A method for constructing generalized
Runge–Kutta methods *

Hiroshi Sugiura and Tatsuo Torii
Department of Information Engineering, Faculty of Engineering, Nagoya University, Nagoya, 464-01 Japan

Received 1 August 1990
Revised 1 April 1991

Abstract

In the implementation of an implicit Runge–Kutta formula, we need to solve systems of nonlinear equations. In this paper, we analyze the Newton iteration process and a modified Newton iteration process for solving these equations. Then we propose the methods in which we take only a fixed finite number of iterations and adopt the last iterate as the approximate solution for the nonlinear equations. Our methods are a kind of generalized Runge–Kutta methods with the same order as the original Runge–Kutta formula and inherit its linear stability properties of the original implicit Runge–Kutta formula, for example, AN-stability, L-stability and S-stability. Based on this fact, we construct three methods of order five imbedding a fourth-order formula for error estimation. Finally, test results for 25 stiff problems are discussed.

Keywords: Implicit Runge–Kutta method, generalized Runge–Kutta method, linear stability, Newton iteration, modified Newton iteration.

1. Introduction
We consider numerical methods for the stiff initial-value problem
\[ y'(x) = f(x, y(x)), \quad y(x_0) = y_0, \quad (1.1) \]
where \( f \) and \( y \) are \( m \)-dimensional vectors. We assume throughout that \( f \) has continuous derivatives of arbitrary order with respect to \( x \) and \( y \).

The \( s \)-stage implicit Runge–Kutta formula \( \mathcal{R} = \mathcal{R}\{s, A, b, c\} \) with fixed parameters \( A = (a_{ij}), \quad b = (b_i) \) and \( c = (c_i) \) is given by
\begin{align*}
Y(x_0 + h) &= y_0 + h \sum_{i=1}^{s} b_i d_i \approx y(x_0 + h), \quad (1.2) \\
d_i &= f \left( x_0 + h c_i, \quad y_0 + h \sum_{j=1}^{s} a_{ij} d_j \right), \quad 1 \leq i \leq s, \quad (1.3)
\end{align*}
where \( Y(x_0 + h) \) and \( d_i, \quad 1 \leq i \leq s, \) are \( m \)-dimensional vectors [3].

* This research was supported by the Nitto Foundation.
For fixed $h > 0$, the left-hand side $Y(x_0 + h)$ in (1.2) gives an approximation to $y(x_0 + h)$ after solving the nonlinear equations (1.3) with respect to $\{d_i\}$.

There are many implicit methods which have high classical order of consistency. For example, the Gauss method is of order $2s$, the Radau I, Radau IA, Radau II, Radau IIA methods are of order $2s - 1$ and the Lobatto III, Lobatto IIIA, Lobatto IIIB, Lobatto IIIC methods are of order $2s - 2$ [5,7].

To solve the stiff problem (1.1), we must analyze the stability of the method. Some stability criteria are concerned with the autonomous one-dimensional test problem

$$y' = \lambda y, \quad y(0) = 1, \quad \lambda \in \mathbb{C}^{-},$$

(1.4)

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

**Definition 1.1.** Let $Y(h)$, $h > 0$, be the approximation to the solution $y(h)$ of (1.4) by some method. The method is said to be

(a) **A-stable** if $|Y(h)| \leq 1$ for all $\lambda \in \mathbb{C}^{-}$;

(b) **L-stable** if it is A-stable and $|Y(h)| \to 0$ as $h \to \infty$, for all $\lambda \in \mathbb{C}^{-}$.

Burrage and Butcher [2] consider a more general test problem of nonautonomous form

$$y' = \lambda(x) y, \quad y(0) = 1, \quad \lambda(x) \in \mathbb{C}^{-} \text{ for } 0 \leq x < \infty,$$

(1.5)

and proposed the criterion called AN-stability.

Prothero and Robinson [16] proposed other criteria: S-stable, strongly S-stable and stiff accuracy, which are concerning the test problem

$$y' = g'(x) + \lambda \{y - g(x)\}, \quad \lambda \in \mathbb{C}^{-}.$$  

(1.6)

We say that a stability criterion is **linear** if the test problem is linear with respect to $y$.

The Radau IIA and Lobatto IIIC satisfy all the criteria introduced above.

These formulae have high order and good stability. But we must solve the nonlinear equation (1.3) numerically without guarantee of existence and uniqueness of the solution in general. Usually, some iteration method is used to solve the equation (1.3) and the iteration process is continued until it can be regarded as converged. But in general, there is no guarantee of convergence of this process, and it is difficult to design a reliable "stopping rule" of the process for a practical computer program. These facts have made the implementation of implicit Runge-Kutta methods difficult.

2. Newton iteration

In Sections 2–4, we fix a formula $\mathcal{S}\{s, A, b, c\}$. We rewrite (1.3) as follows.

$$P(d) \equiv d - F(d) = 0,$$

(2.1)

where $d = (d_1, d_2, \ldots, d_s)^T \in \mathbb{R}^{sm}$, and

$$F(d) \equiv (f(x_i, y_i))_{i=1, \ldots, s}^T, \quad x_i = x_0 + h c_i, \quad y_i = y_0 + h \sum_{j=1}^{s} a_{ij} d_j, \quad 1 \leq i \leq s.$$  

(2.2)
The Newton method for (2.1) with initial value \(d^{(0)}\) is given by
\[
d^{(k+1)} = N(d^{(k)}) := d^{(k)} - \left[ P'(d^{(k)}) \right]^{-1} P(d^{(k)}), \quad k \geq 0,
\] (2.3)
where \(P'\) is the Jacobian matrix of \(P\).

We give the explicit form of \(P'\) as follows:
\[
P'(d) = I_{sm} - h \text{ diag}(J(x_i, y_i))(A \otimes I_m),
\] (2.4)
where \(J(x, y)\) is the Jacobian matrix of \(f\) with respect to \(y\) on \((x, y)\), \(I_m\) is the \(m\)-dimensional identity matrix, \(\text{diag}(U_i)\) is the block diagonal matrix with diagonal blocks \(U_i\), \(1 \leq i \leq s\), and \(\otimes\) means the Kronecker product of matrices.

We introduce the infinity norm in \(\mathbb{R}^{sm}\).

The following theorem due to Kantorovich [14] is essential to investigate the Newton iteration process.

**Theorem 2.1.** Let \(d^{(0)} \in \mathbb{R}^{sm}\) and \(B(r) := \{ d \mid \| d - d^{(0)} \| \leq r \}\) be a closed ball. Suppose
\[
I' = \left[ P'(d^{(0)}) \right]^{-1}
\] exists,
\[
\| I' P(d^{(0)}) \| \leq \eta,
\] (2.5)
\[
\| I' P''(d) \| \leq K, \quad d \in B(r).
\] (2.6)
Then, provided
\[
\alpha = K \eta < \frac{1}{2}, \quad \frac{1 - \sqrt{1 - 2\alpha}}{\alpha} \eta = r_0 \leq r < r_1 = \frac{1 + \sqrt{1 - 2\alpha}}{\alpha} \eta,
\] (2.7)
equation (2.1) has the unique solution \(d^*\) in \(B(r)\) and
\[
\| d^{(k)} - d^* \| \leq \frac{1}{2^k} (2\alpha)^{n-1} \eta, \quad k \geq 0.
\] (2.8)

In our case,
\[
\left[ P'(d^{(0)}) \right]^{-1} = (I_{sm} - h \text{ diag}(J(x_i, y_i))(A \otimes I_m))^{-1} \xrightarrow{h \to 0} I_{sm},
\] (2.9)
\[
\| I' P''(d) \| \leq \| I' \| \max_{\| u \| = 1} \| h \text{ diag}(H(x_i, y_i))(A \otimes I_m) u \| (A \otimes I_m) \|
\] \[
\leq h \| I' \| \| A \| \max_{1 \leq i \leq s} \| H(x_i, y_i) \| = O(h),
\] (2.10)
where \(H(x, y)\) is the second derivative of \(f\) on \((x, y)\).

**Theorem 2.2.** Suppose
\[
P(d^{(0)}) = O(h^n),
\] (2.11)
for some \(n > 1\). Then, for sufficiently small \(h\), (2.1) has the unique solution \(d^*\) and
\[
d^{(k)} = d^* + O(h^{2^{(n+1)-1}}).
\] (2.12)

**Proof.** We already proved the conditions (2.5) and (2.7) for sufficiently small \(h\).

The asymptotic property (2.10) and the assumption (2.12) imply \(\| I' P(d^{(0)}) \| = O(h^n)\) and we can take \(\eta = O(h^n)\). Hence, from (2.11), we get \(\alpha = O(h^{n+1})\) and \(r_0 = O(h^n)\).
Therefore, the conditions (2.6) and (2.8) of Theorem 2.1 are satisfied for sufficiently small $h$. Hence (2.1) has the unique solution $d^*$. Finally, we obtain (2.13) from (2.9).

A trivial initial value $d^{(0)} = (f(x_0, y_0), \ldots, f(x_0, y_0))^T$ satisfies (2.12) with $n = 1$. Therefore, for sufficiently small $h$, we may assume the existence of the solution $d^*$ in the neighborhood of $d^{(0)}$.

In the special case $k = 1$ of (2.13), we get the performance of one iteration with the operator $N$ in (2.3).

**Corollary 2.3.** It holds that

$$d = d^* + O(h^n) \Rightarrow N(d) = d^* + O(h^{2n+1}).$$

**Proof.** For sufficiently small $h$, $d = d^* + O(h^n)$ implies $P(d) = O(h^n)$. Taking $d^{(0)} = d$ and $k = 1$ in Theorem 2.2, we get the result.

In actual implementation, we take some low-order approximation to $(y'(x_0 + c_i h))^T$ as the initial value $d^{(0)}$ and the stepsize is controlled so small that the solution $y(x_0 + ht)$ on $[x_0, x_0 + h]$ is sufficiently smooth with respect to $t$. Since the norm of the correction $\| TP(d^{(0)}) \|$ in (2.6) roughly indicates the smoothness of the solution $y(x_0 + ht)$, we can expect that $\| TP(d^{(0)}) \|$ is not so large. But in some problems, the norm of the second derivatives $\| P''(d) \|$ may be very large independently of the smoothness of $y$, and the stepsize $h$ may be restricted to an uneconomically small length [10]. Our methods proposed in Section 6 may be also influenced by this property of Newton iteration.

Butcher [3] introduced the statement $C(\xi)$ to characterize implicit Runge–Kutta formulae such that

$$C(\xi): \sum_{j=1}^{s} a_{ij}e_j^{k-1} = \frac{c_k^i}{k}, \quad \text{for } 1 \leq i \leq s, \ 1 \leq k \leq \xi.$$

This implies immediately

$$h \sum_{j=1}^{s} a_{ij} \phi(hc_j) = \int_{0}^{hc_i} \phi(x) \, dx + O(h^{k+1}), \quad 1 \leq i \leq s,$$

for a sufficiently smooth function $\phi(x)$.

All formulae introduced in Section 1 satisfy at least $C(s - 1)$.

The solution $d^*$ is an approximation to the derivative $y'$ on the point set $\{ x_0 + hc_i \}_{1 \leq i \leq s}$ as shown in the following corollary.

**Corollary 2.4.** The statement $C(\xi)$ implies that the solution $d^*$ of (2.1) satisfies

$$d^* = y'(x_0 + hc) + O(h^{k+1}),$$

where $y'(x_0 + hc) = (y'(x_0 + hc_1), \ldots, y'(x_0 + hc_s))^T$. 
H. Sugiura, T. Torii / Generalized Runge-Kutta methods 403

Proof. Set $d^{(0)} = y'(x_0 + hc)$ in Theorem 2.2. Then,

$$P(d^{(0)}) = \left(y'(x_0 + hc) - f\left[x_0 + hc_i, y_0 + h \sum_{j=1}^{s} a_{ij} y'(x_0 + hc_i)\right]\right)_{1 \leq i \leq s}^T$$

$$= \left(y'(x_0 + hc) - f\left[x_0 + hc_i, y(x_0 + hc_i) + O(h^{k+1})\right]\right)_{1 \leq i \leq s}^T = O(h^{k+1}),$$

from (2.15) and $y'(x) = f(x, y)$. Therefore, setting $k = 0$ in (2.13), we get the required result.

\[\square\]

3. Modified Newton iteration

We need $s$ evaluations of $f$ and its Jacobian matrix $J$ per one iteration (2.3). And we also need to solve an $sm$-dimensional system of linear equations which requires $\frac{1}{2}(sm)^3$ real multiplications.

To reduce these costs, we consider a modified form of Newton iteration to solve the equation (2.1). The modified Newton iteration can be written as follows:

$$d^{(k+1)} = \tilde{N}(d^{(k)}) := d^{(k)} - \left[I - hA \otimes J(x_0, y_0)\right]^{-1}P(d^{(k)}), \quad k \geq 0. \quad (3.1)$$

In the whole iteration process, we need only one evaluation of the Jacobian matrix $J(x_0, y_0)$. Moreover $J(x_0, y_0)$ can be commonly used in (3.1) between different implicit Runge-Kutta formulae and for different stepsize $h$.

To solve the linear system

$$\{I - h(A \otimes J(x_0, y_0))\} x = b, \quad (3.2)$$

several authors proposed efficient methods [1,4,6,17]. Butcher [4] proposed a method which uses a similarity transform of the coefficient matrix $A$ to the Jordan canonical form. If $A$ is semi-simple, the $sm$-dimensional system of linear equations (3.2) is reduced to the $s$ linear systems of dimension $m$:

$$\{I_m - h\lambda_i J(x_0, y_0)\} x'_i = b'_i, \quad 1 \leq i \leq s, \quad (3.3)$$

where $\lambda_i$, $1 \leq i \leq s_1$, are real eigenvalues of $A$, and $\lambda_{i+s_2} = \sum_{i=1}^{s_1} s_1 + 1 \leq i \leq s_1 + s_2$, are nonreal eigenvalues of $A$. Hence we need $s_1$ LU decompositions of $m$-dimensional real matrices and $s_2$ LU decompositions of $m$-dimensional complex matrices for solving (3.2). Varah [17] proposed a method which uses a similarity transform of the Jacobian matrix $J(x_0, y_0)$ to the Hessenberg form combining with the Butcher’s transform of $A$. His method is efficient for a large system with the dense Jacobian matrix.

Theorem 3.1. For sufficiently small $h$, we get

$$d = d^* + O(h^n) \Rightarrow \tilde{N}(d) = d^* + O(h^{n+2}), \quad n \geq 1. \quad (3.4)$$

Therefore the process (3.1) converges to $d^*$ if $d^{(0)} = d^* + O(h)$.  

Proof. The conditions of (3.4) and (2.16) imply \( d - y'(x_0 + he) + O(h) \). Therefore we get \( \text{diag}(J(x_i, y_i) - J(x_0, y_0)) = O(h) \). Hence, for sufficiently small \( h \),

\[
N(d) - \bar{N}(d) = \left\{ \left[ I - h \text{diag}(J(x_i, y_i))(A \otimes I) \right]^{-1} - \left[ I - hA \otimes J(x_0, y_0) \right]^{-1} \right\} P(d) - O(h^{n+2}).
\]

Then we get the result from (2.14). \( \square \)

For the inner iteration of implicit multistep methods, the counterparts of Theorems 2.2 and 3.1 were proved in [15].

4. Accuracy and stability of the quadrature

In the previous sections, we discussed the accuracy of the iterations \( d = d^{(k)} \) of the Newton process and the modified Newton process as approximations to the solution \( d^* \). In this section, we study the accuracy and stability of the quadrature \( Q(d) \) by the formula (1.2) as

\[
Q(d) := y_0 + h \sum_{i=1}^{s} b_i d_i.
\] (4.1)

Proposition 4.1. Suppose \( h \) is sufficiently small and let \( d^* \) be the solution of (2.1) and the implicit Runge-Kutta formula \( F \) is of order \( p \).

For all \( s \)-dimensional vectors \( d = (d_1, d_2, \ldots, d_s)^T = d^* + O(h^n) \), \( n \geq 1 \), the integration of \( d \) by (1.2) satisfies

\[
Q(d) = y(x_0 + h) + O(h^{\text{min}(n+1, p+1)}).
\] (4.2)

Hence, the quadrature method \( Q(d) \) is of order \( \text{min}(n, p) \).

Proof. From the assumptions, we immediately get

\[
|Q(d) - y(x_0 + h)| = |Q(d) - Q(d^*)| + |Q(d^*) - y(x_0 + h)|
\]

\[
= h \sum_{i=1}^{s} b_i (d_i - d_i^*) + O(h^{p+1}) = O(h^{\text{min}(n+1, p+1)}). \quad \square
\]

Corollary 2.3 (respectively Theorem 3.1) shows that the process (2.3) (respectively (3.1)) starting from the initial guess \( d^{(0)} \) only needs a finite number of iterations in order to get the iteration \( d^{(k)} \) such that \( d^{(k)} = d^* + O(h^p) \). From Proposition 4.1, we obtain the quadrature method \( Q(d^{(k)}) \) having the same order of accuracy as the original implicit Runge-Kutta formula without solving any nonlinear equations. This method belongs to the class of generalized Runge-Kutta methods because it directly contains the Jacobian matrix of \( f \) in the formula.

This method inherits some linear stabilities from the original implicit Runge-Kutta formula.

Proposition 4.2. Suppose the same hypothesis as in Proposition 4.1. If an implicit Runge-Kutta formula \( Y(x_0 + h) = Q(d^*) \) satisfies some stability criterion concerning the linear test problem, then the method \( Q(N(d)) \) also satisfies the same criterion.
And if the method \( Q(d^*) \) satisfies some stability criterion concerning the linear test problem in which the coefficient of \( y \) is constant, then the method \( Q(\hat{N}(d)) \) also satisfies the same criterion.

**Proof.** These two statements are trivial from the following facts. If the problem (1.1) is linear with respect to \( y \), then \( N(d) = d^* \). And if the problem is linear with the constant coefficient of \( y \), then \( \hat{N}(d) = N(d) = d^* \). \( \square \)

Therefore, if we take the Radau IIA or the Lobatto IIIC as an original Runge–Kutta formula, then the method \( Q(N(d)) \) satisfies all the stability criteria mentioned in Section 1, and \( Q(\hat{N}(d)) \) also satisfies all of them except for AN-stability.

5. Continuous approximation to \( y' \)

Let \( \mathcal{F} \{ s, A, b, c \} \) be an implicit Runge-Kutta method of order \( p \) which satisfies the statement \( C(\xi) \) and let \( d^* \) be the solution of (2.1) with respect to \( \mathcal{F} \{ s, A, b, c \} \). By virtue of Corollary 2.4, a continuous approximation to \( y' \) can be obtained by polynomial interpolation.

For an \( sm \)-dimensional vector \( d = (d_1, d_2, \ldots, d_s)^T \), let \( T(x) = (t_1(x), t_2(x), \ldots, t_m(x))^T \) be an \( m \)-dimensional vector with polynomial entries \( t_i(x) \) of degree \( \leq s - 1 \) which satisfies the interpolation conditions
\[
T(x_0 + hc_i) = (t_1(x_0 + hc_i), t_2(x_0 + hc_i), \ldots, t_m(x_0 + hc_i))^T = d_i, \quad 1 \leq i \leq s. \tag{5.1}
\]
Hence, \( T(x) \) is an interpolant for given \( x_0 + hc_i \) and \( d_i \).

If the smallest abscissa of the formula \( c_i \neq 0 \), as in the case of Radau IIA, let \( \bar{T}(x) = (\bar{t}_1(x), \bar{t}_2(x), \ldots, \bar{t}_m(x))^T \) be an \( m \)-dimensional vector with polynomial entries of degree \( \leq s \) which satisfies
\[
\bar{T}(x_0 + hc_i) = (\bar{t}_1(x_0 + hc_i), \bar{t}_2(x_0 + hc_i), \ldots, \bar{t}_m(x_0 + hc_i))^T = d_i, \quad 0 \leq i \leq s, \tag{5.2}
\]
where \( c_0 = 0 \) and \( d_0 = f(x_0, y_0) \). From (2.16), if we assume \( d = d^* + O(h^n) \), (5.1) or (5.2) gives continuous approximation to the derivative \( y' \) such as
\[
T(x) = y'(x) + O(h^{\min(n,s,1,\xi+1)}), \quad 0 \leq x \leq h, \tag{5.3}
\]
\[
\bar{T}(x) = y'(x) + O(h^{\min(n,s+1,\xi+1)}), \quad 0 \leq x \leq h. \tag{5.4}
\]

Let \( \mathcal{F}' \{ s', A', b', c' \} \) be another formula which satisfies \( C(\xi') \). We obtain an approximation to the solution \( d'^* \) of (2.1) with respect to \( \mathcal{F}' \) by the evaluation of (5.3) or (5.4) on \( x_0 + hc' \) such as
\[
I[\mathcal{F}', \mathcal{F}](d) := (T(x + hc'_1), T(x + hc'_2), \ldots, T(x + hc'_s))^T
= d'^* + O(h^{\min(n,s,1,\xi'+1)}), \tag{5.5}
\]
\[
\bar{I}[\mathcal{F}', \mathcal{F}](d) := (\bar{T}(x + hc'_1), \bar{T}(x + hc'_2), \ldots, \bar{T}(x + hc'_s))^T
= d'^* + O(h^{\min(n,s+1,1,\xi'+1)}). \tag{5.6}
\]
6. Construction of generalized Runge–Kutta methods

In this section, we propose three methods of order 5 in which a method of order 4 is embedded to perform error estimation and stepsize control.

We take \( d^{(0)} = (f(x_0, y_0), f(x_0, y_0))^T \) as an initial value, and construct one-step methods by combining operators \( N, N, I \) and \( Q \) on several different implicit Runge–Kutta formulae. The orders of our formulae are easily proved from Corollary 2.4, Proposition 4.1 and (5.5), (5.6). And their stability is proved from Proposition 4.2.

We denote by \( L_s \) the \( s \)-stage Lobatto IIIC formula which satisfies \( C(s - 1) \) and by \( R_s \) the \( s \)-stage Radau IIA formula which satisfies \( C(s) \). Notations \( N[F], N[F], I[F] \) and \( Q[F] \) mean operators \( N, N, I \) and \( Q \) with respect to a Runge–Kutta formula \( F \).

**Method 6.1** (order 5 (4) with 7 functions and 7 Jacobian matrices).

\[
\begin{align*}
d^{(1)} &= N[L_3] \circ I[L_2, L_2] \circ N[L_2](d^{(0)}), \quad d^{(0)} = (f(x_0, y_0), f(x_0, y_0))^T; \\
Y(x_0 + h) &= Q[L_3](d^{(1)}); \\
Y(x_0 + h) &= Q[R_3] \circ N[R_3] \circ I[R_3, L_3](d^{(1)}).
\end{align*}
\]

We need to solve three systems of linear equations of dimension \( 2m, 3m \) and \( 3m \).

Formulae (6.2) and (6.3) satisfy all linear stability criteria introduced in Section 1. Because \( L_s(s, A, b, c) \) satisfies \( C(s - 1) \) and the smallest abscissa of the formula \( c_1 = 0 \), \( N[L_2] \) and \( N[L_3] \) commonly use \( f(x_0, y_0) \) and \( J(x_0, y_0) \) in Method 6.1. Hence we only need 7 function and 7 Jacobian evaluations.

If we require only A-stability for \( Y(x_0 + h) \), we use the \( s \)-stage Lobatto IIIA formula \( L_s' \) instead of \( L_s \) in Method 6.1. Since \( c_1 = 0 \) and \( a_{1,j} = 0 \), \( 1 \leq j \leq s \), for \( L_s'(s, A, b, c) \), we only need to solve the \( (s - 1)m \)-dimensional system of linear equations for Newton iteration \( N[L_s'] \), and we can use \( f(x_0, y_0) \) and \( J(x_0, y_0) \) in \( N[L_2] \) and \( N[L_3] \) commonly.

**Method 6.2** (order 5 (4) with 7 functions and 6 Jacobian matrices).

\[
\begin{align*}
d^{(1)} &= N[L_3'] \circ I[L_2', L_2'] \circ N[L_2'](d^{(0)}), \quad d^{(0)} = (f(x_0, y_0), f(x_0, y_0))^T; \\
\tilde{Y}(x_0 + h) &= Q[L_3'](d^{(1)}); \\
Y(x_0 + h) &= Q[R_3] \circ N[R_3] \circ I[R_3, L_3'](d^{(1)}).
\end{align*}
\]

In this case, we need to solve three systems of linear equations of dimension \( m, 2m \) and \( 3m \).

By replacing Newton iterations with modified Newton iterations in Method 6.2, we have the following method.
Method 6.3 (order 5 (4) with 7 functions and 1 Jacobian matrix).

\[
\mathbf{d}^{(1)} = \hat{N} \left[ \mathcal{L}_2' \right] \circ I \left[ \mathcal{L}_2', \mathcal{L}_2' \right] \circ \hat{N} \left[ \mathcal{L}_2' \right] (\mathbf{d}^{(0)}), \quad \mathbf{d}^{(0)} = (f(x_0, y_0), f(x_0, y_0))^T; \quad (6.7)
\]

formula of order 4:

\[
\hat{Y}(x_0 + h) := Q[\mathcal{L}_3'](\mathbf{d}^{(1)}); \quad (6.8)
\]

formula of order 5:

\[
Y(x_0 + h) := Q[\mathcal{R}_3] \circ \hat{N} \left[ \mathcal{R}_3 \right] \circ I \left[ \mathcal{R}_3, \mathcal{L}_3' \right] (\mathbf{d}^{(1)}). \quad (6.9)
\]

In this case, we need to solve two systems of linear equations of dimension \( m \), and two systems of linear equations with complex coefficients of dimension \( m \) using Butcher’s method mentioned in Section 3.

Remark 6.4. Prothero and Robinson [16] found that the classical orders of consistency of implicit Runge–Kutta formulae break down in the stiff case. Frank et al. [9,10] proposed the concepts of \( B \)-consistency and \( B \)-convergence, and gave realistic error bounds for the local and global error of numerical methods. Hundsdorfer [13] showed some results about \( B \)-convergence for generalized Runge–Kutta methods.

7. Numerical examples

We implement Methods 6.1–6.3 in FORTRAN subroutines for solving the initial-value problem (1.1).

Our routines have two modes LINEAR and Non-LINEAR. In the mode LINEAR, problem (1.1) is regarded as a linear problem with constant coefficients. In this mode, the Jacobian matrix once evaluated is reused on every step, and change of stepsize is inhibited for saving the cost of LU decomposition.

In Method 6.3, the mode turns from LINEAR to Non-LINEAR if the condition

\[
P[\mathcal{L}_2'] \circ \hat{N} \left[ \mathcal{L}_2' \right] (\mathbf{d}^{(0)}) \neq 0, \quad (7.1)
\]

is satisfied, where the notation \( P[\mathcal{F}] \) means the operator \( P \) in (2.1) with respect to a Runge–Kutta formula \( \mathcal{F} \). And the mode turns from Non-LINEAR to LINEAR if the condition (7.1) is false. In Methods 6.1 and 6.2, we use the conditions \( P[\mathcal{L}_2] \circ N[\mathcal{L}_2](\mathbf{d}^{(0)}) \neq 0 \) and \( P[\mathcal{L}_2'] \circ N[\mathcal{L}_2'](\mathbf{d}^{(0)}) \neq 0 \), respectively, instead of (7.1), and also check if the Jacobian matrix is identical on two different points.

On each step, we get the approximation \( Y(x_0 + h) \) to the solution \( y(x_0 + h) \) and the estimation of the local truncation error:

\[
\text{EST} := \| Y(x_0 + h) - \hat{Y}(x_0 + h) \|_\infty. \quad (7.2)
\]

We take the following stepsize control for a given tolerance TOL:

\[
h_{\text{new}} := h \cdot \min \left\{ 2, 0.9 \cdot \left( \frac{TOL \cdot \max \{1, \| Y(x_0 + h) \|_\infty \}}{\text{EST}} \right)^{1/5} \right\}. \quad (7.3)
\]
Table 1
Reliability and robustness of methods for 25 problems with 10 different tolerances

| Method  | $R = \frac{||Y - y||_\infty}{[TOL \cdot \max\{1, ||y||_\infty\}]$ | Failed |
|---------|-------------------------------------------------|--------|
|         | $< 10^{-5}$ | $< 10^{-4}$ | $< 10^{-3}$ | $< 10^{-2}$ | $< 10^{-1}$ | $< 10$ | $< 10^2$ | $< 10^3$ | $< 10^4$ | $< 10^5$ |
| 6.1     | 25          | 21          | 38          | 71          | 1/7         | 18     | 0       | 0       | 0       | 0       | 0       |
| 6.2     | 24          | 22          | 36          | 83          | 75         | 10     | 0       | 0       | 0       | 0       | 0       |
| 6.3     | 20          | 20          | 35          | 67          | 61         | 30     | 11      | 1       | 0       | 0       | 0       |
| RADAUS  | 7           | 17          | 50          | 106         | 42         | 11     | 9       | 0       | 0       | 0       | 1       |
| LSODE   | 2           | 4           | 11          | 20          | 44         | 85     | 67      | 16      | 0       | 0       | 1       |

If $EST > TOL \cdot \max\{1, ||Y(x_0 + h)||_\infty\}$, we retry the last step with a smaller stepsize desired by (7.3). If $EST \leq TOL \cdot \max\{1, ||Y(x_0 + h)||_\infty\}$, the last step is accepted, and the next stepsize is determined by (7.3) in the mode Non-LINEAR. In the mode LINEAR, the stepsize $h$ is changed only if $h_{\text{new}} \geq 1.6 \, h$.

We ran our three methods, RADAUS based on the 3-stage Radau IIA formula written by Hairer and Wanner [11] and LSODE based on Gear’s backward difference formulæ by Hindmarsh [12] on all 25 stiff test problems of [8] with ten different tolerances $TOL = 10^{-1}$, $10^{-2}, \ldots, 10^{-10}$. For RADAUS and LSODE, we set $ATOL = RTOL = TOL$. The exact Jacobian matrices were given as the subroutine programs. Varah’s similarity transform was not used in our Method 6.3 and in RADAUS because of relatively low dimensions of the problems (at most ten). Computations were performed in FORTRAN77 double precision with a 14 hexadecimal mantissa (about 16 decimals) on the FACOM M780 of Nagoya University.

Table 1 shows the distributions of $R = \frac{||Y - y||_\infty}{[TOL \cdot \max\{1, ||y||_\infty\}]$ for all 250 results by 5 methods. Methods 6.1 and 6.2 are the most robust and reliable. Method 6.3 and RADAUS are the next. LSODE is robust but less reliable.

Tables 2a–2c show the statistical summary for 5 methods with a moderate tolerance $TOL = 10^{-4}, 10^{-6}, 10^{-8}$. The unit of the “Load of LU decomposition” is the computational cost for the decomposition of an $m \times m$ real matrix. The load of the LU decomposition of an $m \times m$ complex matrix, a $2m \times 2m$ real matrix and a $3m \times 3m$ real matrix are estimated as 4 units, 8 units and 27 units, respectively. The results show that the three routines RADAUS, LSODE and Method 6.3 are the fastest. Because our methods need the exact Jacobian matrix in each step, Method 6.3 uses more Jacobian calls and LU decompositions than the other two do. It will be a serious problem for large systems with full Jacobian matrices.

Table 2a
Statistics with $TOL = 10^{-4}$

<table>
<thead>
<tr>
<th>Method</th>
<th>Computational time (ms)</th>
<th>Number of steps</th>
<th>Number of function calls</th>
<th>Number of Jacobian calls</th>
<th>Load of LU decomposition</th>
<th>$R \geq 1$</th>
<th>$R \geq 10$</th>
<th>Failed</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>529.03</td>
<td>1023</td>
<td>7161</td>
<td>3486</td>
<td>39556</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6.2</td>
<td>345.74</td>
<td>976</td>
<td>6832</td>
<td>2499</td>
<td>22788</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6.3</td>
<td>236.32</td>
<td>1199</td>
<td>8269</td>
<td>532</td>
<td>8522</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>RADAUS</td>
<td>174.65</td>
<td>988</td>
<td>5521</td>
<td>316</td>
<td>3835</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>LSODE</td>
<td>396.15</td>
<td>4698</td>
<td>8737</td>
<td>617</td>
<td>617</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Table 2b
Statistics with TOL = 10⁻⁶

<table>
<thead>
<tr>
<th>Method</th>
<th>Computational time (ms)</th>
<th>Number of steps</th>
<th>Number of function calls</th>
<th>Number of Jacobian calls</th>
<th>Load of LU decomposition</th>
<th>R &gt; 1</th>
<th>R &gt; 10</th>
<th>Failed</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>967.58</td>
<td>2310</td>
<td>16170</td>
<td>6950</td>
<td>74896</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6.2</td>
<td>613.28</td>
<td>2125</td>
<td>14889</td>
<td>4665</td>
<td>39492</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6.3</td>
<td>425.63</td>
<td>2360</td>
<td>16415</td>
<td>930</td>
<td>13291</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>RADAUS</td>
<td>413.00</td>
<td>2687</td>
<td>13871</td>
<td>649</td>
<td>77030</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LSODE</td>
<td>605.72</td>
<td>7523</td>
<td>9024</td>
<td>839</td>
<td>839</td>
<td>5</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 2c
Statistics with TOL = 10⁻⁸

<table>
<thead>
<tr>
<th>Method</th>
<th>Computational time (ms)</th>
<th>Number of steps</th>
<th>Number of function calls</th>
<th>Number of Jacobian calls</th>
<th>Load of LU decomposition</th>
<th>R &gt; 1</th>
<th>R &gt; 10</th>
<th>Failed</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>2088.56</td>
<td>5704</td>
<td>39948</td>
<td>17008</td>
<td>172748</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6.2</td>
<td>1261.63</td>
<td>5023</td>
<td>35273</td>
<td>10071</td>
<td>82881</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6.3</td>
<td>932.02</td>
<td>5483</td>
<td>38238</td>
<td>2045</td>
<td>27562</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>RADAUS</td>
<td>1116.28</td>
<td>8069</td>
<td>39791</td>
<td>1068</td>
<td>12860</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LSODE</td>
<td>883.58</td>
<td>11500</td>
<td>13446</td>
<td>1079</td>
<td>1079</td>
<td>11</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>

Acknowledgements

The authors would like to thank Prof. T. Mitsui and both referees for their useful suggestions and comments.

References


