BIT Numerical Mathematics (2023) 63:27
https://doi.org/10.1007/s10543-023-00967-x

BIT

Construction of Rosenbrock–Wanner method Rodas5P and numerical benchmarks within the Julia Differential Equations package

Gerd Steinebach[1]

Received: 18 July 2022 / Accepted: 20 March 2023 / Published online: 17 April 2023

Abstract
Rosenbrock–Wanner methods for systems of stiff ordinary differential equations are well known since the seventies. They have been continuously developed and are efficient for differential-algebraic equations of index-1, as well. Their disadvantage that the Jacobian matrix has to be updated in every time step becomes more and more obsolete when automatic differentiation is used. Especially the family of Rodas methods has proven to be a standard in the Julia package DifferentialEquations. However, the fifth-order Rodas5 method undergoes order reduction for certain problem classes. Therefore, the goal of this paper is to compute a new set of coefficients for Rodas5 such that this order reduction is reduced. The procedure is similar to the derivation of the methods Rodas4P and Rodas4P2. In addition, it is possible to provide new dense output formulas for Rodas5 and the new method Rodas5P. Numerical tests show that for higher accuracy requirements Rodas5P always belongs to the best methods within the Rodas family.

Keywords Rosenbrock–Wanner methods · Index-1 DAEs · Order reduction · Julia package Differential Equations · Rodas5

Mathematics Subject Classification 65L04 · 65L80 · 68N15

1 The Rodas family in Julia DifferentialEquations package

Numerical programming in Julia has proven to be very performant. Rackauckas and Nie [16] implemented the powerful package `DifferentialEquations.jl` that

Handling Editor: Antonella Zanna Munthe-Kaas.

✉ Gerd Steinebach
Gerd.Steinebach@h-brs.de

[1] Bonn-Rhein-Sieg University of Applied Sciences, Sankt Augustin, Germany

contains a wide range of solvers for several types of problems. We restrict our considerations to initial value problems of the type

$$M\, y' = f(t, y),\ \ y(t_0) = y_0\,. \tag{1.1}$$

When matrix M is singular, (1.1) is a system of differential-algebraic equations (DAEs), else a system of ordinary differential equations (ODEs). We assume problem (1.1) to be of index not greater than one. For a detailed definition of the index concept see [7]. For solving such problems Rosenbrock–Wanner (ROW) methods are well known, see [4] and [8] for a recent survey.

A ROW scheme with stage-number s for problem (1.1) is defined by:

$$\begin{aligned}(M - h\,\gamma\, f_y)k_i \ &= hf\left(t_0 + \alpha_i h, y_0 + \sum_{j=1}^{i-1} \alpha_{ij} k_j\right) + h\, f_y \sum_{j=1}^{i-1} \gamma_{ij} k_j + h^2 \gamma_i f_t, \\ &\qquad i = 1, ..., s, \end{aligned} \tag{1.2}$$

$$y_1 \ = y_0 + \sum_{i=1}^{s} b_i k_i,$$

$$\text{with} \quad f_y = \frac{\partial f}{\partial y}(t_0, y_0), \quad f_t = \frac{\partial f}{\partial t}(t_0, y_0). \tag{1.3}$$

h is the stepsize and y_1 is the approximation of the solution $y(t_0+h)$. The coefficients of the method are $\gamma, \alpha_{ij}, \gamma_{ij}$, and b_i define the weights. Moreover, it holds $\alpha_i = \sum_{j=1}^{i-1} \alpha_{ij}$ and $\gamma_i = \gamma + \sum_{j=1}^{i-1} \gamma_{ij}$.

ROW methods are alinearly implicit schemes since only a fixed number of s linear systems have to be solved. The index-1 condition guarantees the regularity of the matrix $(M - h\,\gamma\, f_y)$ for sufficiently small stepsizes $h > 0$, see [4]. A disadvantage compared to implicit Runge–Kutta methods is the requirement of evaluating the Jacobian matrix f_y in every timestep.

Within the Julia package `DifferentialEquations.jl` it is possible to compute the Jacobian by automatic differentiation. Therefore, ROW methods proved to be very efficient for the solution of stiff ODEs and DAEs. For the following analysis we choose `ROS3P` [10], `Rodas3` [21], `Rodas4` [4], `Rodas4P` [24], `Rodas4P2` [25] and `Rodas5` [3] from the many implemented ROW methods.
Moreover, we include `Ros3prl2` and `Rodas4PR2` [17] which are successors of `ROS3P` respectively `Ros3PL` [9] and `Rodas4P`, but not yet implemented in `DifferentialEquations.jl`. These schemes are applicable to index-1 DAEs and are stiffly stable. Stiffly stable methods guarantee $R(\infty) = 0$ for the stability function $R(z)$, which is a desired property when solving problem (1.1), see [4, 8]. Since in addition all these methods are A-stable they are L-stable as well. The best known method is certainly `Rodas4` from Hairer and Wanner [4]. The other schemes considered here were constructed based on this inspiration.

Table 1 Stages s and order of convergence for different problems: Index-1 DAE problem (DAE-1), Prothero-Robinson model (Prot-Rob), parabolic problem (Parabol), index-2 DAE problem (DAE-2), DAE problem with inexact Jacobian (inexact Jac)

	ROS3P	Ros3prl2	Rodas3	Rodas4	Rodas4PR2	Rodas4P2	Rodas5
Stages s	3	4	4	6	6	6	8
DAE-1	3 (2)	3 (2)	3 (2)	4 (3)	4 (3)	4 (3)	5 (4)
Prot-Rob	2 (2)	3 (2)	1 (2)	1 (1)	4 (3)	3 (3)	1 (1)
Parabol	3 (2)	3 (2)	2 (2)	2 (2)	4 (3)	4 (3)	2 (2)
DAE-2	2 (1)	2 (1)	1 (2)	1 (1)	2 (2)	2 (2)	1 (1)
inexact Jac	1 (1)	1 (1)	2 (1)	1 (1)	1 (1)	2 (1)	1 (1)

The numbers in brackets denote the order of the embedded methods

It is well known that ROW methods suffer from order reduction when they are applied to the Prothero-Robinson model, see [15, 18, 23]:

$$y' = \lambda\,(y - g(t)) + g'(t), \quad y(0) = g(0). \tag{1.4}$$

For a large stiffness parameter $|\lambda|$ with $\Re(\lambda) << 0$ the order may even drop to one. Scholz [23] and Ostermann and Roche [14] derived additional conditions to be fullfilled such that the order is independent on λ. Ostermann and Roche pointed out that the same conditions occur when semi-discretized parabolic partial differential equations (PDEs) are considered. Methods `ROS3P`, `ROS3PL`, `Ros3prl2`, `Rodas4P`, `Rodas4PR2` and `Rodas4P2` were developed according to these additional conditions. The letter P stands for "Prothero-Robinson" as well as "parabolic problem".

An alternative way to avoid order reduction is considered in [1, 2]. Here, the method does not have to fulfill any additional order conditions. In order to achieve a higher stage order, adapted boundary conditions of the partial differential equation are considered in the calculation of the individual stages. The advantage is that every ROW method is suitable for this. The disadvantage is that additional information about the problem to be solved must be included in the stage evaluation. This may become complicated when pipe networks are considered and thus the boundary conditions must take coupling information into account [26]. A numerical comparison of the two approaches is given in Sect. 4.

W methods for ODEs are ROW methods which do not need an exact Jacobian matrix f_y in every timestep. Examples for such methods can be found in [17]. Recently, Jax [5] was able to enlarge the class of certain W methods to problems of differential algebraic equations. Unfortunately a huge amount of additional order conditions have to be satiesfied as well. Conditions up to order two are fullfilled by the `Rodas4P2` method.

Table 1 summarizes the properties of the schemes considered. The order of convergence for some test problems was obtained numerically by the solution with different constant timesteps. The definition of the test problems is given in Sect. 4. Despite the fact that `Rodas4` and `Rodas5` are very efficient for a couple of typical test problems [4], they show remarkable order reduction for special problems.

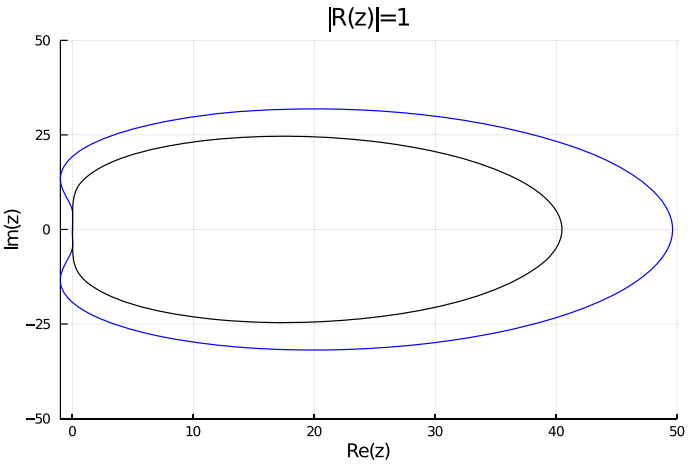

Fig. 1 Values $z \in \mathbb{C}$ with stability function $|R(z)| = 1$, black for `Rodas5`, blue for its embedded method (colour figure online)

`Rodas5` has some further disadvantages. For simple non-autonomous problems such as $y' = \cos(t)$, $y(0) = 0$, errors of this method and its embedded scheme are exactly the same. This leads to a failure of the stepsize control. The embedded method of `Rodas5` is not A-stable. In Fig. 1 we can see that the stability domain does not contain the whole left complex half-plane. This may cause stepsize reductions for problems with eigenvalues near the imaginary axis. Moreover, the original literature [3] does not contain a coefficient set for a dense output formula of `Rodas5`. In the Julia implementation a Hermite interpolation is used which is only applicable to ODE problems.

The aim of this paper therefore is to construct a new coefficient set for `Rodas5`. It should still have order 5(4) for standard DAE problems of index-1, but its order reduction shown in Table 1 should be restricted to that of `Rodas4P2`. Moreover, the embedded method should be A-stable and a dense output at least of order $p = 4$ should be provided. In Sect. 2, all order conditions to be fullfilled by the new method `Rodas5P` are stated. The construction and the computation of the coefficients of the method is explained in Sect. 3, and finally, in Sect. 4, some numerical benchmarks are given.

2 Order conditions

The order conditions for Rosenbrock methods applied to index-1 DAEs of type (1.1) were derived by Roche [20]. They are connected to Butcher trees, as shown in Tables 2 and 3. Table 2 lists the conditions up to order $p = 5$ for ODE problems and Table 3 the additional order conditions up to order $p = 5$ for index-1 DAE problems, see [3, 4].

Table 2 Order conditions up to order $p = 5$ for ODE problems

No	Order	Tree	Condition
1	1		$\sum b_i = 1$
2	2		$\sum b_i \beta_i = 1/2$
3	3		$\sum b_i \alpha_i^2 = 1/3$
4			$\sum b_i \beta_{ij} \beta_j = 1/6$
5	4		$\sum b_i \alpha_i^3 = 1/4$
6			$\sum b_i \alpha_i \alpha_{ij} \beta_j = 1/8$
7			$\sum b_i \beta_{ij} \alpha_j^2 = 1/12$
8			$\sum b_i \beta_{ij} \beta_{jk} \beta_k = 1/24$
9	5		$\sum b_i \alpha_i^4 = 1/5$
10			$\sum b_i \alpha_i^2 \alpha_{ij} \beta_j = 1/10$
11			$\sum b_i \alpha_{ij} \beta_j \alpha_{ik} \beta_k = 1/20$
12			$\sum b_i \alpha_i \alpha_{ij} \alpha_j^2 = 1/15$
13			$\sum b_i \alpha_i \alpha_{ij} \beta_{jk} \beta_k = 1/30$
14			$\sum b_i \beta_{ij} \alpha_j^3 = 1/20$
15			$\sum b_i \beta_{ij} \alpha_j \alpha_{jk} \beta_k = 1/40$
16			$\sum b_i \beta_{ij} \beta_{jk} \alpha_k^2 = 1/60$
17			$\sum b_i \beta_{ij} \beta_{jk} \beta_{kl} \beta_l = 1/120$

The following abbreviations are used:

$$\beta_{ij} = \alpha_{ij} + \gamma_{ij} \quad \text{with} \quad \beta_{ij} = 0 \ \text{ for } \ i < j \ \text{ and } \beta_{ii} = \gamma_{ii} = \gamma, \tag{2.1}$$

$$\beta_i = \sum_{j=1}^{i} \beta_{ij} \quad \alpha_i = \sum_{j=1}^{i-1} \alpha_{ij}, \quad B = (\beta_{ij})_{i,j=1}^{s} \quad W = B^{-1} = (w_{ij})_{i,j=1}^{s}. \tag{2.2}$$

The sums in the tables are formed over all possible indices.

In Table 4 additional order conditions are defined. Conditions No. 41–44 are given in [11] for problems of type $M(y) \cdot y' = f(y)$ with singular matrix $M(y)$. Based on these conditions the method `rowdaind2` was derived in [11]. In the special case of index-2 DAEs of type

$$y' = f(y, z), \tag{2.3}$$

$$0 = g(y) \tag{2.4}$$

Table 3 Additional order conditions up to order $p = 5$ for index-1 DAE problems

No	Order	Tree	Condition
18	3		$\sum b_i w_{ij} \alpha_j^2 = 1$
19	4		$\sum b_i \alpha_i \alpha_{ij} w_{jk} \alpha_k^2 = 1/4$
20			$\sum b_i w_{ij} \alpha_j^3 = 1$
21			$\sum b_i w_{ij} \alpha_j \alpha_{jk} \beta_k = 1/2$
22			$\sum b_i w_{ij} \alpha_j \alpha_{jk} w_{kl} \alpha_l^2 = 1$
23	5		$\sum b_i \alpha_i \alpha_{ij} w_{jk} \alpha_k^3 = 1/5$
24			$\sum b_i \alpha_i \alpha_{ij} w_{jk} \alpha_k \alpha_{kl} \beta_l = 1/10$
25			$\sum b_i \alpha_i \alpha_{ij} w_{jk} \alpha_k \alpha_{kl} w_{lm} \alpha_m^2 = 1/5$
26			$\sum b_i \alpha_{ij} \beta_j \alpha_{ik} w_{kl} \alpha_l^2 = 1/10$
27			$\sum b_i \alpha_i^2 \alpha_{ij} w_{jk} \alpha_k^2 = 1/5$
28			$\sum b_i \beta_{ij} \alpha_j \alpha_{jk} w_{kl} \alpha_l^2 = 1/20$
29			$\sum b_i (\alpha_{ij} w_{jk} \alpha_k^2)^2 = 1/5$
30			$\sum b_i w_{ij} \alpha_j^4 = 1$
31			$\sum b_i w_{ij} \alpha_j^2 \alpha_{jk} \beta_k = 1/2$
32			$\sum b_i w_{ij} \alpha_j^2 \alpha_{jk} w_{kl} \alpha_l^2 = 1$
33			$\sum b_i w_{ij} \alpha_j \alpha_{jk} \beta_{kl} \beta_l = 1/6$
34			$\sum b_i w_{ij} \alpha_j \alpha_{jk} \alpha_k^2 = 1/3$
35			$\sum b_i w_{ij} \alpha_j \alpha_{jk} w_{kl} \alpha_l^3 = 1$
36			$\sum b_i w_{ij} \alpha_j \alpha_{jk} w_{kl} \alpha_l \alpha_{lm} \beta_m = 1/2$
37			$\sum b_i w_{ij} \alpha_j \alpha_{jk} w_{kl} \alpha_l \alpha_{lm} w_{mn} \alpha_n^2 = 1$
38			$\sum b_i w_{ij} (\alpha_{jk} \beta_k)^2 = 1/4$
39			$\sum b_i w_{ij} \alpha_{jk} \beta_k \alpha_{jm} w_{mn} \alpha_n^2 = 1/2$
40			$\sum b_i w_{ij} (\alpha_{jk} w_{kl} \alpha_l^2)^2 = 1$

Table 4 Additional order conditions

No		Order	Condition
41	Index-2	2	$\sum b_i w_{ij} w_{jk} \alpha_k^2 = 2$
42		3	$\sum b_i \alpha_i \alpha_{ij} w_{jk} w_{kl} \alpha_l^2 = 2/3$
43			$\sum b_i w_{ij} \alpha_j \alpha_{jk} w_{kl} w_{lm} \alpha_m^2 = 2$
44			$\sum b_i \alpha_{ij} w_{jk} w_{kl} \alpha_l^2 \alpha_{im} w_{mn} w_{nr} \alpha_r^2 = 4/3$
45	W method	2	$\sum b_i \gamma_{ij} = 0$
46			$\sum b_i \alpha_{ij} w_{jk} \alpha_k = 1/2$
47			$\sum b_i w_{ij} \alpha_j = 1$
48	Prot-Rob	3	$C_2(H) = \sum_{i=0}^{s} A_i H^i = 0$
49		4	$C_3(H) = \sum_{i=0}^{s-1} B_i H^i = 0$

with a non-singular matrix $(\frac{\partial g}{\partial y} \cdot \frac{\partial f}{\partial z})$ in the neighborhood of the solution, condition No. 41 guarantees convergence order $p = 2$. The additonal conditions No. 42–44 lead to order $p = 3$ for the differential variable y and $p = 2$ for the algebraic variable z of such index-2 problems.

Conditions No. 45–47 are introduced by Jax [5]. By these conditions at least order $p = 2$ is achieved for index-1 problems using inexact Jacobian matrices. Method `Rodas4P2` has been derived for that purpose, see [25]. When the Jacobian is computed by finite differences, this property might be advantageous.

Conditions No. 48, 49 are necessary for the Prothero-Robinson model, see equation (1.4). The coefficients of polynomials $C_2(H)$ and $C_3(H)$ with $H = \frac{z}{1-\gamma z}$ and $z = \lambda h$ are defined according to [23]

$$A_0 = -N^{(2)}(-1) + \gamma M(-1) + M(0) \tag{2.5}$$

$$A_i = -N^{(2)}(i-1) + 2\gamma M(i-1) + \gamma^2 M(i-2) + M(i) \tag{2.6}$$

$$\text{for } 0 < i < s$$

$$A_s = \gamma^2 M(s-2) \tag{2.7}$$

$$B_0 = -N^{(3)}(-1) + N^{(2)}(0) \tag{2.8}$$

$$B_i = -N^{(3)}(i-1) + \gamma N^{(2)}(i-1) + N^{(2)}(i) \tag{2.9}$$

$$\text{for } 0 < i < s-1$$

$$B_{s-1} = -N^{(3)}(s-2) + \gamma N^{(2)}(s-2) \tag{2.10}$$

$$M(\nu) = \sum_{i=1}^{s} b_i M_i(\nu), \quad N^{(\sigma)}(\nu) = \sum_{i=1}^{s} b_i N_i^{(\sigma)}(\nu) \quad \text{for } \sigma \geq 2 \tag{2.11}$$

with

$$M_i(\nu) = \begin{cases} 1 & \text{if } \nu < 0 \\ \beta'_i & \text{if } \nu = 0 \\ \sum \beta_{ij_1}\beta_{j_1j_2}\dots\beta_{j_{\nu-1}j_\nu}\beta'_{j_\nu} & \text{if } \nu = 1,\dots,i-2 \\ 0 & \text{if } \nu \geq i-1 \end{cases} \tag{2.12}$$

$$\sigma! N_i^{(\sigma)}(\nu) = \begin{cases} 1 & \text{if } \nu < 0 \\ \alpha_i^\sigma & \text{if } \nu = 0 \\ \sum \beta_{ij_1}\beta_{j_1j_2}\dots\beta_{j_{\nu-1}j_\nu}\alpha^\sigma_{j_\nu} & \text{if } \nu = 1,\dots,i-2 \\ 0 & \text{if } \nu \geq i-1 \end{cases} \tag{2.13}$$

and $\beta'_i = \sum_{j=1}^{i-1}\beta_{ij}$. The summation in (2.12), (2.13) is over $j_\nu < \dots < j_1 < i$. To distinguish summation $\sum_{j=1}^{i-1}\beta_{ij}$ and $\sum_{j=1}^{i}\beta_{ij}$ in the following, we introduce β'_{ij} with $\beta'_{ij} = \beta_{ij}$ for $j < i$ and $\beta'_{ij} = 0$ for $j \geq i$.

In order to fulfill conditions No. 48 and 49 in Table 4 all coefficients A_i and B_i must be zero. As stated in [25], the estimation of the error constant C of the global error in the paper of Scholz [23] is not sharp. It behaves like $C = \frac{1}{z}C_1$ for L-stable methods, see [17]. Therefore, for fixed h asymptotically exact results are obtained for $|\lambda| \to \infty$, but for fixed large stiffness $|\lambda|$ only order $p-1$ is obtained numerically. This can be seen in Table 1. Although the `Rodas4P` and `Rodas4P2` methods satisfy both conditions No. 48 and 49 they only show order $p = 3$ for the Prothero-Robinson model with large stiffness $|\lambda|$, whereas `Rodas4PR2` achieves the full order in stiff case.

An L-stable method is obtained, when $|R(z)| < 1$ for $\Re(z) < 0$ and $R(\infty) = 0$ holds. The stability function $R(z)$ can be expressed in terms of $M(\nu)$ defined in (2.11) as follows

$$R(z) = \sum_{i=0}^{s} M(i-2)H^i. \tag{2.14}$$

3 Construction of Rodas5P

The aim is to construct a method which fullfills all order conditions stated in Tables 2, 3 and 4. Analogously to [3], we choose $s = 8$ and want to construct a stiffly accurate method with

$$b_i = \beta_{8i} \quad \text{for } i = 1,\dots,7, \quad b_8 = \gamma, \quad \alpha_8 = 1. \tag{3.1}$$

The embedded method with stage number $\hat{s} = 7$ is stiffly accurate, too:

$$\hat{b}_i = \beta_{7i} \quad \text{for } i = 1,\dots,6, \quad \hat{b}_7 = \gamma, \quad \alpha_7 = 1. \tag{3.2}$$

It should fullfill the order conditions No. 1-8, 18-22, 41, 48 leading to a method of order $\hat{p} = 4$ for index-1 DAEs. These conditions are denoted by $\hat{1}$, $\hat{2}$,...,$\hat{48}$.

According to [3, 4] we require

$$\alpha_{8i} = \beta_{7i} \quad \text{for } i = 1, \ldots, 7; \quad \alpha_{7i} = \beta_{6i} \quad \text{for } i{=}1, \ldots, 6; \quad \alpha_6{=}1. \tag{3.3}$$

Therefore, the following 40 coefficients remain to be determined:

$$\gamma, \beta_{21}, \beta_{31}, \beta_{32}, ..., \beta_{54}, \beta_{62}, ..., \beta_{65}, \beta_{72}, ..., \beta_{76}, \beta_{82}, ..., \beta_{87},$$
$$\alpha_{21}, \alpha_{31}, \alpha_{32}, \alpha_{41}, \alpha_{42}, \alpha_{43}, \alpha_{51}, \alpha_{52}, \alpha_{53}, \alpha_{54}, \alpha_{62}, \alpha_{63}, \alpha_{64}, \alpha_{65}.$$

Coefficients $\beta_{61}, \beta_{71}, \beta_{81}, \alpha_{61}$ are not listed, since they are determined later by $\alpha_6 = 1$ and $\beta'_8 = \beta'_7 = \beta'_6 = 1 - \gamma$, which follows from condition No.1 and the choice $\alpha_7 = \alpha_8 = 1$.

Our strategy is to fulfill the conditions No. 48, 49 first. In these conditions occur terms belonging to the long trees (see conditions No. 1, 2, 4, 8, 17) and the trees belonging to conditions No. 3, 7, 16. Moreover, we try to fullfill at least some of the conditions to arrive at $C_4(H) = 0$ where terms belonging to trees of conditions No. 5, 14 occur.

Moreover, we can simplify many conditions related to long trees. To give an example we reformulate conditions No. 2, 4:

$$\begin{aligned}
\sum b_i \beta_i &= \frac{1}{2} \quad \Leftrightarrow \\
\sum \beta'_{8i} \beta'_i + \gamma^2 + 2\gamma\beta'_8 &= \frac{1}{2} \quad \Leftrightarrow \\
\sum \beta'_{8i} \beta'_i &= \frac{1}{2} - \gamma^2 - 2\gamma(1-\gamma) = \frac{1}{2} - 2\gamma + \gamma^2 \\
\sum b_i \beta_{ij} \beta_j &= \frac{1}{6} \quad \Leftrightarrow \\
\sum \beta'_{8i} \beta'_{ij} \beta'_j + \gamma^3 + 3\gamma^2\beta'_8 + 3\gamma \sum \beta'_{8i} \beta'_i &= \frac{1}{6} \quad \Leftrightarrow \\
\sum \beta'_{8i} \beta'_{ij} \beta'_j &= \frac{1}{6} - 3\gamma \left(\frac{1}{2} - 2\gamma + \gamma^2 \right) - 3\gamma^2(1-\gamma) - \gamma^3 \\
&= \frac{1}{6} - \frac{3}{2}\gamma + 3\gamma^2 - \gamma^3
\end{aligned}$$

1. In the first step we set $\alpha_{21} = 3\gamma$ and $\beta_{21} = 0$, see [24]. By this choice the following conditions are fullfilled: No. 18, $\hat{18}$, 20, $\hat{20}$, 30, 47. Form No. 48, 49 we obtain $A_8 = 0$, $\hat{A}_7 = 0$, $B_7 = 0$.
2. Now we interpret γ, α_3, α_4, α_5, α_{52}, α_{65} and β'_5 as free parameters and try to compute the remaining ones dependent on these. We set $\beta_{32} = (\frac{\alpha_3}{\alpha_2})^2(\frac{\alpha_3}{3} - \gamma)$ and $\beta'_3 = \frac{9}{2}\beta_{32}$ and get $A_7 = 0$, $\hat{A}_6 = 0$, $B_6 = 0$ from No. 48, 49.
3. We solve the linear system

$$\begin{pmatrix} \frac{1}{2}\alpha_2^2 & \frac{1}{2}\alpha_3^2 - 2\gamma\beta'_3 & -\gamma^2 \\ \alpha_2^2 & \alpha_3^2 & 0 \\ \alpha_2^3 & \alpha_3^3 & 0 \end{pmatrix} \begin{pmatrix} \beta_{42} \\ \beta_{43} \\ \beta'_4 \end{pmatrix} = \begin{pmatrix} 0 \\ \alpha_4^2 \left(\frac{\alpha_4}{3} - \gamma\right) \\ \alpha_4^3 \left(\frac{\alpha_4}{4} - \gamma\right) \end{pmatrix}$$

which yields $A_6 = 0$, $\hat{A}_5 = 0$, $B_5 = 0$.

4. In the next step we get $A_5 = 0$, $\hat{A}_4 = 0$, $B_4 = 0$ from the linear system

$$\begin{pmatrix} \frac{1}{2}\alpha_2^2 & \frac{1}{2}\alpha_3^2 - 2\gamma\beta_3' & \frac{1}{2}\alpha_4^2 - 2\gamma\beta_4' - \beta_{43}\beta_3' \\ \alpha_2^2 & \alpha_3^2 & \alpha_4^2 \\ \alpha_2^3 & \alpha_3^3 & \alpha_4^3 \end{pmatrix} \begin{pmatrix} \beta_{52} \\ \beta_{53} \\ \beta_{54} \end{pmatrix} = \begin{pmatrix} \gamma^2\beta_5' \\ \alpha_5^2\left(\frac{\alpha_5}{3} - \gamma\right) \\ \alpha_5^3\left(\frac{\alpha_5}{4} - \gamma\right) \end{pmatrix}$$

5. Solving

$$\begin{pmatrix} \alpha_2^2 & \alpha_3^2 & \alpha_4^2 & \alpha_5^2 \\ \alpha_2^3 & \alpha_3^3 & \alpha_4^3 & \alpha_5^3 \\ \beta_2' & \beta_3' & \beta_4' & \beta_5' \\ 0 & 0 & \beta_{43}\beta_3' & \beta_{53}\beta_3' + \beta_{54}\beta_4' \end{pmatrix} \begin{pmatrix} \beta_{62} \\ \beta_{63} \\ \beta_{64} \\ \beta_{65} \end{pmatrix} = \begin{pmatrix} \frac{1}{3} - \gamma \\ \frac{1}{4} - \gamma \\ \frac{1}{2} - 2\gamma + \gamma^2 \\ \frac{1}{6} - \frac{3}{2}\gamma + 3\gamma^2 - \gamma^3 \end{pmatrix}$$

gives $A_4 = 0$, $\hat{A}_3 = 0$, $B_3 = 0$.

6. In order to get $A_3 = 0$, $\hat{A}_2 = 0$, $B_2 = 0$ we now solve the underdetermined linear system of equations

$$\begin{pmatrix} \beta_2' & \beta_3' & \beta_4' & \beta_5' & \beta_6' & \beta_7' \\ \alpha_2^2 & \alpha_3^2 & \alpha_4^2 & \alpha_5^2 & 1 & 1 \\ 0 & 0 & \beta_{43}\beta_3' & \beta_{53}\beta_3' + \beta_{54}\beta_4' & \frac{1}{2} - 2\gamma\beta_6' - \gamma^2 & \\ 0 & 0 & 0 & \beta_{54}\beta_{43}\beta_3' & \frac{1}{6} - \frac{3}{2}\gamma + 3\gamma^2 - \gamma^3 & \end{pmatrix} \begin{pmatrix} \beta_{72} \\ \beta_{73} \\ \beta_{74} \\ \beta_{75} \\ \beta_{76} \end{pmatrix}$$
$$= \begin{pmatrix} \frac{1}{2} - 2\gamma + \gamma^2 \\ \frac{1}{3} - \gamma \\ \frac{1}{6} - \frac{3}{2}\gamma + 3\gamma^2 - \gamma^3 \\ \frac{1}{24} - \frac{2}{3}\gamma + 3\gamma^2 - 4\gamma^3 + \gamma^4 \end{pmatrix}$$

The obtained degree of freedom will be used for fullfilling remaining order conditions in the iteration process later on.

7. Now we can finish the computation of the β-coefficients by solving
ss

$$\begin{pmatrix} \beta_2' & \beta_3' & \beta_4' & \beta_5' & \beta_6' & \\ \alpha_2^2 & \alpha_3^2 & \alpha_4^2 & \alpha_5^2 & 1 & \\ 0 & 0 & \beta_{43}\beta_3' & \beta_{53}\beta_3' + \beta_{54}\beta_4' & \frac{1}{2} - 2\gamma\beta_6' - \gamma^2 & \frac{1}{2} - 2\gamma + \gamma^2 \\ 0 & 0 & 0 & \beta_{54}\beta_{43}\beta_3' & \frac{1}{6} - \frac{3}{2}\gamma + 3\gamma^2 - \gamma^3 & \frac{1}{6} - \frac{3}{2}\gamma + 3\gamma^2 - \gamma^3 \\ 0 & 0 & 0 & 0 & \beta_{65}\beta_{54}\beta_{43}\beta_3' & \frac{1}{24} - \frac{2}{3}\gamma + 3\gamma^2 - 4\gamma^3 + \gamma^4 \\ 0 & 0 & 0 & 0 & 0 & \beta_{76}\beta_{65}\beta_{54}\beta_{43}\beta_3' \end{pmatrix} \begin{pmatrix} \beta_{82} \\ \beta_{83} \\ \beta_{84} \\ \beta_{85} \\ \beta_{86} \\ \beta_{87} \end{pmatrix}$$

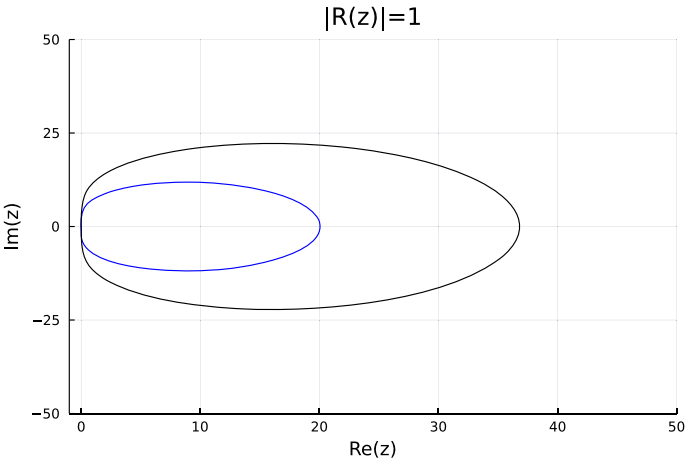

Fig. 2 Values $z \in \mathbb{C}$ with stabilty function $|R(z)| = 1$, black for `Rodas5P`, blue for its embedded method (colour figure online)

$$= \begin{pmatrix} \frac{1}{2} - 2\gamma + \gamma^2 \\ \frac{1}{3} - \gamma \\ \frac{1}{6} - \frac{3}{2}\gamma + 3\gamma^2 - \gamma^3 \\ \frac{1}{24} - \frac{2}{3}\gamma + 3\gamma^2 - 4\gamma^3 + \gamma^4 \\ \frac{1}{120} - \frac{5}{24}\gamma + \frac{5}{3}\gamma^2 - 5\gamma^3 + 5\gamma^4 - \gamma^5 \\ \frac{1}{720} - \frac{1}{20}\gamma + \frac{5}{8}\gamma^2 - \frac{10}{3}\gamma^3 + \frac{15}{2}\gamma^4 - 6\gamma^5 + \gamma^6 \end{pmatrix}$$

After that the following conditions remain to be fullfilled: No. 6, $\hat{6}$, 9, 10, 11, 12, 13, 15, 19, $\hat{19}$, 23, 24, 25, 26, 27, 28, 29, 42, 44, 45, 46.

8. The α_{ij}-coefficients occur linearly in equations No. 6, $\hat{6}$, 10, 12, 19, $\hat{19}$, 23, 28. From these and from the free parameters α_3, α_4, α_5, α_{52}, α_{65} we can compute all α-coefficients. Since conditions No. 13, 27, 42 are automatically fulfilled, too, the remaining conditions read No. 9, 11, 15, 24, 25, 26, 29, 44, 45, 46.
9. For these remaining 10 conditions 7 degrees of freedom are left. We can obtain an exact solution by formulating a nonlinear least-square problem and solving it by the optimization package `Otim.jl` using the Nelder-Mead algorithm. Why this is possible and whether there are structural reasons for it could not be definitely clarified.

The stability region of `Rodas5P` is shown in Fig. 2. It is an A-stable method and due to the stiffly accurate property it is L-stable, too.

Next we derive a dense output formula. According to [4] we compute intermediate values of the numerical solution by replacing equation (1.3) with

$$y_1(\tau) = y_0 + \sum_{i=1}^{s} b_i(\tau) k_i \quad \text{for } \tau \in [0, 1]. \tag{3.4}$$

The coefficients $b_i(\tau)$ are polynomials of degree 4 and should fullfill $b_i(0) = 0$, $b_i(1) = b_i$. Therefore, we set

$$b_i(\tau) = \tau b_i + \tau(\tau - 1)\left(c_i + \tau d_i + \tau^2 e_i\right) \tag{3.5}$$

$$= \tau(b_i - c_i) + \tau^2(c_i - d_i) + \tau^3(d_i - e_i) + \tau^4 e_i \tag{3.6}$$

In order to get a fourth order interpolation conditions No. 1–8 and 18–22 must be fullfilled for the weights $b_i(\tau)$. Note that the right hand side of the conditions must be multiplied with τ^n, where n is the number of the solid (=black) nodes of the corresponding tree, see [4]. For example, condition No. 21 now reads

$$\sum b_i(\tau) w_{ij} \alpha_j \alpha_{jk} \beta_k = \frac{1}{2}\tau^3 .$$

This condition can be fullfilled by

$$\begin{aligned}
\sum b_i w_{ij} \alpha_j \alpha_{jk} \beta_k &= \frac{1}{2} \\
\sum c_i w_{ij} \alpha_j \alpha_{jk} \beta_k &= \frac{1}{2} \\
\sum d_i w_{ij} \alpha_j \alpha_{jk} \beta_k &= \frac{1}{2} \\
\sum e_i w_{ij} \alpha_j \alpha_{jk} \beta_k &= 0,
\end{aligned}$$

where the first equation for coefficients b_i is already true. Thus we have $3 \cdot 13 = 39$ linear equations to be satisfied by $3 \cdot s = 24$ coefficients. Nevertheless, the solution is possible for the new `Rodas5P` as well for the known `Rodas5` method.

`Rodas5P` and the new dense output formula for `Rodas5` are implemented in the Github repository of the Julia `DifferentialEquations` package, see https://github.com/SciML/OrdinaryDiffEq.jl. All coefficients of the methods can be found there in particular.

4 Numerical benchmarks

First we show that the orders given in Table 1 are attained. We solve test problems with known analytical solution $y^{ana}(t)$ by each solver with different numbers of constant stepsizes and compute the numerical errors and orders of convergence. The error is given in the maximum norm at final time:

$$err = \max_i |y_i^{num}(t_{end}) - y_i^{ana}(t_{end})|. \tag{4.1}$$

The order p is computed by $p = \log_2(err_{2h}/err_h)$, where err_h denotes the error obtained with stepsize h. The following test problems have been treated:

Table 5 Numerical results for problem 1 (Index-1 DAE)

Stepsize	ROS3P	Ros3prl2	Rodas3	Rodas4	Rodas4PR2	Rodas4P2	Rodas5	Rodas5P
0.125	1.09e−05	4.78e−05	2.98e−06	3.34e−07	1.01e−06	2.20e−07	8.71e−09	2.93e−08
0.0625	1.41e−06	5.86e−06	3.80e−07	1.95e−08	5.99e−08	1.29e−08	2.41e−10	8.56e−10
0.03125	1.78e−07	7.24e−07	4.83e−08	1.18e−09	3.65e−09	7.81e−10	7.08e−12	2.59e−11
0.015625	2.23e−08	8.99e−08	6.11e−09	7.23e−11	2.26e−10	4.82e−11	2.16e−13	8.01e−13
0.0625	3.0	3.0	3.0	4.1	4.1	4.1	5.2	5.1
0.03125	3.0	3.0	3.0	4.1	4.0	4.0	5.1	5.0
0.015625	3.0	3.0	3.0	4.0	4.0	4.0	5.0	5.0
0.125	4.84e−04	1.12e−05	2.35e−04	5.67e−06	3.09e−05	4.93e−06	6.28e−08	1.13e−06
0.0625	1.21e−04	3.37e−06	6.00e−05	7.64e−07	3.54e−06	5.40e−07	3.72e−09	6.60e−08
0.03125	3.04e−05	1.07e−06	1.51e−05	9.88e−08	4.22e−07	6.26e−08	2.22e−10	4.00e−09
0.015625	7.62e−06	3.17e−07	3.80e−06	1.26e−08	5.14e−08	7.52e−09	1.35e−11	2.46e−10
0.0625	2.0	1.7	2.0	2.9	3.1	3.2	4.1	4.1
0.03125	2.0	1.7	2.0	3.0	3.1	3.1	4.1	4.0
0.015625	2.0	1.8	2.0	3.0	3.0	3.1	4.0	4.0

The error at final time and the attained order of convergence are given for different stepsizes. The results for the embedded methods are shown in the lower part of the Table

1. Index-1 DAE

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}' = \begin{pmatrix} \frac{y_2}{y_1} \\ \frac{y_1}{y_2} - t \end{pmatrix}, \quad \begin{pmatrix} y_1(2) \\ y_2(2) \end{pmatrix} = \begin{pmatrix} \ln(2) \\ \frac{1}{2}\ln(2) \end{pmatrix}, \quad t \in [2, 4]$$

with solution $y_1(t) = \ln(t)$, $y_2(t) = \frac{1}{t}\ln(t)$. The theoretical orders of convergence are achieved by all methods, see Table 5.

2. Prothero-Robinson model

$$y' = -\lambda(y - g(t)) + g'(t), \quad g(t) = 10 - (10 + t)e^{-t}, \quad \lambda = 10^5, \quad t \in [0, 2]$$

with solution $y(t) = g(t)$, see [23, 25]. The results are shown in Table 6 and agree with those shown in Table 1. The new method `Rodas5P` behaves like `Rodas4P2` as expected. Computations with different stiffness parameters in the range $\lambda \in [10^0, 10^5]$ show, that only for the method `Ros3prl2` the convergence is independent on λ. This includes also mildly stiff problems, where `Rodas4PR2` shows order reduction to $p = 3$.

3. Parabolic problem

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + u^2 + h(x, t), \quad x \in [-1, 1], \quad t \in [0, 1] \tag{4.2}$$

This problem is a slight modification of a similar problem treated in [2]. Function $h(x, t)$ is chosen in order to get the solution $u(x, t) = x^3 \cdot e^t$. The initial values and Dirichlet boundary condition are taken from this solution. Since $u(x, t)$ is cubic in x, the discretization $\frac{\partial^2}{\partial x^2} u(x_i, t) = \frac{u(x_{i-1},t) - 2u(x_i,t) + u(x_{i+1},t)}{\Delta x^2}$ is exact. The numerical results for $n_x = 1000$ space discretization points are given in Table 7. The methods do not achieve the full theoretical order for parabolic problems, shown in Tabel 1. The reason is, that the theory given in [14] assumes linear problems and vanishing boundary conditions. Similar computations with a linear parabolic problem resulted in the full theoretical order. We see further that the embedded method of `Rodas5P` has nearly order $p = 4$, too. Nevertheless, the results of the embedded method are slightly worse so that the stepsize control is expected to work. Additionally the results of `Rodas5P` are compared to the approach chosen in [2]. As proposed there, method `GRK4T` is applied to problem (4.2). `GRK4T` [6] is a 4-stage ROW method of order $p = 4$ for ordinary differential equations. Due to a special choice of its coefficients, it needs only three function evaluations of the righthand side of the ODE system per timestep. Usually, an order reduction to $p = 2$ would occur when applying it to semi-discretized parabolic problems. This order reduction is prevented by modifications of the boundary conditions in each stage. Technical details can be found in [2]. For comparison, `Rodas4` was modified accordingly in addition to `GRK4T`. Figure 3 shows the results of `Rodas5P` and the modified methods. For different time stepsizes resulting in different number of function evaluations, the error according to equation (4.1) is plotted. Due to the automatic differentiation for the computation of the Jacobian and the time derivative, only one

additional function call was considered respectively. Additonally, the methods were applied with adaptive stepsizes for different tolerances and the error versus elapsed CPU time is shown. While `Rodas5P` undergoes the small order reduction shown in Table 7, modified `GRK4T` and `Rodas4` have exactly order $p = 4$. Nevertheless, `Rodas5P` is more efficient because it has a lower error constant.

4. Index-2 DAE

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}' = \begin{pmatrix} y_2 \\ y_1^2 - \frac{1}{t^2} \end{pmatrix}, \quad \begin{pmatrix} y_1(1) \\ y_2(1) \end{pmatrix} = \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \quad t \in [1, 2]$$

with solution $y_1(t) = -\frac{1}{t}$, $y_2(t) = \frac{1}{t^2}$. Methods `Rodas3`, `Rodas4`, `Rodas5` show order reduction to $p = 1$. All other methods achieve order $p = 2$, see Table 8.

5. Inexact Jacobian

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}' = \begin{pmatrix} y_2 \\ y_1^2 + y_2^2 - 1 \end{pmatrix}, \quad \begin{pmatrix} y_1(0) \\ y_2(0) \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad t \in [0, 1]$$

with solution $y_1(t) = \sin(t)$, $y_2(t) = \cos(t)$. Instead of the exact Jacobian we apply $J = \begin{pmatrix} 0 & 0 \\ 0 & 2y_2 \end{pmatrix}$. According to Jax [5] the derivative of the algebraic equation with respect to the algebraic variable must be exact. We observe the orders shown in Table 1, `Rodas5P` behaves like `Rodas4P2`.

6. Dense output We check the dense output formulae of the fourth and fifth order methods via the problem

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}' = \begin{pmatrix} n \cdot t^{n-1} \\ y_1 - y_2 \end{pmatrix}, \quad \begin{pmatrix} y_1(0) \\ y_2(0) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad t \in [0, 2]$$

with solution $y_1(t) = y_2(t) = t^n$. A method of order $p \geq n$ should solve this problem exactly within one timestep of size $h = 2$. After the solution with one timestep we apply the dense output formula to interpolate the solution to times $t_i = i \cdot h$, $i = 1, ..., k$, $h = \frac{2}{2^k}$, $k = 1, 2, 3$ and compute the resulting maximum error at these timesteps. The numerical errors for different polynomial degrees n of the solution are given in Table 9. Here we can see that the fourth order methods are equipped with dense output formulae of order $p = 3$, `Rodas5` and `Rodas5P` are able to interpolate with order $p = 4$.

Next we look at work-precision diagrams and compare the fourth and fifth order methods.

In these investigations and in Table 9 `Rodas4PR2` was replaced by `Rodas4P` since no dense output formula is available for `Rodas4PR2`. The work-precision diagrams are computed for eight different problems by the function `WorkPrecisionSet` form the Julia package `DiffEqDevTools.jl` which is part of `DifferentialEquations.jl`. For different tolerances the corresponding computation times and achieved accuracies are evaluated. We show graphs for two different errors: The l_2-error is taken from the solution at every timestep and the L_2-error is taken at 100 evenly

Table 6 Numerical results (error and order) for problem 2 (Prothero-Robinson model)

Stepsize	ROS3P	Ros3prl2	Rodas3	Rodas4	Rodas4PR2	Rodas4P2	Rodas5	Rodas5P
0.25	3.91e−08	2.34e−09	1.45e−06	1.79e−09	5.47e−11	1.21e−09	1.84e−08	1.26e−09
0.125	1.77e−08	2.81e−10	6.41e−07	1.85e−08	3.25e−12	1.47e−10	7.46e−09	1.47e−10
0.0625	4.59e−09	3.45e−11	3.00e−07	1.35e−08	1.97e−13	1.80e−11	3.20e−09	1.78e−11
0.03125	1.15e−09	4.28e−12	1.45e−07	7.69e−09	1.07e−14	2.24e−12	1.46e−09	2.17e−12
0.125	1.1	3.1	1.2	−3.4	4.1	3.0	1.3	3.1
0.0625	1.9	3.0	1.1	0.5	4.0	3.0	1.2	3.0
0.03125	2.0	3.0	1.0	0.8	4.2	3.0	1.1	3.0
0.25	5.57e−03	5.16e−03	1.42e−07	5.07e−07	2.73e−09	3.76e−09	1.58e−07	4.66e−09
0.125	2.54e−03	1.20e−03	3.34e−08	2.12e−07	3.24e−10	4.45e−10	5.61e−08	5.47e−10
0.0625	6.54e−04	2.89e−04	8.01e−09	9.63e−08	3.95e−11	5.42e−11	2.30e−08	6.63e−11
0.03125	1.62e−04	7.09e−05	1.87e−09	4.58e−08	4.87e−12	6.68e−12	1.03e−08	8.16e−12
0.125	1.1	2.1	2.1	1.3	3.1	3.1	1.5	3.1
0.0625	2.0	2.1	2.1	1.1	3.0	3.0	1.3	3.0
0.03125	2.0	2.0	2.1	1.1	3.0	3.0	1.2	3.0

The results for the embedded methods are shown in the lower part of the Table

Table 7 Numerical results (error and order) for problem 3 (parabolic model)

Stepsize	ROS3P	Ros3prl2	Rodas3	Rodas4	Rodas4PR2	Rodas4P2	Rodas5	Rodas5P
0.03125	2.33e−06	1.96e−06	3.35e−05	8.86e−07	6.90e−08	8.72e−09	2.15e−07	5.97e−09
0.015625	3.88e−07	1.87e−07	8.92e−06	2.01e−07	6.12e−09	8.21e−10	5.97e−08	4.72e−10
0.0078125	6.30e−08	1.76e−08	2.30e−06	4.76e−08	5.03e−10	6.95e−11	1.57e−08	3.45e−11
0.00390625	9.52e−09	1.70e−09	5.85e−07	1.16e−08	3.85e−11	5.43e−12	4.01e−09	2.36e−12
0.015625	2.6	3.4	1.9	2.1	3.5	3.4	1.9	3.7
0.0078125	2.6	3.4	2.0	2.1	3.6	3.6	1.9	3.8
0.00390625	2.7	3.4	2.0	2.0	3.7	3.7	2.0	3.9
0.03125	3.20e−04	1.35e−04	9.03e−05	7.36e−06	1.16e−06	2.36e−07	5.40e−07	8.16e−08
0.015625	8.05e−05	3.38e−05	3.03e−05	2.00e−06	1.20e−07	2.92e−08	1.50e−07	6.52e−09
0.0078125	2.02e−05	8.43e−06	9.54e−06	5.22e−07	1.26e−08	3.61e−09	3.95e−08	4.91e−10
0.00390625	5.17e−06	2.09e−06	2.86e−06	1.33e−07	1.37e−09	4.48e−10	1.01e−08	3.54e−11
0.015625	2.0	2.0	1.6	1.9	3.3	3.0	1.8	3.6
0.0078125	2.0	2.0	1.7	1.9	3.3	3.0	1.9	3.7
0.00390625	2.0	2.0	1.7	2.0	3.2	3.0	2.0	3.8

The results for the embedded methods are shown in the lower part of the Table

spaced points via interpolation. Thus the latter should reflect the error of the dense output formulae. The reference solutions of the problems are computed by `Rodas4P2` with tolerances `reltol=abstol=`10^{-14}.

1. Parabolic problem
   We treat again the problem shown in equations (4.2) and Table 7. It turns out that the new method and `Rodas4P2` show the best behavior, see Fig. 4. The order reduction of `Rodas4` and `Rodas5` is clearly visible at the investigated accuracies.
2. Hyperbolic problem
   This problem is discussed in [22, 25]. A hyperbolic PDE is discretized by 250 space points. Since the true solution is linear in space variable x, the approximation of the space derivative by first-order finite differences is exact. In Fig. 4 we can see the improved dense output of `Rodas5`. While the results of `Rodas4` and `Rodas5` with respect to the l_2-errors are very similar `Rodas5` is much better with respect to the L_2-errors. In both cases the new method `Rodas5P` achieves the best numerical results.
3. Plane pendulum
   The pendulum of mass $m = 1$ and length L can be modeled in cartesian coordinates $x(t)$, $y(t)$ by the equations

$$\begin{aligned} \ddot{x} &= \lambda x, \\ \ddot{y} &= \lambda y - g, \\ 0 &= x^2 + y^2 - L^2, \end{aligned}$$

   with Lagrange multiplier $\lambda(t)$ and gravitational constant g. This system is an index-3 DAE which cannot be solved by methods discussed above. By derivation of the algebraic equation with respect to time we achieve the index-2 and index-1 formulation:

$$\begin{aligned} 0 &= x\dot{x} + y\dot{y} \quad \text{(index-2)}, \\ 0 &= \dot{x}^2 + \lambda x^2 + \dot{y}^2 + \lambda y^2 - yg \quad \text{(index-1)}. \end{aligned}$$

   We solve this system in index-1 and index-2 formulation with initial conditions $x(0) = 2$, $\dot{x}(0) = y(0) = \dot{y}(0) = \lambda(0) = 0$ in the time intervall $t \in [0, 10]$. The numerical results shown in Fig. 5 turn out as expected. For the index-1 problem the fifth order methods `Rodas5` and `Rodas5P` yield the best and very similar results. For the index-2 problem `Rodas4` and `Rodas5` show the largest order reduction.
4. Transistor amplifier
   The two-transistor amplifier was intruduced in [19] and further discussed in [12, 13]. It consists of eight equations of type (1.1) with index-1. The work-precision diagram shown in Fig. 6 indicates similar behavior for all methods in the l_2-error. Regarding the L_2-error `Rodas5` and `Rodas5P` perform best and the improved dense output of `Rodas5` is obvious. The new method `Rodas5P` cannot beat `Rodas5` in this case.
5. Water tube problem This example treats the flow of water through 18 tubes which are connected via 13 nodes, see [12, 13]. The 49 unknowns of the system are

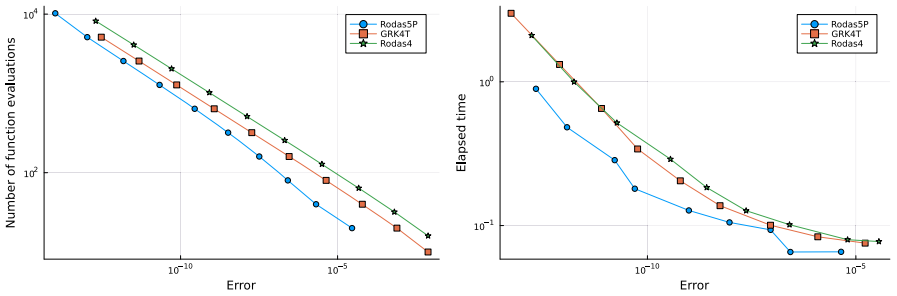

Fig. 3 Comparison of results of Rodas5P and modified methods GRK4T and Rodas4 for parabolic problem. Computations left with constant and right with adaptive stepsizes

the pressure in the nodes, the volume flow and the resistance coefficients of the edges. The equations for the volume flow and for the pressure of two nodes which have a storage function are ordinary differential equations. The equations for the resistance coefficients are of index-1, in the original formulation the equations for the pressure are of index 2. We adapted these equations in order to get a DAE system of index-1. The corresponding results are shown in Fig. 6. It turns out that `Rodas5P` is slightly more efficient than `Rodas5`.

6. Pollution
   This is a standard test problem for stiff solvers and contains 20 equations for the chemical reaction part of an air pollution model, see [12, 13, 27]. The problem is already part of the Julia package `SciMLBenchmarks.jl`. The results shown in Fig. 7 indicate again that the fifth order methods are preferable.
7. Photovoltaic network
   The new method shall be used in network simulation, see [26]. Therefore we finally simulate a small electric network consisting of a photovoltaic (PV) element, a battery and a consumer with currents $i_{PV}(t), i_B(t), i_C(t)$. All elements are connected in parallel between two node potentials $U_0(t)$, $U_1(t)$. The first node is grounded, at the second node the sum of currents equals zero. The battery is characterized further by its charge $q_B(t)$ and an internal voltage $u_B(t)$. These seven states are described by equations

$$
\begin{aligned}
0 &= U_0 \\
0 &= i_B + i_{PV} - i_C \\
0 &= P(t) - i_C\,(U_1 - U_0) \\
0 &= c_1 + c_2\, i_{PV} + c_3(U_1 - U_0) + c_4(\exp(c_5\, i_{PV} + c_6(U_1 - U_0)) - 1) \\
0 &= U_1 - U_0 - (u_0(q_B) - u_B - R_0\, i_B) \\
u_B' &= \frac{1}{C} i_B - \frac{1}{R_1 C} u_B \\
q_B' &= -i_B
\end{aligned}
$$

The third equation describes the consumer that demands a power P(t). It is assumed that $P(t)$ represents a constant power, but it is switched on or off every hour. The

Table 8 Numerical results (error and order) for problem 4 (Index-2 DAE)

Stepsize	ROS3P	Ros3prl2	Rodas3	Rodas4	Rodas4PR2	Rodas4P2	Rodas5	Rodas5P
0.03125	2.73e−05	1.72e−04	2.70e−03	5.92e−05	8.75e−05	3.26e−05	2.23e−05	9.00e−05
0.015625	5.63e−06	4.20e−05	1.33e−03	5.53e−05	2.25e−05	8.05e−06	1.15e−05	2.33e−05
0.0078125	1.37e−06	1.04e−05	6.57e−04	3.39e−05	5.69e−06	2.00e−06	5.88e−06	5.94e−06
0.015625	2.3	2.0	1.0	0.1	2.0	2.0	1.0	1.9
0.0078125	2.0	2.0	1.0	0.7	2.0	2.0	1.0	2.0
0.03125	4.02e−04	1.59e−03	1.94e−04	8.10e−04	3.30e−04	7.58e−05	1.52e−04	1.49e−04
0.015625	1.67e−04	7.27e−04	4.71e−05	4.04e−04	8.05e−05	1.91e−05	8.14e−05	3.58e−05
0.0078125	8.22e−05	3.47e−04	1.16e−05	2.02e−04	1.99e−05	4.78e−06	4.19e−05	8.76e−06
0.015625	1.3	1.1	2.0	1.0	2.0	2.0	0.9	2.1
0.0078125	1.0	1.1	2.0	1.0	2.0	2.0	1.0	2.0

The results for the embedded methods are shown in the lower part of the Table

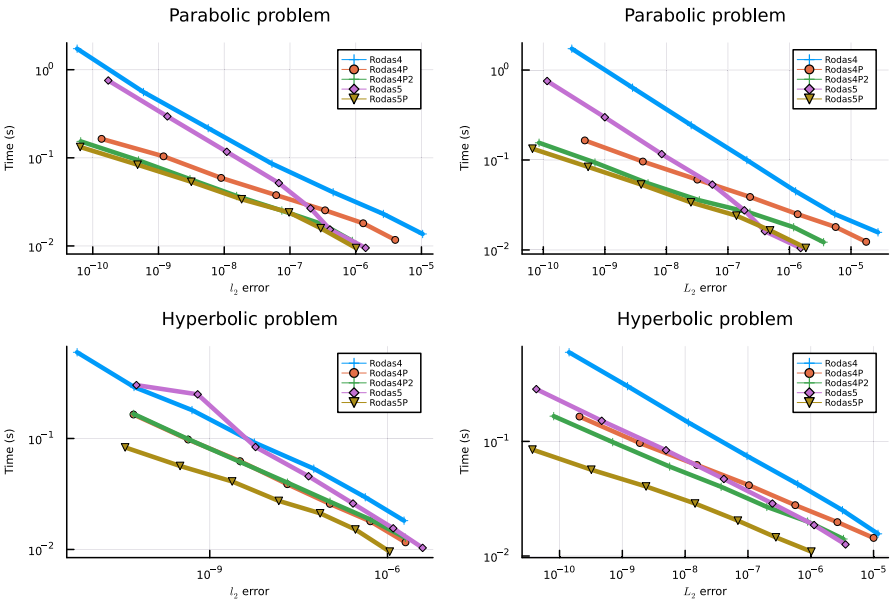

Fig. 4 Work-precision diagrams for parabolic and hyperbolic problems

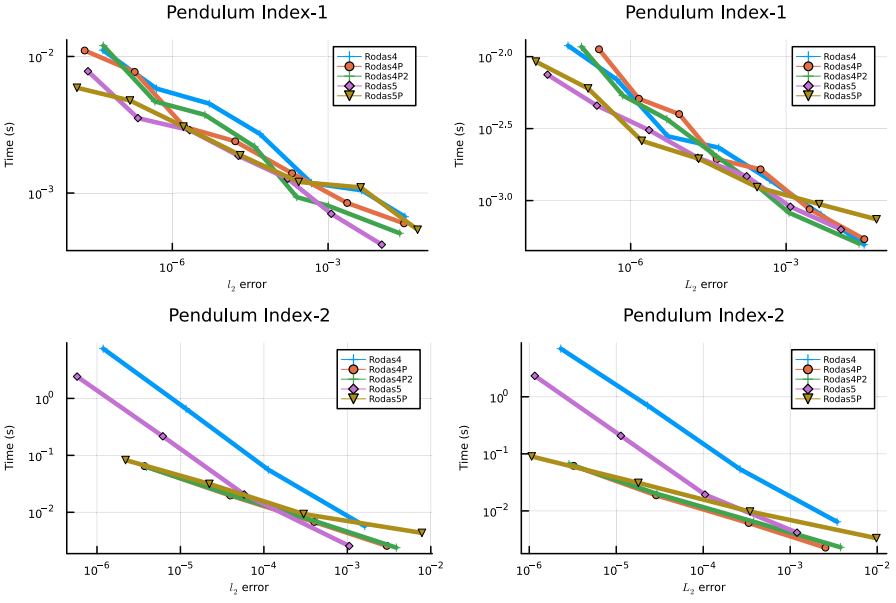

Fig. 5 Work-precision diagrams for the pendulum problem in index-1 and index-2 formulation

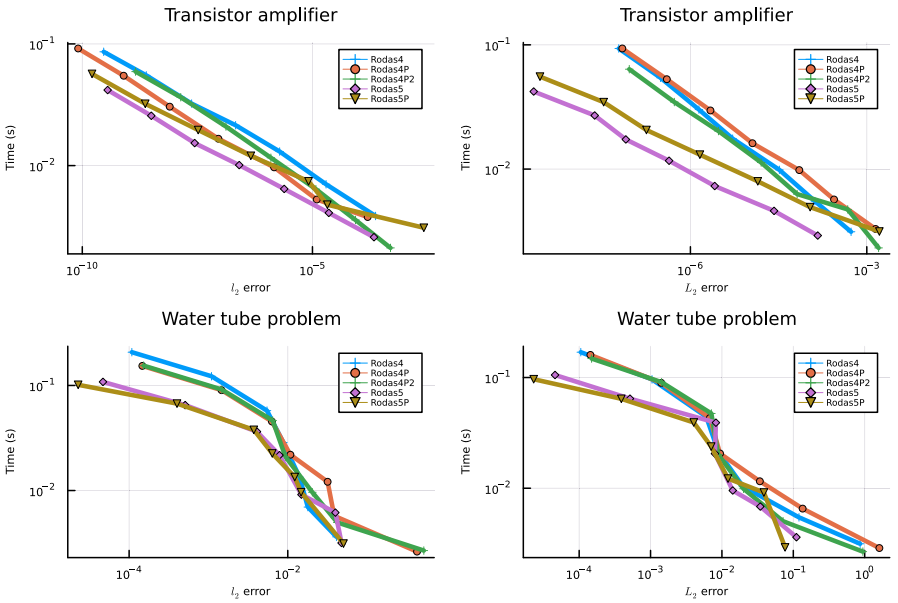

Fig. 6 Work-precision diagrams for two transistor amplifier and water tube problem

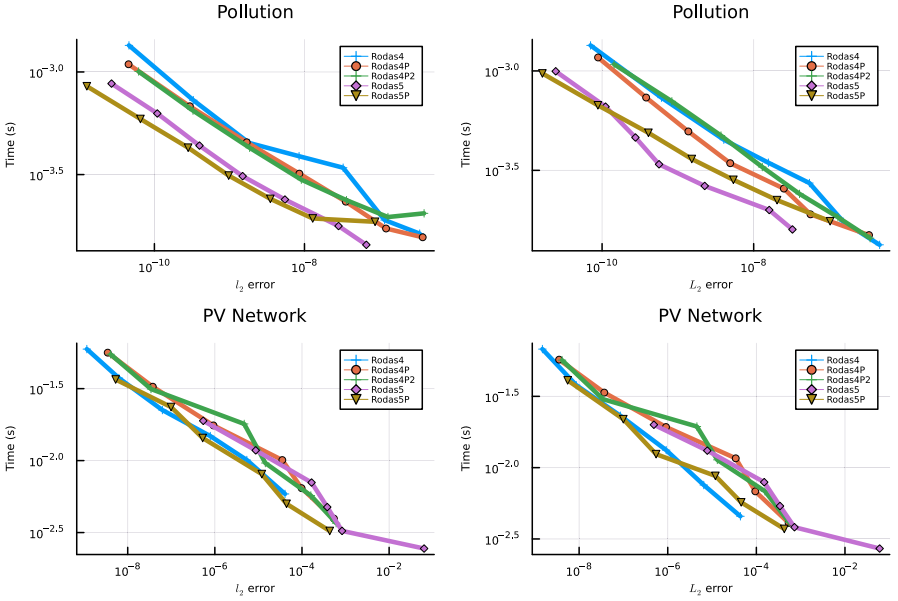

Fig. 7 Work-precision diagrams for pollution problem and photovoltaic network

Table 9 Numerical results (error) for dense output formulae

Solution	Stepsize	Rodas4	Rodas4P	Rodas4P2	Rodas5	Rodas5P
t^3	2.0	1.78e−14	7.99e−15	7.11e−15	3.55e−15	1.78e−14
	1.0	1.78e−14	3.14e−13	7.11e−15	4.44e−15	1.78e−14
	0.5	1.78e−14	3.14e−13	7.11e−15	7.55e−15	1.78e−14
	0.25	1.78e−14	3.14e−13	7.99e−15	7.55e−15	1.78e−14
t^4	2.0	5.86e−14	3.20e−14	3.20e−14	4.62e−14	5.68e−14
	1.0	2.68e+00	1.18e+00	4.47e−01	4.62e−14	5.68e−14
	0.5	3.61e+00	1.18e+00	4.47e−01	4.62e−14	5.68e−14
	0.25	3.61e+00	1.32e+00	4.69e−01	4.62e−14	5.68e−14
t^5	2.0	1.56e−13	8.96e−01	9.76e−01	1.71e−13	3.48e−13
	1.0	9.74e+00	2.68e+00	2.44e+00	3.41e−01	3.12e−01
	0.5	1.24e+01	2.68e+00	2.44e+00	3.41e−01	4.17e−01
	0.25	1.27e+01	2.81e+00	2.47e+00	3.44e−01	4.65e−01

The solutions with stepsize $h = 2$ were computed within one timestep. The solution for stepsizes $h = 1.0$, $h = 0.5$ and $h = 0.25$ were computed from this solution via interpolation

discontinuities occurring in the process are, however, suitably smoothed. The fourth equation models the voltage-current characteristics of the PV element with given constants $c_i, i = 1, .., 6$. The battery is described by equations five to seven. Here, R_0, R_1, C are internal ohmic resistors and an internal capacity, respectively. The open-circuit voltage u_0 is described by a third-degree polynomial depending on the charge q_B. The main difficulties in this example are the solution of the nonlinear characteristics of the PV element and the switching processes of the load. The complete Julia Implementation is listed in the Appendix. Figure 7 shows that in this example the methods `Rodas4` and `Rodas5P` are most suitable.

5 Conclusion

Based on the construction method for `Rodas4P2` a new set of coefficients for `Rodas5` could be derived. The new `Rodas5P` method combines the properties of `Rodas5` (high order for standard problems) and `Rodas4P` or `Rodas4P2` (low order reduction for Prothero-Robinson model and parabolic problems). Moreover, it was possible to compute a fourth-order dense output formula for both methods, `Rodas5` and `Rodas5P`.

In all model problems the numerical results of the new method are in the range of the best methods from the class of Rodas schemes studied.

Therefore, `Rodas5P` can be recommended in the future as a standard method for stiff problems and index-1 DAEs for medium to high accuracy requirements within the Julia package `DifferentialEquations.jl`.

Acknowledgements This article is dedicated to my professor Peter Rentrop, who introduced me to the world of Rosenbrock–Wanner methods in 1986. Sincere thanks to Christopher Rackauckas for supporting

me during implementation of the method in DifferentialEquations.jl and to the two reviewers for valuable comments to improve the paper.

Funding Open Access funding enabled and organized by Projekt DEAL.

Declarations

Conflict of interest The author declare that they have no conflict of interest.

A Appendix

Work-precision diagrams for photovoltaic network with Julia

```
using DifferentialEquations, Plots, LaTeXStrings, DiffEqDevTools

setups = [Dict(:alg=>Rodas4()),Dict(:alg=>Rodas4P()),Dict
(:alg=>Rodas4P2()),Dict(:alg=>Rodas5()),Dict(:alg=>Rodas5P())]

#---- Problem definition ----------------------------------------------
---------------------

function ode(dy, y, p, t)
  c1,c2,c3,c4,c5,c6,R0,R1,C,q_max = p
  U0 = y[1]; U1 = y[2]; iC = y[3]; iPV = y[4]; iB = y[5]; uB = y[6]; qB = y[7];
  dy[1] = U0;
  dy[2] = iB + iPV - iC;
  dy[3] = consumer(t) - iC*(U1-U0);
  dy[4] = c1 + c2*iPV + c3*(U1-U0) + c4*(exp(c5*iPV+c6*(U1-U0))-1);
  dy[5] = U1-U0 - (oc_voltage(qB/q_max) - uB - R0*iB);
  dy[6] = iB/C - uB/(R1*C);
  dy[7] = -iB;
  nothing
end

function consumer(t)
  ts = range(3600.0,36000.0,step = 3600.0)
  return 50.0*onoff(t,ts,60.0)
end

function oc_voltage(soc)
  pp = [6.8072,-10.5555,6.2199,10.2668]; #-- Polynomial coefficients
  return ((pp[1]*soc+pp[2])*soc+pp[3])*soc+pp[4];
end

function fstep(t,t0,switchduration)
#-- atanh(0.999) = 3.8002
   s = 3.8002/switchduration
   return (tanh(s*(t-t0))+1.0)/2.0
end
```

```
function onoff(t,ts,switchduration)
   y = 0.0;
   for i=1:2:length(ts)
       y = y + fstep(t,ts[i],switchduration);
   end
   for i=2:2:length(ts)
       y = y - fstep(t,ts[i],switchduration);
   end
   return y
end

#--- Benchmark for Rodas-type methods ---
 c1 = -3.1037; c2 = 1.0015; c3 = 0.0032; c4 = 1.3984e-09; c5 = 0.4303;
 c6 = 1.5*0.9562;
 R0 = 0.2; R1 = 0.5; C = 4000.0; q_max = 36000.0;
 param = c1,c2,c3,c4,c5,c6,R0,R1,C,q_max
 y0 = [0.0, 11.856598910310167, 0.0, 2.9409008015416687, -2.940900801550821,
0.0, 9000.0]
 M = zeros(7,7); M[6,6] = 1.0; M[7,7] = 1.0;  tspan = [0.0, 36000.0];
 f = ODEFunction(ode, mass_matrix = M);  prob = ODEProblem(f, y0, tspan, param)
 sol = solve(prob,Rodas4P2(),abstol=1.0e-14,reltol=1.0e-14,maxiters=Int(1e8))
#-- reference solution
 test_sol = TestSolution(sol)
 abstols = 1.0 ./ 10.0 .^ (7:12);  reltols = 1.0 ./ 10.0 .^ (7:12)
 wp = WorkPrecisionSet(prob,abstols,reltols,setups; save_everystep=true,
error_estimate=:l2,appxsol=test_sol,numruns=20)
 P_1 = plot(wp,xlabel=latexstring("\$l_2\$ error"))
 wp = WorkPrecisionSet(prob,abstols,reltols,setups; save_everystep=true,
error_estimate=:L2, dense_errors=true,appxsol=test_sol,numruns=20)
 P_2 = plot(wp,xlabel=latexstring("\$L_2\$ error"))
 plot(P_1,P_2,layout=(1,2))
```

References

1. Alonso-Mallo, I., Cano, B.: Spectral/Rosenbrock discretizations without order reduction for linear parabolic problems. APNUM **41**(2), 247–268 (2002)
2. Alonso-Mallo, I., Cano, B.: Efficient time integration of nonlinear partial differential equations by means of Rosenbrock methods. Mathematics **9**(16), 1970 (2021). https://doi.org/10.3390/math9161970
3. Di Marzo, G.: RODAS5(4)-Méthodes de Rosenbrock d'ordre 5(4) adaptées aux problemes différentiels-algébriques. MSc mathematics thesis, Faculty of Science, University of Geneva, Switzerland (1993)
4. Hairer, E., Wanner, G.: Solving Ordinary Differential Equations II, Stiff and Differential Algebraic Problems, 2nd edn. Springer-Verlag, Berlin Heidelberg (1996)
5. Jax, T.: A rooted-tree based derivation of ROW-type methods with arbitrary Jacobian entries for solving index-one DAEs, Dissertation, University Wuppertal (2019)
6. Kaps, P., Rentrop, P.: Generalized Runge–Kutta methods of order four with stepsize control for stiff ordinary differential equations. Numer. Math. **33**, 55–68 (1979)
7. Lamour, R., März, R., Tischendorf, C.: Differential-Algebraic Equations: A Projector Based Analysis, Differential-Algebraic Equations Forum book series. Springer, London (2013)
8. Lang, J.: Rosenbrock–Wanner Methods: Construction and Mission. In: Jax, T., Bartel, A., Ehrhardt, M., Günther, M., Steinebach, G. (eds.) Rosenbrock–Wanner-Type Methods, pp. 1–17. Mathematics Online First Collections Springer, Cham (2021). https://doi.org/10.1007/978-3-030-76810-2_2
9. Lang, J., Teleaga, D.: Towards a fully space-time adaptive FEM for magnetoquasistatics. IEEE Trans. Magn. **44**, 1238–1241 (2008)
10. Lang, J., Verwer, J.G.: ROS3P-An Accurate Third-Order Rosenbrock Solver Designed for Parabolic Problems. J. BIT Numer. Math. **41**, 731 (2001). https://doi.org/10.1023/A:1021900219772

11. Lubich, Ch., Roche, M.: Rosenbrock methods for differential-algebraic systems with solution-dependent singular matrix multiplying the derivative. Computing **43**, 325–342 (1990). https://doi.org/10.1007/BF02241653
12. Mazzia, F., Cash, J.R., Soetaert, K.: A test set for stiff initial value problem solvers in the open source software R. J. Comput. Appl. Math. **236**, 4119–4131 (2012)
13. Mazzia, F., Magherini, C.: Test set for initial value problem solvers, release 2.4 (Rep. 4/2008). Department of Mathematics, University of Bari, Italy. see https://archimede.uniba.it/testset/testsetivpsolvers/
14. Ostermann, A., Roche, M.: Rosenbrock methods for partial differential equations and fractional orders of convergence. SIAM J. Numer. Anal. **30**, 1084–1098 (1993)
15. Prothero, A., Robinson, A.: The stability and accuracy of one-step methods. Math. Comp. **28**, 145–162 (1974)
16. Rackauckas, C., Nie, Q.: Differentialequations.jl-a performant and feature-rich ecosystem for solving differential equations in Julia. J. Open Res. Softw. **5**(1), 15 (2017)
17. Rang, J.: Improved traditional Rosenbrock–Wanner methods for stiff ODEs and DAEs. J. Comput. Appl. Math. **286**, 128–144 (2015)
18. Rang, J.: The Prothero and Robinson example: Convergence studies for Runge–Kutta and Rosenbrock–Wanner methods. Appl. Numer. Math. **108**, 37–56 (2016)
19. Rentrop, P., Roche, M., Steinebach, G.: The application of Rosenbrock–Wanner type methods with stepsize control in differential-algebraic equations. Numer. Math. **55**, 545–563 (1989)
20. Roche, M.: Rosenbrock methods for differential algebraic equations. Numer. Math. **52**, 45–63 (1988)
21. Sandu, A., Verwer, J.G., Van Loon, M., Carmichael, G.R., Potra, F.A., Dabdub, D., Seinfeld, J.H.: Benchmarking stiff ode solvers for atmospheric chemistry problems-I. implicit vs explicit. Atmos. Environ. **31**(19), 3151–3166 (1997). https://doi.org/10.1016/S1352-2310(97)00059-9
22. Sanz-Serna, J.M., Verwer, J.G., Hundsdorfer, W.H.: Convergence and order reduction of Runge–Kutta schemes applied to evolutionary problems in partial differential eqautions. Numer. Math. **50**, 405–418 (1986)
23. Scholz, S.: Order barriers for the B-convergence of ROW methods. Computing **41**, 219–235 (1989)
24. Steinebach, G.: Order-reduction of ROW-methods for DAEs and method of lines applications. Preprint-Nr. 1741, FB Mathematik, TH Darmstadt (1995)
25. Steinebach, G.: Improvement of Rosenbrock–Wanner Method RODASP. In: Reis, T., Grundel, S., Schöps, S. (eds.) Progress in differential-algebraic equations II. Differential-algebraic equations forum, pp. 165–184. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-53905-4_6
26. Steinebach, G., Dreistadt, D.M.: Water and hydrogen flow in networks: modelling and numerical solution by ROW methods. In: Jax, T., Bartel, A., Ehrhardt, M., Günther, M., Steinebach, G. (eds.) Rosenbrock–Wanner-Type Methods, pp. 19–47. Mathematics Online First Collections, Springer, Cham (2021)
27. Verwer, J.G.: Gauss-Seidel iteration for stiff ODEs from chemical kinetics. SIAM J. Sci. Comput. **15**(5), 1243–1259 (1994)