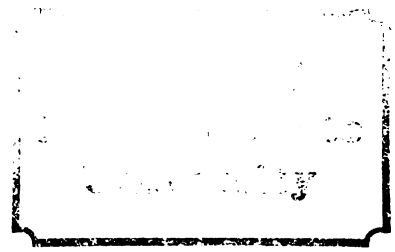


This is to certify that the

dissertation entitled

A NONLINEAR MULTISTEP METHOD FOR SOLVING STIFF INITITAL VALUE PROBLEMS

presented by

Moody Ten-Chao Chu

has been accepted towards fulfillment
of the requirements for

Ph.D. degree in Mathematics

Major professor

Date 5/12/82

A NONLINEAR MULTISTEP METHOD FOR SOLVING STIFF INITIAL VALUE PROBLEMS

By

Moody Ten-Chao Chu

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Department of Mathematics

1982

ABSTRACT

A NONLINEAR MULTISTEP METHOD
FOR SOLVING STIFF INITIAL VALUE PROBLEMS

By

Moody Ten-Chao Chu

A nonlinear multistep method which bears the nature of the classical Adams-Bashforth-Moulton PC formula is developed to solve stiff initial value problems of the form $y' = Ay + g(x,y)$.

It is shown that this method has properties of consistency, convergence and A-stability in the sense of Dahlquist. Several newly developed numerical techniques have been incorporated into this algorithm. A detailed analysis of its structure is also presented to enable us to implement this method in such a way that the step size and the order can be adjusted automatically.

Numerical results from extensive tests by a PECE mode of this method shows its efficiency and several advantages, such as no Jacobian evaluation is needed, much larger step sizes can be used and only a few matirx inversions are involved.

TO

MY PARENTS

AND

MY WIFE

ACKNOWLEDGMENTS

I want to express my deep gratitudes to Professor Tien-Yien Li, my thesis advisor, for all his patience, guidance, and encouragement during the course of my research. His expert advice, useful insight, and stimulating discussions made this work possible.

I also want to thank my wife, Joyce, for her care, patience, and understanding displayed throughout the duration of my graduate studies.

Finally, I am deeply grateful to my parents from whom I learned the virtues of wisdom and faith.

TABLE OF CONTENTS

1. INTRODUCTION

Conventional linear multistep method has been recognized as one of the most effective ways to solve a general initial value problem

$$y' = f(x,y); \quad y(a) = y_0; \quad x \in [a,b] \ . \qquad (1.1)$$

In the last two decades several very sophisticated and reliable codes [12,17,28], based on a variable-step variable-order formulation of the classical Adams method, have been established so that (1.1) can be solved both easily and cheaply. Nevertheless, when applied to stiff systems, the efficiency of these codes becomes very limited because impractically small step sizes must be adopted to ensure the stability [12,19] of the approximate solutions. This difficulty is inherent in the method itself [7] and hence cannot be overcome by any improvement in the computer capacity. On the other hand, most of the methods used for solving stiff systems are implicit of necessity [7,19,29] and hence demand the use of some Newton-like iterations [13] which usually are very expensive in Jacobian evaluations.

The object of this dissertation is to develop a method which, inheriting all the merit of the classical predictor-corrector schemes and being A-stable [7], does not have those difficulties mentioned above when applied to stiff systems.

The search for effective methods to solve stiff systems has received considerable attentions since two decades ago. Brief survey of the literature can be found in Shampine and Gear [29], Enright, Hull and Lindberg [9], Willoughby [37], Bjurel, Dahlquist, Lindberg and Linde [2].

In what follows we shall consider the stiff initial value problem of the form

$$y' = Ay + g(x,y); \quad y(a) = y_0; \quad x \in [a,b] \tag{1.2}$$

where A is an $n \times n$ real constant matrix with all its eigenvalues having negative real parts and $\|\frac{\partial g}{\partial y}\|$ is small relative to $\|A\|$. Notice that a much larger class of problems

$$y' = A(x)y + g(x,y); \quad y(a) = y_0, \tag{1.3}$$

including the linearization of Problem (1.1) near a particular solution, can always be decomposed into

$$y' = Ay + \{(A(x) - A)y + g(x,y)\}; \quad y(a) = y_0$$

where A is chosen to keep $\|A(x) - A\|$ uniformly small at least for a short period of x. So the method developed for Problem (1.2) can still be used to solve stiff system (1.1) or (1.3), provided some prior knowledge about the problem is known.

We begin in Chapter 2 with the main formulation of our nonlinear multistep algorithm. Then after a generalized Adams-Bashforth-Moulton PECE mode is defined, we prove three theorems concerning its consistency, convergence and A-stability. These theorems can be generalized easily to the variable-step variable-order case [28]. Also included in Chapter 2 are some numerical techniques for the calculations of the matrix coefficients of our scheme.

In Chapter 3, we derive a variable-step PECE mode and an algorithm which enables us to compute all the matrix coefficients to any desired order in a very efficient way. In the special case when $A = 0$, this derivation corresponds exactly to that by Shampine and Gordon [28].

Error estimation is given in Chapter 4. We also show how these estimates can be calculated without too much effort. Also treated in Chapter 4 is an automatic control mechanism which is used in selecting step sizes and orders throughout all numerical tests given in this dissertation.

A flowchart consisting of four blocks is provided in Chapter 5 to suggest how our algorithm should be organized and implemented.

In Chapter 6, we give several test results to evidence the effectiveness of our method. In particular, we make comparisons to some existing stiff problem solvers.

Finally, conclusions and recommendations for further study are given in Chapter 7.

We conclude this chapter with a brief historic review of the method we used. The very original idea of our approach for the step number not higher than two was proposed by Certaine [6], but has been regarded as computationally costly because of the difficulty in obtaining the matrix coefficients [25]. Lawson [20] attempted to remove the stiffness of the problem by performing a transformation of the differential equation and then solved the resulting equation with a standard method. This approach is essentially the same as Certaine's method. Some algorithms to compute the matrix coefficients are also suggested. Lee and Preiser [22] tried to carefully select the matrix coefficients for a certain numerical scheme so as to guarantee the consistency of the resulting scheme. In this way several formulas were given. They can be justified to be exactly the same as those obtained from Certaine's method. All these formulations are special cases of our consideration. However, the computational cost can be reduced substantially because of the algorithm presented in Chapter 3 and the discovery of an efficient numerical technique in computing matrix exponentials. Several other approaches considered

by Jain [15], Miranker [24], Murphy [26], Lambert and Sigurdsson [18] are also closely related to our method by certain representations of the integrals which will be introduced in Chapter 2.

2. BASIC FORMULATION --- FIXED STEP SCHEME

2.1 PECE MODE

Let x_k denote the k-th mesh point along the variable axis and $h_{k+1} = x_{k+1} - x_k$. In what follows we shall use y_k to represent a numerical approximation to the exact solution $y(x_k)$ at the mesh point x_k, and let $g_k = g(x_k, y_k)$. By considering the variation of constant formula, the solution to Problem (1.2) satisfies

$$y(x_{n+1}) = e^{Ah_{n+1}} y(x_n) + e^{Ax_{n+1}} \int_{x_n}^{x_{n+1}} e^{-A\tau} g(\tau, y(\tau)) d\tau \ . \quad (2.1.1)$$

Define $\alpha = \dfrac{x}{x_{n+1} - x_n}$, then we are led to form the following formula

$$y_{n+1} = e^{Ah_{n+1}} y_n + h_{n+1} \sum_{i=0}^{k} \Phi_{ki} g_{n-i+1} \quad (2.1.2)$$

where the summation is an approximation to the integral

$$\int_0^1 e^{A(1-\alpha)h_{n+1}} g(x_n + \alpha h_{n+1}, y(x_n + \alpha h_{n+1})) d\alpha \quad (2.1.3)$$

and Φ_{ki}'s are some matrix coefficients to be determined.

We shall consider the fixed-step explicit scheme first. We abbreviate the notation h_{n+1} as h since constant step size is to be used. Assuming $\Phi_{k0} = 0$ in (2.1.2)

and regarding g as a function of α, a vector-valued polynomial $P(\alpha)$ of degree $< k$ is used to interpolate g componentwise at point $x_{n-k+1},\ldots,x_n$ such that $P(1-i) = g_{n-i+1}$ for $i = 1,\ldots,k$. The polynomial given by the Lagrangian formula is

$$P(\alpha) = \sum_{i=1}^{k} g_{n-i+1} \prod_{\substack{j=1 \\ j\neq i}}^{k} \frac{\alpha+j-1}{j-i} . \tag{2.1.4}$$

It follows that the integral (2.1.3) can be approximated by

$$\int_0^1 e^{A(1-\alpha)h} P(\alpha)d\alpha = \sum_{i=1}^{k} \Big(\int_0^1 e^{A(1-\alpha)h} \prod_{\substack{j=1 \\ j\neq i}}^{k} \frac{\alpha+j-1}{j-i} d\alpha \Big) g_{n-i+1} . \tag{2.1.5}$$

It is, therefore, reasonable to choose Φ_{ki} in (2.1.2) as

$$\Phi_{ki} = \int_0^1 e^{A(1-\alpha)h} \prod_{\substack{j=1 \\ j\neq i}}^{k} \frac{\alpha+j-1}{j-i} d\alpha . \tag{2.1.6}$$

As an example, when $k = 4$, (2.1.2) becomes

$$P_{n+1} = e^{Ah} y_n + h \int_0^1 e^{A(1-\alpha)h} \Big[\frac{(\alpha+1)(\alpha+2)(\alpha+3)}{6} g_n - \frac{\alpha(\alpha+2)(\alpha+3)}{2} g_{n-1} + \frac{\alpha(\alpha+1)(\alpha+3)}{2} g_{n-2} - \frac{\alpha(\alpha+1)(\alpha+2)}{6} g_{n-3} \Big] d\alpha. \tag{2.1.7}$$

Here we use P_{n+1} to denote the "predicted value" of $y(x_{n+1})$. We call the scheme (2.1.2) along with (2.1.6) a generalized Adams-Bashforth formula of order k. If $A = 0$, then (2.1.2) is reduced to a classical linear multistep

scheme. The word "order" will be justified in the next section.

An implicit scheme of order $k+1$ can be formed in exactly the same way when the polynomial $p^*(\alpha)$ interpolates the points $x_{n-k+1},\ldots,x_{n+1}$ with values $p^*(1-\alpha) = g_{n-i+1}$ for $i = 0,\ldots,k$. We denote the resulting matrix coefficients of this implicit scheme by Φ^*_{ki} for $i = 0,\ldots,k$, respectively. It is worth noting that $\sum_{i=0}^{k} \Phi^*_{ki} = \sum_{i=1}^{k} \Phi_{ki} = \int_0^1 e^{A(1-\alpha)h} d\alpha$.

Definition 2.1. By a $(k,k+1)$-order PECE mode we mean the following numerical scheme consisting of a predictor of order k and a corrector of order $k+1$ in the form of (2.1.2), i.e.,

$$\begin{aligned}
&\text{P:} \quad p_{n+1} = e^{Ah} y_n + h \sum_{i=1}^{k} \Phi_{ki}\, g_{n-i+1}\,, \\
&\text{E:} \quad g^p_{n+1} = g(x_{n+1}, p_{n+1})\,, \\
&\text{C:} \quad y_{n+1} = e^{Ah} y_n + h \sum_{i=1}^{k} \Phi^*_{ki}\, g_{n-i+1} + h\, \Phi^*_{k0}\, g^p_{n+1}\,, \\
&\text{E:} \quad g_{n+1} = g(x_{n+1}, y_{n+1})\,.
\end{aligned} \tag{2.1.8}$$

As was pointed out in [27,33] and also will be shown in Section 2.3, it is not an easy and economic task to compute these matrix coefficients. The situation is worsened when variable-step scheme is adopted. In Chapter 3, we develop an algorithm which, taking the problem of saving memory storages

into consideration, enables us to obtain these integrals efficiently.

2.2. MAJOR THEOREMS

We shall prove three major theorems concerning the consistency, the convergence and the A-stability for the fixed-step case, which provide the groundwork of this nonlinear algorithm.

2.2.1. CONSISTENCY

With the nonlinear multistep method (2.1.2), we give the following definition.

Definition 2.2. The local truncation error at x_{n+1}, denoted by $\mathcal{L}(y(x_n),h)$, of the method (2.1.2) is defined to be

$$\mathcal{L}(y(x_n),h) = y(x_{n+1}) - e^{Ah}y(x_n) - h \sum_{i=0}^{k} \Phi_{ki}\, g(x_{n-i+1},y(x_{n-i+1})).$$

The notation $\mathcal{L}^p(y(x_n),h)$ will be used to represent the local truncation error associated with an explicit scheme, while $\mathcal{L}(y(x_n),h)$ represents that associated with an implicit scheme. A numerical scheme is said to be consistent if its associated local truncation error is at least $O(h)$.

Theorem 2.1. Assume $g \in C^{k+1}$, then the local truncation error $\mathcal{L}^p(y(x_n),h)$ of the explicit scheme (2.1.2) with step number k is $O(h^{k+1})$.

Proof: Recall that

$$\Phi_{ki} = \int_0^1 e^{A(1-\alpha)h} \ell_i(\alpha)dA \qquad (2.2.1.1)$$

where $\ell_i(\alpha) = \prod_{\substack{j=1 \\ j\neq i}}^{k} \frac{\alpha+j-1}{j-i}$. So

$$\begin{aligned}
&\mathcal{L}^p(y(x_n),h) \\
&= y(x_{n+1}) - e^{Ah}y(x_n) - h \sum_{i=1}^{k} \Phi_{ki}\, g(x_{n-i+1}, y(x_{n-i+1})) \\
&= \int_0^1 d[e^{A(1-\alpha)h}y(x_n+\alpha h)] \\
&\qquad - h \sum_{i=1}^{k} [\int_0^1 e^{A(1-\alpha)} \ell_i(\alpha) g(x_{n-i+1}, y(x_{n-i+1}))d\alpha] \\
&= \int_0^1 [-Ahe^{A(1-\alpha)h}y(x_n+\alpha h) + he^{A(1-\alpha)h}y'(x_n+\alpha h)]d\alpha \\
&\qquad - h \int_0^1 e^{A(1-\alpha)h}P(\alpha)d\alpha \\
&= h \int_0^1 e^{A(1-\alpha)h}[g(x_n+\alpha h, y(x_n+\alpha h)) - P(\alpha)] \\
&= h^{k+1} \int_0^1 e^{A(1-\alpha)h} [\frac{g^{(k)}(\xi(\alpha))}{k!} \prod_{i=1}^{k} (\alpha + i - 1)]d\alpha. \qquad (2.2.1.2)
\end{aligned}$$

It follows that

$$\|\mathcal{L}^P(y(x_n),h)\| \le h^{k+1}\|g^{(k)}\|_\infty \|\int_0^1 e^{A(1-\alpha)h} d\alpha\| = O(h^{k+1}). \quad (2.2.1.3)$$

Notice that if all eigenvalues of A have negative real parts, then the spectral mapping theorem implies that $\|e^{A(1-\alpha)h}\| \le 1$ uniformly for some suitable norm and hence $\|\int_0^1 e^{A(1-\alpha)h} d\alpha\| \le 1$. Consequently, $\|\sum_{i=0}^{k} \Phi^*_{ki}\| = \|\sum_{i=1}^{k} \Phi_{ki}\| \le 1$.

Similar argument shows that the local truncation error $\mathcal{L}(y(x_n),h)$ for the implicit scheme (2.1.2) with step number k is $O(h^{k+2})$.

2.2.2. CONVERGENCE

By convergence we refer to "fixed station convergence" which has the following meaning.

Definition 2.3. The nonlinear multistep method (2.1.2) or the PECE mode (2.1.8) is said to be convergent if, for all initial value problems (1.2), we have that

$\lim_{\substack{h\to 0 \\ nh=x-a}} y_n = y(x_n)$ holds for all $x \in [a,b]$ and for all solutions $\{y_n\}$ satisfying starting conditions $y_i = \eta_i(h)$ for which $\lim_{h\to 0} \eta_i(h) = y(a)$, $i = 0,\ldots,k-1$.

In this sense we have the convergence theorem.

Theorem 2.2. Let the $(k,k+1)$-under PECE mode (2.1.8) be applied to solve Problem (1.2) on $[a,b]$. Suppose

$g \in C'$ and $\|\frac{\partial g}{\partial y}\| \leq L_g$ for some constant $L_g \geq 0$. If all starting values y_i satisfy $\|y(x_i) - y_i\| \leq E_0$ for $i = 0,\ldots,k-1$, then for $x_n \in [a,b]$ we have

$$\|y(x_n) - y_n\| \leq [E_0 + \frac{\delta}{h\Lambda}]e^{(x_n-a)\Lambda} \qquad (2.2.2.1)$$

where

$$\Lambda = L_g(\alpha^* + hL_g\|\Phi^*_{k0}\|\alpha) ,$$

$$\delta = \max_n \|h\ \Phi^*_{k0}\ \frac{\partial g}{\partial y}\ \mathcal{L}^p(y(x_n),h) + \mathcal{L}(y(x_n),h)\| , \qquad (2.2.2.2)$$

$$\alpha = \sum_{i=1}^{k} \|\Phi_{ki}\| \quad \text{and} \quad \alpha^* = \sum_{i=0}^{k} \|\Phi^*_{ki}\| .$$

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

Proof: From the definition of local truncation errors, we know

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

$$y(x_{n+1}) = e^{Ah}y(x_n) + h \sum_{i=0}^{k} \Phi^*_{ki}\ g(x_{n-i+1},y(x_{n-i+1})) + \mathcal{L}(y(x_n),h) . \qquad (2.2.2.3)$$

So the mean value theorem implies that the global error at x_{n+1}

$$\tilde{e}_{n+1} = y(x_{n+1}) - y_{n+1} \qquad (2.2.2.4)$$

is given by

$$\tilde{e}_{n+1} = e^{Ah}\tilde{e}_n + h \sum_{i=1}^{k} \Phi^*_{ki} \frac{\partial g}{\partial y}(x_{n-i+1}, \xi_{n-i+1})\tilde{e}_{n-i+1}$$

$$+ h \Phi^*_{k0} \frac{\partial g}{\partial y}(x_{n+1}, \xi_{n+1})[e^{Ah}\tilde{e}_n +$$

$$+ h \sum_{i=1}^{k} \Phi_{ki} \frac{\partial g}{\partial y}(x_{n-i+1}, \xi_{n-i+1})\tilde{e}_{n-i+1}] + \delta_n \qquad (2.2.2.5)$$

where $\delta_n = h \Phi^*_{k0} \frac{\partial g}{\partial y}(x_{n+1}, \xi_{n+1})\mathcal{L}^p(y(x_n), h) + \mathcal{L}(y(x_n), h)$, ξ_{n-i+1} lies between y_{n-i+1} and $y(x_{n-i+1})$ for $i > 0$; and between p_{n+1} and $y(x_{n+1})$ for $i = 0$. By the conditions on A and $\frac{\partial g}{\partial y}$ we arrive at

$$\|\tilde{e}_{n+1}\| \le \|\tilde{e}_n\| + h L_g \sum_{i=0}^{k} \|\Phi^*_{ki}\| \|\tilde{e}_{n-i+1}\|$$

$$+ h^2 L_g{}^2 \|\Phi^*_{k0}\| \sum_{i=1}^{k} \|\Phi_{ki}\| \|\tilde{e}_{n-i+1}\| . \qquad (2.2.2.6)$$

Define $\chi = 1 + h L_g \alpha^* + h^2 L_g^2 \|\Phi^*_{k0}\| \alpha$ and $E_i = \chi E_{i-1} + \delta$, then obviously $E_{i-1} < E_i$ for each i. By induction, (2.2.2.6) implies that $\|\tilde{e}_j\| \le E_n$ for $j = 0, \ldots, n$. Since $E_n = \chi^n E_0 + \frac{\chi^{n-1}}{\chi - 1} \delta$, the inequality (2.2.2.1) follows immediately from the facts that $1 + x < e^x$ and $nh = x_n - a$. In particular, if $g \in C^{k+1}$, then the theorem of consistency implies $\delta = O(h^{k+2})$.

The above two theorems justify that the nonlinear multistep method, such as the mode (2.1.8), has the consistency and convergence properties. For the computational purpose, we also need the feature of A-stability.

2.2.3. A-STABILITY

Dahlquist [7] defined a numerical method to be A-stable if its region of absolute stability contains the whole of the left half-plane. Equivalently, a method is A-stable if all numerical solutions y_n tend to zero asymptotically as $n \to \infty$ when it is applied to the differential equation $y' = Ay$ where all eigenvalues of A have negative real parts. To prove the A-stability property of our algorithm, we approximate each matrix exponential in (2.1.2) by its corresponding Padé approximation. Recall the following definition [33].

Definition 2.4. A (p,q)-pair Padé approximation to the matrix exponential e^B is the matrix $E_{pq}(B) = [D_{pq}(B)]^{-1}N_{pq}(B)$ where

$$N_{pq}(z) = \sum_{j=0}^{p} \frac{(p+q-j)!p!}{(p+q)!j!(p-j)!} z^j ,$$

$$D_{pq}(z) = \sum_{j=0}^{q} \frac{(p+q-j)!q!}{(p+q)!j!(q-j)!} (-z)^j . \qquad (2.2.3.1)$$

It is known [1,4,34,36] that the diagonal Padé approximation $E(B)$ of e^B is unconditionally stable, provided that all eigenvalues of B have negative real parts. Therefore, we have the following theorem.

Theorem 2.3. The nonlinear multistep scheme (2.1.2) based on diagonal Padé approximations to all its matrix exponentials is A-stable.

Proof: When applied to any test problem $y' = Ay;\ y(a) = y_0$, the scheme (2.1.2) yields values $y_{n+1} = E(Ah)y_n = [E(Ah)]^{n+1}y_0$ since $g(x,y)$ is identically zero. But all eigenvalues of A have negative real parts, the unconditional stability of $E(Ah)$ implies that $\lim_{n\to\infty} y_n = 0$. This establishes the A-stability of scheme (2.1.2).

2.3 NUMERICAL EVALUATION

In this section we shall describe some special numerical techniques for the evaluations of the matrix exponentials and the matrix coefficients.

2.3.1. COMPUTATIONS OF MATRIX EXPONENTIALS

The exponential of a matrix could be computed in many ways [27]. For our purpose we consider only the rational approximations because of its direct applicability. Nevertheless, for stiff systems, direct application of a rational approximation for the matrix exponential gives very poor results as shown in the contour plot of square errors by Blue and Gummel [3]. In fact, the error in approximating the matrix exponential by diagonal Padé table entries has been shown [10] to be increasing as the norm of the matrix increases. Therefore, the first objective of any algorithm involving Padé approximations is to decrease the matrix norm. Since exponentials satisfy $e^a = (e^{a/b})^b$, one may always use this property to reduce the matrix norm and improve the

accuracy of the approximation. Ward [33,35] has given an error estimate for diagonal Padé approximation of order up to $P = 15$ and made it possible to return the minimum number of digits of accuracy in the norm to the user. The algorithm for computing e^A can be summarized as follows:

(i) Compute the mean of eigenvalues $\bar{\lambda} = \frac{\sum_{i=1}^{n} a_{ii}}{n}$.

(ii) Make the translation $A_1 = A - \bar{\lambda}I$.

(iii) Balance the matrix A_1 to obtain $A_2 = D^{-1}P^T A_1 PD$, for which, $\|A_2\|_1 = \min_{D \in \mathcal{D}_\beta} \|D^{-1}A_1 D\|$, where $\mathcal{D}_\beta$ is the set of all nonsingular diagonal matrices with entries as integral powers of the machine base β and P is some preliminary permutation matrix.

(iv) If $\|A_2\|_1 \leq 1$, set $A_3 = A_2$, $m = 0$ and go to (vi).

(v) Find an integer $m > 0$ such that $\|A_2\| \leq 2^m$. Then define $A_3 = \frac{A_2}{2^m}$.

(vi) Use Padé diagonal table to compute e^{A_3}.

(vii) If (v) is skipped, go to (ix).

(viii) Square e^{A_3} m times, i.e. compute $(e^{A_3})^{2^m}$.

(ix) Compute $e^A = e^{\bar{\lambda}} PD(e^{A_3})^{2^m} D^{-1}P^T$.

This algorithm works quite satisfactorily in general. However, there are some numerical "barrier" where care should be taken. A good source of this material can be found in Molen and van Loan [27].

2.3.2. COMPUTATIONS OF INTEGRALS INVOLVING MATRIX EXPONENTIALS

To compute the matrix coefficients defined in (2.1.7), it is necessary to compute the following integral:

$$M_0 = \int_0^1 e^{A(1-\alpha)h} d\alpha \ ,$$
$$M_i = \int_0^1 e^{A(1-\alpha)h} \alpha^i d\alpha \qquad i = 1,2,\ldots \ . \tag{2.3.2.1}$$

Observe that these matrices satisfy the following recursive relations

$$hAM_i = iM_{i-1} - I \ . \tag{2.3.2.2}$$

If A^{-1} exists, then $M_0 = (Ah)^{-1}(e^{Ah} - I)$ and all the other M_i's can be induced by (2.3.2.2). But for stiff systems, the condition number $k(A) \geq \frac{\max |\lambda_i|}{\min |\lambda_i|}$ is large. The a priori error estimate

$$\frac{\|A_c^{-1} - A^{-1}\|}{\|A^{-1}\|} \leq \frac{k(A)}{1 - k(A) \frac{\|A_c - A\|}{\|A\|}} \tag{2.3.2.3}$$

indicates that the computed A_c^{-1} might be in large relative error. Furthermore, these M_i's still exist even if A is singular. So we introduce two methods to obtain them.

The first method works for a normal matrix A and involves its generalized inverse matrix A^+. Recall that the matrix A is normal if and only if $A^*A = AA^*$ where A^* is its corresponding adjoint matrix.

Observe that M_0 satisfies the linear system

$$(Ah)X = e^{Ah} - I \ . \qquad (2.3.2.4)$$

Let $B = Ah$, and let the singular value decomposition of B be $V\begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix}U^*$ where V and U are orthogonal matrices formed from eigenvectors of BB^* and B^*B, respectively; $\Sigma = \text{diag}(\lambda_1,\ldots,\lambda_r)$ where $\lambda_i \neq 0$ for $i = 1,\ldots,r$ are eigenvalues of B^*B; and $v_i = \frac{1}{\lambda_i} Bu_i$ for $i = 1,\ldots,r$, where v_i and u_i are column vectors of V and U, respectively. It is known that $B^+ = U\begin{bmatrix} \Sigma^{-1} & 0 \\ 0 & 0 \end{bmatrix}V^*$ is the generalized inverse of B and

$$\tilde{X}_0 = B^+(e^{Ah} - I) \qquad (2.3.2.5)$$

is a least Frobenius solution to (2.3.2.4).

Theorem 2.4. If B is normal, then

$$M_0 = \tilde{X}_0 + (I - B^+B) \ . \qquad (2.3.2.6)$$

Proof. Since B is normal, it is known that $Bu = \lambda u$ if and only if $B^*u = \bar{\lambda}u$. This implies $B^*Bu = |\lambda|^2 u$ and $BB^* = |\lambda|^2 u$. So the same orthonormal basis $u_1,\ldots,u_n$ which diagonalizes B into $D = \mathrm{diag}(\lambda_1,\ldots,\lambda_r,0,\ldots,0)$ can also serve as an orthonormal basis for both B^*B and BB^*. Hence B has a singular value decomposition $B = U\begin{bmatrix}\Sigma & 0\\ 0 & 0\end{bmatrix}U^*$ and a generalized inverse $B^+ = U\begin{bmatrix}\Sigma^{-1} & 0\\ 0 & 0\end{bmatrix}U^*$. Dente U by $[U_1,U_2]$ where $U_1 = [u_1,\ldots,u_r]$ and $U_2 = [u_{r+1},\ldots,u_n]$, then

$$M_0 = \int_0^1 e^{B(1-\alpha)}d\alpha = U\left(\int_0^1 e^{D(1-\alpha)}d\alpha\right)U^*$$

$$= [U_1,U_2]\begin{bmatrix}\Sigma^{-1}(e^{\Sigma}-I), & 0\\ 0 & , I\end{bmatrix}\begin{bmatrix}U_1^*\\ U_2^*\end{bmatrix}$$

$$= U_1\Sigma^{-1}(e^{\Sigma}-I)U_1^* + U_2U_2^* \quad . \qquad (2.3.2.7)$$

On the other hand,

$$\tilde{X}_0 = B^+(e^B - I) = UD^+U^*U(e^D - I)U^* = UD^+(e^D - I)U^*$$

$$= U_1\Sigma^{-1}(e^{\Sigma}-I)U_1^* \quad . \qquad (2.3.2.8)$$

So (2.3.2.5) follows from (2.3.2.7), (2.3.2.8) and the following identity

$$I - B^+B = I - UD^+U^*UDU^* = I - UD^+DU^* = U_2U_2^* \quad . \qquad (2.3.2.9)$$

Corollary 2.1. If B is normal, then

$$M_n = B^+(nM_{n-1} - I) + \frac{1}{n+1}(I - B^+B) . \qquad (2.3.2.10)$$

Proof. By definition of (2.3.2.1) and the identity (2.3.2.9), we see

$$M_n = \int_0^1 e^{B(1-\alpha)}\alpha^n d\alpha = U(\int_0^1 e^{D(1-\alpha)}\alpha^n d\alpha)U^*$$

$$= U\begin{bmatrix} \Sigma^{-1}(n\int_0^1 e^{\Sigma(1-\alpha)}\alpha^{n-1}d\alpha - I), & O \\ O & , \frac{1}{n+1}I \end{bmatrix}U^*$$

$$= U\begin{bmatrix} \Sigma^{-1}, & O \\ O & O \end{bmatrix}U^*U\begin{bmatrix} n\int_0^1 e^{\Sigma(1-\alpha)}\alpha^{n-1}d\alpha - I, & O \\ O & , O \end{bmatrix}U^* + U\begin{bmatrix} O & O \\ O & \frac{1}{n+1}I \end{bmatrix}U^*$$

$$= B^+(nM_{n-1} - I) + \frac{1}{n+1}(I - B^+B) .$$

The second method works for a general matrix A and involves its Drazin inverse matrix A^d. We first make two definitions.

Definition 2.6. The index of a matrix A is the smallest non-negative integer k such that rank A^k = rank A^{k+1}.

Definition 2.7. The Drazin inverse of A, denoted by A^D, is defined as the matrix $A^D = P\begin{bmatrix} C^{-1} & O \\ O & O \end{bmatrix}P^{-1}$, provided $A = P\begin{bmatrix} C & O \\ O & N \end{bmatrix}P^{-1}$ where C is an invertible submatrix and N is nilpotent of index k.

Note that there are a variety of ways to calculate A^D [5]. In fact, the following lemma can be regarded as the definition of A^D from the algebraic point of view.

Lemma 2.1. If the index of A is k then A^D is the only matrix satisfying the following properties

$$\begin{aligned} &\text{(i)} \quad A^D A A^D = A^D , \\ &\text{(ii)} \quad A A^D = A^D A , \\ &\text{(iii)} \quad A^{k+1} A^D = A^k . \end{aligned} \tag{2.3.2.11}$$

Proof. The proof can be found in [5].

Theorem 2.5. If the index of A is k, then

$$M_0 = B^D(e^B - I) + (I - B^D B) \sum_{i=0}^{k-1} \frac{B^i}{(i+1)!} . \tag{2.3.2.12}$$

Proof. Using the series expansion for e^{Bs} and the above lemma, one can show that

$$\frac{d}{ds}\left[B^D(e^B - I) + (I - B^D B)\, s \sum_{i=0}^{k-1} \frac{B^i s^i}{(i+1)!} \right] = e^{Bs} .$$

Corollary 2.2. If the index of A is k, then

$$M_n = B^D(nM_{n-1} - I) + n(I - B^D B) \sum_{i=0}^{k-1} \frac{(n-1)!B^i}{(n+i+1)!} . \tag{2.3.2.13}$$

3. EFFICIENT IMPLEMENTATION --- VARIABLE STEP SCHEME

3.1. GENERATING COEFFICIENTS

The subroutine STEP by Shampine and Gordon [28] gives an efficient way of generating coefficients for the linear multistep method and a substantial savings of memory storages. In this section we reformulate their settings in the matrix form and hence preserve all the advantages.

We define the following quantities:

$$
\begin{aligned}
h_i &= x_i - x_{i-1} \ , \\
s &= \frac{x-x_n}{h_{n+1}} \ , \\
\psi_i(n+1) &= h_{n+1} + \cdots + h_{n-i+2} \ , \quad i \geq 1 \\
\alpha_i(n+1) &= \frac{h_{n+1}}{\psi_i(n+1)} \ , \ i \geq 1 \ , \\
\beta_1(n+1) &= 1 \ , \\
\beta_i(n+1) &= \frac{\psi_1(n+1)\psi_2(n+1)\cdots\psi_{i-1}(n+1)}{\psi_1(n)\psi_2(n)\cdots\psi_{i-1}(n)} \ , \ i > 1 \ , \\
\varphi_1(n) &= g[x_n] = g_n \ , \\
\varphi_i(n) &= \psi_1(n) \cdots \psi_{i-1}(n) \ g[x_n, x_{n-1}, \cdots, x_{n-i+1}] \quad i > 1 \ .
\end{aligned}
\tag{3.1.1}
$$

Recall that

$$P_{n+1} = e^{Ah_{n+1}} y_n + \int_{x_n}^{x_{n+1}} e^{A(x_{n+1}-\tau)} P_{k,n}(\tau)d\tau \qquad (3.1.2)$$

where $P_{k,n}$ is a polynomial of degree $< k$ and $P_{k,n}(x_{n-i+1}) = g_{n-i+1}$ for $i = 1,\cdots,k$. Since the interpolating polynomial is given by

$$\begin{aligned} &P_{k,n}(x) \\ &= g[x_n] + (x - x_n)g[x_n,x_{n-1}] + \cdots + \\ &\quad (x - x_n) \cdots (x - x_{n-k+2})g[x_n,\cdots,x_{n-k+1}] , \end{aligned} \qquad (3.1.3)$$

a typical term, for $i \geq 2$, of $P_{k,n}(x)$ can be written as

$$\begin{aligned} &(x - x_n) \cdots (x - x_{n-i+2})g[x_n,\cdots,x_{n-i+1}] \\ &= (sh_{n+1})(sh_{n+1} + h_n) \cdots \\ &\quad \cdots (sh_{n+1} + h_n + \cdots + h_{n-i+3}) \frac{\varphi_i(n)}{\psi_1(n)\cdots\psi_{i-1}(n)} \qquad (3.1.4) \\ &= \left(\frac{sh_{n+1}}{\psi_1(n+1)}\right)\left(\frac{sh_{n+1}+h_n}{\psi_2(n+1)}\right) \cdots \\ &\quad \cdots \left(\frac{sh_{n+1} + \psi_{i-2}(n)}{\psi_{i-1}(n+1)}\right) \beta_i(n+1)\varphi_i(n) . \end{aligned}$$

Introduce

$$c_{i,n}(s) = \begin{cases} 1 & i = 1 , \\ \dfrac{sh_{n+1}}{\psi_1(n+1)} & i = 2 , \\ \left(\dfrac{sh_{n+1}}{\psi_1(n+1)}\right)\left(\dfrac{sh_{n+1}+h_n}{\psi_2(n+1)}\right) \cdots \left(\dfrac{sh_{n+1}+\psi_{i-2}(n)}{\psi_{i-1}(n+1)}\right) & i \geq 3 , \end{cases} \qquad (3.1.5)$$

and

$$\varphi_i^*(n) = \beta_i(n+1)\ \varphi_i(n) \ , \tag{3.1.6}$$

then (3.1.3) can be rewritten as

$$P_{k,n}(x) = \sum_{i=1}^{k} c_{i,n}(s)\varphi_i^*(n) \tag{3.1.7}$$

and (3.1.2) becomes

$$P_{n+1} = e^{Ah_{n+1}}y_n + h_{n+1}\sum_{i=1}^{k}\left(\int_0^1 e^{A(1-s)h_{n+1}}c_{i,n}(s)ds\right)\varphi_i^*(n) \ . \tag{3.1.8}$$

Let

$$d_{i,n}(s) = e^{A(1-s)h_{n+1}}c_{i,n}(s) \ , \tag{3.1.9}$$

then it can be shown that

$$d_{i,n}(s) = \begin{cases} e^{A(1-s)h_{n+1}} & i = 1 \ , \\ e^{A(1-s)h_{n+1}}s & i = 2 \ , \\ [\alpha_{i-1}(n+1)s + \dfrac{\psi_{i-2}(n)}{\psi_{i-1}(n)}]d_{i-1,n}(s) & i \geq 3 \ , \end{cases} \tag{3.1.10}$$

and

$$P_{n+1} = e^{Ah_{n+1}}y_n + h_{n+1}\sum_{i=1}^{k}\left(\int_0^1 d_{i,n}(s)ds\right)\varphi_i^*(n) \ . \tag{3.1.11}$$

For fixed n and $i \geq 3$, observe that

$$\int_0^s d_{i,n}(s_0)ds_0$$

$$= \int_0^s [\alpha_{i-1}(n+1)s_0 + \frac{\psi_{i-2}(n)}{\psi_{i-1}(n)}]d(\int_0^{s_0} d_{i-1,n}(s_1)ds_1)$$

$$= [\alpha_{i-1}(n+1)s + \frac{\psi_{i-2}(n)}{\psi_{i-1}(n)}] \int_0^s d_{i-1,n}(s_0)ds_0$$

$$- \int_0^s \alpha_{i-1}(n+1) \int_0^{s_1} d_{i-1,n}(s_0)ds_0ds_1 . \quad (3.1.12)$$

Define

$$d_{i,n}^{(q)}(s) = \int_0^s \int_0^{s_{q-1}} \cdots \int_0^{s_1} d_{i,n}(s_0)ds_0ds_1 \cdots ds_{q-1} , \quad (3.1.13)$$

then (3.1.12) can be rewritten as

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

$$- \alpha_{i-1}(n+1)d_{i\ 1,n}^{(2)}(s) . \quad (3.1.14)$$

In fact, it is not hard to show, by induction and integration by parts, that for $i \geq 3$, the following is true:

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

$$- q\alpha_{i-1}(n+1)d_{i-1,n}^{(q+1)}(s) . \quad (3.1.15)$$

Scaling these quantities by defining

$$w_{i,q} = (q-1)!\, d_{i,n}^{(q)}(1) , \quad (3.1.16)$$

we then arrive at one of the important identities

$$\omega_{i,q} = \omega_{i-1,q} - \alpha_{i-1}^{(n+1)}\omega_{i-1,q+1} \quad \text{for} \quad i \geq 3 . \qquad (3.1.17)$$

Up to this point, all these quantities we have derived are formally the same as those by Shampine and Gordon [28] except that these are in matrix form. When A is identically zero, the quantities defined in (3.1.10), (3.1.12), (3.1.15) and (3.1.17) become diagonal matrices and, in fact, each diagonal matrix is of the form cI where the scalar c is that used by Shampine and Gordon. In this case, the quantities for $i = 1$ and $i = 2$ can be handled trivially. However, if A is a matrix as stated in Problem (1.2), then for $i = 1$, the definition (3.1.13) of $d_{i,n}^{(1)}(s)$ implies that

$$(-Ah_{n+1})d_{1,n}^{(1)}(s) = e^{A(1-s)h_{n+1}} - e^{Ah_{n+1}} , \qquad (3.1.18)$$

and, by induction, for $q \geq 2$,

$$(-Ah_{n+1})d_{1,n}^{(q)}(s) = d_{1,n}^{(q-1)}(s) - \frac{s^{q-1}}{(q-1)!} e^{Ah_{n+1}} . \qquad (3.1.19)$$

Similarly, for $i = 2$, we have

$$(-Ah_{n+1})d_{2,n}^{(1)}(s) = se^{A(1-s)h_{n+1}} - d_{1,n}^{(1)}(s) ,$$

$$(-Ah_{n+1})d_{2,n}^{(q)}(s) = d_{2,n}^{(q-1)}(s) - d_{1,n}^{(q)}(s) \quad \text{for} \quad q \geq 2 . \qquad (3.1.20)$$

Resorting to the definition (3.1.17), we obtain

$$
\begin{aligned}
(-Ah_{n+1})\omega_{1,1} &= I - e^{Ah_{n+1}} , \\
(-Ah_{n+1})\omega_{1,q} &= (q-1)\omega_{1,q-1} - e^{Ah_{n+1}} \quad \text{for} \quad q \geq 2 , \\
(-Ah_{n+1})\omega_{2,1} &= I - \omega_{1,1} , \\
(-Ah_{n+1})\omega_{2,q} &= (q-1)\omega_{2,q-1} - \omega_{1,q} \quad \text{for} \quad q \geq 2 .
\end{aligned}
\tag{3.1.21}
$$

Note that for stiff systems A is nonsingular, so $\omega_{1,q}$ and $\omega_{2,q}$ for every $q \geq 1$ can be solved directly from (3.1.21), once A^{-1} is known. But even if A is singular, the techniques described in section 2.3.2 can be used.

Since $d^{(1)}_{i,n}(1) = \omega_{i,1}$, the scheme (3.1.11) becomes

$$
P_{n+1} = e^{Ah_{n+1}} y_n + h_{n+1} \sum_{i=1}^{k} \omega_{i,1} \varphi^*_i(n) . \tag{3.1.22}
$$

We now make a correction of this predicted value. Recall that

$$
y_{n+1} = e^{Ah_{n+1}} y_n + \int_{x_n}^{x_{n+1}} e^{A(x_{n+1}-\tau)} P^*_{k+1,n}(\tau) d\tau \tag{3.1.23}
$$

where $P^*_{k+1,n}(\tau)$ is a polynomial of degree $< k+1$ with values $P^*_{k+1,n}(x_{n-i+1}) = g_{n-i+1}$ for $i = 1,\cdots,k$ and $P^*_{k+1,n}(x_{n+1}) = g(x_{n+1}, P_{n+1})$. Observe that

$$P^{*}_{k+1,n}(x)$$

$$= P_{k,n}(x) + (x-x_n)\cdots(x - x_{n-k+1})g^{P}[x_{n+1},\cdots,x_{n-k+1}]$$

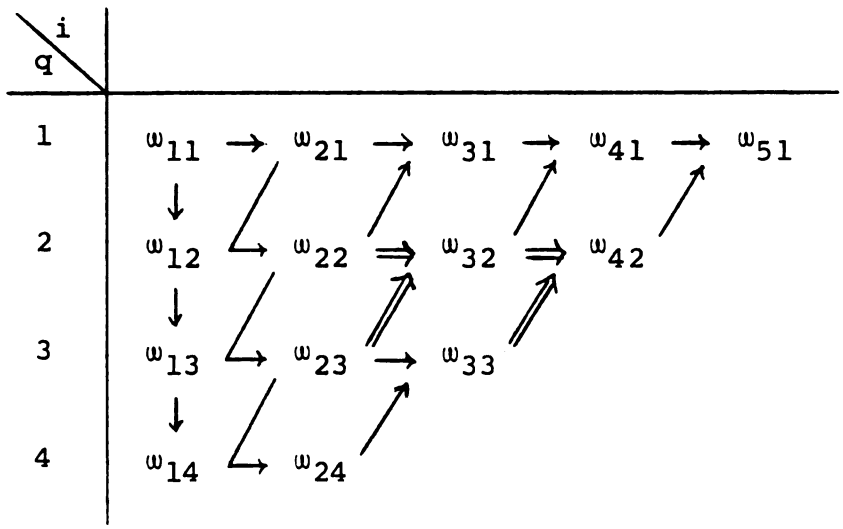
$$= P_{k,n}(x) + (sh_{n+1})\cdots (sh_{n+1} + \cdots + h_{n-k+2}) \frac{\varphi^{P}_{k+1}(n+1)}{\psi_1(n+1)\cdots\psi_k(n+1)}$$

$$= P_{k,n}(x) + c_{k+1,n}(s)\varphi^{P}_{k+1}(n+1) \qquad (3.1.24)$$

where the superscript P is used to indicate that the value g_{n+1} is replaced by $g(x_{n+1},P_{n+1})$. Substituting (3.1.24) into (3.1.23), we see

$$y_{n+1} = P_{n+1} + h_{n+1}\, \omega_{k+1,1}\, \varphi^{P}_{k+1}(n+1) \ . \qquad (3.1.25)$$

We conclude this section by illustrating the interrelation between the coefficients based on (3.1.17) and (3.1.21) for the case $k = 3$ in the following table where arrows emanate from the generators:



3.2. UPDATING DIVIDED DIFFERENCES

At the current point x_n all the quantities $\varphi_i(n)$ for $i = 1,\cdots,k$ are known. To complete the step from x_n to x_{n+1}, we need to know $\varphi^P_{k+1}(n+1)$ as indicated in (3.1.25). To prepare for the advance from x_{n+1} to the next point x_{n+2}, we need to know all the quantities $\varphi_i(n+1)$ for $i = 1,\cdots,k$. Moreover, as will be shown in the next chapter, the quantities $\varphi^P_i(n+1)$ for $i = 1,\cdots,k$ are also useful in the error estimations. In this section we explain how these divided differences can be updated.

First of all, think of $P_{k,n}$ in (3.1.3) as a polynomial of degree k determined by the conditions $P_{k,n}(x_{n-i+1}) = g_{n-i+1}$ for $i = 1,\cdots,k$ and $P_{k,n}(x_{n+1}) = P_{k,n}(x_{n+1})$. The divided difference $\varphi^e_{i+1}(n+1)$ based on the above values at $x_{n+1},x_n,\cdots,x_{n-k+1}$ is given by,

$$
\begin{aligned}
&\varphi^e_{i+1}(n+1) \\
&= \psi_1(n+1)\cdots\psi_i(n+1)g^e[x_{n+1},\cdots,x_{n-i+1}] \\
&= \psi_1(n+1)\cdots\psi_{i-1}(n+1)(x_{n+1}-x_{n-i+1}) \\
&\qquad\qquad \frac{g^e[x_{n+1},\cdots,x_{n-i+2}]-g[x_n,\cdots,x_{n-i+1}]}{x_{n+1}-x_{n-i+1}} \\
&= \psi_1(n+1)\cdots\psi_{i-1}(n+1)g^e[x_{n+1},\cdots,x_{n-i+1}] - \\
&\qquad\qquad - \frac{\psi_1(n+1)\cdots\psi_{i-1}(n+1)}{\psi_1(n)\cdots\psi_{i-1}(n)}\,\varphi_i(n) \\
&= \varphi^e_i(n+1) - \beta_i(n+1)\varphi_i(n) = \varphi^e_i(n+1) - \varphi^*_i(n) \ . \qquad (3.2.1)
\end{aligned}
$$

This implies

$$\varphi_i^e(n+1) = \varphi_{i+1}^e(n+1) + \varphi_i^*(n) \quad \text{for} \quad i = 1,\cdots,k \ . \tag{3.2.2}$$

But $P_{k,n}(x)$ actually is a polynomial of degree $< k$. The k-th order divided difference $\varphi_{k+1}^e(n+1) = \psi_1(n+1)\cdots\psi_k(n+)g^e[x_{n+1},\cdots,x_{n-k+1}]$ is identically zero. Thus $\varphi_i^e(n+1)$ for $i = k,\cdots,1$ can be generated according to (3.2.2). Exactly the same argument as in (3.2.1) shows the following two identities:

$$\begin{aligned} \varphi_{i+1}(n+1) &= \varphi_i(n+1) - \varphi_i^*(n) \ , \\ \varphi_{i+1}^P(n+1) &= \varphi_i^P(n+1) - \varphi_i^*(n) \quad \text{for} \quad i = 1,\cdots,k \ . \end{aligned} \tag{3.2.3}$$

But then it follows, by taking difference between (3.2.1) and (3.2.3), that

$$\begin{aligned} &\varphi_{i+1}(n+1) - \varphi_{i+1}^e(n+1) = \varphi_i(n+1) - \varphi_i^e(n+1) = \cdots = \varphi_1(n+1) - \varphi_1^e(n+1) \\ &\varphi_{i+1}^P(n+1) - \varphi_{i+1}^e(n+1) = \varphi_i^P(n+1) - \varphi_i^e(n+1) = \cdots = \varphi_1^P(n+1) - \varphi_1^e(n+1) \ . \end{aligned} \tag{3.2.4}$$

So the quantities $\varphi_i^P(n+1)$ and $\varphi_i(n+1)$ for $i = k,\cdots,1$ can now be generated from the following two important identities:

$$\varphi_i^P(n+1) = \varphi_i^e(n+1) + (g_{n+1}^P - \varphi_1^e(n+1)) \tag{3.2.5}$$

and

$$\varphi_i(n+1) = \varphi_i^e(n+1) + (g_{n+1} - \varphi_1^e(n+1)) \ . \qquad (3.2.6)$$

3.3. MODIFIED PECE MODE

We now summarize the (k,k+1)-order PECE mode (2.1.5) which advances from x_n to x_{n+1} as follows:

Compute $\omega_{i,l}$ $\quad i = 1,\cdots,k+1;$

$$\begin{aligned}
&\text{P:}\quad \varphi_i^*(n) = \beta_i(n+1)\varphi_i(n) \qquad i = 1,\cdots,k \ , \\
&\qquad P_{n+1} = e^{Ah_{n+1}} y_n + h_{n+1} \sum_{i=1}^{k} \omega_{i,1}\, \varphi_i^*(n) \ , \\
&\qquad \varphi_{k+1}^e(n+1) = 0 \\
&\qquad \varphi_i^e(n+1) = \varphi_{i+1}^e(n+1) + \varphi_i^*(n) \qquad i = k,\cdots,1 \ ; \\
&\text{E:}\quad g_{n+1}^P = g(x_{n+1}, P_{n+1}) \ ; \qquad (3.3.1) \\
&\text{C:}\quad y_{n+1} = P_{n+1} + h_{n+1}\omega_{k+1,1}(g_{n+1}^P - \varphi_1^e(n+1)) \ ; \\
&\text{E:}\quad g_{n+1} = g(x_{n+1}, y_{n+1}), \\
&\qquad \varphi_{k+1}(n+1) = g_{n+1} - \varphi_1^e(n+1) \ , \\
&\qquad \varphi_i(n+1) = \varphi_i^e(n+1) + \varphi_{k+1}(n+1) \ , \qquad i = k,\cdots,1 \ .
\end{aligned}$$

3.4. ADVANTAGES

In this section we describe briefly the advantages of the method given in the proceding three sections:

(i) Throughout the calling of the whole algorithm we only need to compute A^{-1} (or the generalized inverse) once and for all because $(Ah_{new})^{-1} = (Ah_{old})^{-1} \cdot \frac{h_{old}}{h_{new}}$ can be updated by scalar multiplications.

(ii) If the step succeeds and the step size is doubled, then $e^{Ah_{new}} = (e^{Ah_{old}})^2$ 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.

(iii) When an implicit linear multistep scheme, usually given in the form

$$\sum_{i=0}^{k} \alpha_i y_{n-i+1} = h \sum_{i=0}^{k} \beta_i f_{n-i+1} \qquad (3.4.1)$$

with fixed constant step size h and constant coefficients α_i's and β_i's, is applied to solve Problem (1.2), at each step one has to solve the nonlinear equation

$$y_{n+1} = h\beta_0 f(x_{n+1}, y_{n+1}) + \omega \qquad (3.4.2)$$

where $f(x,y) = Ay + g(x,y)$ and $\omega = h \sum_{i=1}^{k} \beta_i f_{n-i+1} - \sum_{i=1}^{k} \alpha_i y_{n-i+1}$. Notice that ω is a known quantity at the current stage. In order to apply the contraction mapping theorem, it is required to have

$$k = h|\beta_0|L_f < 1 \qquad (3.4.3)$$

where L_f is the Lipschitz constant for f. Meanwhile, when an implicit nonlinear multistep scheme (2.1.2) with constant step size $\hat{h}$ is applied, the nonlinear equation to be solved becomes

$$y_{n+1} = \hat{h}\, \Phi_{k0}\, g(x_{n+1}, y_{n+1}) + \hat{\omega} \qquad (3.4.4)$$

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

$$\hat{k} = \hat{h}\, \|\Phi_{k0}\|\, L_g < 1 \ . \qquad (3.4.5)$$

Under the assumption that the iterative method is used to solve (3.4.2) and (3.4.4) with the same rate of convergence, i.e., $k = \hat{k}$, it is seen that

$$\frac{\hat{h}}{h} = \frac{|\beta_0|}{\|\Phi_{k0}\|}\ \frac{L_f}{L_g} \ . \qquad (3.4.5)$$

This shows that for problem (1.2) where L_f is much greater than L_g, the step size $\hat{h}$ for the nonlinear scheme can be chosen significantly larger than that of the linear scheme.

(iv) Since the equation (3.1.17) and all the quantities (3.1.1) are exactly in the same format as those in [28] except that ω_{iq}'s are in matrix form, φ_i's are defined in terms of g and that ω_{1q}'s and ω_{2q}'s are generated according to (3.1.21), all the efficient software designs by Shampine and Gordon, such as the substantial reduction in the overhead and the memory storages, can be perfectly transferred.

4. ERROR ANALYSIS

In this chapter we first discuss the estimates of errors which occur during the numerical integration. Then we explain how those estimates can be used to control the step size and the order.

4.1. ERROR ESTIMATION

Let $u_n(x)$ denote the exact solution to the problem

$$u_n'(x) = Au_n(x) + g(x,u_n(x)); \quad u_n(x_n) = y_n \ . \qquad (4.1.1)$$

Then the global error (2.2.2.4) can be split into

$$\begin{aligned} \tilde{e}_{n+1} &= y(x_{n+1}) - y_{n+1} \\ &= [y(x_{n+1}) - u_n(x_{n+1})] + [u_n(x_{n+1}) - y_{n+1}] \ . \end{aligned} \qquad (4.1.2)$$

We use the term "local error" to denote the second term of (4.1.2). Since $y(x)$ and $u_n(x)$ are solutions to the same differential equation with different initial values at x_n, we see that

$$\begin{aligned} \|y(x) - u_n(x)\| \le \|e^{A(x-x_n)}(y(x_n) - u_n(x_n))\| \\ + \int_{x_n}^{x} \|e^{A(x-\tau)}[g(\tau,y(\tau)) - g(\tau,u_n(\tau))]d\tau \ . \end{aligned} \qquad (4.1.3)$$

Use the condition on A and the Gronwall inequality we have

$$\|y(x) - u_n(x)\| \leq \|y(x_n) - u_n(x_n)\| e^{L_g(x-x_n)}$$

$$\text{for every } x \geq x_n . \tag{4.14}$$

Given $\epsilon > 0$, let

$$\|u_n(x_{n+1}) - y_{n+1}\| < h_{n+1}\,\epsilon \tag{4.1.5}$$

Then (4.1.4) and (4.1.5) imply that

$$\|\tilde{e}_{n+1}\| \leq \|\tilde{e}_n\| e^{L_g h_{n+1}} + h_{n+1}\,\epsilon . \tag{4.1.6}$$

Repeating the inequality for $i = 0,\cdots,n$, since $\tilde{e}_0 = 0$, we obtain

$$\|\tilde{e}_{n+1}\| \leq \epsilon(x_{n+1} - x_0) e^{L_g(x_{n+1}-x_0)} \tag{4.1.7}$$

which is the same kind of bound as (2.2.2.1) obtained by controlling the general "local truncation error". Therefore it only needs to control the local error in (4.1.5).

4.1.1. LOCAL ERROR FOR (k,k)-ORDER MODE

We first analyze the local error for the predicted value. Equation (3.1.2) implies that

$$u_n(x_{n+1}) - P_{n+1} = \int_{x_n}^{x_{n+1}} e^{A(x_n+1-\tau)} [g(\tau, u_n(\tau)) - P_{k,n}(\tau)]d\tau . \tag{4.1.1.1}$$

Let $\varphi_{k,n}(\tau)$ be a polynomial of degree $< k$ such that $\varphi_{k,n}(x_{n-i+1}) = g(x_{n-i+1},u_n(x_{n-i+1}))$ for $i = 1,\cdots,k$, then (4.1.1.1) can be split into

$$u_n(x_{n+1}) - P_{n+1} = \int_{x_n}^{x_{n+1}} e^{A(x_{n+1}-\tau)}[g(\tau,u_n(\tau)) - \varphi_{k,n}(\tau)]d\tau + \int_{x_n}^{x_{n+1}} e^{A(x_{n+1}-\tau)}[\varphi_{k,n}(\tau) - P_{k,n}(\tau)]d\tau \ . \tag{4.1.1.2}$$

The first integral is the error due to approximating $g(\tau,u_n(\tau))$ by $\varphi_{k,n}(\tau)$. As was proved in Section 2.2.1, this integral is $O(H^{k+1})$ where H is the maximum step size considered. The second integral is the error due to inaccurate values y_{n-i+1}. Notice that

$$\left\| \int_{x_n}^{x_{n+1}} e^{A(x_{n+1}-\tau)}[\varphi_{k,n}(\tau) - P_{k,n}(\tau)]d\tau \right\| = \left\| h_{n+1} \sum_{i=1}^{k} \Phi_{ki}[g(x_{n-i+1},u_n(x_{n-i+1})) - g(x_{n-i+1},y_{n-i+1})] \right\| \tag{4.1.1.3}$$

where $\Phi_{ki} = \int_0^1 e^{A(1-\alpha)h_{n+1}} \ell_i(x_n+\alpha h_{n+1})d\alpha$ and $\ell_i(x) = \prod_{\substack{j=1 \\ j\neq i}}^{k} \frac{x-x_{n-j+1}}{x_{n-i+1}-x_{n-j+1}}$ for $i = 1,\cdots,k$. Let the ratios of successive step sizes be bounded, with the condition on A, it can be shown [28] that $\alpha = \sum_{i=1}^{k} \|\Phi_{ki}\|$ is uniformly bounded. In this case (4.1.1.3) is bounded by $H\alpha L_g \sum_{i=1}^{k} \|u_n(x_{n-i+1}) - y_{n-i+1}\|$. To get an estimate on this summation, consider the variational equation to (4.1.1), i.e.,

$$V'(x) = [A + \frac{\partial g(x,y(x))}{\partial y}]V(x); \quad V(x_n) = I \ . \qquad (4.1.1.4)$$

It is known that (4.1.1.4) has the solution

$$V(x) = \frac{\partial y(x,z)}{\partial z}\Big|_{z=y(x_n)} \qquad (4.1.1.5)$$

where $y(x,z)$ is the solution to (4.1.1) with initial value $y(x_n) = z$. Thus

$$u_n(x,y_n) = y(x,y(x_n)) + V(x)(y_n - y(x_n)) + O(\|y_n - y(x_n)\|^2) \ . \qquad (4.1.1.6)$$

Recall that $y_n - y(x_n) = O(H^{k+1})$ by Theorem 2.2 in section 2.2.2. If we assume $y(x_{n-i+1}) - y_{n-i+1} = y(x_n) - y_n + O(H^{k+2})$, then, since $V(x) = I + O(H)$, it follows that

$$u_n(x_{n-i+1}) - y_{n-i+1} = O(H^{k+2}) \qquad (4.1.1.7)$$

and

$$u_n(x_{n+1}) - P_{n+1} = O(H^{k+1}) \ . \qquad (4.1.1.8)$$

We now analyze the local error for the corrected value. Let $y_{n+1}(k)$ denote the result of correcting the predicted value $P_{n+1}(k) = P_{n+1}$ of order k with a correction of the same order. Let $\varphi_{k,n+1}(\tau)$ be the polynomial of degree $< k$ such that $\varphi_{k,n+1}(x_{n-i+1}) = g(x_{n-i+1}, u_n(x_{n-i+1}))$ for $i = 0,\cdots,k-1$. Then

$$u_n(x_{n+1}) - y_{n+1}(k)$$

$$= \int_{x_n}^{x_{n+1}} e^{A(x_{n+1}-\tau)}[g(\tau,u_n(\tau)) - P^*_{k,n}(\tau)]d\tau$$

$$= \int_{x_n}^{x_{n+1}} e^{A(x_{n+1}-\tau)}[g(\tau,u_n(\tau)) - \varphi_{k,n+1}(\tau)]d\tau$$

$$+ \int_{x_n}^{x_{n+1}} e^{A(x_{n+1}-\tau)}[\varphi_{k,n+1}(\tau) - P^*_{k,n}(\tau)]d\tau \ . \qquad (4.1.1.9)$$

As in (4.1.1.2), the first integral is $O(H^{k+1})$. The second integral is bounded by

$$\|h_{n+1} \sum_{i=1}^{k-1} \Phi^*_{ki}[g(x_{n-i+1},u_n(x_{n-i+1})) - g(x_{n-i+1},y_{n-i+1})]\|$$

$$+ \|h_{n+1} \Phi^*_{k0}[g(x_{n+1},u_n(x_{n+1})) - g(x_{n+1},P_{n+1})]$$

$$\leq H\alpha^* L_g \{ \sum_{i=1}^{k-1} \|u_n(x_{n-i+1}) - y_{n-i+1}\| + \|u_n(x_{n+1}) - P_{n+1}\| \}. \qquad (4.1.1.10)$$

By (4.1.1.7) and (4.1.1.8), we see that the second term on (4.1.1.9) is $O(H^{k+2})$. And hence,

$$u_n(x_{n+1}) - y_{n+1}(k) = O(H^{k+1}) \ . \qquad (4.1.1.11)$$

By the same argument, we may show that

$$u_n(x_{n+1}) - y_{n+1} = O(H^{k+2}) \ . \qquad (4.1.1.12)$$

Define

$$le_{n+1}(k) = u_n(x_{n+1}) - y_{n+1}(k) \ , \qquad (4.1.1.13)$$

then

$$\begin{aligned} le_{n+1}(k) &= u_n(x_{n+1}) - y_{n+1}(k) \\ &= y_{n+1} - y_{n+1}(k) + u_n(x_{n+1}) - y_{n+1} \\ &= y_{n+1} - y_{n+1}(k) + O(H^{k+2}) \\ &\approx y_{n+1} - y_{n+1}(k) \ . \end{aligned} \qquad (4.1.1.14)$$

The importance of the estimate (4.1.1.14) is that the value $y_{n+1}(k)$ can be readily obtained from P_{n+1}. In fact, from the definitions of $P^*_{k,n}(\tau)$ and $P^*_{k+1,n}(\tau)$, we see that

$$\begin{aligned} & P^*_{k+1,n}(x) \\ &= P_{k,n}(x) + (x-x_n)\cdots(x-x_{n-k+1})g^p[x_{n+1},\cdots,x_{n-k+1}] \\ &= P^*_{k,n}(x) + (x-x_{n+1})\cdots(x-x_{n-k+2})g^p[x_{n+1},\cdots,x_{n-k+1}] \ . \end{aligned} \qquad (4.1.1.15)$$

It follows that

$$\begin{aligned} & P^*_{k,n}(x) \\ &= P_{k,n}(x) + (x-x_n)\cdots(x-x_{n-k+2})(x_{n+1}-x_{n-k+1}) \\ &\qquad\qquad g^p[x_{n+1},\cdots,x_{n-k+1}] \\ &= P_{k,n}(x) + (sh_{n+1})\cdots(sh_{n+1}+\psi_{k-2}(n))\psi_k(n+1)\frac{\varphi^p_{k+1}(n+1)}{\psi_1(n+1)\cdots\psi_k(n+1)} \\ &= P_{k,n}(x) + c_{k,n}(s)\varphi^p_{k+1}(n+1) \ . \end{aligned} \qquad (4.1.1.16)$$

This implies

$$y_{n+1}(k) = P_{n+1} + h_{n+1}\, \omega_{k,1}\, \varphi^{P}_{k+1}(n+1) \ . \qquad (4.1.1.17)$$

Together with (3.1.25), we obtain the (k,k)-order error estimate

$$le_{n+1}(k) \approx h_{n+1}(\omega_{k+1,1}-\omega_{k,1})\varphi^{P}_{k+1}(n+1) \ . \qquad (4.1.1.18)$$

4.1.2. LOCAL ERROR FOR LOWER ORDER MODE

By (4.1.1.11), the local error resulting from a PECE mode with both predictor $P_{n+1}(k-1)$ and corrector $y_{n+1}(k-1)$ of order $k-1$ is

$$le_{n+1}(k-1) = u_n(x_{n+1}) - y_{n+1}(k-1) = O(H^k) \ . \qquad (4.1.2.1)$$

Thus, we may write

$$\begin{aligned} le_{n+1}(k-1) &= u_n(x_{n+1}) - y_{n+1}(k-1) \\ &= y_{n+1}(k) - y_{n+1}(k-1) + u_n(x_{n+1}) - y_{n+1}(k) \\ &= y_{n+1}(k) - y_{n+1}(k-1) + O(H^{k+1}) \\ &\approx y_{n+1}(k) - y_{n+1}(k-1) \ . \end{aligned} \qquad (4.1.2.2)$$

The value $y_{n+1}(k-1)$ cannot be obtained as cheaply as $y_{n+1}(k)$ because

$$y_{n+1}(k-1) = e^{Ah_{n+1}} y_n + h_{n+1} \sum_{i=1}^{k-2} \Phi^*_{k-1,i} g_{n-i+1}$$

$$+ h_{n+1}\Phi^*_{k-1,0} g(x_{n+1}, P_{n+1}(k-1)) \qquad (4.1.2.3)$$

where $g(x_{n+1}, P_{n+1}(k-1))$ is not readily known. Define

$$\bar{y}_{n+1}(k-1) = e^{Ah_{n+1}} y_n + h_{n+1} \sum_{i=1}^{k-2} \Phi^*_{k-1,i}\, g_{n-i+1}$$

$$+ h_{n+1}\, \Phi^*_{k-1,0}\, g(x_{n+1}, P_{n+1}) \ . \qquad (4.1.2.4)$$

From (3.1.23), we have

$$P_{n+1}(k-1) = e^{Ah_{n+1}} y_n + h_{n+1} \sum_{i=1}^{k-1} \omega_{i,1}\, \varphi^*_i(n) \ , \qquad (4.1.2.5)$$

and, similar to (3.1.25),

$$\bar{y}_{n+1}(k-1) = P_{n+1}(k-1) + h_{n+1}\, \omega_{k-1,1}\, \varphi^P_k(n+1) \ . \qquad (4.1.2.6)$$

It follows from (3.2.3) that

$$\bar{y}_{n+1}(k-1) = e^{Ah_{n+1}} y_n + h_{n+1} \sum_{i=1}^{k-1} (\omega_{i,1} - \omega_{i-1,1})\varphi^P_i(n+1). \qquad (4.1.2.7)$$

On the other hand,

$$\|\bar{y}_{n+1}(k-1) - y_{n+1}(k-1)\|$$

$$= \|h\Phi^*_{k-1,0}[g(x_{n+1}, P_{n+1}) - g(x_{n+1}, P_{n+1}(k-1))]$$

$$\le H\alpha^* L_g \| P_{n+1} - P_{n+1}(k-1)\| \qquad (4.1.2.8)$$

$$\le H\alpha^* L_g \{\| P_{n+1} - u_n(x_{n+1})\| + \| u_n(x_{n+1}) - P_{n+1}(k-1)\| \}$$

Estimate (4.1.1.8) guarantees that (4.1.2.8) is $O(H^{k+1})$. Thus (4.1.2.2) can be modified as

$$\begin{aligned} le_{n+1}(k-1) &= y_{n+1}(k) - y_{n+1}(k-1) + O(H^{k+1}) \\ &= y_{n+1}(k) - \bar{y}_{n+1}(k-1) + \bar{y}_{n+1}(k-1) \\ &\qquad - y_{n+1}(k-1) + O(H^{k+1}) \\ &= y_{n+1}(k) - \bar{y}_{n+1}(k-1) + O(H^{k+1}) \\ &\approx y_{n+1}(k) - \bar{y}_{n+1}(k-1) \ . \end{aligned} \tag{4.1.2.9}$$

By (4.1.1.17) and (4.1.2.6), we obtain the (k-1,k-1)-order error estimate

$$le_{n+1}(k-1) \approx h_{n+1}(\omega_{k,1} - \omega_{k-1,1})\varphi_k^P(n+1) \ . \tag{4.1.2.10}$$

By the same argument, we may obtain the (k-2,k-2)-order error estimate

$$le_{n+1}(k-2) \approx h_{n+1}(\omega_{k-1,1} - \omega_{k-2,1})\varphi_{k-1}^P(n+1) \ . \tag{4.1.2.10}$$

4.1.3. LOCAL ERROR FOR (k+1,k+1)-ORDER MODE

The procedure described in the last section fails for the estimate

$$le_{n+1}(k+1) = u_n(x_{n+1}) - y_{n+1}(k+1) \tag{4.1.3.1}$$

because there is no dominant term in this case. That is, $u_n(x_{n+1}) - y_{n+1} = O(H^{k+2})$ is of the same order as $u_n(x_{n+1}) - y_{n+1}(k+1) = O(H^{k+2})$. If we were to take a step with a (k+1,k+2)-order mode, then (4.1.1.18) implies that

$$le_{n+1}(k+1) = h_{n+1}(\omega_{k+2,1}-\omega_{k+1,1})\varphi^P_{k+2}(n+1) + O(H^{k+3}). \quad (4.1.3.2)$$

For convenience, we assume that a constant step size has been used. Then

$$\begin{aligned}
\varphi^P_{k+2}(n+1) &= \nabla^{k+1} g(x_{n+1}, P_{n+1}(k+1)) \\
&= g(x_{n+1}, P_{n+1}(k+1)) + \sum_{m=1}^{k+1} (-1)^m g_{n-m+1} \binom{k+1}{m} \\
&= g(x_{n+1}, P_{n+1}(k+1)) - g(x_{n+1}, y_{n+1}) + \nabla^{k+1} g_{n+1} \\
&= \nabla^{k+1} g_{n+1} + O(H^{k+2}) \\
&= \varphi_{k+1}(n+1) - \varphi_{k+1}(n) + O(H^{k+2}) \qquad (4.1.3.3)
\end{aligned}$$

Here we have used the fact that $P_n(k+1) - y_{n+1} = O(H^{k+2})$ from (4.1.1.8). We thus obtain the (k+1,k+1)-order error estimate

$$le_{n+1}(k+1) = h_{n+1}(\omega_{k+2-1}-\omega_{k+1,1})(\varphi_{k+1}(n+1)-\varphi_{k+1}(n)) . \quad (4.1.3.4)$$

Notice that it is necessary to know $\omega_{k+2,1}$, $\varphi_{k+1}(n+1)$ and $\varphi_{k+1}(n)$ in order to apply this estimate.

4.2. AUTOMATIC CONTROL

Any program embodying a multistep method will have to invoke techniques in starting, changing step size and changing order as necessary not only for the purpose of reducing the cost but also for the consideration of stability. For an A-stable method, the latter factor probably is less concerned then the former when dealing with stiff systems. In this section we follow closely the algorithm adopted by Shampine and Gordon to develop a step size and order selection mechanism.

4.2.1. ACCEPTION CRITERION

Let $\epsilon > 0$ be given. When the step from x_n to x_{n+1} has been calculated to the stage PE in the mode (3.3.1), we test the criterion

$$ERR = \| le_{n+1}(k) \| = \| h_{n+1}(\omega_{k+1,1}-\omega_{k,1})\varphi^{P}_{k+1}(n+1) \| \leq \epsilon \qquad (4.2.1.1)$$

according to (3.2.5) and (4.1.1.18). If this criterion is satisfied, we accept this step by completing the remaining CE part in that mode, overwriting the updated $\varphi_i(n+1)$ on the $\varphi^{e}_i(n+1)$ and determining the step size and order for the next step. If this test fails, we cut the current step size in half and restart to predict the value P_{n+1}. If there are three consecutive failures, then we reduce the order to be 1.

4.2.2. ERROR PREDICTION FOR FIXED STEP SCHEME

Recall that for $i \geq 2$,

$$\omega_{i,1} = d_{i,n}^{(1)}(1)$$

$$= \int_0^1 e^{A(1-s)h_{n+1}} \left(\frac{sh_{n+1}}{\psi_1(n+1)}\right)\left(\frac{sh_{n+1}+h_n}{\psi_2(n+1)}\right) \cdots \left(\frac{sh_{n+1}+\psi_{i-2}(n)}{\psi_{i-1}(n+1)}\right)ds \quad . \qquad (4.2.2.1)$$

If constant step size h is used, then

$$\omega_{i,1} = \frac{1}{(i-1)!} \int_0^1 e^{A(1-s)h}\, s(s+1)\cdots(s+i-2)ds \quad . \qquad (4.2.2.2)$$

This suggests that if the mesh sizes are not wildly different, neither are these matrices $\omega_{i,1}$'s. In fact, one can show that

$$\omega_{k+1,1} - \omega_{k,1}$$

$$= \int_0^1 e^{A(1-s)h_{n+1}} \left(\frac{sh_{n+1}}{\psi_1(n+1)}\right)\left(\frac{sh_{n+1}+h_n}{\psi_2(n+1)}\right) \cdots \left(\frac{sh_{n+1}+\psi_{k-2}(n)}{\psi_{k-1}(n+1)}\right)\left(\frac{(s-1)h_{n+1}}{\psi_k(n+1)}\right)ds \quad . \qquad (4.2.2.3)$$

Thus if we assume that $h_{n+1} \geq \frac{\psi_i(n)}{i}$ for every $i = 2,---,k$, then

$$\|\omega_{k+1,1} - \omega_{k,1}\| \leq \gamma_k^* \qquad (4.2.2.4)$$

where γ_k^* is the coefficient for the classical Adams-Moulton method, and is known to be relatively small. Therefore, (4.1.1.18) suggests that the error at x_{n+2} is predicted to be

$$le_{n+2}(k) = h_{n+2}(\omega_{k+1,1}-\omega_{k,1})\varphi^P_{k+2}(n+2) \qquad (4.2.2.5)$$

where we have replaced $d^{(1)}_{k+1,n+1}(1) - d^{(1)}_{k,n+1}(1)$ by $d^{(1)}_{k+1,n}(1) - d^{(1)}_{k,n}(1)$. Suppose $h = h_{n+1}$ and we wish to consider $h_{n+2} = \gamma h$ with γ near 1. For simplicity, we assume that the preceding steps were all taken with the same step size γh. Suppose also that the order k has been chosen properly so that the k-th order divided difference is nearly constant. Then

$$\varphi^P_{k+1}(n+2) = (\gamma h)(2\gamma h)\cdots(k\gamma h)f^P[x_{n+2},\cdots,x_{n-k+2}]$$

$$\approx \gamma^k \sigma_{k+1}(n+1)\varphi^P_{k+1}(n+1) \qquad (4.2.2.6)$$

where $\sigma_1(n+1) = 1$ and $\sigma_i(n+1) = \dfrac{h\cdot 2h\cdot\cdots\cdot(i-1)h}{\psi_1(n+1)\cdots\psi_{i-1}(n+1)}$ for $i = 2,\cdots,k+1$. In exactly the same context, we may use the quantity

$$ERK = \|h_{n+1}(\omega_{k+1,1}-\omega_{k,1})\sigma_{k+1}(n+1)\varphi^P_{k+1}(n+1)\| \qquad (4.2.2.7)$$

as the error estimate at x_{n+1} when the preceding steps were all taken with step size h_{n+1}. Notice that the predicted error at x_{n+2} using step size $h_{n+2} = \gamma h_{n+1}$ is

now given by γ^{k+1} ERK according to (4.2.2.5). Moreover, the same quantity γ^{k+1}ERK also estimates the error at x_{n+1} had a step size γh_{n+1} been used. We also use quantities

$$\text{ERKM1} = \|h_{n+1}(\omega_{k,1}-\omega_{k-1,1})\sigma_k(n+1)\varphi_k^P(n+1)\| \tag{4.2.2.8}$$

$$\text{ERKM2} = \|h_{n+1}(\omega_{k-1,1}-\omega_{k-2,1})\sigma_{k-1}(n+1)\varphi_{k-1}^P(n+1)\|$$

to estimate the local error at x_{n+1} and x_{n+2} had the preceding steps been taken with constant step size h_{n+1} at orders k-1 and k-2, respectively. Finally, when the preceding steps are actually taken with constant step size h_{n+1}, we use

$$\text{ERKP1} = \|le_{n+1}(k+1)\|$$

$$= \|h_{n+2}(\omega_{k+2,1}-\omega_{k+1,1})(\varphi_{k+1}(n+1)-\varphi_k(n))\| \tag{4.2.2.9}$$

as the predicted error at x_{n+2} had the order k+1 been used.

4.2.3. ORDER SELECTION

The purpose of order adjustment is to find a polynomial with degree as low as possible which represents the nonlinear function g(x,y(x)) locally with a reasonable accuracy. We adopt the philosophy that the order is changed only if the predicted error is reduced and there is a trend in doing so. Therefore the order is lowered only if one of the following happens:

(i) $k = 2$ and $ERKM1 \leq 0.5ERK$,

(ii) $k > 2$ and $\max(ERKM1, ERKM2) \leq ERK$,

(iii) $ERKM1 \leq \min(ERK, ERKP1)$,

(iv) There are repeated step failures.

And the order is raised only if one of the following happens:

(i) $ERKP1 < ERK$,

(ii) $k = 1$ and $ERKP1 < 0.5ERK$,

(iii) The starting phase flag is still on.

4.2.4. STEP SIZE SELECTION

After the order to be used is selected, we rename it k, and the corresponding predicted error at constant step size, ERK. Then the predicted error at x_{n+2} with step size γh_{n+1} is given by $\gamma^{k+1} ERK$. Since it is desired to have $\gamma^{k+1} ERK \leq \varepsilon$, we select the step size according to the ratio

$$\gamma = \left(\frac{0.5\varepsilon}{ERK}\right)^{\frac{1}{k+1}} , \qquad (4.2.4.1)$$

and

(i) Double the step size if $\gamma \geq 2$,

(ii) Retain the step size if $1 \leq \gamma < 2$,

(iii) Reduce the step size by a factor $\max\{0.5, \min\{0.9, \gamma\}\}$ if $\gamma < 1$,

(iv) Halve the step size if there is a failure,

(v) Double the step size if the starting phase flag is on.

4.2.5. STARTING PHASE

The local error at x_1 for order 0 is given by

$$le_1(0) \approx h\omega_{1,1}g(x_0,y_0) \ . \qquad (4.2.5.1)$$

As suggested from (4.1.1.11) and (4.1.1.12), we assume $\ell e_1(1) = h\ell e_1(0)$. Then we would choose h such that

$$\|le_1(1)\| \approx \|h^2\omega_{1,1}g(x_0,y_0)\| < \frac{\varepsilon}{2} \ . \qquad (4.2.5.2)$$

But $\|\omega_{1,1}\| \leq 1$, so we choose the starting step size

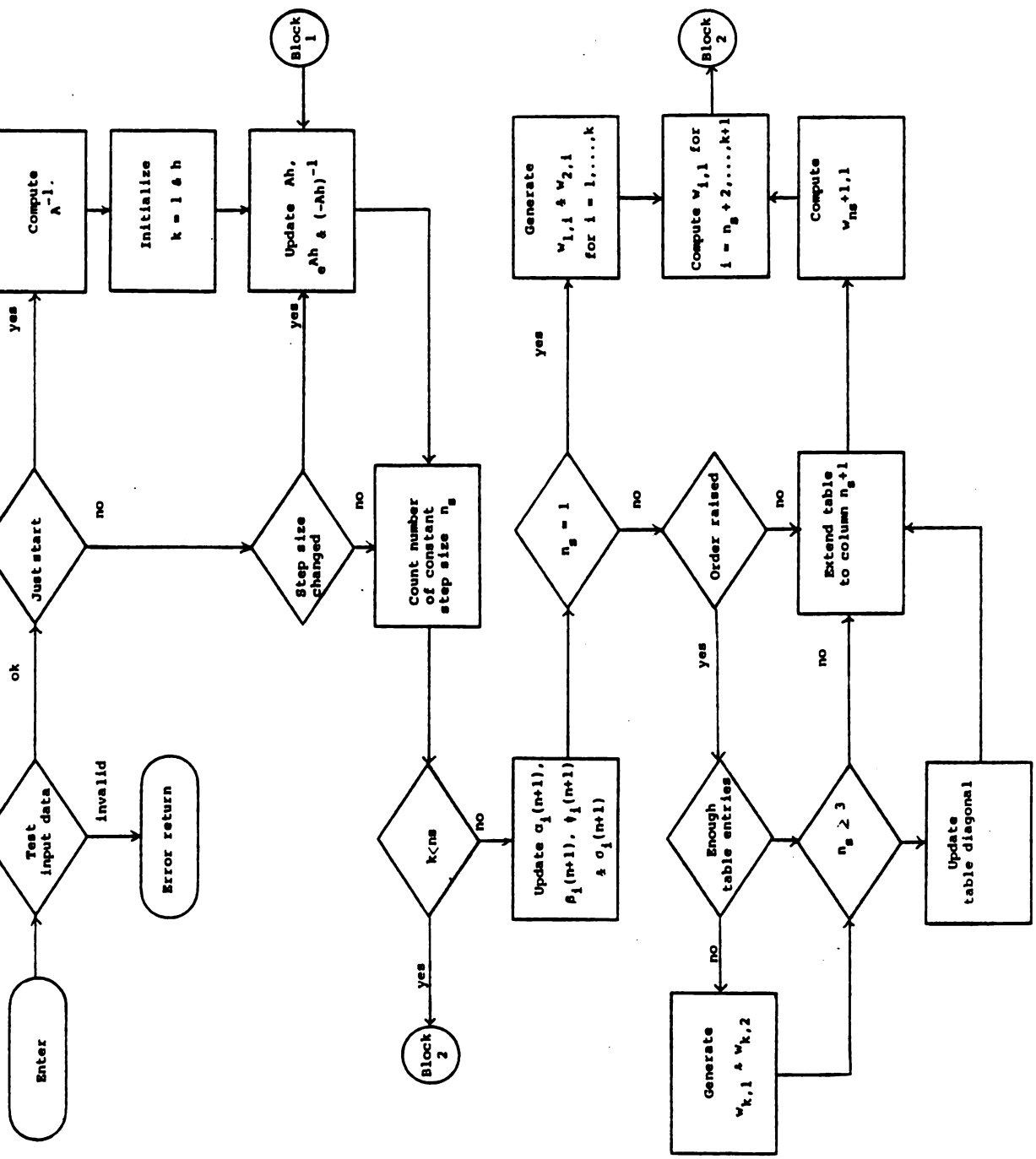
$$h = \min(h \text{ input}, \frac{1}{4}\|\frac{.5\varepsilon}{g(x_0,y_0)}\|^{1/2}) \ . \qquad (4.2.5.3)$$

The starting phase flag is turned off whenever the appropriate order and step size are found, which is indicated by the lowering of order or a step failure.

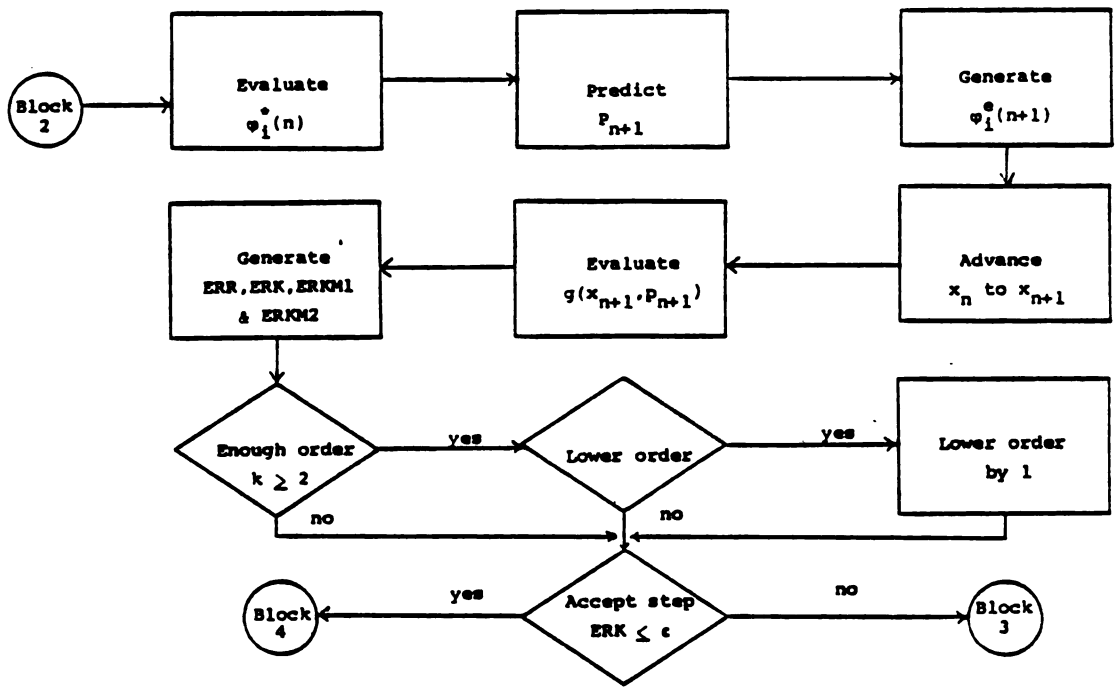
5. PROGRAMMING FLOWCHARTS

In this chapter we assemble together all pieces of algorithms mentioned in previous sections to give a general overview of this nonlinear method. The organization is composed of four blocks which are presented in terms of flowcharts.

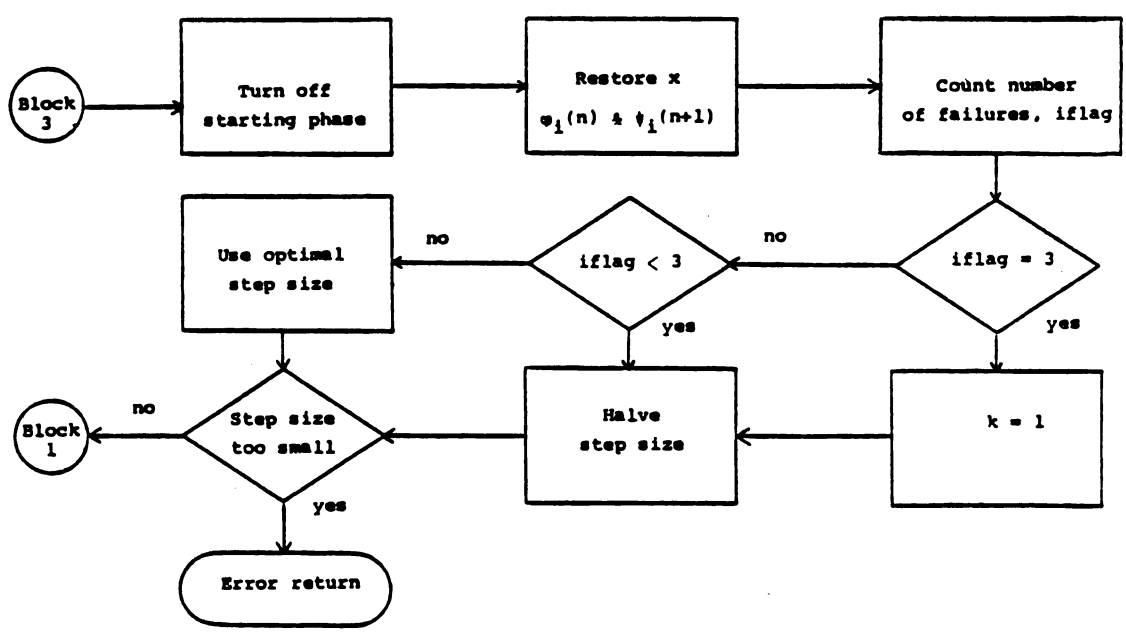
5.1. BLOCK 1 -- PREPARATION OF COEFFICIENTS



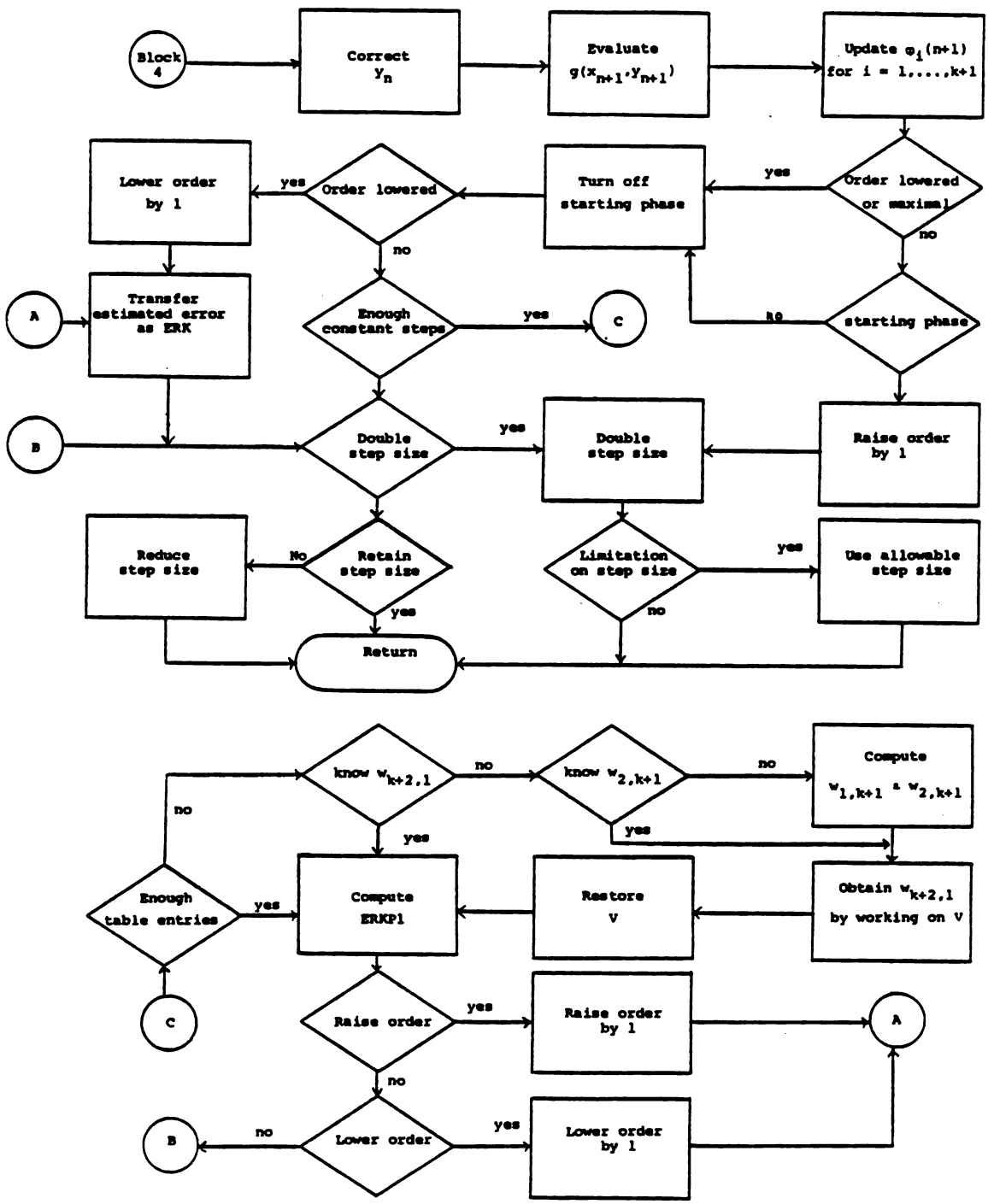
5.2. BLOCK 2 -- PREDICTION AND ERROR ESTIMATION



5.3. BLOCK 3 -- RESTART OF STEP



5.4 BLOCK 4 -- CORRECTION AND AUTOMATIC ADJUSTMENT



6. NUMERICAL EXAMPLES

Test results from several problems representing both linear and nonlinear systems of equations are presented in this chapter. We also give the comparison of our algorithm NLMSUB with Gear's DIFSUB or Lawson's IRKSUB. It should be emphasized that the principal objective of these tests is to confirm the effectiveness of this non-linear method, rather than to make the detailed comparison of available method, although several results do show some significant advantages.

6.1. HOMOGENEOUS LINEAR SYSTEM WITH COMPLEX EIGENVALUES [9]

Problem:

$$y' = \begin{bmatrix} -1, & 1, & 0, & 0 \\ -100, & -1, & 0, & 0 \\ 0, & 0, & -100, & 1 \\ 0, & 0, & -10000, & -100 \end{bmatrix} y; \quad y(0) = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} .$$

Exact Solution:

$$y(x) = \begin{bmatrix} e^{-x}\cos(10x) \\ -10e^{-x}\sin(10x) \\ e^{-100x}\cos(100x) \\ -100e^{-100x}\sin(100x) \end{bmatrix} .$$

Test Parameters:

Local Error Tolerance $\quad eps = 10^{-6}$,

Time Interval $\quad x \in [0,20]$.

Eigenvalues:

$$-1 \pm 10i \ , \quad -100 \pm 100i \ .$$

Test Results:

	DIFSUB^{+}	IRKSUB^{+}	NLMSUB
MAX	5.80	5.62	12.73*
STEPS	1488	404	11
FNS	3839	4594	23
JACOB	133	56	0
INV	133	168	2
PADE	0	0	1
TIME	1.222	1.287	0.060$^{\#}$

Notations:

+ These data were picked up from the reference.

* These maximum errors were measured by
$-\log \sqrt{\sum [y_i(x)-y_i]^2}$.

This execution was done on CYBER 750 in seconds.

Remarks:

This problem is desigend to assess the effect on the performance of a method when the eigenvalue forms a rather large angle with the negative x-axis. Our scheme is

extremely efficient on this type of problem because of the stability of diagonal Padé approximations. Notice that no Jacobian evaluation is needed for this nonlinear method.

6.2. NONHOMOGENEOUS LINEAR SYSTEM WITH REAL EIGENVALUES [20]

Problem:

$$y' = \begin{bmatrix} -4498, & -5996 \\ 2248.5, & 2997 \end{bmatrix} y + \begin{bmatrix} 0.006 - x \\ -0.503 + 3x \end{bmatrix};$$

$$y(0) = \frac{1}{1500} \begin{bmatrix} 25498 \\ -16499 \end{bmatrix}.$$

Exact Solution:

$$y(x) = \begin{bmatrix} -2e^{-x} + 7e^{-1500x} + \frac{17998 - 14991x}{1500} \\ 1.5e^{-x} - 3.5e^{-1500x} - \frac{13499 - 1124.5x}{1500} \end{bmatrix}.$$

Test Parameter:

Local Error Tolerance	$eps = 10^{-7}$,
Time Interval	$x \in [0,25]$

Eigenvalues:

$$-1, \quad -1500 .$$

Test Results:

	DIFSUB+	IRKSUB+	NLMSUB
MAX	6.42	6.54	6.73*
AVE	7.73	7.70	7.09*
MIN	9.82	11.99	8.56*
STEPS	235	104	16
FNS	539	592	33
JACOB	23	23	0
PADE	0	0	1
TIME	5.26	4.21	0.05#

Remarks:

This problem fits exactly into our model and hence we expect accurate numerical outputs. The step control seemed to allow the step sizes to be doubled all the way. In fact, the step size allowable from $x = 25.03$ to the next point was nearly 25. This is an optimistic indication that even larger step sizes can be used as the time goes on.

6.3. NONHOMOGENEOUS LINEAR SYSTEM WITH COMPLEX EIGENVALUES [20]

Problem:

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

$$\text{where} \quad U = \frac{1}{2} \begin{bmatrix} -1, & 1, & 1, & 1 \\ 1, & -1, & 1, & 1 \\ 1, & 1, & -1, & 1 \\ 1, & 1, & 1, & -1 \end{bmatrix}$$

Exact Solution:

$$y(x) = U \begin{bmatrix} \sin x + x^2 \\ \cos X - x^2 \\ e^{-100x} \cos(900x) + x \\ e^{-100x} \sin(900x) - x \end{bmatrix} .$$

Test Parameters:

Local Error Tolerance $\quad eps = 10^{-7}$,

Time Interval $\quad x \in [0,25]$

Eigenvalues:

$$-100 \pm 900i, \quad \pm i$$

Test Results:

	DIFSUB+	IRKSUB+	NLMSUB
MAX	4.55	4.22	6.75*
AVE	6.77	6.44	6.86*
MIN	9.32	8.37	7.31*
STEPS	2982	532	25
FNS	8332	2986	51
JACOB	37	41	0
PADE	0	0	1
TIME	106.03	41.88	.19#

Remarks:

This problem is characterized by the large angles formed by its eigenvalues with the negative real axis. Gear's method has some difficulties in this situation, Lawson's method works better. The matrix U is used to couple all the components and hence makes the problem complicated. The computation in our algorithm actually terminated at $x = 39.21$ instead of 25 whereas the next allowable step size was as big as 39. So conceivably the above data is to be even more prominent if we take time interval from 0 to 80. Notice that the numerical difficulty for a stiff problem usually appears in the steady-state part, i.e. when x is large. So at least for this particular problem, the nonlinear method is more efficient in the long run.

6.4. NONHOMOGENEOUS LINEAR SYSTEM WITH VARIABLE COEFFICIENTS AND REAL NON-CONSTANT EIGENVALUES [20]

Problem:

$$y' = U \begin{bmatrix} -1-0.01\sin(0.01x), & 0, & 0, & 0 \\ 0, & -100-\sin x, & 0, & 0 \\ 0, & 0, & -1000+\cos x, & 0 \\ 0, & 0, & 0, & -10 \end{bmatrix} U^T y$$

$$+ U \begin{bmatrix} 3+0.03\sin(0.01x) \\ 200+2\sin x \\ -1000+\cos x \\ -20 \end{bmatrix}; \quad y(0) = \begin{bmatrix} -1 \\ 0 \\ 2+e \\ 3+e \end{bmatrix} .$$

Exact Solution:

$$y(x) = U \begin{bmatrix} e^{-x+\cos(0.01x)} + 3 \\ e^{-100x+\cos x} + 2 \\ e^{-1000x+\sin x} - 1 \\ e^{-10x-2} \end{bmatrix}$$

Test Parameter:

Local Error Tolerance $\quad eps = 10^{-7}$,

Time Interval $\quad x \in [0,100]$.

Eigenvalues:

$-1-0.01\sin(0.01x)$, $-100-\sin x$, $-1000+\cos x$, -10 .

Program Setting:

$$y' = U \begin{bmatrix} -1, & 0, & 0, & 0 \\ 0, & -100, & 0, & 0 \\ 0, & 0, & -1000, & 0 \\ 0, & 0, & 0, & -10 \end{bmatrix} U^T y$$

$$+ U \left\{ \begin{bmatrix} -0.01\sin(0.01x), & 0, & 0, & 0 \\ 0, & -\sin x, & 0, & 0 \\ 0, & 0, & \cos x, & 0 \\ 0, & 0, & 0, & 0 \end{bmatrix} U^T y + \begin{bmatrix} 3+0.03\sin(0.01x) \\ 200+2\sin x \\ -1000+\cos x \\ -20 \end{bmatrix} \right\}$$

Test Results:

	DIFSUB+	IRKSUB+	NLMSUB
MAX	5.83	4.82	6.04*
AVE	8.02	7.00	6.34*
MIN	10.11	8.36	12.06*
STEPS	323	154	177
FNS	774	867	155
JACOB	31	44	0
PADE	0	0	1
TIME	16.54	26.47	39#

Remarks:

Being a variable coefficient linear system, this problem has real non-constant eigenvalues whose values are in the neighborhood of -1, -10, -100 and -1000. We have

to change its form to fit into our model problem (1.2) in the program setting and thus make $g(x,y(x))$ depend on y. Notice that 75 steps were used for this problem over the interval $[0,36.6]$ in contrast to 25 steps over $[0,39.2]$ for the previous problem. The reason that much greater amount of work was needed is probably due to the poor representation of the severely nonlinear function $g(x,y(x))$ by polynomial $P(x)$. There is some numerical evidence supporting this conjecture: in the previous problem the optimal order used was $k = 3$ (average $k = 2.5$) while in this problem the order used was as high as $k = 6$ (average $k = 3.7$); once the nonlinear part died out (approximately when $x \geq 20$) so that $g(x,y(x))$ became constant, the order dropped abruptly to $k = 1$ and the step size began to grow dramatically.

6.5. NONLINEAR SYSTEM WITH COUPLING COMPONENTS [9]

6.5.1. COUPLING FROM TRANSIENT COMPONENTS TO SMOOTH COMPONENTS

Problem:

$$y' = \begin{bmatrix} -1, & 0, & 0, & 0 \\ 0, & -10, & 0, & 0 \\ 0, & 0, & -40, & 0 \\ 0, & 0, & 0, & -100 \end{bmatrix} y + \begin{bmatrix} y_2^2 + y_3^2 + y_4^2 \\ 10(y_3^2 + y_4^2) \\ 40\, y_4^2 \\ 2 \end{bmatrix}; \quad y(0) = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}.$$

Test Parameters:

Local Error Tolerance $\quad eps = 10^{-6}$

Time Interval $\quad x \in [0,20]$

Eigenvalues:

$$-1, -10, -40, -100$$

Test Results:

	DIFSUB+	IRKSUB+	NLMSUB
MAX	6.00	6.00	?
STEPS	221	40	104
FNS	529	450	209
JACOB	22	31	0
INV	22	93	2
PADE	0	0	1
TIME	0.178	0.314	0.351#

Notation:

? No error report because the close form of exact solution is not known.

Remarks:

This problem has a nonlinear coupling from the transient components to the smooth components and hence causes the high nonlinearity on the function $g(x,y(x))$. The order used was as high as $k = 9$ (average $k = 4.68$). But as in the previous problem, once the transient solutions became

stabilized, the order dropped to $k = 1$ and the step size began to be large.

6.5.2. COUPLING FROM SMOOTH COMPONENTS TO TRANSIENT COMPONENTS

Problem:

$$y' = \begin{bmatrix} -1, & 0, & 0, & 0 \\ 0, & -10, & 0, & 0 \\ 0, & 0, & -40, & 0 \\ 0, & 0, & 0, & -100 \end{bmatrix} y + \begin{bmatrix} 2 \\ \beta y_1^2 \\ 4\beta(y_1^2 + y_2^2) \\ 10\beta(y_1^2 + y_2^2 + y_3^2) \end{bmatrix} ; \quad y(0) = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

where $\beta = 20$.

Test Paramters:

Local Error Tolerance $\quad eps = 10^{-6}$,

Time Interval $\quad x \in [0,20]$.

Eigenvalues:

$$-1, -10, -40, -100 .$$

Test Results:

	DIFSUB$^+$	IRKSUB$^+$	NLMSUB
MAX	5.31	5.40	?
STEPS	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

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 transient components to have very large positive derivative at the very beginning, but eventually this tendency is diminshed and the order is lowered when the smooth components cannot match up the rapid decay of the transient components. A nearly periodic reduction on step sizes results in frequent calls on the Pade approximations. This phenomenon is very unusual and is not known to us what causes this to happen.

6.6. NONLINEAR SYSTEM WITH REAL EIGENVALUES [9]

Problem:

$$y' = \begin{bmatrix} -0.2, & 0.2, & 0 \\ 10, & -60, & 0.125 \\ 0, & 0, & -1 \end{bmatrix} y + \begin{bmatrix} 0 \\ 0.125\ y_1 y_2 \\ y_3 + 1 \end{bmatrix} ; \quad y(0) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Test Parameter:

Local Error Tolerance	$eps = 10^{-6}$,
Time interval	$x \in [0,400]$.

Eigenvalues:

$$0, \quad -0.167 \rightarrow -0.012, \quad -60 \rightarrow -11 \ .$$

Test Results:

	DIFSUB^{+}	IRKSUB^{+}	NLMSUB
MAX	6.10	4.51	?
STEPS	186	508	36??
FNS	630	7918	73
JACOB	46	532	0
INV	46	1596	2
PADE	0	0	1
TIME	0.149	3.200	0.105$^{\#}$

Notation:

?? The execution was actually done to the point $x = 147.5$ only.

Remark:

The execution had been carried out with step sizes as large as 8.19 very smoothly to the point $x = 147.5$ when there was an indication to reduce the step size by a factor $\gamma = 0.9$. Then the execution was terminated because of an overflow in the computation of $e^{Ah_{new}}$. There are two reasons for this fact: the denominator in the Padé approximation for $e^{Ah_{new}}$ is nearly singular and the matrix $e^{Ah_{new}}$ has a "bump" phase [27] near h_{new}. We believe a careful programming should overcome these difficulties.

6.7. NONLINEAR SYSTEM WITH REAL MIXED TYPE EIGENVALUES [11]

Problem:

$$y' = U\begin{bmatrix} -1000, & 0, & 0, & 0 \\ 0, & -800, & 0, & 0 \\ 0, & 0, & 10, & 0 \\ 0, & 0, & 0, & -0.001 \end{bmatrix} U^T y + U\begin{bmatrix} z_1^2 \\ z_2^2 \\ z_3^2 \\ z_4^2 \end{bmatrix} ;$$

$$y(0) = \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \end{bmatrix} \quad \text{where} \quad z = Uy \ .$$

Exact Solution:

$$y(x) = U \begin{bmatrix} \dfrac{1000}{1-1001e^{1000x}} \\ \dfrac{800}{1-801e^{800x}} \\ \dfrac{-10}{1+9e^{-10x}} \\ \dfrac{0.001}{1-0.001e^{0.001x}} \end{bmatrix}$$

Test Parameters:

Local Error Tolerance $\quad eps = 10^{-6}$,

Time Interval $\quad x \in [0,100]$.

Eigenvalues:

$$-1000 + \frac{2000}{1-1001e^{1000x}} \quad , \quad -800 + \frac{1600}{1-801e^{800x}}$$

$$10 - \frac{20}{1+9e^{-10x}} \quad , \quad -0.001 + \frac{0.002}{1-1.001e^{0.001x}}$$

Program Setting:

$$y' = U \begin{bmatrix} -1000, & 0, & 0, & 0 \\ 0, & -800, & 0, & 0 \\ 0, & 0, & -10, & 0 \\ 0, & 0, & 0, & -0.001 \end{bmatrix} U^T y + U \begin{bmatrix} z_1^2 \\ z_2^2 \\ z_3^2 + 20z_3 \\ z_i^2 \end{bmatrix}$$

Test Results:

	+ DIFSUB	NLMSUB	+ DIFSUB	NLMSUB	+ DIFSUB	NLMSUB	+ DIFSUB	NLMSUB	+ DIFSUB	# NLMSUB
Current Time	$.1024\times10^{-1}$	$.1074\times10^{-1}$	$.1049\times10^{0}$	$.1022\times10^{0}$	$.1012\times10^{1}$	$.1043\times10^{1}$	$.1001\times10^{2}$	$.1045\times10^{2}$	$.1025\times10^{3}$	$.1009\times10^{3}$
Steps	70	29	110	43	168	115	216	144	252	211
Present Error	9.100×10^{-8}	3.683×10^{-6}	2.667×10^{-6}	1.470×10^{-5}	2.208×10^{-6}	1.851×10^{-5}	2.870×10^{-6}	1.562×10^{-5}	2.984×10^{-6}	6.565×10^{-7}
Evaluations	179	59	282	89	406	233	523	291	616	425
Inversions	7	2	12	3	15	8	20	9	25	10
Average Step	1.463×10^{-4}	3.703×10^{-4}	9.535×10^{-4}	2.377×10^{-3}	6.025×10^{-3}	9.070×10^{-3}	4.635×10^{-2}	7.26×10^{-2}	4.067×10^{-1}	# 4.782×10^{-1}
Pade Calling		1		2		7		8		9

It takes .98 seconds execution time on CYBER 750.

\# Step size has been restricted not to be greater than 1.5.

Remarks:

In the initial state of integration, the problem is a mixed type nonlinear stiff system because of the presence of a positive eigenvalues. But our algorithm still seems to work satisfactorily. When $x \geq 24$, the underflow of the computer makes it impossible to recompute $e^{Ah_{new}}$ through the Padé approximation in case $e^{Ah_{old}}$ is obtained by squaring and then h_{new} is obtained by reduction but Ah_{new} is too small to be the argument for exponentials. To monitor this we may either constrain the maximal allowable step size or reduce the order P used in Padé approximation. In either case this does not contradict the A-stability since numerical results show a strong tendency to increase the step size.

6.8. NONLINEAR SYSTEM WITH COMPLEX MIXED TYPE EIGENVALUES [17]

Problem:

$$y' = U \begin{bmatrix} -\beta_1, & \beta_2, & 0, & 0 \\ -\beta_2, & -\beta_1, & 0, & 0 \\ 0, & 0, & -\beta_3, & 0 \\ 0, & 0, & 0, & -\beta_1 \end{bmatrix} U^T y + U \begin{bmatrix} \frac{z_1^2 - z_2^2}{2} \\ z_1 z_2 \\ z_3^2 \\ z_4^2 \end{bmatrix};$$

$$y(0) = U \begin{bmatrix} -2 \\ 0 \\ -1 \\ -1 \end{bmatrix} \quad \text{where} \quad z = Uy .$$

Exact Solution:

$$z(x) = \begin{bmatrix} \frac{2(\beta_1\omega_1 - \beta_2\omega_2)}{\omega_1^2 + \omega_2^2} \\ \frac{2(\beta_2\omega_1 + \beta_1\omega_2)}{\omega_1^2 + \omega_2^2} \\ \frac{\beta_3}{1-(1+\beta_3)e^{\beta_3 x}} \\ \frac{\beta_4}{1-(1+\beta_1)e^{\beta_4 x}} \end{bmatrix} \quad \text{and} \quad y(x) = Uz(x)$$

where $\omega_1 = 1 - e^{\beta_1 x}[(1+\beta_1)\cos \beta_2 x - \beta_2 \sin \beta_2 x]$

$\omega_2 = e^{\beta_1 x}[\beta_2 \cos \beta_2 x + (1+\beta_1)\sin \beta_2 x]$

Eigenvalues:

$$z_1 - \beta_1 \pm |z_2 - \beta_2| i\,, \quad 2z_3 - \beta_3, \quad 2z_4 - \beta_4\,.$$

Test Parameters:

Local Error Tolerance $\quad eps = 10^{-4}$

Time Interval $\quad x \in [0,50]$

$\beta_3 = 100$, $\beta_4 = 0.1$, β_1 and β_2 are varied.

Test Results:

	$\beta_1 = -10$; $\beta_2 = 0$	$\beta_1 = 1$; $\beta_2 = 100$	$\beta_1 = 10$; $\beta_2 = 100$	$\beta_1 = 10$, $\beta_2 = 10$
AVE	2.84	3.45	3.70	5.35!
STEPS	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

Notation:

! This work was done with local error tolerance $eps = 10^{-6}$.

Remarks:

This set of problems will have different special characters depending on different values of β_1 and β_2. When $\beta_1 = -10$ and $\beta_2 = 0$, all eigenvalues are real

numbers because $z_2(x) \equiv 0$. But there are positive values at the transient region, so this problem is similar to that in the proceding section. When $\beta_1 = 1$ and $\beta_2 = 100$, all eigenvalues have negative real parts but 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 situations (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. When $\beta_1 = -10$ and $\beta_2 = 10$, this is a mixed type problem with eigenvalues having positive real part in the beginning and then turning negative eventually. The performance is very poor. Particularly, there are frequent reductions on the step sizes whereas the "global error" implies that larger step sizes could be used. Probably this is an indication that the error control mechanism built in Chapter 4 is not perfect.

7. CONCLUSION AND RECOMMENDATION

The numerical examples in the preceding chapter have demonstrated the workability of this nonlinear multistep method which we have developed to solve stiff initial value problems. Its efficiency can be certainly improved with more careful programming work. We draw the following conclusion from our investigation:

The formulas derived in Section 2.1 are somewhat cumbersome to apply to practical situations. But since the method is endowed with the A-stability property, the feature of varying step sizes to avoid instability is not important. Therefore in those cases where memory storage is limited, this fixed step scheme can be used.

The numerical techniques described in Section 2.3.2 deserve further study because integrals involving matrix exponentials occur frequently in optimal linear regulator problems.

As was mentioned in Chapter 3, when $A = 0$, most of the quantities developed can be reduced to scalars which are used by Shampine and Gordon in their famous subroutine STEP. Because of the closeness, it is plausible that these two codes can be combined with a built-in, stiff on

and non-stiff off, automatic switch for general problems. To accomplish this goal the primary work would be the detection of stiffness which, unfortunately, is still on a heuristic basis and needs more exploration.

Because of the presence of large Lipschitz constants and the use of large step sizes, the error estimates derived in Chapter 4 do not work completely satisfactorily. We believe that more realistic error estimates would give faster convergence. The global error analysis is still a wide open problem.

Besides the weakness of demanding lots of memory, this nonlinear method has not yet been completely developed to handle the case when A is time dependent. Although at the current stage a periodic updating mechanism, such as the one suggested in (1.4), can be used, we wish to establish an automatic one in the algorithm so that a much larger class of problem can be handled.

Finally, more investigation needs to be done to explain the stumbling behaviors of this nonlinear method when applied to problems in Section 6.8.

LIST OF REFERENCES

LIST OF REFERENCES

1. G. Birkhoff and R. Varga, Discretization errors for well set Cauchy problems, I., J. Math. Phys., 44 (1965), pp. 1-23.

2. G. Bjurel, G. Dahlquist, B. Lindberg and S. Linde, Survey of stiff ordinary differential equations, Report NA70.11, Department of Information, Processing, Royal Inst. of Tech., Stockholm, 1970.

3. J.L. Blue and H.K. Gummel, Rational approximation to matrix exponential for systems of stiff differential equations, J. Comput. Phys. 5 (1970), pp. 70-83.

4. D. Calahan, Numerical solution of linear system with widely separated time constants, Proc. I.E.E.E., 55 (1967), pp. 2016-2017.

5. S.L. Campbell and C.D. Meyer, Generalized inverses of linear transformations, Pitman, London, 1979.

6. J. Certaine, The solution of ordinary differential equations with large time constants, in Mathematical Methods for Digital Computers, ed. A. Ralston and H. Wilf, Wiley, N.Y., 1960, pp. 128-132.

7. G.G. Dahlquist, A special stability problem for linear multistep method, BIT, 3 (1963), pp. 27-43.

8. B.L. Ehle and J.D. Lawson, Generalized Runge-Kutta processes for stiff initial-value problems, J. Inst. Maths. Applic., 16 (1975), pp. 11-21.

9. W.H. Enright, T.E. Hull and B. Lindberg, Comparing numerical methods for stiff systems of O.D.E.'s, BIT, 15 (1975), pp. 10-48.

10. W. Fair and Y.L. Luke, Pade approximation to the operator exponential, Numer. Math., 14 (1970), pp. 379-382.

11. C.W. Gear, The automatic integration of stiff ordinary differential equations, in Information Processing, 68 ed. A.J.H. Morrel, North Holland Publishing Company, Amsterdam, pp. 187-193.

12. C.W. Gear, Numerical initial value problems in ordinary differential equations, Prentice-Hall, Englewood Cliffs, N.J., 1971.

13. C.W. Gear, Numerical solution of ordinary differential equations: is there anything left to do?, SIAM Review, 23 (1978), pp. 10-24.

14. P. Henrici, Discrete variable methods in ordinary differential equations, Wiley, N.Y., 1962.

15. R.K. Jain, Some A-stable methods for stiff ordinary differential equations, Math. Comp., 26 (1972), pp. 71-77.

16. F. Krough, A variable step variable order multistep method for the numerical solution of ordinary differential equations, in Information Processing 68, ed. A.J.H. Morrell, North Holland Publishing Company, Amsterdam, pp. 194-199.

17. F. Krough, On testing a subroutine for numerical integration of ordinary differential equations, JACM, 20 (1973), pp. 545-562.

18. J.D. Lambert and S.T. Sigurdsson, Multistep methods with variable matrix coefficient, SIAM J. Numer. Anal., 9 (1972), pp. 715-734.

19. J.D. Lambert, Computational methods in ordinary differential equations, Wiley, London, 1976.

20. J.D. Lawson, Generalized Runge-Kutta processes for stable system with large Lipschitz constant, SIAM J. Numer. Anal., 4 (1967), pp. 372-380.

21. J.D. Lawson, Some numerical methods for stiff ordinary and partial differential equations, in Proc. Second Manitoba Conference on Numer. Math., 1972, pp. 27-34.

22. D. Lee and S. Preiser, A class of nonlinear multistep A-stable numerical methods for solving stiff differential equations, Comput. Math. Appl., 4 (1978), pp. 43-51.

23. W. Liniger and R. Willoughby, Efficient integration methods for stiff systems of ordinary differential equations, SIAM J. Numer. Anal., 7 (1970), pp. 47-66.

24. W. Miranker, Matricial difference scheme for integrating stiff systems of ordinary differential equations, Math. Comput., 25 (1971), pp. 717-728.

25. W. Miranker, Numerical methods for stiff equations and singular perturbation problems, D. Reidel Publishing Company, Holland, 1981.

26. W.D. Mutphy, Hermit interpolation and A-stable methods for stiff ordinary differential equations, Appl. Math. Comput., 3 (1977), pp. 103-112.

27. C. Molen and C. van Loan, Nineteen dubious ways to compute the exponential of a matrix, SIAM Review, 20 (1978), pp. 801-836.

28. L.F. Shampine and M.K. Gordon, Computer solution of ordinary differential equations; the initial value problem, Freeman, San Francisco, 1975.

29. L.F. Shampine and C.W. Gear, A user's view of solving stiff ordinary differential equations, SIAM Review, 21 (1979), pp. 1-17.

30. L.F. Shampine, Stiffness and non-stiff differential equation solvers, in Numerische Behandling von Differentialgleichungen, ISNM, 27 (1975), pp. 287-301.

31. L.F. Shampine, Stiffness and nonstiff differential equation solver, II, detecting stiffness with Runge-kutta method, ACM. Trans. Math. Software, 3 (1977), pp. 44-53.

32. J.M. Varah, Stiffly stable linear multistep method of extended order, SIAM J. Numer. Anal., 15 (1978), pp. 1234-1246.

33. R.S. Varga and E.B. Saff, Pade and rational approximation, theory and application, Academic, N.Y., 1977.

34. R.S. Varga, On higher order stable implicit method for solving parabolic partial differential equations, J. Math. Phys. 40 (1961), pp. 220-231.

35. R.C. Ward, Numerical computation of the matrix exponential with accuracy estimate, SIAM J. Numer. Anal., 14 (1977), pp. 600-610.

36. G. Wanner, E. Hairer and S.P. Norsett, Order stars and stability theorems, BIT, 18 (1978), pp. 475-489.

37. R.A. Willoughby, Stiff differential systems, Plenum Press, N.Y., 1974.