Fourth-Order Parallel Rosenbrock Formulae for Stiff Systems

D. A. Voss
Department of Mathematics, Western Illinois University
1 University Circle, Macomb, IL 61455-1390, U.S.A.
d-voss@wiu.edu

Abstract—Rosenbrock methods are frequently used for the numerical solution of stiff initial value problems. Such linearly implicit methods are characterized by a relatively easy implementation together with excellent linear stability properties. In this paper, we consider modified Rosenbrock methods with s external linearly implicit stages each of which contains p additional linearly implicit internal stages. The internal stages are already parallel so that they can be solved for independently of each other and, consequently, the processors need to exchange their results only after the completion of each of the s external stages. We focus on the design of fourth-order methods with three external stages. Using embedded third-order methods, a variable step size implementation is compared with well-known Rosenbrock codes for performance on the Robertson problem. © 2005 Elsevier Ltd. All rights reserved.

Keywords—Rosenbrock methods, A-stable, Parallel algorithm, Stiff initial value problem, Adaptivity.

1. INTRODUCTION

We consider the numerical solution of systems of initial value ordinary differential equations (ODEs), i.e., initial value problems (IVPs), of the form

\[ y'(t) = f(y(t)), \quad y(t_0) = y_0, \] 

where \( y : \mathbb{R} \to \mathbb{R}^m \) and \( f : \mathbb{R}^m \to \mathbb{R}^m \). Runge-Kutta methods applied to stiff systems of form (1) must necessarily be implicit due to stability constraints on the step size, but ultimately necessitate significant computational effort due to their iterative nature. Linearly implicit methods, such as Rosenbrock methods [1], have already proven very effective at low to modest accuracies for a wide variety of stiff problems [2]. Moreover, they are noniterative in design which is an asset for parallel implementation as true iteration may create load imbalancing in a parallel implementation [3,4]. When applied to (1), an s-stage Rosenbrock method has the form [5,6]

\[ y_{n+1} = y_n + \sum_{i=1}^{s} b_i k_i, \]

\[ (I - \gamma_i h J_n) k_i = h f \left( y_n + h \sum_{j=1}^{i-1} a_{i,j} k_j \right) + h J_n \sum_{j=1}^{i-1} \gamma_{i,j} k_j, \]
where \( J_n = \left( \frac{\partial f}{\partial y} \right)(y_n) \) and \( k_1, k_2, \ldots, k_s \) are the stages. These methods can be viewed as linearization of diagonally implicit Runge-Kutta (DIRK) formulae. Of special interest are methods for which \( \gamma_i = \gamma \), which can be viewed as linearizations of singly diagonally implicit Runge-Kutta (SDIRK) formulae; only one LU-decomposition is required per step. Cash [7] proposed a similar class of Rosenbrock-type methods given by

\[
y_{n+1} = y_n + \sum_{i=1}^{s} b_i k_i,
\]

where

\[
(I - \gamma h J_n)^2 k_i = h[I + c_i h J_n] f \left( y_n + \sum_{j=1}^{i-1} a_{ij} k_j \right).
\]

Several authors have discussed implementation details of Rosenbrock methods (see, for example, [7–10], and references therein); in particular, for scheme (3), computing each stage \( k_i \) calls for the solution of linear systems of equations of the form

\[
(I - \gamma h J_n)^2 k_i = d_i,
\]

which can be implemented efficiently by LU decomposing \( I - \gamma h J_n \) and solving the systems

\[
LU z = d_i, \quad LUK_i = z.
\]

Rosenbrock methods given by (2), or via the formulation (3), were designed for sequential computers. Recently, the author [11,12] proposed a class of parallel Rosenbrock-type formulae which allows the freedom of Rosenbrock methods arising from the linearization of DIRKs while maintaining the computational efficiency of those arising from SDIRKs, or those of Cash [7], on a computer with at least two processors. For this environment, in the next section, we develop an efficient A-stable parallel Rosenbrock method of order four with error estimation provided by an embedded third-order method. We focus on the design of inherently parallel methods wherein linear systems can be apportioned to concurrent processors. This “parallelism across the method” has the potential to be enhanced through the use of parallel linear algebra software. Numerical results are presented in Section 3 which include an adaptive implementation.

2. A FOURTH-ORDER MODIFIED ROSENBROCK SCHEME

The class of formulae which we consider in this section is given by

\[
y_{n+1} = y_n + \sum_{i=1}^{s} b_i k_i,
\]

where

\[
(I - \gamma_1 h J_n)(I - \gamma_2 h J_n) k_i = h(I + c_i h J_n) f \left( y_n + \sum_{j=1}^{i-1} a_{ij} k_j \right).
\]

With \( s = 3 \) stages and

\[
\begin{align*}
\alpha_{1,i} &= \gamma_1 + \gamma_2 + c_i, \\
\alpha_{2,i} &= \gamma_1^2 + \gamma_2^2 + \gamma_1 \gamma_2 + (\gamma_1 + \gamma_2) c_i, \\
\alpha_{3,i} &= \gamma_1^3 + \gamma_2^3 + \gamma_1 \gamma_2^2 + \gamma_1^2 \gamma_2 + (\gamma_1^2 + \gamma_2^2 + \gamma_1 \gamma_2) c_i,
\end{align*}
\]
formula (4) has order four if
\[ b_1 + b_2 + b_3 = 1, \]
\[ b_1 \alpha_{1,1} + b_2 (a_{2,1} + a_{1,2}) + b_3 (a_{3,1} + a_{3,2} + a_{1,3}) = \frac{1}{2}, \]
\[ b_1 \alpha_{2,1} + b_2 (a_{2,1} (a_{1,1} + a_{2,2}) + b_3 (a_{3,1} \alpha_{1,1} + a_{3,2} (a_{2,1} + a_{1,2} + a_{1,3})) + b_3 (a_{3,1} + a_{3,2}) = \frac{1}{6}, \]
\[ b_2 a_{2,1}^2 + b_3 (a_{3,1} + a_{3,2})^2 = \frac{1}{3}. \]

The seven equations (6)-(12) involve the eleven unknowns \( \gamma_1, \gamma_2, c_1, c_2, c_3, a_{2,1}, a_{3,1}, a_{3,2}, b_1, b_2, b_3. \)

If we apply method (5) to the scalar test equation \( y' = \lambda y, \) the numerical solution satisfies
\[ y_{n+1} = R(z) y_n, \quad z = \lambda h, \]
where the stability function, \( R(z), \) is a rational approximation to \( \exp(z). \) A search for parameters to ensure that \( R(z) \) was an \( A \)-acceptable approximation to \( \exp(z) \) while controlling the size of the error constants yielded the following coefficients of a fourth-order \( A \)-stable method, denoted ROP4, for which \( |R(\infty)| \approx 0.0459: \)
\[ \gamma_1 = \frac{1}{3}, \quad \gamma_2 = \frac{1}{4}, \quad a_{21} = -0.411726120900520, \quad a_{31} = \frac{3}{5}, \quad a_{32} = \frac{2}{5}, \]
\[ c_1 = 0.075998857314201, \quad c_2 = -0.440810868549577, \quad c_3 = -0.325572011917899, \]
\[ b_1 = 0.377477860885340, \quad b_2 = 0.348218196893093, \quad b_3 = 0.274303942221567. \]

For the purpose of error estimation, we construct a third-order method by solving (6)-(9) while keeping the same values of \( \gamma_1, \gamma_2, a_{21}, a_{31}, a_{32} \) as the fourth-order method. Consequently, this embedded method will not require any new function evaluations or solutions of any new linear systems. A third-order \( A \)-stable embedded method, denoted ROP3, is given by
\[ c_1 = -\frac{23}{40}, \quad c_2 = -0.246607212432814, \quad c_3 = -\frac{1}{10}, \quad b_1 = \frac{2}{3}, \quad b_2 = 0, \quad b_3 = \frac{1}{3}. \]

In the sequel, this embedded pair will be denoted ROP4(3). To implement ROP4(3) on a computer with at least two processors, note that
\[ \prod_{j=1}^{2} [I - h \gamma_j J_n]^{-1} [I + h c_i J_n] = \sum_{j=1}^{2} u_{ij} [I - h \gamma_j J_n]^{-1}, \]
Consequently, three sweeps over the two processors are needed to compute the update, that is, for \( i = 1, 2, 3, \)

\[
\text{Processor 1: solve } \left[ I - h\gamma_1 J_n \right] \Delta_1 k_i = hf \left( y_n + \sum_{j=1}^{i-1} a_{ij} k_j \right),
\]

\[
\text{Processor 2: solve } \left[ I - h\gamma_2 J_n \right] \Delta_2 k_i = hf \left( y_n + \sum_{j=1}^{i-1} a_{ij} k_j \right),
\]

\[
k_i = u_{i1} \Delta_1 k_i + u_{i2} \Delta_2 k_i,
\]

giving

\[
y_{n+1} = y_n + b_1 k_1 + b_2 k_2 + b_3 k_3.
\]

With

\[
y_{n+1} = y_n + \bar{b}_1 \bar{k}_1 + \bar{b}_2 \bar{k}_2 + \bar{b}_3 \bar{k}_3,
\]

where \( \bar{k}_i = u_{i1} \Delta_1 k_i + u_{i2} \Delta_2 k_i, \) and

\[
\bar{u}_{ij} = \frac{\gamma_j + \bar{c}_i}{\prod_{k=1 \atop k \neq j}^{\infty} (\gamma_j - \gamma_k)},
\]

the difference

\[
\text{err} = y_{n+1} - \bar{y}_{n+1},
\]

between the fourth- and third-order approximations, respectively, to the solution at \( t_{n+1} \) provides a computable estimate of the local truncation error of the third-order formula on which a step size control policy can be based. In summary, on a two processor machine, the implementation of ROP4(3) requires one LU decomposition, three function evaluations, and three backsolves per step on each processor.

The nonautonomous ODE system

\[
y'(t) = f(t, y(t)), \quad y(t_0) = y_0,
\]

can be converted to the autonomous form (1) by adding \( t' = 1 \). If the modified Rosenbrock method (4),(5) is applied to the augmented system, the components corresponding to the \( t \)-variable can be computed explicitly and we arrive at

\[
y_{n+1} = y_n + \sum_{i=1}^{s} b_i k_i,
\]

where

\[
\begin{align*}
\left( I - \gamma_1 h \frac{\partial f}{\partial y} (t_n, y_n) \right) \left( I - \gamma_2 h \frac{\partial f}{\partial y} (t_n, y_n) \right) k_i \\
= h \left( I + c_i h \frac{\partial f}{\partial y} (t_n, y_n) \right) f \left( t_n + a_i h, y_n + \sum_{j=1}^{i-1} a_{ij} k_j \right) \\
+ h^2 \left( c_i + \gamma_1 + \gamma_2 \right) I - \gamma_1 \gamma_2 h \frac{\partial f}{\partial t} (t_n, y_n) \frac{\partial f}{\partial y} (t_n, y_n),
\end{align*}
\]
with the additional coefficients given by
\[
a_i = \sum_{j=1}^{i-1} a_{ij}.
\]  

Consequently, in addition to requiring the Jacobian \( \frac{\partial f}{\partial y} \), the modified Rosenbrock methods (4),(5) also require the vector \( \frac{\partial f}{\partial t} \) as do Rosenbrock methods in the usual form (2) when applied to nonautonomous ODEs (see, for example, [2]). However, parallelism across the method is not affected, so that modified Rosenbrock methods can be implemented on nonautonomous ODE systems in a manner analogous to autonomous systems.

3. NUMERICAL RESULTS

We consider the Robertson chemical kinetics problem [13],
\[
\begin{align*}
y_1 &= -0.04y_1 + 10^4y_2y_3, \\
y_2 &= 0.04y_1 - 10^4y_2y_3 - 3 \times 10^7y_2, \\
y_3 &= 3 \times 10^7y_2,
\end{align*}
\]
y1(0) = 1, y2(0) = 0, y3(0) = 0, to provide an initial investigation into a variable step algorithm. The Rosenbrock pair, ROP4(3), was embedded into the ROS4 suite consisting of six classical Rosenbrock codes wherein each method has three stages and is of order four with an embedded third-order error estimator [2]. The code monitors the local error estimate
\[
\|\text{err}\| = \sqrt{\frac{1}{m} \sum_{j=1}^{m} \frac{\text{err}_j}{\text{sc}_j}},
\]
where
\[
\text{err}_j = y_{n+1,j} - \bar{y}_{n+1,j}, \\
\text{sc}_j = \text{Atol}_j + \max\{|y_{n,j}|, |y_{n+1,j}|\} \text{Rtol}_j,
\]
and uses the step size prediction formula
\[
h_{\text{new}} = h_{\text{old}} \min\left(6, \max\left(0.2, 0.9\|\text{err}\|^{1/4}\right)\right).
\]
Table 1 contains the statistics output from the code using three different values of Rtol, with Atol = 10^{-6} Rtol, over the four intervals defined by x_out = 1, 10, 100, and 1000. The results using ROS42 and ROS43 are also included in Table 1 wherein a count of, for example, 1 DEC for ROP4(3) corresponds to one decomposition on each of the two processors. ROS42 is the default integrator and is representative of the better codes in the ROS4 suite on this problem, while ROS43 gives the worst performance.

Table 1. Robertson problem: Atol = 10^{-6} Rtol.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10^{-2}</td>
<td>ROS42</td>
<td>121</td>
<td>39</td>
<td>41</td>
<td>39</td>
<td>2</td>
<td>41</td>
<td>164</td>
</tr>
<tr>
<td></td>
<td>ROS43</td>
<td>801</td>
<td>267</td>
<td>267</td>
<td>267</td>
<td>0</td>
<td>267</td>
<td>1068</td>
</tr>
<tr>
<td></td>
<td>ROP4(3)</td>
<td>170</td>
<td>54</td>
<td>58</td>
<td>54</td>
<td>4</td>
<td>58</td>
<td>174</td>
</tr>
<tr>
<td>10^{-3}</td>
<td>ROS42</td>
<td>239</td>
<td>75</td>
<td>82</td>
<td>75</td>
<td>7</td>
<td>82</td>
<td>328</td>
</tr>
<tr>
<td></td>
<td>ROS43</td>
<td>1280</td>
<td>426</td>
<td>427</td>
<td>426</td>
<td>1</td>
<td>427</td>
<td>1708</td>
</tr>
<tr>
<td></td>
<td>ROP4(3)</td>
<td>311</td>
<td>101</td>
<td>105</td>
<td>101</td>
<td>4</td>
<td>105</td>
<td>315</td>
</tr>
<tr>
<td>10^{-4}</td>
<td>ROS42</td>
<td>449</td>
<td>131</td>
<td>159</td>
<td>131</td>
<td>28</td>
<td>159</td>
<td>636</td>
</tr>
<tr>
<td></td>
<td>ROS43</td>
<td>1459</td>
<td>485</td>
<td>487</td>
<td>485</td>
<td>2</td>
<td>487</td>
<td>1948</td>
</tr>
<tr>
<td></td>
<td>ROP4(3)</td>
<td>734</td>
<td>238</td>
<td>248</td>
<td>238</td>
<td>10</td>
<td>248</td>
<td>744</td>
</tr>
</tbody>
</table>
These preliminary results indicate that ROP4(3) can be competitive especially at low tolerances. Further study will include the determination of optimal parameters, in particular, to satisfy additional order conditions for differential-algebraic equations as does the six stage Rosenbrock code, RODAS [2], while maintaining only three external stages. This may require the more general form \[ \prod_{j=1}^{p} (I - h\gamma_J J_n), \] of the coefficient matrix of the \(i^{th}\) external stage in (5) which introduces more parameters while exploiting more processors.

REFERENCES