Journal of Computational and Applied Mathematics 9 (1983) 229–238
North-Holland

An automatic multistep method for solving stiff initial value problems

Moody T. CHU
Department of Mathematics, North Carolina State University, Raleigh, NC 27650, USA

Received 15 September 1982
Revised 8 February 1983

Abstract: A multistep method with matricial coefficients is developed. It can be used to solve stiff initial value problems of the form $y' = Ay + g(x, y)$. This method bears the nature of the classical Adams–Bashforth–Moulton PC formula and can be shown to be consistent, convergent and A-stable. A careful reformulation of this method legitimatizes the implementation of this algorithm in a variable-step variable-order fashion. Numerical test results from a PECE mode of this method show its possible advantages.

Keywords: Stiff IVPs, Adams PC formula, variable-step variable-order multistep method.

1. Introduction

Since the codes DVDQ of Krogh [14] and DIFSUB of Gear [10,11] were developed, the effectiveness of a variable-step variable-order formulation of the classical Adams method has gained a great recognition in solving general initial value problems. This spirit was also carried on by Shampine and Gordon in their code STEP [26] which is known to be among the most effective and the most carefully decumented available at this time. Though the code STEP can diagnose the stiffness, still not is it adequate for handling stiff systems. The reason for this shortcoming is well understood from the Dahlquist stability theory [5]. It is also generally believed that most of the methods used for solving stiff systems are necessarily implicit [5,17,27] and hence demand the use of some expensive Newton-like iterations [12].

In this paper we consider a special class of stiff systems, i.e. a stiff initial value problem of the form

$$y' = Ay + g(x, y), \qquad y(a) = y_0, \quad x \in [a, b] \tag{1.1}$$

where A is an $n \times n$ real constant matrix of which all eigenvalues have negative real parts and $\|\partial g/\partial y\|$ is small relative to $\|A\|$. It deserves to note that a much larger class of problems

$$y' = A(x)y + g(x, y), \qquad y(a) = y_0 \tag{1.2}$$

can also be considered by rewriting it in the form

$$y' = Ay + \{[A(x) - A]y + g(x, y)\}, \qquad y(a) = y_0 \tag{1.3}$$

if A can be chosen to satisfy conditions of (1.1) and to keep $\|A(x) - A\|$ uniformly small for at least a short period of x. A multistep method which does not suffer from the difficulties mentioned above is developed for solving (1.1). This method bears the merit of the classical predictor–corrector schemes and can be regarded as the nature extension of the code STEP to matricial coefficients. As a result, we may take a great advantage of the software designs of the code STEP.

The development of this method is also motivated by several other researchers. The formulations considered by Certaine [4], Lawson [19], Lee and Preiser [20] are special cases of our consideration. The methods suggested by Jain [13], Miranker [22,23], Murphy [24], Lambert and Sigurdsson [16] are different from our method only in the numerical representations of the integrals which will be introduced in the next section.

We begin at Section 2 with the basic fixed-step formulation of this multistep method. After defining a generalized Adams–Bashforth–Moulton PECE mode, we then state some theorems concerning the consistency, convergence and A-stability of this mode. These theorems can easily be generalized to the variable-step variable-order case [26]. In Section 3 a variable-step PECE mode is reformulated. This approach makes it possible to calculate the matricial coefficients to any desired order in a very efficient way. In the special case when $A = 0$, this derivation coincides with that by Shampine and Gordon. Finally we present some test results in Section 4 to evidence the possible effectiveness of this method.

2. Basic formulation – fixed-step scheme

Let x_k denote the kth mesh point along the variable axis and $h_{k+1} = x_{k+1} - x_k$ the step size to be used at the next move. We shall use y_k to represent a numerical approximation to the exact solution $y(x_k)$ at the mesh point x_k. Let $g_k = g(x_k, y_k)$. By considering the integral equation

$$y(x_{n+1}) = \mathrm{e}^{Ah_{n+1}}y(x_n) + \mathrm{e}^{Ax_{n+1}}\int_{x_n}^{x_{n+1}} \mathrm{e}^{-A\tau}g(\tau, y(\tau))\mathrm{d}\tau \tag{2.1}$$

which is equivalent to problem (1.1), we are motivated to form the following formula

$$y_{n+1} = \mathrm{e}^{Ah_{n+1}}y_n + h_{n+1}\sum_{i=0}^{k} \Phi_{ki}g_{n-i+1}. \tag{2.2}$$

The summation in (2.2) is an approximation to the integral

$$\int_0^1 \mathrm{e}^{A(1-\alpha)h_{n+1}}g(x_n + \alpha h_{n+1}, y(x_n + \alpha h_{n+1}))\mathrm{d}\alpha \tag{2.3}$$

where $\alpha = (\tau - x_n)/h_{n+1}$, and Φ_{ki}'s are some matricial coefficients to be determined.

Let us consider the fixed-step explicit scheme first. Assuming $\Phi_{k0} = 0$ in (2.2) and regarding g as a function of α, we use a vector-valued polynomial $P(\alpha)$ of degree $< k$ to interpolate g componentwise at points $x_{n-k+1}, \dots, x_n$ so that $P(1-i) = g_{n-i+1}$ for $i = 1, \dots, k$. This polynomial is given by the Lagrangian formula

$$P(\alpha) = \sum_{i=1}^{k} g_{n-i+1}\prod_{\substack{j=1\\ j\neq i}}^{k} \frac{\alpha + j - 1}{j - i}. \tag{2.4}$$

Now the integral (2.3) is approximated by

$$\int_0^1 \mathrm{e}^{A(1-\alpha)h}P(\alpha)\mathrm{d}\alpha = \sum_{i=1}^{k}\left(\int_0^1 \mathrm{e}^{A(1-\alpha)h}\prod_{\substack{j=1\\ j\neq i}}^{k}\frac{\alpha + j - 1}{j - 1}\mathrm{d}\alpha\right)g_{n-i+1} \tag{2.5}$$

where h_{n+1} is abbreviated as h. Comparing (2.2) and (2.5), we see it is reasonable to choose Φ_{ki} in (2.2) as

$$\Phi_{ki} = \int_0^1 \mathrm{e}^{A(1-\alpha)h}\prod_{\substack{j=1\\ j\neq 1}}^{k}\frac{\alpha + j - 1}{j - i}\mathrm{d}\alpha. \tag{2.6}$$

As an example, when $k = 4$, (2.2) becomes

$$\begin{aligned} p_{n+1} = \mathrm{e}^{Ah}y_n + h\int_0^1 \mathrm{e}^{A(1-\alpha)h}\big[&\tfrac{1}{6}(\alpha+1)(\alpha+2)(\alpha+3)g_n \\ &- \tfrac{1}{2}\alpha(\alpha+2)(\alpha+3)g_{n-1} + \tfrac{1}{2}\alpha(\alpha+1)(\alpha+3)g_{n-2} - \tfrac{1}{2}\alpha(\alpha+1)(\alpha+2)g_{n-3}\big]\mathrm{d}\alpha. \end{aligned} \tag{2.7}$$

Here the notation p_{n+1} is used to indicate that it is the 'predicted value' of $y(x_{n+1})$. The scheme (2.2) along with (2.6) is called a generalized Adams–Bashforth formula of order k because, if $A = 0$, (2.2) is reduced to the classical linear multistep scheme. The word 'order' will be justified later. An implicit scheme of order $k+1$ can be formed in exactly the same way if the polynomial $P^*(\alpha)$ interpolating the prints $x_{n-k+1}, \ldots, x_{n+1}$ with values $P^*(1-\alpha) = g_{n-i+1}$ for $i = 0, \ldots, k$ is used. The resulting matricial coefficients of this implicit scheme is denoted by Φ_{ki}^*. It is worth noting that

$$\sum_{i=0}^{k} \Phi_{ki}^* = \sum_{i=1}^{k} \Phi_{ki} = \int_0^1 e^{A(1-\alpha)h} d\alpha$$

is always bounded. Now we make the following definition.

Definition 2.1. By a $(k, k+1)$-order PECE mode we mean the numerical scheme consisting of a predictor of order k and a corrector of order $k+1$ in the form of (2.2), i.e.

$$\begin{aligned}
&\text{P:} \quad p_{n+1} = e^{Ah} y_n + h \sum_{i=1}^{k} \Phi_{ki} g_{n-i+1}, \\
&\text{E:} \quad g_{n+1}^p = g(x_{n+1}, p_{n+1}), \\
&\text{C:} \quad y_{n+1} = e^{Ah} y_n + h \sum_{i=1}^{k} \Phi_{ki}^* g_{n-i+1} + h \Phi_{k0}^* g_{n+1}^p, \\
&\text{E:} \quad g_{n+1} = g(x_{n+1}, y_{n+1}).
\end{aligned} \tag{2.8}$$

Apparently, to compute these matricial coefficients is not an easy and economical task. The situation is even worsened when the variable-step scheme is to be adopted. The algorithm we develop in the next section, however, helps to facilitate this difficulty. In fact, we shall see in that algorithm only matrix multiplications are involved. Let us conclude this section with three major theorems concerning the consistency, convergence and A-stability of the mode (2.8). They are obvious extensions of classical results [26].

Theorem 2.2. *Assume $g \in C^{k+1}$, then the local truncation error $\mathcal{L}^p(y(x_n), h)$, defined by*

$$\mathcal{L}^p(y(x_n), h) = y(x_{n+1}) - e^{Ah} y(x_n) - h \sum_{i=1}^{k} \Phi_{ki} g(x_{n-i+1}, y(x_{n-i+1})) \tag{2.9}$$

of the explicit scheme (2.2) with step number k is $O(h^{k+1})$. With a similar definition the local truncation error $\mathcal{L}(y(x_n), h)$ of the implicit scheme is $O(h^{k+2})$.

Theorem 2.3. *Let the $(k, k+1)$-order PECE mode (2.8) be applied to solve problem (1.1) on $[a, b]$. Suppose $g \in C^1$ and $\|\partial g/\partial y\| \leqslant L_g$ for some constant $L_g > 0$. If all starting values y_i for $i = 1, \ldots, k-1$ satisfy $\|y(x_i) - y_i\| \leqslant E_0$, then for $x_n \in [a, b]$ we have*

$$\|y(x_n) - y_n\| \leqslant \left[E_0 + \frac{\partial}{h\Lambda}\right] e^{(x_n - a)\Lambda} \tag{2.10}$$

where

$$\begin{aligned}
&\Lambda = L_g(\alpha^* + hL_g \|\Phi_{k0}^*\| \alpha), \\
&\delta = \max_n \|h\Phi_{k0}^* \frac{\partial g}{\partial y} \mathcal{L}^p(y(x_n), h) + \mathcal{L}(y(x_n), h)\|, \\
&\alpha = \sum_{i=1}^{k} \|\Phi_{ki}\| \quad \text{and} \quad \alpha^* = \sum_{i=0}^{k} \|\Phi_{ki}^*\|.
\end{aligned} \tag{2.11}$$

In particular, if $g \in C^{k+1}$ and $E_0 = O(h^{k+1})$, then $\|y(x_n) - y_n\| = O(h^{k+1})$.

Theorem 2.4. *The multistep scheme (2.2) based on the Padé approximation to all its matrix exponentials is A-stable.*

3. Efficient formulation – variable-step scheme

The subroutine STEP in [26] gives an efficient way of generating coefficients for the linear multistep method which results in substantial savings of the overhead and memory storages. We now reformulate their settings in the matrix form. It will be seen at the end of this section that all the advantages of STEP are preserved.

Let the following quantities be defined.

$$\begin{aligned}
&h_i = x_i - x_{i-1}, \qquad s = \frac{x - x_n}{h_{n+1}},\\
&\psi_i(n+1) = h_{n+1} + \cdots + h_{n-i+2}, \quad i \geqslant 1\\
&\alpha_i(n+1) = \frac{h_{n+1}}{\psi_i(n+1)}, \quad i \geqslant 1,\\
&\beta_1(n+1) = 1, \qquad \beta_i(n+1) = \frac{\psi_1(n+1)\psi_2(n+1)\cdots\psi_{i-1}(n+1)}{\psi_1(n)\psi_2(n)\cdots\psi_{i-1}(n)}, \quad i > 1,\\
&\phi_1(n) = g[x_n] = g_n, \qquad \phi_i(n) = \psi_1(n)\cdots\psi_{i-1}(n) g[x_n, x_{n-1}, \ldots, x_{n-i+1}], \quad i > 1.
\end{aligned} \tag{3.1}$$

Recall that the predicted value of $y(x_{n+1})$ is the vector

$$p_{n+1} = e^{Ah_{n+1}} y_n + \int_{x_n}^{x_{n+1}} e^{A(x_{n+1}-\tau)} p_{k,n}(\tau) d\tau \tag{3.2}$$

where $p_{k,n}$ is the interpolating polynomial given by

$$p_{k,n}(x) = g[x_n] + (x - x_n) g[x_n, x_{n-1}] + \cdots + (x - x_n)\cdots(x - x_{n-k+2}) g[x_n, \ldots, x_{n-k+1}]. \tag{3.3}$$

A typical term, for $i \geqslant 2$, of $p_{k,n}(x)$ can be written as

$$\begin{aligned}
&(x - x_n)\cdots(x - x_{n-i+2}) g[x_n, \ldots, x_{n-i+1}] =\\
&= (sh_{n+1})(sh_{n+1} + h_n)\cdots(sh_{n+1} + h_n + \cdots + h_{n-i+3}) \frac{\phi_i(n)}{\psi_1(n)\cdots\psi_{i-1}(n)}\\
&= \left(\frac{sh_{n+1}}{\psi_1(n+1)}\right)\left(\frac{sh_{n+1} + h_n}{\psi_2(n+1)}\right)\cdots\left(\frac{sh_{n+1} + \psi_{i-2}(n)}{\psi_{i-1}(n+1)}\right)\beta_i(n+1)\phi_i(n).
\end{aligned} \tag{3.4}$$

If we introduce quantities $c_{i,n}(s)$ and $\phi_i^*(n)$ defined by

$$c_{i,n}(s) = \begin{cases} 1, & i = 1,\\ \dfrac{sh_{n+1}}{\psi_1(n+1)}, & i = 2,\\ \left(\dfrac{sh_{n+1}}{\psi_1(n+1)}\right)\left(\dfrac{sh_{n+1} + h_n}{\psi_2(n+1)}\right)\cdots\left(\dfrac{sh_{n+1} + \psi_{i-2}(n)}{\psi_{i-1}(n+1)}\right), & i \geqslant 3, \end{cases} \tag{3.5}$$

and

$$\phi_i^*(n) = \beta_i(n+1)\phi_i(n), \tag{3.6}$$

then (3.3) can be written as

$$p_{k,n}(x) = \sum_{i=1}^{k} c_{i,n}(s)\phi_i^*(n) \tag{3.7}$$

and (3.2) becomes

$$p_{n+1} = e^{Ah_{n+1}} y_n + h_{n+1} \sum_{i=1}^{k} \left(\int_0^1 e^{A(1-s)h_{n+1}} c_{i,n}(s) ds\right)\phi_i^*(n). \tag{3.8}$$

Let

$$d_{i,n}(s) = \mathrm{e}^{A(1-s)h_{n+1}} c_{i,n}(s), \tag{3.9}$$

it follows that

$$d_{i,n}(s) = \begin{cases} \mathrm{e}^{A(1-s)h_{n+1}}, & i=1, \\ \mathrm{e}^{(A(1-s)h_{n+1}} s, & i=2, \\ \left[\alpha_{i-1}(n+1)s + \dfrac{\psi_{i-2}(n)}{\psi_{i-1}(n)}\right] d_{i-1,n}(s), & i \geqslant 3, \end{cases} \tag{3.10}$$

and

$$p_{n+1} = \mathrm{e}^{Ah_{n+1}} y_n + h_{n+1} \sum_{i=1}^{k} \left(\int_0^1 d_{i,n}(s)\mathrm{d}s \right) \phi_i^*(n). \tag{3.11}$$

Observe, for a fixed n and any $i \geqslant 3$,

$$\begin{aligned} \left. \int_0^s d_{i,n}(s_0)\mathrm{d}s_0 \right) &= \int_0^s \left[\alpha_{i-1}(n+1)s_0 + \frac{\psi_{i-2}(n)}{\psi_{i-1}(n)}\right] \mathrm{d}\left(\int_0^{s_0} d_{i-1,n}(s_1)\mathrm{d}s_1 \right) \\ &= \left[\alpha_{i-1}(n+1)s + \frac{\psi_{i-2}(n)}{\psi_{i-1}(n)}\right] \int_0^s d_{i-1,n}(s_0)\mathrm{d}s_0 \\ &\quad - \int_0^s \alpha_{i-1}(n+1) \int_0^{s_1} d_{i-1,n}(s_0)\mathrm{d}s_0\mathrm{d}s_1. \end{aligned} \tag{3.12}$$

If we define

$$d_{i,n}^{(q)}(s) = \int_0^s \int_0^{s_{q-1}} \cdots, \int_0^{s_1} d_{i,n}(s_0)\mathrm{d}s_0\mathrm{d}s_1 \cdots \mathrm{d}s_{q-1}, \tag{3.13}$$

then (3.12) is equivalent to

$$d_{i,n}^{(1)}(s) = \left[\alpha_{i-1}(n+1)s + \frac{\psi_{i-2}(n)}{\psi_{i-1}(n)}\right] d_{i-1,n}^{(1)}(s) - \alpha_{i-1}(n+1) d_{i-1,n}^{(2)}(s). \tag{3.14}$$

It is not hard to show now, by induction and integration by parts, the following identity holds for any $i \geqslant 3$.

$$d_{i,n}^{(q)}(s) = \left[\alpha_{i-1}(n+1)s + \frac{\psi_{i-2}(n)}{\psi_{i-1}(n)}\right] d_{i-1,n}^{(q)}(s) - q\alpha_{i-1}(n+1) d_{i-1,n}^{(q+1)}(s). \tag{3.15}$$

Scaling these quantities by

$$\omega_{i,q} = (q-1)! d_{i,n}^{(q)}(1), \tag{3.16}$$

we then arrive at one of the important identifies

$$\omega_{i,q} = \omega_{i-1,q} - \alpha_{i-1}(n+1)\omega_{i-1,q+1} \quad \text{for } i \geqslant 3. \tag{3.17}$$

Up to this point, all these quantities we have derived are formally then same as those by Shampine and Gordon [26] except that these are in matrix form. When A is identically zero, the quantities defined in (3.10), (3.12), (3.15) and (3.17) become diagonal matrices and, in fact, each diagonal matrix is of the form cI where the scalar c is the coefficient used by Shampine and Gordon. In this case, the quantites for $i = 1$ and $i = 2$ can be handled trivially. In the other case when A is a matrix as stated in problem (1.1), it can be shown from (3.13) that $d_{1,n}^{(1)}(s)$ satisfies the equation

$$(-Ah_{n+1}) d_{1,n}^{(1)}(s) = \mathrm{e}^{A(1-s)h_{n+1}} - \mathrm{e}^{Ah_{n+1}}, \tag{3.18}$$

and, by induction, $d_{1,n}^{(q)}(s)$ satisfies

$$(-Ah_{n+1})d_{1,n}^{(q)}(s)=d_{1,n}^{(q-1)}(s)-\frac{s^{q-1}}{(1-q)!}\mathrm{e}^{Ah_{n+1}} \tag{3.19}$$

for all $q \geqslant 2$. Similarly, we have

$$\begin{aligned}&(-Ah_{n+1})d_{2,n}^{(1)}(s)=s\mathrm{e}^{A(1-s)h_{n+1}}-d_{1,n}^{(1)}(s),\\&(-Ah_{n+1})d_{2,n}^{(q)}(s)=d_{2,n}^{(q-1)}(s)-d_{1,n}^{(q)}(s)\quad \text{for } q\geqslant 2.\end{aligned} \tag{3.20}$$

Resorting to the definition (3.16), we obtain

$$\begin{aligned}&(-Ah_{n+1})\omega_{1,1}=I-\mathrm{e}^{Ah_{n+1}},\\&(-Ah_{n+1})\omega_{1,q}=(q-1)\omega_{1,q-1}-\mathrm{e}^{Ah_{n+1}}\quad \text{for } q\geqslant 2,\\&(-Ah_{n+1})\omega_{2,1}=I-\omega_{1,1},\qquad (-Ah_{n+1})\omega_{2,q}=(q-1)\omega_{2,q-1}-\omega_{1,q}\quad \text{for } q\geqslant 2.\end{aligned} \tag{3.21}$$

In general, $\omega_{1,q}$ and $\omega_{2,q}$ for every $q \geqslant 1$ can be solved directly from (3.21) once A^{-1} is known. But for the case when A is very ill-conditioned or even is singular, it would be wise to reformulate the problem in a way similar to (1.3) so that A is better conditioned and nonsingular.

Since $d_{i,n}^{(1)}(1)=\omega_{i,1}$, the scheme (3.11) becomes

$$p_{n+1}=\mathrm{e}^{Ah_{n+1}}y_n+h_{n+1}\sum_{i=1}^{k}\omega_{i,1}\phi_i^*(n). \tag{3.22}$$

We now make a correction on this predicted value. The corrected value of $y(x_{n+1})$ is given by

$$y_{n+1}=\mathrm{e}^{Ah_{n+1}}y_n+\int_{x_n}^{x_{n+1}}\mathrm{e}^{A(x_{n+1}-\tau)}p_{k+1,n}^*(\tau)\mathrm{d}\tau \tag{3.23}$$

where $p_{k+1,n}^*(\tau)$ is the interpolating polynomial of degree $<k+1$ such that $p_{k+1,n}^*(x_{n-i+1})=g_{n-i+1}$ for $i=1,\dots,k$ and $p_{k+1,n}^*(x_{n+1})=g(x_{n+1},p_{n+1})$. Since

$$\begin{aligned}p_{k+1,n}^*(x)&=p_{k,n}(x)+(x-x_n)\cdots(x-x_{n-k+1})g^p[x_{n+1},\dots,x_{n-k+1}]\\&=p_{k,n}(x)+(sh_{n+1})\cdots(sh_{n+1}+\cdots h_{n-k+2})\frac{\phi_{k+1}^p(n+1)}{\psi_1(n+1)\cdots\psi_k(n+1)}\\&=p_{k,n}(x)+c_{k+1,n}(s)\phi_{k+1}^p(n+1)\end{aligned} \tag{3.24}$$

where the superscript p is used to indicate that the value g_{n+1} is replaced by $g(x_{n+1},p_{n+1})$, we can substitute (3.24) into (3.23) to get

$$y_{n+1}=p_{n+1}+h_{n+1}\omega_{k+1,1}\phi_{k+1}^p(n+1). \tag{3.25}$$

To appreciate the above scheme we illustrate the inter-relation between the coefficients of (3.17)) and (3.21) in Table 1. The arrows emanate from the generators and point to the generated values.

Table 1

$$\begin{tikzcd}
q \diagdown i & 1 & 2 & 3 & 4 & 5 \\
1 & \omega_{11} \arrow[r] \arrow[d] & \omega_{21} \arrow[r] \arrow[dl, dash] & \omega_{31} \arrow[r] & \omega_{41} \arrow[r] & \omega_{51} \\
2 & \omega_{12} \arrow[r] \arrow[d] & \omega_{22} \arrow[r] \arrow[ur] \arrow[dl, dash] & \omega_{32} \arrow[r] \arrow[ur] & \omega_{42} \arrow[ur] & \\
3 & \omega_{13} \arrow[r] \arrow[d] & \omega_{23} \arrow[r] \arrow[ur] \arrow[dl, dash] & \omega_{33} \arrow[ur] & & \\
4 & \omega_{14} \arrow[r] & \omega_{24} \arrow[ur] & & &
\end{tikzcd}$$

Several advantages are manifestly clear now:

(i) We only need to computer A^{-1} once and for all throughout the calling of the whole algorithm because $(Ah_{\text{new}})^{-1} = (Ah_{\text{old}})^{-1} \cdot h_{\text{old}}/h_{\text{new}}$ can be updated by scalar multiplications.

(ii) If the step succeeds and the step size is doubled, then $e^{Ah_{\text{new}}} = (e^{Ah_{\text{old}}})^2$ can be obtained by matrix multiplications. Furthermore, any reasonable step size selection mechanism should not cause frequent reductions of step size, so the overhead resulting from the computation of e^{Ah} is expected to be small in the long run.

(iii) When the classical implicit linear multistep scheme with step size h

$$\sum_{i=0}^{k} \alpha_i y_{n-i+1} = h \sum_{i=0}^{k} \beta_i f_{n-i+1} \tag{3.26}$$

is applied to solve problem (1.1), at each step one has to solve the nonlinear equation

$$y_{n+1} = h\beta_0 f(x_{n+1}, y_{n+1}) + \omega \tag{3.27}$$

where

$$f(x, y) = Ay + g(x, y) \quad \text{and} \quad \omega = h \sum_{i=1}^{k} \beta_i f_{n-i+1} - \sum_{i=1}^{k} \alpha_i y_{n-i+1}$$

(ω is a known quantity at the current state). In order to apply the contraction mapping theorem, it is required to have

$$k = h|\beta_0| L_f < 1 \tag{3.28}$$

where L_f is the Lipschitz constant for f. In constrast, when an implicit multistep scheme (2.2) with step size $\hat{h}$ is applied, the nonlinear equation to be solved becomes

$$y_{n+1} = \hat{h}\Phi_{k0} g(x_{n+1}, y_{n+1}) + \hat{\omega} \tag{3.29}$$

where $\hat{\omega} = \hat{h}\sum_{i=1}^{k} \Phi_{ki} g_{n-i+1} - e^{A\hat{h}} y_n$. Again we require the condition

$$\hat{k} = \hat{h} \|\Phi_{k0}\| L_g < 1. \tag{3.30}$$

Under the assumption that the iterative method used to solve (3.27) and (3.29) converges at the same rate, it is seen that

$$\frac{\hat{h}}{h} = \frac{|\beta_0|}{\|\Phi_{k0}\|} \frac{L_f}{L_g}. \tag{3.31}$$

Thus for the problem (1.1) where L_f is much greater than L_g, the step size $\hat{h}$ of our scheme (2.2) can be chosen significantly larger than that of the linear scheme (3.26).

(iv) Apart from the facts that ω_{iq}'s are matrices, ϕ_i's are defined over g, and ω_{iq}'s for $i = 1$ or 2 are generated according to (3.21), symbolically (3.17) and (3.1) are exactly the same as those in [26]. Therefore, the efficient software designs of the code STEP can be perfectly transfered into our code. This advantage includes the substantial reduction of the overhead and the memory storages. In particular, the PECE mode (2.8) which advances from x_n to x_{n+1} is now given by

Computing $\omega_{i,1}$, $i = 1, \ldots, k+1$;

P: $\phi_i^*(n) = \beta_i(n+1)\phi_i(n), \quad i = 1, \ldots, k,$

$$p_{n+1} = e^{Ah_{n+1}} y_n + h_{n+1} \sum_{i=1}^{k} \omega_{i,1} \phi_i^*(n),$$

$$\phi_{k+1}^{e}(n+1) = 0,$$

$$\phi_i^{e}(n+1) = \phi_{i+1}^{e}(n+1) + \phi_i^*(n), \quad i = k, \ldots, 1;$$

E: $g_{n+1}^{p} = g(x_{n+1}, p_{n+1})$;

$$\begin{aligned}
\text{C:}\quad & y_{n+1} = p_{n+1} + h_{n+1}\omega_{k+1,1}\left(g_{n+1}^{p} - \phi_1^{e}(n+1)\right);\\
\text{E:}\quad & g_{n+1} = g(x_{n+1}, y_{n+1}),\\
& \phi_{k+1}(n+1) = g_{n+1} - \phi_1^{e}(n+1),\\
& \phi_i(n+1) = \phi_i^{e}(n+1) + \phi_{k+1}(n+1), \quad k = 1,\dots,1.
\end{aligned} \tag{3.32}$$

4. Numerical examples

Test results of problems representing both linear and nonlinear stiff systems are presented in this section. We also give an 'informal' comparison of our algorithm NLMSUB with Gear's DIFSUB or Lawson's IRKSUB by listing their test data from the reference. It should be emphasized that the principal objective of these tests is to confirm the workability of this multistep method rather than to make the detailed comparison with available methods, although several results seem to reveal some significant advantages.

The subroutine NLMSUB is developed based on the prototype code STEP. We replace the scalar coefficients in STEP by matricial coefficients which are generated according to the scheme (3.17) and (3.21). The matrix exponential e^{Ah} is calculated by Ward's method [25,33]. Finally and most importantly, the error control mechanism is modified so as to fit into our matrix formulation. The actual treatments would be too tedious to discuss here and we would rather avoid the details.

Problem 1 [18].

$$y' = U\begin{bmatrix} 0, & 1, & 0, & 0\\ -1, & 0, & 0, & 0\\ 0, & 0, & -100, & -900\\ 0, & 0, & 900, & -100\end{bmatrix} U^{T}y + u\begin{bmatrix} x^2+2x\\ x^2-2x\\ -800x+1\\ -1000x-1\end{bmatrix}; \qquad y(0) = \begin{bmatrix}1\\0\\0\\1\end{bmatrix},$$

$$U = \tfrac{1}{2}\begin{bmatrix} -1, & 1, & 1, & 1\\ 1, & -1, & 1, & 1\\ 1, & 1, & -1, & 1\\ 1, & 1, & 1, & -1\end{bmatrix} \quad \text{and} \quad x \in [0,25].$$

Test results (with local error control 10^{-7}) are given in Table 2.

Notation. The maximum error ERR is measured by $-\log(\Sigma[y_i(x) - y_i]^2)^{1/2}$ on the CYBER 750 system.

Remarks. This problem fits exactly into our model and hence we expect accurate numerical outputs. The large angles formed by its eigenvalues ($-100 \pm 900i$ and $\pm i$) with the negative real axis give Gear's method some hardtime as is expected. The matrix U is used to couple all components and hence makes the problem more complicated. The computation in our algorithm actually terminated at $x = 39.21$ instead of 25 whereas the next allowable step size was as large as 39. So conceivably the above data is to be even more prominent if the integration range is [0,80]. Since the numerical difficulty for a stiff system usually appears in the steady-state part, our method seems to be more efficient for at least this class of problems.

Table 2

	DIFSUB	IRKSUB	NLMSUB
ERR	4.55	4.22	6.75
STEP	2982	532	25
FNS	8332	2986	51
JACOB	37	41	0
PADE	0	0	1
TIME	106.03	41.88	0.19

Table 3

	DIFSUB	IRKSUB	NLMSUB
ERR	5.31	5.40	5.23
STEP	650	1575	286
FNS	1839	18805	322
JACOB	66	319	0
INV	66	957	37
PADE	0	0	36
TIME	0.597	6.171	1.776

Problem 2 [7].

$$y' = \begin{bmatrix} -1, & 0, & 0, & 0 \\ 0, & -10, & 0, & 0 \\ 0, & 0, & -40, & 0 \\ 0, & 0, & 0, & -100 \end{bmatrix} y + \begin{bmatrix} 2 \\ 20y_1^2 \\ 80(y_1^2 + y_2^2) \\ 200(y_1^2 + y_2^2 + y_3^2 \end{bmatrix}; \qquad y(0) = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

where $x \in [0,20]$.

Test results (with local error control 10^{-6}) are given in Table 3.

Remarks. This problem has a very strong nonlinear coupling from the smooth components to the transient components and hence causes great difficulty. In fact, this coupling causes the trasient components to have very large positive derivative at the initial phase. This tendency eventually is diminished and the order is lowered when the smooth components cannot match the rapid decay of the transient. It is probably due to this reason that the step size has to be changed frequently.

Problem 3 [15].

$$y' = U \begin{bmatrix} -\beta_1, & \beta_2, & 0, & 0 \\ -\beta_2, & -\beta_1, & 0, & 0 \\ 0, & 0, & -100, & 0 \\ 0, & 0, & 0, & -0.1 \end{bmatrix} U^T y + U \begin{bmatrix} \dfrac{z_1^2 - z_2^2}{2} \\ z_1 z_2 \\ z_3 \\ z_4^2 \end{bmatrix}; \qquad y(0) = U \begin{bmatrix} -2 \\ 0 \\ -1 \\ -1 \end{bmatrix}$$

where $z = Uy$ and $x \in [0,50]$.

Test results (with local error control 10^{-4}) are given in Table 4.

Remarks. It can be shown that eigenvalues of the Jacobian matrix of the above system are given by $z_1 - \beta_1 \pm |z_2 - \beta_2| i$, $2z_3 - 100$ and $2z_4 - 0.1$. By varying values of β_1 and β_2, this nonlinear system will present different characteristics. When $\beta_1 = -10$ and $\beta_2 = 0$, all eigenvalues are real numbers because $z_2(x) \equiv 0$. One has to note, however, that this actually is a mixed-type problem because of the presence of positive eigenvalues in the transient region. If $\beta_1 = 1$ and $\beta_2 = 100$, then all eigenvalues have negative real parts and the first two form very large angles (nearly 88°) with the negative x-axis. The accuracy returned is satisfactory, but the step sizes used are relatively small (average 6.2×10^{-2}). When $\beta_1 = 10$ and $\beta_2 = 100$, we have similar distribution of eigenvalues (with angles nearly 84°), but the step sizes used are relatively large (average 5.2×10^{-1}). The reason for this contrast is not clear. If $\beta_1 = -10$ and $\beta_2 = 10$, then again this is a mixed-type problem of which some eigenvalues are moving from the right half-plane to the left half-plane. The result shows a frequent reduction on the step size whereas the 'global error' implies a larger step size could have been used. Probably this is due to the deficiency of the error control mechanism that we used.

Table 4

	$\beta_1 = -10$ $\beta_1 = 0$	$\beta_1 = 1$ $\beta_2 = 100$	$\beta_1 = 10$ $\beta_2 = 100$	$\beta_1 = -10$ $\beta_2 = 10$
ERR	2.84	3.45	3.70	5.35
STEP	63	809	96	1866
FNS	127	1619	195	3933
INV	4	3	6	200
PADE	3	2	5	199
TIME	0.440	2.711	0.667	17.942

Acknowledgment

The author is indebted to the anonymous referee for his many constructive comments and suggestions.

References

[1] G. Birkhoff and R. Varga, Discretization errors for well set Cauchy problems, I, *J. Math. Phys.* **44** (1965) 1–23.
[2] J.L. Blue and H.K. Gummel, Rational approximation to matrix exponential for systems of stiff differential equations, *J. Comput. Phys.* **5** (1970) 70–83.
[3] D. Calahan, Numerical solution of linear system with widely separated time constants, *Proc. IEEE* **55** (1967) 2016–2017.
[4] J. Certaine, The solution of ordinary differential equations with large time constants, in: A. Raston and H. Wilf, Eds., *Mathematical Methods for Digital Computers* (Wiley, New York, 1960) 128–132.
[5] G.G. Dahlquist, A special stability problem for linear multistep method, *BIT* **3** (1963) 27–43.
[6] B.L. Ehle and J.D. Lawson, Generalized Runge–Kutta processes for stiff initial-value problems, *J. Inst. Math. Appl.* **16** (1975) 11–21.
[7] W.H. Enright, T.E. Hull and B. Lindberg, Comparing numerical methods for stiff systems of O.D.E.s, *BIT* **15** (1975) 10–48.
[8] W.H. Enright, On the efficient and reliable numerical solution of large linear system of ODEs, *IEEE Trans. Automat. Control* **24** (1979) 905–910.
[9] W. Fair and Y.L. Luke, Padé approximation to the operator exponential, *Numer. Math.* **14** (1970) 379–382.
[10] C.W. Gear, The automatic integration of stiff ordinary differential equations, in: A.J.H. Morrell, Ed. *Information Processing 68* (North-Holland, Amsterdam, (1968) 187–193.
[11] C.W. Gear, *Numerical Initial Value Problems in Ordinary Differential Equations* (Prentice-Hall, Englewood Cliffs, NJ, 1971).
[12] C.W. Gear, Numerical solution of ordinary differential equations: is there anything left to do?, *SIAM Rev.* **23** (1978) 10–24.
[13] R.K. Jain, Some A-stable methods for stiff ordinary differential equations, *Math. Comp.* **26** (1972) 71–77.
[14] F. Krough, A variable step variable order multistep method for the numerical solution of ordinary differential equations, in A.J.H. Morrell, Ed., *Information Processing 68* (North-Holland, Amsterdam, 1968) 194–199.
[15] F. Krough, On testing a subroutine for numerical integration of ordinary differential equations, *J. ACM* **20** (1973) 545–562.
[16] J.D. Lambert and S.T. Sigurdsson, Multistep methods with variable matrix coefficient, *SIAM J. Numer. Anal.* **9** (1972) 715–734.
[17] J.D. Lambert, *Computational Methods in Ordinary Differential Equations* (Wiley, London, 1976).
[18] J.D. Lawson, Generalized Runge–Kutta processes for stable system with large Lipschitz constant, *SIAM J. Numer. Anal.* **4** (1967) 372–380.
[19] J.D. Lawson, Some numerical methods for stiff ordinary and partial differential equations, in: R.S.D. Thomas and H.C. Williams, Eds., *Proc. Second Manitoba Conference on Numer. Math.* (Utilitas Mathematica, Winnipeg, 1972) 27–34.
[20] D. Lee and S. Preiser, A class of nonlinear multistep A-stable numerical methods for solving stiff differential equations, *Comput. Math. Appl.* **4** (1978) 43–51.
[21] W. Liniger and R. Willoughby, Efficient integration methods for stiff systems or ordinary differential equations, *SIAM J. Numer. Anal.* **7** (1970) 47–66.
[22] W. Miranker, Matricial difference scheme for integrating stiff systems of ordinary differential equations, *Math. Comput.* **25** (1971) 717–728.
[23] W. Miranker, *Numerical Methods for Stiff Equations and Singular Perturbation Problems* (Reidel, Dordrecht, 1981).
[24] W.D. Murphy, Hermit interpolation and A-stable methods for stiff ordinary differential equations, *Appl. Math. Comput.* **3** (1977) 103–112.
[25] C. Molen and C. van Loan, Nineteen dubious ways to compute the exponential of a matrix, *SIAM Rev.* **20** (1978) 801–836.
[26] L.F. Shampine and M.K. Gordon, *Computer Solution of Ordinary Differential Equations; the Initial Value Problem* (Freeman, San Francisco, CA, 1975).
[27] L.F. Shampine and C.W. Gear, A user's view of solving stiff ordinary differential equations, *SIAM Rev.* **21** (1979) 1–17.
[28] L.F. Shampine, Stiffness and non-stiff differential equation solvers, in: L. Collatz, Ed., *Numerische Behandling von Differentialgleichungen, Internat. Series Numer. Math.* **27** (Birkhäuser, Basel, 1975) 287–301.
[29] L.F. Shampine, Stiffness and nonstiff differential equation solver, II, detecting stiffness with Runge–Kutta method, *ACM Trans. Math. Software* **3** (1977) 4–53.
[30] J.M. Varah, Stiffly stable linear mulststep method of extended order, *SIAM J. Numer. Anal.* **15** (1978) 1234–1246.
[31] R.S. Varga and E.B. Saff, *Padé and Rational Approximation, Theory and Application* (Academic Press, New York, 1977).
[32] R.S. Varga, On higher order stable implicit method for solving parabolic partial differential equations, *J. Math. Phys.* **40** (1961) 220–321.
[33] R.C. Ward, Numerical computation of the matrix exponential with accuracy estimate, *SIAM J. Numer. Anal.* **14** (1977) 600–610.
[34] G. Wanner, E. Hairer and S.P. Norsett, Order stars and stability theorems, *BIT* **18** (1978) 475–489.
[35] J.H. Wilkinson and C. Reinsch, *Linear Algebra* (Springer, Berlin, 1971).
[36] R.A. Willoughby, *Stiff Differential Systems* (Plenum Press, New York, 1974).