretching is equal in both x and y direction and clusters the nodes towards the center of the domain (see Fig. 1 for mesh generated with the baseline $SR = 1.1$). The stretching in the mesh is an easy way to increase the condition number of the system matrix for the investigation into preconditioner effectiveness.

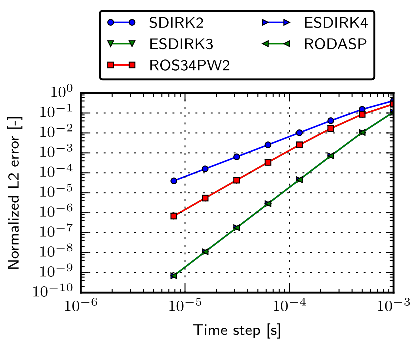
The error at the end of the simulation is computed with respect to a temporally exact solution which is obtained with a fifth order ESDIRK scheme and a time step of $\Delta t = 0.002/256$. The L_2-norm of the error is normalized by the L_2-norm of the difference between the temporally exact solution and the steady state solution (uniform field $u = 1$); for an L-stable time integration scheme the expected solution for $\Delta t \rightarrow \infty$ is the steady state solution, which is considered an error of 100 %.

4.2 Effect of non-linearity on accuracy

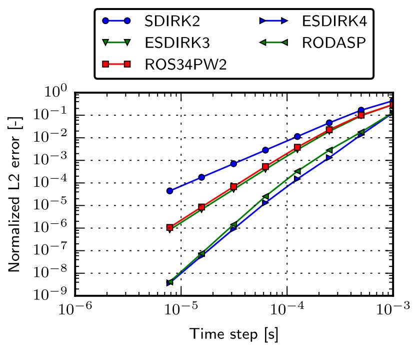
As a first investigation, the effect of non-linearity in the model problem on the (reduction of) accuracy of Rosenbrock schemes versus ESDIRK schemes is considered. To this end the error of the solution with respect to a temporally exact solution is compared for a range of time steps $\Delta t = 0.002/2^m$, $m = 1 \ldots 8$. The linear and nonlinear systems are solved up to the strict convergence tolerance 10^{-10} as we do not consider computational efficiency for this investigation. The non-linearity is varied from none (linear), baseline $k_c = 1$, $k_d = 0$, stronger nonlinear convection $k_c = 3$ to stronger non-linearities due to a larger variation in the solution u by prescribing a larger initial jump $\Delta u = 0.5$. It was chosen not to vary the non-linearity of the diffusion term as the ratio of 200 between convection and diffusion coefficients $\tilde{\beta}$ puts more emphasis on convection than diffusion.

The results for the different fixed time step sizes are shown in Fig. 2. In the linear case the chosen Rosenbrock and ESDIRK methods result in the same accuracy, therefore any observed differences in accuracy for the nonlinear cases is caused by the difference in dealing with non-linearities by either Rosenbrock or ESDIRK. For the baseline case, as shown in Fig. 2b, the non-linearity is not that strong and resembles the non-linearity observed in the convection term of the Navier-Stokes equations. The linearization of stages by the Rosenbrock schemes shows a small increase of the error, with the largest increase of about a factor 3 for the fourth order schemes. This means that the linearization error of Rosenbrock compared to ESDIRK methods for Navier-Stokes should be small and the schemes can be expected to be competitive.

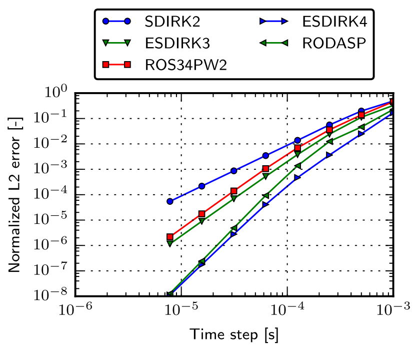
Increasing the nonlinear convection component to $k_c = 3$ in Fig. 2c reduces the accuracy of Rosenbrock schemes compared to ESDIRK schemes even further, especially for the third order schemes. Increasing the non-linearity by increasing the Δu jump in the initial condition, Fig. 2d, shows about the same effect as increasing the nonlinear convection component. Additionally, for the larger time steps instability was observed for the Rosenbrock schemes. This indicates that, although all methods possess L-stability, the nonlinear stability properties of the Rosenbrock schemes are reduced compared to their ESDIRK counterparts.



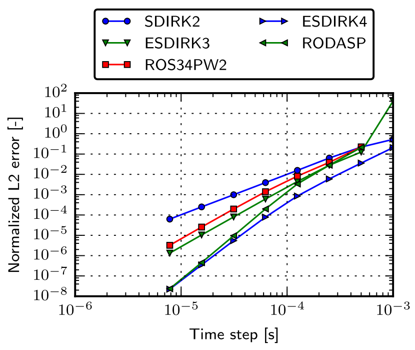
(a) Linear, $k_c = 0$, $k_d = 0$



(b) Baseline, $k_c = 1$, $k_d = 0$



(c) Increased nonlinear convection, $k_c = 3$, $k_d = 0$



(d) Increased initial jump $\Delta u = 0.5$, $k_c = 1$, $k_d = 0$

Fig. 2 Accuracy of Rosenbrock and ESDIRK schemes with varying non-linearity

4.3 Effect of mesh stretching on efficiency

In this section we wish to investigate the benefit of the constant stage matrix of the Rosenbrock schemes compared to the ESDIRK schemes when iteratively solving the system using preconditioned GMRES. It is expected that when the system becomes less well conditioned, the effectiveness of the preconditioner starts to play a more prominent role. Since stiffness can be introduced to the system by high aspect ratio cells, often encountered in boundary layers, the mesh stretching is adjusted from $SR = 1.0$ (uniform mesh) to $SR = 1.3$ (mesh with highly stretched cells).

An indication of the mesh properties and resulting condition numbers for the stage matrix $[I - \gamma \Delta t J]$ and preconditioned (ILU(0)) stage matrix are presented in Table 2. The condition numbers are determined for the baseline model $k_c = 1$, $k_d = 0$, on an 80×80 grid for the initial solution, the diagonal coefficient for the third order schemes $\gamma \approx 0.436$, and a time step of $\Delta t = 0.001$. Note that for this test case the exact Jacobian is used for both ESDIRK and Rosenbrock. Simulations are run with the iterative solution strategies aligned with the settings used for the more complicated problems: i.e. the ESDIRK schemes use the Eisenstat-Walker update strategy

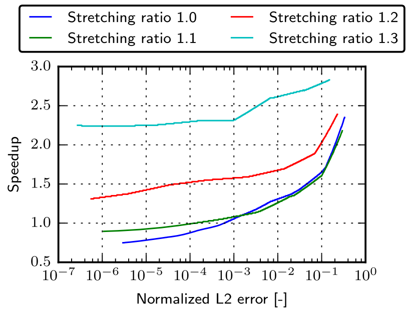
Table 2 Grid properties (stretching ratio SR and maximum aspect ratio AR) and stage matrix properties (condition number and preconditioned condition number)

SR	max. AR	cond. number	prec. cond. number
1.0	1	47	4.7
1.1	41	11670	216
1.2	1225	$11.6{\cdot}10^6$	1827
1.3	27784	$6.55{\cdot}10^9$	7150

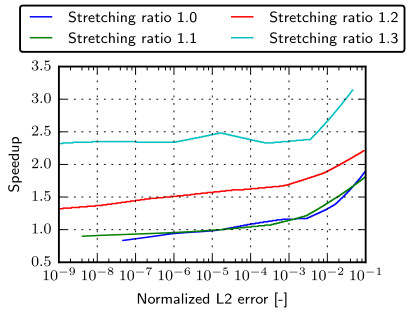
with a tolerance 10^{-10} and GMRES tolerance selection. For the Rosenbrock schemes the tolerance for GMRES is set to 10^{-10}. For this test case we do not use adaptive time stepping as the time scales over the simulation time are not varying much.

For the ESDIRK schemes, two update strategies for the preconditioner have been considered: (a) one update at the start of a new time step, (b) an update for every stage and Newton update. Although the second option results in less GMRES iterations, the amount of CPU time required was increased significantly compared to the first option. We therefore estimate the amount of work as the total number of GMRES iterations required for the simulation using update strategy (a) for ESDIRK.

Fixed time step simulations were run as before for a range of time steps $\Delta t = 0.002/2^m, m = 1\ldots 8$. From work-precision plots the speedup between the work required for Rosenbrock compared to ESDIRK was determined as the ratio between the required number of GMRES iterations for ESDIRK and Rosenbrock. The results of this investigation is shown in Fig. 3. The results show that the Rosenbrock schemes particularly have a benefit at lower precision (larger time steps) and higher stretching ratios. Both can be related to the conditioning of the stage matrix as both increase the condition number. The results support the idea that the preconditioner is more effective when the stage matrix does not change within the time step. For the highest stretching ratio the speedup of Rosenbrock can be as large as a factor of three for the largest times steps (lowest accuracies). For higher accuracies the speedup gradually reduces. For the better conditioned systems (with uniform mesh and stretching ratio



(a) ROS34PW2 speedup with respect to ESDIRK3



(b) RODASP speedup with respect to ESDIRK4

Fig. 3 Computational speedup in terms of GMRES iterations of Rosenbrock over ESDIRK schemes

of 1.1), the ESDIRK schemes become more efficient for errors below 10^{-4}. From these results it is concluded that the Rosenbrock schemes are expected to perform well in terms of efficiency in the presence of substantial stiffness. For large time steps, Rosenbrock schemes perform well in terms of efficiency but may suffer from instability, as shown in the previous section.

5 Numerical results for the Navier-Stokes equations

The Navier-Stokes equations are a second order system of conservation laws (mass, momentum, energy) modelling viscous compressible flow. Written in conservative variables density ρ, momentum $\mathbf{m} = (m_1, \ldots, m_d)^T$ and total energy per unit volume ρE:

$$\begin{aligned}
\partial_t \rho + \nabla \cdot \mathbf{m} &= 0, \\
\partial_t m_i + \sum_{j=1}^{d} \partial_{x_j}(m_i v_j + p\delta_{ij}) &= \frac{1}{Re}\sum_{j=1}^{d} \partial_{x_j} S_{ij}, \qquad i = 1, \ldots, d \\
\partial_t(\rho E) + \nabla \cdot (H\mathbf{m}) &= \frac{1}{Re}\sum_{j=1}^{d} \partial_{x_j}\left(\sum_{i=1}^{d} S_{ij} v_i - \frac{1}{\mathrm{Pr}} W_j\right).
\end{aligned}$$

Here, d stands for the number of dimensions, H for the total enthalpy, $\mathbf{S}$ represents the viscous shear stress tensor and W the heat flux. As the equations are dimensionless, the Reynolds number Re and the Prandtl number Pr appear. The equations are closed by the equation of state for the pressure $p = (\gamma - 1)\,\rho\, e$ introducing the internal energy per unit mass e, and the ratio of specific heats $\gamma = 1.4$. We assume an ideal gas.

5.1 Two dimensional flow around a cylinder

The first test case is a two dimensional flow around a circular cylinder [7]. The cylinder is held fixed in a uniform inflow, resulting in a vortex-street behind it. When the initial transient has disappeared, an unsteady periodic flow is present. This test case has been used in [2] and [43] to study the order of the ESDIRK schemes in comparison with BDF2.

The cylinder with diameter D is located in a fixed position in a uniform flow field with Mach number $M_\infty = 0.3$ and Reynolds number $Re_\infty = 1,000$, simulating a laminar flow. The radius of the cylinder is used as the characteristic length to determine the Reynolds number. The flow solver used for this test case is the commercial flow solver Hexstream, which is developed by NUMECA Int. A cell centered finite volume scheme is applied, with a second order central discretization with Jameson type scalar artificial dissipation [13] for the convective terms. The diffusive terms are discretized using second order central schemes. The same flow solver is also used for the three dimensional test case discussed in Section 5.3.

The computational domain consists of $2.5D$ upstream of the centre of the cylinder, $4.5D$ above and below the cylinder centre, and $16.5D$ downstream of the centre of the

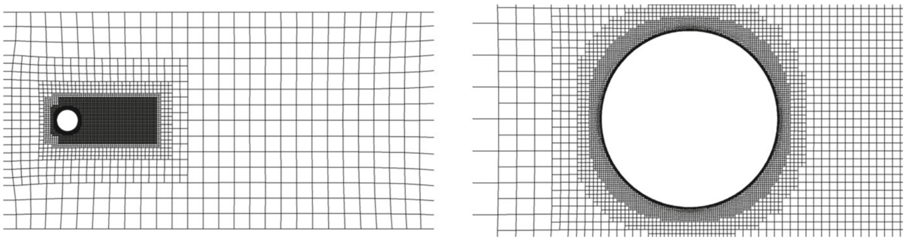


Fig. 4 Computational mesh used for the uniform flow around a circular cylinder case

cylinder. The mesh is refined in twelve steps to obtain a highly refined region close to the cylinder and in the wake downstream, resulting in an unstructured mesh with 10 608 cells. Close to the cylinder five extra layers of body conformal cells are generated resulting in an accurate representation of the boundary layer. The smallest cells which are located in the boundary layer, are of size $6.6 \cdot 10^{-5}\ D$. The maximum aspect ratio of the cells in the mesh is 6.3, and the minimum aspect ratio is 1.0. Refinement in the wake is performed, since the vortex street needs to be resolved accurately to obtain a good accuracy for the simulations. The generated mesh is shown in Fig. 4.

The second preconditioning strategy is employed with the ILU preconditioner. Thus, the preconditioner is kept frozen until the total computational time spent on Krylov subspace iterations with the current frozen preconditioner is less than the computational time needed for an evaluation of the preconditioner. The maximum number of Newton iterations is 40, and the maximal dimension of the Krylov subspace is 50. The resulting linear systems to be solved are of dimension 42 432.

5.1.1 Choice of GMRES tolerance for Rosenbrock time integration

Figures 5 and 6 show the results for varying the tolerance for the adaptive time step controller, as well as the tolerance in the linear solver for ROS34PW2 and RODASP. Thereby, we solve the linear systems up to the same tolerance or tolerance that are a factor of 10 or 100 more accurate. The reference solution is obtained with ESDIRK5 using a fixed time step $\Delta t = 3.125 \cdot 10^{-5}$.

When choosing the same tolerance for both time integration and linear solver, the overall error is significantly larger than for stricter tolerances in the linear solver and the efficiency is lower as well. This is because a large number of time steps is rejected. For RODASP, when comparing the variant with a factor of ten and 100, we see that the error of the computation does change, and the efficiency is about the same, for some cases a factor of two lower. We thus suggest to use the factor 10^{-2} in order to have a good compromise between robustness and computational efficiency. For ROS34PW2, we suggest to use the factor 10^{-1} using a similar rationale.

5.1.2 Effect of tolerance calibration on numerical accuracy

When the time step is selected with the adaptive time step controller, a large difference in accuracy for the same adaptive tolerance is observed between the applied

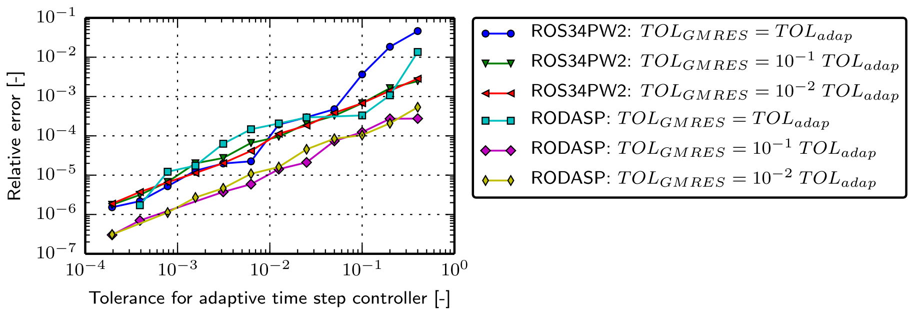


Fig. 5 Two-dimensional cylinder benchmark: effect of GMRES tolerance on numerical accuracy for Rosenbrock time integration schemes. Tolerance for the adaptive time step controller costs versus relative error is shown, i.e. the error is scaled by the L_2 norm of the reference solution

time integration schemes as shown in Fig. 7. The difference in accuracy is more than one order of magnitude between ESDIRK3 and RODASP4.

Relative large computational times are observed for the most inaccurate computations of ESDIRK3 and ESDIRK4. This is caused by the fact that many time steps are being rejected, which leads to an increase in computational time. The tolerance for the iterative solvers is not sufficient in order to meet the required tolerance for the adaptive time stepping controller.

To reduce the large difference in accuracy between the different time integration schemes, the tolerance calibration procedure (14) is now used. The coefficients used are shown in Table 3, and are determined by computing a linear fit through the data points of Fig. 7. Thus, the difference in accuracy for a given tolerance between the time integration schemes is essentially eliminated, as shown in Fig. 8. Furthermore, for coarse tolerances only the SDIRK2 scheme outperforms the other methods. For tighter tolerances, third order methods are better, which are in turn worse than fourth order methods. Finally, RODASP performs slightly better compared to ESDIRK4.

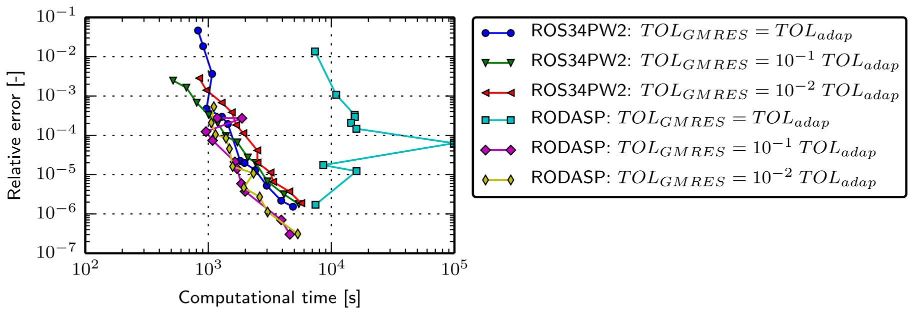


Fig. 6 Two-dimensional cylinder benchmark: effect of GMRES tolerance on numerical accuracy for Rosenbrock time integration schemes. Computational costs versus relative error is shown

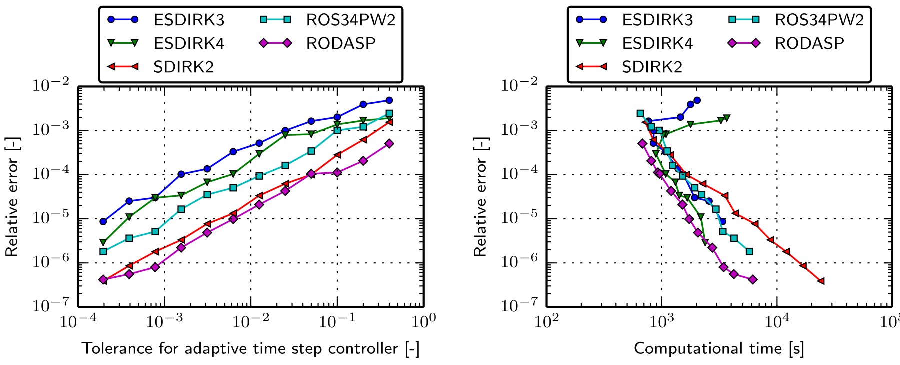


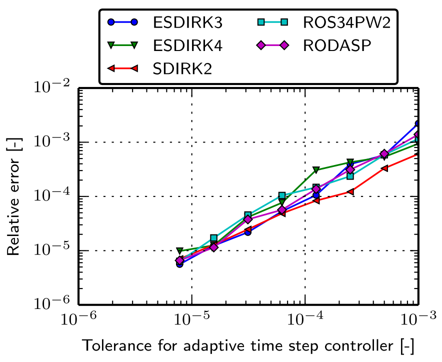
Fig. 7 Adaptive time stepping: accuracy and computational efficiency for different time integration schemes for the two dimensional flow around a cylinder test case

5.2 Two dimensional flow in cooling of flanged shaft

Another two dimensional case is considered, where we use a problem stemming from gas quenching [45]. In practice, high pressured air would be blown from two tubes at a flanged shaft in a cooling process. Here, we instead choose a low Mach number of 0.01 at the outlets of the tubes, to examine the effect of a different source of stiffness. The Reynolds number is 10,000. The grid is unstructured and has 142,052 triangular cells. It is illustrated in Fig. 9. Regarding initial conditions, the initial velocity is zero, the density 1.2 and the temperature of the gas is 300 K. We employ the code TEMPO, developed at the University of Kassel. In particular, a cell centered finite volume scheme with linear reconstruction and the Barth Jesperson limiter is employed [22]. As a flux function, we use L^2Roe, a recently designed low Mach low dissipation variant of the Roe flux [23]. The maximal number of Newton iterations is 30, the maximal dimension of the Krylov subspace is 40 and ILU preconditioning is used, with the ILU decomposition being updated every 25 time steps. The resulting linear systems are of dimension 568 208. As initial time step size, we choose $\Delta t = 3.27e-8$, which corresponds to a CFL number of 10. The computation runs until $2 \cdot 1e - 4$ seconds of real time, which is after the flow of air has been deflected at the shaft. To compute errors, a reference solution was obtained using RODASP with a tolerance of 10E-6. This solution is depicted in Fig. 10.

Table 3 Calibration coefficients determined with the two dimensional flow around a cylinder test case

Time integration scheme	ξ	κ
SDIRK2	1.06	286.73
ESDIRK3	0.83	56.28
ESDIRK4	0.85	120.24
ROS34PW2	0.97	148.81
RODASP	0.95	952.69



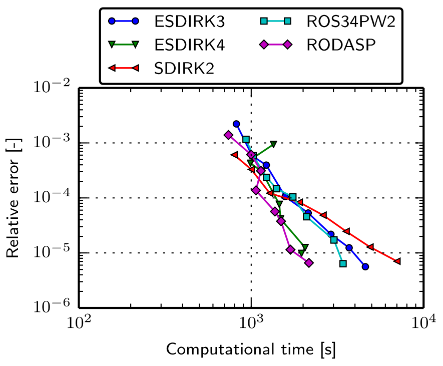
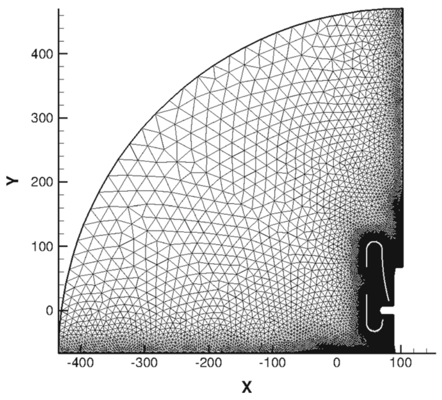


Fig. 8 Adaptive time stepping with tolerance calibration: accuracy and computational efficiency for different time integration schemes for the two dimensional flow around a cylinder test case

Furthermore, tolerance scaling is employed. Hereby, tolerance is scaled (κ) by 1/100 for RODASP, 1/1000 for ROS34PW2 and SDIRK2 and 1/10000 for ESDIRK3 and ESDIRK4, in all cases $\xi = 1$. The results of the simulations with calibration are shown in Fig. 11.

In this case, only the Rosenbrock schemes were able to provide a solution for all tolerances. With three exceptions at large tolerances, the ESDIRK schemes and SDIRK2 ended up at around $t = 1.7 \cdot 1e - 4$s in a situation where a right hand side is evaluated with NaN, causing the time step to be repeated with reduced time step size. However, this was repeated again and again, resulting in a stall of the computations with the time step converging to zero. The cause of this is not entirely clear and a reduction of the initial time step did not solve this problem.

Comparing ROS34PW2 and RODASP, we can see that ROS34PW2 performs better for most tolerances, but loses out for smaller tolerances due to the lower order.



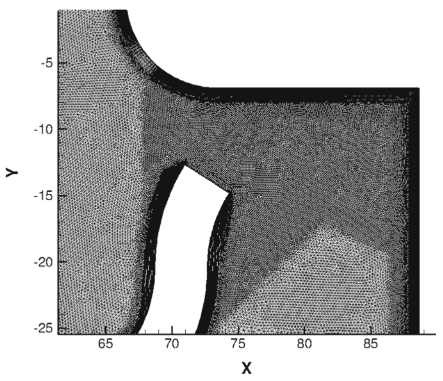


Fig. 9 Grid around flanged shaft. Left: Complete computational domain. Right: Zoom on region around lower tube and shaft

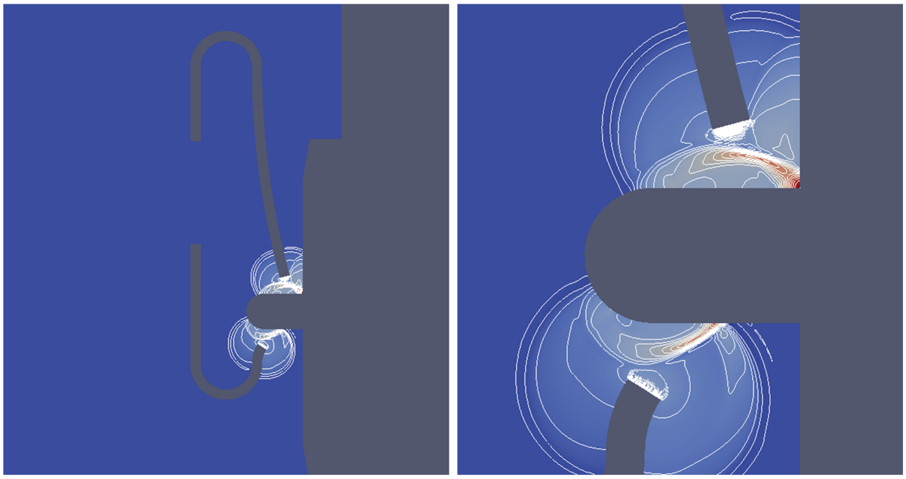


Fig. 10 Pressure contours after $2 \cdot 10^{-4}$ seconds. *Left*: Zoom on region around tubes. *Right*: Zoom on region around tube exits

5.3 Three dimensional flow around a square cylinder

Finally, we consider a three dimensional test case, namely the flow around a square cylinder as introduced in [21]. The Mach number is 0.3, and the square cylinder is rotated 45°. The Reynolds number for this test case is 300. A laminar flow is again simulated with the Hexstream compressible flow solver. The object is held fixed in a uniform flow field, resulting in a vortex-street behind the square cylinder. When the initial transient has disappeared after approximately 100 seconds, an unsteady periodic flow is present. We then compute 20 s of real time.

The computational domain consists of $20.5L$ upstream of the centre of the square cylinder, $11L$ above and below the centre, and $60.5L$ downstream of the centre of the square cylinder. Close to the square cylinder twenty extra layers of body conformal

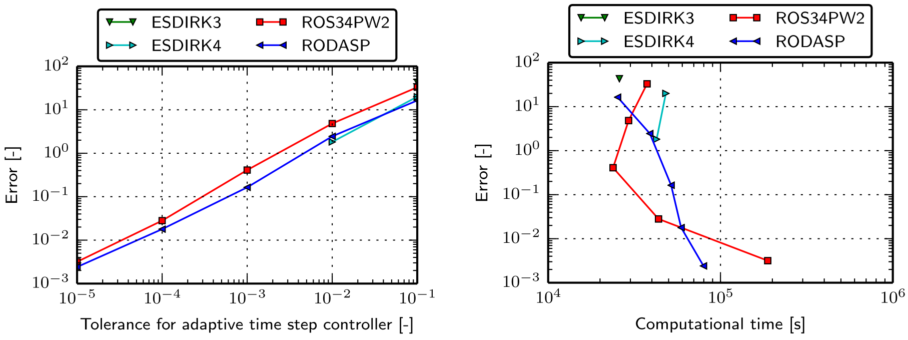


Fig. 11 Adaptive time stepping: accuracy and computational efficiency for different time integration schemes for the cooling of a flanged shaft test case

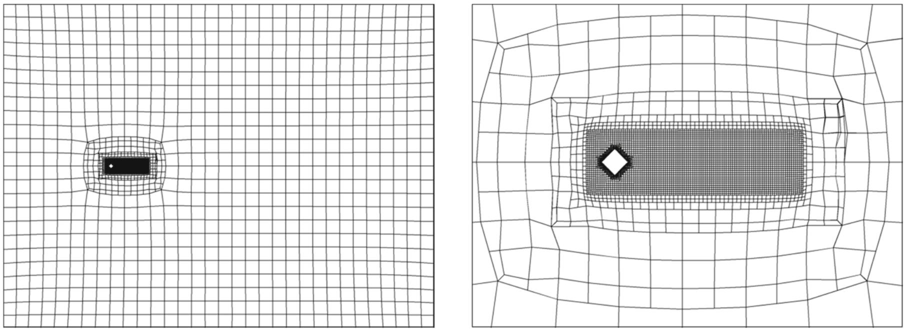


Fig. 12 Grid around a three dimensional square cylinder

cells are generated, resulting in cells with a maximum aspect ratio of 365. In the third dimension, the computational domain has length $5L$. The unstructured hexahedral mesh has 166 160 cells and 180 869 vertices, as shown in Fig. 12.

A series of computations is performed with the adaptive time stepping algorithm, and with the tolerance calibration applied reusing the coefficients shown in Table 3. Again, the second preconditioner update strategy with the ILU(1) preconditioner is employed to automatically update the preconditioner. The maximum number of Newton iterations is 40, and the maximal dimension of the Krylov subspace is 50. The resulting linear systems to be solved are of dimension 830 800. The reference solution is obtained with ESDIRK5 using a fixed time step $\Delta t = 0.0625\,s$. The error of a simulation is computed by taking the L_2 norm of the difference with the reference solution scaled by the L_2 norm of the reference solution.

As shown in Fig. 13 on the left, due to the tolerance calibration, the accuracies of the different time integration schemes are close to each other for equal adaptive tolerances. The figure on the right of Fig. 13 shows that the fourth order time integration schemes clearly outperform the third order schemes and SDIRK2 in terms of computational efficiency. ESDIRK4 outperforms RODASP slightly.

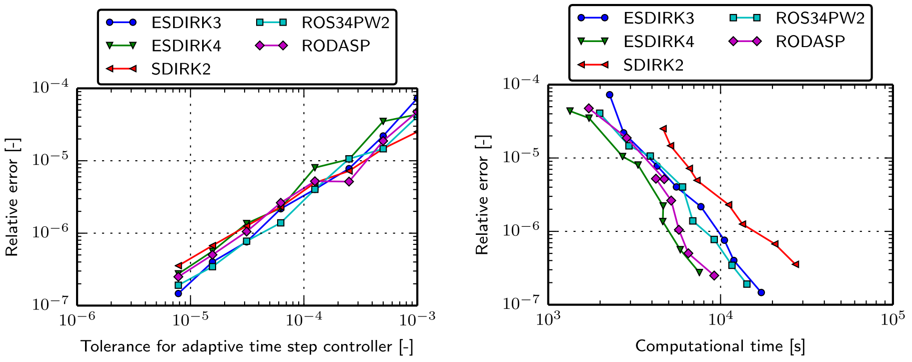


Fig. 13 Adaptive time stepping with tolerance calibration: accuracy and computational efficiency for different time integration schemes for the three dimensional flow around a square cylinder test case

6 Summary and conclusions

We considered finite volume discretizations of the time dependent compressible Navier-Stokes equations. Thereby, we compared time adaptive implicit and linearly implicit time integration schemes and judged them on computational efficiency. Specifically, we compared RODASP, ROS34PW2, ESDIRK3, ESDIRK4 and SDIRK2. The efficiency of these schemes is highly dependent on how the nonlinear and linear systems are solved and how the tolerances for the respective iterative solvers are chosen. Therefore, we used state of the art ILU preconditioned inexact Jacobian-free Newton-GMRES for the fully implicit methods and ILU preconditioned Jacobian-free GMRES for the Rosenbrock methods. For the ESDIRK methods, Eisenstat and Walker [9] give suggestions how to automatically set the tolerance for the linear solver resulting in a robust and efficient scheme. For the inexact Newton solver, we suggest to use TOL/5, which avoids oversolving, while giving sufficiently small iteration errors compared to time integration errors. Regarding the Rosenbrock methods, current theory does not provide us with a suggestion on how to choose the tolerance there. Based on experiments, our suggestion is to use TOL/100, resulting in in stable schemes and avoiding oversolving.

Nonlinear convection-diffusion simulations show that the loss in accuracy from the linearization for the Rosenbrock schemes is compensated by a more effective preconditioner for the constant stage matrix in particular for stiff problems and large tolerances.

For a proper use of the adaptive time stepping strategy in real applications, a calibration is employed. We then compared the different schemes for 2D and 3D test cases for different codes. Thereby, it shows that the finer the tolerance, the more higher order in the time integration method pays off. Thus, SDIRK2 is only interesting for very coarse tolerances. Otherwise, it is ESDIRK4 and RODASP that are the best schemes with the difference in efficiency between being small. However, RODASP is more robust with ESDIRK4 failing to complete some simulations. Since RODASP is also easier to implement, we recommend the use of this in an unsteady compressible CFD solver.

This statement might change depending on improvements of solvers and may depend on the test case considered. For ESDIRK, this can be an interpolation of stage solutions to get better initial guesses. We experimented with extrapolation methods, but have not found a scheme so far that is consistently better. In particular in the Rosenbrock case, basis vectors of GMRES might be reused in subsequent stages, since the matrix is constant and only the right hand sides differ. Again, we did not find a scheme that is consistently better yet.

Acknowledgment Financial support has been provided by the German Research Foundation (Deutsche Forschungsgemeinschaft - DFG) via the Sonderforschungsbereich Transregio 30, project C2.

Appendix: Coefficients for time integration schemes

Table 4 Set of coefficients for SDIRK2, where $\alpha = 1 - \sqrt{2}/2$ and $\hat{\alpha} = 2 - \frac{5}{4}\sqrt{2}$

α_{ii}	$=$	α			
α_{21}	$=$	$1 - \alpha$			
b_1	$=$	$1 - \alpha$	$\hat{b}_1$	$=$	$1 - \hat{\alpha}$
b_2	$=$	α	$\hat{b}_2$	$=$	$\hat{\alpha}$

Table 5 Set of coefficients for ESDIRK3

α_{ii}	$=$	$\frac{1767732205903}{4055673282236}$			
α_{21}	$=$	$\frac{1767732205903}{4055673282236}$	α_{31}	$=$	$\frac{2746238789719}{10658868560708}$
α_{32}	$=$	$-\frac{640167445237}{6845629431997}$	α_{41}	$=$	$\frac{1471266399579}{7840856788654}$
α_{42}	$=$	$-\frac{4482444167858}{7529755066697}$	α_{43}	$=$	$\frac{11266239266428}{11593286722821}$
b_1	$=$	$\frac{1471266399579}{7840856788654}$	$\hat{b}_1$	$=$	$\frac{2756255671327}{12835298489170}$
b_2	$=$	$-\frac{4482444167858}{7529755066697}$	$\hat{b}_2$	$=$	$-\frac{10771552573575}{22201958757719}$
b_3	$=$	$\frac{11266239266428}{11593286722821}$	$\hat{b}_3$	$=$	$\frac{9247589265047}{10645013368117}$
b_4	$=$	$\frac{1767732205903}{4055673282236}$	$\hat{b}_4$	$=$	$\frac{2193209047091}{5459859503100}$

Table 6 Set of coefficients for ESDIRK4

α_{ii}	$=$	$\frac{1}{4}$			
α_{21}	$=$	$\frac{1}{4}$	α_{31}	$=$	$\frac{8611}{62500}$
α_{32}	$=$	$-\frac{1743}{31250}$	α_{41}	$=$	$\frac{5012029}{34652500}$
α_{42}	$=$	$-\frac{654441}{2922500}$	α_{43}	$=$	$\frac{174375}{388108}$
α_{51}	$=$	$\frac{15267082809}{155376265600}$	α_{52}	$=$	$-\frac{71443401}{120774400}$
α_{53}	$=$	$\frac{730878875}{902184768}$	α_{54}	$=$	$\frac{2285395}{8070912}$
α_{61}	$=$	$\frac{82889}{524892}$	α_{62}	$=$	0
α_{63}	$=$	$\frac{15625}{83664}$	α_{64}	$=$	$\frac{69875}{102672}$
α_{65}	$=$	$-\frac{2260}{8211}$			
b_1	$=$	$\frac{82889}{524892}$	$\hat{b}_1$	$=$	$\frac{4586570599}{29645900160}$
b_2	$=$	0	$\hat{b}_2$	$=$	0
b_3	$=$	$\frac{15625}{83664}$	$\hat{b}_3$	$=$	$\frac{178811875}{945068544}$
b_4	$=$	$\frac{69875}{102672}$	$\hat{b}_4$	$=$	$\frac{814220225}{1159782912}$
b_5	$=$	$-\frac{2260}{8211}$	$\hat{b}_5$	$=$	$-\frac{3700637}{11593932}$
b_6	$=$	$\frac{1}{4}$	$\hat{b}_6$	$=$	$\frac{61727}{225920}$

Table 7 Set of coefficients for ROS34PW2

γ	=	$4.3586652150845900 \cdot 10^{-1}$			
α_{21}	=	$8.7173304301691801 \cdot 10^{-1}$	γ_{21}	=	$-8.7173304301691801 \cdot 10^{-1}$
α_{31}	=	$8.4457060015369423 \cdot 10^{-1}$	γ_{31}	=	$-9.0338057013044082 \cdot 10^{-1}$
α_{32}	=	$-1.1299064236484185 \cdot 10^{-1}$	γ_{32}	=	$5.4180672388095326 \cdot 10^{-2}$
α_{41}	=	$0.0000000000000000 \cdot 10^{+0}$	γ_{41}	=	$2.4212380706095346 \cdot 10^{-1}$
α_{42}	=	$0.0000000000000000 \cdot 10^{+0}$	γ_{42}	=	$-1.2232505839045147 \cdot 10^{+0}$
α_{43}	=	$1.0000000000000000 \cdot 10^{+0}$	γ_{43}	=	$5.4526025533510214 \cdot 10^{-1}$
b_1	=	$2.4212380706095346 \cdot 10^{-1}$	$\hat{b}_1$	=	$3.7810903145819369 \cdot 10^{-1}$
b_2	=	$-1.2232505839045147 \cdot 10^{+0}$	$\hat{b}_2$	=	$-9.6042292212423178 \cdot 10^{-2}$
b_3	=	$1.5452602553351020 \cdot 10^{+0}$	$\hat{b}_3$	=	$5.0000000000000000 \cdot 10^{-1}$
b_4	=	$4.3586652150845900 \cdot 10^{-1}$	$\hat{b}_4$	=	$2.1793326075422950 \cdot 10^{-1}$

Table 8 Set of coefficients for RODASP

γ	=	$2.5000000000 \cdot 10^{-1}$			
α_{21}	=	$7.5000000000 \cdot 10^{-1}$	γ_{21}	=	$-7.5000000000 \cdot 10^{-1}$
α_{31}	=	$8.6120400814 \cdot 10^{-2}$	γ_{31}	=	$-1.3551200000 \cdot 10^{-1}$
α_{32}	=	$1.2387959919 \cdot 10^{-1}$	γ_{32}	=	$-1.3799200000 \cdot 10^{-1}$
α_{41}	=	$7.7403453551 \cdot 10^{-1}$	γ_{41}	=	$-1.2560800000 \cdot 10^{+0}$
α_{42}	=	$1.4926515495 \cdot 10^{-1}$	γ_{42}	=	$-2.5014500000 \cdot 10^{-1}$
α_{43}	=	$-2.9419969046 \cdot 10^{-1}$	γ_{43}	=	$1.2209300000 \cdot 10^{+0}$
α_{51}	=	$5.3087466826 \cdot 10^{+0}$	γ_{51}	=	$-7.0731800000 \cdot 10^{+0}$
α_{52}	=	$1.3308921400 \cdot 10^{+0}$	γ_{52}	=	$-1.8056500000 \cdot 10^{+0}$
α_{53}	=	$-5.3741378117 \cdot 10^{+0}$	γ_{53}	=	$7.7438300000 \cdot 10^{+0}$
α_{54}	=	$-2.6550101103 \cdot 10^{-1}$	γ_{54}	=	$8.8500300000 \cdot 10^{-1}$
α_{61}	=	$-1.7644376488 \cdot 10^{+0}$	γ_{61}	=	$1.6840700000 \cdot 10^{+0}$
α_{62}	=	$-4.7475655721 \cdot 10^{-1}$	γ_{62}	=	$4.1826600000 \cdot 10^{-1}$
α_{63}	=	$2.3696918469 \cdot 10^{+0}$	γ_{63}	=	$-1.8814100000 \cdot 10^{+0}$
α_{64}	=	$6.1950235906 \cdot 10^{-1}$	γ_{64}	=	$-1.1378600000 \cdot 10^{-1}$
α_{65}	=	$2.5000000000 \cdot 10^{-1}$	γ_{65}	=	$-3.5714300000 \cdot 10^{-1}$
b_1	=	$-8.0368370789 \cdot 10^{-2}$	$\hat{b}_1$	=	$-1.7644376488 \cdot 10^{+0}$
b_2	=	$-5.6490613592 \cdot 10^{-2}$	$\hat{b}_2$	=	$-4.7475655721 \cdot 10^{-1}$
b_3	=	$4.8828563004 \cdot 10^{-1}$	$\hat{b}_3$	=	$2.3696918469 \cdot 10^{+0}$
b_4	=	$5.0571621148 \cdot 10^{-1}$	$\hat{b}_4$	=	$6.1950235906 \cdot 10^{-1}$
b_5	=	$-1.0714285714 \cdot 10^{-1}$	$\hat{b}_5$	=	$2.5000000000 \cdot 10^{-1}$
b_6	=	$2.5000000000 \cdot 10^{-1}$	$\hat{b}_5$	=	$0.0000000000 \cdot 10^{+0}$

References

1. Alexander, R.: Design and implementation of DIRK integrators for stiff systems. Appl. Numer. Math. **46**(1), 1–17 (2003)
2. Bijl, H., Carpenter, M.H., Vatsa, V.N., Kennedy, C.A.: Implicit time integration schemes for the unsteady compressible Navier-Stokes equations: Laminar flow. J. Comp. Phys. **179**, 313–329 (2002)
3. Birken, P.: Numerical simulation of tunnel fires using preconditioned finite volume schemes. ZAMP **59**, 416–433 (2008)
4. Birken, P.: Numerical Methods for the Unsteady Compressible Navier-Stokes Equations. Habilitation Thesis, University of Kassel (2012)
5. Birken, P.: Solving nonlinear systems inside implicit time integration schemes for unsteady viscous flows. In: Ansorge, R., Bijl, H., Meister, A., Sonar, T. (eds.) Recent Developments in the Numerics of Nonlinear Hyperbolic Conservation Laws, pp. 57–71. Springer (2013)
6. Colonius, T., Lele, S.K.: Computational aeroacoustics: progress on nonlinear problems of sound generation. Prog. Aerosp. Sci. **40**(6), 345–416 (2004). http://linkinghub.elsevier.com/retrieve/pii/S0376042104000570
7. Cox, J.S., Rumsey, C.L., Brentner, K.S., Younis, B.A.: Computation of vortex shedding and radiated sound for a circular cylinder. Theor. Comput. Fluid Dyn. **12**(4), 233–253 (1998)
8. Dembo, R., Eisenstat, R., Steihaug, T.: Inexact Newton methods. SIAM J. Numer. Anal. **19**, 400–408 (1982)
9. Eisenstat, S.C., Walker, H.F.: Choosing the forcing terms in an inexact newton method. SIAM J. Sci. Comput. **17**(1), 16–32 (1996)
10. Farhat, C.: CFD-based nonlinear computational aeroelasticity. In: Stein, E., de Borst, R., Hughes, T.J.R. (eds.) Encyclopedia of Computational Mechanics: Fluids, chap. 13, vol. 3, pp. 459–480. Wiley (2004)
11. Hairer, E., Wanner, G.: Solving Ordinary Differential Equations II, Series in edn. Springer, Berlin (2004)
12. Hindmarsh, A.C., Brown, P.N., Grant, K.E., Lee, S.L., Serban, R., Shumaker, D.E., Woodward, C.S.: SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. ACM TOMS **31**(3), 363–396 (2005)
13. Jameson, A.: Aerodynamics. In: Stein, E., de Borst, R., Hughes, T.J.R. (eds.) Encyclopedia of Computational Mechanics: Fluids, chap. 11, vol. 3, pp. 325–406. Wiley (2004)
14. John, V., Rang, J.: Adaptive time step control for the incompressible Navier-Stokes equations. Comp. Meth. Appl. Mech. Eng. **199**, 514–524 (2010)
15. Jothiprasad, G., Mavriplis, D.J., Caughey, D.A.: Higher-order time integration schemes for the unsteady Navier-Stokes equations on unstructured meshes. J. Comp. Phys. **191**, 542–566 (2003)
16. Kelley, C.T.: Iterative Methods for Linear and Nonlinear Equations. SIAM, Philadelphia (1995)
17. Kennedy, C.A., Carpenter, M.H.: Additive Runge-Kutta schemes for convection-diffusion-reaction equations. Appl. Num. Math. **44**, 139–181 (2003)
18. Knoll, D.A., Keyes, D.E.: Jacobian-free Newton-Krylov methods: A survey of approaches and applications. J. Comp. Phys. **193**, 357–397 (2004)
19. Kværnø, A.: Singly diagonally implicit Runge-Kutta methods with an explicit first stage. BIT Numer. Math. **44**(3), 489–502 (2004)
20. Lior, N.: The cooling process in gas quenching. J. Mater. Process. Technol. **155–156**, 1881–1888 (2004)
21. Lucas, P., Bijl, H., Van Zuijlen, A.H.: Efficient unsteady high Reynolds number flow computations on unstructured grids. Comput. Fluids **39**(2), 271–282 (2010)
22. Meister, A., Sonar, T.: Finite-volume schemes for compressible fluid flow. Surv. Math. Ind. **8**, 1–36 (1998)
23. Oßwald, K., Siegmund, A., Birken, P., Hannemann, V., Meister, A.: L2Roe: A low dissipation version of Roe's approximate Riemann solver for low Mach numbers. Int. J. Numer. Methods Fluids (2015). doi:10.1002/fld.4175

24. Pierce, N.A., Giles, M.B.: Preconditioned multigrid methods for compressible flow calculations on stretched meshes. Tech. rep. (1997)
25. Qin, N., Ludlow, D.K., Shaw, S.T.: A matrix-free preconditioned Newton/GMRES method for unsteady Navier-Stokes solutions. Int. J. Num. Meth. Fluids **33**, 223–248 (2000)
26. Rang, J.: A new stiffly accurate Rosenbrock-Wanner method for solving the incompressible Navier-Stokes equations. In: Ansorge, R., Bijl, H., Meister, A., Sonar, T. (eds.) Recent Developments in the Numerics of Nonlinear Hyperbolic Conservation Laws, Notes on Numerical Fluid Mechanics and Multidisciplinary Design, vol. 120, pp. 301–315. Springer (2013)
27. Rang, J.: An analysis of the Prothero–Robinson example for constructing new DIRK and ROW methods. J. Comput. Appl. Math. **262**, 105–114 (2014)
28. Rang, J.: Improved traditional Rosenbrock-Wanner methods for stiff ODEs and DAEs. J. Comput. Appl. Math. (2015)
29. Rang, J., Angermann, L.: New Rosenbrock W-methods of order 3 for partial differential algebraic equations of index 1. BIT **45**, 761–787 (2005)
30. Reisner, J., Mousseau, V., Wyszogrodzki, A., Knoll, D.A.: A fully implicit hurricane model with physics-based preconditioning. Mon. Weather Rev. **133**, 1003–1022 (2005)
31. Reisner, J., Wyszogrodzki, A., Mousseau, V., Knoll, D.: An efficient physics-based preconditioner for the fully implicit solution of small-scale thermally driven atmospheric flows. J. Comp. Phys. **189**(1), 30–44 (2003). doi:10.1016/S0021-9991(03)00198-0. http://linkinghub.elsevier.com/retrieve/pii/S0021999103001980
32. Reitsma, F., Strydom, G., de Haas, J.B.M., Ivanov, K., Tyobeka, B., Mphahlele, R., Downar, T.J., Seker, V., Gougar, H.D., Da Cruz, D.F., Sikik, U.E.: The PBMR steadystate and coupled kinetics core thermal-hydraulics benchmark test problems. Nucl. Eng. Des. **236**(5–6), 657–668 (2006)
33. Schmitt, B.A., Weiner, R.: Matrix-free W-methods using a multiple Arnoldi iteration. Appl. Num. Math. **18**, 307–320 (1995)
34. Shampine, L.F.: Numerical Solution of Ordinary Differential Equations. Springer (1994)
35. Silva, R.S., Almeida, R.C., Galeão, A.C.: A preconditioner freeze strategy for numerical solution of compressible flows. Commun. Numer. Methods Eng. **19**(3), 197–203 (2003)
36. Skvortsov, L.M.: Diagonally implicit Runge—Kutta methods for differential algebraic equations of indices two and three. Comput. Math. Math. Phys. **50**(6), 993–1005 (2010)
37. Söderlind, G.: Digital filters in adaptive time–stepping. ACM Trans. Math. Softw., 1–26 (2003)
38. Söderlind, G., Wang, L.: Adaptive time-stepping and computational stability. J. Comp. Appl. Math. **185**, 225–243 (2006). doi:10.1016/j.cam.2005.03.008
39. Söderlind, G., Wang, L.: Evaluating numerical ODE/DAE methods, algorithms and software. J. Comp. Appl. Math. **185**, 244–260 (2006). doi:10.1016/j.cam.2005.03.009
40. St-Cyr, A., Neckels, D.: A fully implicit jacobian-free high-order discontinuous Galerkin Mesoscale flow solver, pp. 243–252. Springer, Berlin, Heidelberg (2009)
41. Steinebach, G.: Order-reduction of ROW-methods for DAEs and method of lines applications, vol. Preprint 1741. Technische Universität Darmstadt (1995)
42. Strehmel, K., Weiner, R.: Linear-implizite Runge-Kutta-Methoden und ihre Anwendung. Teubner, Stuttgart (1992)
43. Van Zuijlen, A.H.: Fluid-Structure Interaction Simulations - Efficient Higher Order Time Integration of Partitioned Systems. Ph.D. thesis, Delft University of Technology (2006)
44. Wang, L., Mavriplis, D.J.: Implicit solution of the unsteady Euler equations for high-order accurate discontinuous Galerkin discretizations. J. Comp. Phys. **225**, 1994–2015 (2007)
45. Weidig, U., Saba, N., Steinhoff, K.: Massivumformprodukte mit funktional gradierten Eigenschaften durch eine differenzielle thermo-mechanische Prozessführung. WT-Online, pp. 745–752 (2007)
46. Weiner, R., Schmitt, B.A.: Order Results for Krylov-W-Methods. Computing **61**, 69–89 (1998)
47. Weiner, R., Schmitt, B.A., Podhaisky, H.: ROWMAP a ROW-code with Krylov techniques for large stiff ODEs. Appl. Num. Math. **25**, 303–319 (1997)
48. Williams, R., Burrage, K., Cameron, I., Kerr, M.: A four-stage index 2 Diagonally Implicit Runge–Kutta method. Appl. Numer. Math. **40**(3), 415–432 (2002)
49. Zahle, F., Soerensen, N.N., Johansen, J.: Wind turbine rotor-tower interaction using an incompressible overset grid method. Wind Energy **12**, 594–619 (2009). doi:10.1002/we.327