Punjab University
Journal of Mathematics (ISSN 1016-2526)
Vol. 51(6)(2019) pp. 33-51

Efficient Nested Implicit Runge–Kutta Methods for Nonlinear Stiff Problems

Attique Ur-Rehman*[1], Shafiq Ur Rehman[2], Junaid Ahmad[3]

[1] Department of Mathematics, Virtual University of Pakistan, Lahore, Pakistan.
[2,3] Department of Mathematics, University of Engineering and Technology, Lahore, Pakistan.
Email: attique_rehman@vu.edu.pk*[1], srehman@uet.edu.pk[2], junaidshaju@gmail.com[3]

Received: 29 May, 2018 / Accepted: 04 January, 2019 / Published online: 20 March, 2019

Abstract. Gauss-type nested implicit Runge–Kutta methods exhibit many vital properties of implicit Runge–Kutta methods, such as stability, high-order accuracy, and symmetry. Moreover, nested implicit Runge–Kutta methods have a cheap practical implementation in comparison of implicit Runge–Kutta methods because of their explicit nature of internal stages. In this paper, we provide a complete mechanism for the construction, formulation, and implementation of 4^{th}- and 6^{th}-order nested implicit Runge–Kutta methods. The numerical testing is performed using 6^{th}-order nested implicit Runge–Kutta method that uses an embedded 4^{th}-order nested implicit Runge–Kutta method for the local error estimation on a collection of test problems from stiff detest. We also compare the nested implicit Runge–Kutta method with the MATLAB integrator ode15s and assess the effectiveness of four variations on the local error estimation.

AMS (MOS) Subject Classification Codes: 65L04; 65L05; 65L06
Key Words: Nested implicit Runge–Kutta methods, Nonlinear stiff IVPs, Local error estimation.

1. Overview

When dealing with initial value problems (IVPs) of the form

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

where $y_0 \in \mathbb{R}^m$ denote the initial positions, the operator $'$ denotes differentiation with respect to time, m is the dimension of the IVP, and $g : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}^m$ is a sufficiently smooth function, one could use the following one-step numerical scheme:

$$Y_i = y_k + h \sum_{j=1}^{s} a_{ij} g(t_k + c_j h, Y_j), \quad i = 1, 2, ..., s, \tag{1. 2 a}$$

$$y_{k+1} = y_k + h \sum_{i=1}^{s} b_i g(t_k + c_i h, Y_i), \tag{1. 2 b}$$

where $h = t_{k+1} - t_k$ is the step size and Y_i are referred to as the stage values of an s-stage Runge–Kutta method [4]. If $a_{ij} = 0$ for $i \leq j$, i.e., A is a strictly lower triangular matrix, the method (1. 2) becomes an explicit Runge–Kutta (ERK) method. However, if $a_{ij} \neq 0$ for some $i \leq j$ in (1. 2), the method becomes an Implicit Runge–Kutta (IRK) method. There are three important classes of IRK methods based on Gauss, Radau, and Lobatto quadrature formulas. Here, we only focus on Gauss quadrature formulas, because these form the basis of nested implicit Runge–Kutta (NIRK) methods which are at the core of our experimentation in this paper. We refer to [4, 10] for in-depth analysis of methods based on Radau and Lobatto quadrature formulas.

The Gauss methods are based on the Gaussian quadrature formulas in such a way that the abscissae c_j are zeros of the shifted Legendre polynomial of degree s [10]. Butcher [3] and Kuntzmann [9] discovered that the maximum order of an s–stage Gauss Runge–Kutta method is $2s$.

ERK methods, because of their bounded stability domain, are not suitable for solving stiff IVPs. However, IRK methods are a suitable candidate for solving stiff ODEs because they have unbounded stability domains.

The stability of a numerical method is related to the ability of the method to solve stiff ODEs efficiently. A stable numerical solution is the one which remains bounded for a given differential equation having bounded exact solution. To study the stability of a numerical method, Dahlquist in [7] propose to use the following test equation

$$y'(t) = \lambda y(t), \quad \forall \quad \lambda \in \mathbb{C}^-, \tag{1. 3}$$

where $\mathbb{C}^-$ denotes the negative complex half plane.

Many systems of ODEs modelling several physical phenomenons are stiff. These phenomenons include: chemical kinetics, mathematical biology, atmospheric pollution, and control systems. The intuitive meaning of stiff is quite clear yet its correct mathematical definition is controversial [2]. The most practical opinion is: *stiff equations are equations where certain implicit methods perform better than explicit ones* [10].

One good way to recognise a stiff problem is the magnitude of its eigenvalues. The larger the eigenvalue is in the negative half plane, the stiff the problem is. Another characteristic of stiff ODEs is that their solutions have a varying time scale.

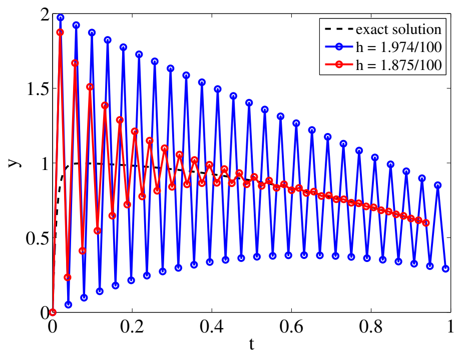
Curtiss and Hirschfelder [6] discussed the term stiff for the first time while dealing with the problems in chemical kinetics. According to them, a differential equation is stiff if the implicit Euler method gives much better performance than the explicit Euler method. They consider the equation

$$\frac{dy}{dt} = \frac{y - F(t)}{a(t, y)}. \tag{1. 4}$$

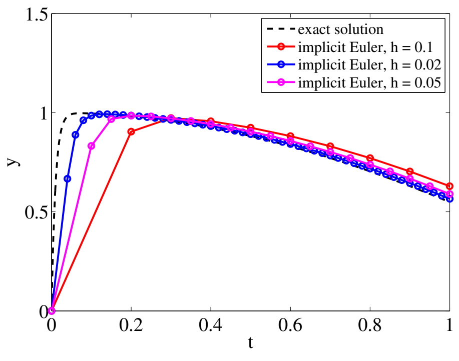
Explicit methods do not work well to solve these kinds of equations because the bounded stability regions of these methods only allow choosing excessively small step sizes. In order to analyse the stability of explicit and implicit Euler methods in relation to the behaviour of exact solution of the test equation (1. 3) we consider

$$y' = -100(y - \cos t), \quad \text{where} \quad y(0) = 0. \tag{1. 5}$$

Solution curves of this problem are shown in Figure 1. The exact solution lies close to $y \approx \cos x$. The numerical solutions obtained by the explicit and implicit Euler methods approach the exact solution after a rapid “transient phase (a region in which the behaviour of coupled equations is examined)". The Figure 1(a) plots the exact solution and the numerical solutions obtained from the explicit Euler method. We have taken two different step sizes $h_1 = 1.974/100$ and $h_2 = 1.875/100$. Both of these solutions are chosen such that $|R(z)| \leq 1$ where $R(z)$ is the stability function of explicit Euler method. We note that the numerical solution oscillates around the exact solution with different magnitude of perturbations depending on the choice of a step size. The Figure 1(b) plots the exact solution and the numerical solutions obtained from the implicit Euler method. Since implicit Euler method is a stable numerical method, we observe in Figure 1(b) that the numerical solution approximates the exact solution effectively for several choices of freely chosen step sizes.



(a)



(b)

FIGURE 1. Plots between t and y for Euler method applied to IVP (1. 5), (a) with explicit Euler method, (b) with implicit Euler method.

In the next section, we provide a mechanism for the construction and implementation of Gauss-type nested implicit Runge–Kutta (NIRK) methods. We also provide local error estimation techniques used in the implementation of NIRK methods. Numerical testing is done in Section 3 involving different classes of nonlinear stiff IVPs and comparison of numerical methods applied to these problems. Finally, Section 4 provides conclusions drawn from the numerical testing performed in Section 3.

2. NESTED IMPLICIT RUNGE–KUTTA METHODS

Kulikov and Shindin presented Nested Implicit Runge–Kutta (NIRK) formulas in [14, 13]. They exploit the idea of MIRK scheme and implemented it efficiently in NIRK scheme of order four. They used a simplified Newton scheme and proved it as effective as modified Newton scheme with exact Jacobian [12]. An s-stage Nested implicit Runge–Kutta method to solve IVP (1. 1) is presented as

$$Y_j^2 = a_{j1}^2 y_k + a_{j2}^2 y_{k+1} + h\Big(d_{j1}^2 g(t_k, y_k) + d_{j2}^2 g(t_{k+1}, y_{k+1})\Big), \quad j = 1, 2, \tag{2. 6 a}$$

$$Y_j^i = a_{j1}^i y_k + a_{j2}^i y_{k+1} + h\Big(d_{j1}^i g(t_k, y_k) + d_{j2}^i g(t_{k+1}, y_{k+1})\Big)$$
$$+ h \sum_{m=1}^{i-1} d_{j,m+2}^i g(T_m^{i-1}, Y_m^{i-1}), \quad i = 3, 4, \ldots, s, \quad j = 1, 2, \ldots, i, \tag{2. 6 b}$$

$$y_{k+1} = y_k + h \sum_{i=1}^{s} b_i g(T_i^s, Y_i^s), \tag{2. 6 c}$$

where $T_j^i = t_k + hc_j^i$ and Y_j^i are the i-th level stage values. For an N-dimensional differential system, the step update formula (2. 6 c) can be expanded as a system of N nonlinear equations as

$$y_{k+1} - y_k - h\Big(b_1 g(T_1^s, Y_1^s) + b_2 g(T_2^s, Y_2^s) + \ldots + b_s g(T_s^s, Y_s^s)\Big) = 0. \tag{2. 7}$$

The formulas (2. 6 a) and (2. 6 b) represent two levels at which the internal stage values are evaluated. For the first level stage values, if two numerical solutions y_k and y_{k+1} and their derivatives y'_k and y'_{k+1} at the grid points t_k and t_{k+1} are available then a cubic Hermite interpolation polynomial can be used in constructing the method of order four [14]. Whereas, for the second level stage values to obtain order six NIRK method, the solutions $y(t_k + c_1^2 h)$ and $y(t_k + c_2^2 h)$ at two fixed points $t_k + c_1^2 h$ and $t_k + c_2^2 h$ are approximated using cubic Hermite interpolation polynomial. Similarly, a quintic Hermite interpolation polynomial is used to approximate the solutions $y(t_k + c_1^3 h)$, $y(t_k + c_2^3 h)$ and $y(t_k + c_3^3 h)$ at three fixed points $t_k + c_1^3 h$, $t_k + c_2^3 h$ and $t_k + c_3^3 h$.

Now we look in to the construction of Gauss-type NIRK methods of order four and order six. The order four NIRK methods are based on Gauss quadrature formulas. Let Y_1^2 and Y_2^2 be the exact solutions of IVP (1. 1) at two internal points $t_k + c_1^2 h$ and $t_k + c_2^2 h$ within integration interval $[t_k, t_{k+1}]$, respectively. These internal stage values are evaluated using an initial guess of the value of $y(t_{k+1})$. The NIRK methods of order four have the form

$$Y_1^2 = a_{11}^2 y_k + a_{12}^2 y_{k+1} + h\Big(d_{11}^2 g(t_k, y_k) + d_{12}^2 g(t_{k+1}, y_{k+1})\Big), \tag{2. 8 a}$$

$$Y_2^2 = a_{21}^2 y_k + a_{22}^2 y_{k+1} + h\Big(d_{21}^2 g(t_k, y_k) + d_{22}^2 g(t_{k+1}, y_{k+1})\Big), \tag{2. 8 b}$$

$$y_{k+1} = y_k + hh\Big(b_1 g(T_1^2, Y_1^2) + b_2 g(T_2^2, Y_2^2)\Big), \tag{2. 8 c}$$

where $T_1^2 = t_k + c_1^2 h$, $T_2^2 = t_k + c_2^2 h$ and b_i, c_i^2 represent the nodes and weights of Gauss quadrature formulas, respectively. These are given as

$$\begin{aligned} b_1 &= \frac{1}{2}, \quad b_2 = \frac{1}{2}, \\ c_1^2 &= \frac{1}{2} - \frac{\sqrt{3}}{6}, \quad c_2^2 = \frac{1}{2} + \frac{\sqrt{3}}{6}. \end{aligned} \tag{2. 9}$$

To ensure the NIRK method (2. 8) is of order four, we need to show that its local error is $O(h^5)$. This condition can be achieved by the following coefficients as shown in [14] with

θ as a free real parameter.

$$\begin{aligned} &a^2_{11} = \theta, \quad a^2_{12} = 1-\theta, \quad a^2_{21} = 1-\theta, \quad a^2_{22} = \theta, \\ &d^2_{11} = \frac{1}{2}\theta - \frac{1}{6} - \frac{\sqrt{3}}{12}, \quad d^2_{12} = \frac{1}{2}\theta - \frac{1}{3} - \frac{\sqrt{3}}{12}, \\ &d^2_{21} = \frac{1}{3} + \frac{\sqrt{3}}{12} - \frac{1}{2}\theta, \quad d^2_{22} = \frac{1}{6} + \frac{\sqrt{3}}{12} - \frac{1}{2}\theta. \end{aligned} \tag{2. 10}$$

The primary role of θ is to change the coefficients in the coefficient matrix of the Butcher tableau. The above coefficients (2. 10) are calculated by using Taylor expansion for local error evaluation of NIRK methods (2. 8). The NIRK methods (2. 8) with their coefficients (2. 10) can be represented as RK methods. The Butcher tableau for these methods is shown in Table 1.

0	0	0	0	0
c^2_1	$\frac{1}{2}c^2_1 + \frac{1}{2}\theta - \frac{5}{12}$	$\frac{1}{2} - \frac{1}{2}\theta$	$\frac{1}{2} - \frac{1}{2}\theta$	$\frac{1}{2}c^2_1 + \frac{1}{12}\theta - \frac{7}{12}$
$1 - c^2_1$	$\frac{7}{12} - \frac{1}{2}c^2_1 - \frac{1}{12}\theta$	$\frac{1}{2}\theta$	$\frac{1}{2}\theta$	$\frac{5}{12} - \frac{1}{2}c^2_1 - \frac{1}{12}\theta$
1	0	$\frac{1}{2}$	$\frac{1}{2}$	0
	0	$\frac{1}{2}$	$\frac{1}{2}$	0

TABLE 1. Butcher tableau for Gauss–type NIRK methods (2. 8) with their coefficients (2. 10).

The choice of the free parameter θ leads to different NIRK methods of order four. However, different choices of the parameter θ leads to methods with different stage orders. Thus if we take $\theta = 1/2 + 2\sqrt{3}/9$, we get NIRK method of order four and stage order three whereas for all other θ values we get stage order two [13].

For the construction of order six NIRK methods, 3-rd level stage values Y^3_1, Y^3_2 and Y^3_3 are evaluated in addition to 2-nd level stage values. These 3-rd level stage values are calculated at three further internal points $t_k + c^3_1 h$, $t_k + c^3_2 h$ and $t_k + c^3_2 h$ within $[t_k, t_{k+1}]$, respectively. Also the 2-nd level stage values are used for the evaluation of 3-rd level stage values. The NIRK methods of order six have the form

$$Y^2_j = a^2_{j1} y_k + a^2_{j2} y_{k+1} + h\Big(d^2_{j1} g(t_k, y_k) + d^2_{j2} g(t_{k+1}, y_{k+1})\Big), \quad j = 1, 2, \tag{2. 11 a}$$

$$\begin{aligned} Y^3_j = {} & a^3_{j1} y_k + a^3_{j2} y_{k+1} + h\Big(d^3_{j1} g(t_k, y_k) + d^3_{j2} g(t_{k+1}, y_{k+1})\Big) \\ & + h\Big(d^3_{j3} g(T^2_1, Y^2_1) + d^3_{j3} g(T^2_2, Y^2_2)\Big), \quad j = 1, 2, 3, \end{aligned} \tag{2. 11 b}$$

$$y_{k+1} = y_k + h\Big(b_1 g(T^3_1, Y^3_1) + b_2 g(T^3_2, Y^3_2) + b_3 g(T^3_3, Y^3_3)\Big), \tag{2. 11 c}$$

where $T^2_j = t_k + hc^2_j$ and $T^3_j = t_k + hc^3_j$.

To evaluate the coefficients in the above formulas (2. 11), Kulikov and Shindin, in [13, 11], used a RK scheme with the following Butcher tableau

$$
\begin{array}{c|ccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
c_2 & a_{21} & 0 & 0 & a_{24} & a_{25} & a_{26} & a_{27} \\
c_3 & a_{31} & 0 & 0 & a_{34} & a_{35} & a_{36} & a_{37} \\
c_4 & a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} & a_{47} \\
c_5 & a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} & a_{57} \\
c_6 & a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66} & a_{67} \\
1 & 0 & 0 & 0 & b_4 & b_5 & b_6 & 0 \\
\hline
 & 0 & 0 & 0 & b_4 & b_5 & b_6 & 0
\end{array}
$$

TABLE 2. Butcher tableau, which was used to evaluate the coefficients of order 6 Gauss–type NIRK methods.

The following conditions are required to be satisfied by the coefficients of the Table 2 as stated in [13]:

(1) The coefficients c_2 and c_3 are distinct roots of 2^{nd} degree Legendre polynomial

$$\frac{d^2}{dt^2}(t^2(1-t)^2) = 0. \tag{2. 12 a}$$

(2) The coefficients c_4, c_5 and c_6 are distinct roots of 3^{rd} degree Legendre polynomial

$$\frac{d^3}{dt^3}(t^3(1-t)^3) = 0. \tag{2. 12 b}$$

(3) The coefficients b_4, b_5 and b_6 represent the weights of the order six Gauss formula.

It can be seen that the first condition embeds NIRK methods ((2. 8) and (2. 10)) into a new one and the other two conditions assume that the method has order six. Following conditions are needed to be satisfied as given in [13]

$$a_{ij} = a_{i2}^2 b_j, \quad i = 2,3, j = 4,5,6,$$
$$a_{ij} = a_{i2}^3 b_j, \quad i = 4,5,6, j = 4,5,6.$$

All other free coefficients in Table 2 are evaluated by assuming that stage order is three for 2-nd level stage values and stage order is five for 3-rd level stage values. This is obtained by solving the following system. The coefficient matrices in these system, as suggested by the conditions (2. 12), are non-singular.

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1/2 & 1 \\ 0 & 1/3 & 1 \end{bmatrix} \begin{bmatrix} a_{i1} \\ a_{i2}^2 \\ a_{i7} \end{bmatrix} = \begin{bmatrix} c_i \\ c_i^2/2 \\ c_i^3/3 \end{bmatrix}, \quad i = 2,3, \tag{2. 13 a}$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & c_2 & c_3 & 1/2 & 1 \\ 0 & c_2^2 & c_3^2 & 1/3 & 1 \\ 0 & c_2^3 & c_3^3 & 1/4 & 1 \\ 0 & c_2^4 & c_3^4 & 1/5 & 1 \end{bmatrix} \begin{bmatrix} a_{i1} \\ a_{i2} \\ a_{i3} \\ a_{i2}^3 \\ a_{i7} \end{bmatrix} = \begin{bmatrix} c_i \\ c_i^2/2 \\ c_i^3/3 \\ c_i^4/4 \\ c_i^5/5 \end{bmatrix}, \quad i = 4,5,6. \tag{2. 13 b}$$

Theorem 1. [13] The Gauss–type NIRK method with Butcher tableau given in Table 3 is of classical order six and stage order three.

The coefficients of the NIRK methods of order six are

0	0	0	0	0	0	0	0
c_2	$\frac{c_3}{6}$	0	0	$\frac{35}{108}-\frac{10c_3}{27}$	$\frac{14}{27}-\frac{16c_3}{27}$	$\frac{35}{108}-\frac{10c_3}{27}$	$\frac{c_3}{6}-\frac{1}{6}$
c_3	$\frac{c_2}{6}$	0	0	$\frac{35}{108}-\frac{10c_2}{27}$	$\frac{14}{27}-\frac{16}{27}c_2$	$\frac{35}{108}-\frac{10c_2}{27}$	$\frac{c_2}{6}-\frac{1}{6}$
c_4	$\frac{c_6}{10}-\frac{3}{200}$	$\frac{9c_3}{50}-\frac{9}{100}+\theta$	θ	$\frac{v_1}{360}-\frac{5\theta}{9}$	$\frac{v_1}{225}-\frac{8\theta}{9}$	$\frac{v_1}{360}-\frac{5\theta}{9}$	$\frac{3}{200}-\frac{c_4}{10}$
c_5	$\frac{1}{32}$	$\frac{\sqrt{27}}{32}$	$-\frac{\sqrt{27}}{32}$	$\frac{5}{36}$	$\frac{2}{9}$	$\frac{5}{36}$	$-\frac{1}{32}$
c_6	$\frac{c_4}{10}-\frac{3}{200}$	$-\theta$	$\frac{9c_2}{50}-\frac{9}{100}-\theta$	$\frac{v_2}{360}+\frac{5\theta}{9}$	$\frac{v_2}{225}+\frac{8\theta}{9}$	$\frac{v_2}{360}+\frac{5\theta}{9}$	$\frac{3}{200}-\frac{c_6}{10}$
1	0	0	0	$\frac{5}{18}$	$\frac{4}{9}$	$\frac{5}{18}$	0
	0	0	0	$\frac{5}{18}$	$\frac{4}{9}$	$\frac{5}{18}$	0

TABLE 3. Butcher tableau presenting coefficients of order six NIRK methods.

where

$$\begin{aligned} c_2 &= \frac{1}{2}-\frac{\sqrt{3}}{6}, \quad c_3 = \frac{1}{2}+\frac{\sqrt{3}}{6}, \\ c_4 &= \frac{1}{2}-\frac{\sqrt{15}}{10}, \quad c_5 = \frac{1}{2}, \quad c_6 = \frac{1}{2}+\frac{\sqrt{15}}{6}, \\ \theta &= \frac{9c_6}{50}-\frac{9c_3}{100}-\frac{9}{200}, \\ v_1 &= 120c_4-18c_3-1, \quad v_2 = 120c_6-18c_2-1. \end{aligned} \tag{2. 14}$$

Implementation of Gauss-type NIRK Methods: NIRK methods are effective and efficient methods due to their lower computational cost. In order to implement NIRK methods, we first approximate the exact solution y_{k+1} at t_{k+1} then use this approximation to calculate the stage values ((2. 11 a) and (2. 11 b)). The step update formula (2. 11 c) becomes a single nonlinear equation to be solved. Method (2. 11) can be represented in a convenient form to perform the iterative scheme to solve this nonlinear problem. For given values of θ_1 and θ_2, method (2. 11) is equivalent to NIRK method given in Table 3. The coefficients of both levels of stage values used in [11] are

- For 2-nd level stage values, where $\theta_1 = \frac{1}{2}+\frac{2\sqrt{3}}{9}$,

$$\begin{aligned} &a^2_{11} = \theta_1, \quad a^2_{12} = 1-\theta_1, \quad a^2_{21} = 1-\theta_1, \quad a^2_{22} = \theta_1, \\ &d^2_{11} = \frac{\theta_1}{2}-\frac{1}{6}-\frac{\sqrt{3}}{12}, \quad d^2_{12} = \frac{\theta_1}{2}-\frac{1}{3}-\frac{\sqrt{3}}{12}, \\ &d^2_{21} = \frac{1}{3}+\frac{\sqrt{3}}{12}-\frac{\theta_1}{2}, \quad d^2_{22} = \frac{1}{6}+\frac{\sqrt{3}}{12}-\frac{\theta_1}{2}. \end{aligned} \tag{2. 15}$$

- For 3-rd level stage values, where $\theta_2 = \frac{9\sqrt{15}}{200}-\frac{\sqrt{27}}{200}$

$$a^3_{11} = \frac{1}{2} + \frac{3\sqrt{15}}{25} + \frac{\sqrt{27}}{100} + 2\theta_2, \quad a^3_{12} = \frac{1}{2} - \frac{3\sqrt{15}}{25} - \frac{\sqrt{27}}{100} - 2\theta_2, \quad a^3_{21} = \frac{1}{2},$$
$$a^3_{22} = \frac{1}{2}, \quad a^3_{31} = \frac{1}{2} - \frac{3\sqrt{15}}{25} - \frac{\sqrt{27}}{100} - 2\theta_2, \quad a^3_{32} = \frac{1}{2} + \frac{3\sqrt{15}}{25} + \frac{\sqrt{27}}{100} + 2\theta_2,$$
$$d^3_{11} = \frac{7}{200} + \frac{\sqrt{15}}{100}, \quad d^3_{12} = \theta_2 + \frac{\sqrt{27}}{100}, \quad d^3_{13} = \theta_2, \quad d^3_{14} = -\frac{7}{200} + \frac{\sqrt{15}}{100},$$
$$d^3_{21} = \frac{1}{32}, \quad d^3_{22} = \frac{\sqrt{27}}{32}, \quad d^3_{23} = -\frac{\sqrt{27}}{32}, \quad d^3_{24} = -\frac{1}{32},$$
$$d^3_{31} = \frac{7}{200} - \frac{\sqrt{15}}{100}, \quad d^3_{32} = -\theta_2, \quad d^3_{33} = -\theta_2 - \frac{\sqrt{27}}{100}, \quad d^3_{34} = -\frac{7}{200} - \frac{\sqrt{15}}{100}. \tag{2. 16}$$

Generally, for stiff problems, Newton's iterative scheme is more efficient than the fixed point iterative scheme. Kulikov presented a modified Newton scheme in [11] and is given by

$$Y^{2,(l)}_j = a^2_{j1}\bar{y}_k + a^2_{j2}y^{(l-1)}_{k+1} + h\Big(d^2_{j1}g(t_k,\bar{y}_k) + d^2_{j2}g(t_{k+1},y^{(l-1)}_{k+1})\Big), \quad j = 1,2, \tag{2. 17 a}$$

$$Y^{3,(l)}_j = a^3_{j1}\bar{y}_k + a^3_{j2}y^{(l-1)}_{k+1} + h\Big(d^3_{j1}g(t_k,\bar{y}_k) + d^3_{j2}g(t_{k+1},y^{(l-1)}_{k+1})\Big)$$
$$+ h\sum_{m=1}^{2} d^3_{j,m+2}g(T^2_m, Y^{2,(l-1)}_m), \quad j = 1,2,3, \tag{2. 17 b}$$

$$U(hJ)(y^{(l)}_{k+1} - y^{(l-1)}_{k+1}) = -y^{(l-1)}_{k+1} + \bar{y}_k + h\sum_{i=1}^{s} b_i g(T^s_i, Y^{s,(l)}_i), \quad l = 1,2,\ldots,N, \tag{2. 17 c}$$

where $J \overset{\text{def}}{=} \partial_t f(t_{k+1}, y^{(0)}_{k+1})$ represents the Jacobian of the function provided in IVP (1. 1) and $\bar{y}_k = y^M_k, k = 0,1,\ldots,N-1$ is the numerical solution obtained by method (2. 6) using M iterations of (2. 17) per step.

A major constraint for iteration (2. 17) is its practical efficiency because the coefficient matrix $U(hJ)$ in (2. 17 c) is a matrix-valued cubic polynomial of the form

$$U(hJ) \overset{\text{def}}{=} I_m - \frac{1}{2}hJ + \frac{1}{10}(hJ)^2 - \frac{1}{20}(hJ)^3, \tag{2. 18}$$

where I_m is an $m \times m$ identity matrix. The evaluation of the coefficient matrix U as in (2. 18) is impractical due to its large computational cost. One way to overcome this is to use its linear part and ignore the higher degree terms. However, this approach does not reduce the CPU time considerably and damages the stability of MIRK methods including NIRK formulas [5, 12, 14]. Cash and Singhal, in [5], suggested to approximate the coefficient matrix U by some power equal to the degree of the original matrix-valued polynomial (2. 18). Kulikov approximated it for NIRK methods [11] as

$$\widetilde{U}(hJ) \overset{\text{def}}{=} \Big(I_m - \frac{1}{6}hJ\Big)^3. \tag{2. 19}$$

The above polynomial has many advantages. Firstly, it is effectively utilized in iterative scheme (2. 17). The matrix $I_m - hJ/6$ is decomposed once per integration step. The three linear systems are then solved with the decomposed matrix. So, this uses reasonably less computational cost. Secondly, the polynomial (2. 19) is an approximation of order 2, that is $U(hJ) = \widetilde{U}(hJ) + O(h^2J^2)$. So, replacement of $U(hJ)$ with $\widetilde{U}(hJ)$ has no negative influence on the convergence rate of the iteration.

Error Estimation Techniques and Automatic Step Size Control: The local error estimation in a RK method leads to automatic adjustment of the step size and is controlled by some prescribed tolerance (TOL). With an initial step size, an embedded RK pair approximates two solutions y_1 and $\widehat{y}_1$ of an IVP. Their difference $y_1 - \widehat{y}_1$ computes the local error (le) which is then compared with the given tolerance. If $\|le\|$ is greater than TOL, we take it as a rejected step and reduce the stepsize using the formula below and repeat the step.

$$h_{new} = h\Big(\frac{\text{TOL}}{\|\text{le}\|}\Big)^{1/p} \times 0.9, \tag{2. 20}$$

where p is the order of the numerical method. If the step is accepted then the step size is increased also by using same relation (2. 20). The choice of an initial step size is also important. Though a bad choice is repaired by the step size control, yet it utilizes more CPU time.

For a Gauss–type NIRK method, a pair of embedded RK formulas can be formed by order four and order six methods [13]. Both of these methods are applied to obtain two numerical solutions of the given IVP, and the difference of both numerical solutions estimates the error. In this way, a family of order six Gauss-type methods with built-in error estimation is achieved [11]. Kulikov and Shindin introduced an embedded stage error estimation technique in [13, 12] and tested it successfully on order four NIRK methods. They termed it as *Embedded Stages Approach (ESA)*. It is based on the result attained in [14], that the stage order of the NIRK methods is raised by one when a specific value of the free parameter θ is used [11, 14].

For order four and order six Gauss-type NIRK methods, four different error estimation techniques are examined numerically in [12] and [11], respectively. Here, we discuss these techniques for the order six NIRK methods.

1) The embedded method error estimation (EMEE) scheme estimates the local error by comparing the two numerical solutions obtained from embedded NIRK method of order four given in Table 1 and order six NIRK methods given in Table 3 respectively and are as follows

$$y_{k+1} = y_k + h\Big(\frac{1}{2}g(T_1^2, Y_1^2) + \frac{1}{2}g(T_2^2, Y_2^2)\Big), \tag{2. 21 a}$$

$$y_{k+1} = y_k + h\Big(\frac{5}{18}g(T_1^3, Y_1^3) + \frac{4}{9}g(T_2^3, Y_2^3) + \frac{5}{18}g(T_3^3, Y_3^3)\Big), \tag{2. 21 b}$$

so the local error can be found as

$$le_{k+1} = y_{k+1} - \widehat{y}_{k+1},$$

$$le_{k+1} = h\Big(\frac{1}{2}g(T_1^2, Y_1^2) + \frac{1}{2}g(T_2^2, Y_2^2) - \frac{5}{18}g(T_1^3, Y_1^3) - \frac{4}{9}f(T_2^3, Y_2^3) - \frac{5}{18}f(T_3^3, Y_3^3)\Big). \tag{2. 22}$$

This error estimation approach is quite simple and cheap as all $f(T_j^i, Y_j^i)$ are already evaluated within every step of method (2. 11). The stability function for embedded NIRK method of order four is calculated as

$$R_{ENIRK4}(z) = \frac{1 + 1/2z + 1/10z^2 + 1/120z^3 - 1/750z^5}{1 - 1/2z + 1/10z^2 - 1/120z^3}, \quad \text{where } z = h\lambda. \tag{2. 23}$$

which is not bounded within complex plane and is not A-stable as well [11]. So EMEE approach (2. 22) can put unnecessary restriction on the step sizes of the NIRK method ((2. 11) and Table 3), when dealing with very stiff IVPs. It will result in extra computational time that is not affordable for some large-scale problems.

2) The second scheme is called as modified embedded method error estimation (MEMEE) scheme. It might be time consuming to implement directly the EMEE for some large-scale differential equations. This issue can be resolved by applying Shampine's idea [10] to modify the EMEE technique. The Shampine's idea is to modify the error estimation technique for an embedded method, especially for stiff equations in which the embedded error becomes unbounded. The resulting MEMEE technique is suitable for stiff ODEs. The error estimation technique EMEE is modified and computed by solving a linear system as follows:

$$\widetilde{U}_1 \widetilde{le}_{k+1} = le_{k+1}, \tag{2. 24}$$

where le_{k+1} is same as calculated in (2. 22), $\widetilde{U}_1 = (I_m - hJ/6)^2$. This MEMEE technique is not practically expensive as it uses solutions of two linear systems with same coefficient matrices $I_m - hJ/6$. This matrix is already decomposed within the step of the numerical method ((2. 11) and Table 3), so no extra work is required. We compute the new stability function that follows from MEMEE technique using the formula

$$\widetilde{R}_{ENIRK4}(z) = R_{NIRK6}(z) + \widetilde{Q}_1^{-1}(R_{ENIRK4}(z) - R_{NIRK6}(z)), \tag{2. 25}$$

where $R_{NIRK6}(z)$ represents the stability function of order six NIRK method. Solving formulas (2. 23) and (2. 25), we get the stability function as

$$R_{ENIRK4}(z) = \frac{1 + 1/6z - 7/180z^2 - 1/90z^3 - 1/864z^5}{1 - 5/6z + 53/180z^2 - 1/18z^3 + 1/180z^4 - 1/4320z^5}. \tag{2. 26}$$

As the above stability function (2. 26) is bounded within the complex plane so is the error estimate (2. 24). Thus MEMEE is suitable for stiff ODEs.

3) The embedded stage error estimation (ESEE) technique is successfully tested for order 4 NIRK methods in [13, 12]. This idea is extended to order six NIRK methods. The stage values for methods ((2. 11) and Table 3) approximate an exact solution of order 6 when the value of free parameter θ is used as $(36c_6 - 18c_3 - 9)/200$. For any other θ these stage values approximate order five numerical solution. The difference of the two solutions provides an order five approximation. The ESEE technique can be represented by the following formula:

$$\widehat{le}_{k+1} = rh\Big(\frac{1}{2}g(T_1^2, Y_1^2) + \frac{1}{2}g(T_2^2, Y_2^2) - \frac{5}{18}g(T_1^3, Y_1^3) - \frac{4}{9}g(T_2^3, Y_2^3) - \frac{5}{18}g(T_3^3, Y_3^3)\Big), \tag{2. 27}$$

where $r = \widehat{\theta} - \theta$.

4) Similar to EMEE, ESEE cannot be much effective for stiff problems for any r. So, we again use Shampine's idea and modify the ESEE technique as follows:

$$\widetilde{U}_1 \bar{le}_{k+1} = \widehat{le}_{k+1}, \tag{2. 28}$$

where $\widetilde{U}_1$ and $\widehat{le}_{k+1}$ are the same as calculated in MEMEE and (2. 27), respectively. This modified embedded stage error estimation (MESEE) technique is bounded for any step size with $0 < |r| \leq 1/6$ and is cheaper as well [11].

3. Numerical testing

In this section, we apply the order six Nested Implicit Runge–Kutta (NIRK) method on a variety of nonlinear stiff IVPs. These IVPS are from the stiff DETEST problems [8] and are given in the Appendix. Since the exact solutions of the test problems are not available, we compared our numerical solutions with reference solutions. These reference solutions were computed using highly accurate MATLAB integrator. The test problems are from the following classes:

- Problem class D (nonlinear with real eigenvalues)- Six different systems modeling practical applications constitute this class.
- Problem class E (nonlinear with complex eigenvalues)- This class has five systems, four of which have eigenvalues close to real axis.

We apply two numerical methods. The first method is Gauss-type NIRK method of order six with built-in error estimation. The second method is highly accurate MATLAB integrator ode15s [15]. The pseudo-code we used for our driver is provided and explained in [16].

The order six NIRK method is applied to the stiff DETEST problems. NIRK methods are embedded methods with built-in error estimation. There are four techniques of this error estimation, namely, EMEE, MEMEE, ESEE, and MESEE. These techniques were studied in detail in Section 2.

We have used modified Newton's method to solve stages iteratively. We allow a maximum of three to four iterations for convergence. The iterative scheme uses a coefficient matrix U as given in Section 2. We take two different values of the matrix U as given in (2. 18) and (2. 19). The convergence of our iterative scheme is bounded by a small positive number which is 10^{-2} times local error tolerance. We have checked it with other values as well, but we did not find any significant difference. The local error tolerance (TOL) is taken as $10^{-i}, i = 2, 3, \ldots, 10$. This variation is used to perform different experiments. We evaluate an exact Jacobian and check our results by providing it in every step or every iteration. Although we have experimented with all the problems of both classes, we have only included few important results here.

Experiment 1

In our first set of experiments, we check the accuracy of order six NIRK methods in terms of end-point global error. Figures 2 – 3 show the graphs of the end-point global errors (ERROR) computed at various tolerances (TOL) for the two classes of stiff DETEST problems.

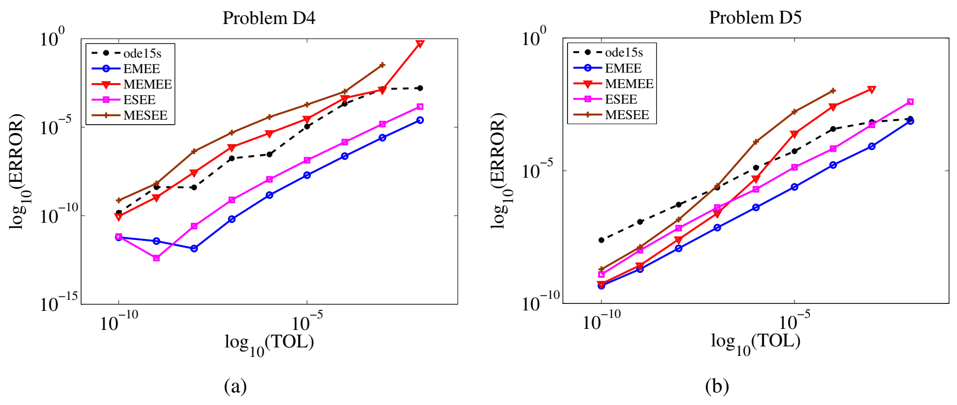


FIGURE 2. Experiment 1d, tolerance vs end-point global error for NIRK and ode15s, (a) = with $\widetilde{U}$ and mitn = 3, (b) = with U and mitn = 3.

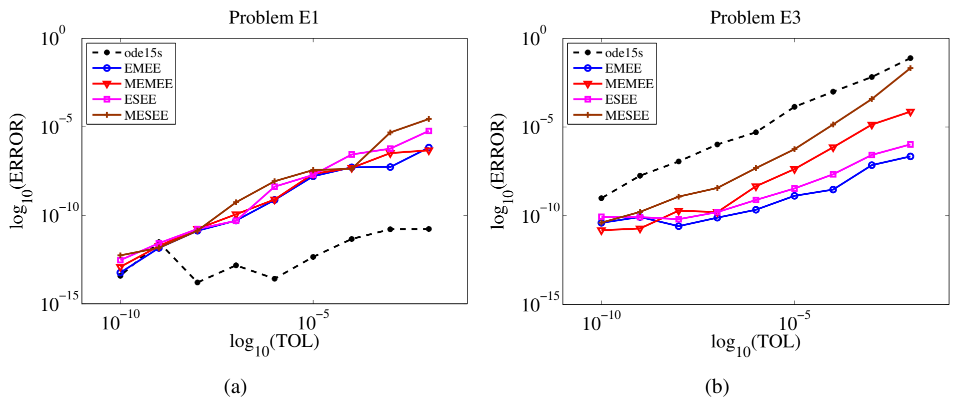


FIGURE 3. Experiment 1e, tolerance vs end-point global error for NIRK and ode15s, (a) = with U and mitn = 3, (b) = with $\widetilde{U}$ and mitn = 3.

For the problem class D (nonlinear with real eigenvalues), NIRK method shows more accuracy than ode15s at all tolerances only for problem D1. For problems D2 and D3, ode15s scheme performs relatively better at larger tolerances. In case of problem D5 as shown in Figure 2(b), ode15s is more accurate than EMEE and ESEE at higher tolerances, but at lower ones, the former scheme is less accurate than all error estimation techniques. Like previous problem classes, EMEE scheme performs at the best accuracy level than other ones in many tests. However, unlike previous results, ESEE becomes more accurate than MEMEE for this problem class as given in Figure 2(a). In case of problem D6, ode15s is more accurate than MEMEE and MESEE techniques but less than the other two. Also in this problem class, there is almost no effect whether U or $\widetilde{U}$ is used as the coefficient matrix in the iterative scheme.

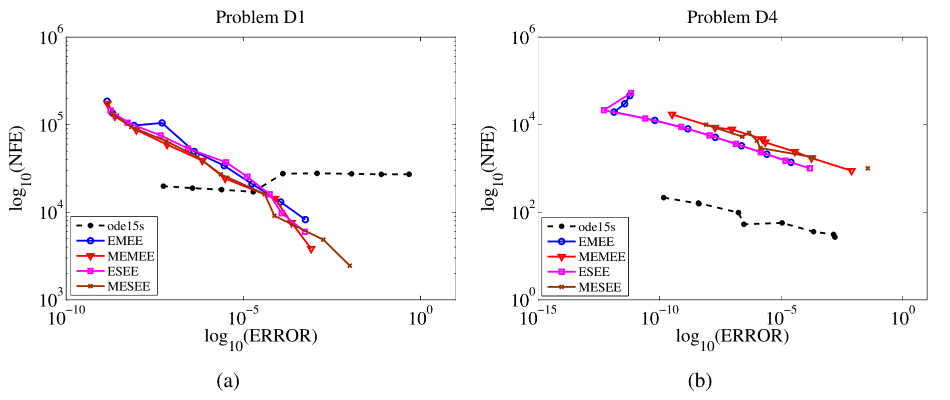


FIGURE 4. Experiment 2d, end-point global error vs function evaluation for NIRK and ode15s, (a) = with U and mitn = 3, (b) = with $\widetilde{U}$ and mitn = 4.

For the problem class E (nonlinear with non-real eigenvalues), NIRK method performs well for problems E3 and E4 as compared to the ode15s method. However, the latter method exhibits quite more accuracy in solving problem E1 as shown in Figure 3(a). For different error estimation techniques, EMEE is the most accurate and MESEE is the least one. Moreover, MEMEE technique is better than ESEE in all problems of this class except in E3, whereas MEMEE is more accurate only at a few lower tolerances. Figure 3(b) shows this result. Like problem class D, here again, there is no effect for the choice of U or $\widetilde{U}$ in our iterative scheme.

EXPERIMENT 2

One way to measure the efficiency of the NIRK scheme is to count the number of function evaluations (NFE) against the end-point global errors (ERROR), computed at different tolerances $(10^{-i}, i = 2, 3, \ldots, 10.)$. The results for the second set of experiments are displayed in Figures 4 – 5. The initial values are same as in the case of the first set of experiments.

For the problem class D, the problem D1 exhibits an unexpected result relative to other problems of this set. The integrator od15s uses more function evaluations at larger tolerances and lesser at the smaller ones in comparison with NIRK method as shown in Figure 4(a). On the other hand, NIRK method indicates poor efficiency in case of problems D4 and D6 where it uses 20 times more function evaluations than ode15s as shown in Figure 4(b). Note it that, any variation in the choice of U or $\widetilde{U}$ and choice of the maximum allowed iterations do not affect this result. Similarly, for problem D2, ode15s is a lot more efficient than NIRK scheme, though lesser as compared to problem D4. Finally, NIRK is found to be very efficient than ode15s for problem D5. The four error estimate techniques use nearly the same number of function evaluations except in problem D3, where EMEE and ESEE use relatively lesser than other two.

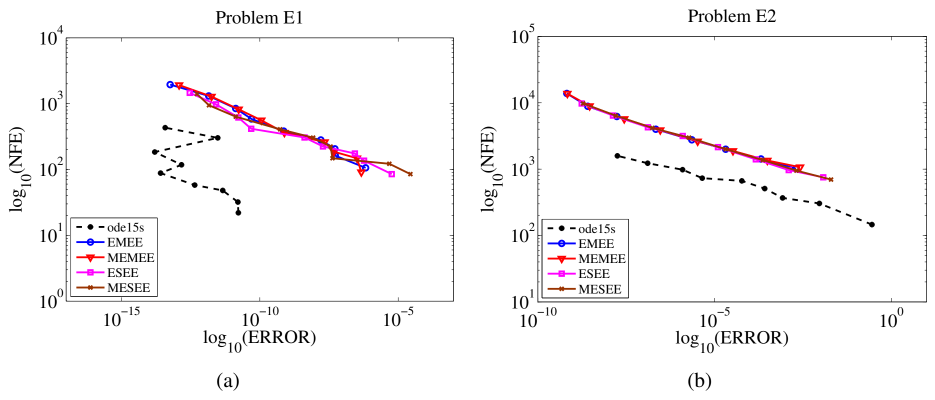


FIGURE 5. Experiment 2e, end-point global error vs function evaluation for NIRK and ode15s, (a) $=$ with U and mitn = 3, (b) $=$ with $\widetilde{U}$ and mitn = 3.

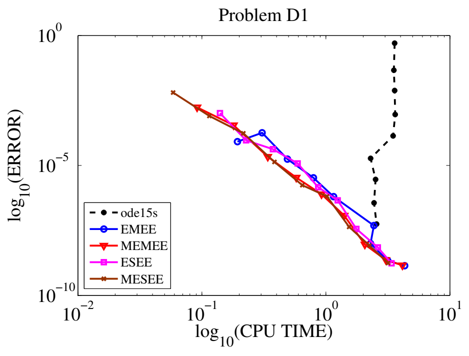
For the problem class E, ode15s is comparatively more efficient than NIRK. The lowest difference in function evaluations between both schemes is about two times as in case of problem E2 (Figure 5(b)), and the highest one is about 30 times which is in problem E3. In problem E1, the behaviour of the ode15s scheme is different than other problems due to the irregular behaviour of values of global errors. Also in this problem, the four error estimation techniques show slightly different efficiency with one another as shown in Figure 5(a).

EXPERIMENT 3

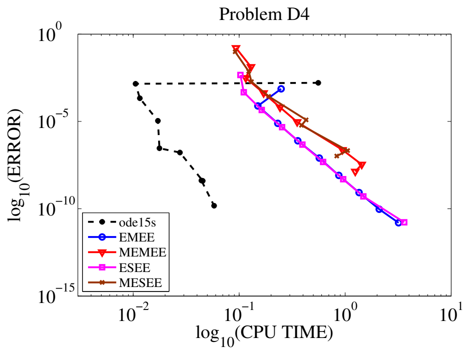
Finally, we conduct a set of experiments to compute the computational time (CPU-time) and end-point global errors (ERROR) at nine different tolerances ($10^{-i}, i = 2, 3, \ldots, 10.$). The results of this set of experiments give a reasonable estimate of the efficiency of NIRK methods. Some of the results are shown in Figures 6 – 7.

For the problem class D, NIRK method uses the lesser computational time for problems D1 and D5 and more computational time for problems D2 and D4 as compared to the ode15s scheme. The results of problems D1 and D4 are shown in Figures 6(a) and 6(b), respectively. In problem D4, the behaviour at the tolerance of 10^{-2} is similar as seen in previous problem classes. However, error estimate techniques EMEE and ESEE performed better than the other two. In case of problem D3, with $\widetilde{U}$ as the coefficient matrix, ode15s scheme takes lesser CPU time than NIRK method for MEMEE and MESEE techniques, but the other two; EMEE and ESEE give better results than ode15s scheme. When U is used, all the error estimation techniques exhibit nearly same results among themselves as well as compared to ode15s scheme.

For the problem class E, NIRK method uses lesser CPU time than the ode15s scheme for problem E2 only when the maximum allowed iterations are chosen as three. If we allow four iterations, then both schemes took almost same time as shown in Figure 7(a). For problem E1, at smaller tolerances, NIRK method performs better than the ode15s scheme.

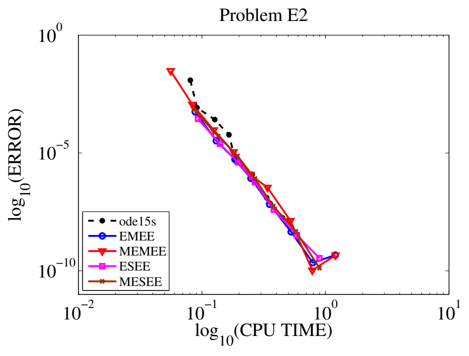


(a)

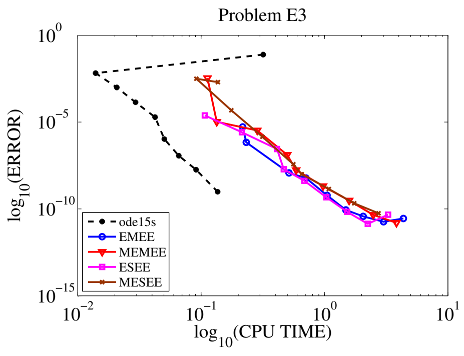


(b)

FIGURE 6. Experiment 3d, CPU time vs end-point global error for NIRK and ode15s, (a) = with U and mitn = 3, (b) = with $\widetilde{U}$ and mitn = 4.



(a)



(b)

FIGURE 7. Experiment 3e, CPU time vs end-point global error for NIRK and ode15s, (a) = with U and mitn = 4, (b) = with $\widetilde{U}$ and mitn = 3.

Moreover, MEMEE and ESEE techniques use relatively lesser computational time than the other two. Figure 7(b) shows the result of problem E3. The problems E4 and E5 exhibit better performance of ode15s scheme as compared to NIRK method. It has been observed that the choice of U or $\widetilde{U}$ does not affect while the choice of Jacobian evaluation affects the results in a similar way as in previous problem classes.

4. Conclusions

Gauss-type NIRK methods have recently been introduced by G. Yu. Kulikov and S. K. Shindin [12, 13, 11]. These methods have not only cheap practical implementation but also exhibit many essential properties of implicit Runge–Kutta (IRK) methods, such as stability, high-order accuracy, and symmetry. In this paper, we have provided an implementation of 4^{th}- and 6^{th}-order NIRK methods. The NIRK methods have built-in local error estimates, and we have examined four different error estimation techniques given in Section 2. The numerical testing is performed on a variety of nonlinear stiff problems from stiff DETEST with both real and complex eigenvalues. As a part of numerical testing, we performed experiments to compare the accuracy and efficiency of NIRK method with the MATLAB integrator ode15s. Different sets of experiments were performed for nonlinear stiff problems with real and with complex eigenvalues at different tolerances.

It has been observed that when nonlinear problems with real eigenvalues are tested, the NIRK method exhibited good accuracy than ODE15s scheme. However, two error estimation techniques of NIRK method are relatively less accurate in case of some problems. Whereas, the function evaluations of NIRK method are considerably larger than the ode15s scheme for most of the problems. The CPU times taken by both methods showed that for some problems of this class, NIRK method is more efficient than ode15s scheme and for the others, the results are opposite.

In case of nonlinear problems with complex eigenvalues, the ode15s scheme is more accurate for one problem whereas for other problems NIRK method performed well. The results for function evaluations are similar to the previous problem class. Moreover, NIRK method showed poor efficiency than ode15s scheme. The former method used relatively more computational time except for one problem.

Our work has several possible implications. Overall, we observed that the accuracy of NIRK method is promising. The NIRK method is more efficient for most of the problems in terms of CPU time. However, NIRK method showed poor results in terms of function evaluations for most of the problems. We found among the four local error estimation techniques we tested that EMEE technique performed better as compared to the other techniques. Further work can be done by investigating the possibility of implementing such numerical methods for non-linear problems of fractional order [1].

5. Appendix

Test problems (Stiff DETEST)

The problems tested in this work were from the well known stiff DETEST problems [8]. The differential equations of all the tested problems, their intervals of integration, initial conditions, step sizes, and the eigenvalues of the Jacobian for each problem are listed below.

Problem Class D: *Nonlinear With Real Eigenvalues*

D1:
$$\begin{aligned} y_1' &= 0.2(y_2 - y_1, & y_1(0) &= 0,\\ y_2' &= 10y_1 - (60 - 0.125y_3)y_2 + 0.125y_3, & y_2(0) &= 0,\\ y_3' &= 1, & y_3(0) &= 0, \end{aligned}$$

$$t_{end} = 400, \quad h_{initial} = 1.7 \times 10^{-2},$$
(Eigenvalues: $0, -0.17 \rightarrow -0.012, -60 \rightarrow -11$)

D2:

$$\begin{aligned} y_1' &= -0.04y_1 + 0.01y_2y_3, & y_1(0) &= 1,\\ y_2' &= 400y_1 - 100y_2y_3 - 3000y_2^2, & y_2(0) &= 0,\\ y_3' &= 30y_2^2, & y_3(0) &= 0, \end{aligned}$$

$$t_{end} = 40, \quad h_{initial} = 10^{-5},$$

(Eigenvalues: $0, 0 \to -3100, -0.040 \to -0.40 \to -0.030$)

D3:

$$\begin{aligned} y_1' &= y_3 - 100y_1y_2, & y_1(0) &= 1,\\ y_2' &= y_3 + 2y_4 - 100y_1y_2 - 2\times 10^4y_2^2, & y_2(0) &= 1,\\ y_3' &= -y_3 + 100y_1y_2, & y_3(0) &= 0,\\ y_4' &= -y_4 + 10^4y_2^2, & y_4(0) &= 0, \end{aligned}$$

$$t_{end} = 20, \quad h_{initial} = 2.5\times 10^{-5},$$

(Eigenvalues: $0, -100 \to -1.4, -4.0\times 10^4 \to -290$)

D4:

$$\begin{aligned} y_1' &= -0.013y_1 - 1000y_1y_3, & y_1(0) &= 1,\\ y_2' &= -2500y_2y_3, & y_2(0) &= 1,\\ y_3' &= -0.013y_1 + 1000y_1y_3 - 2500y_2y_3, & y_3(0) &= 0, \end{aligned}$$

$$t_{end} = 50, \quad h_{initial} = 2.9\times 10^{-4},$$

(Eigenvalues:
$0, -9.3\times 10^{-3} \to -4.0\times 10^{-3} \to -6.3\times 10^{-3}, -3.5\times 10^3 \to -3.8\times 10^3$)

D5:

$$\begin{aligned} y_1' &= 0.01 - [1 + (y_1+1000)(y_1+1)](0.01 + y_1 + y_2), & y_1(0) &= 0,\\ y_2' &= 0.01 - (1+y_2^2)(0.01 + y_1 + y_2), & y_2(0) &= 0, \end{aligned}$$

$$t_{end} = 100, \quad h_{initial} = 10^{-4},$$

(Eigenvalues: $-.01 \to -.0002 \to -.002, -1000 \to -400)$)

D6:

$$\begin{aligned} y_1' &= -y_1 + 10^8y_3(1-y_1), & y_1(0) &= 1,\\ y_2' &= -10y_2 + 3\times 10^7y_3(1-y_2), & y_2(0) &= 0,\\ y_3' &= -y_1' - y_2', & y_3(0) &= 0, \end{aligned}$$

$$t_{end} = 1, \quad h_{initial} = 3.3\times 10^{-8},$$

(Eigenvalues: $0, -1.0 \to -8.6, -3.0\times l0^7 \to -4.0\times 10^7$)

PROBLEM CLASS E: *Nonlinear With Complex Eigenvalues*

E1:
$$\begin{aligned}
y_1' &= y_2, & y_1(0) &= 0,\\
y_2' &= y_3, & y_2(0) &= 0,\\
y_3' &= y_4, & y_3(0) &= 0,\\
y_4' &= (y_1^2) - \sin y_1 - 10^8)y_1 + (\frac{y_2 y_3}{y_1^2+1} - 4\times 10^6)y_2 + \\
&\quad (1-6\times 10^4)y_3 + (10exp(-y_4^2) - 4\times 10^2)y_4 + 1, & y_4(0) &= 0,
\end{aligned}$$

$$t_{end} = 1, \quad h_{initial} = 6.8\times 10^{-3},$$

(Eigenvalues: $-130 \pm 69i, -64 \pm 22i$)

E2:
$$\begin{aligned}
y_1' &= y_2, & y_1(0) &= 2,\\
y_2' &= 5(1-y_1^2)y_2 - y_1, & y_2(0) &= 0,
\end{aligned}$$

$$t_{end} = 1, \quad h_{initial} = 10^{-3},$$

(Eigenvalues: -0.067 and $-15 \to 5.7$ and $-1.5 \to 3.6$ and $1.4 \to 2.4 \pm 2.8i \to -0.052 \pm 8.8i \to -2.0 \pm 9.5i \to -5.9 \pm 4.5i \to -2.0$ and $-12 \to 0.050$ and $-15 \to 1.1$ and -3.4)

E3:
$$\begin{aligned}
y_1' &= -(55+y_3)y_1 + 65y_2, & y_1(0) &= 1,\\
y_2' &= 0.0785(y_1 - y_2), & y_2(0) &= 1,\\
y_3' &= 0.1y_1, & y_3(0) &= 0,
\end{aligned}$$

$$t_{end} = 500, \quad h_{initial} = 0.02,$$

(Eigenvalues: $0.0062 \pm 0.01i \to 0.0014 \pm 0.014i \to -0.015$ and $-4.0\times 10^{-4}, -55 \to -81$)

E4:
$$y' = -Q^T \begin{bmatrix} -10 & -10 & 0 & 0\\ 10 & -10 & 0 & 0\\ 0 & 0 & 1000 & 0\\ 0 & 0 & 0 & 0.01 \end{bmatrix} Qy + G(y), \quad y(0) = \begin{bmatrix} 0\\ -2\\ -1\\ -1 \end{bmatrix},$$

where

$$Q = \frac{1}{2}\begin{bmatrix} -1 & 1 & 1 & 1\\ 1 & -1 & 1 & 1\\ 1 & 1 & -1 & 1\\ 1 & 1 & 1 & -1 \end{bmatrix} Qy + G(y), \quad G(y) = Q^T \begin{bmatrix} (S_1^2 - S_1^2)/2\\ S_1 S_2\\ S_3^2\\ S_4^2 \end{bmatrix}, \quad Qy = \begin{bmatrix} S_1\\ S_2\\ S_3\\ S_4 \end{bmatrix}$$

$$t_{end} = 1000, \quad h_{initial} = 10^{-3},$$

(Eigenvalues: $-2.0 \to -1.0\times 10^{-2}, 8.0 \pm 10i \to 7.1 \pm 7.9i \to 9.0 \pm 0.030i \to 12 \pm 13i \to 0.19 \pm 29i \to -17 \pm 18i \to -10 \pm 10i, -10^3$)

E5:
$$\begin{aligned}
y_1' &= -7.89\times10^{-10}y_1 - 1.1\times10^{7}y_1y_3, && y_1(0)=1.76\times10^{-3},\\
y_2' &= 7.89\times10^{-10}y_1 - 1.13\times10^{9}y_2y_3, && y_2(0)=0,\\
y_3' &= -7.89\times10^{-10}y_1 - 1.1\times10^{7}y_1y_3 && \\
&\quad + 1.13\times10^{3}y_4 - 1.13\times10^{9}y_2y_3, && y_3(0)=0,\\
y_4' &= 1.1\times10^{7}y_1y_3 - 1.13\times10^{3}y_4, && y_4(0)=0,
\end{aligned}$$

$$t_{end}=1000, \quad h_{initial}=5\times10^{-5},$$

(Eigenvalues:
$0, -7.5\times10^{-10}\pm9.2\times10^{-4}i \to 2.9\times10^{-4}\pm8.7\times10^{-4}i \to -9.4\times10^{-5}$ and $-0.019, -2.0\times10^{4}$)

References

[1] M. A. Anwar, S. U. Rehman, F. Ahmad and M. I. Qadir, *A Numerical Iterative Scheme for Solving Nonlinear Boundary Value Problems of Fractional Order 0 < alpha < 1*, Punjab Univ. j. math. **51**, No. 1 (2019) 115-126.

[2] R. C. Aiken, *Stiff Computation*, Oxford University Press, 1985.

[3] J. C. Butcher, *Implicit Runge-Kutta processes*, MATH COMPUT. **18**, (1964) 50-64.

[4] J. C. Butcher, *Numerical Methods for Ordinary Differential Equations*, John Wiley & Sons, 2008.

[5] J. R. Cash and A. Singhal, *Mono-implicit Runge–Kutta formulae for the numerical integration of stiff differential systems*, IMA J. Numer. Anal. **2**, (1982) 211-227.

[6] C. F. Curtiss and J. O. Hirschfelder, *Integration of Stiff Equations*, Proc. Natl. Acad. Sci. U.S.A. **38**, (1952) 235-243.

[7] G. Dahlquist and A. Bjorck, *Numerical Methods*, Prentice-Hall Inc. 1974.

[8] W. H. Enright, T. E. Hull and B. Lindberg, *Comparing numerical methods for stiff systems of ODEs*, BIT. **15**, (1975) 10-48.

[9] E. Hairer and S. P. Nørsett and G. Wanner, *Solving Ordinary Differential Equations I: Nonstiff Problems*, Springer-Verlag, 1993.

[10] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II: Stiff and Differential Algebraic Problems*, Springer-Verlag, 1996.

[11] G. Yu. Kulikov, *Automatic error control in the Gauss-type nested implicit Runge–Kutta formula of order 6*, Russ J Numer Anal Math Model. **24**, (2009) 123-144.

[12] G. Yu. Kulikov, A. I. Merkulov and S. K. Shindin, *Asymptotic error estimate for general Newton-type methods and its applications to differential equations*, Russ. J. Numer Anal. Math. Model. **22**, (2007) 567-590.

[13] G. Yu. Kulikov and S. K. Shindin, *Adaptive nested implicit Runge–Kutta formulas of Gauss type*, Appl. Numer. Math. **59**, (2009) 707-722.

[14] G. Yu. Kulikov and S. K. Shindin, *On a family of cheap symmetric one-step methods of order four*, In International Conference on Computational Science. Springer, Berlin, Heidelberg (2006) 781-785.

[15] L. F. Shampine and M. W. Reichelt, *The matlab ode suite*, SIAM J SCI COMPUT. **18**, (1997) 1-22.

[16] A. Ur-Rehman, S. Rehman and J. Ahmad, *Numerical approximation of stiff problems using efficient nested implicit Runge–Kutta methods*, Punjab Univ. j. math. **51**, No. 2 (2019) 1-19.