juliacon

Rosenbrock methods within OrdinaryDiffEq.jl - Overview, recent developments and applications -

Gerd Steinebach[1]

[1]University of Applied Sciences Bonn-Rhein-Sieg

ABSTRACT

Currently, more than 30 Rosenbrock-type methods are implemented in the widely used Julia package `OrdinaryDiffEq.jl`. We discuss the differences and similarities of the various methods and explain why there is room for further improvements. In particular, this concerns the solution of time dependent algebraic equations and continuous output within the solution of DAE problems. We present new methods `Rodas23W` and `Rodas3P` that are equipped with an error control of the interpolation. We compare this approach with alternative concepts such as residual control and modification of the stiffly accurate embedded method of `Rodas5P`, perform some benchmarks and present an application in the field of energy network simulation.

Keywords

Julia, OrdinaryDiffEq.jl, Rosenbrock methods, continuous output, differential-algebraic equations, Rodas family, residual control

1. Rosenbrock-type methods

We consider Rosenbrock-type methods for initial value problems of type

$$M\,y' = f(t,y)\,,\ y(t_0) = y_0\,. \tag{1}$$

These problems include systems of ordinary differential equations (ODEs) and in case of singular matrix M, systems of differential-algebraic equations (DAEs). In the second case we assume that the DAE system has the index one. When we can split (1) into

$$y' = f(t,y,z)\,, \tag{2}$$
$$0 = g(t,y,z)\,, \tag{3}$$

the matrix $g_z = \frac{\partial g}{\partial z}$ must be non singular. We refer further to [2] for the index definition.
Rosenbrock-Wanner (ROW) methods for problems of type (1) are defined by

$$(M - h\,\gamma\,f_y)k_i = hf(t_0 + \alpha_i h, y_0 + \sum_{j=1}^{i-1}\alpha_{ij}k_j) + h\,f_y\sum_{j=1}^{i-1}\gamma_{ij}k_j + h^2\gamma_i f_t, \tag{4}$$

$$y_1 = y_0 + \sum_{i=1}^{s} b_i k_i, \tag{5}$$
$$\text{with}\quad f_y = \frac{\partial f}{\partial y}(t_0,y_0)\,,\quad f_t = \frac{\partial f}{\partial t}(t_0,y_0)\,.$$

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

The theory of these methods is described very well in the books of Hairer, Wanner [2] and Strehmel, Weiner, Podhaisky [32], a good overview of the development of the methods since the 1960s is given by Lang [7].
One disadvantage of ROW methods is that a new Jacobian matrix f_y must be calculated at each time step. A key reason for the efficiency of the methods in Julia is the calculation of the Jacobian with automatic differentiation.
To reduce the number of evaluations of the Jacobian, so-called Rosenbrock-W methods were introduced in [26]. These use an approximation of the Jacobian and can therefore be more efficient. While some W methods are known for ODEs, such methods are difficult to derive for DAEs. Jax [4] showed that for DAEs of index one the number of order conditions to be fulfilled is considerable. For a method of order 3, for example, 26 order conditions must be fulfilled. The derivatives g_z of the algebraic equations contained in (3) must nevertheless be exact, or at least keep to the error order $\mathcal{O}(h)$. For this reason, the most common W methods currently have a maximum order of two for DAEs.
The largest collection of implemented Rosenbrock-type methods is certainly contained in the Julia package `OrdinaryDiffEq.jl`, which is part of `DifferentialEquations.jl` developed by Rackauckas and Nie [14]. Table 1 summarizes the characteristics of the currently implemented schemes.
Only the methods are specified which have a step size control. This is done using an embedded scheme with its own weights $\hat{b}_i$, such that

$$\hat{y}_1 = y_0 + \sum_{i=1}^{s}\hat{b}_i k_i$$

is an approximation of the solution with order $\hat{p}$. The orders of the embedded method are shown in brackets in Table 1.
The stage number s is a measure of the numerical effort per time step. For some methods, however, the number of function evaluations can also be less than s. This applies, for example, to the first

Table 1. Rosenbrock-type methods within `OrdinaryDiffEq.jl` and their properties.

Method	Ref.	s	ODE	W	DAE	Para	A	$R(\infty)$	Cont
GRK4T	[6] 1979	4	4(3)	-	2(-)	2(-)	-	0.45(2.6)	-
GRK4A	[6] 1979	4	4(3)	-	2(2)	2(2)	x	0.99(0.31)	-
RosShamp4	[24] 1982	4	4(3)	-	2(2)	2(2)	x	0.33(1.0)	-
Veldd4	[33] 1984	4	4(3)	-	2(-)	2(-)	-	0.24(2.6)	-
Velds4	[33] 1984	4	4(3)	2/-	2(2)	2(2)	x	0.33(0.33)	-
Scholz4_7	[23] 1989	3	3(1)	-	3(2)	3(2)	x	0.73(0.52)	-
Ros4LStab	[2] 1991	4	4(3)	-	2(2)	2(2)	x	0.0(0.55)	-
Rodas4	[2] 1991	6	4(3)	-	4(3)	2(2)	x	0.0(0.0)	x
Rodas42	[2] 1991	6	4(3)	-	4(3)	2(2)	x	0.0(0.0)	x
Rodas5	[11] 1993	8	5(4)	-	5(4)	2(2)	x	0.0(0.0)	x
Rodas4P	[27] 1995	6	4(3)	-	4(3)	4(3)	x	0.0(0.0)	x
ROS3	[22] 1997	3	3(2)	-	2(2)	2(2)	x	0.0(0.5)	-
Rodas3	[22] 1997	4	3(2)	-	3(2)	2(2)	x	0.0(0.0)	-
Rosenbrock23	[25] 1997	3	2(3)	2/-	2(-)	2(-)	x	0.0(1.6)	x
Rosenbrock32	[25] 1997	3	3(2)	2/-	-(2)	-(2)	-	1.6(0.0)	x
ROS3P	[8] 2001	3	3(2)	-	3(2)	3(2)	x	0.73(0.73)	-
ROS34PW1a	[15] 2005	4	3(2)	3/-	3(2)	3(2)	x	0.0(0.0)	-
ROS34PW1b	[15] 2005	4	3(2)	3/-	3(2)	3(2)	x	0.0(0.0)	-
ROS34PW2	[15] 2005	4	3(2)	3/2	3(2)	3(2)	x	0.0(0.48)	-
ROS34PW3	[15] 2005	4	4(2)	3/-	3(2)	3(2)	x	0.63(0.43)	-
ROS2S	[3] 2012	3	2(1)	2/-	2(2)	2(1)	x	0.0(0.33)	-
ROS2PR	[16] 2012	3	2(1)	-	2(2)	2(1)	x	0.0(0.0)	-
ROS34PRw	[16] 2012	4	3(2)	3/2	3(2)	3(2)	x	0.0(0.25)	-
ROS3PR	[17] 2014	3	3(2)	-	3(2)	3(2)	x	0.73(0.73)	-
ROS3PRL	[17] 2014	4	3(2)	-	3(2)	3(2)	x	0.0(0.25)	-
ROS3PRL2	[18] 2015	4	3(2)	-	3(2)	3(2)	x	0.0(0.25)	-
Rodas4P2	[29] 2020	6	4(3)	2/2	4(3)	4(3)	x	0.0(0.0)	x
Rodas5P	[31] 2023	8	5(4)	2/2	5(4)	4(4)	x	0.0(0.0)	x
Rodas23W	2024	5	2(3)	2/2	2(3)	2(3)	x	0.0(0.0)	x
Rodas3P	2024	5	3(2)	-	3(2)	3(2)	x	0.0(0.0)	x
Rodas5Pe	2024	8	5(4)	2/2	5(3)	4(4)	x	0.0(0.46)	x
Rodas5Pr	2024	8	5(4)	2/2	5(4)	4(4)	x	0.0(0.0)	x

s = number of stages, order of convergence for ODEs, DAEs and parabolic problem (embedded method in brackets), W = order of W method for ODEs / DAEs, A = A stable, $R(\infty)$ = limit of stablity function, Cont = continuous output for DAEs.

five methods listed and is achieved by a clever choice of the coefficients α_{ij}.

Roche [21] first considered ROW methods for DAEs and derived corresponding order conditions. Therefore, all methods before 1988 are only designed for stiff ODEs and achieve a maximum order of two for DAEs. With `Rodas4`, Hairer and Wanner [2] then laid the foundation for a whole family of stiffly accurate methods for ODEs and DAEs. This property guarantees that the stability function $R(z)$ vanishes at infinity. The stability function results when a method is applied to Dahlquist's test equation $y' = \lambda y$, $y(t_0) = y_0$. We obtain $y_1 = R(z)y_0$ with $z = \lambda \cdot h$. For A-stable methods $|R(z)| \leq 1$ must hold for all z with $Re(z) \leq 0$. If in addition $\lim_{z\to\infty} R(z) = R(\infty) = 0$, the method is called L-stable. $|R(\infty)| < 1$ is also a requirement for the convergence concerning DAE problems.

With ODEs, the continuous output of the solution can simply be done using Hermite interpolation, because y_0, $y_0' = f(t_0, y_0)$, y_1, $y_1' = f(t_1, y_1)$ are known. Ostermann [12] investigated the continuous extensions of Rosenbrock-type methods. All schemes of the `Rodas` family (except `Rodas3`) as well as `Rosenbrock23/32` have their own interpolation, which is also suitable for DAEs. If the matrix M in (1) is a diagonal matrix, a distinction can be made between differential and algebraic equations. In this case, all other methods use Hermite interpolation for the differential equations and linear interpolation for the algebraic equations.

W methods for ODEs were derived in particular by Rang (`ROS34P` family), while W methods for DAEs are only available up to order 2. Currently, however, the W property is not utilized in the implementation, e.g. to reduce the number of evaluations of the Jacobian matrix. However, it can be assumed that these methods are more suitable if the Jacobian matrix is approximated for example by finite differences and is therefore not exact.

Scholz [23] investigated the order reduction of Rosenbrock methods. Further results were provided by Ostermann, Roche [13], Lubich, Ostermann [9] and Rang [19]. There, the behavior of the methods for the Prothero-Robinson equation and semi-discretized parabolic problems were analyzed. Lang and Rang derived several methods (`ROS3P` and `ROS34P` family) that fulfill certain order conditions for the Prothero-Robinson equation. Methods `Rodas4P`, `Rodas4P2`, `Rodas5P` are based on the additional conditons of Scholz. Since order reductions for the Prothero-Robinson and parabolic problems are strongly related, we restrict ourselves to numerical investigations. All methods were applied to the nonlinear parabolic problem (4.2) form [31] with $n_x = 500$ space discretization points. The numerical order obtained is shown in Table 1.

Although a variety of methods exist, there are applications that are not solved satisfactorily. This applies in particular to the continu-

ous output of the solution for certain DAEs, which is discussed in Section 2. This was the reason to derive new low order methods `Rodas23W`, `Rodas3P` (Section 3) and to implement two new variants of `Rodas5P` (Section 4). Finally, some benchmarks and applications are presented in Section 5.

2. Continuous output for DAE problems

The first problem was discussed as an issue in the repository of `OrdinaryDiffEq`. We consider the pure algebraic equation

$$0 = y_1 - \sin(20\pi t)\,, \quad y_1(0) = 0\,. \tag{6}$$

Figures 1 shows the numerical solution obtained with methods `Rosenbrock23` and `Rodas4`. Here we see the following effects: With both methods, the time step control does not work, only seven time steps are required. However, the failure of the step size control has different causes.

`Rodas4` is a stiffly accurate method and its embedded scheme, too. Therefore, the solution of an algebraic equation like (6) is almost exact (see [2]) and the difference between the fourth order and the embedded thrid order method does not provide any useful information for the step size control. This applies regardless of the selected error tolerance. The large step sizes make the third-order interpolation, which is integrated in the method, extremely inaccurate. Of course, this can be avoided when a suitable bound for the maximum time step width h_{max} is used. This example was the reason to issue a warning for the interpolation results when Rosenbrock methods are applied to pure algebraic equations. Of course, this can easily be circumvented by adding a second equation $y_2' = 0$. Therefore, it would be desirable to check not only the error of the solution at the discrete time steps but also the error of the interpolation. Another possibility would be the modification of the embedded scheme so that it is no longer stiffly accurate.

With `Rosenbrock23`, however, the solution at the discrete points in time is already completely wrong. The reason for this is that the third-order method used for step size control is not stable for DAEs. Figure 2 shows the stability regions of the two second- and third-order methods. For the stability function of the third order method, $R(\infty) = 1.6 > 1$ holds. Therefore, the error test is only applied internally to the differential variables. In order to prevent such large errors an additional error test for the algebraic variables was introduced recently which controls the residuum $||f(t, y)||$ at the end of each timestep. Nevertheless the modification of (6) into the equivalent problem

$$\begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} y_1' \\ y_2' \end{pmatrix} = \begin{pmatrix} y_1 - \sin(20\pi t) + 1 \\ 1 \end{pmatrix}, \begin{pmatrix} y_1(0) \\ y_2(0) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{7}$$

leads to an error exit of the method due to instability and shows, that this measure dose not solve the inherent problem with `Rosenbrock23`.

Next, we tested all of the available Rosenbrock and Rosenbrock-W methods in `DifferentialEquations.jl` on problem (7). All variants of `Rodas4`, like `Rodas42`, `Rodas4P`, `Rodas4P2`, `Rodas5`, `Rodas5P`, `Rodas3` showed the same behaviour as illustrated in Figure 1. Note, that `Rodas3` has only a Hermite interpolation, which is not suited for DAEs.

`Rosenbrock32` is the same method as `Rosenbrock23`, but the 3rd order scheme is used for the solution, here. Therefore, it is unstable and not able to solve the problem.

All other methods are not equipped with their own interpolation. Since the mass matrix M is not a diagonal matrix, it is not possible to distinguish between differential and algebraic equations

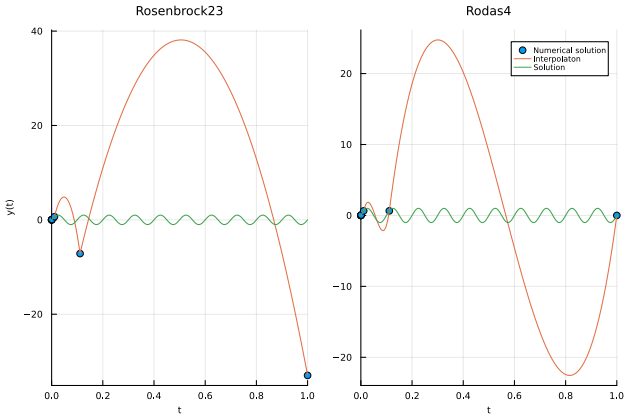

Fig. 1. Numerical and analytical solution of problem (6) with `Rosenbrock23` and `Rodas4` (OrdinaryDiffEq v6.40.1).

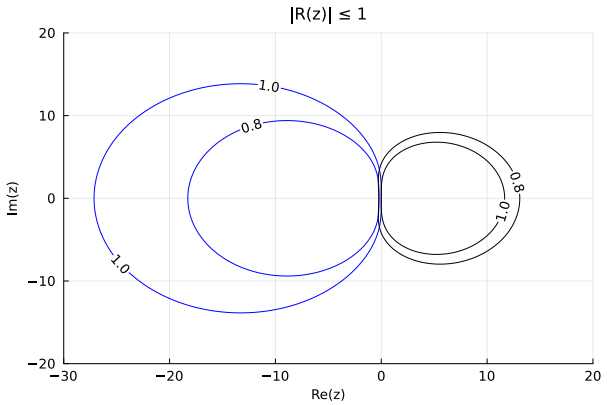

Fig. 2. Stability regions of `Rosenbrock23`. Black method of order two, blue method of order three.

and interpolation cannot be applied. The following methods are only successful if the option `dense=false` is used: `scholz4_7`, `ROS2S`, `ROS3`, `ROS3P`, `ROS3PR`, `ROS3PRL`, `ROS3PRL2`, `ROS34PW1a`, `ROS34PW1b`, `ROS34PW2`, `ROS34PW3` and `ROS34PRw`.

The next example is a simple DAE problem:

$$y_1' = -y_1\,, \tag{8}$$
$$0 = y_2 - (1 - t^2)^4\,. \tag{9}$$

For the initial values $y_1(0) = 1$, $y_2(0) = 1$ the solution is given by $y_1(t) = e^{-t}$, $y_2(t) = (1 - t^2)^4$. We solve this system in the time interval $t \in [0, 10]$ using different methods with different tolerances and show the work-precision diagram in Figure 3. The L_2-error shown is taken at 100 evenly spaced points via interpolation and should reflect the error of the dense output formulae. We can see that almost all Rosenbrock methods used have a similar behavior. Only `ROS2PR` requires significantly more computing time. The reason for the similarities is the linear interpolation which is implemented for the algebraic variables. On the other hand, we see that Rodas5P works much more efficiently, but achieves a considerably lower accuracy. The reason for the lower accuracy is again due to the stiffly accurate property of the `Rodas` methods, which all behave similarly. Due to the almost exact solution of the algebraic equation, too large time steps are again used and the in-

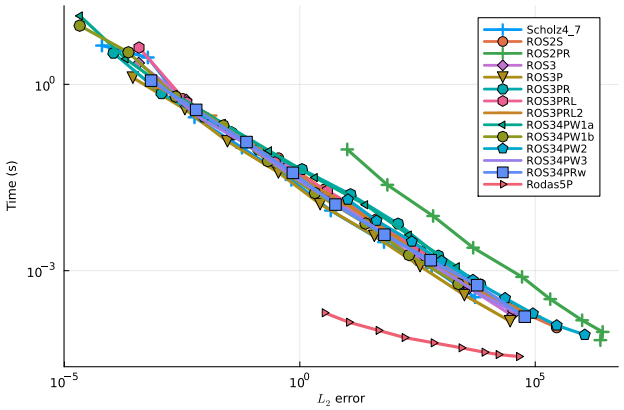

Fig. 3. Work-precision diagram for DAE system (8,9).

terpolation is inaccurate. A comparison of the required time steps illustrates the characteristics of the different methods: With a tolerance of `abstol=reltol=1.0e-8`, for example, `ROS34PRw` requires 83070 time steps, whereas `Rodas5P` requires only 52. With `Rosenbrock23` the solution was not possible within a reasonable computing time.
Therefore, the need for improved methods exists. We discuss three different approaches in Sctions 3 and 4.

3. Construction of Rodas23W / Rodas3P

First, we want an alternative for `Rosenbrock23` that is often efficient for stiff ODEs, but not for DAEs. A stiffly accurate method is desirable, but coupled with error control for the interpolation scheme. Moreover, the number of only three evaluations of the right hand side of (1) should be preserved and the W property, so that an inexact Jacobian matrix may be used. Our goal is to develop a `Rodas23W` and `Rodas3P` method. It is the same procedure, depending on which scheme is interpreted as an embedded method. `Rodas23W` is of order $p = 2$ with W property for DAEs, `Rodas3P` is of order $p = 3$ and is analogous to `Rodas4P` particularly suitable for the Prothero-Robinson equation and semi-discretized parabolic problems. Both methods have a dense-output formula of the same order. This makes it possible to check the errors of the interpolation and the problems that occurred in equation (6) can be avoided.
A method of order $p = 3$ for index one DAEs of type (1) must fulfill conditions No.1, 2, 3, 4, 5 of Table 2, see [2, 21]. Here coefficients β_i are given by $\beta_i = \sum_{j=1}^{i} \beta_{ij}$ with $\beta_{ij} = \alpha_{ij} + \gamma_{ij}$ for $j < i$ and $\beta_{ii} = \gamma$. When collecting coefficients β_{ij} in matrix B, coefficients w_{ij} arise from matrix $W = B^{-1}$.
If an inexact Jacobian f_y or f_t is to be used in equation (4), additionally the W conditions must be considered. Up to order $p = 2$ these are conditions No.6, 7, 8 of Table 2. No.7 and No.8 only apply, when DAEs are solved. Note, that nevertheless derivatives g_z of the algebraic equations (3) with respect to the algebraic variables should be exact or at least of error order $\mathcal{O}(h)$, see [4]. Condition No.9 guarantees convergence order $p = 2$ for certain DAEs of index two, see [10].
To reduce order reduction, condition No.10 is taken into account. It is discussed in [23, 31] and means, that in the stiff case order reduction for the Prothero-Robinson model is restricted to $p = 2$ and for semi-discretized parabolic PDEs order $p = 3$ is preserved.
Now, we want to construct a method of order $p = 3$ with an embedded scheme of order $\hat{p} = 2$ and stages $\hat{s} = s - 1$, fulfilling as

Table 2. Order conditions for methods `Rodas23W` and `Rodas3P`.

No		Order	Condition
1	ODE	1	$\sum b_i = 1$
2	ODE	2	$\sum b_i \beta_i = 1/2$
3	ODE	3	$\sum b_i \alpha_i^2 = 1/3$
4	ODE	3	$\sum b_i \beta_{ij} \beta_j = 1/6$
5	DAE	3	$\sum b_i w_{ij} \alpha_j^2 = 1$
6	W ODE	2	$\sum b_i \alpha_i = 1/2$
7	W DAE	2	$\sum b_i \alpha_{ij} w_{jk} \alpha_k = 1/2$
8	W DAE	2	$\sum b_i w_{ij} \alpha_j = 1$
9	Index-2	2	$\sum b_i w_{ij} w_{jk} \alpha_k^2 = 2$
10	Prot-Rob	3	$C_2(H) = \sum_{i=0}^{s} A_i H^i = 0$

Table 3. Fulfilled order conditions for methods `Rodas23W` and `Rodas3P` and its continuous output formulas.

	1	2	3	4	5	6	7	8	9	10
Rodas3P	x	x	x	x	x			x	x	x
continuous output	x	x	x	x	x					
Rodas23W	x	x			x	x	x	x	x	x
continuous output	x	x			x	x		x		

many conditions as possible. Both methods should be stiffly accurate. This leads to conditions $b_i = \beta_{si}$ for $i = 1, ..., s$, $\hat{b}_i = \beta_{s-1,i}$ for $i = 1, ..., s-1$ and $\alpha_s = \alpha_{s-1} = 1$.
Moreover, both methods should be equipped with interpolation formulas for dense output of the same order. For interpolation between y_0 and y_1 we apply equations (10), (11), see [2].

$$y(t_0 + \tau\, h) = y_0 + \sum_{i=1}^{s} b_i(\tau) k_i\,, \quad \tau \in [0, 1]\,, \tag{10}$$

$$b_i(\tau) = \tau(b_i - c_i) + \tau^2(c_i - d_i) + \tau^3 d_i\,. \tag{11}$$

The computation of coeffcients c_i and d_i, $i = 1, ..., s$ requires as many linear equations as order conditions are to be fulfilled. Thus we need $s = 5$ stages for a third order interpolation. In order to minimize computational efforts per timestep, the number of function evaluations should be lower than the stagenumber $s = 5$.
It turnes out that it is possible to derive a method `Rodas3P` with embedded scheme `Rodas23W` fulfilling the order conditions given in Table 3.
The coefficients are shown in Table 4. As we can see from the α coefficients, only three evaluations of the right-hand side of the system (1) are required. oth methods are A stable, their stability regions are shown in Figure 4. If we need the maximum possible order, we use `Rodas3P` and `Rodas23W` as embedded methods for stepsize control. If we want to take advantage of the W properties, we can also proceed in exactly the opposite way.
Because the interpolation formulas of both methods also have the order $p = 3$ and $p = 2$ respectively, we can check their errors, too. The difference $y(t_0 + \tau\, h) - \hat{y}(t_0 + \tau\, h)$ leads to a third order polynomial in τ. Thus we can compute the maximum difference

$$err_i = \max_{\tau \in [0,1]} |y_i(t_0 + \tau\, h) - \hat{y}_i(t_0 + \tau\, h)|$$

for each component y_i of the interpolated solution. Similar to the error control after each timestep we require $err_i \leq abstol + reltol\,|y_i|$ and reduce the stepsize when this condition is violated.

Table 4. Coefficients of `Rodas23W`(ˆ) and `Rodas3P`.

$$\alpha = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ \frac{4}{9} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -\frac{217}{384} & \frac{183}{128} & \frac{13}{96} & 0 & 0 \\ -\frac{217}{384} & \frac{183}{128} & \frac{13}{96} & 0 & 0 \end{pmatrix}, \quad \beta = \begin{pmatrix} \frac{1}{3} & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & 0 & 0 & 0 \\ -\frac{1}{12} & \frac{3}{4} & \frac{1}{3} & 0 & 0 \\ \frac{3}{8} & \frac{3}{8} & -\frac{1}{12} & \frac{1}{3} & 0 \\ \frac{33}{8} & -\frac{27}{8} & -\frac{3}{4} & \frac{2}{3} & \frac{1}{3} \end{pmatrix},$$

$$\gamma = \frac{1}{3}, \quad b_i = \beta_{5i}, \quad \hat{b}_i = \beta_{4i},$$

$$c = (\frac{51}{4}, -\frac{27}{2}, -\frac{9}{4}, \frac{8}{3}, \frac{1}{3}); \; d = (-\frac{135}{8}, \frac{135}{8}, 3, -3, 0),$$

$$\hat{c} = (-\frac{3}{8}, -\frac{3}{8}, \frac{1}{12}, \frac{19}{30}, \frac{1}{30}); \; \hat{d} = (0, 0, 0, 0, 0).$$

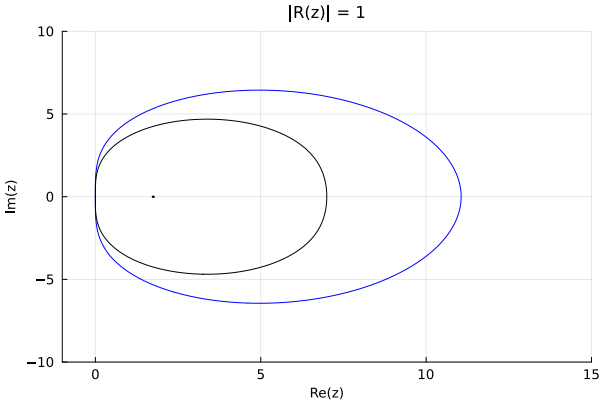

Fig. 4. Stability regions of `Rodas3P` (black) and `Rodas23W` (blue).

4. Residual control and modified embedded scheme for Rodas5P

Next we want to improve the higher order methods. We picked `Rodas5P` because it is the most efficient method of the Rodas family in many cases.

First, we equip the method with an additional error test, which checks the residual of the interpolation in half of the step size. With the help of the built-in interpolation of `Rodas5P` we can calculate $y_{int}(t_0 + \frac{h}{2})$ and $y'_{int}(t_0 + \frac{h}{2})$ and determine the residual of (1):

$$res = ||M\, y'_{int}(t_0 + \frac{h}{2}) - f(t_0 + \frac{h}{2}, y_{int}(t_0 + \frac{h}{2}))||\,.$$

If res is too large, the step size is reduced accordingly. As an error test, $res \leq abstol + reltol||y_{int}(t_0 + \frac{h}{2})||$ has proven to be practicable. We call `Rodas5P` together with this residual test `Rodas5Pr`. The additional calculation effort per step is limited to one extra function evaluation of the right-hand side of (1).

Secondly, we change the embedded method so that it is no longer stiffly accurate. The embedded scheme of `Rodas5P` has order $\hat{p} = 4$ and must fulfill 13 order conditions, see [31]. For determining new coefficients $\hat{b}_1, ..., \hat{b}_8$, $s = 8$ degrees of freedom are available . In order for these to differ from the previous ones, not all order conditions can be fulfilled. It is possible to compute new coefficients in such a way that only conditions (21),(22) from Table 3 in [31] are not fulfilled. The new embedded method therefore has order $\hat{p} = 4$ for ODEs, but only $\hat{p} = 3$ for DAEs. A-stability is preserved, but due to $\hat{R}(\infty) = 0.46$, L-stability of the embedded scheme is lost. This new method is called `Rodas5Pe`.

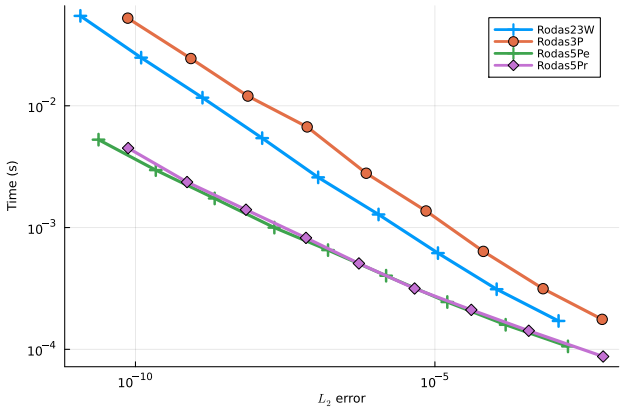

Fig. 5. Work-precision diagram for DAE system (7).

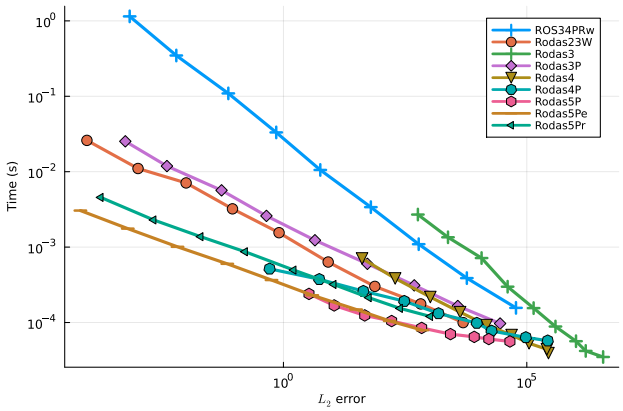

Fig. 6. Work-precision diagram for DAE system (8,9).

5. Benchmarks and applications

First, we consider problem (7). Figure 5 shows the work-precision diagram for the new methods. Again, the L_2 error is shown, which is strongly influenced by the interpolation. Here `Rodas23W` delivers even better results than `Rodas3P`. This is because `Rodas23W` also fulfils the order condition No.5 from Table 2. `Rodas5Pe` and `Rodas5Pr` provide very similar results. The four new methods are the only ones that produce meaningful results for this problem with respect to the L_2 error.

Next, we consider problem (8,9) and create a work-precision diagram analogous to Figure 3. Figure 6 shows that the accuracy is considerably improved by the new approaches and that the efficiency is significantly better than with all other Rosenbrock-type methods.

Figure 7 shows the results for a simple linear stiff ODE system

$$\begin{pmatrix} y'_1 \\ y'_2 \\ y'_3 \end{pmatrix} = \begin{pmatrix} -0.8 & 12.5 & 0 \\ -12.5 & -0.8 & 0 \\ 0 & 0 & -100 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} \tag{12}$$

with initial conditions $y_1(0) = y_2(0) = y_3(0) = 0$. Shown here is the l_2 error for $t \in [0, 10]$, which does not include any interpolation results. The different orders of the Rodas methods are easy to distinguish and correspond to expectations. Why `ROS3P`, `ROS3PR`,

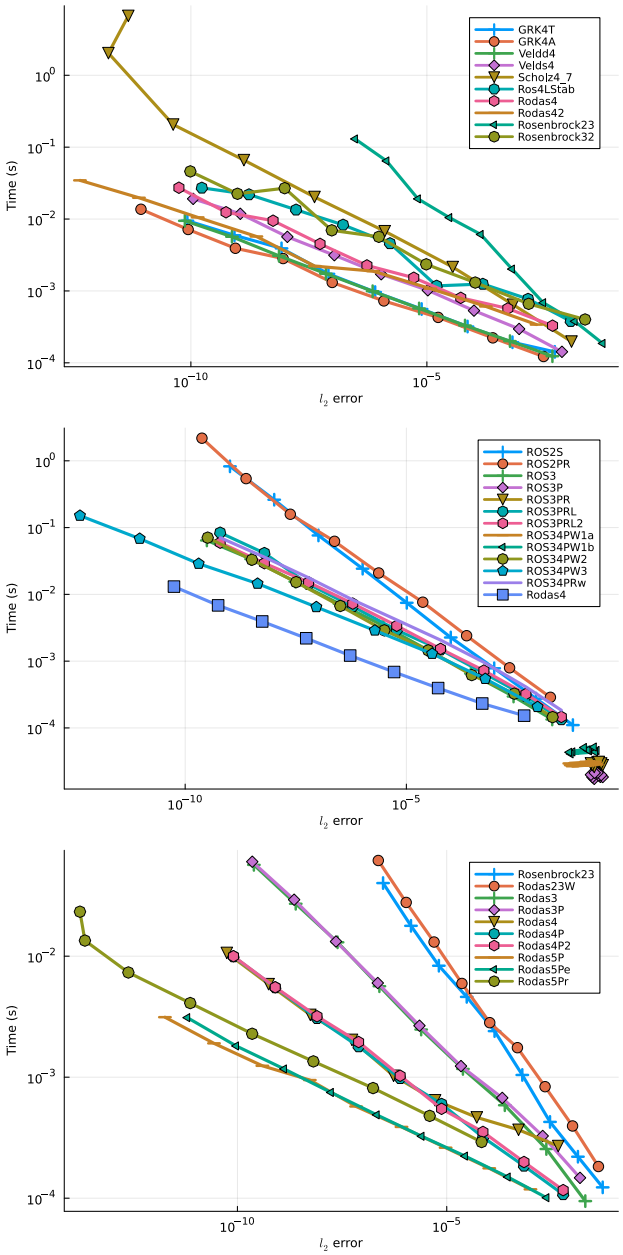

Fig. 7. Work-precision diagram for problem (12).

`ROS34PW1a` and `ROS34PW1b` do not provide plausible results could not be clarified.
All procedures from Table 1 that have the W property were applied to the following DAE system:

$$y_1' = \frac{1}{2} y_2^3 y_3 \,, \tag{13}$$

$$y_2' = \frac{1}{6} y_2 y_3 \,, \tag{14}$$

$$0 = y_3 + \frac{6 y_1}{y_2^3} \,, \tag{15}$$

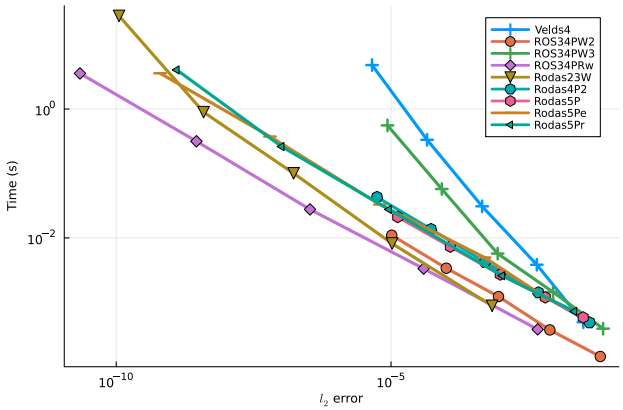

Fig. 8. Work-precision diagram for problem (13,14,15).

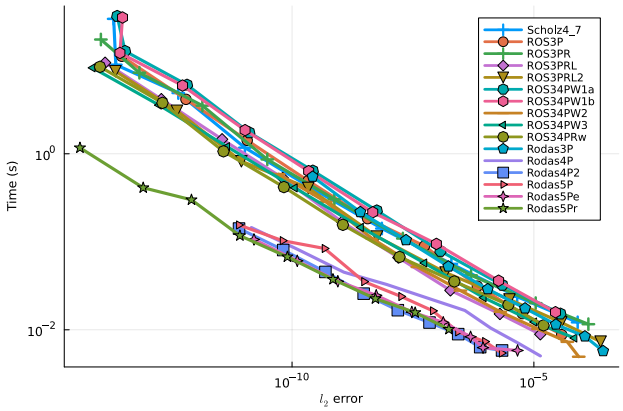

Fig. 9. Work-precision diagram for parabolic problem (4.2) in [11].

with $y_1(0) = y_2(0) = 1$, $y_3(0) = -6$ and $t \in [0, 1]$. The Jacobian matrix was evaluated at time $t = 0$ and kept constant. It should be noted that the derivative of the algebraic equation with respect to y_3 is thus exact for all t. The methods `Rosenbrock23`, `Rosenbrock32`, `ROS2S`, `ROS34PW1a`, `ROS34PW1b` could not solve the problem in a reasonable time. The results of the other methods are shown in Figure 8. Here, the convergence properties for W methods applied to DAEs given in Table 1 are confirmed.
Finally, all 16 methods from Table 1, which have an order greater than two for parabolic problems, were applied to the nonlinear parabolic problem (4.2) presented in [31]. The corresponding results using $n_x = 500$ space discretization points are shown in Figure 9. This clearly shows the advantages of the `Rodas4P`, `Rodas4P2`, `Rodas5P`, `Rodas5Pe` and `Rodas5Pr` methods in terms of efficiency. The accuracies achieved by the Rodas methods correspond to the specified tolerances $reltol = abstol \in [10^{-10}, 10^{-2}]$. The residual error control in `Rodas5Pr` additionally increases the accuracy and is within the range of the other methods.
Fortran, MATLAB and Julia Implementations of `Rodas4P` respectively `Rodas5P` have been used successfully in many applications concerning network simulation and are characterized in particular by their robustness, see [5, 20, 28, 30]. Finally, we would like to show that the behavior shown in Figure 1 may also occur in such real applications. For this purpose, we reduce the model presented in [1] to a minimal demonstration network consisting of a battery,

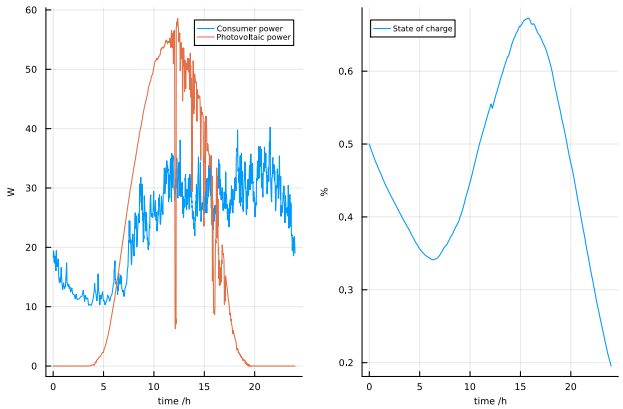

Fig. 10. Measured time series for photovoltaic and the consumer power and simulated state of charge of the battery.

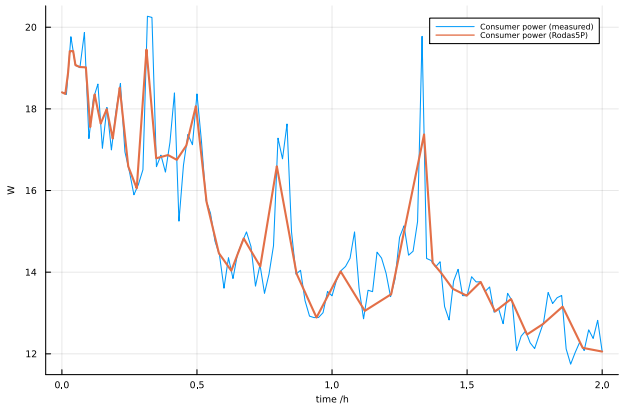

Fig. 11. Measured input data of the model and resolution generated by the simulation.

a photovoltaic feed-in and a consumer. Figure 10 shows the typical behavior over one day. The simulated state of charge of the battery is much smoother than the power of the photovoltaic and the consumer provided by measured time series.

Typically, the energy network is modeled by a DAE system and algebraic equations of type $0 = I \cdot U - P(t)$ occur internally. $P(t)$ is the measured power and I, U are time-dependent state variables for current and voltage. Figure 11 shows that the dynamics of the input data is not fully resolved by the simulation using `Rodas5P` due to time steps that are too large. The reason is the same as discussed in the example of equation (6). The problem also occurs here because the remaining components of the network have a much smoother behavior than the input data.

Table 5 shows the solver statistics for the various methods of the Rodas family. A visual match of the curves shown in Figure 11 can only be observed when using the new methods `Rodas23W`, `Rodas3P`, `Rodas5Pe`. This means, that at least around 10000 time steps are required. All other methods show the same behavior as `Rodas5P`, albeit to a lesser extent. For the residual control of `Rodas5Pr` to be effective, the relative tolerance `reltol` must be reduced slightly. For all results shown in Figure 11 and Table 5, `abstol=reltol=1.0e-4` was chosen.

Table 5. Solver statistics for the simplified energy model.

	nfcn	nsucc	nfail
Rodas23W	134172	25043	2985
Rodas3	36417	4679	2085
Rodas3P	134491	25102	2993
Rodas4	60688	6661	1233
Rodas4P	53164	5707	1251
Rodas4P2	44898	4736	1168
Rodas5	9856	871	143
Radas5P	11628	1013	187
Rodas5Pr	12063	983	156
Rodas5Pe	112542	9450	2255

$nsucc$ = number of successful time steps, $nfail$ = number of rejected steps.

In conclusion, it can be summarized that with `Rodas5Pe` and `Rodas5Pr` two new variants of `Rodas5P` are available. These can be used when time-dependent algebraic equations cannot be solved with sufficient resolution. `Rodas5Pe` uses an alternative embedded method that is not stiffly accurate, `Rodas5Pr` is equipped with an error control for the residual of the interpolation. Alternatively, lower order methods can be applied, which are equipped with an error control by the second and third order interpolation schemes. `Rodas3P` has similar properties to `Rodas4P`, `Rodas5P` and `Rodas23W` is preferable due to its W property if the Jacobian matrix is not exactly known.

6. References

[1] M. Bareev-Rudy, S. Meiswinkel, M. Pfennig, S. Schedler, B. Schiffer, G. Steinebach, T. Clees, Analysis of PtGtX Systems with Metal Hydride Storage Based on Coupled Electrochemical and Thermodynamic Simulation, Energy Conversion and Management, submitted.

[2] E. Hairer and G. Wanner, Solving Ordinary Differential Equations II, Stiff and differential algebraic Problems, (2nd ed.), Springer-Verlag, Berlin Heidelberg (1996)

[3] A. W. Hamkar, S. Hartmann, J. Rang, A stiffly accurate Rosenbrock-type method of order 2. Appl. Num. Math., 62(12):1837–1848 (2012)

[4] T. Jax, A rooted-tree based derivation of ROW-type methods with arbitrary jacobian entries for solving index-one DAEs, Dissertation, University Wuppertal (2019)

[5] T. Jax, G. Steinebach, Generalized ROW-Type Methods for Simulating Water Supply Networks, In: Quintela, Barral et al. (Eds.): Progress in Industrial Mathematics at ECMI 2016, 19th European Conference on Mathematics for Industry, Springer, Cham,447-454 (2017). https://doi.org/10.1007/978-3-319-63082-3_70

[6] P. Kaps, P. Rentrop, Generalized Runge-Kutta Methods of Order Four with Stepsize Control for Stiff Ordinary Differential Equations, Numerische Mathematik, 33, 55-68 (1979)

[7] J. Lang, Rosenbrock-Wanner Methods: Construction and Mission, In: Jax T., Bartel A., Ehrhardt M., Günther M., Steinebach G. (eds) Rosenbrock—Wanner–Type Methods. Mathematics Online First Collections, Springer, Cham., 1-17 (2021). https://doi.org/10.1007/978-3-030-76810-2_2

[8] J. Lang, J.G. Verwer, ROS3P—An Accurate Third-Order Rosenbrock Solver Designed for Parabolic Problems, J. BIT Numerical Mathematics 41, 731–738 (2001). doi:10.1023/A:1021900219772

[9] C. Lubich, A. Ostermann, Linearly implicit time discretization of non-linear parabolic equations, IMA Journal of Numerical Analysis, 15(4), 555–583 (1995). https://doi.org/10.1093/imanum/15.4.555

[10] C. Lubich, M. Roche, 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

[11] G. Di Marzo, 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)

[12] A. Ostermann, Continuous extensions of Rosenbrock-type methods, Computing 44, 59–68 (1990). https://doi.org/10.1007/BF02247965

[13] A. Ostermann, M. Roche, Rosenbrock methods for partial differential equations and fractional orders of convergence, SIAM J. Numer. Anal. 30, 1084-1098 (1993)

[14] C. Rackauckas, Q. Nie, Differentialequations.jl–a performant and feature-rich ecosystem for solving differential equations in julia, Journal of Open Research Software, 5(1), p.15, Ubiquity Press (2017)

[15] J. Rang, L. Angermann, New Rosenbrock W-methods of order 3 for partial differential algebraic equations of index 1, BIT 45, 761-787 (2005)

[16] J Rang, An analysis of the Prothero–Robinson example for constructing new DIRK and ROW methods, Informatik-Bericht 2012-03, TU Braunschweig (2012)

[17] J. Rang, The Prothero and Robinson example: Convergence studies for Runge-Kutta and Rosenbrock-Wanner methods, Informatikbericht 2014-05, TU Braunschweig (2014). https://doi.org/10.24355/dbbs.084-201408121139-0

[18] J. Rang, Improved traditional Rosenbrock-Wanner methods for stiff ODEs and DAEs, Journal of Computational and Applied Mathematics 286 , 128-144 (2015)

[19] J. Rang, The Prothero and Robinson example: Convergence studies for Runge-Kutta and Rosenbrock-Wanner methods, Applied Numerical Mathematics 108, 37-56 (2016)

[20] P. Rentrop, G. Steinebach, Model and numerical techniques for the alarm system of river Rhine, Surveys Math. Industry 6(4), 245-265, Springer, Wien (1997)

[21] M. Roche, Rosenbrock methods for differential algebraic equations, Numerische Mathematik, 52, 45-63 (1988)

[22] A. Sandu, J.G. Verwer, M. Van Loon, G.R. Carmichael, F.A. Potra, D. Dabdub, J.H. Seinfeld, Benchmarking stiff ode solvers for atmospheric chemistry problems-I. implicit vs explicit, Atmospheric Environment, 31(19), 3151-3166 (1997). doi.org/10.1016/1352-2310(97)00059-9

[23] S. Scholz, Order barriers for the B-Convergence of ROW Methods, Computing 41, 219-235 (1989)

[24] L. F. Shampine, Implementation of Rosenbrock Methods, ACM Transactions on Mathematical Software (TOMS), 8: 2, 93-113 (1982). doi:10.1145/355993.355994

[25] L. F. Shampine, M. W. Reichelt, The MATLAB ODE Suite, SIAM Journal on Scientific Computing, 18(1), 1-22 (1997). https://doi.org/10.1137/S1064827594276424

[26] T. Steihaug, A. Wolfbrandt, An Attempt to Avoid Exact Jacobian and Nonlinear Equations in the Numerical Solution of Stiff Differential Equations, Math. Comp., 33, 521 - 534 (1979)

[27] G. Steinebach, Order-reduction of ROW-methods for DAEs and method of lines applications. Preprint-Nr. 1741, FB Mathematik, TH Darmstadt (1995)

[28] G. Steinebach, From River Rhine Alarm Model to Water Supply Network Simulation by the Method of Lines, In: Russo, Capasso et al. (Eds.): Progress in Industrial Mathematics at ECMI 2014. Mathematics in Industry, Vol 22, Springer International, Cham, 783-792, (2026). https://doi.org/10.1007/978-3-319-23413-7_109

[29] G. Steinebach, Improvement of Rosenbrock-Wanner Method RODASP, In: Reis T., Grundel S., Schöps S. (eds) Progress in Differential-Algebraic Equations II. Differential-Algebraic Equations Forum. Springer, Cham., 165-184, (2020). doi:10.1007/978-3-030-53905-4_6

[30] G. Steinebach, D. M. Dreistadt, 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. Mathematics Online First Collections, Springer, Cham., 19-47, (2021). https://doi.org/10.1007/978-3-030-76810-2_2

[31] G. Steinebach, Construction of Rosenbrock-Wanner method Rodas5P and numerical benchmarks within the Julia Differential Equations package, Bit Numer Math 63, 27 (2023). https://doi.org/10.1007/s10543-023-00967-x

[32] K. Strehmel, R. Weiner, H. Podhaisky, Numerik gewöhnlicher Differentialgleichngen, 2. Auflage, Springer Spektrum (2012)

[33] M. V. van Veldhuizen, D-stability and Kaps-Rentrop-methods, Computing 32, 229-237 (1984). doi:10.1007/BF02243574