ISSN 2070-0482, Mathematical Models and Computer Simulations, 2014, Vol. 6, No. 4, pp. 415–424. © Pleiades Publishing, Ltd., 2014.
Original Russian Text © L.M. Skvortsov, O.S. Kozlov, 2014, published in Matematicheskoe Modelirovanie, 2014, Vol. 26, No. 1, pp. 96–108.

Efficient Implementation of Diagonally Implicit Runge–Kutta Methods

L. M. Skvortsov and O. S. Kozlov

Bauman Moscow State Technical University, Moscow, Russia

e-mail: lm_skvo@rambler.ru, os.kozlov@gmail.com

Received September 12, 2012

Abstract—Efficient schemes for the implementation of diagonally implicit Runge–Kutta methods are considered. Methods of the 3rd and 4th orders are implemented. They are compared with known implicit solvers as applied to the solution of stiff and differential-algebraic equations.

Keywords: diagonally implicit Runge–Kutta methods, stiff problems, differential-algebraic problems, efficient implementation

DOI: 10.1134/S2070048214040085

1. INTRODUCTION

We consider diagonally implicit Runge–Kutta methods as applied to the Cauchy problem for the system of differential-algebraic equations (DAEs) defined in a semi-implicit form

$$y' = f(t,y,z), \quad y(t_0) = y_0, \tag{1a}$$

$$0 = g(t,y,z), \quad z(t_0) = z_0, \tag{1b}$$

where y is the vector of differential variables, z is the vector of algebraic variables, and t is an independent variable. In a particular case it may be an ODE system. We assume that the initial conditions y_0, z_0 are consistent [1] and the dimension of the vector function g coincides with the dimension of vector z.

For the numerical solution of stiff systems implicit methods are commonly used that provide a sustainable solution at a sufficiently small computational cost. Implicit methods are also used in solving DAEs of higher indices (above the 1st), which cannot be solved by explicit methods, as in this case an algebraic subsystem (1b) taken separately is not solvable with respect to z (for the definition of the DAE differentiation index see [1]).

Among the implicit Runge–Kutta methods, diagonally implicit (DIRK) are the most easily implementable. There are two types of such methods in common practice, i.e., singly diagonally implicit (SDIRK) [1, 2] and similar methods with an explicit first stage (ESDIRK) [3–6]. ESDIRK methods can have a second stage order, which is their advantage over SDIRK methods, which only possess a first stage order. Another advantage is the fact that with the same number of implicit stages, ESDIRK methods contain more parameters. Due to this, in [4–6] stiffly accurate ESDIRK methods of the fourth and fifth orders were constructed, in which the number of implicit stages coincided with the order. There are no similar SDIRK methods [2].

We consider ESDIRK methods with Butcher's tableau

$$\begin{array}{c|cccccc} 0 & 0 & & & & & \\ c_2 & a_{21} & \gamma & & & & \\ \vdots & \vdots & \vdots & \ddots & & & \\ c_{s-1} & a_{s-1,1} & a_{s-1,2} & \dots & \gamma & & \\ 1 & a_{s1} & a_{s2} & \cdots & a_{s,s-1} & \gamma & \\ \hline & a_{s1} & a_{s2} & \cdots & a_{s,s-1} & \gamma \end{array}, \tag{2}$$

where s is the number of stages. The integration step coincides with the last stage; i.e., the methods are stiffly accurate. We designate by y_n, z_n the solution of problem (1) obtained at the next point $t = t_n$. Formulas of the next step of the method (2) can be written as

$$F_1 = f(t_n, y_n, z_n), \tag{3a}$$

$$\left.\begin{aligned} Y_i = y_n + h\sum_{j=1}^{i-1} a_{ij}F_j + h\gamma F_i, \quad F_i = f(t_0 + c_i h, Y_i, Z_i), \\ 0 = g(t_n + c_i h, Y_i, Z_i), \end{aligned}\right\} \quad i = 2,3,\ldots,s, \tag{3b}$$

$$y_{n+1} = Y_s, \quad z_{n+1} = Z_s. \tag{3c}$$

One step of the numerical solution of problem (1) by the ESDIRK method is reduced to the successive solution of $s-1$ systems of nonlinear algebraic equations (3b), while the dimension of each is equal to the total number of differential and algebraic variables. Algebraic equations are usually solved by a modified Newton method, in which the Jacobian matrix is frozen for several integration steps. In order to reduce the number of iterations, a formula is employed for predicting the values of the vectors Y_i^0 and Z_i^0 used for starting iterations [1, 3, 5, 7–11]. In [10], in solving the ODE it was proposed to predict not only variables but also their derivatives (i.e., for vectors F_i). This made it possible to reduce the number of computations to one per preliminary stage (at the final stage one or two computations are performed). Detailed descriptions of such implementation schemes for the second-order methods are given in [11]. Here, a similar approach is extended to the case of DAE solution. Detailed descriptions are given of the implementation algorithms of specific ESDIRK methods of the 3rd and 4th orders.

2. IMPLEMENTATION SCHEMES

2.1. General Scheme

In implementing implicit methods, it is convenient to deal with increments that are denoted by $\Delta Y_i = Y_i - y_n$ and $\Delta Z_i = Z_i - z_n$. We designate by f_y, f_z, g_y, and g_z the corresponding matrices of partial derivatives that are computed at some point of the numerical solution (on the assumption that these matrices are not changed during a few steps). Iterations of modified Newton's method under the implementation of the ith stage will be written as

$$\begin{bmatrix} I - h\gamma f_y & -h\gamma f_z \\ -g_y & -g_z \end{bmatrix}\begin{bmatrix} \Delta Y_i^k - \Delta Y_i^{k-1} \\ \Delta Z_i^k - \Delta Z_i^{k-1} \end{bmatrix} = \begin{bmatrix} h\sum_{j=1}^{i-1} a_{ij}F_j + h\gamma F_i^{k-1} - \Delta Y_i^{k-1} \\ G_i^{k-1} \end{bmatrix}, \quad k = 1,\ldots,N, \tag{4a}$$

$$\left.\begin{aligned} Y_i^j = y_n + \Delta Y_i^j, \quad Z_i^j = z_n + \Delta Z_i^j, \\ F_i^j = f\left(t_n + c_i h, Y_i^j, Z_i^j\right), \quad G_i^j = g\left(t_n + c_i h, Y_i^j, Z_i^j\right), \end{aligned}\right\} \quad j = 1,\ldots,N-1, \tag{4b}$$

$$\Delta Y_i = \Delta Y_i^N, \quad \Delta Z_i = \Delta Z_i^N, \quad Y_i = y_n + \Delta Y_i, \quad Z_i = z_n + \Delta Z_i, \tag{4c}$$

where I is an identity matrix of appropriate size and N is the number of iterations.

Before starting the iterations, we must specify the initial values ΔY_i^0, ΔZ_i^0, F_i^0, and G_i^0, and after the iterations have been completed we are to calculate the value of F_i, which will be used at subsequent stages or (when $i = s$) at the next step. At the first stage, according to (3a), we take

$$F_1 = f_n, \tag{5}$$

and after completion of the last stage we set

$$y_{n+1} = Y_s, \quad z_{n+1} = Z_s, \quad f_{n+1} = F_s. \tag{6}$$

Thus, the implementation scheme of a single step of the ESDIRK method is given by (4)–(6) and calculation formulas ΔY_i^0, ΔZ_i^0, F_i^0, G_i^0, and F_i. Let us consider specific schemes.

2.2. Trivial (T) Scheme

The easiest way is to use a trivial prediction, wherein the values obtained at the initial point of the integration step are applied as initial values for iterations. Thus, the trivial scheme is given by the values

$$\Delta Y_i^0 = 0, \quad \Delta Z_i^0 = 0, \quad F_i^0 = f_n, \quad G_i^0 = g(t_n, y_n, z_n), \quad F_i = f(t_n + c_i h, Y_i, Z_i). \tag{7}$$

2.3. Trivial Modified (TM) Scheme

We can reduce the number of computations for functions f and g. From the relation $0 = g(t_n, y_n, z_n)$, we obtain $G_i^0 = 0$, and from the formula for the calculation of the stage values $\Delta Y_i = h\sum_{j=1}^{i-1} a_{ij}F_j + h\gamma F_i$ we have

$$F_i = \frac{1}{\gamma}\left(\frac{\Delta Y_i}{h} - \sum_{j=1}^{i-1} a_{ij}F_j\right). \tag{8}$$

A similar method for calculating F_i, a smoothed derivative, was employed in [3] for the implementation of the second-order method TR-BDF2. Several examples were presented there to demonstrate that this method makes it possible to significantly reduce the computational costs for the solution of stiff problems.

2.4. Standard (S) Scheme

In order to reduce the number of iterations, a nontrivial prediction is used, in which the initial values for the iteration are given as a linear combination of the previous stage values. Such a prediction is used in many solvers (e.g., RADAU5 [1]); therefore, it is often referred to as standard. As applied to DIRK methods in [1, 3, 5], it is proposed to construct the prediction formula only using the stage values of the current step. At the first stages it is advisable also to use the values of the previous step. A two-step prediction is written in the form

$$\Delta Y_i^0 = \sum_{j=1}^{s-1}\alpha_{ij}\bar{Y}_j + \sum_{j=1}^{i-1}\beta_{ij}Y_j, \quad \Delta Z_i^0 = \sum_{j=1}^{s-1}\alpha_{ij}\bar{Z}_j + \sum_{j=1}^{i-1}\beta_{ij}Z_j, \tag{9}$$

where $\bar{Y}_j$ and $\bar{Z}_j$ are stage values obtained at the previous step. Some formulas for calculating the prediction coefficients are given in [10].

The initial value of Y_i^0 has order q if $Y_i^0 - Y_i^* = O\left(h^{q+1}\right)$, where Y_i^* is the exact solution of Eqs. (3b). Generally, the prediction order coincides with the stage order of the method; then, the prediction formula can be specified as the value of the interpolation polynomial constructed from $q + 1$ stage values. After calculating the initial values of (9) we assume

$$F_i^0 = f\left(t_n + c_i h, Y_i^0, Z_i^0\right), \quad G_i^0 = g\left(t_n + c_i h, Y_i^0, Z_i^0\right), \tag{10}$$

and after completion of iterations (4), we compute F_i using formula (8).

2.5. Economical (E) Scheme

Making use of a nontrivial prediction (9) reduces the number of iterations but necessitates additional calculations by formulas (10). Instead of this, the economical scheme employs a prediction for calculating the initial values of vectors F_i^0 and G_i^0. Assuming that algebraic relations (1b) are satisfied with sufficient accuracy at all stages, we obtain

$$F_i^0 = f_n + \sum_{j=1}^{s-1}\alpha_{ij}\bar{F}_i + \sum_{j=1}^{i-1}\beta_{ij}F_j, \quad G_i^0 = 0. \tag{11}$$

Therefore, the economical scheme is given by formulas (9), (11), (4), and (8). In [10], such a scheme was considered for the case of an ODE system. It should be noted that a similar scheme was used in [7] for implementation of the theta method.

The DAE system is often given in the following form:

$$My' = f(t, y), \quad y(t_0) = y_0$$

where square matrix M has an incomplete rank. In solving such problems, iterations (4a) can be written as

$$\left(M - h\gamma f_y\right)\left(\Delta Y_i^k - \Delta Y_i^{k-1}\right) = h\sum_{j=1}^{i-1} a_{ij}F_j + h\gamma F_i^{k-1} - M\Delta Y_i^{k-1}, \quad k = 1, \ldots, N,$$

Table 1. Solution errors of problem (12)

nf	Scheme			
	T	TM	S	E
1	1.09×10^{-2}	4.45×10^{-4}	2.04×10^{-5}	2.84×10^{-7}
2	2.18×10^{-4}	3.60×10^{-5}	4.94×10^{-7}	3.50×10^{-7}
3	2.15×10^{-5}	3.54×10^{-6}	3.59×10^{-7}	3.53×10^{-7}
4	2.33×10^{-6}	5.06×10^{-7}	3.54×10^{-7}	3.53×10^{-7}
5	4.50×10^{-7}	3.54×10^{-7}	3.53×10^{-7}	3.53×10^{-7}

and instead of (8) we use the formula

$$F_i = \frac{1}{\gamma}\left(\frac{M\Delta Y_i}{h} - \sum_{j=1}^{i-1} a_{ij}F_j\right).$$

3. EFFICIENCY OF ITERATION SCHEMES

The convergence of iteration schemes for solving the Cauchy problem was studied in [12, 13]. Under certain conditions, each iteration with an inaccurate Jacobian matrix increases the order of the scheme by 1 until the order of the method has been reached. Therefore, in implementing a method of order p by a trivial scheme no less than p iterations must be used. Attaining the desired order on using a prediction of order q requires no less than $p-q$ iterations.

Many applied problems are described by expressions, whose implementation takes a significant portion of computer time. Let us consider, for instance, Andrew's squeezing mechanism described by an DAE system of index 3. The equations are given in [1, Section VII.7] and contain a rather large number of operations. They were solved on the interval $0 \le t \le 0.3$ by the ESDIRK method of the 4th order, implemented using a software package (SP) MVTU [14, 15]. The process of solution included 309 738 computations of the right-hand side and 14 070 computations of the Jacobian matrix (the Jacobian was calculated using finite differences, which required another 379 890 computations of the right-hand side). The solution required 14.3s of PC time, with a CPU frequency of 3 GHz, of which 4.8 s was spent on the calculations of the right-hand side, 5.9s was spent on the calculations of the Jacobian, and 3.6s was spent on the rest of the computations. As we can see, even for solving a relatively simple problem, the calculations of the right-hand side and the Jacobian required significantly more time than all the other computations.

More complex technical objects contain elements of different physical nature (mechanical, electrical, hydraulic, etc.), many of which are described by cumbersome nonlinear dependences. In the simulation of such objects using the SP MVTU, the calculation cost for the right-hand side and the Jacobian can account for 90% of the total computation time. At the same time, solution of a linear system with a prefactorized coefficient matrix involves relatively low computation costs. Therefore, as a key parameter in assessing the complexity of iterations with a frozen Jacobian matrix, we take the number of computations of the right-hand side, which coincides with the number of iterations for schemes T and S and is lower by 1 than the number of iterations for schemes TM and E. For all schemes, the number of solutions of a linear system coincides with the number of iterations.

We evaluate the effectiveness of these schemes for a problem

$$y_1' = -102y_1 + 100y_2^2, \quad y_2' = y_1 - y_2(1+z), \quad 0 = y_2 - z + 0.1\left(y_1 - z^2\right), \qquad (12)$$
$$y_1(0) = y_2(0) = z(0) = 1, \quad 0 \le t \le 1,$$

whose exact solution is $y_1(t) = \exp(-2t)$, $y_2(t) = z(t) = \exp(-t)$. Here, the 4th-order method DIRK54, described in Section 5, is used. We designate by *nf* the number of computations of the right-hand side at each implicit stage and as an indicator of accuracy take the Euclidean norm of the error vector at the end of the interval. The results for using these schemes for a single calculation of the Jacobian matrix over the entire interval are shown in Table 1.

Iterations should be stopped at the moment when it is no longer feasible to perform the next iteration (i.e., in order to achieve the same accuracy, it would be more appropriate to reduce the step size than to make another iteration). From this viewpoint, the E-scheme secures an acceptable convergence by a sin-

gle calculation of the right-hand side at each implicit stage, whereas the S-scheme requires two computations. Similar results are presented in [10]. They show that the E-scheme saves one calculation of the right-hand side at each implicit stage compared to the standard scheme.

Test problems confirmed that for methods up to the 4th order with the prediction not lower than the 2nd order, it is quite possible to confine ourselves to two iterations. For the E-scheme, this corresponds to a single computation of the right-hand side at each implicit stage. An additional iteration at the last stage allows us to estimate the convergence rate of the iterations and develop a criterion for the update of the Jacobian matrix. Therefore, in the schemes below at each implicit stage, except the last one, we perform two iterations and a single computation of the right-hand side, and at the last stage, we perform three iterations and two computations of the right-hand side. Since the first (explicit) stage does not require computations, the number of computations of the right-hand side at one step coincides with the number of stages s.

An increase in the step size or inaccuracy of the Jacobian matrix may reduce the convergence rate or even lead to the divergence of iterations. In this case, we should reduce the step size or update the Jacobian matrix. Thus, the convergence of the iterations is automatically adjusted by the algorithms of change in the step size and updating of the Jacobian matrix. For this purpose, it is necessary at every step to estimate the local error and the convergence of the iterations

4. CHANGE IN THE STEP SIZE AND UPDATING THE JACOBIAN MATRIX

All schemes employ a standard procedure for adjusting the step size [1]. As an embedded method for error estimation, we use the prediction of the last stage whose order is lower than the main method's by 1 (the advantage of such a method for error assessment is justified in [11]). The local error assessment is obtained in the form

$$\delta y = \begin{bmatrix} \Delta Y_s - \Delta Y_s^0 \\ \Delta Z_s - \Delta Z_s^0 \end{bmatrix} \tag{13}$$

(in the general case it is assumed that vector δy contains differential and algebraic components of the error estimate). If the DAE index is higher than 1, there emerge components of the error estimate proportional to negative powers of h [16, Chapter 8]. As a result, the step size can be reduced abruptly leading to an emergency stop of the numerical solution (usually a message is generated about an overflow or division by zero). This situation can be prevented either by a special scaling or ignoring the components of the estimate (13) for the higher index variables (see [16]).

The normalized error estimate is taken as $\delta = \mathrm{normer}(\delta y)$, where

$$\mathrm{normer}(\delta y) = \max_i \left(\frac{|\delta y_i|}{Atol + Rtol \times \max(|y_{ni}|, |y_{n+1,i}|)} \right),$$

$Atol$ is the permissible absolute error, $Rtol$ is the admissible relative error, and δy_i, y_{ni}, and $y_{n+1,i}$ are the ith components of the respective vectors. If $\delta \le \delta_{\max}$, the step is considered successful, otherwise the step is discarded (in all schemes, we assume $\delta_{\max} = 2$). The new step size is given as

$$h_{\mathrm{new}} = w_{\mathrm{new}} h, \quad w_{\mathrm{new}} = \begin{cases} 1, & |1 - w_0| \le 0.1, \\ w_0, & |1 - w_0| > 0.1, \end{cases} \quad w_0 = \max\left(1/8, \min\left(8, 0.8\delta^{-1/p}\right)\right),$$

where p is the order of the method.

The Jacobian matrix can only be updated after a successful step. In this case we calculate

$$\delta_1 = \mathrm{normer}\left(\Delta Y_s^2 - \Delta Y_s^1\right), \quad \delta_2 = \mathrm{normer}\left(\Delta Y_s^3 - \Delta Y_s^2\right), \quad \theta = \delta_2/\delta_1.$$

The quantity θ is the estimate of iterations convergence rate and the error of the last iteration is estimated by the value $\varepsilon = \theta\delta_2/(1-\theta)$. The Jacobian matrix is updated when at least one of the following two conditions is fulfilled:

$$\theta > \theta_{\max} \ \text{ or } \ \varepsilon > K\delta, \tag{14}$$

where the coefficients $\theta_{\max}$ and K are adjusted experimentally.

5. METHODS

We consider three methods, which will be denoted by DIRK43, DIRK54, and DIRK64. The first digit in the name is the number of stages and the second digit is the order of the method. In the respective

implementation schemes, the number of computations of the right-hand side at one step coincides with the number of stages. For a complete description of the schemes, it remains to determine the coefficients of the methods (3) and the coefficients of prediction formulas (9), (11).

5.1. Method DIRK43

This is the method proposed in [5] with coefficients

$$\gamma = a_{21} = 0.158983899988677,\quad c_2 = 2\gamma,\quad c_3 = \left(2+\sqrt{2}\right)\gamma,\quad c_4 = 1,$$

$$a_{31} = a_{32} = \frac{c_3-\gamma}{2},\quad a_{43} = \left(\sqrt{2}-1\right)\frac{6\gamma^2-6\gamma+1}{6\gamma^2},\quad a_{41} = a_{42} = \frac{1-a_{43}-\gamma}{2}.$$

It is L(α)-stable when $\alpha = 75.6°$.

We designate by $w = h/\overline{h}$ the ratio of the current step to the size of the previous step. The next integration step begins with an explicit stage (5). Before the start of iterations at each implicit stage, we calculate the initial values using formulas (9) and (11). At all stages, a second-order prediction is used with coefficients

$$\alpha_{21} = \frac{wc_2}{c_3}(wc_2-c_3+1),\quad \alpha_{22} = 0,\quad \alpha_{23} = \frac{wc_2(wc_2+1)}{c_3(c_3-1)},\quad \beta_{21} = -\alpha_{21}-\alpha_{23},$$

$$\alpha_{31} = \alpha_{32} = 0,\quad \beta_{31} = \frac{wc_3(c_3-c_2)}{c_2(c_3-1)} - \frac{c_3}{c_2},\quad \beta_{32} = \frac{c_3(wc_3-c_3+1)}{c_2(wc_2-c_3+1)},\quad \alpha_{33} = -\beta_{31}-\beta_{32},$$

$$\alpha_{41} = \alpha_{42} = \alpha_{43} = 0,\quad \beta_{42} = \frac{1-c_3}{c_2(c_2-c_3)},\quad \beta_{43} = \frac{1-c_2}{c_3(c_3-c_2)},\quad \beta_{41} = -\beta_{42}-\beta_{43}.$$

At the first stage, for all methods, we assume

$$\alpha_{ij} = 0,\quad 2 \le i \le s-1,\quad 1 \le j \le s-1;\quad \beta_{21} = 0,\quad \beta_{31} = -c_3/c_2,\quad \beta_{32} = c_3/c_2.$$

Once the prediction has been computed, iterations (4) are performed, i.e., two iterations at $i = 2, 3$ and three iterations under $i = 4$. Upon completion of the iterations, we use formula (8) to calculate vector F_i.

5.2. Method DIRK54

This method is proposed in [5]. It is L(α)-stable at $\alpha = 89.56°$ and has coefficients

$$\gamma = a_{21} = 0.220428410259212,\quad c_2 = 2\gamma,\quad c_3 = 0.752589667839344,$$
$$c_4 = 0.610097451414243,\quad c_5 = 1,\quad a_{31} = a_{32} = 0.266080628790066,$$
$$a_{41} = a_{42} = 0.227031047465079,\quad a_{43} = -0.064393053775127,$$
$$a_{51} = a_{52} = 0.175575441883476,\quad a_{53} = -0.415534431720558,$$
$$a_{54} = 0.843955137694394.$$

The prediction has a second order at preliminary stages and a third order at the final fifth stage. Nonzero coefficients of the prediction formulas have the form

$$\alpha_{21} = \frac{wc_2}{c_4}(wc_2-c_4+1),\quad \alpha_{24} = \frac{wc_2(wc_2+1)}{c_4(c_4-1)},\quad \beta_{21} = -\alpha_{21}-\alpha_{24},$$

$$\beta_{31} = \frac{wc_3(c_3-c_2)}{c_2(c_4-1)} - \frac{c_3}{c_2},\quad \beta_{32} = \frac{c_3(wc_3-c_4+1)}{c_2(wc_2-c_4+1)},\quad \alpha_{34} = -\beta_{31}-\beta_{32},$$

$$\beta_{42} = \frac{c_4(c_4-c_3)}{c_2(c_2-c_3)},\quad \beta_{43} = \frac{c_4(c_4-c_2)}{c_3(c_3-c_2)},\quad \beta_{41} = -\beta_{42}-\beta_{43},$$

$$\beta_{51} = -0.533270955358986,\quad \beta_{52} = -2.23348959717643,$$
$$\beta_{53} = 2.08190712545191,\quad \beta_{54} = -\beta_{51}-\beta_{52}-\beta_{53}.$$

5.3. Method DIRK64

This method belongs to the one-parameter family proposed in [6]. It is L(α)-stable for $\alpha = 89.95°$. It was shown in [17] that methods of this family secure higher accuracy on solving the DAEs of indices 2 and

3 than other diagonally implicit methods. By selecting the value of the free parameter $c_4 = 1/2$, we obtain the following table of coefficients:

0	0					
1/3	1/6	1/6				
8/15	31/150	4/25	1/6			
1/2	1685/8448	157/1056	−125/8448	1/6		
1/2	97/576	1/36	−625/576	11/9	1/6	
1	1/6	0	0	0	2/3	1/6
	1/6	0	0	0	2/3	1/6

The prediction has a second order at the 2nd, 3rd, and 4th stages, and at the 5th and 6th stages a prediction of the 3rd order is used. When calculating the prediction coefficients of the last stage the condition $\hat{R}(\infty) = 0$ was also taken into account, where $\hat{R}(z)$ is the stability function of the embedded method. Nonzero coefficients of the prediction formulas

$$\alpha_{21} = \frac{w}{9}(2w+3),\ \ \beta_{21} = \frac{w}{9}(2w+9),\ \ \alpha_{25} = -\alpha_{21} - \beta_{21},\ \ \alpha_{35} = \frac{1.28w^2}{2w+3},$$
$$\beta_{31} = -0.64w - 1.6,\ \ \beta_{32} = -\alpha_{35} - \beta_{31},\ \ \beta_{41} = -33/32,\ \ \beta_{42} = 1/4,\ \ \beta_{43} = 25/32,$$
$$\beta_{51} = -121/160,\ \ \beta_{52} = -39/20,\ \ \beta_{53} = -195/32,\ \ \beta_{54} = 44/5,$$
$$\beta_{61} = -109/200,\ \ \beta_{62} = 84/25,\ \ \beta_{63} = 309/8,\ \ \beta_{64} = -1056/25,\ \ \beta_{65} = 4/5.$$

6. NUMERICAL EXPERIMENTS

The considered schemes were implemented in the SP MVTU environment. In the first place, a series of experiments were conducted to determine the optimum values of $\theta_{\max}$ and K in (14). For the DIRK43 and DIRK54 methods, we obtained the values $\theta_{\max} = 0.4$ and $K = 0.2$ in accordance with the condition of the minimized number of computations for the Jacobian matrix. The main criterion for DIRK64 was the number of computations of the right-hand side, and the resulting values were $\theta_{\max} = 0.05$ and $K = 0.02$. For DAE systems of indices 2 and 3, these values do not always provide an effective and accurate solution, so, for these problems in solvers DIRK43 and DIRK54, we use values $\theta_{\max} = 0.05$ and $K = 0.02$; and in DIRK64 the Jacobian matrix is computed at each step.

We present the results of solving five problems of the test set [18], providing detailed descriptions of these problems, including the physical statement, equations, initial conditions, and the integration interval. This study also shows exact solutions at the end of the interval obtained at very small values of error tolerance. As in [18], we assume for the selected problems, $Rtol = Atol = Tol$, and the accuracy will be estimated by

$$scd = -\log_{10}\left(\max_i\left(\frac{|y_i - \tilde{y}_i|}{|y_i|}\right)\right),\quad mescd = -\log_{10}\left(\max_i\left(\frac{|y_i - \tilde{y}_i|}{Atol/Rtol + |y_i|}\right)\right),$$

where y_i is the exact solution and $\tilde{y}_i$ is the numerical solution of the ith component at the end of the integration interval (it is assumed that vector y also contains algebraic elements). The quantity *scd* is an estimate of the relative error and shows the minimum number of correct significant digits among the components of the numerical solution. If for some components we have $|y_i| \ll Atol/Rtol$, then *scd* does not allow us to correctly assess whether the error exceeds the established tolerance, because in this case the step size control is to a greater extent based on the absolute error. Therefore, in such cases, we present the *mescd* value. The computation cost is estimated by the number of calculations of the right-hand side, *Nf*, and the number of computations of the Jacobian matrix, *NJ*. For comparison, we present the results of the most efficient solvers of medium accuracy, i.e., RODAS (Rosenbrock's method of the 4th order), DASSL (a method based on backward differentiation formulas of orders one through five) and RADAU5 (the Radau IIA method of the 5th order).

Tables 2–4 present the solution for systems of ODE (problems VDPOL, OREGO, and HIRES) for the initial step size of $h_0 = 10^{-6}$. The VDPOL problem was solved in a scaled form (on the interval [0, 2]). The

Table 2. Solution results for the VDPOL problem

Solver	$Tol = 10^{-2}$			$Tol = 10^{-3}$			$Tol = 10^{-4}$		
	scd	*Nf*	*NJ*	*scd*	*Nf*	*NJ*	*scd*	*Nf*	*NJ*
DIRK43	2.30	781	17	3.08	1405	16	4.13	2961	15
DIRK54	2.41	841	21	3.36	1171	19	4.59	2106	16
DIRK64	2.78	991	67	4.11	1333	95	4.84	2575	129
RODAS	2.51	697	107	3.72	1169	179	4.75	1951	311
RADAU5	2.52	1329	92	4.31	1649	121	4.96	2253	162

Table 3. Solution results for the OREGO problem

Solver	$Tol = 10^{-2}$			$Tol = 10^{-3}$			$Tol = 10^{-4}$		
	scd	*Nf*	*NJ*	*scd*	*Nf*	*NJ*	*scd*	*Nf*	*NJ*
DIRK43	1.08	1009	52	2.41	1625	48	3.45	3221	50
DIRK54	1.46	1006	56	2.64	1461	55	3.90	2426	54
DIRK64	1.53	1243	122	2.81	1573	163	3.88	2641	200
RODAS	0.61	856	126	2.26	1292	202	3.62	2171	351
RADAU5	2.99	1968	147	4.12	2166	183	4.47	2781	236

Table 4. Solution results for the HIRES problem

Solver	$Tol = 10^{-3}$			$Tol = 10^{-4}$			$Tol = 10^{-5}$		
	mescd	*Nf*	*NJ*	*mescd*	*Nf*	*NJ*	*mescd*	*Nf*	*NJ*
DIRK43	3.61	157	10	4.09	253	9	5.08	473	9
DIRK54	3.52	161	10	4.41	206	10	7.08	361	11
DIRK64	3.21	199	18	4.61	265	25	5.87	385	37
RODAS	2.19	137	22	4.19	196	31	4.99	291	47
RADAU5	2.89	259	18	2.93	333	21	5.24	406	25

Table 5. Solution results for the Chemical Akzo Nobel problem

Solver	$Tol = 10^{-4}$			$Tol = 10^{-5}$			$Tol = 10^{-7}$		
	mescd	*Nf*	*NJ*	*mescd*	*Nf*	*NJ*	*mescd*	*Nf*	*NJ*
DIRK43	4.66	113	4	5.61	197	5	7.56	781	4
DIRK54	4.90	106	5	5.57	161	5	7.36	411	4
DIRK64	6.00	127	13	6.72	205	15	8.17	475	17
DASSL	4.55	81	12	4.89	128	14	6.90	222	26
RADAU5	–	–	–	–	–	–	8.50	372	33

Table 6. Solution results for the Car Axis problem

Solver	$Tol = 10^{-4}$			$Tol = 10^{-6}$			$Tol = 10^{-8}$		
	mescd	*Nf*	*NJ*	*mescd*	*Nf*	*NJ*	*mescd*	*Nf*	*NJ*
DIRK43	1.46	1137	53	2.62	4525	63	3.44	20425	64
DIRK54	1.17	781	50	3.05	2371	55	4.12	9716	57
DIRK64	1.46	805	131	3.68	2827	471	5.19	12553	2092
RADAU5	1.34	850	95	2.93	1742	196	4.53	3783	411

results of the solvers RODAS and RADAU5 were obtained using an interactive system ODELab [19]. Results of other well-known solvers can be found in [18]. Table 5 present the solution for a system of DAE of index 1 (Chemical Akzo Nobel problem) at $h_0 = Tol$. The results of DASSL and RADAU5 are taken from [20, 21]. Some well-known solvers could not cope with this problem with moderate accuracy (including RADAU5 at $Tol = 10^{-4}, 10^{-5}$). Table 6 shows the solution results for a system of DAE of index 3 (Car Axis problem) at $h_0 = Tol$. Results of RADAU5 are taken from [20]. It should be noted that solvers RODAS and DASSL are not suitable for the solution of problems of index 3.

7. CONCLUSIONS

The proposed schemes are simple to implement and are not inferior to the best implicit solvers in solving stiff and differential-algebraic problems with low or medium accuracy. The advantage of these schemes is also a small number of computations of the Jacobian matrix, which is especially important, for the numerical calculations of a Jacobian.

REFERENCES

1. E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems* (Springer, Berlin, 1996).
2. R. Alexander, "Diagonally implicit Runge–Kutta methods for stiff O.D.E.'S," SIAM J. Numer. Anal **14** (6), 1006–1021 (1977).
3. M. E. Hosea and L. F. Shampine, "Analysis and implementation of TR-BDF2," Appl. Numer. Math **20** (1-2), 21–37 (1996).
4. A. Kvœrnø, "Singly diagonally implicit Runge–Kutta methods with an explicit first stage," BIT **44** (3), 489–502 (2004).
5. L. M. Skvortsov, "Diagonally implicit Runge–Kutta FSAL methods for stiff and differential-algebraic systems," Mat. Model. **14** (2), 3–17 (2002).
6. L. M. Skvortsov, "Diagonally implicit Runge–Kutta methods for stiff problems," Comput. Math. Math. Phys. **46** (12), 2110–2123 (2006).
7. M. Berzins and R. M. Furzeland, "An adaptive theta method for the solution of stiff and nonstiff differential equations," Appl. Numer. Math. **9** (1), 1–19 (1992).
8. H. Olsson and G. Söderlind, "Stage value predictors and efficient newton iterations in implicit Runge-Kutta methods," SIAM J. Sci. Comput **20** (1), 185–202 (1998).
9. G. Yu. Kulikov, "Numerical solution of the Cauchy problem for a system of differential-algebraic equations with the use of implicit Runge–Kutta methods with a nontrivial predictor," Comput. Math. Math. Phys. **38** (1), 64–80 (1998).
10. L. M. Skvortsov, "An efficient scheme for the implementation of implicit Runge–Kutta methods," Comput. Math. Math. Phys. **48** (11), 2007–2017 (2008).
11. L. M. Skvortsov, "Efficient implementation of second-order implicit Runge–Kutta methods," Math. Models Comput. Simul. **5** (6), 565–574 (2013).
12. G. Yu. Kulikov, "Convergence theorems for iterative Runge–Kutta methods with a constant integration step," Zh. Vychisl. Mat. Mat. Fiz. **36** (8), 73–89 (1996).

13. K. R. Jackson, A. Kvœrnø, and S. P. Norsett, "An analysis of the order of Runge–Kutta methods that use an iterative scheme to compute their internal stage values," BIT **36** (4), 713–765 (1996).
14. O. S. Kozlov, D. E. Kondakov, L. M. Skvortsov, et al., "Programmnyi kompleks dlya issledovaniya dinamiki i proektirovaniya tekhnicheskikh sistem," Inf. Tekhnol, No. 9, 20–25 (2005).
15. O. S. Kozlov, D. E. Kondakov, L. M. Skvortsov, et al. http://model.exponenta.ru/mvtu/20050615.html.
16. E. Hairer, Ch. Lubich, and M. Roche, *The Numerical Solution of Differential-Algebraic Systems by Runge-Kutta Methods* (Springer, Berlin, 1989), Lecture Notes in Mathematics, Vol. 1409.
17. L. M. Skvortsov, "Diagonally implicit Runge–Kutta methods for differential algebraic equations of indices two and three," Comput. Math. Math. Phys. **50** (6), 993–1005 (2010).
18. F. Mazzia and C. Magherini, Test Set for Initial Value Problem Solvers, Release 4, 2008. www.dm.uniba.it/~testset/report/testset.pdf
19. http://num-lab.zib.de/public/odelab/.
20. L. Brugnano, C. Magherini, and F. Mugnai, "Blended implicit methods for the numerical solution of DAE problems," J. Comput. Appl. Math. **189** (1–2), 34–50 (2006).
21. http://web.math.unifi.it/users/brugnano/BiM/BiMD/index_BiMD.htm.

Translated vy I. Pertsovskaya