Comput. Methods Appl. Mech. Engrg. 199 (2010) 514–524

Contents lists available at ScienceDirect

Comput. Methods Appl. Mech. Engrg.

journal homepage: www.elsevier.com/locate/cma

Adaptive time step control for the incompressible Navier–Stokes equations

Volker John [a,b,*], Joachim Rang [c]

[a] Weierstrass Institute for Applied Analysis and Stochastics (WIAS), Mohrenstr. 39, 10117 Berlin, Germany
[b] Free University of Berlin, Department of Mathematics and Computer Science, Arnimallee 6, 14195 Berlin, Germany
[c] Institut für Wissenschaftliches Rechnen, Technische Universtät Braunschweig, 38092 Braunschweig, Germany

ARTICLE INFO

Article history:
Received 8 July 2009
Received in revised form 7 October 2009
Accepted 9 October 2009
Available online 21 October 2009

Keywords:
Incompressible Navier–Stokes equations
Implicit θ-schemes
DIRK methods
ROW methods
Adaptive time step control

ABSTRACT

Adaptive time stepping is an important tool in Computational Fluid Dynamics for controlling the accuracy of simulations and for enhancing their efficiency. This paper presents a systematic study of three classes of implicit and linearly implicit time stepping schemes with adaptive time step control applied to a 2D laminar flow around a cylinder: θ-schemes, diagonal-implicit Runge–Kutta (DIRK) methods and Rosenbrock–Wanner (ROW) methods. The time step is controlled using embedded methods. It is shown that several ROW methods clearly outperform the more standard θ-schemes and the DIRK methods. The results depend on a prescribed tolerance in the time step control algorithm, whose appropriate choice varies from scheme to scheme.

1. Introduction

Let $\Omega \subset \mathbb{R}^d$, $d \in \{2,3\}$, be a bounded domain and $T > 0$. The motion of incompressible flows is modeled by the incompressible Navier–Stokes equations, which are given in dimensionless form by

$$\begin{aligned} &\mathbf{u}_t - Re^{-1}\Delta\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p = \mathbf{f} \quad \text{in } (0,T]\times\Omega, \\ &\nabla\cdot\mathbf{u} = 0 \quad \text{in } [0,T]\times\Omega. \end{aligned} \tag{1}$$

Here, $\mathbf{u}$ is the velocity, p the pressure, $\mathbf{f}$ represents body forces and the parameter Re is the Reynolds number. The system of Eq. (1) has to be closed with appropriate initial and boundary conditions. If Dirichlet conditions are prescribed on the whole boundary $\partial\Omega$, a condition for the pressure, like $\int_\Omega p(\mathbf{x})\,d\mathbf{x} = 0$, has to be added. The accurate and fast solution of the Navier–Stokes equations is the core of many numerical simulations of complex processes in nature and industry.

This paper considers the simulation of time-dependent laminar flows. Thus, discretizations in space and time as well as a linearization for solving the nonlinear problem in each discrete time are required. There are many possible approaches, see [7] for a comprehensive presentation, and the question of optimal methods is still an active field of research. With respect to the spatial discretization, we will use an inf–sup stable finite element method [6]. It has been demonstrated in a number of numerical studies, e.g. in [27,15,10,12], that the pair of second order velocity Q_2 and first order discontinuous pressure P_1^{disc} on quadrilateral and hexahedral meshes is among the best performing finite element methods. Thus, the Q_2/P_1^{disc} finite element is a popular choice if finite element methods are used in the simulation of incompressible flows [7]. Concerning the linearization, a fixed point approach will be used in this paper, which has been proven to be more efficient than a Newton method in [13].

The topic of the paper is the temporal discretization of the incompressible Navier–Stokes equations. By far the most simulations of incompressible flows use explicit schemes or simple implicit schemes, like the backward Euler scheme, the Crank–Nicolson scheme or the fractional-step θ-scheme. We will concentrate in this paper on implicit and linearly implicit schemes, which are appropriate for laminar flow simulations and which avoid the nasty CFL condition. The study [17] showed that for obtaining accurate results at least a second order time stepping scheme is necessary. For this reason, only schemes with at least this accuracy will be considered.

The main focus of this paper is on an adaptive time step control for implicit and linearly implicit schemes. An adaptive time step control may undoubtedly improve the accuracy and efficiency of incompressible flow simulations substantially. However, with the simple implicit schemes commonly used, an efficient time step control is hard to achieve. For this reason, simulations with implicit temporal discretizations and adaptive time step control are rather rare in the literature. In [31], a strategy for controlling the length of the time step with θ-schemes has been proposed. This approach compares the results of the fractional-step θ-scheme and the

* Corresponding author. Address: Weierstrass Institute for Applied Analysis and Stochastics (WIAS), Mohrenstr. 39, 10117 Berlin, Germany. Tel.: +49 30 20372561; fax: +49 30 2044975.
E-mail addresses: john@wias-berlin.de (V. John), j.rang@tu-bs.de (J. Rang).

0045-7825/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.cma.2009.10.005

Crank–Nicolson scheme. These schemes have a different constant in the leading term of their error expansions. This difference, together with the difference of the results obtained with both schemes, can be used to estimate an appropriate length of the time step. The main drawback is the high computational effort of this approach. The step with the Crank–Nicolson scheme is used only to determine the size of the next time step. The costs of this step are of a similar order as for the fractional-step θ-scheme. Thus, the adaptive time step control increases the costs per time step by almost a factor of two. A simple approach is considered in [1], where the time step in a semi-implicit Euler scheme is chosen on the basis of comparing the change of the solution of two subsequent time steps in the L^2-norm of the space–time interval. Another possibility offer predictor–corrector schemes, for example the Adams–Bashforth method combined with the Crank–Nicolson scheme, see [7]. An adaptive time step control based on embedding techniques is presented in [30], but not studied in detail.

The embedding technique requires the use of more sophisticated time stepping schemes. Such schemes will be studied in this paper: diagonally implicit Runge–Kutta methods and linearly implicit Runge–Kutta methods (Rosenbrock–Wanner methods (ROW methods)). Both classes allow the computation of a second numerical solution with almost the same coefficients such that an effective time step control can be achieved [18,9]. We are not aware of any systematic studies of (linearly) implicit time stepping schemes with adaptive time step control for solving the incompressible Navier–Stokes equations.

An adaptive time step control needs some error indicator or estimator on which the determination of the next time step is based. This error estimator suggests a new time step size to reach a given accuracy. If the time step size is too small then a lot of unnecessary computational work has to be done. Otherwise, if the time step size is too large, the results may become too inaccurate. The computational studies will consider the flow around a cylinder. Besides standard estimators, which estimate the error for a certain size of the time step in norms of function spaces, also error indicators for outputs of interest, like drag and lift coefficient, might be interesting in this example. Here, in our first study, we will restrict to a standard estimator. The incorporation of indicators for outputs of interest will be postponed to forthcoming studies.

The paper is structured as follows. First, we give a short presentation of the spatial discretization of the incompressible Navier–Stokes equations. Then, the θ-schemes, the DIRK and ROW methods are introduced and their application to the incompressible Navier–Stokes equations is explained as well as the control of the time step. In Section 6, the numerical studies at a 2D laminar flow around a cylinder are presented. Finally, the most important observations are summarized and an outlook is given.

2. The finite element discretization in space

For simplicity of presentation, we consider the case that (1) is equipped with homogeneous Dirichlet boundary conditions in $[0,T]$. Then, the velocity ansatz space and test space can be chosen the same in the weak formulation of (1) as well as in the finite element method. Let $V = \left(H_0^1(\Omega)\right)^d$, $Q = L_0^2(\Omega)$, then the time-continuous weak or variational problem reads as follows: find $(\mathbf{u},p) \in V \times Q$ such that

$$(\mathbf{u}_t,\mathbf{v}) + (Re^{-1}\nabla\mathbf{u},\nabla\mathbf{v}) + ((\mathbf{u}\cdot\nabla)\mathbf{u},\mathbf{v}) - (p,\nabla\cdot\mathbf{v}) = (\mathbf{f},\mathbf{v}) \quad \forall \mathbf{v}\in V,$$
$$(\nabla\cdot\mathbf{u},q) = 0 \quad \forall q \in Q. \tag{2}$$

The symbol $(\cdot,\cdot)$ denotes the inner product in $(L^2(\Omega))^d$ and $(L^2(\Omega))^{d\times d}$, $d\in\{1,2,3\}$.

Finite element methods are a widely used approach for discretizing (a linearization of) (2) in space [7]. The unique solvability of the arising discrete (linear) systems requires that the velocity finite element space V_h is sufficiently large compared to the pressure finite element space Q_h, which is mathematically formulated with the inf–sup condition [6]

$$\inf_{q_h\in Q_h}\sup_{\mathbf{v}_h\in V_h}\frac{(q_h,\nabla\cdot\mathbf{v}_h)}{\|q_h\|_{L^2}\|\nabla\mathbf{v}_h\|_{L^2}} \geqslant \beta > 0.$$

Here, conforming finite element spaces will be considered to avoid technical difficulties in the presentation of the methods, i.e., $V^h \subset V$ and $Q^h \subset Q$ are assumed. The space-discretized Navier–Stokes equations read as follows: find $(\mathbf{u}_h,p_h)\in V_h\times Q_h$ such that

$$\begin{aligned}&(\dot{\mathbf{u}}_h,\mathbf{v}_h) + (Re^{-1}\nabla\mathbf{u}_h,\nabla\mathbf{v}_h) + ((\mathbf{u}_h\cdot\nabla)\mathbf{u}_h,\mathbf{v}_h)\\ &\quad - (p_h,\nabla\cdot\mathbf{v}_h) = (\mathbf{f},\mathbf{v}_h) \quad \forall\mathbf{v}_h\in V_h,\\ &(\nabla\cdot\mathbf{u}_h,q_h) = 0 \quad \forall q_h\in Q_h,\end{aligned} \tag{3}$$

where the dot denotes the temporal derivative. Let the space V_h be equipped with the basis

$$\{\boldsymbol{\phi}_i\}_{i=1}^{dN_u} = \left\{\begin{pmatrix}\varphi_i\\0\\\vdots\end{pmatrix}\right\}_{i=1}^{N_u} \cup \left\{\begin{pmatrix}0\\\varphi_i\\\vdots\end{pmatrix}\right\}_{i=1}^{N_u} \cup \cdots$$

and the space Q_h with the basis $\{\psi_i\}_{i=1}^{N_p}$. Here, N_u is the number of degrees of freedom for each component of the velocity and N_p is the number of degrees of freedom for the pressure. Then, the solution of (3) can be written in the form:

$$\mathbf{u}_h(t,\mathbf{x}) = \sum_{i=1}^{dN_u} u_i(t)\boldsymbol{\phi}_i(\mathbf{x}), \quad p_h(t,\mathbf{x}) = \sum_{i=1}^{N_p} p_i(t)\psi_i(\mathbf{x}),$$

with the unknown vectors of coefficients $\mathbf{u}_h := \mathbf{u}_h(t) = (u_1(t),\ldots,u_{dN_u}(t))^T$, $\mathbf{p}_h := \mathbf{p}_h(t) = (p_1(t),\ldots,p_{N_p}(t))^T$. For shortness, the algebraic objects will be given for the two-dimensional case. The extension to three dimensions is straightforward. The superscript (k) denotes the k-th component of a vector-valued function. Then, the following matrices and vectors are defined

$$\begin{aligned}&(M)_{ij} = (\varphi_j,\varphi_i), \quad i,\ j = 1,\ldots,N_u,\\ &(A(\mathbf{u}_h))_{ij} = Re^{-1}(\nabla\varphi_j,\nabla\varphi_i)\\ &\qquad + \left(\mathbf{u}_h^{(1)}\partial_x\varphi_j + \mathbf{u}_h^{(2)}\partial_y\varphi_j,\varphi_i\right), \quad i,\ j = 1,\ldots,N_u,\\ &(B_1)_{ij} = -(\partial_x\varphi_i,\psi_j), \quad i = 1,\ldots,N_u,\ j = 1,\ldots,N_p,\\ &(B_2)_{ij} = -(\partial_y\varphi_i,\psi_j), \quad i = 1,\ldots,N_u,\ j = 1,\ldots,N_p,\\ &(\mathbf{f}_h^{(k)})_i = (\mathbf{f}^{(k)},\varphi_i), \quad i = 1,\ldots,N_u,\ k = 1,2.\end{aligned} \tag{4}$$

This leads to the following algebraic analog of (3)

$$\begin{pmatrix}M&0&0\\0&M&0\\0&0&0\end{pmatrix}\begin{pmatrix}\dot{\mathbf{u}}_h^{(1)}(t)\\\dot{\mathbf{u}}_h^{(2)}(t)\\\dot{\mathbf{p}}_h(t)\end{pmatrix} = \begin{pmatrix}\mathbf{f}_h^{(1)}\\\mathbf{f}_h^{(2)}\\\mathbf{0}\end{pmatrix} - \begin{pmatrix}A(\mathbf{u}_h)&0&B_1\\0&A(\mathbf{u}_h)&B_2\\B_1^T&B_2^T&0\end{pmatrix}\begin{pmatrix}\mathbf{u}_h^{(1)}\\\mathbf{u}_h^{(2)}\\\mathbf{p}_h\end{pmatrix}.$$

This is a system of differential algebraic equations (DAE). The matrix M is called mass matrix and the matrix A stiffness matrix. All matrices, M, A, $\left(B_1^T,B_2^T\right)$, possess full rank.

This paper will study one-step schemes for the temporal discretization of (3). Let t_{m+1} denote the new discrete time, t_m the previous discrete time and $\tau_m = t_{m+1} - t_m$, $m = 0,1,2,\ldots$ the length of the time step. The initial time is $t_0 = 0$. For simplicity of notation, we omit in the following the index h. Moreover, we denote by $\mathbf{u}_m = \left(\mathbf{u}_m^{(1)},\mathbf{u}_m^{(2)}\right)^T$ the numerical approximation of the solution $\mathbf{u}(t_m)$.

3. θ-Schemes

3.1. Application to ordinary differential equations (ODEs)

Consider the following ODE:

$$M\dot{\mathbf{u}} = \mathbf{F}(t,\mathbf{u}) + \mathbf{F}_1(t), \quad \mathbf{u}(0) = \mathbf{u}_0. \tag{5}$$

A θ-scheme for solving (5) has the form:

$$\mathbf{u}_{m+1} = \mathbf{u}_m + \tau_m(\theta_1\mathbf{F}(t_{m+1},\mathbf{u}_{m+1}) + \theta_2\mathbf{F}(t_m,\mathbf{u}_m) + \theta_3\mathbf{F}_1(t_m) + \theta_4\mathbf{F}_1(t_{m+1})).$$

3.2. Application to a the Navier–Stokes equations

The application of a θ-scheme to the semi-discretized Navier–Stokes equations (3) leads to the following algebraic system:

$$\begin{pmatrix} M+\tau_m\theta_1 A(\mathbf{u}_{m+1}) & 0 & \tau_m\theta_1 B_1 \\ 0 & M+\tau_m\theta_1 A(\mathbf{u}_{m+1}) & \tau_m\theta_1 B_2 \\ B_1^T & B_2^T & 0 \end{pmatrix} \begin{pmatrix} \mathbf{u}_{m+1}^{(1)} \\ \mathbf{u}_{m+1}^{(2)} \\ \mathbf{p}_{m+1} \end{pmatrix} = \begin{pmatrix} (M-\tau_m\theta_2 A(\mathbf{u}_m))\mathbf{u}_m^{(1)} - \tau_m\theta_2 B_1\mathbf{p}_m + \tau_m\theta_3\mathbf{f}_m^{(1)} + \tau_m\theta_4\mathbf{f}_{m+1}^{(1)} \\ (M-\tau_m\theta_2 A(\mathbf{u}_m))\mathbf{u}_m^{(2)} - \tau_m\theta_2 B_2\mathbf{p}_m + \tau_m\theta_3\mathbf{f}_m^{(2)} + \tau_m\theta_4\mathbf{f}_{m+1}^{(2)} \\ \mathbf{0} \end{pmatrix}, \tag{6}$$

where $\mathbf{f}_m$ denotes here the vector which is obtained by testing the right hand side of the Navier–Stokes equations at t_m with finite element test functions. This class of schemes are also called pressure-corrected θ-schemes [24]. If $\theta_2 \neq 0$, then this approach requires the pressure from time t_m for the computation of the solution at time t_{m+1}. In particular, an initial pressure for the Navier–Stokes equations has to be defined. To circumvent this difficulty, often an inconsistent treatment of the pressure is applied

$$\begin{pmatrix} M+\tau_m\theta_1 A(\mathbf{u}_{m+1}) & 0 & \tau_m B_1 \\ 0 & M+\tau_m\theta_1 A(\mathbf{u}_{m+1}) & \tau_m B_2 \\ B_1^T & B_2^T & 0 \end{pmatrix} \begin{pmatrix} \mathbf{u}_{m+1}^{(1)} \\ \mathbf{u}_{m+1}^{(2)} \\ \mathbf{p}_{m+1} \end{pmatrix} = \begin{pmatrix} (M-\tau_m\theta_2 A(\mathbf{u}_m))\mathbf{u}_m^{(1)} + \tau_m\theta_3\mathbf{f}_m^{(1)} + \tau_m\theta_4\mathbf{f}_{m+1}^{(1)} \\ (M-\tau_m\theta_2 A(\mathbf{u}_m))\mathbf{u}_m^{(2)} + \tau_m\theta_3\mathbf{f}_m^{(2)} + \tau_m\theta_4\mathbf{f}_{m+1}^{(2)} \\ \mathbf{0} \end{pmatrix},$$

see [11] for a detailed discussion of this way to discretize the pressure in time.

3.3. Studied methods

We will consider only implicit second order pressure-corrected θ-schemes. Such schemes are the Crank–Nicolson scheme (CN), given by $\theta_i = 0.5,\ i = 1,\dots,4$, and the fractional-step θ-scheme (FS), which is given in detail in Section 4. In particular, the use of the Crank–Nicolson scheme is quite popular, see [31]. It has been shown in [12,17] that the application of the first order backward Euler scheme ($\theta_1 = \theta_4 = 1,\ \theta_2 = \theta_3 = 0$) leads to a considerable loss in accuracy in comparison to the second order θ-schemes. For this reason, the backward Euler scheme will not be considered in this paper. For the definition of an initial pressure, we refer to Section 4.

3.4. Adaptive time step control

A strategy for controlling the length of the time step adaptively with θ-schemes has been proposed in [31], see Section 1 for details. Because of the high computational costs, we do not consider this approach in our studies. Instead, comparisons with the Crank–Nicolson scheme and equidistant time steps (CN) will be included in the numerical studies.

4. DIRK schemes

4.1. Application to ODEs

Consider now an ODE of the form:

$$M\dot{\mathbf{u}} = \mathbf{F}(t,\mathbf{u}), \mathbf{u}(0) = \mathbf{u}_0. \tag{7}$$

Let $s \in \mathbb{N}$. An s-stage Runge–Kutta method (RK method) [9,29], is a one-step-method for solving (7) given by

$$M\mathbf{k}_i = \mathbf{F}(t_m + c_i\tau_m, \mathbf{U}_i), \quad \mathbf{U}_i = \mathbf{u}_m + \tau_m\sum_{j=1}^{s} a_{ij}\mathbf{k}_j,\ i = 1,\dots,s,$$

$$\mathbf{u}_{m+1} = \mathbf{u}_m + \tau_m\sum_{i=1}^{s} b_i\mathbf{k}_i.$$

The coefficients of an RK method are usually represented with the help of a Butcher table [3],

$$\begin{array}{c|ccc} c_1 & a_{11} & \cdots & a_{1s} \\ c_2 & a_{21} & \cdots & a_{2s} \\ \vdots & \vdots & & \vdots \\ c_s & a_{s1} & \cdots & a_{ss} \\ \hline & b_1 & \cdots & b_s \end{array} = \begin{array}{c|c} \mathbf{c} & A \\ \hline & \mathbf{b}^T \end{array}.$$

The value s is called number of stages. The vector $\mathbf{c}$ includes the grid points of the time discretization and $\mathbf{b}$ is vector with weights. The coefficients $a_{ij},\ b_i$ and c_i should be chosen in such a way that certain conditions are satisfied to obtain a sufficient consistency order [9,29].

In the numerical studies, only RK methods with $s \geqslant 2$ and with coefficients satisfying

$$(H1): a_{ij} = 0, \quad i < j,\ i,j \in \{1,\dots,s\},$$
$$(H2): a_{11} = 0,$$
$$(H3): a_{ii} \neq 0, \quad i \in \{2,\dots,s\},$$
$$(H4): b_i = a_{si}, \quad i \in \{1,\dots,s\},$$

will be studied. RK methods satisfying $(H1)$ are called diagonal-implicit RK methods (DIRK methods). An RK method satisfying $a_{si} = b_i$, i.e., $(H4)$, and $c_s = 1$ is called stiffly accurate. This is an essential property for applying this method for solving DAEs since it guarantees that the index-1 constraints are satisfied for the numerical solution. An RK method satisfying $a_{11} \neq 0$ and $(H1)$ has at most stage order $q = 1$, see [23]. The stage order can be improved if $a_{11} = 0$, i.e., $(H2)$ is satisfied. In this case an RK method satisfying $(H1)$, $(H2)$ and $(H3)$ has at most stage order $q = 2$ [23].

4.2. Application to DAEs of index 2

Consider the DAE

$$M\dot{\mathbf{u}} = \mathbf{F}(t,\mathbf{u},\mathbf{p}), \tag{8}$$
$$0 = \mathbf{G}(t,\mathbf{u}). \tag{9}$$

We will assume in the following that the matrix $\partial_{\mathbf{u}}\mathbf{G}M^{-1}\partial_{\mathbf{p}}\mathbf{F}$ is non-singular, where $\partial_{\mathbf{u}}\mathbf{G}$ denotes the Jacobian of $\mathbf{G}$ with respect to the space variable $\mathbf{u}$ and $\partial_{\mathbf{p}}\mathbf{F}$ the Jacobian of $\mathbf{F}$ with respect to $\mathbf{p}$. The semi-discretized Navier–Stokes equations possess this property since

$$\mathbf{F}(t,\mathbf{u}_h,\mathbf{p}_h) = (\mathbf{f}_h,\boldsymbol{\phi}_i) - (Re^{-1}\nabla\mathbf{u}_h,\nabla\boldsymbol{\phi}_i) + ((\mathbf{u}_h\cdot\nabla)\mathbf{u}_h,\boldsymbol{\phi}_i) + (p_h,\nabla\cdot\boldsymbol{\phi}_i), \tag{10}$$
$$\mathbf{G}(t,\mathbf{u}_h) = (\nabla\cdot\mathbf{u}_h,\psi_i), \tag{11}$$
$$(\partial_{\mathbf{p}}\mathbf{F}(t,\mathbf{u}_h,\mathbf{p}_h))_{ij} = (\psi_j,\nabla\cdot\boldsymbol{\phi}_i),$$
$$(\partial_{\mathbf{u}}\mathbf{G}(t,\mathbf{u}_h))_{ij} = (\nabla\cdot\boldsymbol{\phi}_j,\psi_i).$$

It follows, see (4),

$$\partial_{\mathbf{u}}\mathbf{G}M^{-1}\partial_{\mathbf{p}}\mathbf{F} = \left(-B_1^T, -B_2^T\right)\begin{pmatrix} M^{-1} & 0 \\ 0 & M^{-1}\end{pmatrix}\begin{pmatrix} -B_1 \\ -B_2\end{pmatrix}.$$

This matrix is non-singular since all factors possess full rank. It is known [2,9] that in this case the DAE (8) and (9) has the differentiation index 2.

Only an initial velocity $\mathbf{u}_0$ is given for the Navier–Stokes equations. However, the application of DIRK methods to (8) and (9) requires also the definition of an initial pressure $\mathbf{p}_0$. To this end, the algebraic constraint (9) is differentiated which leads to

$$0 = \mathbf{G}_t(t,\mathbf{u}) + \mathbf{G}_{\mathbf{u}}(t,\mathbf{u})\dot{\mathbf{u}}.$$

Inserting this into (8) yields

$$-\mathbf{G}_t(t,\mathbf{u}) = \mathbf{G}_{\mathbf{u}}(t,\mathbf{u})\dot{\mathbf{u}} = \mathbf{G}_{\mathbf{u}}(t,\mathbf{u})M^{-1}\mathbf{F}(t,\mathbf{u},\mathbf{p}). \tag{12}$$

This gives an equation for the pressure, in particular at the initial time.

To derive an RK method for the DAE (8) and (9), one considers instead of the algebraic constraint (9) the differential equation

$$\varepsilon I\dot{\mathbf{p}} = \mathbf{G}(t,\mathbf{u}), \quad \varepsilon > 0.$$

For the system of this equation together with (8), an RK method can be applied. By letting $\varepsilon \to 0$, the RK method for the DAE (8) and (9) is obtained [9]:

$$\begin{aligned} &M\mathbf{k}_i = \mathbf{F}(t_m + c_i\tau_m, \mathbf{U}_i, \mathbf{P}_i), \quad \mathbf{U}_i = \mathbf{u}_m + \tau_m\sum_{j=1}^{i} a_{ij}\mathbf{k}_j,\ i = 1,\dots,s,\\ &\mathbf{0} = \mathbf{G}(t_m + c_i\tau_m, \mathbf{U}_i), \quad \mathbf{P}_i = \mathbf{p}_m + \tau_m\sum_{j=1}^{i} a_{ij}\mathbf{l}_j,\ i = 1,\dots,s,\\ &\mathbf{u}_{m+1} = \mathbf{u}_m + \sum_{i=1}^{s} b_i\mathbf{k}_i, \quad \mathbf{p}_{m+1} = \mathbf{p}_m + \sum_{i=1}^{s} b_i\mathbf{l}_i. \end{aligned} \tag{13}$$

Note, that in this case the coefficient matrix A of the Butcher table is singular and the values $\mathbf{l}_j$ in (13) are not well-defined. To circumvent this difficulty, the second equation of (13) is multiplied by M and the first equation of (13) is inserted, leading to the system

$$M\mathbf{U}_i = M\mathbf{u}_m + \tau_m\sum_{j=1}^{i} a_{ij}\mathbf{F}(t_m + c_j\tau_m, \mathbf{U}_j, \mathbf{P}_j), \tag{14}$$

$$\mathbf{0} = \mathbf{G}(t_m + c_i\tau_m, \mathbf{U}_i). \tag{15}$$

Having solved (14) and (15) gives in particular $\mathbf{U}_s$ and $\mathbf{P}_s$. If an RK method is stiffly accurate, then the solution in time t_{m+1} is $(\mathbf{u}_{m+1}, \mathbf{p}_{m+1}) = (\mathbf{U}_s, \mathbf{P}_s)$.

4.3. Application to the Navier–Stokes equations

For the application of RK methods to the Navier–Stokes equations, we consider only stiffly accurate DIRK methods with $a_{11} = 0$. Then, the system (14) and (15) for $i = 1$ reduces to $\mathbf{U}_1 = \mathbf{u}_m$. The associated pressure field is $P_1 = \mathbf{p}_m$. Using (10) and (11) leads to

$$\begin{pmatrix} M + \tau_m a_{ii}A(\mathbf{U}_i) & 0 & \tau_m a_{ii}B_1 \\ 0 & M + \tau_m a_{ii}A(\mathbf{U}_i) & \tau_m a_{ii}B_2 \\ B_1^T & B_2^T & 0 \end{pmatrix}\begin{pmatrix} \mathbf{U}_i^{(1)} \\ \mathbf{U}_i^{(2)} \\ \mathbf{P}_i \end{pmatrix}$$

$$= \begin{pmatrix} M\mathbf{u}_m^{(1)} + \tau_m\sum_{j=1}^{i-1} a_{ij}\left(\mathbf{f}^{(1)}(t_m + c_j\tau_m) - A(\mathbf{U}_j)\mathbf{U}_j^{(1)} - B_1\mathbf{P}_j\right) \\ \qquad + \tau_m a_{ii}\mathbf{f}^{(1)}(t_m + c_i\tau_m) \\ M\mathbf{u}_m^{(2)} + \tau_m\sum_{j=1}^{i-1} a_{ij}\left(\mathbf{f}^{(2)}(t_m + c_j\tau_m) - A(\mathbf{U}_j)\mathbf{U}_j^{(2)} - B_2\mathbf{P}_j\right) \\ \qquad + \tau_m a_{ii}\mathbf{f}^{(2)}(t_m + c_i\tau_m) \\ \mathbf{0} \end{pmatrix},$$

$i = 2,\dots,s$. Eq. (12) for the initial pressure $\mathbf{p}_0$ has the form:

$$\Delta p(0,\mathbf{x}) = \nabla\cdot(\mathbf{f}(0,\mathbf{x}) + Re^{-1}\Delta\mathbf{u}_0(\mathbf{x}) - (\mathbf{u}_0(\mathbf{x})\cdot\nabla)\mathbf{u}_0(\mathbf{x})),$$

and it has to be closed with boundary conditions. In applications, an initial state is often unknown and one has to start the simulations with a developed flow field which was computed in a preprocessing step. In this case, the pressure of the developed flow field can be used as initial pressure.

4.4. Adaptive time step control

RK methods have the advantage that they allow an easy implementation of an adaptive time step length control. Consider an RK method of order $p \geqslant 2$. An adaptive time step control employs a second RK method which has the coefficients a_{ij}, $\hat{b}_i$ and c_i, $i,\ j = 1,\dots,s$, and order $p-1$. The solution of the second method at t_{m+1} is given by

$$\hat{\mathbf{u}}_{m+1} = \mathbf{u}_m + \sum_{i=1}^{s}\hat{b}_i\mathbf{k}_i.$$

Now, the next time step τ_{m+1} is proposed to be

$$\tau_{m+1} = \rho\frac{\tau_m^2}{\tau_{m-1}}\left(\frac{TOL\cdot r_m}{r_{m+1}^2}\right)^{1/p}, \tag{16}$$

where $\rho \in (0,1]$ is a safety factor, $TOL > 0$ is a given tolerance and

$$r_{m+1} := |J(\mathbf{u}_{m+1}) - J(\hat{\mathbf{u}}_{m+1})| \quad \text{or} \quad r_{m+1} := |J(\mathbf{u}_{m+1} - \hat{\mathbf{u}}_{m+1})|, \tag{17}$$

where $J(\cdot)$ is some functional. This step size selection rule is called PI-controller and it is due to [8]. For details on the numerical error and the implementation of automatic step length controls, we refer to [9,18].

4.5. Studied methods

The numerical studies presented in Section 6 will use DIRK methods with $s \in \{2,3,4\}$ internal stages. The method with $s = 2$ is the pressure-corrected Crank–Nicolson scheme (CN), which is recovered for $a_{21} = a_{22} = 1/2$, see (6) for its formulation applied to the Navier–Stokes equations. The Butcher table of the pressure-corrected fractional-step θ-scheme (FS) is given by [24],

0	0	0	0	0
θ	$\theta\beta$	$\theta\alpha$	0	0
$\theta+\tilde{\theta}$	$\theta\beta$	$(\theta+\tilde{\theta})\alpha$	$\tilde{\theta}\beta$	0
1	$\theta\beta$	$(\theta+\tilde{\theta})\alpha$	$(\theta+\tilde{\theta})\beta$	$\theta\alpha$
	$\theta\beta$	$(\theta+\tilde{\theta})\alpha$	$(\theta+\tilde{\theta})\beta$	$\theta\alpha$

with the coefficients

$$\theta = 1 - \frac{\sqrt{2}}{2}, \quad \tilde{\theta} = 1 - 2\theta, \quad \alpha = \frac{\tilde{\theta}}{1-\theta}, \quad \beta = 1 - \alpha.$$

An embedded method of first order is given by

$$\hat{b}_1 = 0.11785113033497070959,$$
$$\hat{b}_2 = 0.49509379160690495120,$$
$$\hat{b}_3 = 0.29636243203812433921,$$
$$\hat{b}_4 = 0.09069264621404818692.$$

Three DIRK methods of convergence order $p = 3$ from [23] will be studied. The first one, called DIRK3, has the non-zero coefficients

$$a_{21} = a_{22} = a_{33} = \frac{1}{2} + \frac{\sqrt{3}}{6}, \quad a_{32} = -\frac{1}{12a_{22}(2a_{22}-1)},$$
$$a_{31} = 1 - a_{33} - a_{32}$$

and an embedded method with $p = 2$ is given by

$$\hat{b}_1 = \frac{5}{12} + \frac{\sqrt{3}}{12}, \quad \hat{b}_2 = \frac{3}{4} + \frac{\sqrt{3}}{12}, \quad \hat{b}_3 = -\frac{1}{6} - \frac{\sqrt{3}}{6}.$$

The method DIRK3L possesses the non-zero coefficients

$$a_{21} = a_{22} = a_{33} = 1 - \frac{\sqrt{2}}{2}, \quad a_{32} = \frac{1}{4(1-a_{22})}, \quad a_{31} = 1 - a_{33} - a_{32}$$

and the coefficients for the embedded method with $p = 2$ are

$$\hat{b}_1 = \hat{b}_2 = \frac{1}{2} - \frac{1}{8}\sqrt{2}, \quad \hat{b}_3 = \frac{1}{4}\sqrt{2}.$$

The non-zero coefficients of the third method, DIRK34, are given by

$$a_{21} = a_{22} = a_{33} = a_{44} = 0.1558983899988677,$$
$$a_{31} = 1 - a_{32} - a_{22}, \quad a_{32} = 1.072486270734370,$$
$$a_{42} = 0.7685298292769537, \quad a_{43} = 0.09666483609791597.$$

This method is stiffly accurate, L-stable. An embedded method with convergence order $p = 2$ is given by $\hat{b}_i = a_{3i},\ i = 1, \ldots, 4$.

The other coefficients of the DIRK methods are defined by $c_i = \sum_{j=1}^{i} a_{ij},\ i = 1, \ldots, s$, and $(H4)$.

5. Rosenbrock–Wanner schemes

5.1. Application to ODEs

Consider, as in the case of DIRK schemes, an ODE of form (7). An s-stage Rosenbrock–Wanner method (ROW method) is given by

$$M\mathbf{k}_i = \mathbf{F}(t_m + \alpha_i \tau_m, \widetilde{\mathbf{U}}_i) + \tau_m W \sum_{j=1}^{i} \gamma_{ij}\mathbf{k}_j + \tau_m \gamma_i \dot{\mathbf{F}}(t_m, \mathbf{u}_m), \tag{18}$$

$$\widetilde{\mathbf{U}}_i = \mathbf{u}_m + \tau_m \sum_{j=1}^{i-1} a_{ij}\mathbf{k}_j, \quad i = 1, \ldots, s,$$

$$\mathbf{u}_{m+1} = \mathbf{u}_m + \tau_m \sum_{i=1}^{s} b_i \mathbf{k}_i, \tag{19}$$

where s is the number of internal stages, $\alpha_{ij},\ \gamma_{ij},\ b_i$ are the parameters of the method,

$$\alpha_i := \sum_{j=1}^{i-1} \alpha_{ij}, \quad \gamma_i := \sum_{j=1}^{i} \gamma_{ij}, \quad \gamma := \gamma_{ii} > 0, \quad i = 1, \ldots, s,$$

and $W := \partial_{\mathbf{u}} \mathbf{F}(t_m, \mathbf{u}_m)$.

A sufficient consistency order can be obtained if the parameters $\alpha_{ij},\ \gamma_{ij}$, and b_i are chosen appropriately. If W is only an approximation to $\partial_{\mathbf{u}}\mathbf{F}(t_m, \mathbf{u}_m)$ or if W is an arbitrary matrix, additional consistency conditions arise, see [9,22]. These methods are called W-methods [29]. It could be already observed [18], that approximated Jacobians can lead to an order reduction. We will not consider this option in the numerical studies. If a ROW method is applied to a semi-discretized partial differential equation, further order condition should be satisfied to avoid order reduction, see [20].

An efficient implementation of a ROW method introduces the new variables [18],

$$\mathbf{U}_i := \tau_m \sum_{j=1}^{i} \gamma_{ij}\mathbf{k}_j, \quad i = 1, \ldots, s. \tag{20}$$

The matrix $(\gamma_{ij})_{i,j=1}^{s}$ is a lower triangular matrix with the entries $\gamma > 0$ in the main diagonal. Consequently, (20) can be solved for $\mathbf{k}_i$, leading to

$$\mathbf{k}_i = \frac{1}{\tau_m} \sum_{j=1}^{i} c_{ij}\mathbf{U}_j, \quad (c_{ij})_{i,j=1}^{s} = \left((\gamma_{ij})_{i,j=1}^{s}\right)^{-1}, \quad i = 1, \ldots, s. \tag{21}$$

The matrix $(c_{ij})_{i,j=1}^{s}$ can be computed a priori and it is also a lower triangular matrix with the entries γ^{-1} in its main diagonal. Substituting (21) into (18) and (19) and rearranging terms give

$$(M - \gamma\tau_m W)\mathbf{U}_i = \gamma\tau_m \mathbf{F}(t_m + \alpha_i\tau_m, \widehat{\mathbf{U}}_i) - \gamma M \sum_{j=1}^{i-1} c_{ij}\mathbf{U}_j + \gamma\gamma_i\tau_m^2 \dot{\mathbf{F}}(t_m, \mathbf{u}_m), \tag{22}$$

$$\widehat{\mathbf{U}}_i = \mathbf{u}_m + \tau_m \sum_{j=1}^{i-1} \tilde{a}_{ij}\mathbf{U}_j, \quad \tilde{a}_{ij} = \sum_{l=1}^{i-1} a_{il}c_{lj}, \quad i = 1, \ldots, s,$$

$$\mathbf{u}_{m+1} = \mathbf{u}_m + \sum_{i=1}^{s} b_i \left(\sum_{j=1}^{i} c_{ij}\mathbf{U}_j\right). \tag{23}$$

The ROW method (22) and (23) requires the successive solution of s linear systems of equations with the same matrix $M - \gamma\tau_m W$. Note, W depends only on $\mathbf{u}_m$ but not on $\mathbf{u}_{m+1}$. The right hand side of the ith linear system of equations depends on the solutions of the first $(i-1)$ systems. Thus, a main difference of ROW methods to implicit θ-schemes and DIRK methods is that it is not necessary to solve a nonlinear system of equations in each discrete time but a fixed number of linear systems of equations with the same matrix $M - \gamma\tau_m W$.

5.2. Application to DAEs of index 2

Consider again the DAE (8) and (9) and let

$$\mathbf{U}_i := \tau_m \sum_{j=1}^{i} \gamma_{ij}\mathbf{k}_j, \quad \mathbf{P}_i := \tau_m \sum_{j=1}^{i} \gamma_{ij}\mathbf{l}_j, \quad i = 1, \ldots, s.$$

The ROW method (22) and (23) applied to (8) and (9) reads as follows:

$$\begin{pmatrix} M - \gamma\tau_m \partial_{\mathbf{U}}\mathbf{F}(t_m, \mathbf{u}_m, \mathbf{p}_m) & -\gamma\tau_m \partial_{\mathbf{P}}\mathbf{F}(t_m, \mathbf{u}_m, \mathbf{p}_m) \\ -\gamma\tau_m \partial_{\mathbf{U}}\mathbf{G}(t_m, \mathbf{u}_m) & 0 \end{pmatrix} \begin{pmatrix} \mathbf{U}_i \\ \mathbf{P}_i \end{pmatrix}$$
$$= \gamma\tau_m \begin{pmatrix} \mathbf{F}(t_m + \alpha_i\tau_m, \widehat{\mathbf{U}}_i, \widehat{\mathbf{P}}_i) \\ \mathbf{G}(t_m + \alpha_i\tau_m, \widehat{\mathbf{U}}_i) \end{pmatrix} - \gamma \begin{pmatrix} M \sum_{j=1}^{i-1} c_{ij}\mathbf{U}_j \\ \mathbf{0} \end{pmatrix} + \gamma\gamma_i\tau_m^2 \begin{pmatrix} \dot{\mathbf{F}}(t_m, \mathbf{u}_m, \mathbf{p}_m) \\ \dot{\mathbf{G}}(t_m, \mathbf{u}_m) \end{pmatrix},$$

$$\widehat{\mathbf{U}}_i = \mathbf{u}_m + \tau_m \sum_{j=1}^{i-1} \tilde{a}_{ij}\mathbf{U}_j, \quad \widehat{\mathbf{P}}_i = \mathbf{p}_m + \tau_m \sum_{j=1}^{i-1} \tilde{a}_{ij}\mathbf{P}_j, \quad i = 1, \ldots, s,$$

$$\mathbf{u}_{m+1} = \mathbf{u}_m + \sum_{i=1}^{s} b_i \left(\sum_{j=1}^{i} c_{ij}\mathbf{U}_j\right), \quad \mathbf{p}_{m+1} = \mathbf{p}_m + \sum_{i=1}^{s} b_i \left(\sum_{j=1}^{i} c_{ij}\mathbf{P}_j\right).$$

5.3. Application to the Navier–Stokes equations

The derivatives with respect to the phase space variable are given by [18, pp. 53/54],

$$\partial_{u_1} F_1 = \nu\Delta - \partial_x u_1 - u_1\partial_x - u_2\partial_y, \quad \partial_{u_2} F_1 = -\partial_y u_1,$$
$$\partial_p F_1 = -\partial_x, \quad \partial_{u_1} F_2 = -\partial_x u_2,$$
$$\partial_{u_2} F_2 = \nu\Delta - u_1\partial_x - u_2\partial_y - \partial_y u_2, \quad \partial_p F_2 = -\partial_y,$$
$$\partial_{u_1} G = -\partial_x, \quad \partial_{u_2} G = -\partial_y.$$

Defining the matrices

$$(J_{11})_{ij} := (A)_{ij} + \left((\partial_x \mathbf{u}_m^{(1)})\varphi_j, \varphi_i\right), \quad (J_{12})_{ij} := \left((\partial_y \mathbf{u}_m^{(1)})\varphi_j, \varphi_i\right),$$
$$(J_{21})_{ij} := \left((\partial_x \mathbf{u}_m^{(2)})\varphi_j, \varphi_i\right), \quad (J_{22})_{ij} := (A)_{ij} + \left((\partial_y \mathbf{u}_m^{(2)})\varphi_j, \varphi_i\right),$$

the algebraic form of a ROW method applied to the Navier–Stokes equations reads as

$$\begin{pmatrix} M+\gamma\tau_m J_{11} & \gamma\tau_m J_{12} & \gamma\tau_m B_1 \\ \gamma\tau_m J_{21} & M+\gamma\tau_m J_{22} & \gamma\tau_m B_2 \\ \gamma\tau_m B_1^T & \gamma\tau_m B_2^T & 0 \end{pmatrix} \begin{pmatrix} \mathbf{U}_i^{(1)} \\ \mathbf{U}_i^{(2)} \\ \mathbf{P}_i \end{pmatrix}$$
$$= \gamma\tau_m \begin{pmatrix} \mathbf{f}^{(1)}(t_m+\alpha_i\tau_m) - A(\widehat{\mathbf{U}}_i)\widehat{\mathbf{U}}_i^{(1)} - B_1\widehat{\mathbf{P}}_i \\ \mathbf{f}^{(2)}(t_m+\alpha_i\tau_m) - A(\widehat{\mathbf{U}}_i)\widehat{\mathbf{U}}_i^{(2)} - B_2\widehat{\mathbf{P}}_i \\ \mathbf{0} \end{pmatrix} - \gamma \begin{pmatrix} M\sum_{j=1}^{i-1} c_{ij}\mathbf{U}_j^{(1)} \\ M\sum_{j=1}^{i-1} c_{ij}\mathbf{U}_j^{(2)} \\ \mathbf{0} \end{pmatrix}$$
$$+ \gamma\gamma_i\tau_m^2 \begin{pmatrix} \dot{\mathbf{f}}^{(1)}(t_m) \\ \dot{\mathbf{f}}^{(2)}(t_m) \\ \mathbf{0} \end{pmatrix},$$

$$\widehat{\mathbf{U}}_i^{(k)} = \mathbf{u}_m^{(k)} + \tau_m \sum_{j=1}^{i-1} \tilde{a}_{ij}\mathbf{U}_j^{(k)}, \quad \widehat{\mathbf{P}}_i = \mathbf{p}_m + \tau_m \sum_{j=1}^{i-1} \tilde{a}_{ij}\mathbf{P}_j,$$

$$\mathbf{u}_{m+1}^{(k)} = \mathbf{u}_m^{(k)} + \sum_{i=1}^{s} b_i \left(\sum_{j=1}^{i} c_{ij}\mathbf{U}_j^{(k)} \right), \quad \mathbf{p}_{m+1} = \mathbf{p}_m + \sum_{i=1}^{s} b_i \left(\sum_{j=1}^{i} c_{ij}\mathbf{P}_j \right),$$

$i = 1, \ldots, s$, $k \in \{1, 2\}$. Note that $\mathbf{G}(t_m + \alpha_i\tau_m, \widehat{\mathbf{U}}_i) = \mathbf{0}$ because $B_1^T\widehat{\mathbf{U}}_i^{(1)} + B_2^T\widehat{\mathbf{U}}_i^{(2)} = \mathbf{0}$, $i = 1, \ldots, s$.

5.4. Adaptive time step control

As for DIRK methods, an automatic step length control can be implemented with the help of embedded methods which use the coefficients α_{ij} and γ_{ij}. Only the coefficients $\hat{b}_i$ are new and therefore the computed values $\mathbf{k}_i$ from (21) can be used to compute $\hat{\mathbf{u}}_{m+1}$.

5.5. Studied methods

A number of ROW methods were already included into the numerical studies of Ref. [17]. Since a clear ranking of these methods could not be obtained in [17], all of them will be considered in the numerical studies presented in Section 6. Moreover, some additional ROW methods will be studied, for instance RODASP, a fourth order, stiffly accurate method for linear PDEs from [28] and ROSI2P1, a third order, L-stable W-method [26]. The considered ROW methods are summarized in Table 1. For a detailed description of the methods which were studied already in [17], including all coefficients, we refer to [17]. The coefficients of ROSI2P1 and RODASP, including their embedded schemes, are given in Appendix A, Tables A.1 and A.2, respectively. The coefficients of the embedded schemes of the other ROW methods are given in Table A.3.

Table 1
ROW methods considered in the numerical studies.

Method	Stages	Order	Remarks
ROS3P	3	3	From [19], A-stable
ROWDAIND2	4	3	From [21], stiffly accurate, only designed for DAEs of index 2 and not for PDEs
ROS3Pw	3	3	From [25], A-stable
ROS34PW2	4	3	From [25], L-stable W-method, stiffly accurate
ROS34PW3	4	3	From [25], A-stable W-method
RODASP	6	4	From [28], L-stable, stiffly accurate
ROSI2P1	4	3	From [26], L-stable, W-method, for PDAEs of index 2

6. Numerical studies

The main goal of our studies consists in investigating the impact of an adaptive time step control on the accuracy of the results and the efficiency of the simulations. In order to keep the paper at a reasonable length, we decided to perform our studies only at one example and with a fixed discretization in space.

On the one hand, the considered example should be simple enough such that it can be implemented easily in many codes and it can serve as a benchmark problem. On the other hand, the flow described by this example should posses some features which occur in flows coming from applications. In our opinion, a good test problem fulfilling these requirements is the 2D flow around a circular cylinder defined in [27]. Numerical studies at this problem can be found, e.g. in [12,17,27].

The flow domain is presented in Fig. 1. The Navier–Stokes equations (1) have the right hand side $\mathbf{f} = \mathbf{0}$ and the final time is set to be $T = 8$. The inflow and outflow boundary conditions are given by

$$\mathbf{u}(t; 0, y) = \mathbf{u}(t; 2.2, y) = 0.41^{-2} \sin(\pi t/8)(6y(0.41 - y), 0) \text{ m/s},$$
$$0 \leqslant y \leqslant 0.41.$$

On all other boundaries, the no-slip condition $\mathbf{u} = \mathbf{0}$ m is prescribed. The Reynolds number of the flow, based on the mean inflow, the diameter of the cylinder and the prescribed viscosity $\nu = 10^{-3}$ m^2/s is $0 \leqslant Re(t) \leqslant 100$.

Important parameters of flows around bodies are the drag coefficient $c_d(t)$ and the lift coefficient $c_l(t)$ at the body. These coefficients can be computed by

$$c_d(t) = -20[(\mathbf{u}_t, \mathbf{v}_d) + (\nu\nabla\mathbf{u}, \nabla\mathbf{v}_d) + ((\mathbf{u}\cdot\nabla)\mathbf{u}, \mathbf{v}_d) - (p, \nabla\cdot\mathbf{v}_d)],$$
$$c_l(t) = -20[(\mathbf{u}_t, \mathbf{v}_l) + (\nu\nabla\mathbf{u}, \nabla\mathbf{v}_l) + ((\mathbf{u}\cdot\nabla)\mathbf{u}, \mathbf{v}_l) - (p, \nabla\cdot\mathbf{v}_l)]$$

for any function $\mathbf{v}_d \in (H^1(\Omega))^2$ with $(\mathbf{v}_d)|_S = (1, 0)^T$, S being the boundary of the body, and $\mathbf{v}_d$ vanishes on all other boundaries and for any test function $\mathbf{v}_l \in (H^1(\Omega))^2$ with $(\mathbf{v}_l)|_S = (0, 1)^T$ and $\mathbf{v}_l$ vanishes on all other boundaries, respectively. The numerical studies in [12,27] showed that the accurate computation of the lift coefficient is more difficult than of the drag coefficient. A third benchmark parameter defined in [27] is the difference of the pressure between the front and the back at the cylinder at the final time $\Delta p(8) := p(8; 0.15, 0.2) - p(8; 0.25, 0.2)$.

The discretization in space was performed with quadrilateral finite elements, see Fig. 2 for the initial grid, using the inf–sup stable Q_2/P_1^{disc} pair of finite element spaces, see Section 1 for the motivation of this choice. It is known [15] that for an accurate computation of the coefficients at the body, the circular boundary has to be approximated in an appropriate way, e.g. using isoparametric finite elements. The computations were performed on a mesh with

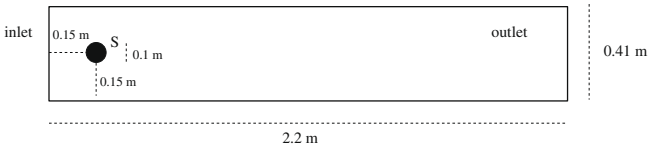

Fig. 1. Channel with a cylinder.

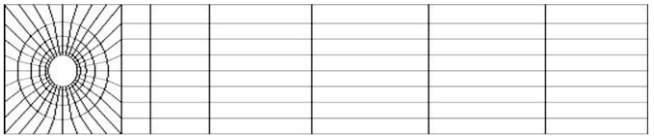

Fig. 2. Initial grid (level 0).

107,712 velocity degrees of freedom and 39,936 pressure degrees of freedom.

For assessing the accuracy of the results, reference values for the coefficients are necessary. The reference values for $c_{d,\max}$, $c_{l,\max}$ and $\Delta p(8)$ given in [12] are based on simulations with the Crank–Nicolson scheme and the fractional-step θ-scheme (both second order) with the fixed time step $\Delta t = 0.00125$. In these simulations, finite element discretizations with second order velocity and first order pressure with around half a million of degrees of freedom were used. Since our numerical studies will use sometimes a similar fineness of the discretizations, we performed a simulation with the third order ROW scheme ROS3Pw and the fixed time step $\Delta t = 0.00125$ (6400 time steps) on a grid with 2,347,776 degrees of freedom in space. The numerical studies below will show the good accuracy of ROS3Pw. A comparison of the new reference values with the reference values from [12] is presented in Table 2. It can be seen that the differences are rather small.

A core of Linux cluster with a 3 GHz system processor was used for the computations. The linear systems were solved directly with an LU-decomposition using the package UMFPACK [4,5]. The fixed point iteration for the θ-schemes and the DIRK schemes was stopped if the Euclidean norm of the residual vector was smaller than 10^{-8}. The simulations were performed with the code MooNMD [16].

The adaptive time step control is based on two solutions which are computed with schemes of different order. The difference of these solutions is measured with some functional, see (17). A standard approach consists in using some norm:

$$r_{m+1} := \|(\mathbf{u}_{m+1}, p_{m+1}) - (\hat{\mathbf{u}}_{m+1}, \hat{p}_{m+1})\|. \tag{24}$$

However, for flows around obstacles, also the difference of flow coefficients like

$$\begin{aligned} r_{m+1} &:= |c_d(\mathbf{u}_{m+1}, p_{m+1}) - c_d(\hat{\mathbf{u}}_{m+1}, \hat{p}_{m+1})| \quad \text{or} \\ r_{m+1} &:= |c_l(\mathbf{u}_{m+1}, p_{m+1}) - c_l(\hat{\mathbf{u}}_{m+1}, \hat{p}_{m+1})| \end{aligned} \tag{25}$$

would be feasible choices. Finally, it is also possible to define r_{m+1} as a linear combination of the proposals (24) and (25). To keep the paper at a reasonable length, we consider here only the definition (24) with $\|\cdot\|$ being the Euclidean vector norm. Other approaches like (25) will be explored in forthcoming studies. The safety factor in (16) was set to be $\rho = 0.9$. Additionally, the length of the time step was restricted to the interval $[5e-4, 0.1]$.

The application of an adaptive time step control pursues two goals: to increase the accuracy as well as the efficiency of the simulations. These two aspects have to be considered in the evaluation of the numerical results.

The accuracy is measured with respect to the computed reference values. For the drag and lift coefficient, the Euclidean distance to the reference value (in the coefficient-time-plane) is used. For instance, let $\left(t^{\text{ref}}_{d,\max}, c^{\text{ref}}_{d,\max}\right)$ be the reference drag, see Table 2, and $(t_{d,\max}, c_{d,\max})$ be the value of some simulation, the error with respect to the drag coefficient is defined by

$$\text{err}_d := \sqrt{\left(t^{\text{ref}}_{d,\max} - t_{d,\max}\right)^2 + \left(c^{\text{ref}}_{d,\max} - c_{d,\max}\right)^2}.$$

With respect to $\Delta p(8)$, the distance to the reference value is used.

The efficiency is measured in computing time because this is the most important criterion in applications. A second measure will be the number of performed time steps. In the evaluation, only simulations will be considered whose computing time was less than 500,000 s. It should be emphasized that the computing time depends on the implementation of the methods. The used code MooNMD is a flexible research code which is not tailored for

Table 2
Comparison of reference values from [12] and from the computation with ROS3Pw and 2,347,776 degrees of freedom (here).

Coefficient	Computation	Value	Time
$c^{\text{ref}}_{d,\max}$	[12]	2.950921575	3.93625
	Here	2.950918381	3.93625
$c^{\text{ref}}_{l,\max}$	[12]	0.47795	5.693125
	Here	0.47787543	5.692500
$\Delta p(8)^{\text{ref}}$	[12]	−0.1116	
	Here	−0.11161567	

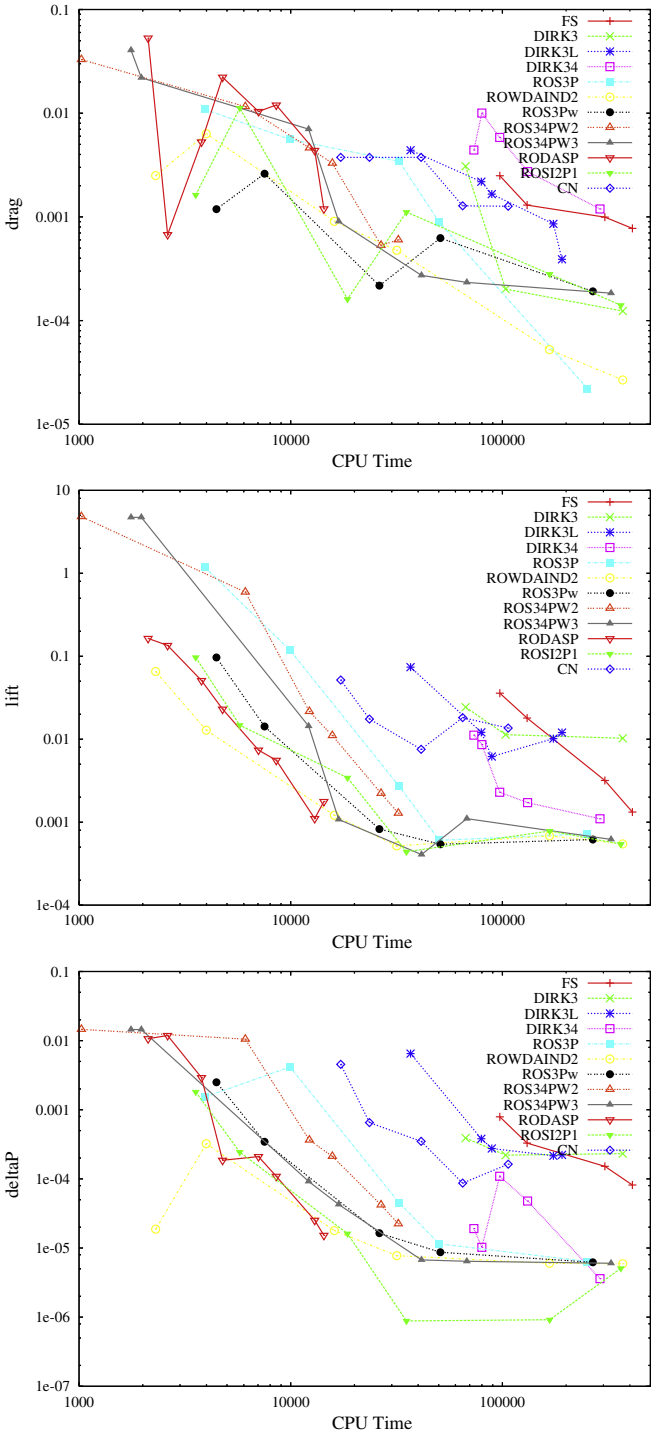

Fig. 3. Computing times versus differences to reference values.

solving a special class of problems or to use a special discretization. Since all routines of the solution process (assembling, solver) were the same for all considered methods, we think that the computing times give nevertheless a rather fair comparison.

The errors with respect to the reference values versus computing time are presented in Fig. 3. The lines start left with the largest tolerance *TOL* and they are drawn with respect to decreasing *TOL* (decreasing τ_m for CN). Simulations in the upper part of the pictures are inaccurate, simulations in the right hand part needed a long computing time. The best simulations with respect to accuracy and efficiency can be found towards the lower left corner. It can be already seen at these pictures that the θ-schemes and DIRK methods show a bad ratio of accuracy and efficiency. Several methods seem to have reached a minimal error plateau for the lift coefficient and $\Delta p(8)$. Probably, the error in the spatial discretization starts to dominate the temporal error.

We like to present also a quantitative evaluation of the simulations. For this reason, all results are grouped into classes, from very good to very bad. Each class obtains points from $\{2, 1, 0, -1, -2\}$. These points are multiplied with a factor which weights the importance of the considered parameter. The definitions of the classes and the weighting factors are given in Table 3.

It can be seen that we decided to weight accuracy somewhat higher than efficiency, with the ratio 3:2, because we think that the first goal should be to perform accurate simulations. It is known [27,12] that the computation of the lift coefficient is considerably more delicate than the computation of the drag coefficient. For this reason, the lift has a larger weight. The pressure difference seemed us to be the least important parameter. After having evaluated the simulations with respect to the accuracy, the worst simulations (accuracy points lower or equal than -5), were not considered further, because the obtained results are practically worthless. The best computing time and number of time steps were determined among the remaining simulations. Results which are close to the next better class got intermediate points. That means, if the class is defined in the interval $(r_0, r_1]$, an intermediate value is given if the result is in $(r_0, r_0 + 0.25(r_1 - r_0)]$. For the worst class, an intermediate value is given if the result is not worse than 1.25 times the limit for getting -1 point.

The evaluations with respect to the criteria from Table 3 are presented in Tables 4–9. For the individual parameters, only the best result are given in detail. Table 9 gives an overview on the quality of all results for all methods.

Table 3
Definition of classes and weighting factors for the quantitative evaluation of the results. The results of a simulation should be less or equal than the given numbers to obtain the given points.

Points	$c_{d,\max}$	$c_{l,\max}$	$\Delta p(8)$	Time	# Time steps
2	1e−4	1e−3	1e−5	2·best	2·best
1	1e−3	5e−3	1e−4	4·best	4·best
0	1e−2	1e−2	1e−3	8·best	8·best
−1	1e−1	5e−2	1e−2	16·best	16·best
−2	>1e−1	>5e−2	>1e−2	>16·best	>16·best
Weight	4	6	2	6	2

Table 4
Best methods with respect to drag coefficient.

Method	*TOL*	Time	$c_{d,\max}$	Error	Points
ROS3P	1.0e−5	3.936271	2.9509238408	2.21e−5	8
ROWDAIND2	5.0e−4	3.936224	2.9509238442	2.68e−5	8
ROWDAIND2	1.0e−3	3.936302	2.9509238423	5.25e−5	8
DIRK3	5.0e−6	3.936219	2.9507989116	1.24e−4	6
ROSI2P1	5.0e−6	3.936391	2.9509238516	1.41e−4	6
ROSI2P1	1.0e−4	3.936411	2.9509226624	1.61e−4	6
ROS34PW3	1.0e−6	3.936434	2.9509238499	1.84e−4	6
ROS3Pw	1.0e−5	3.936441	2.9509238485	1.91e−4	6
DIRK3	5.0e−5	3.936420	2.9508103135	2.01e−4	6
ROS3Pw	1.0e−4	3.936033	2.9509228400	2.17e−4	6
ROS34PW3	5.0e−6	3.936017	2.9509238130	2.33e−4	6
ROS34PW3	1.0e−5	3.935977	2.9509238140	2.73e−4	6
ROSI2P1	1.0e−5	3.936531	2.9509238396	2.81e−4	6

Table 5
Best methods with respect to lift coefficient.

Method	*TOL*	Time	$c_{l,\max}$	Error	Points
ROS34PW3	1.0e−5	5.692574	0.4783535647	4.07e−4	12
ROSI2P1	5.0e−5	5.692586	0.4783225874	4.40e−4	12
ROWDAIND2	5.0e−3	5.692226	0.4783157020	5.17e−4	12
ROS3Pw	5.0e−5	5.692201	0.4783032054	5.41e−4	12
ROSI2P1	5.0e−6	5.692876	0.4783606393	5.44e−4	12
ROWDAIND2	5.0e−4	5.692880	0.4783606945	5.47e−4	12
ROS3P	5.0e−5	5.692470	0.4781503872	6.05e−4	12
ROS3Pw	1.0e−5	5.692977	0.4783611948	6.18e−4	12
ROS34PW3	1.0e−6	5.692979	0.4783613903	6.20e−4	12
ROWDAIND2	1.0e−3	5.693059	0.4783610276	6.84e−4	12
ROS3P	1.0e−5	5.693104	0.4783588544	7.22e−4	12
ROSI2P1	1.0e−5	5.693176	0.4783593009	7.83e−4	12
ROS3Pw	1.0e−4	5.693109	0.4781962762	8.26e−4	12

Table 6
Best methods with respect to pressure difference.

Method	*TOL*	$\Delta p(8)$	Error	Points
ROSI2P1	5.0e−5	−0.11161655	8.81e−7	4
ROSI2P1	1.0e−5	−0.11161659	9.18e−7	4
DIRK34	1.0e−6	−0.11161206	3.61e−6	4
ROSI2P1	5.0e−6	−0.11162077	5.10e−6	4
ROWDAIND2	5.0e−4	−0.11162159	5.92e−6	4
ROWDAIND2	1.0e−3	−0.11162167	6.00e−6	4
ROS34PW3	1.0e−6	−0.11162169	6.02e−6	4
ROS3Pw	1.0e−5	−0.11162188	6.21e−6	4
ROS3P	1.0e−5	−0.11162203	6.36e−6	4
ROS34PW3	5.0e−6	−0.11162209	6.42e−6	4
ROS34PW3	1.0e−5	−0.11162239	6.72e−6	4
ROWDAIND2	5.0e−3	−0.11162342	7.75e−6	4
ROS3Pw	5.0e−5	−0.11162436	8.69e−6	4

Table 7
Best methods with respect to computing time (in s).

Method	*TOL*	CPU	Points
ROWDAIND2	5.0e−2	3990	12
ROSI2P1	5.0e−4	5742	12
RODASP	1.0e−5	7042	12
ROS3Pw	5.0e−4	7519	12
RODASP	5.0e−6	8546	9
ROS34PW3	1.0e−4	12,149	6
RODASP	1.0e−6	12,980	6
RODASP	5.0e−7	14,336	6
ROS34PW2	5.0e−6	15,731	6

Table 8
Best methods with respect to needed time steps.

Method	*TOL*	nts	Points
ROWDAIND2	5.0e−2	219	4
ROSI2P1	5.0e−4	300	4
DIRK34	1.0e−4	334	4
RODASP	1.0e−5	334	4
RODASP	5.0e−6	400	4
DIRK34	5.0e−5	406	4
ROS3Pw	5.0e−4	432	4
FS	1.0e−4	526	3

Table 9
Final evaluation of the simulations; for CN stands in the column *TOL* the length of the equidistant time step.

No.	Method	*TOL*	Drag	Lift	$\Delta p(8)$	CPU	nts	Total
1	RODASP	5.0e−7	2	9	3	6	2	22
2	ROS3Pw	1.0e−4	6	12	3	0	0	21
3	ROWDAIND2	1.0e−2	4	9	3	3	1	20
4	ROS34PW3	5.0e−5	4	9	2	3	2	20
5	RODASP	1.0e−6	0	9	3	6	2	20
6	ROWDAIND2	5.0e−3	4	12	4	0	−1	19
7	ROSI2P1	1.0e−4	6	6	3	3	1	19
8	ROS3Pw	5.0e−4	2	−3	0	12	4	15
9	RODASP	1.0e−5	−2	0	1	12	4	15
10	RODASP	5.0e−6	−2	3	1	9	4	15
11	ROWDAIND2	5.0e−2	0	−3	1	12	4	14
12	ROS34PW3	1.0e−5	6	12	4	−6	−2	14
13	ROSI2P1	5.0e−5	2	12	4	−3	−1	14
14	ROS3P	5.0e−5	4	12	3	−6	0	13
15	ROS3Pw	5.0e−5	4	12	4	−6	−2	12
16	ROS34PW2	1.0e−6	4	6	2	0	0	12
17	ROS34PW2	5.0e−7	4	9	3	−3	−1	12
18	ROSI2P1	5.0e−4	−2	−3	1	12	4	12
19	ROS3P	1.0e−5	8	12	4	−12	−4	8
20	ROWDAIND2	1.0e−3	8	12	4	−12	−4	8
21	ROWDAIND2	5.0e−4	8	12	4	−12	−4	8
22	ROS3P	1.0e−4	0	6	2	−3	2	7
23	ROS34PW3	1.0e−4	0	−3	2	6	2	7
24	ROS34PW3	5.0e−6	6	9	4	−9	−3	7
25	ROS3Pw	1.0e−5	6	12	4	−12	−4	6
26	ROS34PW3	1.0e−6	6	12	4	−12	−4	6
27	ROSI2P1	1.0e−5	6	12	4	−12	−4	6
28	ROSI2P1	5.0e−6	6	12	4	−12	−4	6
29	ROS34PW2	5.0e−6	0	−3	1	6	1	5
30	DIRK34	5.0e−6	2	9	2	−12	1	2
31	FS	5.0e−6	4	9	2	−12	−2	1
32	DIRK34	1.0e−6	2	9	4	−12	−2	1
33	FS	1.0e−5	4	6	1	−12	0	−1
34	CN	1.0e−2	0	−3	0	0	2	−1
35	DIRK34	1.0e−5	0	6	1	−12	2	−3
36	DIRK34	1.0e−4	0	−3	3	−9	4	−5
37	CN	5.0e−3	0	0	0	−6	0	−6
38	DIRK34	5.0e−5	−2	0	3	−12	4	−7
39	DIRK3L	5.0e−6	2	3	1	−12	−2	−8
40	DIRK3	5.0e−5	6	−3	1	−12	−2	−10
41	DIRK3L	1.0e−5	2	−3	0	−9	0	−10
42	CN	2.5e−3	2	−3	2	−9	−2	−10
43	FS	5.0e−5	2	−3	0	−12	2	−11
44	DIRK3	5.0e−6	6	−3	1	−12	−4	−12
45	FS	1.0e−4	2	−6	0	−12	3	−13
46	DIRK3	1.0e−4	2	−6	0	−9	0	−13
47	DIRK3L	1.0e−6	4	−3	1	−12	−4	−14
48	DIRK3L	5.0e−7	4	−3	1	−12	−4	−14
49	CN	1.3e−3	2	−3	1	−12	−4	−16
50	ROS34PW2	1.0e−5	0	−6	0	−15	−15	−36
51	ROWDAIND2	1.0e−1	2	−12	3	−15	−15	−37
52	RODASP	5.0e−5	−2	−6	1	−15	−15	−37
53	RODASP	1.0e−4	0	−9	−1	−15	−15	−40
54	ROS3Pw	1.0e−3	2	−12	−1	−15	−15	−41
55	RODASP	5.0e−4	4	−12	−3	−15	−15	−41
56	ROSI2P1	1.0e−3	2	−12	−1	−15	−15	−41
57	CN	2.0e−2	0	−9	−2	−15	−15	−41
58	DIRK3L	5.0e−5	0	−12	−2	−15	−15	−44
59	ROS3P	5.0e−4	0	−12	−2	−15	−15	−44
60	ROS3P	1.0e−3	−2	−12	−1	−15	−15	−45
61	ROS34PW2	5.0e−5	−2	−12	−3	−15	−15	−47
62	ROS34PW3	5.0e−4	−2	−12	−4	−15	−15	−48
63	RODASP	1.0e−3	−4	−12	−3	−15	−15	−49
64	ROS34PW2	1.0e−4	−4	−12	−4	−15	−15	−50
65	ROS34PW3	1.0e−3	−4	−12	−4	−15	−15	−50

The final results can be grouped into five classes: best (1–7), good (8–18), medium (19–29), poor (30–49) and worthless (50–65). Only ROW methods can be found in the first three classes. The second class contains mainly simulations which are either very fast, but rather inaccurate (8–11, 18), or very accurate but rather slow (12–15, 17). The θ-schemes and DIRK methods build the fourth class.

We explored also an alternative way to define the reference values. The idea consists in considering the same spatial grid as in all other simulations such that the error in space is the same. In order to obtain a very small error in time, a high order method (RODASP) with a very small time step ($\tau_m = 1.25\text{e}-3$) was used:

$$c_{d,\max} : (2.950923849, 3.93625), \quad c_{l,\max} : (0.47834818, 5.6925),$$
$$\Delta p(8) = -0.11162153.$$

With these reference values, exactly the same classes as in Table 9 were obtained. Thus, our evaluation of the results is the same for either choice of the reference values. Only the order of the methods within the classes changed somewhat.

The behavior of the ROW schemes with respect to the tolerance *TOL* is summarized in Table 10. It can be observed that there is no universal parameter *TOL* which yields good results for all ROW methods. A good choice of *TOL* depends on the used method. A rather wide range of good parameters has ROWDAIND2, ROS3Pw, RODASP and ROSI2P1. The good results of RODASP are mainly due to the efficiency of the method, in particular for $TOL \in \{1\text{e}-5, 1\text{e}-6\}$. The computed results with this method do

Table 10
Overview of the results obtained with the ROW methods with respect to the choice of *TOL*.

Method	To inaccurate	Good	Too slow
ROS3P	$\geqslant 1\text{e}-4$	$\{5\text{e}-5\}$	$\leqslant 1\text{e}-5$
ROWDAIND2		$\{5\text{e}-2, 1\text{e}-2, 5\text{e}-3\}$	$\leqslant 1\text{e}-3$
ROS3Pw	$\{1\text{e}-3\}$	$\{5\text{e}-4, 1\text{e}-4, 5\text{e}-5\}$	$\leqslant 1\text{e}-5$
ROS34PW2	$\geqslant 5\text{e}-6$	$\{1\text{e}-5, 5\text{e}-7\}$	
ROS34PW3	$\geqslant 1\text{e}-4$	$\{5\text{e}-5, 1\text{e}-5\}$	$\leqslant 5\text{e}-6$
RODASP	$\geqslant 5\text{e}-4$	$\{1\text{e}-5, 5\text{e}-6, 1\text{e}-6, 5\text{e}-7\}$	
ROSI2P1	$\{1\text{e}-3\}$	$\{5\text{e}-4, 1\text{e}-4, 5\text{e}-5\}$	$\leqslant 1\text{e}-5$

Table 11
Overview of the accuracy obtained with the θ-schemes and the DIRK methods with respect to the choice of *TOL*; for CN: with respect to the length of the equidistant time step.

Method	Very accurate	Medium accurate	Inaccurate
FS	$\leqslant 1\text{e}-5$		$\geqslant 5\text{e}-5$
CN			$\geqslant 1.25\text{e}-3$
DIRK34	$\leqslant 5\text{e}-6$	1e−5	$\geqslant 5\text{e}-5$
DIRK3		$\leqslant 5\text{e}-5$	1e−4
DIRK3L		5e−6	$\geqslant 1\text{e}-5$

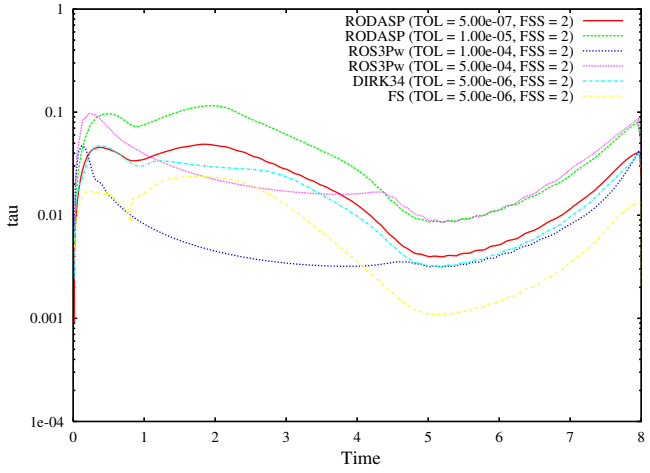

Fig. 4. Evolution of the length of the time steps, schemes no. 1, 2, 8, 9, 30, 31 from Table 9.

not belong to the most accurate ones. This method can be recommended if fast simulations are necessary and an average accuracy is sufficient. If a higher accuracy is required, ROS3Pw and ROWDAIND2 (with appropriate parameter *TOL*) seem to be good choices. The methods ROS34PW2 and ROS3P have no result in the best group.

The behavior of the θ-schemes and the DIRK schemes with respect to the accuracy is summarized in Table 11. It can be observed that there are methods (FS, DIRK34) and choices of *TOL* which lead to very accurate results. Their rather bad rating is due to their long computing times. Also the inaccuracy of CN compared to the other methods becomes obvious. The averaged number of iterations in the fixed point iteration was 1.6 or below (sometimes very close to 1) for small *TOL* and between 2 and 2.5 for larger *TOL*. These numbers have to be multiplied by the number of stages for the DIRK methods and by 3 for FS to obtain the averaged number of iterations per time step. There are several options for improving the computing times, e.g. using a weaker stopping criterion in the fixed point iteration (which might have negative consequences for the accuracy) or to construct better initial guesses for the fixed point iteration. The study of these options is beyond the scope of this paper. At any rate, for small *TOL*, which led to the most accurate results, not more than the factor 1.6 could be saved.

Fig. 4 provides information on the evolution of the length of the time steps for two methods of the best class (nos. 1, 2), two fast but comparatively inaccurate methods of the second class (nos. 8, 9) and two accurate methods of the fourth class (nos. 30, 31). The most methods used rather large time steps in the first half of the time interval. The vortex shedding starts at around 3.5 (non-dimensionalized) s [12], which resulted in a rather strong decrease of the time steps. Only ROS3Pw (*TOL* 1e−4) used already a small time step in the first half, which was also sufficient to simulate the vortex shedding. By far the smallest time steps for computing the vortex shedding were needed by FS.

7. Summary and outlook

A number of different time stepping schemes for the instationary Navier–Stokes equations were assessed at a laminar 2D flow around a cylinder. The emphasis of the numerical studies was on the effects of using an adaptive time step control with respect to the accuracy of the results and to the efficiency of the simulations.

A main observation is that several ROW schemes clearly outperformed standard approaches (θ-schemes, DIRK schemes), in particular the popular Crank–Nicolson scheme with an equidistant time step. The results depend much on the tolerance *TOL* for the adaptive choice of the length of the time step. Different methods possessed in general different appropriate values of *TOL*. For good parameters *TOL*, RODASP showed a high efficiency combined with an average accuracy. ROS3Pw and ROWDAIND2 produced very accurate results with medium efficiency.

The investigations of adaptive time stepping schemes have to be continued in several directions. Instead of using the Euclidean vector norm of the difference of two solutions as criterion, differences in functionals of interest have to be considered. Also, the study of 3D flows, in particular turbulent flows, is of great interest. A main difference concerning the efficiency will be that it is not longer advisable to use a sparse direct solver for the linear systems in 3D. Such solvers become inefficient because of the considerably larger fill-in. Instead, iterative solvers have to be use, like multigrid methods [13–15]. That means, the ROW methods lose the advantage that they need only one LU-decomposition of the system matrix and the second to s-th system solve are much cheaper than the first one.

Appendix A. Coefficients of some ROW methods and all embedded ROW methods

The appendix presents the coefficients of the schemes ROSI2P1, RODASP and of all embedded ROW methods. The coefficients of the other ROW methods can be found in [17].

Table A.1
Coefficients of ROSI2P1 and its embedded method.

$\gamma = 4.3586652150845900e-1$	
$\alpha_{21} = 5.0000000000000000e-1$	$\gamma_{21} = -5.0000000000000000e-1$
$\alpha_{31} = 5.5729261836499822e-1$	$\gamma_{31} = -6.4492162993321323e-1$
$\alpha_{32} = 1.9270738163500176e-1$	$\gamma_{32} = 6.3491801247597734e-2$
$\alpha_{41} = -3.0084516445435860e-1$	$\gamma_{41} = 9.3606009252719842e-3$
$\alpha_{42} = 1.8995581939026787e+0$	$\gamma_{42} = -2.5462058718013519e-1$
$\alpha_{43} = -5.9871302944832006e-1$	$\gamma_{43} = -3.2645441930944352e-1$
$b_1 = 5.2900072579103834e-2$	$\hat{b}_1 = 1.4974465479289098e-1$
$b_2 = 1.3492662311920438e+0$	$\hat{b}_2 = 7.0051069041421810e-1$
$b_3 = -9.1013275270050265e-1$	$\hat{b}_3 = 0.0000000000000000e+0$
$b_4 = 5.0796644892935516e-1$	$\hat{b}_4 = 1.4974465479289098e-1$

Table A.2
Coefficients of RODASP and its embedded method.

$\gamma = 2.5000000000e-1$	
$\alpha_{21} = 7.5000000000e-1$	$\gamma_{21} = -7.5000000000e-1$
$\alpha_{31} = 8.6120400814e-2$	$\gamma_{31} = -1.3551200000e-1$
$\alpha_{32} = 1.2387959919e-1$	$\gamma_{32} = -1.3799200000e-1$
$\alpha_{41} = 7.7403453551e-1$	$\gamma_{41} = -1.2560800000e+0$
$\alpha_{42} = 1.4926515495e-1$	$\gamma_{42} = -2.5014500000e-1$
$\alpha_{43} = -2.9419969046e-1$	$\gamma_{43} = 1.2209300000e+0$
$\alpha_{51} = 5.3087466826e+0$	$\gamma_{51} = -7.0731800000e+0$
$\alpha_{52} = 1.3308921400e+0$	$\gamma_{52} = -1.8056500000e+0$
$\alpha_{53} = -5.3741378117e+0$	$\gamma_{53} = 7.7438300000e+0$
$\alpha_{54} = -2.6550101103e-1$	$\gamma_{54} = 8.8500300000e-1$
$\alpha_{61} = -1.7644376488e+0$	$\gamma_{61} = 1.6840700000e+0$
$\alpha_{62} = -4.7475655721e-1$	$\gamma_{62} = 4.1826600000e-1$
$\alpha_{63} = 2.3696918469e+0$	$\gamma_{63} = -1.8814100000e+0$
$\alpha_{64} = 6.1950235906e-1$	$\gamma_{64} = -1.1378600000e-1$
$\alpha_{65} = 2.5000000000e-1$	$\gamma_{65} = -3.5714300000e-1$
$b_1 = -8.0368370789e-2$	$\hat{b}_1 = -1.7644376488e+0$
$b_2 = -5.6490613592e-2$	$\hat{b}_2 = -4.7475655721e-1$
$b_3 = 4.8828563004e-1$	$\hat{b}_3 = 2.3696918469e+0$
$b_4 = 5.0571621148e-1$	$\hat{b}_4 = 6.1950235906e-1$
$b_5 = -1.0714285714e-1$	$\hat{b}_5 = 2.5000000000e-1$
$b_6 = 2.5000000000e-1$	$\hat{b}_6 = 0.0000000000e+0$

Table A.3
Coefficients of the embedded methods for the other ROW methods.

ROS3P	$\hat{b}_1 = 3.3333333333e-1$, $\hat{b}_2 = 3.3333333333e-1$, $\hat{b}_3 = 3.3333333333e-1$
ROWDAIND2	$\hat{b}_1 = 4.7990028004e-1$, $\hat{b}_2 = 5.1762038112e-1$, $\hat{b}_3 = 2.4793388430e-3$, $\hat{b}_4 = 0.0000000000e+0$
ROS3PW	$\hat{b}_1 = -1.7863279495e-1$, $\hat{b}_2 = 3.3333333333e-1$, $\hat{b}_3 = 8.4529946162e-1$
ROS34PW2	$\hat{b}_1 = 3.7810903145e-1$, $\hat{b}_2 = -9.6042292212e-2$, $\hat{b}_3 = 5.0000000000e-1$, $\hat{b}_4 = 2.1793326075e-1$
ROS34PW3	$\hat{b}_1 = 3.1300297285e-1$, $\hat{b}_2 = -2.8946895245e-1$, $\hat{b}_3 = 9.7646597959e-1$, $\hat{b}_4 = 0.0000000000e+0$

References

[1] S. Berrone, M. Marro, Space–time adaptive simulations for unsteady Navier–Stokes problems, Comput. Fluid 38 (2009) 1132–1144.

[2] K.E. Brenan, S.L. Campbell, L.R. Petzold, Numerical solution of initial-value problems in differential-algebraic equations, Classics in Applied Mathematics, vol. 14, SIAM, Philadelphia, 1996.

[3] J.W. Butcher, On Runge–Kutta processes of high order, J. Aust. Math. Soc. 4 (1964) 179–194.

[4] T.A. Davis, Algorithm 832: Umfpack, an unsymmetric-pattern multifrontal method, ACM Trans. Math. Softw. 30 (2) (2004) 166–199.

[5] T.A. Davis, A column pre-ordering strategy for the unsymmetric-pattern multifrontal method, ACM Trans. Math. Softw. 30 (2) (2004) 165–195.

[6] V. Girault, P.-A. Raviart, Finite Element Methods for Navier–Stokes equations, Springer-Verlag, Berlin–Heidelberg–New York, 1986.

[7] P.M. Gresho, R.L. Sani, Incompressible Flow and the Finite Element Method, Wiley, Chichester, 2000.

[8] K. Gustafsson, M. Lundh, G. Söderlind, A PI stepsize control for the numerical solution of ordinary differential equations, BIT 28 (2) (1988) 270–287.

[9] E. Hairer, G. Wanner, Solving ordinary differential equations. II: Stiff and differential-algebraic problems, Springer Series in Computational Mathematics, second ed., vol. 14, Springer-Verlag, Berlin, 1996.

[10] V. John, Higher order finite element methods and multigrid solvers in a benchmark problem for the 3D Navier–Stokes equations, Int. J. Numer. Method Fluid 40 (2002) 775–798.

[11] V. John, Large Eddy simulation of turbulent incompressible flows. Analytical and numerical results for class of LES models, Lecture Notes in Computational Science and Engineering, vol. 34, Springer-Verlag, Berlin, Heidelberg, New York, 2004.

[12] V. John, Reference values for drag and lift of a two-dimensional time dependent flow around a cylinder, Int. J. Numer. Method Fluid 44 (2004) 777–788.

[13] V. John, On the efficiency of linearization schemes and coupled multigrid methods in the simulation of a 3D flow around a cylinder, Int. J. Numer. Method Fluid 50 (2006) 845–862.

[14] V. John, P. Knobloch, G. Matthies, L. Tobiska, Non-nested multi-level solvers for finite element discretizations of mixed problems, Computing 68 (2002) 313–341.

[15] V. John, G. Matthies, Higher order finite element discretizations in a benchmark problem for incompressible flows, Int. J. Numer. Method Fluid 37 (2001) 885–903.

[16] V. John, G. Matthies, MooNMD – a program package based on mapped finite element methods, Comput. Visual. Sci. 6 (2004) 163–170.

[17] V. John, G. Matthies, J. Rang, A comparison of time-discretization/linearization approaches for the time-dependent incompressible Navier–Stokes equations, Comput. Method Appl. Mech. Engrg. 195 (2006) 5995–6010.

[18] J. Lang, Adaptive multilevel solution of nonlinear parabolic PDE systems, Lecture Notes in Computational Science and Engineering, vol. 16, Springer-Verlag, Berlin, 2001.

[19] J. Lang, J. Verwer, ROS3P – an accurate third-order Rosenbrock solver designed for parabolic problems, BIT 41 (2001) 730–737.

[20] C. Lubich, A. Ostermann, Linearly implicit time discretization of non-linear parabolic equations, IMA J. Numer. Anal. 15 (4) (1995) 555–583.

[21] C. Lubich, M. Roche, Rosenbrock methods for differential-algebraic systems with solution-dependent singular matrix multiplying the derivative, Computing 43 (1990) 325–342.

[22] J. Rang, Stability estimates and numerical methods for degenerate parabolic differential equations, Ph.D. Thesis, Institut für Mathematik, TU Clausthal, 2004 (appeared also as book from Papierflieger Verlag Clausthal-Zellerfeld).

[23] J. Rang, Design of DIRK schemes for solving the Navier–Stokes-equations, Informatik-Bericht 2007-02, TU Braunschweig, Braunschweig, 2007.

[24] J. Rang, Pressure corrected implicit θ-schemes for the incompressible Navier–Stokes equations, Appl. Math. Comput. 201 (1–2) (2008) 747–761.

[25] J. Rang, L. Angermann, New Rosenbrock W-methods of order 3 for PDAEs of index 1, BIT 45 (2006) 761–787.

[26] J. Rang, L. Angermann, New Rosenbrock methods of order 3 for PDAEs of index 2, Adv. Differ. Eq. Control. Process. 1 (2) (2008) 193–217.

[27] M. Schäfer, S. Turek, The benchmark problem "Flow around a cylinder", in: E.H. Hirschel (Ed.), Flow Simulation with High-Performance Computers II, Notes on Numerical Fluid Mechanics, vol. 52, Vieweg, 1996, pp. 547–566.

[28] G. Steinebach, Order-reduction of ROW-methods for DAEs and method of lines applications, Preprint 1741, Technische Universität Darmstadt, Darmstadt, 1995.

[29] K. Strehmel, R. Weiner, Linear-implizite Runge–Kutta-Methoden und ihre Anwendung, Teubner-Texte zur Mathematik, vol. 127, Teubner, Stuttgart, 1992.

[30] D. Teleaga, J. Lang, Higher-order linearly implicit one-step methods for three-dimensional incompressible Navier–Stokes equations, Studia Babes-Bolyai Matematica 53 (2008) 109–121.

[31] S. Turek, Efficient solvers for incompressible flow problems: an algorithmic and computational Approach, Lecture Notes in Computational Science and Engineering, vol. 6, Springer, 1999.