SEMI-IMPLICIT BACKWARD DIFFERENTIATION FORMULAS

BY

KRIS STEWART

B.A., UNIVERSITY OF CALIFORNIA, SAN DIEGO, 1973

M.S., SAN DIEGO STATE UNIVERISTY, 1979

DISSERTATION

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Doctor of Philosophy in Mathematics

The University of New Mexico
Albuquerque, New Mexico

April, 1987

KRIS STEWART

Candidate

Mathematics

Department

This dissertation is approved, and it is acceptable in quality and form for publication on microfilm:

Approved by the Dissertation Committee:

[illegible], Chairperson

Lawrence F. Shampine

Steven A. Pruess

Stanly Steinberg

Accepted:

Dean, Graduate School

Date

SEMI-IMPLICIT BACKWARD DIFFERENTIATION FORMULAS

BY

KRIS STEWART

ABSTRACT OF DISSERTATION

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Doctor of Philosophy in Mathematics

The University of New Mexico
Albuquerque, New Mexico

April, 1987

SEMI-IMPLICIT BACKWARD DIFFERENTIATION FORMULAS

Kris Stewart

B.A. Mathematics, University of California, San Diego, 1973

M.S. Computer Science, San Diego State University, 1979

The Semi-Implicit Backward Differentiation Formulas (SIBDF) form a variable order family of methods designed for stiff ordinary differential equations. They are based on a prediction followed by a fixed number of corrections obtained by an approximate Newton scheme. We present the method and relate it to the BDF.

Stability models for the SIBDF are explored and related to stabilty for the BDF. One model yields upper bounds on the spectral radius of the relative correction matrix error for each order of the SIBDF. Numerical estimation of the largest eigenvalue can then be used to predict the need for updated correction matrices and determine the number of corrections to perform in an order dependent manner.

The local truncation error is derived. It shows how accuracy depends on the error of the predictor and the number of corrections. When implemented, a code will vary order and step size. A low cost matrix modification technique is developed to reduce the number of factorizations of the matrix due to changes of order or step size.

Experimental evidence confirms the usefulness of these techniques for stiff problems.

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

Chapter 1. Introduction

The numerical solution of stiff ordinary differential equations (ODEs) is the focus of this dissertation. An excellent introduction to the numerical implications of stiffness is given in the SIAM Review article by Shampine and Gear [47]. Following their lead, we say that a problem is stiff if the step size is limited more severely by stability than by accuracy when using one of the classical methods such as the Adams. Methods more appropriate for solving stiff ODEs require a set of algebraic equations be solved to some fixed accuracy at each time step.

The Backward Differentiation Formulas (BDF) have proven to be an effective family of formulas for stiff ODEs. First explored by Curtiss and Hirschfelder [10], they were popularized by Gear [15,16,17] in the code DIFSUB and by Brayton et. al [1]. Production codes cased on DIFSUB were made available by Hindmarsh, first GEAR [23] and then LSODE [25].

Most implementations of BDF methods obtain an accurate solution to the algebraic equations by using an explicit predictor followed by Newton-like corrections. Reliable convergence tests are difficult. This suggests an alternative family of formulas we call the Semi-Implicit BDF (SIBDF) based on a predictor and a fixed number of corrections formed in an inexact Newton process. This fixed number, in theory, is held constant throughout the integration interval. The distinction in method is similar to that between an iterated Adams-Moulton corrector, as in DIFSUB (and LSODE), and a PECE Adams-Bashforth-Moulton method, as in STEP [48] by Shampine and Gordon.

The purpose of this dissertation is to demonstrate and investigate the differences between the SIBDF and the BDF in both analysis and practice. The SIBDF first received analytical attention from

Klopfenstein [29]. Details in the implementation of the BDF that do not affect its analysis play an important role in the performance of the SIBDF. Our analysis of these matters leads us to make specific algorithmic design decisions.

With the SIBDF, we gain the simplicity of a fixed number of corrections, but introduce more parameters into the specification of the method. An important quantity in the analysis of both stability and accuracy is the spectral radius of the relative error matrix which results from the inexact correction matrix. We call these formulas Semi-Implicit to emphasize that a linear algebraic system must be solved at each step. These formulas have been implemented by Stewart and Krogh [52] where they have proven effective in solving stiff ODEs.

We present the formulas and fundamental theory for the BDF in Chapter 2 to make the exposition self-contained, define notation and emphasize points. The recommended reference is Hall and Watt [20]. The accuracy of the formulas is a consequence of the theory of the interpolating polynomial. The results of zero stability ($h \to 0$) are stated, but more is required in practice. To ensure numerical stability when the step size is nonzero for the linear, constant coefficient model system, one must satisfy the algebraic equations either

a) exactly (yielding the homogeneous BDF difference equation) or

b) to some fixed accuracy (yielding an inhomogeneous difference equation for the iterated BDF).

The formulas for the SIBDF are presented in Chapter 3. Since the form of the predictor is significant, we give examples to highlight the performance of a 4-step corrector formula when coupled with a 3-step predictor formula or a 4-step predictor formula. The SIBDF depend explicitly on the relative error in the 'Newton-like' matrix used to compute corrections. Examples are given to show the performance of the two predictor forms when there is no error in the correction matrix and when error is introduced.

In Chapter 4 we present the <u>vector generalized difference equation</u> (<u>VGDE</u>) developed by Krogh and Stewart [35] which models stability for the SIBDF for the linear, constant coefficient system. Through this difference equation, we investigate how the SIBDF methods inherit the stability properties of the BDF. The VGDE is not solvable in the general case, and we explore assumptions that will yield solutions:

a) vector test problem with
   i) finite h and no correction matrix error
   ii) $h \to \infty$ and matrix error
b) scalar test problem with finite h and matrix error.

Case ii) yields the <u>Asymptotic Absolute Stability</u> (<u>AAS</u>) model, presented by Klopfenstein in [29].

To balance the discussions of stability, we consider the accuracy of the SIBDF in Chapter 5. The local truncation error of the PEC method (only one correction) for a nonlinear system is derived and compared with the BDF. We then consider local truncation error for the linear system and a fixed number (greater than one) of corrections. We end with a discussion of the practical estimation of the local error.

In Chapter 6 the implications of relative matrix error are examined. In Section 6.a, we examine low cost methods to reduce the error in the correction matrix. In Section 6.b we discuss the ways in which a practical code uses estimates of relative matrix error. This provides an order-dependent mechanism to predict the need to form a new correction matrix and separately the need for a new Jacobian of the differential equation. Experimental evidence is cited [52] to demonstrate that the number of corrections need not remain fixed throughout the entire integration as assumed for the Asymptotic ($h\to\infty$) Absolute Stability model. Thus, these same matrix error estimators provide an effective mechanism to determine the number of corrections that need be performed. Finally, we discuss how relative matrix error is estimated in a code.

We conclude with a discussion of how the SIBDF provide an effective alternative family of formulas for solving stiff ODEs.

Chapter 2. The Backward Differentiation Formulas (BDF)

We begin with a brief review of the BDF to establish the accuracy and absolute stability characteristics of the method. Three cases of absolute stability are considered:

(1) $h \to 0$

and the case of finite, nonzero, h when the corrector is

(2) solved exactly and

(3) appproximately satisfied.

Consider the initial value problem

$$y' = f(x,y) \qquad y \in \mathbb{R}^n \tag{2-1}$$
$$y(x_0) = y_0$$

We assume f satisfies a Lipschitz condition

$$\|f(x,y) - f(x,z)\| \le L \|y-z\| \tag{2-2}$$

for some positive constant L. This assures the existence of a unique solution near the initial point (x_0,y_0).

2.a. The Formulas

Let $y_n, \ldots, y_{n-k}$ denote the computed solution of (2-1) at k+1 equally spaced points, $x_{n-j} = x_n - jh$, $j=0,\ldots,k$. The basis for the k^{th} order, k+1 step BDF at x_n is given by the k^{th} degree polynomial, $\wp_{k+1,n}(x)$, defined by k+1 interpolatory conditions

$$\wp_{k+1,n}(x_j) = y_j \qquad j=n,\ldots,n-k$$

We define y'_n by

$$y'_n = \wp'_{k+1,n}(x_n)$$

and the discrete approximation to $y'(x) = f(x,y)$, by

$$y'_n = f(x_n,y_n)$$

This is an implicit condition which we call the <u>collocation condition</u>. Also, one of the interpolatory conditions is implicit, $\wp_{k+1,n}(x_n) = y_n$, for the unknown value y_n.

The polynomial in the case of constant stepsize is

$$\wp_{k+1,n}(x) = y_n + \frac{(x-x_n)}{h}\nabla y_n + \frac{(x-x_n)(x-x_{n-1})}{h^2\ 2!}\nabla^2 y_n$$
$$+ \ldots + \frac{(x-x_n)\ldots(x-x_{n-k+1})}{h^k\ k!}\nabla^k y_n$$

and the <u>collocation condition scaled by h</u> is

$$\sum_{r=1}^{k} \delta^*_r \nabla^r y_n = h\, f(x_n,y_n) \tag{2-3}$$

The coefficients will be referred to as δ^*_r $(= \frac{1}{r})$ in analogy with the explicit Adams-Bashforth coefficients, γ_r, and implicit Adams-Moulton coefficients, γ^*_r.

In Lagrangian form (Gear [17] p. 217), we have

$$y_n = \sum_{r=1}^{k} \alpha_r \, y_{n-r} + h \, \beta_0 \, f(x_n, y_n) \tag{2-4}$$

When k=1, we have $\alpha_1=1$, $\beta_0=1$ to yield the Backward Euler method.

2.b. Accuracy

We define the local truncation error to be the amount by which the true solution fails to satisfy the scaled collocation condition (Eqn. 2-3). This is obtained by differentiating the error of the interpolating polynomial and evaluating at x_n. Using $f(x_n, y(x_n)) = y'(x_n)$, in backward difference form this is

$$\tilde{\tau}_n = h \, y'(x_n) - \sum_{r=1}^{k} \delta_r^* \, \nabla^r y(x_n) = \delta_{k+1}^* \, h^{k+1} \, y^{(k+1)}(\xi) \tag{2-5}$$

with $\xi \in [x_{n-k}, x_n]$. This is equivalent, up to a scale factor, to the difference between the true solution and the method given exact data, $y_j = y(x_j)$, j=n-1,...,n-k and $y'_n = y'(x_n)$. Using the right hand side of the Lagrangian form (Eqn. 2-4) yields

$$\begin{aligned} \tau_n = y(x_n) - y_n &= y(x_n) - \sum_{r=1}^{k} \alpha_r \, y(x_{n-r}) - h \, \beta_0 \, y'(x_n) \\ &= h \, \beta_0 \left[\mathcal{P}'_{k+1,n}(x_n) - y'(x_n) \right] = - \beta_0 \tilde{\tau}_n \end{aligned} \tag{2-6}$$

In contrast, the explicit Adams-Bashforth formula through these past values of x_i would have truncation error $\gamma_k \, h^{k+1} \, y^{(k+1)}(\nu)$. The implicit Adams-Moulton formula with these back points would have truncation error $\gamma_{k+1}^* \, h^{k+2} \, y^{(k+2)}(\xi)$. (See Henrici [21] pp. 203-204)

2.c. Absolute Stability

Early discussions of stability were not concerned with stiff problems, where $h\lambda$ is expected to be large. They focussed on the case of $h \to 0$, calling this <u>zero stability</u>. The difference equation (2-3) is solvable then since the derivative $y'(x) = f(x,y)$ no longer appears. For methods based on numerical integration such as Adams, this led to a stability condition that was easily characterized. To quote from the ground-breaking text written by Henrici in 1962, <u>Discrete Variable Methods in Differential Equations</u> [21, p. 220], (in Henrici's notation below, q refers to order (we use k), r=0 specifies the BDF, $\rho(\zeta)$ is the first characteristic polynomial (see Eqn 2-10))

> "For methods based on numerical differentiation there is no such simple general result, because the location of the roots of $\rho(\zeta)$ is not so easily ascertained. Either by numerical calculation or by applying an algebraic criterion due to Hurwitz it can be shown that for r=0 (differentiation at the foremost point) these methods are stable for q=1,2,...,6. For q=7, and very probably for all larger q, the methods are unstable."

In 1972, Cryer (Thm. 1.1, [5]) published a proof that the BDF have a region of stability as $h \to 0$ if, and only if, the order is less than 7.

Though one requires zero stability for a usable method, stiff problems also require that an effective method be stable for large values of h. For finite h, the derivative explicitly appears, and a convenient model that yields results is given by the linear, constant coefficient system

$$y' = J\,y \qquad y \in \mathbb{R}^n$$

We apply the method (Eqn. 2-4) to obtain the <u>vector difference equation</u>

$$(2\text{-}7) \qquad (I - h\beta_0 J)\, y_n = \sum_{r=1}^{k} \alpha_r\, y_{n-r}$$

Assume $I - h\beta_0 J$ is invertible, else we need to consider the topic of Differential-Algebraic Equations [40]. Eqn. 2-7 will then be directly solvable at the current point, x_n, for the vector of difference solutions in terms of past information

$$y_n = (I - h\beta_0 J)^{-1} \sum_{r=1}^{k} \alpha_r\, y_{n-r} \quad .$$

Let J have a full set of eigenvectors forming the columns of a matrix T. The same similarity transformation that diagonalizes J,

$$T\, J\, T^{-1} = \Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n),$$

will diagonalize the vector difference equation (Eqn. 2-7) and yield a solution $w_n = Ty_n$, whose i^{th} component, $[w_n]_i$ satisfies

$$(2\text{-}8) \qquad [w_n]_i = \frac{1}{1-h\beta_0\lambda_i} \sum_{r=1}^{k} \alpha_r\, [w_{n-r}]_i$$

Once the equations are uncoupled, we may consider an arbitrary component, call it w, of the transformed vector Ty along with λ, the corresponding eigenvalue of J. Focussing on this single component, the true solution to the differential equation, $w(x_n)$, yields a local truncation error from Eqn. 2-6 of

$$w(x_n) - \sum_{r=1}^{k} \alpha_r \, w(x_{n-r}) - h\beta_0\lambda \, w(x_n) = \tau_n$$

The actual computed value, w_n, has been subjected to rounding errors or other perturbations (e.g. not exactly satisfying the collocation condition), and therefore satisfies a perturbed version of (2-4)

$$w_n - \sum_{r=1}^{k} \alpha_r \, w_{n-r} - h\beta_0\lambda \, w_n = \delta_n$$

Subtracting, we have the global error, $\epsilon_n = w(x_n) - w_n$, of the transformed problem satisfying

(2-9) $$(1 - h\beta_0\lambda) \, \epsilon_n - \sum_{r=1}^{k} \alpha_r \, \epsilon_{n-r} = \tau_n - \delta_n$$

causing us to examine the <u>stability polynomial</u>

(2-10) $$\begin{aligned} \pi(z;h\lambda) &= \rho(z) + h\lambda \, \sigma(x) \\ &= (1 - h\beta_0\lambda) \, z^k - \sum_{r=1}^{k} \alpha_r \, z^{k-r} \end{aligned}$$

The homogeneous version of Eqn. 2-9 will have solutions that do not grow as long as this stability polynomial has all its roots strictly within the unit circle for a given value of $h\lambda$. We call this the <u>root condition</u> which characterizes <u>absolute stability</u> (Definition 10 from Hall and Watt [19] p. 37) for finite, nonzero step size.

The particular solution to the inhomogeneous difference equation (Eqn. 2-9) will be kept small if in addition to the root condition above on $\pi(z;h\lambda)$, we have τ_n and δ_n small. This is a more practical definition of stability when the BDF are implemented in a computer code and we call this <u>absolute stability for the iterated BDF</u> (see Section 2.d) to allow for the local truncation error of the formula and that the collocation condition (Eqn. 2-4) is not satisfied exactly. τ_n can be kept small by controlling the stepsize, δ_n can be kept small by satisfying the collocation condition well.

To appreciate what restrictions are placed on the method by the root condition, we focus first on absolute stability of the BDF. For a given value of $h\lambda$, there are k solutions to the scalar difference equation, the k roots of $\pi(z;h\lambda)$. Let us call them $z_1, \ldots , z_k$. Without great loss of generality, we can assume the z_j's are distinct. (In the general case, the difference solution will have polynomial-like terms such as jz_j, which do not substantially alter the final conclusions.) One could compute these k roots and verify their magnitude is less than one for values of $h\lambda$ of interest. This is a laborious task.

Alternate, more compact, schemes have been pursued. We know the behavior as $h \to 0$: the BDF are stable only when the order is 6 or less. Another characterization is the $A(\alpha)$ stability region. From Gear [17, p. 213] Definition 11.2, "A method is $A(\alpha)$-stable, $\alpha \epsilon (0,\pi/2)$, if all numerical approximations to $y'=\lambda y$ converge to 0 as $n \to \infty$ with h fixed for all $|\arg(-\lambda)| < \alpha$, $|\lambda| \neq 0$." Analytical techniques were explored by Norsett [38] to specify the angle, α, for $A(\alpha)$ stability. Table 1 (p. 263 of [38]) gives:

Table 1

α_{max} for $A(\alpha)$ Stability Boundary of the BDF

Order k:	1	2	3	4	5	6
Angle α:	90^o	90^o	$88^o27'$	$73^o14'$	$51^o50'$	$18^o47'$

$A(\pi/2)$ stable indicates the method is stable in the entire left half plane, usually called A-stable.

To gain a broader view of the restrictions on $h\lambda$ for different order formulas, we pursue the boundary locus technique to characterize absolute stability graphically (See Lambert [36] p. 82). Instead of finding the k roots, z_j, of $\pi(z;h\lambda)$ for fixed $h\lambda$, one maps the unit circle in the z-plane, into the $h\lambda$-plane using the difference equation in backward difference form (Eqn 2-3). This is a continuous transformation of complex values and therefore takes closed curves into closed curves.

It is well known that a solution to a linear, constant coefficient, homogeneous difference equation is of the form $y_n = c\ z^n$, where z is any root of the corresponding characteristic polynomial, $\pi(z;h\lambda)$, and c is some constant [36, p.8]. Considerations for an effective computation lead us to substitute this into the difference form of the collocation condition (Eqn. 2-3), cancelling the coefficient c to yield

$$h\lambda\ z^n = \sum_{r=1}^{k} \frac{1}{r} \nabla^r z^n$$

which is satisfied by any root of $\pi(z,h\lambda)$. The identity

$$\nabla^r z^n = z^n \left(1 - \frac{1}{z}\right)^r \tag{2-11}$$

easily proved by induction, then yields

$$h\lambda = \sum_{r=1}^{k} \frac{1}{r} \left(1 - \frac{1}{z}\right)^r$$

The interior of the unit circle in the z-plane contains all possible roots of the difference equation with magnitude less than one. The argument will be simplified by examining an intermediate transformation.

$$w = 1 - \frac{1}{z}$$

is a one to one transformation of the interior of the unit circle in the z-plane to the exterior of a disk in the w-plane centered on the real axis at one with radius one.

The homogenous difference equation is now a polynomial in w

$$f(w) = \sum_{r=1}^{k} \frac{1}{r} w^r - h\lambda$$

For $h\lambda$ sufficiently large, the roots, w_i, will be large. In fact, f(w) will have k distinct roots since f'(w) is nonzero. We claim that

$$|h\lambda| > \sum_{r=1}^{k} \frac{1}{r} 2^r \tag{2-12}$$

characterizes when $h\lambda$ is sufficiently large to insure all roots of the original difference equation in the z-plane will have its roots within the unit circle. We use the triangle inequality to show that all roots of f(w) must satisfy $|w| > 2$ if $|h\lambda|$ satisfies Eqn. 2-13.

$$|f(w)| = |h\lambda - \sum_{r=1}^{k} \frac{1}{r} w^r| \geq |h\lambda| - |\sum_{r=1}^{k} \frac{1}{r} w^r| \geq |h\lambda| - \sum_{r=1}^{k} \frac{1}{r} |w|^r$$

If $|w| \leq 2$, then $|f(w)| \geq |h\lambda| - \sum_{r=1}^{k} \frac{1}{r} 2^r$ and by Eqn. 2-12 $|f(w)| > 0$, so f(w) can have no such root. $|w| > 2$ guarantees w outside the image in the w-plane of the unit disk in the z-plane, i.e. $|z| < 1$.

This says that when the magnitude of $h\lambda$ is large enough, the BDF are stable at all orders for all λ ,even for orders greater than 6 and when $\mathrm{Re}(\lambda) > 0$ and the differential equation $y'=\lambda y$ is itself unstable. This maximum value is attained for $z=-1$, i.e. $\theta = \pi$, corresponding to $h\lambda$

on the positive real axis. The boundary on the positive real axis in the $h\lambda$-plane is given by

Table 2. BDF Stability Bound for $h\lambda$
(If $|h\lambda| > STB(k)$, then BDF stable)

Order k	1	2	3	4	5	6	7
STB(k)	2	4	6.67	10.67	17.07	27.73	46.02

To compute the <u>regions of absolute stability</u>, we plot the image of the unit circle in the z-plane and so set $z = e^{i\theta}$

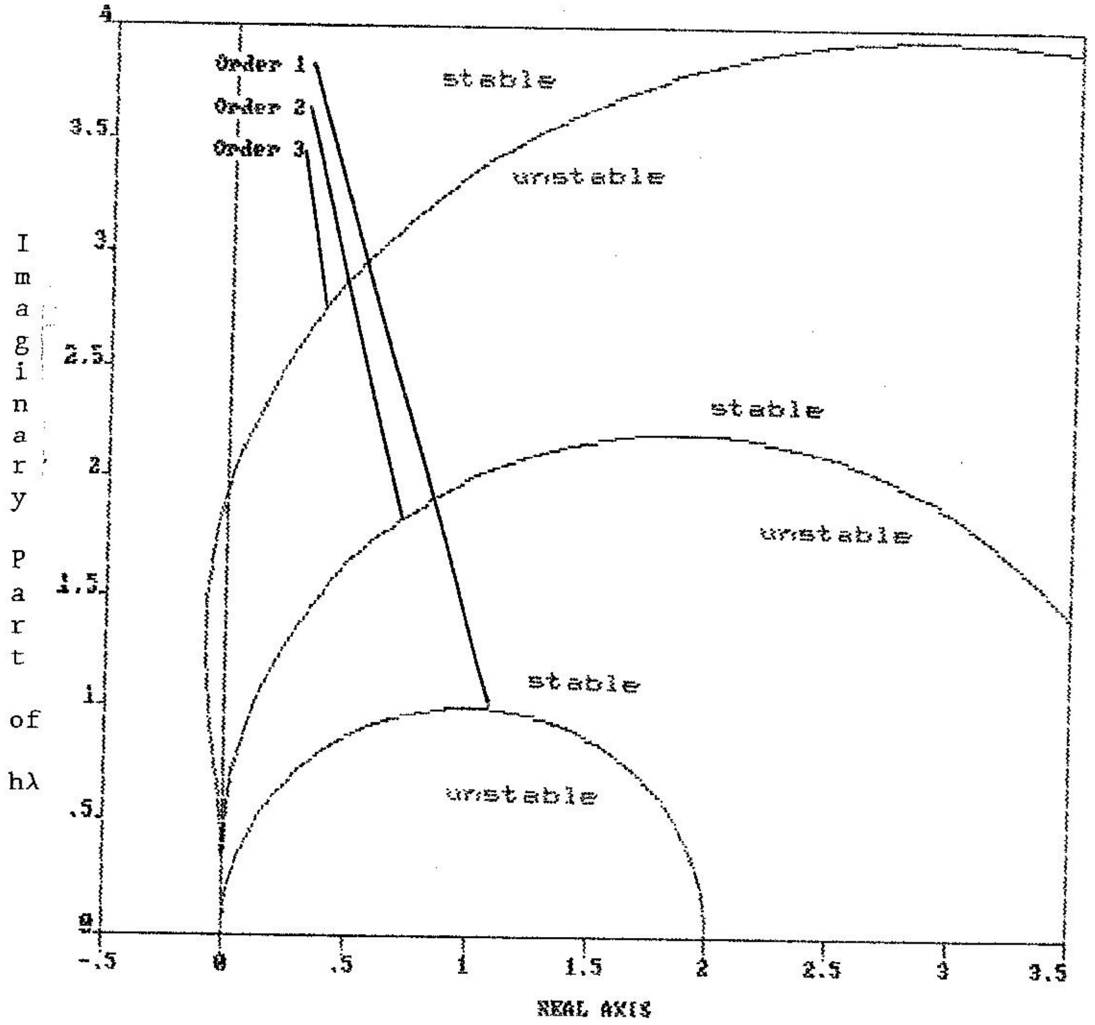
(2-13) $$h\lambda = \sum_{r=1}^{k} \frac{1}{r} (1 - e^{-i\theta})^{r}$$

One selects points on the unit circle in the z-plane, say values of $e^{i\theta}$ for θ between 0° and 360° in increments of 5°. This is substituted into (Eqn 2-12) to yield a value of $h\lambda$ which is then plotted as part of the image curve in the $h\lambda$-plane. Since we know now that the BDF are stable for large $h\lambda$, we see the exterior of the curves forms the region of absolute stability.

We present the results for the first through third order BDFs. Table 2 indicates where the absolute stability region will intersect the positive real axis.

Figure 1.

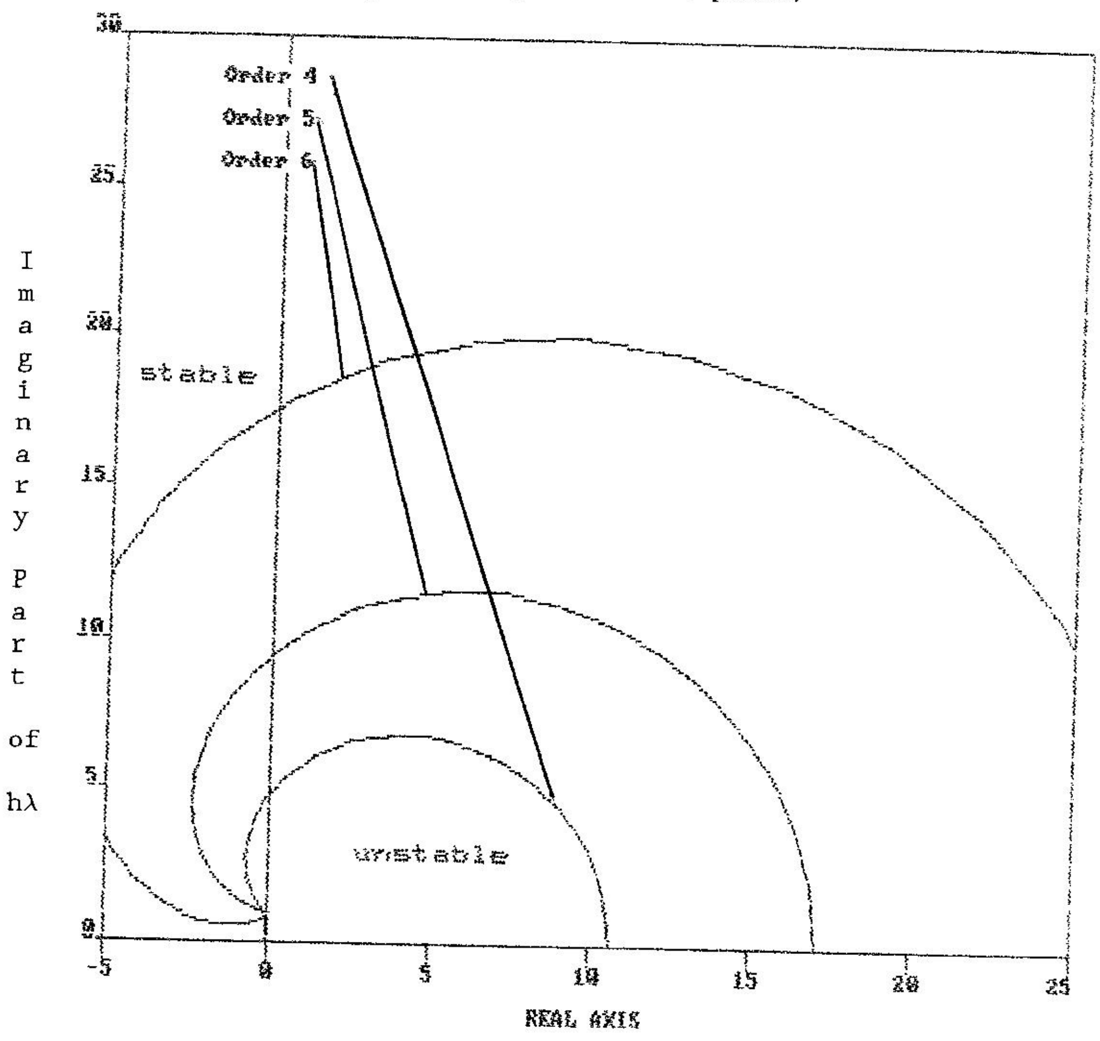
Absolute Stability for BDF orders 1 to 3
(Stability Boundaries in the $h\lambda$-plane)



(Equivalently $G = \hat{G}$ for SIBDF)

Next we present the absolute stability regions for the fourth through sixth order BDFs. Notice the correspondence with the α-angle given in Table 1 from Norsett's results.

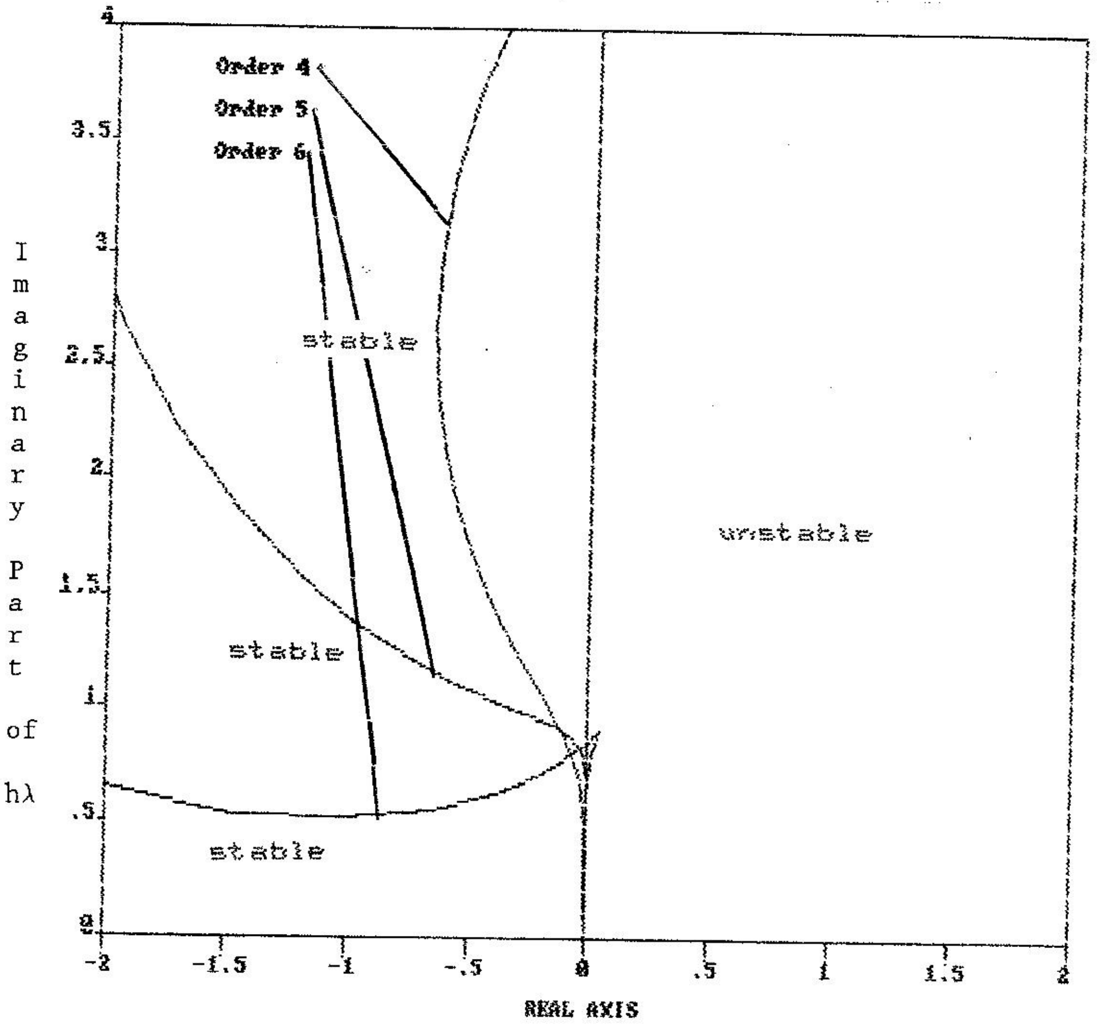
Figure 2.
Absolute Stability for the BDF orders 4 to 6
(Stability Boundary in the hλ-plane)



(Equivalently $G = \hat{G}$ for SIBDF)

Since the behavior near the origin for the higher order formulas is difficult to see in Figure 2, we refine the scale of the plot and focus on the behavior at the origin. We know these BDF are stable as $h \to 0$ from the results of Cryer.

Figure 3.
Absolute Stabilty for BDF order 4 to 6
(Detail near Origin of hλ-plane)



(Equivalently $G = \hat{G}$ for SIBDF)

The absolute stability of the transformed scalar (k+1)-step method (Eqn. 2-8) can also be examined using the k by k companion matrix, H, to the stability polynomial, $\pi(z;h\lambda)$. We pursue this approach because it helps clarify the stability when we no longer assume the collocation condition is solved exactly.

Let $W_n = \begin{bmatrix} w_n \\ \vdots \\ w_{n-k+1} \end{bmatrix} \in \Re^k$. The homogeneous difference equation (2-8)

takes the form

(2-14) $$W_n = H\, W_{n-1}$$

with

(2-15) $$H = \begin{bmatrix} \alpha_1/\Omega & \alpha_2/\Omega & & \cdots & \alpha_k/\Omega \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & & \\ & & & \ddots & \\ 0 & 0 & & & 1 \end{bmatrix}$$

and $\Omega = 1 - h\beta_0\lambda$.

The condition that all roots of π be less than one in modulus is equivalent to all eigenvalues of H being less than one in modulus. Given $\rho(H) < 1$, we have that any two solutions of the homogeneous difference equation, say W_n and U_n, satisfy

$$\| W_n - U_n \| = \| H^n (W_0 - U_0) \| \to 0 \text{ as } n \to \infty$$

in some norm. (See for example, Thm. 4.2.2 (Stability Theorem) Ortega [39, p. 71].)

2.d. Iterating the Implicit BDF Collocation Condition

In order to implement a method based on the BDF, one must consider how to handle the implicit collocation condition (Eqn 2-3). We adopt the notation Hindmarsh [23, pp. 28-29] uses in describing the formulas implemented in GEAR. One could attempt functional iteration on the nonlinear equation for y_n

$$g(y_n) = y_n - \beta_0 \; h \; f(x_n, y_n) - \sum_{r=1}^{k} \alpha_r \; y_{n-r} = 0, \tag{2-16}$$

but convergence will generally require $|h\beta_0 L| < 1$, where L is a Lipschitz constant (Eqn 2-2) for the ODE. This restriction on the step size when solving stiff problems is not acceptable.

Let y_n^{bdf} denote the exact solution to the homogeneous BDF difference equation (Eqn 2-4). Thus

$$g(y_n^{bdf}) = 0$$

In practical codes, the most common way of approximating the solution of $g(y) = 0$ is to predict a value for $y_n^{(0)}$ and improve it with an inexact Newton method. In DIFSUB [13] and its descendants, the predictor formula is

$$y_n^{(0)} = \varphi_{k+1,n-1}(x_n)$$

which has the same order as the k+1 step BDF used. The iteration matrix, $\hat{P}$, approximates

$$\hat{P}\, K\, \partial g/\partial y = I - \beta_o\, h\, \partial f/\partial y$$

with the iteration defined for $i = 0, 1, \ldots$ by

$$\hat{P}\, \Delta y_n^{(i)} = -g(y_n^{(i)})$$

$$y_n^{(i+1)} = y_n^{(i)} + \Delta y_n^{(i)}$$

The matrix controlling convergence to the fixed point is $I - \hat{P}^{-1}\ \partial g/\partial y$ since we have

$$y_n^{(i+1)} - y_n^{bdf} = y_n^{(i)} - \hat{P}^{-1}\, g(y_n^{(i)}) - y_n^{bdf}$$

$$= y_n^{(i)} - \hat{P}^{-1} \left\{ g(y_n^{bdf}) + [\nabla g^*]^t\, (y_n^{(i)} - y_n^{bdf}) \right\} - y_n^{bdf}$$

$$= (I - \hat{P}^{-1}\, [\nabla g^*]^t)\, (y_n^{(i)} - y_n^{bdf})$$

where $[\nabla g^*]^t$ comes from an application of a Mean Value Theorem to $g(y_n^{(i)})$ and denotes the matrix of transposed gradient vectors of the components of g, each evaluated at some point between $y_n^{(i)}$ and y_n^{bdf}.

When solving the model problem, $y' = Jy$, we have convergence controlled by

$$I - \hat{P}^{-1} (I - \beta_0 hJ) = I - \hat{P}^{-1}P$$

where it is hoped that the matrix $P = I-\beta_0 hJ$ is approximated well by $\hat{P}$.

A useful relationship results from the fact that Newton's method is exact for linear problems. If $\hat{P} = P$, the first corrrection $\Delta y_n^{(0)} = -P^{-1}g(y_n^{(0)})$ yields the BDF for the linear, constant coefficient system no matter what value is used for $y_n^{(0)}$.

(2-17) $$y_n^{bdf} = y_n^{(0)} - P^{-1} g(y_n^{(0)})$$

2.e. Stability for an Iterated BDF

In practice, this iteration must be terminated. There is much debate on an effective scheme. One can measure the <u>residual</u>

(2-18) $$e_n = g(y_n^{(i)}).$$

Hindmarsh [22, p. 3] uses a test based primarily on the size of successive differences

$$\delta^{(i)} = h\dot{y}_n^{(i)} - h\dot{y}_n^{(i-1)}$$

where $h\dot{y}_n^{(i)}$ is the scaled derivative of the polynomial passing through $y_n^{(i)}$ and the appropriate past points. Shampine [46] discusses techniques used in actual working codes along with the problems associated with unreliable convergence tests.

The stability of the computed results is of interest here. Define the k-vector representing the accepted solutions from the previous k points by

$$Y_n^{it} = \begin{bmatrix} y_n^{(2)} \\ y_{n-1}^{(1)} \\ \vdots \\ y_{n-k+1}^{(3)} \end{bmatrix}$$

where the superscripts emphasize that the number of iterations varies from step to step. The inhomogeneous difference equation satisfied by Y_n^{it} is

$$Y_n^{it} = H\, Y_{n-1}^{it} + E_n$$

The k-vector, $E_n = \begin{bmatrix} e_n \\ \vdots \\ e_{n-k+1} \end{bmatrix}$ relative to the current time step represents the residual of each of the previously accepted $y_{n-j}^{(i)}$, $j=0,\ldots,k-1$; typically i = 1,2 or 3. The inhomogeneous difference equation can be written

$$(2\text{-}19) \qquad \begin{aligned} Y_n^{it} &= H\, Y_{n-1}^{it} + E_n = H\,(H\, Y_{n-2}^{it} + E_{n-1}) + E_n \\ &= H^n\, Y_0 + \sum_{r=0}^{n-1} H^r\, E_{n-r} \end{aligned}$$

with Y_0 representing the initial starting error for the k+1 step method.

Absolute stability of the <u>homogeneous</u> difference equation (Eqn

2-14) requires $\rho(H)<1$ and guarantees that the difference of two solutions tends to zero as $n\to\infty$ for fixed h. Given that h has been chosen so that $h\lambda$ lies within the absolute stability region (Figures 1 and 2), we have for the inhomogeneous difference equation (Eqn. 2-18) that, in some norm, there exists a constant $\gamma < 1$ such that

(2-20) $$\|Y_n^{it}\| \le \gamma^n \|Y_0\| + \sum_{r=0}^{n-1} \gamma^r \|E_{n-r}\|$$

(Thm. 4.2.5, Ortega [34, p. 75]). Thus, in some norm, the global error of the actual computed solution will be uniformly bounded and not large as long as the root condition holds for the BDF and the residuals due to not satisfying the collocation condition exactly at each step are small enough with respect to $\gamma < 1$.

2.f. Implications of the BDF Analysis

The homogeneous difference equation for the BDF (Eqn. 2-7) leads to regions in the complex plane, primarily near the imaginary axis, where the performance of the method does not match the behavior of the differential equation. Table 2 demonstrates that the higher order BDF will be stable for stable problems ($Re(\lambda) < 0$) when the step size, h, is sufficiently large. Figures 1, 2 and 3 demonstrate that smaller values of h for stable problems can yield unstable methods.

The inhomogeneous difference equation (Eqn. 2-19) more closely describes the way the BDF are actually implemented by dropping the assumption that the implicit equations have a zero residual. We see from the analysis that it is not necessary to satisfy the collocation condition to get absolute stability. In addition to requiring $h\lambda$ strictly within the appropriate region of Figures 1 and 2, it is only necessary that the residuals, e_j, be kept sufficiently small. The scheme to ensure small residuals does not enter into the stability argument.

It is in the actual implementation that efficiency of the scheme to generate solutions sufficiently accurate becomes important. A true Newton iteration requires an accurate starting value along with an iteration matrix formed as each iterate is produced within each time step, a very costly procedure. A variety of Newton-like schemes with differing predictors and iteration matrix strategies are possible:

(1) A frequent modification is to use finite difference approximations to the ODE Jacobian. This does not significantly degrade performance (Thm. 5.4.1, [11]).

(2) Implementations simplify the Newton-like process by holding the Jacobian approximation constant over several time steps. This degrades the rate of convergence to linear but is an efficient scheme.

(3) There is a choice of predictor to start the iteration. A predictor of same order as corrector will use one more past point than a predictor with order one less.

(4) The step size and order will change over the integration interval in an effective method. This causes a change in the iteration matrix. Linear algebra costs can enter into step size and order control.

(5) A specific test (or combination of tests) must be formulated to terminate the iteration.

(6) A strategy must be formed to deal with nonconvergence, or insufficiently rapid convergence for an effective method. Step size reduction and/or reevaluation of the ODE Jacobian are remedies.

In Chapters 3 and 4 we will consider a method whose analysis and implementation are significantly different. The number of corrections computed is a fixed value. This is the idea that spawned the Semi-Implicit Backward Differentiation Formulas. The predictor and iteration scheme that disappeared in the stability analysis of the BDF will play a significant role in the stability analysis of the SIBDF.

Chapter 3. Semi-Implicit Backward Differentiation Formulas

We develop the formulas for a SIBDF method in this chapter. We develop formulas using two possible predictors and a fixed number of corrections computed with a correction matrix that is not the exact Newton matrix. Then, in Chapter 4, we proceed with the stability analysis which will determine the amount of relative error in the correction matrix that can be tolerated by a stable method.

We also discuss the implementation decision to use a variable step size formulation of the basic formulas instead of the more commonly used fixed step size formulation. This will require that special attention be paid to the effective implementation of the correction process. A technique that appears to work in practice is discussed in Section 6.a.

3.a. Higher order Corrector than Predictor: $P(k)\left[EC(k+1)\right]^m$

For simplicity, let the step size, h, and step number, k, remain constant. The method also involves a fixed number, m, of corrections. As a result of considerations of Asymptotic Absolute Stability (Chapter 4.d.i), we will prefer m=2.

The polynomial interpolating the computed solution at the previous k equally spaced points, $x_{n-1},\ldots,x_{n-k}$, is

$$\mathcal{P}_{k,n-1}(x) = y_{n-1} + \frac{(x-x_{n-1})}{h}\nabla y_{n-1} + \frac{(x-x_{n-1})(x-x_{n-2})}{h^2\;2!}\nabla^2 y_{n-1}$$

$$+ \ldots + \frac{(x-x_{n-1})\ldots(x-x_{n-k+1})}{h^{k-1}\;(k-1)!}\nabla^{k-1} y_{n-1}$$

One predicts the k-step solution at the forward point by extrapolating this polynomial to yield

(3-1) $$p_n = \mathcal{P}_{k,n-1}(x_n) = \sum_{r=0}^{k-1} \nabla^r y_{n-1}$$

The predicted derivative is

3-2) $$p'_n = \mathcal{P}'_{k,n-1}(x_n) = \frac{1}{h} \sum_{r=1}^{k-1} \delta_r \nabla^r y_{n-1}$$

where

(3-3) $$\frac{\delta_r}{h} = \frac{d}{dx}\left[\frac{(x-x_{n-1})\ldots(x-x_{n-r})}{h^r r!}\right]\Bigg|_{x=x_n} = \frac{1}{h}\sum_{i=1}^{r}\frac{1}{i} = \frac{1}{h}\sum_{i=1}^{r}\delta_i^*$$

the δ_i^* being the coefficients from the backward difference form of the BDF (Eqn 2-3). We have $\delta_r^* = \delta_r - \delta_{r-1}$.

All subscripts now refer to the n^{th} time step and will be suppressed along with reference to the independent variable, i.e. $f(y)$ denotes $f(x_n,y)$. Setting $y^{(0)} = p$, one computes m corrections by solving the sequence of linear systems

(3-4) $$\hat{G}(c,\hat{\alpha})\ \Delta y^{(i)} = -\ r^{(i)}$$

$$y^{(i+1)} = y^{(i)} + \Delta y^{(i)} \quad i=0,\ldots,m-1$$

The right hand side is given by the residual

(3-5) $$r^{(i)} = f(y^{(i)}) - p' - \frac{\delta_k}{h}(y^{(i)} - p)$$

Label the final result, $y^{mc} = y^{(m)}$, the superscript indicating <u>m</u> <u>c</u>orrections were performed.

The <u>correction matrix</u>, $\hat{G}(c,\hat{\alpha})$ is a parameterized approximation to a Newton matrix, G

$$G = \partial r^{(i)}/\partial y^{(i)} = \partial f(y^{(i)})/\partial y^{(i)} - \frac{\delta_k}{h} I$$

$\hat{G}$ is called the correction matrix rather than iteration matrix to emphasize the <u>fixed</u> number of corrections to be computed.

We begin by requiring $\hat{G}(c,\hat{\alpha})$ to have the form

(3-6) $$\hat{G}(c,\hat{\alpha}) = c\,(\hat{J} - \hat{\alpha} I)$$

where the parameter $\hat{\alpha}$ is an approximation to δ_k/h; the matrix $\hat{J}$ is an approximation to the Jacobian of the differential equation; and the parameter c is a scalar which is manipulated to help reduce the error when $\hat{\alpha}$ approximates the coefficient in the residual (3-5) in the case of the variable step size formulation (Section 3.c). Analysis of the accuracy and stability of the SIBDF method in Chapters 4 and 6 will guide us in choosing values for the parameters $\hat{\alpha}$ and c.

We think of (3-5) as a k+1 step residual associated with the k+1 step corrector because if one were so fortunate as to have $r^{(i)} = 0$ for some $y^{(i)}$, then setting $y_n = y^{(i)}$ one has

$$\varphi'_{k+1,n}(x_n) = f(y^{(i)}),$$

the collocation condition (Eq. 2-3) for the k+1 step BDF. This follows from the relation

$$\varphi_{k+1,n}(x) = \varphi_{k,n-1}(x) + \frac{(x-x_{n-k})\ldots(x-x_{n-1})}{h^k\ k!}\nabla^k y_n,$$

definitions (3-2) and (3-3), and

$$y_n - p = y_n - \sum_{r=0}^{k-1} \nabla^r y_{n-1} = \nabla^k y_n$$

As an illustration of possible choices for these parameters, Table 3 presents the results of solving

$$y' = -10\ y$$
$$y(1) = 1$$

(which has the true solution $y = e^{-10(x-1)}$) with m = 1, and k = 3 for a P_3EC_4 SIBDF method. The remaining method parameters are h = 0.2 and c = 1, $\hat{\alpha} = \delta_3/h = 2.08888/0.2 = 10.41666$, $\hat{J} = -10$ (corresponding to $\hat{G}$=G).

Table 3. P_3EC_4; c=1; $\hat{\alpha} = \delta_3/h$

3-step Predictor/ 4-step Corrector $h\lambda = -2$ $\mu = 0$

The exact past data:

y(0.600)	=	0.2479E-02
y(0.800)	=	0.3355E-03
y(1.000)	=	0.4540E-04

x	Predictor Error	Solution	$\lvert\Delta y^{(1)}\rvert$	Global Error	Local Error
1.20	1.602E-03	1.1981E-04	1.489E-03	1.137E-04	1.137E-04
1.40	5.425E-04	1.0517E-04	4.535E-04	1.043E-04	8.895E-05
1.60	1.275E-05	3.9372E-05	3.789E-05	3.926E-05	2.514E-05
1.80	8.291E-05	7.8253E-08	7.766E-05	6.302E-08	5.250E-06
2.00	1.272E-05	-6.2001E-06	6.513E-06	6.202E-06	6.211E-06
2.20	2.138E-05	-1.4593E-06	2.200E-05	1.460E-06	6.202E-07
2.40	1.450E-05	1.2909E-06	1.301E-05	1.291E-06	1.488E-06
2.60	1.876E-06	1.0421E-06	1.008E-06	1.042E-06	8.674E-07
2.80	2.347E-06	1.8356E-07	2.389E-06	1.836E-07	4.253E-08
3.00	1.310E-06	-1.5189E-07	1.133E-06	1.519E-07	1.767E-07

We see the stable correction process revealed in the global error.

To further illustrate the parameters, we model the situation where the matrix used in the correction process has the correct Jacobian, $\hat{J} = J$, but the coefficient is incorrect due to a change in the step number used, k, or the step size used, h, i.e. $\hat{\alpha} \neq \delta_k/h$. Since the differential equation is stable, we can hope that the numerical computation will remain stable. The computation with c = 1, $\hat{\alpha} = 7.01388 \neq \delta_3/h$, and $\hat{J} = J$ is presented in Table 4.

Table 4. P_3EC_4: c=1; $\hat{\alpha} \neq \delta_3/h$

3-step Predictor/ 4-step Corrector $h\lambda = -2$ $\mu = -.2$

The exact past data:

y(0.600)	=	0.2479E-02
y(0.800)	=	0.3355E-03
y(1.000)	=	0.4540E-04

x	Predictor Error	Solution	$\lvert\Delta y^{(1)}\rvert$	Global Error	Local Error
1.20	1.602E-03	-1.7795E-04	1.787E-03	1.841E-04	1.841E-04
1.40	3.105E-04	-8.6513E-05	2.481E-04	8.734E-05	6.243E-05
1.60	3.314E-04	-5.6892E-05	3.766E-04	5.700E-05	4.518E-05
1.80	8.138E-05	-1.3557E-05	7.553E-05	1.357E-05	5.858E-06
2.00	4.533E-05	-3.7432E-06	4.723E-05	3.745E-06	1.908E-06
2.20	2.694E-05	2.4041E-06	2.985E-05	2.404E-06	2.911E-06
2.40	4.559E-06	1.6238E-06	3.261E-06	1.624E-06	1.298E-06
2.60	6.304E-06	1.2223E-06	7.306E-06	1.222E-06	1.003E-06
2.80	1.034E-06	3.9637E-07	8.032E-07	3.964E-07	2.309E-07
3.00	9.076E-07	1.3853E-07	9.925E-07	1.385E-07	8.489E-08

It is seen that with this incorrect coefficient in the correction matrix, the process remains stable.

3.b. Predictor - Corrector of Same Order: $P(k+1)\left[EC(k+1)\right]^m$

One might ask the effect of using a k+1 step predictor with the k+1 step residual (3-5). The only change in the method above would be to use

(3-7) $$\tilde{p} = \sum_{r=0}^{k} \nabla^r y_{n-1}$$

and

(3-8) $$\tilde{p}' = \frac{1}{h} \sum_{r=1}^{k} \delta_r \nabla^r y_{n-1}$$

with the residual, $\tilde{r}$, defined by

(3-9) $$\tilde{r} = f(y) - \tilde{p}' - \frac{\delta_k}{h}(y - \tilde{p})$$

and the same correction matrix, $\hat{G}$ (3-6).

We first examine the case corresponding to Table 3, i.e. c=1 and $\hat{\alpha}=\delta_3/h = 10.41666$. The error in the predicted value is different, but the corrected solution is the same.

Table 5. P_4EC_4; c=1; $\hat{\alpha} = \delta_3/h$

4-step Predictor/ 4-step Corrector $h\lambda = -2$ $\mu = 0$

The exact past data:

y(0.400)	=	0.1832E-01
y(0.600)	=	0.2479E-02
y(0.800)	=	0.3355E-03
y(1.000)	=	0.4540E-04

x	Predictor Error	Solution	$\lvert\Delta y^{(1)}\rvert$	Global Error	Local Error
1.20	1.024E-02	1.1981E-04	1.035E-02	1.137E-04	1.137E-04
1.40	9.463E-04	1.0517E-04	1.035E-03	1.043E-04	8.895E-05
1.60	4.663E-04	3.9371E-05	4.914E-04	3.926E-05	2.514E-05
1.80	4.502E-05	7.8249E-08	3.977E-05	6.302E-08	5.250E-06
2.00	6.493E-05	-6.2001E-06	7.114E-05	6.202E-06	6.211E-06
2.20	2.789E-05	-1.4592E-06	2.851E-05	1.460E-06	6.202E-07
2.40	7.497E-06	1.2909E-06	8.986E-06	1.291E-06	1.488E-06
2.60	1.113E-05	1.0421E-06	1.200E-05	1.042E-06	8.674E-07
2.80	3.355E-06	1.8356E-07	3.397E-06	1.836E-07	4.252E-08
3.00	1.079E-06	-1.5188E-07	1.256E-06	1.519E-07	1.767E-07

Table 6 is similar to Table 4, i.e. $\hat{\alpha} = 7.013888 \neq \delta_3/h$ was used with c=1 and $\hat{\lambda}=\lambda$, except that the k+1 step predictor is used.

Table 6. P_4EC_4; c=1; $\hat{\alpha} \neq \delta_3/h$

4-step Predictor/ 4-step Corrector $h\lambda = -2$ $\mu = -.2$

The exact past data:

y(0.400)	=	0.1832E-01
y(0.600)	=	0.2479E-02
y(0.800)	=	0.3355E-03
y(1.000)	=	0.4540E-04

x	Predictor Error	Solution	$\lvert\Delta y^{(1)}\rvert$	Global Error	Local Error
1.20	1.024E-02	2.1901E-03	1.242E-02	2.184E-03	2.184E-03
1.40	7.055E-03	6.0026E-04	6.751E-03	5.994E-04	3.039E-04
1.60	1.097E-02	1.7188E-03	1.261E-02	1.719E-03	1.638E-03
1.80	1.176E-02	-8.3690E-04	1.283E-02	8.369E-04	1.070E-03
2.00	1.334E-02	1.1595E-03	1.461E-02	1.159E-03	1.273E-03
2.20	1.578E-02	-1.5256E-03	1.746E-02	1.526E-03	1.683E-03
2.40	1.792E-02	1.5606E-03	1.969E-02	1.561E-03	1.767E-03
2.60	2.066E-02	-1.8712E-03	2.274E-02	1.871E-03	2.082E-03
2.80	2.386E-02	2.1728E-03	2.628E-02	2.173E-03	2.426E-03
3.00	2.739E-02	-2.4552E-03	3.014E-02	2.455E-03	2.749E-03

There is a lack of stability (revealed by growth in the global error) due to the incorrect coefficient $\hat{\alpha}$ being used.

The choice of method parameters c=1 and $\hat{\alpha} = \delta_k/h$ specifies the k+1 step BDF since the exact Newton matrix solves a linear problem in one iteration no matter what value is used as a predictor (Eqn. 2-17). This is the reason the corrected solution in Tables 3 and 5 were identical. To see this, examine the equation

(3-10) $$f(y_n^{bdf}) \overset{?}{=} p' + \frac{\delta_k}{h} (y_n^{bdf} - p)$$

Labelling $y_n^{bdf} = y_n$, we have

$$f(y_n) \overset{?}{=} \frac{1}{h} \left[\sum_{r=1}^{k-1} \delta_r \nabla^r y_{n-1} + \delta_k \left\{ y_n - \sum_{r=0}^{k-1} \nabla^r y_{n-1} \right\} \right]$$

$$= \frac{1}{h}\left[\sum_{r=1}^{k-1} \delta_r \nabla^r y_{n-1} + \delta_k \nabla^k y_n\right]$$

$$= \frac{1}{h}\left[\sum_{r=1}^{k-1} \delta_r \left[\nabla^r y_n - \nabla^{r+1} y_n\right] + \delta_k \nabla^k y_n\right]$$

or

(3-11) $$f(y_n) = \frac{1}{h}\sum_{r=1}^{k} \delta_r^* \nabla^r y_n$$

since $\delta_r^* = \delta_r - \delta_{r-1}$. This is the k+1 step BDF in backward difference form (Eqn. 2-3) and so

(3-12) $$r(y_n^{bdf}) = 0$$

If one examines $\tilde{r}(y_n^{bdf})$, a manipulation similar to the above shows

$$f(y_n) \overset{?}{=} \frac{1}{h}\left[\sum_{r=1}^{k} \delta_r \nabla^r y_{n-1} + \delta_k \left\{y_n - \sum_{r=0}^{k} \nabla^r y_{n-1}\right\}\right]$$

$$= \frac{1}{h}\left[\sum_{r=1}^{k} \delta_r \nabla^r y_{n-1} + \delta_k \nabla^{k+1} y_n\right]$$

$$= \frac{1}{h}\sum_{r=1}^{k} \delta_r^* \nabla^r y_n$$

The residual of the BDF, g(y) (Eqn 2-16), and that of SIBDF are related by

$$\frac{h}{\delta_k} r(y) = h\,\beta_0\, r(y) = g(y)$$

It was chosen to write the residual, r(y), with f isolated so that analysis of the case $h \to \infty$ would easily eliminate the $\frac{1}{h}$ term. Also, when the correction matrix is formed, the Jacobian, J, would be isolated in anticipation of implementing a code based on this method. This allows the savings of the n^2 cost to form a matrix $h\beta_0 J$ from J and facilitates the design goal of re-using old Jacobians, if space is provided, when forming a new correction matrix because of changes in the order or step size of the formula used.

If one were to consider this explicit correction scheme to be an iterative method, i.e. i=0,1,..., the convergence would be governed by the eigenvalues of

$$I - \hat{G}^{-1}G$$

which we will soon label B. In the case of the model system, $y' = Jy$, fixing $\hat{c} = 1$ and $\alpha = \delta_k/h$, this is the same convergence matrix derived in (Eqn. 2-16) for the BDF.

$$I - \hat{G}^{-1}G = I - \frac{1}{h\beta_0}\left(\hat{J} - \frac{1}{h\beta_0} I\right)^{-1}(h\beta_0)\left(J - \frac{1}{h\beta_0} I\right) \tag{3-13}$$

$$= I - (h\beta_0\hat{J} - I)^{-1}(h\beta_0 J - I)$$

$$= I - \hat{P}^{-1}P$$

assuming both methods are approximating the Jacobian by $\hat{J}$.

Experimentation [52] has convinced the author that the parameter c can be used successfully to reduce matrix error when $\hat{\alpha} \neq \frac{\delta_k}{h}$ due to changes in the step number, k, or step size, h. As an example of this, examine the results of Table 6, where $\hat{\alpha}$ error caused the P_4EC_4 computation to be mildly unstable. This computation was repeated using a value of c = 1.189 which is the result of the parameter prediction algorithm to be presented in Chapter 6.a.

Table 7. P_4EC_4; $c \neq 1$; $\hat{\alpha} \neq \delta_3/h$

4-step Predictor/ 4-step Corrector $h\lambda = -2$ $\mu = -.2$

The computed value for c is 1.189 giving a μ value of -0.009

The exact past data:

y(0.400)	=	0.1832E-01
y(0.600)	=	0.2479E-02
y(0.800)	=	0.3355E-03
y(1.000)	=	0.4540E-04

x	Predictor Error	Solution	$\lvert\Delta y^{(1)}\rvert$	Global Error	Local Error
1.20	1.024E-02	2.1356E-04	1.045E-02	2.074E-04	2.074E-04
1.40	5.840E-04	1.8518E-04	7.403E-04	1.843E-04	1.563E-04
1.60	7.195E-04	7.2187E-05	7.667E-04	7.207E-05	4.713E-05
1.80	2.328E-05	2.7485E-06	1.626E-05	2.733E-06	7.021E-06
2.00	1.047E-04	-1.1035E-05	1.161E-04	1.104E-05	1.141E-05
2.20	4.443E-05	-3.8546E-06	4.679E-05	3.855E-06	2.361E-06
2.40	9.878E-06	1.6486E-06	1.205E-05	1.649E-06	2.170E-06
2.60	1.739E-05	2.0111E-06	1.918E-05	2.011E-06	1.788E-06
2.80	6.503E-06	6.5541E-07	6.886E-06	6.554E-07	3.832E-07
3.00	9.155E-07	-1.4094E-07	1.145E-06	1.409E-07	2.296E-07

The results from Table 5 were also repeated using c=1.1899 (the same matrix, $\hat{G}$, and incorrect coefficient, $\hat{\alpha}$, are involved) yielding a computation slightly more accurate.

Table 8. P_3EC_4; $c \neq 1$; $\hat{\alpha} \neq \delta_3/h$

3-step Predictor/ 4-step Corrector $h\lambda = -2$ $\mu = -.2$

The computed value for c is 1.189 giving a μ value of -0.009

x	Predictor Error	Solution	$\lvert\Delta y^{(1)}\rvert$	Global Error	Local Error
1.20	1.602E-03	1.0632E-04	1.502E-03	1.002E-04	1.002E-04
1.40	5.038E-04	9.0778E-05	4.275E-04	8.995E-05	7.639E-05
1.60	1.352E-05	3.3701E-05	3.493E-05	3.359E-05	2.142E-05
1.80	6.947E-05	6.8665E-07	6.560E-05	6.714E-07	3.874E-06
2.00	8.356E-06	-4.7243E-06	3.539E-06	4.726E-06	4.817E-06
2.20	1.811E-05	-1.2030E-06	1.867E-05	1.203E-06	5.637E-07
2.40	1.141E-05	8.7370E-07	1.038E-05	8.737E-07	1.037E-06
2.60	1.388E-06	7.3681E-07	7.691E-07	7.368E-07	6.186E-07
2.80	1.713E-06	1.4593E-07	1.760E-06	1.459E-07	4.622E-08
3.00	9.187E-07	-9.0882E-08	8.080E-07	9.088E-08	1.106E-07

We develop local truncation error for the SIBDF in Chapter 5. To accompany the previous examples on for a stable problem, $y'=-10y$, which exercises primarily the stability properties of an method, we present the following examples where a forcing function is introduced to the stable problem. The following examples highlight the interplay of stability and accuracy with respect to the choice of a P_kEC_{k+1} SIBDF or a $P_{k+1}EC_{k+1}$ SIBDF. We will see in Chapter 5 (Eqn 5-3) how the local truncation error of the SIBDF involves the parameter μ and the predictor formula used.

Consider the simple, scalar problem as in the previous examples, but with a forcing function present.

$$y' = \lambda(y-\cos(x)) - \sin(x)$$

$$y(a) = y_a$$

The solution is

$$y(x) = (y_a - \cos(a))\, e^{\lambda(x-a)} + \cos(x)$$

If we use as initial condition, y(1.0) = 1.1*cos(1.0), the solution will still be dominated by the second term for accuracy requirements, though the Jacobian, λ, will always limit stability.

Assume the correct ODE Jacobian value is used, $\hat{\lambda} = \lambda$. Let $\lambda = -10$. Fix $h = .2$ so that $h\lambda = -2$. We present examples similar to those in Tables 3 through 8 above, which show the effect of relative matrix error (μ=0 and μ=-.2) for the P_3EC_4 SIBDF and the P_4EC_4 SIBDF.

For a base case, we have the results for μ=0, i.e. the BDF solution is returned by either P_3EC_4 or P_4EC_4 since the Newton method is exact in one correction for linear problems (Eqn 2-17). The correction coefficient is $\hat{\alpha} = \delta_3/h = 10.4167$.

Table 9. P_3EC_4 Forced Scalar Problem; c=1; $\hat{\alpha}$=10.41666

3-step Predictor/ 4-step Corrector $h\lambda = -2$ $\mu = 0$

the exact past data:

y(0.600) = 0.3775E+01

y(0.800) = 0.1096E+01

y(1.000) = 0.5943E+00

x	Predictor Error	Solution	$\lvert\Delta y^{(1)}\rvert$	Global Error	Local Error
1.20	1.901E+00	5.0500E-01	1.765E+00	1.353E-01	1.353E-01
1.40	6.387E-01	2.9521E-01	5.327E-01	1.243E-01	1.059E-01
1.60	2.277E-02	1.7724E-02	5.274E-02	4.679E-02	2.997E-02
1.80	1.066E-01	-2.2708E-01	1.004E-01	9.960E-05	6.233E-03
2.00	2.308E-02	-4.2354E-01	1.567E-02	7.400E-03	7.414E-03
2.20	1.785E-02	-5.9030E-01	1.864E-02	1.795E-03	7.935E-04
2.40	1.030E-02	-7.3595E-01	8.613E-03	1.443E-03	1.686E-03
2.60	3.817E-03	-8.5577E-01	4.736E-03	1.114E-03	9.191E-04
2.80	7.695E-03	-9.4216E-01	7.608E-03	6.467E-05	8.613E-05
3.00	5.117E-03	-9.9035E-01	4.752E-03	3.564E-04	3.651E-04

Although the corrected solution is the same for both methods, it is valuable to examine the error of the higher order predictor. Compare the predictor error below for P_4EC_4 with that obtained in Table 9.

Table 10. P_4EC_4 Forced Scalar Problem; c=1; $\hat{\alpha}$=10.41666

4-step Predictor/ 4-step Corrector $h\lambda$ = -2 μ = 0

the exact past data:

y(0.400)	=	0.2272E+02
y(0.600)	=	0.3775E+01
y(0.800)	=	0.1096E+01
y(1.000)	=	0.5943E+00

x	Predictor Error	Solution	$\lvert\Delta y^{(1)}\rvert$	Global Error	Local Error
1.20	1.219E+01	5.0500E-01	1.232E+01	1.353E-01	1.353E-01
1.40	1.127E+00	2.9521E-01	1.233E+00	1.243E-01	1.059E-01
1.60	5.555E-01	1.7724E-02	5.855E-01	4.679E-02	2.997E-02
1.80	5.388E-02	-2.2708E-01	4.764E-02	9.960E-05	6.233E-03
2.00	7.731E-02	-4.2354E-01	8.472E-02	7.400E-03	7.414E-03
2.20	3.351E-02	-5.9030E-01	3.431E-02	1.795E-03	7.935E-04
2.40	8.341E-03	-7.3595E-01	1.003E-02	1.443E-03	1.686E-03
2.60	1.243E-02	-8.5577E-01	1.335E-02	1.114E-03	9.191E-04
2.80	2.959E-03	-9.4216E-01	2.873E-03	6.467E-05	8.613E-05
3.00	2.491E-03	-9.9035E-01	2.857E-03	3.564E-04	3.651E-04

We now explore the effect of error in the correction matrix due to error in the coefficient, $\hat{\alpha}$. Both are constructed to have μ = -.2. Since we assumed $\hat{\lambda}=\lambda$, we can compute that $\hat{\alpha}$ = 7.01388.

Table 11. P_3EC_4 Forced Scalar Problem; c=1; $\hat{\alpha}$=7.01388

3-step Predictor/ 4-step Corrector $h\lambda$ = -2 μ = -.2

the exact past data:

y(0.600)	=	0.3775E+01
y(0.800)	=	0.1096E+01
y(1.000)	=	0.5943E+00

x	Predictor Error	Solution	$\lvert\Delta y^{(1)}\rvert$	Global Error	Local Error
1.20	1.901E+00	1.5190E-01	2.119E+00	2.178E-01	2.178E-01
1.40	3.728E-01	6.8924E-02	3.003E-01	1.020E-01	7.256E-02
1.60	3.883E-01	-9.4525E-02	4.399E-01	6.546E-02	5.165E-02
1.80	1.024E-01	-2.4082E-01	9.763E-02	1.363E-02	4.773E-03
2.00	4.804E-02	-4.1806E-01	4.812E-02	1.920E-03	7.477E-05
2.20	3.751E-02	-5.8322E-01	4.305E-02	5.282E-03	5.541E-03
2.40	3.976E-04	-7.3325E-01	3.036E-03	4.148E-03	3.433E-03
2.60	1.182E-02	-8.5351E-01	1.463E-02	3.378E-03	2.816E-03
2.80	2.251E-03	-9.4019E-01	3.823E-03	2.030E-03	1.573E-03
3.00	3.572E-03	-9.8870E-01	4.594E-03	1.297E-03	1.022E-03

In Table 11, the computation remains stable even in the presence of relative matrix error of 20%. When the same order predictor and corrector are used, Table 12 reveals dramatic loss of stability.

Table 12. P_4EC_4 Forced Scalar Problem; $c=1$; $\hat{\alpha}=7.01388$

4-step Predictor/ 4-step Corrector $h\lambda = -2$ $\mu = -.2$

the exact past data:

y(0.400)	=	0.2272E+02
y(0.600)	=	0.3775E+01
y(0.800)	=	0.1096E+01
y(1.000)	=	0.5943E+00

x	Predictor Error	Solution	$\lvert\Delta y^{(1)}\rvert$	Global Error	Local Error
1.20	1.219E+01	2.9691E+00	1.478E+01	2.599E+00	2.599E+00
1.40	8.396E+00	8.8460E-01	8.034E+00	7.136E-01	3.618E-01
1.60	1.306E+01	2.0167E+00	1.501E+01	2.046E+00	1.949E+00
1.80	1.399E+01	-1.2231E+00	1.526E+01	9.959E-01	1.273E+00
2.00	1.587E+01	9.6387E-01	1.739E+01	1.380E+00	1.515E+00
2.20	1.878E+01	-2.4044E+00	2.078E+01	1.816E+00	2.003E+00
2.40	2.133E+01	1.1198E+00	2.343E+01	1.857E+00	2.103E+00
2.60	2.459E+01	-3.0843E+00	2.707E+01	2.227E+00	2.479E+00
2.80	2.839E+01	1.6433E+00	3.128E+01	2.585E+00	2.887E+00
3.00	3.260E+01	-3.9128E+00	3.588E+01	2.923E+00	3.273E+00

We can again verify the benefit of choosing the scalar $c\neq1$. The results for the P_3EC_4 computation in Table 13 demonstrate a modest improvement in an already stable computation. The scalar c produces a dramatic improvement in the P_4EC_4 computation displayed in Table 14.

Table 13. P_3EC_4 Forced Scalar Problem with $c \neq 1$; $\hat{\alpha}=7.01388$

3-step Predictor/ 4-step Corrector $h\lambda$ = -2 μ = -.2

The computed value for c is 1.189 giving a mu value of -0.009

the exact past data:

y(0.600)	=	0.3775E+01
y(0.800)	=	0.1096E+01
y(1.000)	=	0.5943E+00

x	Predictor Error	Solution	$\lvert\Delta y^{(1)}\rvert$	Global Error	Local Error
1.20	1.901E+00	4.8901E-01	1.781E+00	1.193E-01	1.193E-01
1.40	5.929E-01	2.7820E-01	5.018E-01	1.072E-01	9.109E-02
1.60	2.355E-02	1.1104E-02	4.921E-02	4.017E-02	2.566E-02
1.80	9.052E-02	-2.2622E-01	8.604E-02	9.600E-04	4.476E-03
2.00	1.777E-02	-4.2165E-01	1.214E-02	5.507E-03	5.637E-03
2.20	1.407E-02	-5.8986E-01	1.468E-02	1.358E-03	6.125E-04
2.40	6.733E-03	-7.3632E-01	5.480E-03	1.070E-03	1.254E-03
2.60	4.303E-03	-8.5603E-01	5.020E-03	8.612E-04	7.164E-04
2.80	6.864E-03	-9.4211E-01	6.860E-03	1.120E-04	4.530E-06
3.00	4.595E-03	-9.9021E-01	4.366E-03	2.135E-04	2.286E-04

Table 14. P_4EC_4 Forced Scalar Problem with $c \neq 1$; $\hat{\alpha}=7.01388$

4-step Predictor/ 4-step Corrector $h\lambda$ = -2 μ = -.2

The computed value for c is 1.189 giving a μ value of -0.009

the exact past data:

y(0.400)	=	0.2272E+02
y(0.600)	=	0.3775E+01
y(0.800)	=	0.1096E+01
y(1.000)	=	0.5943E+00

x	Predictor Error	Solution	$\lvert\Delta y^{(1)}\rvert$	Global Error	Local Error
1.20	1.219E+01	6.1658E-01	1.243E+01	2.469E-01	2.469E-01
1.40	6.956E-01	3.9045E-01	8.817E-01	2.195E-01	1.861E-01
1.60	8.569E-01	5.6791E-02	9.131E-01	8.586E-02	5.615E-02
1.80	2.800E-02	-2.2390E-01	1.966E-02	3.286E-03	8.334E-03
2.00	1.246E-01	-4.2930E-01	1.382E-01	1.315E-02	1.360E-02
2.20	5.319E-02	-5.9315E-01	5.606E-02	4.650E-03	2.870E-03
2.40	1.118E-02	-7.3553E-01	1.367E-02	1.859E-03	2.489E-03
2.60	1.989E-02	-8.5464E-01	2.189E-02	2.254E-03	2.002E-03
2.80	6.721E-03	-9.4161E-01	7.025E-03	6.081E-04	3.031E-04
3.00	2.277E-03	-9.9036E-01	2.724E-03	3.651E-04	4.474E-04

Stability along with better accuracy is obtained by using a suitable value of the parameter c to reduce μ.

An explanation of these examples will be postponed until we discuss stability for the SIBDF in the next chapter. We see in these examples that with a given order corrector formula, the predictor of order one lower produces a more stable algorithm in the presence of matrix error than does the predictor of the same order. Also with proper choice of the scalar parameter c we turn an unstable method into a stable one.

3.c. Variable Coefficient Implementation

When implementing a method, a facility to allow the step size to change comes into consideration. There are two major choices: constant coefficient formulas and variable coefficient formulas (see [20, p. 90-95]). The constant coefficient formula would use precisely the formulas presented above and the solution history would be at equally spaced back points. The Nordsieck representation for the solution history is the one most often seen in codes.

The variable coefficient formula uses past solution data that is variably spaced (reflecting past changes in step size). One way to accomplish this effectively is by using the Modified Divided Differences (MDDs) [34]. The implementation chosen for the code STRUT uses the MDDs and this distinguishes the code from other stiff solvers. Other techniques to work with variable spaced past data are possible. EPISODE [2,3] uses the Nordsieck history array to represent variable spaced past data to represent the BDF. DASSL [40] employs the Fixed Leading Coefficient technique [28] with the MDDs to solve Differential-Algebraic Equations with a BDF based method.

3.c.i. Modified Divided Difference Representation

The Modified Divided Difference representation due to Krogh [34] allows efficient implementation of variable order, variable step size methods with error estimators. The MDDs avoid the lessened stability properties of methods which use constant coefficient, fixed step size formulas and interpolate in the past data when the step size is changed. Fixed step size formulas were used in the code DIFSUB and its descendants GEAR and LSODE. An interesting plot demonstrating the poor behavior of high order divided differences for approximating a function when the spacing is cut by $\frac{1}{2}$ is given by Krogh [33]. A discussion of the implications this of property for high order Adams methods is given by Hall and Watt [20, p. 101-102]. In this case, a reduction of step size will not result in a reduction in error and a code would be forced to drop its order, and possibly restart, to get any error reduction.

The derivation of the MDD representation for the Adams formulas and BDFs is presented by Krogh [34] for first and higher order equations and is directly applicable to systems of equations. The reader is also referred to Shampine and Gordon [48] for a lucid explanation of the derivation of the Adams representation for first order equations. Krogh's Adams and Backward Differentiation Formulas were in terms of a fixed number of corrections, namely two. This was, then, an algorithm for the SIBDF. Krogh recommended using ratios of successive corrections to judge the need for new correction matrices. We present an alternative scheme based on eigenvalue estimates in Section 6.c.

Since the MDD representation stores solution information at variable spaced past points, coefficients for the k^{th} order method must be recomputed when the step size changes and for k-1 steps after a single change in step size. The coefficients are labelled d_k, corresponding to the fixed step coefficient, δ_k, and we have at the n^{th} step, x_n,

$$d_k = d_{k-1} + \frac{h_n}{h_n + \ldots + h_{n-k+1}}$$

Starting with $d_0 = 0$, we have $d_1 = 1$, $d_2 = 1 + \frac{h_n}{h_n + h_{n-1}}$, and
$$d_3 = 1 + \frac{h_n}{h_n + h_{n-1}} + \frac{h_n}{h_n + h_{n-1} + h_{n-2}}$$

Using the algorithms developed by Krogh, the cost is only linear in k. Obviously, the cost of calculating coefficients is not important for the BDF, except in how it affects relative matrix error.. Since this can have significant impact on the algorithm, we examine the matrix $B = I - \hat{G}^{-1}G$. Consider the $P(k)\left[EC(k+1)\right]^m$ method. The $k-1^{st}$ order predictor is a sum of d_j's and MDDs. The residual at x_n using the variable step size coefficient, d_k, is

$$r_n^{(i)} = f(y_n^{(i)}) - p_n' - \frac{d_k}{h_n}(y_n^{(i)} - p_n) \tag{3-14}$$

which defines $G = \partial r/\partial y = \partial f/\partial y - \frac{d_k}{h_n} I$. But the matrix used to compute corrections is $\hat{G} = c\,(\hat{J} - \hat{\alpha} I)$. Suppose the step size had been changed the step before, at x_{n-1}. For simplicity, assume that $\hat{G}$ was formed then, the order and step size have not changed and $\hat{J}$ is the correct ODE Jacobian. If $\hat{\alpha}$ was chosen to match the method coefficient of the residual at x_{n-1}, then the coefficient in the residual at x_n is slightly different. If the matrix $\hat{G}$ is not refactored, then error will have been introduced due to a previous step size change.

3.c.ii. Fixed Leading Coefficient

We mention the idea of Fixed Leading Coefficient, due to Jackson and Sacks-Davis [28], because

a) it addresses the problem of correction matrix error due to a variable step size implementation,

b) it was used in the implementation by Petzold of DASSL [40] of the BDF-based method for Differential-Algebraic Differential equations and

c) it can be modelled by our stability model of Chapter 4.b.

The MDDs are used to represent the past solution history. A variable coefficient formula is used to form the predictor. Then, given that the correction matrix currently available is $\hat{G} = \hat{J} - \hat{\alpha} I$, the residual used to compute corrections is

$$r_n^{(i)} = f(y_n^{(i)}) - p_n' - \hat{\alpha}\,(y_n^{(i)} - p_n)$$

This defines a new corrector formula of the same order as predictor, though the coefficient of the error term for the corrector may be quite large. Any error in the correction matrix can only be due to $\hat{J}$ differing from $\partial f/\partial y$, not to changes in the variable step coefficient.

3.d. Implications

It is expensive to reform the correction matrix and we seek a device that will help avoid this. For a variable coefficient implementation, great care must be taken. The stability analysis of the next chapter will provide a firm basis for deciding how much relative error can be tolerated by a stable method using a fixed number of corrections. This will make it feasible to use the higher order corrector (Eqn 3-5). This is not possible with the Fixed Leading Coefficient technique. The scalar c modification of the correction matrix will be relied upon to reduce error in the correction matrix due to small changes in the d_k's after a change in step size.

Chapter 4. Models of Stability for the SIBDF

We investigate the stability properties of the methods presented in Chapter 3, using two possible predictors and a fixed number of corrections computed with a correction matrix that is in error, returning to the assumption of fixed h and k. We then determine the amount of relative error that can be tolerated by a stable method. This is convenient because it eliminates the need to judge convergence. It also removes concerns over what tactics to employ in the event of convergence failure.

Klopfenstein [29] first investigated a class of explicit $P(EC)^m$ algorithms involving a predictor (Eqn 3-1) followed by m applications of the Newton-like process (Eqn 3-4) applied to the model system, $y' = Jy$. Krogh and Stewart [35] presented a simplified derivation which postponed diagonalizing the matrices involved in the algebraic manipulations so that more could be understood about the relationship between predictor and corrector. We will see this difference equation is more involved than that for the BDF. Nevertheless, statements relating predictor - corrector order to relative matrix error can be made.

The derivations in [29] and [35] focussed on the order of the predictor. In this dissertation we select the order of the corrector, and then study the effects that two different order predictors have on the stability of the resulting formula. First we examine the general nonlinear equation to make clear when the simplifying assumption of the linear model problem is needed.

4.a. The Nonlinear Problem

For the general nonlinear problem,

$$y' = f(x,y)$$

the SIBDF will use the correction matrix $\hat{G}$ (Eqn. 3-6) which is associated with either

a) the k step predicted solution p (Eqn 3-1) and derivative p' (Eqn 3-2) and k+1 step residual r (Eqn 3-5) or

b) the k+1 step predicted solution $\tilde{p}$ (Eqn 3-7) and derivative $\tilde{p}'$ (Eqn 3-8) and k+1 step residual $\tilde{r}$ (Eqn 3-9).

We will consider case a) first. Expressing the $(i+1)^{st}$ corrected value at the n^{th} time step in terms of previous information and suppressing the subscript, we have

$$r^{(i)} = -\frac{\delta_k}{h}(y^{(i)} - p) + f(y^{(i)}) - p' + f(p) - f(p)$$

Let

$$G^* = J^* - \frac{\delta_k}{h} I,$$

with J^* defined as $J^* = [\nabla f^*]^T$ by

$$J^* = \begin{bmatrix} \nabla f_1(\zeta_1)^t \\ \vdots \\ \nabla f_j(\zeta_j)^t \\ \vdots \\ \nabla f_n(\zeta_n)^t \end{bmatrix}$$

where ζ_j is a point somewhere on the line between $y^{(i)}$ and p. Each row of J^* results from the application of a Mean Value Theorem to the j^{th} component of f

$$f(y^{(i)}) = f(p) + J^*(y^{(i)}-p)).$$

Recognizing $r^{(0)} = f(p) - p'$, we have

$$r^{(i)} = G^*(y^{(i)}-p) + r^{(0)} = G^*(y^{(i)}-p) - \hat{G}(y^{(1)}-p).$$

Also,

$$r^{(i)} = -\hat{G}\ (y^{(i+1)} - y^{(i)})$$

As in Section 2.c, we assume $\hat{G}$ invertibe. We now have

$$\text{(4-1)} \qquad y^{(i+1)} = -\hat{G}^{-1}G^*\ (y^{(i)} - p) + y^{(i)} - p + y^{(1)}$$

$$= -\hat{G}^{-1}(G^* - \hat{G})\ (y^{(i)} - p) + y^{(1)}$$

This shows that no change occurs after one correction if $\hat{G} = G^*$, i.e. the BDF (Eqn. 2-17). To obtain the vector difference equation, we must write $y^{(i+1)}$ in terms of past data. We trace the term $y^{(i)}-p$ back to $y^{(1)}-p = -\hat{G}^{-1}r^{(0)} = -\hat{G}^{-1}(f(p)-p')$ which uses only differences of past time steps via p and p'. The general nonlinear case will have G^* changing on each correction within a time step. To proceed further, we assume the linear, constant coefficient model.

4.b. The Vector Generalized Difference Equation (VGDE) for y'=Jy

We now concentrate on $y' = Jy$ and remove the asterisk from the matrix G to indicate that it does not depend on which correction is being computed. The SIBDF solution results from m corrections and is denoted y^{mc}. We repeat and expand the derivation from [35] for definiteness. Let

$$\Delta = G - \hat{G}.$$

Rewriting Eqn 4-1,

$$y^{mc} = y^{(m)} = -\hat{G}^{-1}\Delta\ (y^{(m-1)}-p) + y^{(1)} + p - p$$

$$= -\hat{G}^{-1}\Delta\left[-\hat{G}^{-1}\Delta\ (y^{(m-2)}-p) + y^{(1)}-p\right] + (y^{(1)}-p) + p$$

$$= p + \sum_{r=0}^{m-1} (-\hat{G}^{-1}\Delta)^r\ (y^{(1)}-p)$$

$$= p + \left[\sum_{r=0}^{m-1} (-\hat{G}^{-1}\Delta)^r\right] (-\hat{G}^{-1}\ r^{(0)})$$

We call B the relative error matrix

$$B = -\hat{G}^{-1}\Delta = -\hat{G}^{-1}(G-\hat{G}) = I - \hat{G}^{-1}G \tag{4-2}$$

Assuming I−B is invertible, the sum above can be simplified using $\sum_{r=0}^{m-1} B^r = (I - B^m)\ (I - B)^{-1}$. We note that $I - B = I + \hat{G}^{-1}\Delta$. Also $(I-B)^{-1} = (-\hat{G}^{-1}r^{(0)}) = -G^{-1}r^{(0)}$ which comes from

$$(I + \hat{G}^{-1}\Delta)^{-1}\ (-\hat{G}^{-1}r^{(0)}) = (I + \hat{G}^{-1}G - I)^{-1}\ (-\hat{G}^{-1}r^{(0)}) = -G^{-1}r^{(0)}$$

We write

$$y^{mc} = \left[I - B^m\right] G^{-1} r^{(0)} + p$$

$$= \left[I - B^m\right]\left[J - \frac{\delta_k}{h} I\right]^{-1}\left[p' - Jp + \frac{\delta_k}{h} p - \frac{\delta_k}{h} p\right] + p$$

$$= \left[I - B^m\right] \left[-p + G^{-1}(p' - \frac{\delta_k}{h} p)\right] + p$$

Thus

$$y^{mc} = B^m p + \left[I - B^m\right] G^{-1} (p' - \frac{\delta_k}{h} p) \tag{4-3}$$

Using the difference equations for p (Eqn.3-1) and p′ (Eqn. 3-2) and δ_0=0, the sums can be collapsed. If we label the matrix

$$(I - B^m) G^{-1} = C_m, \tag{4-4}$$

the <u>vector generalized difference equation</u> (<u>VGDE</u>) then is

$$y_n = y^{mc} = \sum_{r=0}^{k-1} \left[B^m + C_m (\frac{\delta_r}{h} - \frac{\delta_k}{h})\right] \nabla^r y_{n-1} \tag{4-5}$$

The VGDE (Eqn 4-5) will not be diagonalizable even when solving $y'=Jy$ unless there is some special relationship between J and $\hat{J}$. For example, if one knew J were diagonal, one could force the approximation $\hat{J}$ also to be diagonal and so be able to diagonalize Eqn 4-5.

We have a <u>k-step vector difference equation</u>. In Section 2.e, we treated the k-step scalar difference equation. We assumed above Eqn.

4-1 that the iteration matrix $\hat{G}$ could be inverted. This assumption allows the simpler techniques for <u>monic matrix polynomials</u> to be applicable [14, pp. 18-19]. Without the assumption that the matrix coefficient of y_n can be inverted, the nonmonic matrix polynomial must be dealt with [14, Thm. S1.8 and Prop. S1.9 where a basis for the solution space is constructed]. This is of concern when solving differential equations coupled with algebraic equations and is beyond the scope of this dissertation. See the article by Petzold [40] for an introduction to differential-algebraic equations.

We assume h, k and the matrices B and C_m involved in Eqn. 4-5 are constant. For readability we suppress the fixed value of the superscript m to yield the vector difference equation

$$\begin{bmatrix} y_n \\ y_{n-1} \\ \vdots \\ y_{n-k+1} \end{bmatrix} = \begin{bmatrix} B-\frac{\delta_k}{h}C & B+\frac{1}{h}(\delta_1-\delta_k)C & B+\frac{1}{h}(\delta_2-\delta_k)C & B+\frac{1}{h}(\delta_3-\delta_k)C & \cdots \\ I & 0 & 0 & \cdots & \\ I & -I & 0 & \cdots & \\ I & -2I & I & \cdots & \\ & \vdots & & & \end{bmatrix} \begin{bmatrix} y_{n-1} \\ \nabla y_{n-1} \\ \\ \nabla^{k-1} y_{n-1} \end{bmatrix}$$

with the n by n identity matrix I and n by n matrix of zeros, 0. The vector backward differences of y_{n-1} are expressed in ordinate form using Pascal's triangle. Eqn. 4-5 takes the form

$$(4\text{-}6) \qquad \begin{bmatrix} y_n \\ y_{n-1} \\ \vdots \\ y_{n-k+1} \end{bmatrix} = \begin{bmatrix} A_1 & A_2 & & \cdots & A_k \\ I & 0 & 0 & \cdots & 0 \\ 0 & I & 0 & \cdots & 0 \\ 0 & 0 & 0 & I & 0 \end{bmatrix} \begin{bmatrix} y_{n-1} \\ y_{n-2} \\ \vdots \\ y_{n-k} \end{bmatrix}$$

the n by n matrices A_i representing the coefficients of the ordinate form of the backward difference equation (Eqn 4-6). Following the development in [35], we define the kn vector

(4-7) $$Y_{n-1}^{mc} = \begin{bmatrix} y_{n-1} \\ y_{n-2} \\ \vdots \\ y_{n-k} \end{bmatrix}$$

and label the kn by kn matrix in Eqn 4-6, W. We have

(4-8) $$Y_n^{mc} = W\ Y_{n-1}^{mc} = W^2\ Y_{n-2}^{mc} = \ldots = W^n\ Y_0$$

Y_0 again represents the kn vector of starting errors for the k step method.

As in [35], there are two simplifications of some generality for which the VGDE can be diagonalized and stability analyzed easily. This is discussed in the next two subsections. In a third subsection the full equation is investigated in the scalar case.

4.c. Zero Matrix Error (BDF)

It is of value to develop here an alternative relationship between the SIBDF and the BDF. Rewriting Eqn 3-10 for the linear, constant coefficient problem, we have

$$J\ y_n^{bdf} = p' + \frac{\delta_k}{h}\ (y_n^{bdf} - p)$$

or

$$G\ y_n^{bdf} = p' - \frac{\delta_k}{h}\ p$$

Thus an alternative representation of the SIBDF from Eqn. 4-3 is

$$(4\text{-}9) \qquad y_n^{mc} = B^m p_n + (I-B^m)\, y_n^{bdf} = y_n^{bdf} + B^m \left[p_n - y_n^{bdf}\right]$$

This shows that the correction process for the SIBDF simply introduces powers of the relative error matrix, B, acting on the difference between the k step predictor and the BDF solution in order to bring the SIBDF nearer the BDF.

If one assumes B=0, equivalently $G=\hat{G}$, one has from Eqn 4-9 the equivalence of the SIBDF and the BDF when solving $y'=Jy$. Thus, the absolute stability for the VGDE under the assumption that $G=\hat{G}$ is the same as absolute stability of the BDF. Accordingly Figures 1, 2 and 3 also represent the absolute stability regions for the SIBDF when there is no matrix error. We now proceed to analyze cases that allow matrix error.

4.d. Asymptotic ($h\to\infty$) Absolute Stability (AAS).

Examining the difference equation (Eqn 4-5), we notice that the limit as $h\to\infty$ will cause the coefficient of C_m to vanish. In the super-matrix W (4-6), this causes the blocks of W to simplify, each A_i is a scalar multiple of B^m. The solution of a difference equation can be expressed in terms of the eigenvalues of the associated companion matrix. Since the eigenvalues of a matrix are continuous functions of the elements of that matrix [27, Thm. 4-2], the limit matrix resulting from W as $h\to\infty$ will have eigenvalues which are close to those of W when h is sufficiently large. Working with this simplified case, we need only assume B can be diagonalized in order to produce a solvable difference equation.

In solving stiff problems we are concerned with stability for large h. We have analytic tools that allow us to analyze the VGDE in the limit $h \to \infty$. Experimentation and the analsis in 4.e will guide us in determining how well these limit results pertain to the practical case of finite, but large h.

We proceed to solve the difference equation for the limit case. This case was originally studied by Klopfenstein [29] who defined <u>Asymptotic ($h \to \infty$) Absolute Stability (AAS)</u> in the following manner [29, p. 453, Def. 1]

> 'The algorithm is said to be asymptotically ($h \to \infty$) absolute stable in solving the system $y' = Jy$ if the sequence $\left\{y_n\right\}_{n=0}^{\infty}$ is uniformly bounded in norm for all sufficiently large $h \geq h_0 > 0$ when the eigenvalues of $I - J^{-1}\hat{J}$ lie within a region R containing a circle of radius $c > 0$ centered on the origin where c is independent of h.'

The characterization here (and in [35]) differs from that of Klopfenstein in using the error of the correction matrix relative to the approximate matrix, $I - \hat{G}^{-1}G$, in place of the error of the Jacobian relative to the true Jacobian, $I - J^{-1}\hat{J}$. Equivalent results are obtained, but the analysis is not done in the same technical manner since different assumptions are made. Klopfenstein assumed J was diagonalizable with eigenvalues entirely in the left half plane. We assume $B = I - \hat{G}^{-1}G$ is diagonalizable and make no assumption on the eigenvalues of J. Klopfenstein assumed that all error was in the ODE Jacobian. Taking the limit as $h \to \infty$, $\hat{G} \to \hat{J}$, $G \to J$ and the relative error matrix, B, becomes

$$B = I - \hat{G}^{-1}G = \hat{G}^{-1}(\hat{G}-G) = \hat{J}^{-1}(\hat{J}-J) = I - \hat{J}^{-1}J$$

In a code we need a simple way of describing the stability region. For this purpose we use the radius of the largest circle contained in the stability region. One reason we prefer error relative to the approximate matrix is that the regions that result are more nearly circular.

The definition of Asymptotic Absolute Stability requires a uniform bound on the solution of the difference equation. The limit case when $h=\infty$ is not of interest in itself but it is solvable and provides guidance. If all roots of the difference equation in the limit case are less than one, we are assured that solution growth is bounded. If any root has magnitude larger than one, then we can conclude that some solution will be unbounded. We require, then, all roots of the difference equation to be strictly less than one in modulus. By continuous dependence of eigenvalues on the elements of the matrix, we can conclude that for h sufficiently large, the solution to the VGDE (4-6) will be bounded.

From Eqn 4-5, the limiting difference equation depends only on the step number of the predictor and B.

(4-10) $$y_n = B^m \sum_{r=0}^{k-1} \nabla^r y_{n-1} = B^m p_n$$

If we had used the k+1 step predictor scheme (case b at the top of this section), the derivation of Eqn 4-3 would be unchanged. When the backward difference form for $\tilde{p}$ (Eqn 3-6), and $\tilde{p}'$ (Eqn 3-7) are substituted into Eqn 4-3, we have

(4-11) $$\tilde{y}_n = \tilde{y}^{mc} = \sum_{r=0}^{k} \left[B^m + C_m \left(\frac{\delta_r}{h} - \frac{\delta_k}{h}\right) \right] \nabla^r y_{n-1}$$

which in the limit of h=∞ gives

(4-12) $$\tilde{y}_n = B^m \sum_{r=0}^{k} \nabla^r y_n$$

As in Section 2.c on absolute stability for the BDF, the solution of a difference equation can be expressed in terms of the eigenvalues and eigenvectors of the associated companion matrix. For the BDF, these eigenvalues involve $h\lambda$, where λ is an eigenvalue of J. The eigenvalues of interest now are those of the relative error matrix B which we call μ.

$$\mu \in \sigma(B)$$

If the matrix B can be diagonalized, the same matrix will diagonalize Eqn 4-10. We begin with the standard substitution, $y_n = z^n$, in the diagonalized form of the difference equation (4-10). Using the difference identity (Eqn. 2-11), we obtain

(4-13) $$z^n = \mu^m \sum_{r=0}^{k-1} \nabla^r z^{n-1} = \mu^m \sum_{r=0}^{k-1} z^{n-1} \left[1 - \frac{1}{z}\right]^r$$

Collapsing the finite geometric sum and solving for μ, we have

(4-14) $$\mu^m = \frac{1}{1 - \left[1 - \frac{1}{z}\right]^k}$$

We will use the boundary locus technique (See Section 2.c) to delimit the <u>stability region in the μ plane</u> of eigenvalues of the relative error matrix, B. Allow z to take on values from the root unit circle, $z = e^{i\theta}$. The right hand side of 4-14 can then be evaluated and we rewrite this complex number in polar form,

$$\mu^m = \frac{1}{1 - [1 - e^{-i\theta}]^k} = \rho\, e^{i\omega}$$

The m roots are then given by

$$\mu_j = \rho^{1/m} e^{i(\omega/m + 2\pi j/m)} \qquad j=0,\ldots,m-1$$

Figures 4 and 5 present the AAS regions for the 3 step predictor and 4 step predictor with m=1, 2, 3, and 4 corrections. Since $\mu=0$ corresponds to z=0, which is within the root unit circle, the origin of the μ-plane is within the stability region. We therefore have stability in the μ-plane for values of μ in the interior of the closed curves. AAS with m=1 correction must have μ in the most interior region. AAS with m=2 must have μ^2 in this same region, equivalently, μ must be in the two correction region. The effect of additional corrections is to enlarge the region of stability. Notice the tear drop shaped region for m=1.

Figure 4. Asymptotic (h→∞) Absolute Stability
3 Step Predictor (Order 2)
Stability Limitation in the μ-plane

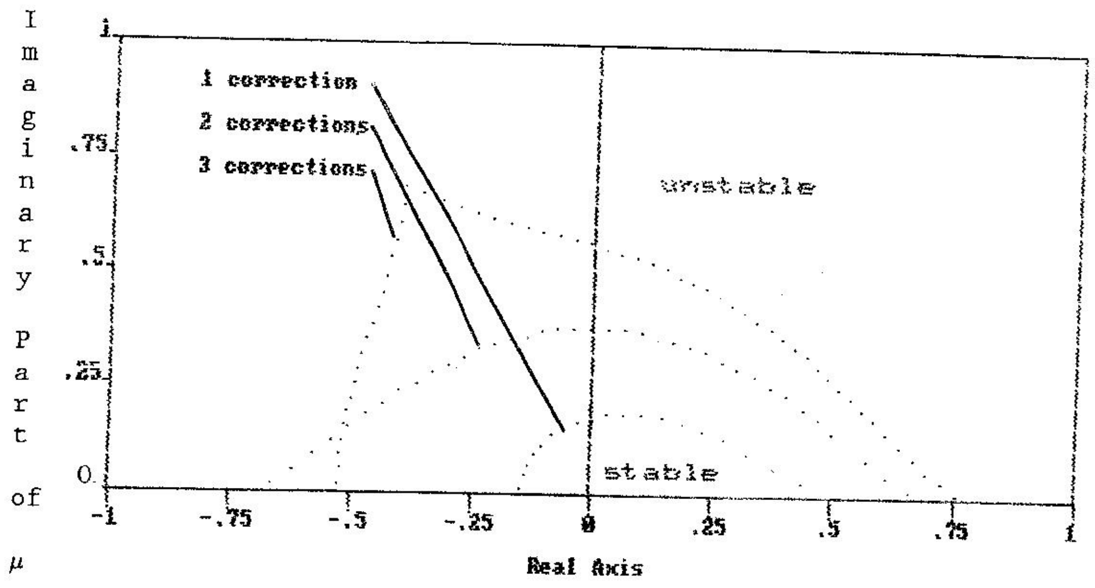
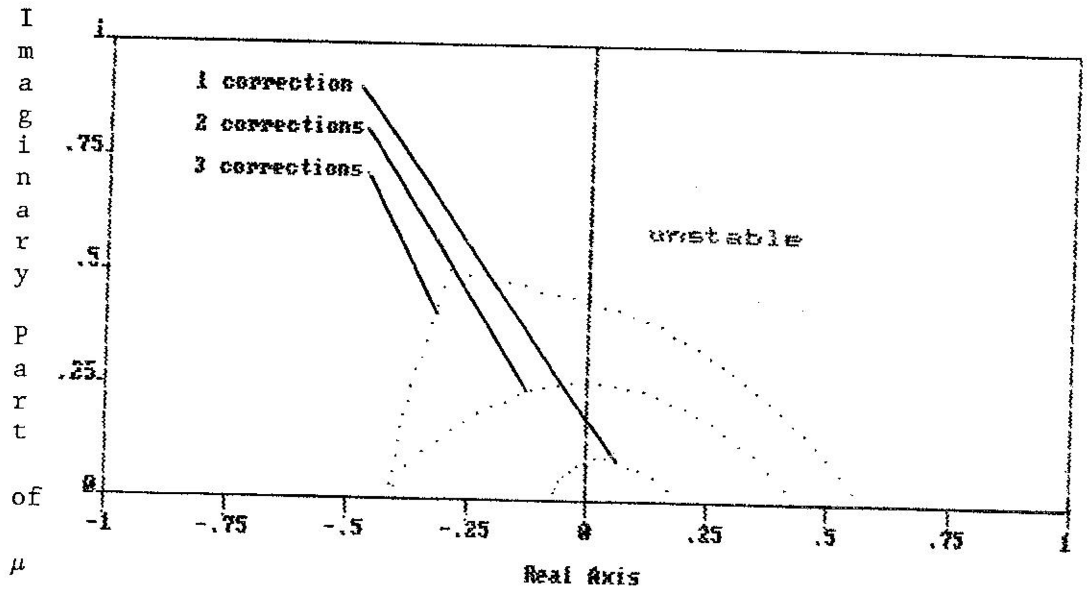


Figure 5. Asymptotic Absolute Stability
4 Step Predictor (Order 3)
Stability Limitation in the μ-plane



Additional plots of AAS regions are contained in Appendix B. This is most easily summarized by tabulating the radius of the largest circle contained entirely within the appropriate AAS region. Table 15 is repeated from [35] giving the minimum radius of AAS region when a kth order predictor is followed by m corrections.

Table 15. Minimum $|\mu|$ for μ on the Boundary of the Region of AAS for a $P(EC)^m$ Algorithm with Predictor of Order k

	Values of k						
m	0	1	2	3	4	5	6
1	1.0	0.333	0.143	0.067	0.032	0.016	0.008
2	1.0	0.577	0.378	0.258	0.179	0.126	0.088
3	1.0	0.693	0.523	0.405	0.318	0.251	0.199
4	1.0	0.760	0.615	0.508	0.424	0.355	0.298
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
∞	1.0	1.0	1.0	1.0	1.0	1.0	1.0

4.d.i The Choice of m=2

Examinination of Table 15 shows that the effect of more corrections with a chosen predictor is to allow more relative matrix error to be tolerated by a stable method. One sees a dramatic increase in relative matrix error that can be tolerated between m=1 and m=2 for all choices of predictor order. Larger values of m bring only a modest increase in tolerable error. This suggests that we choose m=2, $P(EC)^2$, as the value for an implementation of a SIBDF method. The experimental code STRUT [52] successfully used this strategy on both the stiff [12] and non-stiff test sets [26] It demonstrated that the theory developed for $h=\infty$ could be used in practice for finite, and even small h. See Appendix A for summary details.

4.d.ii The Choice of $P(k)[EC(k+1)]^2$ over $P(k+1)[EC(k+1)]^2$

The VGDE for the k+1 step corrector took two forms depending on whether the lower order predictor, P(k), (Eqn 4-5) or predictor of same order as corrector, P(k+1), (Eqn 4-9) was used. The analysis of the asymptotic ($h\rightarrow\infty$) case shows that stability depends on the stability of P(k) and P(k+1) and that P(k+1) tolerates less relative matrix error than P(k). This is confirmed by the computed examples in Chapter 3 where the P(3)EC(4) method clearly outperformed the P(4)EC(4) when error was introduced in the iteration matrix. We therefore choose to implement the $P(k)\left[EC(k+1)\right]^2$ method for the SIBDF.

4.e. The Scalar Problem (SGDE)

We extend the models presented in [35] in this section. The matrices B and C_m appearing in the generalized vector difference equation (Eqn 4-5) will not, in general, be simultaneously diagonalizable. We have examined two limiting cases that eliminate one of these matrices, namely, either B is zero or the scalar coefficient of C_m goes to zero. Either assumption yields a difference system easily solved.

Another way to glean information from the stability model is to examine the scalar case. This is possible for finite $h\lambda$, $\lambda\in$. The results will show that the AAS bounds derived in Section 4.c are somewhat conservative values. The analysis will show the important fact that the scalar SIBDF with finite, real $h\lambda$ will be stable for larger amounts of relative matrix error as h gets smaller. This strengthens the argument that using the scalars in Table 9 to judge acceptable matrix error will be adequate for the actual implementation case of finite h.

We focus now on the scalar problem

$$y' = \lambda y \quad \lambda \in$$

The results when $\lambda = \hat{\lambda}$, $\hat{\alpha} = \delta(k)/h$, c=1 constitute the case of the familiar BDF absolute stability already studied in the system case (Section 4.c). The stability plots in the $h\lambda$ plane were given in Figures 1, 2 and 3. Now we examine the effect of error in the matrix used to solve for the corrections.

We begin by examining the results of error in the coefficient $\hat{\alpha}$ to explain the behavior of the examples for a single correction presented in Tables 3 through 8 in Chapter 3. Let m = 1, $\lambda = \hat{\lambda}$, c = 1 and $\hat{\alpha} \neq \delta(k)/h$. This causes

$$B = I - \hat{G}^{-1}G = -\frac{-\delta_k/h + \hat{\alpha}}{\lambda - \hat{\alpha}} = \frac{\delta_k - h\hat{\alpha}}{h\lambda - h\hat{\alpha}}$$

and

$$C_1 = (I-B)\, G^{-1} = \hat{G}^{-1} = \frac{1}{\lambda - \hat{\alpha}}$$

When the standard substitution, $y_n = z^n$, is made in the scalar version of (4-5), the result is

$$z^n = \sum_{r=0}^{k-1}\left[\frac{\delta_k - h\hat{\alpha}}{h\lambda - h\hat{\alpha}} + \frac{1}{\lambda - \hat{\alpha}}\left\{\frac{\delta_r - \delta_k}{h}\right\}\right] z^{n-1}\left(1 - \frac{1}{z}\right)^r$$

$$z = \sum_{r=0}^{k-1} \left[\frac{\delta_r - h\hat{\alpha}}{h\lambda - h\hat{\alpha}}\right] (1 - \frac{1}{z})^r$$

We can therefore solve for $h\hat{\alpha}$ given a value of z and $h\lambda$.

(4-15) $$h\hat{\alpha} = \frac{z\, h\lambda - \sum_{r=0}^{k-1} \delta_r \left[1 - \frac{1}{z}\right]^r}{z \left[1 - \frac{1}{z}\right]^k}$$

Again, we use Eqn 4-15 to map the unit circle in the root plane, $z=e^{i\theta}$, into a stability limitation on $h\hat{\alpha}$. The finite values chosen for $h\lambda$ should be interesting values which are easy to work with and analyze. The previous computational examples (Tables 3 through 8) had $h\lambda = -2$.

Once $h\hat{\alpha}$ is found, the value for the eigenvalue of the relative error matrix is

(4-16) $$\mu = \frac{\delta_k - h\hat{\alpha}}{h\lambda - h\hat{\alpha}}$$

which is plotted in the Figure 6 for the 3 step predictor with 4 step corrector (the second order predictor with a third order corrector). The region of absolute stability is the interior of the region since the origin, $\mu=0$, corresponds to the BDF third order corrector which is absolutely stable when $h\lambda = -2$. (See Figure 1).

Figure 6. P_3EC_4 Scalar SIBDF with $h\lambda = -2$

Stability Limitation in the μ-plane

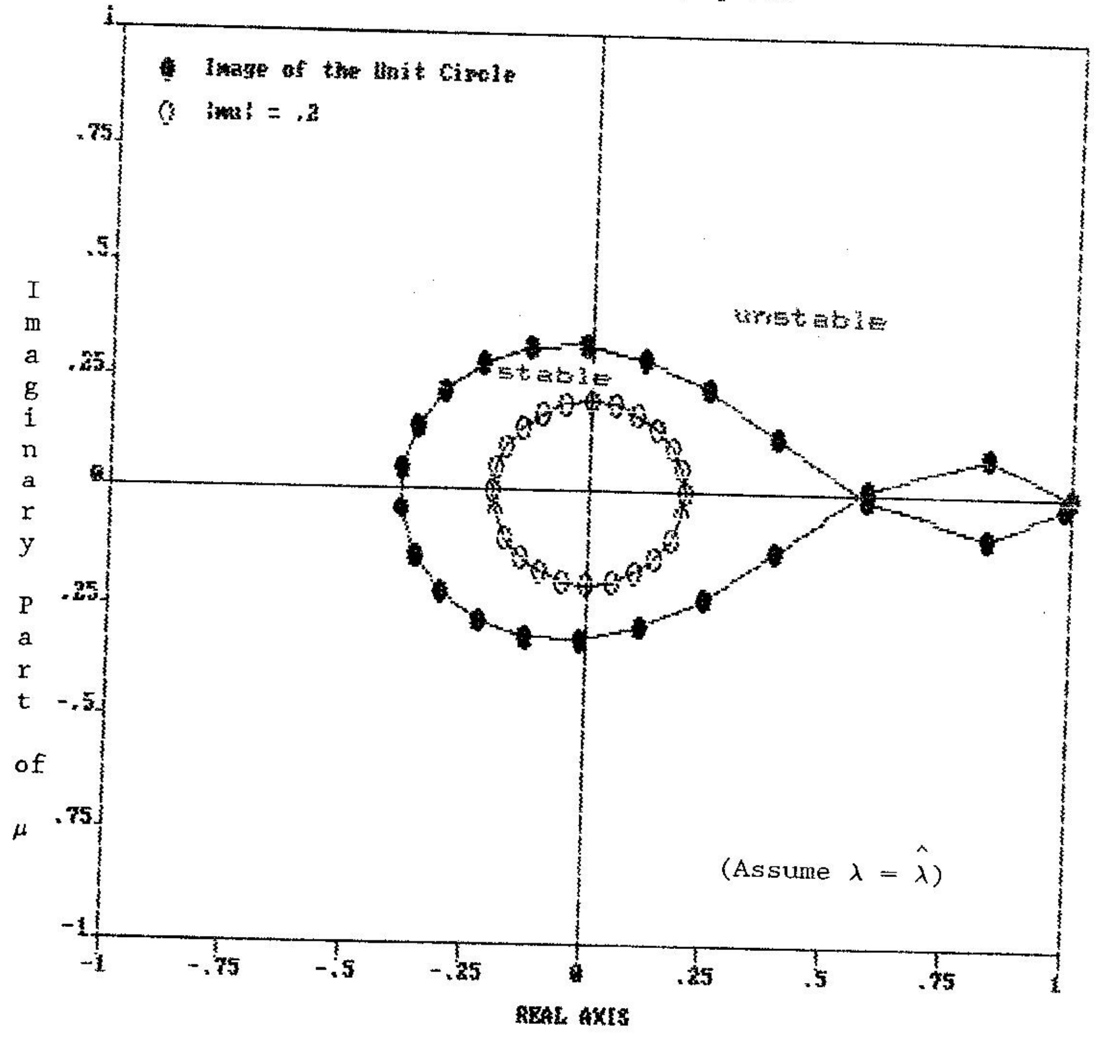


Table 3 (from Section 3) presented the example results for μ=0, $h\lambda$ = -2 and revealed the stability of the P_3EC_4 SIBDF (equivalently the BDF) for $\hat{\alpha} = \delta_3/h = 10.41667$, the correct coefficient since μ=0.

Table 4 presented the case for $h\lambda = -2$ with non-zero μ. The value of μ will also be real, since the scalars involved are real. The value $\mu = -.2$ was chosen for this example because we will relate μ to the rate of convergence of an iterated corrector and some codes accept a convergence rate of .2 as acceptable. Notice the circle of radius .2 is entirely within the boundary image curve in Figure 6, The value $\hat{\alpha}$ = 7.01388 was computed to give this value of μ. Although this represents significant absolute error in the correction matrix, it is a relative error of 20% and is stable for the SIBDF P_3EC_4. We see from Figure 6, that for $h\lambda = -2$, the P_3EC_4 will be stable for any $|\mu| \le .391$.

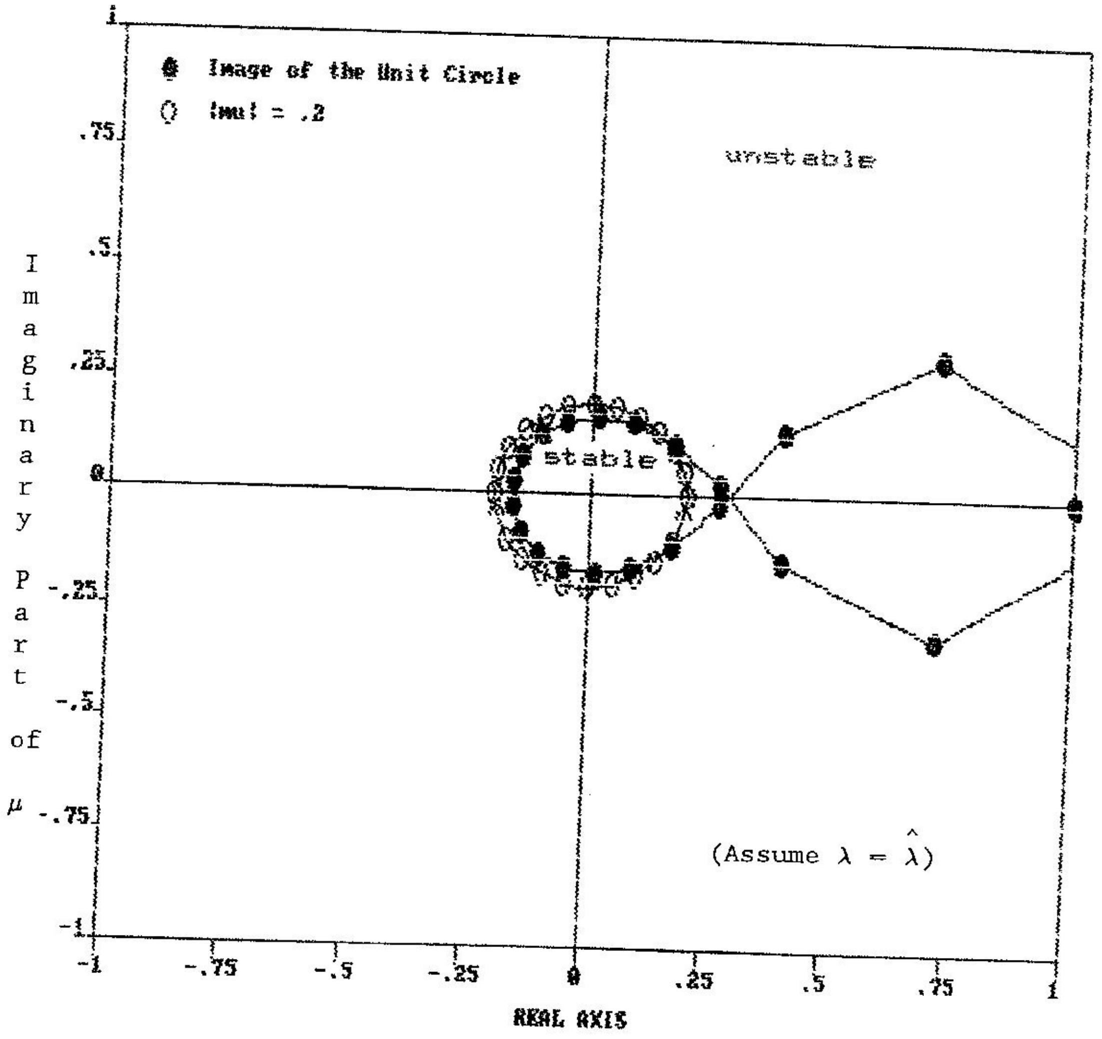
We pass over Table 5 which demonstrated the P_4EC_4 with μ=0 and reproduced the BDF stability behavior.

To explain Table 6, we must examine the stability for a P_4EC_4 SIBDF with matrix error. As we saw in Section 3-4, this uses the same correction matrix, with residual defined by the 4 step predicted solution and derivative. The difference equation (Eqn 4-6) is derived in Section 4.b, differing only in the summation termination. Repeating the development above, we have the scalar equation

$$z = \sum_{r=0}^{k} \left[\frac{\delta_r - h\hat{\alpha}}{h\lambda - h\hat{\alpha}}\right] (1 - \frac{1}{z})^r$$

We fix the value of $h\lambda$, solve for $h\hat{\alpha}$ given z on the root unit circle and compute the corresponding μ. The result is presented in Figure 7, revealing a greatly reduced region for stability in the μ-plane.

Figure 7. P_4EC_4 Scalar SIBDF Method with $h\lambda = -2$
Stability Limitation in the μ-plane

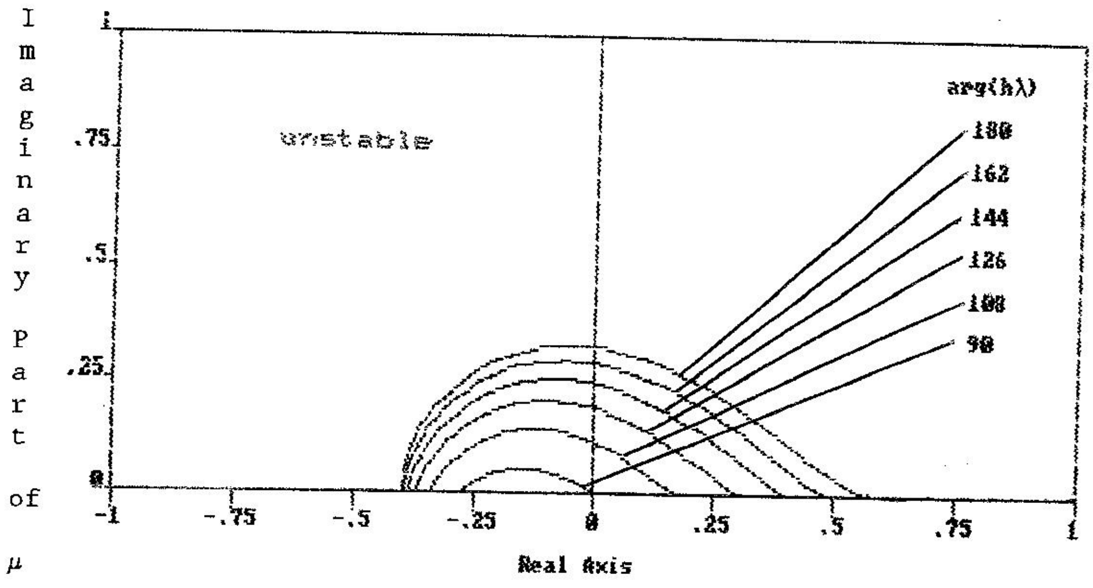


The computed values show the severe effect of correction matrix error when the step number (or order) of the predictor matches that of the residual. We emphasize both examples use the <u>same</u> corrector formula. The same matrix with the same error was able to compute stable values for the SIBDF P_3EC_4 method but not for the P_4EC_4. Of course, if a second correction had been performed for the P_4EC_4 example, stability would have been assured since then $\mu^2 = (-.2)^2 = .04$ would lie well within the 4 step predictor, 4 step corrector stability region. Examining Figure 7, we require $|\mu| \leq .143$ for a P_4EC_4 to remain stable when $h\lambda = -2$.

In Table 7, we saw the benefit of the chosen parameterization for the relative error matrix involving a scalar value, c. Although we have not discussed the arguments that go into the computation of this scalar, the effect of using it is clearly evidenced for the P_4EC_4 case. It has reduced the size of the relative matrix eigenvalue to $\mu = -.009$ which produces a stable computation.

We want more information on the relationship between μ and different linear problems, which are categorized by the choice of λ. It is not convenient to examine in detail the results for each specific $h\lambda$. Thus, we choose to characterize a broad spectrum of $h\lambda$ in the following manner. Given the magnitude of $|h\lambda| = 2$, we need only specify its argument. Six equally spaced arguments for $h\lambda$ in the upper left quadrant were chosen and the corresponding μ computed using Eqns 4-15 and 4-16. The curves would be reflected about the real axis if λ were replaced by $\bar{\lambda}$. Since λ and $\bar{\lambda}$ appear in pairs when solving $y' = Jy$, we need only look at the intersection of the two regions, which happens to be that above the real axis.

Figure 8. P_3EC_4 Scalar SIBDF Method with $|h\lambda| = 2$
Stability Limitation in the μ-plane



Examining Figure 8 for $|h\lambda| = 2$ with $\arg(h\lambda) = 90^\circ$, we see the boundary image curve passes through the origin of the μ-plane. This is consistent since $\mu=0$ is the BDF, and $h\lambda = -2i$ is a point on the stability boundary in the $h\lambda$-plane for the third order BDF (See Figure 1). This indicates that even for the third order, 4 step corrector, better stability is possible for λ purely imaginary if one chooses a non-zero value for μ, say $\mu = -.1i$, and computes corrections with the incorrect $\hat{\alpha}$ which results. Accuracy, though, would be expected to be impaired.

The case of finite $h\lambda$ allows us to respond to the question, "When do the asymptotics apply?", i.e. for what finite value of h will the model results of AAS be applicable. Figures 9, 10, and 11 show the case for P_3EC_4 for $|h\lambda|=1$, $|h\lambda|=5$ and $|h\lambda|=25$, respectively. We see the smooth emergence of the tear-drop shaped region characteristic of the AAS model for m=1. The indication is that for $|h\lambda|=25$, the region is nearly indistinguishable from the asymptotic case. When solving stiff problems, having $|h\lambda| \geq 25$ is not uncommon for some eigenvalue of J, but stability requires all eigenvalues be examined..

Figure 9. P_3EC_4 Scalar SIBDF Method with $|h\lambda| = 1$ Stability Limitation in the μ-plane

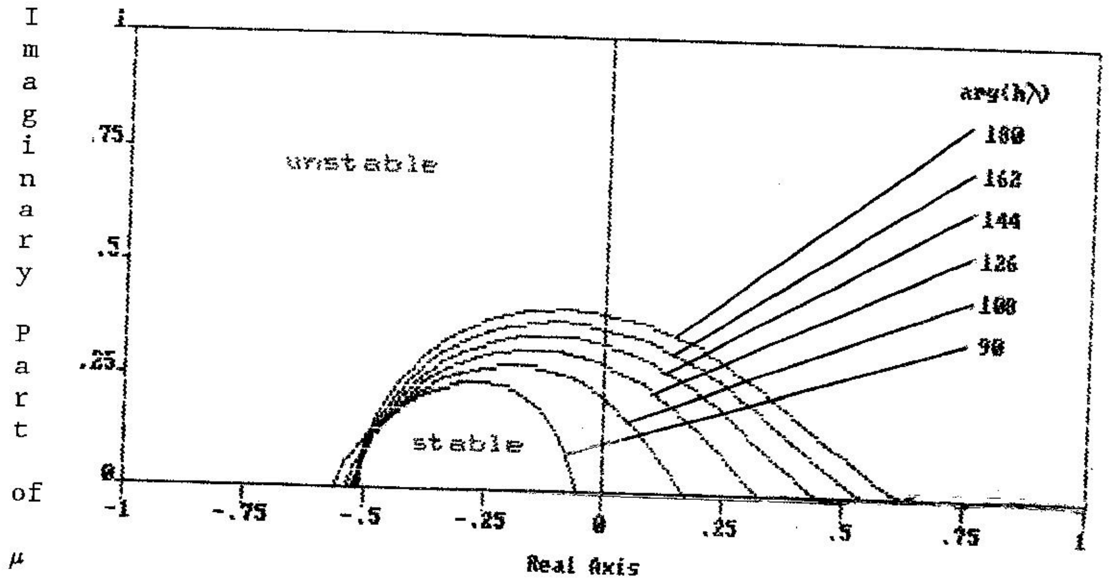


Figure 10. P_3EC_4 Scalar SIBDF Method with $|h\lambda|=5$ Stability Limitation in the μ-plane

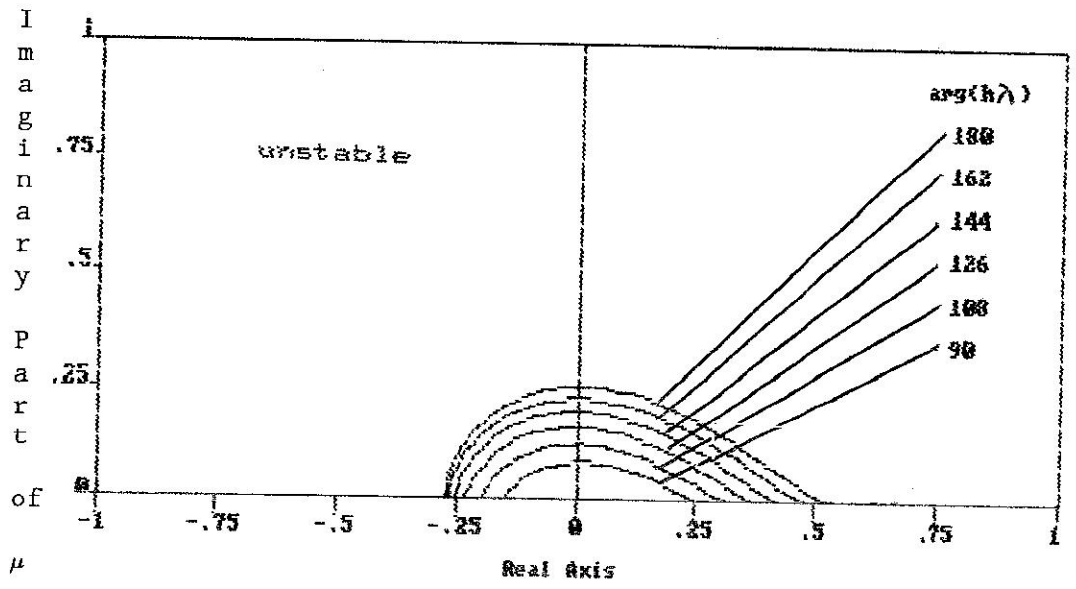
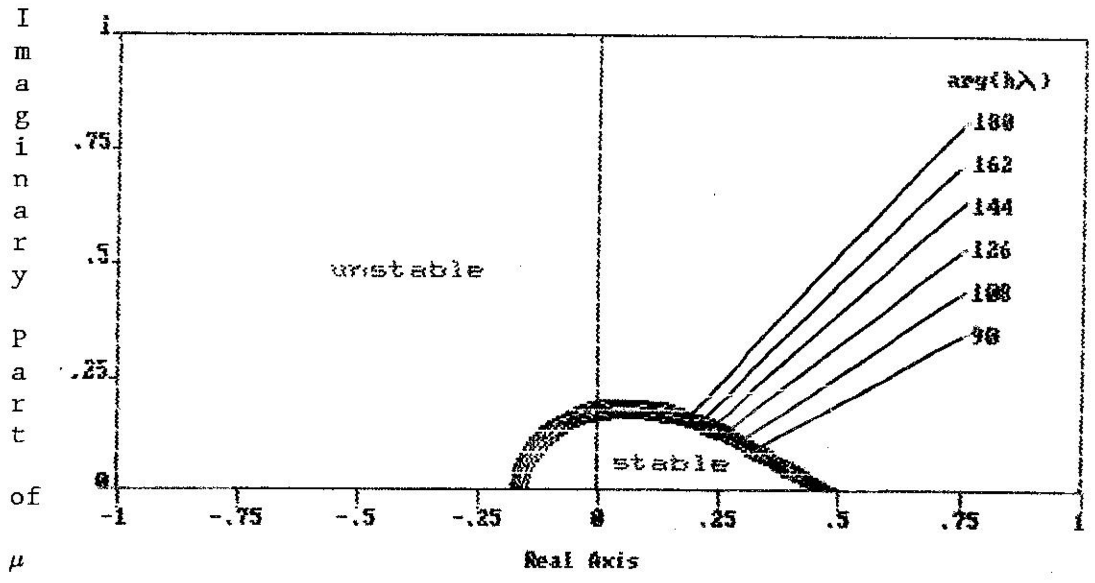


Figure 11. P_3EC_4 Scalar SIBDF Method with $|h\lambda|=25$
Stability Limitation in the μ-plane



Section 4.f. The VGDE Focussed on the Predictor Formula

We extend the applicability of the VGDE by focussing on the predictor formula. This was the approach pursued originally by Klopfenstein [29] and by Krogh and Stewart [35] and allows us to highlight other implementation schemes.

We have seen graphically that the limit case ($h \to \infty$) is applicable for the scalar problem when $|h\lambda| \approx 25$. We have also seen that stability in the limit case $h \to \infty$ of $P(k)\left[EC(k)\right]^m$ and $P(k)\left[EC(k+1)\right]^m$ is the same. These two methods differ only in the coefficient used in the residual. This is the coefficient that multiplies the highest difference in the corrector formula. We parameterize the basic method in terms of this coefficient. Let $p_n = \mathcal{P}_{k,n-1}(x_n)$ and $p'_n = \mathcal{P}'_{k,n-1}(x_n)$ as before in (Eqn 3-1) and (Eqn 3-2). Define the residual

(4-17)
$$r_\alpha = f(y) - p'_n - \alpha\,(y - p)$$

where $\alpha=\delta_k/h$ specifies the k+1 step corrector formula of order k, $\alpha=\delta_{k-1}/h$ specifies the k step corrector of order k-1. The VGDE takes the form

(4-18)
$$y_n = \sum_{r=0}^{k-1} \left[B^m + C_m \left\{\frac{\delta_r}{h} - \alpha\right\}\right] \nabla^r y_{n-1}$$

Klopfenstein used this flexibility in his paper to investigate a choice for method coefficient, α, that would improve the stability of the corrector. Though optimal values were found, the method would be of order k-1. The improvement of stability for the modified BDF corrector appeared was modest from Klopfenstein's analysis.

If we view this as a model of a variable coefficient method (Section 3.c), one could also choose α to implement the fixed-leading coefficient technique (Section 3.c.ii). In this case, the method coefficient $\alpha = \hat{\alpha}$ eliminates the difference with the iteration coefficient. Still we have a method of order k-1, but μ=0 is the Jacobian is exact.

This gives us another interpretation of the $P(k)[EC(k+1)]^m$ choice made earlier. Since stability in the asymptotic limit is the same, we choose to implement the higher order BDF corrector to improve accuracy of the overall scheme. This can be considered local extrapolation [49] of the (k,k) BDF predictor-corrector pair, both of order k-1 contrary to the choice of Byrne and Hindmarsh [2, p. 77] for their variable coefficient implementation of the BDF called EPISODE.. Choosing the higher order corrector with a variable step implementation means the method must be explicitly concerned with relative matrix error. This is discussed in Chapter 6.

Section 4.g. Implications of the Stability Models

The Asymptotic ($h \to \infty$) Absolute Stability model allowed us to choose

$$P(k)\left[EC(k+1)\right]^2$$

as a good method to implement (Sections 4.d.i and 4.d.ii). If a way could be found to estimate the spectral radius of B, $\rho(B)$, dynamically, then this could be used with the results of Table 8 to predict when the two correction matrix is about to go unstable in an order dependent manner. It is to be expected that in comparison to lower order methods, the high order methods would need to update the correction matrices frequently to keep $\rho(B)$ within their more stringent asymptotic limits.

The scalar results indicate that the asymptotic results are conservative. Let us examine the extreme cases of λ real and λ imaginary. When solving problems where $Im(\lambda)=0$, for all λ_i, the μ-region shrinks uniformly as $h\lambda \to \infty$. Therefore, the AAS results are conservative when applied to the problem with $h\lambda$ real and finite for all values of h. Of course, the BDF have excellent behavior on the negative real axis and we anticipated a correspondingly favorable result for the SIBDF.

As $Im(\lambda_i)$ begins to grow for some λ_i, we must be content with the poor behavior of the BDF on the imaginary axis. Figure 9 confirms the fact that the AAS does not contain the origin for λ pure imaginary, with no matrix error ($\mu=0$). The stability regions which must contain $\mu \in \sigma(B)$ are larger for small $h\lambda$. As $h\lambda \to \infty$, the regions uniformly shrink, thereby allowing less error to be tolerated in the eigenvalues of the relative error matrix, B.

Chapter 5. Accuracy

Following Chapter 2 (Eqn. 2-6), we take the local truncation error (τ) to be the difference between the true solution and that provided by the method using past true solution values. The resulting form for τ will reveal the dependence of τ on h and k, but also on μ, the eigenvalues of the relative error matrix, as well on as the number of corrections that are made. Since the SIBDF and the BDF are closely related, we will find that τ_n^{SIBDF} and τ_n^{BDF} are equal when either $\mu=0$ (the BDF), or $|\mu|<1$ and $m\to\infty$ (BDF has been iterated to completion).

Since the order of the predictor played a prominent role in the Asymptotic Absolute Stability model, we begin with the local truncation error of the k-step predicted solution, p_n, (Eqn. 3-1). This is the result of extrapolation of the polynomial defined by k interpolation conditions at equally spaced back points.

$$\begin{aligned} y(x_n) - p_n &= h(2h)(3h)\dots(kh)\; y[x_{n-1},\dots,x_{n-k},\zeta_{x_n}] \\ &= \nabla^k y(\xi_{x_n}) = h^k\, y^{(k)}(\eta_{x_n}) \end{aligned} \tag{5-1}$$

where ξ_x, ζ_x, and η_x represent unknown points depending on x in the smallest interval containing $x_{n-k},\dots,x_{n-1}$, and x.

Section 5.a. P_kEC_{k+1} Method for a Nonlinear System

To obtain the local truncation error for the computed solution, y_n^{mc}, we must take into consideration the algebraic system defining it. First, we look at the case for m=1 applied to the <u>nonlinear system</u>, $y' = f(x,y)$. The method is then defined by equations 3-1, 3-2, 3-4, and 3-5 from Chapter 3 to yield

$$y_n^{1c} = p_n - \hat{G}^{-1} r_n^{(0)}$$

and the local truncation error results from subtracting from the true solution to yield

$$\tau_n^{1c} = y(x_n) - \varphi_{k,n-1}(x_n) + \hat{G}^{-1} \left[f(x_n, p_n) - \varphi'_{k,n-1}(x_n)\right]$$

All work is at $x = x_n$, thus the independent variable will be suppressed. We do not assume autonomous systems. Using a Mean Value Theorem,

$$f(p_n) = f(y(x_n)) + J^* \, (p_n - y(x_n))$$

We have

$$\tau_n^{1} = \hat{G}^{-1}\Big[\hat{G} \, (y(x_n)-p_n) + f(y(x_n) \,) + J^* \, (p_n-y(x_n) \,) - \varphi'_{k,n-1}(x_n) \Big]$$

$$= \hat{G}^{-1} \left[(\hat{G}-J^*) \, (y(x_n) - p_n) + \frac{d}{dx} \left\{ y(x) - \varphi_{k,n-1}(x) \right\}\Big|_{x=x_n} \right]$$

Since $y(x) - \varphi_{k,n-1}(x) = \dfrac{(x-x_{n-1})\ldots(x-x_{n-k})}{h^k \, k!} \nabla^k y(\xi_x)$, using the definition of δ_k (Eqn 3-3), we have

$$\tau_n^{1c} = \hat{G}^{-1} \left[(\hat{G}-J^*) \, (y(x_n) - p_n) + \frac{\delta_k}{h} \nabla^k y(\xi_{x_n}) + \frac{d}{dx} \nabla^k y(\xi_x)\Big|_{x=x_n} \right]$$

Let $G^* = J^* - \frac{\delta_k}{h} I$. Using $y(x_n) - p_n = \nabla^k(\xi_{x_n})$ from (Eqn. 5-1), we have

(5-2) $$\tau_n^1 = \hat{G}^{-1}[\hat{G}-G^*]\ \nabla^k y(\xi_{x_n}) + \hat{G}^{-1}\left\{h^k\ y^{(k+1)}(\nu_{x_n})\right\}$$

Consider the situation in the nonlinear case when $\hat{G} = G^*$ so that the first term in (Eqn 5-2) disappears. With the assumption of no matrix error, we know $y_n^{1c} = y_n^{bdf}$, whose local truncation error (Eqn 2-6) can then be expressed

$$\tau(y_n^{bdf}) = \left[\ G^*\ \right]^{-1}\left\{\ h^k\ y^{(k+1)}(\nu_{x_n})\right\}$$

This leads us to interpret the local truncation error of the one correction SIBDF as the sum of an approximation to the local truncation error of the BDF plus the effect on the error of the predictor due to the term $-\hat{G}^{-1}(G-\hat{G})$, which we have already seen controls the stability of the formula.

Section 5.b. $P(k)\left[EC(k+1)\right]^m$ Method for $y'=Jy$

To analyze the case of a two or more correction method adds the increased complexity of a different J^* for each correction. To make the analysis tractable, we assume the case of the linear, constant coefficient system so that $J^* = J$ for all corrections.

We begin with Eqn. 4-9 in the derivation of the vector generalized difference equation

$$y_n^{mc} = y_n^{bdf} + B^m\left[p_n - y_n^{bdf}\right]$$

The local truncation error, $\tau_n^{mc} \equiv y(x_n) - y_n^{mc}$ is expressed as

(5-3) $$\tau_n^{mc} = \tau_n^{bdf} + B^m \left[y_n^{bdf} - p_n\right]$$

The effect of additional corrections is to reduce the local truncation error by the effect of B^m, which for a stable method has all its eigenvalues less than one in modulus. However, the term τ_n^{bdf} is always present. Although the two terms may combine to give an answer even smaller on any particular step, over a sequence of steps any large values appearing will dominate.

This form also relates the SIBDF to BDF when an iteration scheme is applied to solve the implicit equation. Recall Eqn. (3-13) stating that

$$B = I - \hat{G}^{-1}G = I - \hat{P}^{-1}P$$

when the BDF uses the iteration matrix $\hat{P}$. Convergence of the iteration requires $\rho(B) < 1$. We interpret τ_n^{mc} to represent the error of the BDF iteration when m corrections are formed. Taking norms, we have

(5-4) $$\|\tau_n^{mc}\| \leq \|\tau_n^{bdf}\| + \|B^m\| \; \|y_n^{bdf} - p_n\| \leq \|\tau_n^{bdf}\| + \|B\|^m \; \|y_n^{bdf} - p_n\|$$

Section 5.c. Estimation of the Local Error

The estimation of local error, LE, is more expensive for the SIBDF than for an Adams predictor/corrector due to the need to solve a linear system in the solution process. The method attempts to satisfy the collocation condition on the derivative and we want the error of the corresponding solution. The estimate used in the code STRUT was first presented by Krogh [34] Eqn (3.2.4). We estimate local error by the difference of a result of order k, y_n^k, and a result of order k-1, y_n^{k-1}.

$$LE \approx y_n^k - y_n^{k-1}$$

The solution process computes m corrections to the predicted value. Let the forward difference operator,

$$\Delta^{(r)} y_n^k ,$$

be used to denote the r th correction at x_n for the k th order corrector formula (when less detail was needed (Eqn 3-4), we called this term $\Delta y^{(r)}$). Similarly for $\Delta^{(r)} y_n^{k-1}$. The correction process (Eqns 3-1, 3-2 and 3-4) yields

$$y_n^k = p_n + \sum_{r=1}^{m} \Delta^{(r)} y_n^k$$

Eqns. 3-7, 3-8, 3-9 must be adjusted to produce the $P(k)\left[EC(k)\right]^m$ method using a different correction matrix, $\hat{G} = J - \frac{\delta_{k-1}}{h} I$, than $P(k)\left[EC(k+1)\right]^m$, which used $\hat{G} = J - \frac{\delta_k}{h} I$. This defines

$$y_n^{k-1} = p_n + \sum_{r=1}^{m} \Delta^{(r)} y_n^{k-1}$$

To reduce the cost, we derive an approximation for LE in terms of information already available from the correction process. We focus on the <u>first correction</u> defined by the relations

(5-5) $$(J - \frac{\delta_k}{h} I)\, \Delta^{(1)} y_n^{k} = \varphi'_{k,n}(x_n) - f(p_n)$$

(5-6) $$(J - \frac{\delta_{k-1}}{h} I)\, \Delta^{(1)} y_n^{k-1} = \varphi'_{k,n}(x_n) - f(p_n)$$

Adding $\frac{\delta_k - \delta_{k-1}}{h} \Delta y^k$ to both sides of the first equation and subtracting gives

(5-7) $$(J - \frac{\delta_{k-1}}{h} I)\, (\Delta y_k^{(1)} - \Delta y_{k-1}^{(1)}) = \frac{\delta_k - \delta_{k-1}}{h} \Delta y_k^{(1)} = \frac{\delta_k^*}{h} \Delta y_k^{(1)}$$

It was felt impractical to solve a linear system to obtain an error estimate, though this is the tactic used by Sacks-Davis [43] in his implementation of the second derivative method for stiff ODEs. Additionally, there is the complication that the matrix available to our $P(k)\left[EC(k+1)\right]^2$ method is $J - \frac{\delta_k}{h} I$. We develop a simpler estimator through reasonable assumptions.

We can examine the behavior as $h \to 0$ in Eqn 5-7, by first multiplying through by h and then taking the limit and solving for

(5-8) $$LE \approx \frac{\delta_k^*}{\delta_{k-1}} \Delta^{(1)} y_n^k \ .$$

At the other end of the spectrum, when $|h\lambda_i|$ is large, we assume that $J = \text{diag}(\lambda_1, \ldots, \lambda_n)$, $\text{Re}(\lambda_i) < 0$. We can directly solve for the i^{th} component of the estimator

$$\left[\Delta^{(1)} y_n^k - \Delta^{(1)} y_n^{k-1}\right]_i = \frac{\delta_k^*}{h\lambda_i - \delta_{k-1}} \left[\Delta^{(1)} y_n^k\right]_i$$

Setting $\lambda_i = 0$ makes this more conservative and yields the same result.

Thus, Eqn. 5-8 is the actual local error estimator used in the code STRUT. Note that it is based on first correction information only.

Chapter 6. Implications of Relative Matrix Error

Stability theory in Chapter 4 guided us toward the $P_k\left[EC_{k+1}\right]^2$ method. Limits are placed on the size of the eigenvalues, μ, of relative matrix error, B, for an Asymptotically ($h\to\infty$) Absolutely Stable method. The code must now explicitly monitor this to keep the process stable. We explore in this final chapter the implications for a SIBDF implementation of the theory developed in the previous chapters.

The Modified Divided Difference (MDD) representation was chosen for the SIBDF formulas based on its improved performance in Adams implementations (Section 3.c.i). This allows variable step size formulas, at the expense of recomputing the differentiation coefficients for k steps following a single change in step size. For an effective MDD method, it is necessary to have some mechanism to smooth out the effect of coefficient error in the linear algebra. Otherwise, one would reform and factor the correction matrix due to the small changes that occur after a single change in stepsize. This task is accomplished by the parameter c (Section 6.a).

In Section 6.b., we discuss how one dynamically uses estimates of μ to improve performance. This includes using PEC steps when possible, and PECC steps to perform iterative improvement, saving on a function evaluation. Commonly used strategies of updating a matrix whenever order or step change were avoided in the implementation of a code using the SIBDF since these changes are frequent. Instead the estimates of μ signal the need for correction matrices and ODE Jacobian.

Finally, in Section 6.c., we discuss how estimates of $|\mu|$ are computed. We relate μ to convergence rates used by BDF codes to judge the iteration process and point out the limitation of using the norm to estimate eigenvalues. Direct estimation of the eigenvalues of $-\hat{G}^{-1}(G-\hat{G})$ are used to aid in method selection and inexpensive techniques were developed.

Section 6.a. Low Cost Reduction in Matrix Error

The scalar c modification of the correction matrix is relied upon to reduce error in the correction matrix due to changes in the MDD coefficients after a change in step size and, when possible, a change in order.

It is common in the integration of a stiff system that there will be regions where the stepsize will grow rapidly when a method without stability restrictions is used. Consider the model problem, $y' = Jy$, with J given by the 2 by 2 matrix

$$J = \begin{bmatrix} \lambda & \omega \\ 0 & \mu \end{bmatrix}$$

with $\mathrm{Re}(\lambda) \ll \mathrm{Re}(\mu) < 0$. The solution given starting value

$$y(0) = y_0$$

is

$$y(x) = e^{Jt} y_0 = \begin{bmatrix} e^{\lambda t} & \omega \dfrac{e^{\lambda t} - e^{\mu t}}{\lambda - \mu} \\ 0 & e^{\mu t} \end{bmatrix} y_0$$

The truncation error will involve scalar multiples of

$$h^{k+1}y^{(k+1)}(x_n) = h^{k+1}\,J^{k+1}\,y(x_n) =$$

$$\begin{bmatrix} (h\lambda)^{k+1} & h^{k+1}\omega\,(\lambda^k+\lambda^{k-1}\mu+\ldots+\lambda\mu^{k-1}+\mu^k) \\ 0 & (h\mu)^{k+1} \end{bmatrix} y(x_n)$$

whose norm will die out only as the components of $y(x_n)$ decay.

Once $e^{\lambda t}$ has decayed out, accurate computation of the first component in the solution may still impose a limitation on the step size due to the off diagonal term coupling in the $e^{\mu t}$ term. This depends on the relative size of λ, μ and ω. In the case where ω is small, the step size will be limited only by stability until the product $h\mu$ becomes active in the error estimator. We would like to allow increases in step size as rapidly as possible while maintaining stability.

We use the scalar parameter c to reduce the portion of the relative matrix error that is due to changes in the step size or order. Therefore, we assume there is no error in the ODE Jacobian, $J = \hat{J}$. Actual implementation of the strategy we will develop demonstrated that this is a reasonable assumption since it produced improved performance even on the nonlinear problems. We repeat and expand on the presentation in [35].

Let α represent the currrent ratio of method coefficient, d_k, to step size, h (as in Section 4.f)

$$\alpha = \frac{d_k}{h}$$

Let $\hat{\alpha}$ represent the corresponding value that was used when the correction matrix $\hat{G}$ was formed. As in Eqn. 3-6, the correction matrix is $\hat{G} = c\,(J - \hat{\alpha} I)$, but now we want to think of the $\hat{\alpha}$ corresponding to some previous time step when, perhaps, a different order and step size was being used. The relative error matrix is then

$$B = -\hat{G}^{-1}(G - \hat{G}) = \frac{1}{c}\,(\hat{\alpha} I - J)^{-1}\,[(1-c)\,J - (\alpha - \hat{\alpha})\,I]$$

If λ is an eigenvalue of J, then

$$\mu(c,\lambda) = 1 - \frac{1}{c}\,\frac{\alpha - \lambda}{\hat{\alpha} - \lambda} \tag{6-1}$$

is an eigenvalue of B.

An argument is presented in [35] to specify an optimal c under the restriction that it be a real scalar. This yields a simple scheme for incorporating c into the solution process. We have the linear system, $c(\hat{J} - \hat{\alpha} I)\,\Delta y = -r$, which we rewrite as

$$(\hat{J} - \hat{\alpha} I)\,\Delta y = -\frac{1}{c}\,r$$

The LU factors of $(\hat{J} - \hat{\alpha} I)$ are retained from step to step, and the residual is multiplied by $1/c$ before the solve is done. It is then simple to vary c on every step to compensate for matrix error.

The derivation of the strategy used to select c is long and will not be repeated here. Instead, we look at some of the highlights. For a scalar problem, $y' = \lambda y$, one can show that when the step size is consistently increasing, a value of $c < 1$ will reduce the size of μ.

Similarly, when the step size is decreasing, c > 1 will reduce μ. Since the nominal value for c is near one, we write c as

$$c = 1 + \frac{\alpha - \hat{\alpha}}{s}$$

for a real value s. It is then shown that for any given $\lambda\epsilon$, the value for s which minimizes $|\mu|$ is

$$s = \hat{\alpha} - \mathrm{Re}(\lambda) + \frac{\mathrm{Im}(\lambda)^2}{\alpha - \mathrm{Re}(\lambda)}$$

It is then found that the argument for λ that produces the largest $|\mu|$ is $\arg(\lambda) = \pi/2$. Assuming tht $\lambda = iy$ is purely imaginary in order to reduce the complexity of the task produces a conservative estimate for the worst case λ as

$$s = \hat{\alpha} + \frac{y^2}{\alpha}$$

It is not possible to choose a single value to minimize

$$\rho(B) = \max_i |\mu_i|$$

when J has n eigenvalues, λ_i. To determine whether $\rho(B)$ will be increasing or decreasing with y over the whole range of eigenvalues, we look at the boundaries of the spectrum. Let

$$s_1 = \hat{\alpha} + \frac{|\lambda_{\max}|^2}{\alpha} \qquad s_2 = \hat{\alpha} + \frac{|\lambda_{\min}|^2}{\alpha}$$

One can show by algebraic arguments that if $s_1 < 2\hat{\alpha}$, then $|\mu|$ increases as $|\lambda|$ increases and we minimize $|\mu|$ by using the largest value of $|\lambda|$, $|\lambda_{max}|$ and set $s = s_1$. If $s_2 > 2\hat{\alpha}$, then $|\mu|$ decreases as $|\lambda|$ increases. We use the smallest value of $|\lambda|$, $|\lambda_{min}|$, and set $s = s_2$. It is quite likely that neither condition holds since we know $s_1 \geq s_2$. In this case one balances between the largest and smallest $|\lambda|$ and uses $s = 2\hat{\alpha}$.

A suitable value for c is thus determined from the range of the spectrum of J.

Section 6.b. Dynamic Uses of Relative Matrix Error

When a BDF method is implemented, one must decide how to terminate the iteration on the collocation condition and when to update matrices. Many implementations update the iteration matrix whenever step size or order is changed. When using a fixed step size formula, this implies that the only error in the iteration matrix is due to changes in the Jacobian of ODE. Termination of the iteration is typically decided using ratios of successive corrections. A lack of convergence is due to a step size too large or ODE Jacobian error.

In Section 4.c., the Asymptotic Absolute Stability model assumed a linear constant coefficient system and fixed order and step size. The analysis allowed us to choose the $P_k[EC_{k+1}]^2$ SIBDF as a reasonable method to implement. The implementation used in the test code STRUT for general nonlinear differential equations varies the step size and the order at each step. Because this differs significantly from the theoretical assumptions, experimentation was needed to verify the practicality of the idea.

Our approach to the SIBDF uses estimates of the eigenvalues of the relative error matrix, B, to indicate the need for new correction matrices before the step is attempted. Although there is no termination decision since m=2 is the fixed number of corrections (see Section 4.d.i), a $\rho(B)$ estimate below the m=1 limit will be used to justify making only one correction whenever possible.

6.b.i. Requesting a New Correction Matrix or Jacobian

We postpone discussing how the matrix error estimates are obtained until Section 6.c. There are two estimators,

$$\mathrm{REG} \approx \rho(B) = \rho(\hat{G}^{-1}(G-\hat{G}))$$

approximating the largest eigenvalue of the relative error matrix assuming $J = \hat{J}$ and

$$\mathrm{REJ} \approx \rho(\hat{G}^{-1}(J-\hat{J}))$$

approximating the largest eigenvalue of the relative error of the Jacobian.

At the end of each step, the order and step size are selected by an algorithm similar to those proposed by Krogh [34]. The value of REG is predicted. The implementation in STRUT uses a modified version of the AAS limits presented in Section 4.c. The order 6 predictor is not used. The minimum $|\mu| = \rho(B)$ for predictors of order zero and one predictor are more conservative. The following array is called STABLE.

Table 16. The array STABLE (k,m) used in STRUT

	Values of k					
	0	1	2	3	4	5
m						
1	0.35	0.30	0.143	0.067	0.032	0.016
2	0.50	0.45	0.387	0.258	0.179	0.126

If REG > STABLE(2,k), indicating that more than two corrections will be needed to keep the method stable, STRUT will form a new correction matrix before the next step is taken. There is still the question as to whether a significant change in the local behavior of the ODE has occurred since $\hat{J}$ was obtained or whether changes in order or step size caused the error in $-\hat{G}^{-1}(G-\hat{G})$. Hindmarsh has used the technique of using the old Jacobian when forming a new iteration matrix if order and step size changes account for at least a 30% change from $\hat{\alpha}$. [Hindmarsh, personal communication, 1982]. He has seen evidence in the use of the sparse matrix code LSODI of the benefit of this strategy.

With the estimator REJ, STRUT attempts to measure directly how much $\hat{J}$ differs from a J^* defined by a Mean Value Theorem. If REJ is less than the STABLE(1,k) value, the old $\hat{J}$ is judged sufficiently accurate, so $\hat{G}$ is formed using the old $\hat{J}$ and the updated coefficient, $\hat{\alpha}$ (provided, of course, that the user has been willing to provide the extra n^2 storage locations for the old Jacobian).

6.b.ii Reducing the Number of Corrections

At the beginning of the step, the order has already been selected at the end of the previous step. The value REG is tested against

STABLE(k,m) to select the number of corrections to perform, m=1 or m=2. A value of REG larger than the 2 correction limit is unlikely since the value was predicted at the end of the previous step and such a value would have caused a new matrix to be formed. Should this occur, though, two corrections are performed and a new correction matrix formed to reduce $|\mu|$ before the next step is attempted.

There is a close relationship between the size of the largest eigenvalue of the relative error matrix and the rate of convergence should the corrector be iterated as described in Section 6-c (Eqn. 6-3). The rate of convergence is used by most of the current production BDF codes to gain assurance that the corrector iteration has solved the implicit equation to a sufficient accuracy. A rate of at least 0.2 is commonly required. The analysis of Asymptotic Absolute Stability showed an order dependent relationship between stability and $|\mu|$, hence that using a single bound would not be effective for the SIBDF.

We found experimentally that using the rate of convergence estimators was impractical for the needs of STRUT for three reasons:

1) Unreliable. Noise in the computation of the residual is typically magnified by the large condition of the correction matrix into the corresponding Δy.

2) Crude. The norm provides only a crude upper bound on the eigenvalue. Overestimating this value results in a loss of efficiency for our implementation of the SIBDF due to requesting correction matrices much more often than necessary.

3) Expensive. A two iteration method (PECEC) is needed to estimate $\|\Delta y^{i+1}\|/\|\Delta y^{i}\|$ or a two residual method (PECE) is needed to estimate $\|r^{i+1}\|/\|r^{i}\|$ if these estimators are to be formed at each step.

Though we selected $P\left[EC\right]^2$ for implementation, PEC steps are frequent in STRUT.

The essence of the matter is that the algorithms in STRUT attempt to predict error rather than respond after the trouble has been observed.

6.b.iii Reducing Function Evaluations - PECC Steps

When a two correction method is to be attempted at the current step, a small value of REJ suggests that the error in the correction matrix is due to a discrepancy between $\hat{\alpha}$ and d_k/h. Thus iterative improvement ([13], p. 49-54) might have an effect similar to that of a correction using a new function evaluation. The typical correction scheme is:

$$y^{(0)} = p$$
$$r^{(0)} = f(p) - p'$$
$$y^{(1)} = y^{(0)} + \Delta y^{(1)}$$
$$r^{(1)} = f(y^{(1)}) - p' - \frac{d_k}{h} (y^{(1)} - p)$$

We rewrite the second residual using a Mean Value Theorem

$$f(y^{(1)}) = f(y^{(0)}) + J^* \Delta y^{(1)}$$

as

$$r^{(1)} = (J^* - \frac{d_k}{h} I) \Delta y^{(1)} + r^{(0)}$$

$$= (J^{*} - \frac{d_k}{h} I)\ \Delta y^{(1)} - \hat{G}\ \Delta y^{(1)}$$

$$= [(J^{*} - c\hat{J}) - (\frac{d_k}{h} - c\hat{\alpha})]\ \Delta y^{(1)}$$

If J^{*} is sufficiently near $c\hat{J}$, we can use

$$r^{(1)} = (c\hat{\alpha} - \frac{d_k}{h})\ \Delta y^{(1)} \tag{6-2}$$

to generate the second residual. The linear system is then solved for $\Delta y^{(2)}$.

The decision on when $c\hat{J} - J^{*}$ is sufficently small is made in an order dependent manner by using the STABLE array. It must be the case that REG > STABLE(k,2) to force two corrections. If REJ < STABLE(k,1), then the PECC is performed with (6-2) used to generate the second right hand side.

6.c. Estimation of the Relative Matrix Error

The Asymptotic Absolute Stability model for the SIBDF restricts the eigenvalues of the correction matrix, $B = -\hat{G}^{-1}\ (G - \hat{G})$. Thus an implementation of the SIBDF must control these values to remain stable. There are estimates of $\|B\|$ available from the ratio of the norms of successive corrections. Let J^{*} be the matrix guaranteed by a Mean Value Theorem satisfying

$$f(y^{m+1}) = f(y^{m}+\Delta y^{m}) = f(y^{m}) + J^{*}\ \Delta y^{m}\ .$$

In forming the correction, Δy^m, we have

$$\hat{G}\ \Delta y^{m+1} = -r^{m+1} = -\ [f(y^m) - p' - \frac{\delta_k}{h}(y^m - p)]$$

$$= -\ [\ (J^* - \frac{\delta_k}{h} I)\ \Delta y^m + r^m]$$

$$= -\ [G^* - \hat{G}]\ \Delta y^m$$

with $G^* = J^* - \frac{\delta_k}{h} I$. Alternatively, one has

$$r^{m+1} = [G^* - \hat{G}]\ (-\ \hat{G}^{-1}\ r^m)$$

If one is willing to perform two corrections, lower bounds are available for the norm of $B^* = I - \hat{G}^{-1}G^*$ from norm of successive corrections.

(6-3) $$\frac{\|\Delta y^{m+1}\|}{\|\Delta y^m\|} \leq \|B^*\| \approx \|B\|$$

The norm of the matrix which was the basis of Klopfenstein's analysis, $(G-\hat{G})G^{-1}$, can, then, be assessed by the ratio of norms of successive residuals.

Unfortunately, an upper bound on the spectral radius of B is needed for any stability decisions and, as we observed earlier, the norm approach provides too crude an estimate of the spectral radius. We therefore estimate directly the largest eigenvalue of B.

A scheme that has proven effective is to estimate the error in B assuming that the ODE Jacobian is correct. Recalling the form chosen for the correction matrix with $J = \hat{J}$, $\hat{G} = c(J-\hat{\alpha}I)$, we developed an estimate of the maximum eigenvalue of

$$B = \frac{1}{c} (J - \hat{\alpha}I)^{-1} [(1-c)J - (\alpha - c\hat{\alpha})I]$$

in Section 6.a in the process of obtaining a good value for the scalar parameter c using estimates of $|\lambda_{max}|$ and $|\lambda_{min}|$ of the ODE Jacobian. Given c, we compute using Eqn. 6-1 the value

$$|\mu| \Big|_{J=\hat{J}} \tag{6-4}$$

as an estimate of the spectral radius of B.

A power method in real arithmetic, based on the algorithm of Franklin ([14], pp. 238-239), is performed to estimate the magnitude of the possibly complex eigenvalue λ_{max}. Similarly, an inverse power method is used to estimate $|\lambda_{min}|$. These are expensive procedures and are used sparingly.

We estimate separately the absolute error of the Jacobian. Limit consideration to autonomous systems

$$y' = f(y)$$

and examine the difference of a J^* spanning several time steps and the $\hat{J}$ currently used in $\hat{G}$. Let

$$J^* \Delta y = \Delta f$$

where $\Delta y = y(x_{curr}) - y(x_{past})$ and $\Delta f = y'(x_{curr}) - y'(x_{past})$. The Δy is chosen large enough to allow differences to be formed that will not be dominated by round off error. We must store two n-vectors of past information, but it is hoped this Δy will reveal the current action of the Jacobian. The effect of a forcing function in $f(x,y)$ is being ignored, but one can argue that for stiff problems, this is not a serious consideration. A forcing function would be holding the step size down for accuracy if it were contributing significantly to the derivative and the problem would not be stiff.

One estimates the absolute error in $\hat{J}$ by

$$\| (J^* - \hat{J}) \Delta y \| / \|\Delta y\| = \| \Delta f - \hat{J} \Delta y \| / \|\Delta y\|$$

$$= \| \Delta f - (\tfrac{1}{c} \hat{G} + \hat{\alpha} I) \Delta y \| / \|\Delta y\|$$

which has the same cost as a backsolve/forward substitution since $\hat{G}\Delta y$ must be formed from the LU factors of the correction matrix. This absolute error is then taken relative to the size of $\hat{G}$ producing the estimate

(6-5) $$REJ = \frac{\|\Delta f - \hat{J}\Delta y\|}{\|\hat{G}\| \; \|\Delta y\|}$$

The relative error of $\hat{G}$ is then estimated as the sum of Eqn. 6-4 and Eqn. 6-5

(6-6) $$\mathrm{REG} = |\mu| \Big|_{J=\hat{J}} + \mathrm{REJ} \quad .$$

Chapter 7. Conclusions

The Semi-Implicit BDF form an effective alternative family of variable order, variable step size formulas for solving stiff ODEs. When compared to the BDF, they have the additional cost of estimating eigenvalues of the Jacobian to produce estimates of $\rho(B)$. The benefit is better control over the linear algebra needs of the algorithm.

The Asymptotic ($h\to\infty$) Absolute Stability model presented in 4.d relates the order of predictor to the amount of tolerable relative matrix error. Since the tolerance of relative matrix error decreases strongly as the order of the formula increases, we conclude that more care must be taken with the higher order formulas. Since relative matrix error is related to the convergence rate when the BDF are iterated, BDF implementations may benefit from incorporating an order dependent test on convergence.

The Asymptotic ($h\to\infty$) Absolute Stability results showed that performing two corrections substantially increases the amount of tolerable relative matrix error over a one correction method, but more than two is not justified. We therefore chose to implement the $m=2$ correction method. When more than two corrections would be required, the matrix error is reduced by forming a new correction matrix.

The stabilty models and example performance also clearly show the benefit of using a predictor of order one lower than corrector. Though the justification of the lower order predictor in the Adams fixed number of correction method is different, STRUT uses Shampine and Gordon's implementation of the

$$P_k E C_{k+1} E$$

for the Adams-Bathforth-Moulton predictor/corrector and the

$$P_k\left[EC_{k+1}\right]^2$$

for the SIBDF.

Implementation of the SIBDF differs from the BDF in that no convergence estimates are needed. Instead, the SIBDF must estimate $\rho(-\hat{G}^{-1}(G-\hat{G}))$. Through experimentation we have verified a reasonable approximation scheme requiring only estimates of the range of the spectrum of the Jacobian. The benefit is better control over the linear algebra needs of the algorithm.

Another implementation choice for the code STRUT was to use the variable coefficient representation of the underlying formulas to enhance stability. This implies the coefficient in the residual changes on the steps following a change in step size which introduces error in the correction matrix unless it is frequently reformed. The AAS theory provides guidelines for the amount of tolerable error. The scalar c proved an effective matrix modification technique to mitigate the effect of both step size and order changes during the integration process without forming new correction matrices.

We feel the demonstrated strengths [52 and Appendix A] of the approach are:

1) Savings in matrix formations/factorizations at the expense of some matrix/vector multiplies with matrices held as LU factors.

2) Separating the decision about the need for ODE Jacobians from that about the need for a new correction matrix. STRUT <u>always</u> uses exactly one Jacobian on linear, constant coefficient problems.

3) Predicting stability behavior helps in many ways to reduce the number of steps needed to complete the integration.

Acknowledgements

Working with Fred Krogh at the Jet Propulsion Laboratory for two years provided a learning experience in mathematical software research that was the starting point of this dissertation. Though the code STRUT was written and maintained by myself, endless hours were spent with Fred discussing implementation considerations and learning from him how to experimentally test alternatives. Fred provided an unending flow of ideas that was a challenge for me to keep up with. I am indebted to his patience to explain things more than once.

Larry Shampine provided technical and editorial guidance in the writing of this dissertation. He consistently gave me encouragement to keep trying to shorten and focus the exposition. His input to clarify ideas went beyond writing style and forced me to finally understand details that I had previously thought I understood.

The different stages of the dissertation benefitted from being read and commented on by Fred Krogh, Bill Gear, Steve Pruess and Dick Allen.

I thank my committee members Dick Allen, Steve Pruess and Stan Steinberg for having faith that my student career at UNM would ever be completed. Finally, I thank Cleve Moler who convinced me to come to UNM, was my advisor until he left the University and was instrumental in getting me into the right department

The computational results reported in Appendix A were performed at the San Diego Supercomputer Center. The facilities were made available through a block grant to San Diego State University by the National Science Foundation.

Appendix A. Numerical Results for STRUT, LSODE, LSODA and DC03A

The following results are an extension of those presented in [52]. They document the performance of the codes LSODE (written by A.C. Hindmarsh [25]), LSODA (LSODE modified by L. Petzold [41]), DC03A (written by A.C. Curtis [9]) and STRUT (written by the author). All are variable order, variable step size codes. The results presented are only a subset of those run and are intended only to demonstrate the viability of the ideas present in this dissertation.

LSODE implements an Adams and BDF predictor-corrector scheme with predictor the same order as corrector. Functional iteration is used on the Adams corrector; a Newton-like iteration is used on the BDF corrector. The BDF code has provisions to handle dense or banded Jacobians. A new Jacobian is formed with each correction matrix. The user must select between the Adams or BDF methods.

LSODA is a modification of LSODE that automatically selects Adams or BDF depending on internally generated diagnostics of the problem behavior. The Adams corrector is still iterated, but is forced through at least two iterations each step to yield an estimate of the Lipschitz constant which is used for stiffness detection following the idea of Shampine [44].

DC03A is from the Harwell Library. It implements an iterated BDF and is intended for stiff ODEs and mixed algebraic differential equations. It has provisions for handling sparse matrices as well as dense. DC03A separates the request for iteration matrices from that for ODE Jacobians which can provide a savings.

STRUT implements the SIBDF with m=2 corrections unless the relative matrix error estimator indicates an m=1 correction method is possible.

It contains the code STEP by Shampine and Gordon [48] which is a $P_kEC_{k+1}E$ Adams implementation. STRUT performs automatic method selection between the Adams and SIBDF based on estimates of the eigenvalues of the ODE Jacobian.

A selection from the Stiff Test Set [12] are presented for absolute error tolerances of 10^{-4} and 10^{-6}. They exercise the method selection properties of LSODA and STRUT. Runs were made with both true Jacobians and differenced Jacobians.

A selection from the Non-Stiff Test Set [26] is also presented for absolute error tolerances of 10^{-4} and 10^{-6} to examine performance on a wider selection of problems. Since no Jacobians are provided with this test set, differenced Jacobians were used throughout. LSODA and STRUT automatically select method. LSODE was run as an Adams method. We additionally forced STRUT and LSODE to run as SIBDF and BDF codes, respectively. This was expected to exercise the matrix error estimator in STRUT and the applicability of the AAS analysis which assumed $h\to\infty$ and would be applied to on non-stiff problems where h would be far from large. This also provides a more reasonable comparison with the performance of DC03A which has no Adams method implemented.

Subject to the code computing a solution within the accuracy that has been requested by the user, we would like to minimize work. We identify work as:

<u>1. Overhead</u>

a. Computing the variable step coefficients and updating the modified divided difference table in STRUT. Interpolating stored data on step size changes and updating the Nordsieck vector in LSODE, LSODA and DC03A.

b. Estimating the error

c. Choosing step size and order

d. Monitoring the stability of the numerical solution for the SIBDF code in STRUT. (The Adams code in STRUT uses a PECE and this is not required.) Monitoring the corrector convergence in LSODE, LSODA and DC03A.

e. Deciding when to switch methods in LSODA and STRUT.

<u>2. Number of evaluations of the differential equation</u>

If Stiff,

<u>3. Number of evaluations of the ODE Jacobian</u>

<u>4. Number of LU decompositions of the correction matrix</u>

<u>5. Number of backsolves/forward substitutions used</u>

The following measures were tabulated for performance evaluation:

Error:

The code LSODE was run with tolerance 10^{-4} times the requested accuracy to generate 'true solution' values at 15 geometrically spaced points in the integration integration. The absolute error was computed at these 15 points and the maximum value is printed for each code.

Step:	Number of steps taken
NFE:	Number of evaluations of the differential equation
NG:	Number of factorizations of the correction matrix
NJ:	Number of evaluations of the ODE Jacobian
TIME:	Total CPU time on NSF San Diego Super Computer Cray X-MP/48
NBDF:	Number of steps taken using the SIBDF/BDF method
BSOL:	Number of backsolves/forward substitutions (and matrix/vector multiplies using the existing LU decomposition by STRUT)

Also of interest:

K	The order of predictor used on the final step
H	Step size used on the final step
EndTol	If the code increased the user's requested error tolerance, then this changed value is printed, else blank.

Selected Problems from the Stiff Test Set

Class A and Class B are linear, constant coefficient problems. STRUT is given space to store both the correction matrix and the ODE Jacobian and requires only one Jacobian to complete the entire integration. DC03A also distinguishes iteration matrix updates from Jacobian updates and uses only one Jacobian. An interesting problem from this set is B5 because it has a dominant eigenvalue near the imaginary axis. The BDFs are known to perform poorly there as evidenced by the performance of LSODE and DC03A. Both LSODA and STRUT do not because they were able to get past the unstable lobe in the left half of the hλ stability plane using an Adams method. When the switch to BDF and SIBDF methods occurred, the step size was large enough to be beyond this stability restriction. LSODE and DC03A were trapped between the origin and the lobe.

Stiff Set No. B5 Tolerance= 1.0e-04 System Size=4 Diffed. J's = t

Code	Error	Step	NFE	NG	NJ	K	EndH	EndTol	Time	NBDF	BSOL
STRUT	1.8e-03	239	444	9	1	3	2.4e+00		5.5e-02	55	97
LSODA	8.2e-04	318	620	9	9	4	1.8e+00		5.2e-01	44	71
LSODE	1.6e-03	2393	3639	146	146	5	9.9e-03		5.6e-01	2393	3639
DC03A	1.2e-03	2458	2752	57	1	4	8.6e-03		4.9e-01	2458	2752

Stiff Set No. B5 Tolerance= 1.0e-04 System Size=4 Diffed. J's = f

Code	Error	Step	NFE	NG	NJ	K	EndH	EndTol	Time	NBDF	BSOL
STRUT	1.8e-03	239	438	9	1	3	2.4e+00		5.5e-02	55	97
LSODA	8.2e-04	318	620	9	9	4	1.8e+00		5.1e-01	44	71
LSODE	1.7e-03	2350	2716	142	142	5	7.8e-03		5.3e-01	2350	2716
DC03A	1.2e-03	2460	2760	65	1	4	9.2e-03		4.9e-01	2460	2760

Stiff Set No. B5 Tolerance= 1.0e-06 System Size=4 Diffed. J's = t

Code	Error	Step	NFE	NG	NJ	K	EndH	EndTol	Time	NBDF	BSOL
STRUT	1.8e-05	453	855	8	1	4	1.0e+00		9.4e-02	73	140
LSODA	9.7e-06	633	1247	8	8	4	7.0e-01		9.3e-01	74	125
LSODE	3.6e-05	2615	3920	156	156	5	9.9e-03		6.1e-01	2615	3920
DC03A	2.6e-05	722	828	42	1	5	9.7e-01		1.5e-01	722	828

```
Stiff Set No.  B5 Tolerance= 1.0e-06   System Size=4   Diffed. J's = f
Code     Error  Step    NFE   NG   NJ K  EndH     EndTol     Time  NBDF BSOL
STRUT 1.8e-05   453     849    8    1 4 1.0e+00
LSODA 9.7e-06   633    1247    8    8 4 7.0e-01           9.3e-02   73  140
LSODE 3.6e-05  2621    2997  158  158 5 6.8e-03           9.2e-01   74  125
DC03A 2.6e-05   722     833   42    1 5 9.7e-01           5.9e-01 2621 2997
                                                          1.5e-01  722  833
```

Another interesting pair of problems is C1, representing moderate coupling from fast to smooth components, and C5, representing strong coupling from smooth components to fast.

```
Stiff Set No.  C1 Tolerance= 1.0e-04   System Size=4   Diffed. J's = t
Code     Error  Step   NFE   NG   NJ K  EndH     EndTol     Time  NBDF BSOL
STRUT 2.0e-04    76    123    9    1 0 6.4e+00
LSODA 6.3e-05   118    212   10   10 2 1.1e+01           2.5e-02   47   93
LSODE 1.5e-04   100    195   19   19 3 3.6e+00           2.3e-01   43   55
DC03A 7.2e-05   110    192   18    1 3 2.2e+00           2.2e-02  100  195
                                                         2.7e-02  110  192
```

```
Stiff Set No.  C1 Tolerance= 1.0e-04   System Size=4   Diffed. J's = f
Code     Error  Step   NFE   NG   NJ K  EndH     EndTol     Time  NBDF BSOL
STRUT 2.0e-04    76    119    9    1 0 6.4e+00
LSODA 6.3e-05   118    212   10   10 2 1.1e+01           2.5e-02   47   93
LSODE 1.5e-04   100    119   19   19 3 3.6e+00           2.4e-01   43   55
DC03A 7.2e-05   110    197   18    1 3 2.2e+00           2.0e-02  100  119
                                                         2.7e-02  110  197
```

```
Stiff Set No.  C1 Tolerance= 1.0e-06   System Size=4   Diffed. J's = t
Code     Error  Step   NFE   NG   NJ K  EndH     EndTol     Time  NBDF BSOL
STRUT 2.8e-06   155    275   10    1 0 5.1e+00
LSODA 1.2e-06   220    422   10   10 5 1.4e+00           4.6e-02   67  132
LSODE 4.5e-06   178    302   24   24 2 2.5e+00           4.0e-01   63   98
DC03A 4.1e-06   186    258   18    1 5 9.6e-01           3.7e-02  178  302
                                                         3.4e-02  186  258
```

```
Stiff Set No.  C1 Tolerance= 1.0e-06   System Size=4   Diffed. J's = f
Code     Error  Step   NFE   NG   NJ K  EndH     EndTol     Time  NBDF BSOL
STRUT 2.8e-06   155    271   10    1 0 5.1e+00
LSODA 1.2e-06   220    422   10   10 5 1.4e+00           4.6e-02   67  132
LSODE 4.5e-06   178    206   24   24 2 2.5e+00           3.9e-01   63   98
DC03A 4.1e-06   186    263   18    1 5 9.6e-01           3.4e-02  178  206
                                                         3.4e-02  186  263
```

Stiff Set No. C5 Tolerance= 1.0e-04 System Size=4 Diffed. J's = t

Code	Error	Step	NFE	NG	NJ	K	EndH	EndTol	Time	NBDF	BSOL
STRUT	1.8e-03	319	469	14	10	0	2.8e+00	1.2e-03	1.0e-01	288	410
LSODA	1.9e-03	352	518	40	40	3	5.6e-01		5.3e-01	321	453
LSODE	7.9e-04	285	480	28	28	5	4.5e-01		5.9e-02	285	480
DC03A	6.5e-04	332	470	55	4	5	7.2e-01		6.7e-02	332	470

Stiff Set No. C5 Tolerance= 1.0e-04 System Size=4 Diffed. J's = f

Code	Error	Step	NFE	NG	NJ	K	EndH	EndTol	Time	NBDF	BSOL
STRUT	1.3e-02	323	463	21	11	0	3.0e+00	1.4e-02	1.1e-01	292	461
LSODA	1.9e-03	352	518	40	40	3	5.6e-01		5.3e-01	321	453
LSODE	7.9e-04	284	366	30	30	4	5.6e-01		5.6e-02	284	366
DC03A	6.5e-04	332	490	55	4	5	7.2e-01		6.7e-02	332	490

Stiff Set No. C5 Tolerance= 1.0e-06 System Size=4 Diffed. J's = t

Code	Error	Step	NFE	NG	NJ	K	EndH	EndTol	Time	NBDF	BSOL
STRUT	2.4e-05	731	1039	32	13	3	3.9e-01	2.4e-06	2.3e-01	657	929
LSODA	1.5e-05	591	931	49	49	3	2.4e-01		8.4e-01	533	808
LSODE	3.1e-05	575	968	62	62	4	2.6e-01		1.2e-01	575	968
DC03A	1.2e-05	588	891	61	3	4	2.4e-01		1.2e-01	588	891

Stiff Set No. C5 Tolerance= 1.0e-06 System Size=4 Diffed. J's = f

Code	Error	Step	NFE	NG	NJ	K	EndH	EndTol	Time	NBDF	BSOL
STRUT	3.0e-05	672	933	32	11	2	4.2e-01	2.3e-05	2.2e-01	598	881
LSODA	1.5e-05	591	931	49	49	3	2.4e-01		8.4e-01	533	808
LSODE	3.1e-05	565	712	63	63	4	4.4e-01		1.1e-01	565	712
DC03A	1.2e-05	589	896	64	3	5	2.9e-01		1.2e-01	589	896

D3 and D4 represent nonlinear systems with real eigenvalues. The BDF and SIBDF do not have significant stability limitations as evidenced by ending step size. The convergence tests of the BDF will be exercised. The linear stability theory of the SIBDF is tested on nonlinear problems. D3 has a change in dominant eigenvalue from -40,000 to -290. D4 has the dominant eigenvalue change only from $-3.5*10^3$ to $-3.8*10^3$.

Stiff Set No. D3 Tolerance= 1.0e-04 System Size=4 Diffed. J's = t

Code	Error	Step	NFE	NG	NJ	K	EndH	EndTol	Time	NBDF	BSOL
STRUT	1.4e-04	95	162	10	1	0	5.5e+01		2.6e-02	41	82
LSODA	8.0e-05	108	181	9	9	2	5.2e+00		2.7e-01	41	46
LSODE	2.6e-04	120	254	27	27	2	7.0e+00		2.5e-02	120	254
DC03A	1.3e-04	133	196	17	2	1	1.2e+01		2.3e-02	133	196

Stiff Set No. D3 Tolerance= 1.0e-04 System Size=4 Diffed. J's = f

Code	Error	Step	NFE	NG	NJ	K	EndH	EndTol	Time	NBDF	BSOL
STRUT	1.4e-04	95	158	10	1	0	5.5e+01		2.6e-02	41	82
LSODA	8.0e-05	108	181	9	9	2	5.2e+00		2.7e-01	41	46
LSODE	2.6e-04	120	146	27	27	2	7.0e+00		2.3e-02	120	146
DC03A	1.3e-04	138	211	17	2	2	2.9e+00		2.4e-02	138	211

Stiff Set No. D3 Tolerance= 1.0e-06 System Size=4 Diffed. J's = t

Code	Error	Step	NFE	NG	NJ	K	EndH	EndTol	Time	NBDF	BSOL
STRUT	2.4e-06	231	373	13	2	0	8.0e+00		6.2e-02	123	204
LSODA	1.9e-06	207	358	13	13	1	1.1e+01		5.0e-01	85	113
LSODE	3.4e-06	242	460	39	39	2	3.2e+00		5.0e-02	242	460
DC03A	2.9e-06	230	337	42	3	1	4.7e+00		4.6e-02	230	337

Stiff Set No. D3 Tolerance= 1.0e-06 System Size=4 Diffed. J's = f

Code	Error	Step	NFE	NG	NJ	K	EndH	EndTol	Time	NBDF	BSOL
STRUT	2.4e-06	231	365	13	2	0	8.0e+00		6.1e-02	123	204
LSODA	1.9e-06	207	358	13	13	1	1.1e+01		5.0e-01	85	113
LSODE	3.4e-06	242	304	39	39	2	3.2e+00		4.7e-02	242	304
DC03A	1.8e-06	232	348	40	3	3	1.4e+00		4.6e-02	232	348

Stiff Set No. D4 Tolerance= 1.0e-04 System Size=3 Diffed. J's = t

Code	Error	Step	NFE	NG	NJ	K	EndH	EndTol	Time	NBDF	BSOL
STRUT	4.6e-04	34	73	9	3	3	1.0e+01		1.3e-02	20	56
LSODA	5.7e-05	37	61	6	6	4	1.2e+01		1.1e-01	16	18
LSODE	1.3e-04	16	36	6	6	4	1.2e+01		3.2e-03	16	36
DC03A	1.8e-04	19	35	8	2	4	9.3e+00		4.6e-03	19	35

Stiff Set No. D4 Tolerance= 1.0e-04 System Size=3 Diffed. J's = f

Code	Error	Step	NFE	NG	NJ	K	EndH	EndTol	Time	NBDF	BSOL
STRUT	4.6e-04	34	64	9	3	3	1.0e+01		1.2e-02	20	56
LSODA	5.7e-05	37	61	6	6	4	1.2e+01		1.1e-01	16	18
LSODE	1.3e-04	16	18	6	6	4	1.2e+01		2.8e-03	16	18
DC03A	1.8e-04	19	43	8	2	4	9.3e+00		4.8e-03	19	43

Stiff Set No. D4 Tolerance= 1.0e-06 System Size=3 Diffed. J's = t

Code	Error	Step	NFE	NG	NJ	K	EndH	EndTol	Time	NBDF	BSOL
STRUT	2.3e-06	42	72	10	3	2	3.4e+00		1.5e-02	32	73
LSODA	2.1e-06	47	78	6	6	4	4.6e+00		1.2e-01	26	35
LSODE	1.7e-06	34	73	10	10	4	4.7e+00		6.5e-03	34	73
DC03A	1.4e-06	32	48	10	1	4	4.9e+00		6.3e-03	32	48

Stiff Set No. D4 Tolerance= 1.0e-06 System Size=3 Diffed. J's = f

Code	Error	Step	NFE	NG	NJ	K	EndH	EndTol	Time	NBDF	BSOL
STRUT	2.De-06	42	63	10	3	2	3.4e+00		1.5e-02	32	73
LSODA	2.1e-06	47	78	6	6	4	4.6e+00		1.2e-01	26	35
LSODE	1.7e-06	34	43	10	10	4	4.7e+00		6.0e-03	34	43
DC03A	1.4e-06	32	52	10	1	4	4.9e+00		6.4e-03	32	52

E3 and E4 represents a nonlinear system whose eigenvalues have both real and imaginary parts. The size of the imaginary part in E3 is only $.001 \pm .01i$, which is not the dominant eigenvalue. E4 is a modified version of a problem due to Krogh. Here the dominant eigenvalue is -10^3 and the complex one ranges from $8 \pm 10i$ to $.19 \pm 29i$ to $10 \pm 10i$ in the range of the integration.

Stiff Set No. E3 Tolerance= 1.0e-04 System Size=3 Diff'd J's = t

Code	Error	Step	NFE	NG	NJ	K	EndH	EndTol	Time	NBDF	BSOL
STRUT	6.3e-04	94	201	20	8	4	2.7e+01	1.0e-04	3.6e-02	77	233
LSODA	1.3e-03	122	187	16	16	5	1.8e+01	1.0e-04	1.7e-01	95	132
LSODE	6.0e-04	98	180	17	17	4	1.7e+01	1.0e-04	1.6e-02	98	180
DC03A	1.7e-03	114	195	26	2	4	2.1e+01	1.0e-04	2.1e-02	114	195

Stiff Set No. E3 Tolerance= 1.0e-04 System Size=3 Diff'd J's = f

Code	Error	Step	NFE	NG	NJ	K	EndH	EndTol	Time	NBDF	BSOL
STRUT	7.3e-04	83	158	19	9	4	1.9e+01	1.0e-04	3.2e-02	66	210
LSODA	1.3e-03	122	187	16	16	5	1.8e+01	1.0e-04	1.7e-01	95	132
LSODE	6.0e-04	98	129	17	17	4	1.7e+01	1.0e-04	1.5e-02	98	129
DC03A	1.7e-03	114	203	26	2	4	2.1e+01	1.0e-04	2.1e-02	114	203

Stiff Set No. E4 Tolerance= 1.0e-04 System Size=4 Diff'd J's = t

Code	Error	Step	NFE	NG	NJ	K	EndH	EndTol	Time	NBDF	BSOL
STRUT	1.1e-03	255	389	25	7	0	3.0e+02	1.0e-04	8.2e-02	213	389
LSODA	7.5e-04	239	324	28	28	3	1.5e+02	1.0e-04	7.0e-01	196	237
LSODE	3.0e-03	223	410	34	34	3	1.9e+02	1.0e-04	4.0e-02	223	410
DC03A	1.9e-03	234	354	49	5	1	2.5e+02	1.0e-04	4.7e-02	234	549

Stiff Set No. E4 Tolerance= 1.0e-04 System Size=4 Diff'd J's = f

Code	Error	Step	NFE	NG	NJ	K	EndH	EndTol	Time	NBDF	BSOL
STRUT	1.1e-03	254	361	25	7	0	3.0e+02	1.0e-04	8.2e-02	212	389
LSODA	7.5e-04	239	324	28	28	3	1.5e+02	1.0e-04	6.9e-01	196	237
LSODE	3.0e-03	223	274	34	34	3	1.9e+02	1.0e-04	3.9e-02	223	274
DC03A	1.9e-03	234	378	49	5	1	3.0e+02	1.0e-04	4.6e-02	234	581

Stiff Set No. E3 Tolerance= 1.0e-06 System Size=3 Diff'd J's = t

Code	Error	Step	NFE	NG	NJ	K	EndH	EndTol	Time	NBDF	BSOL
STRUT	1.1e-05	153	301	23	11	5	1.6e+01	1.5e-06	5.2e-02	116	295
LSODA	2.2e-05	228	374	27	27	5	1.3e+01	1.0e-06	3.1e-01	177	267
LSODE	2.1e-05	236	411	34	34	4	6.9e+00	1.0e-06	3.9e-02	236	411
DC03A	3.0e-05	188	298	34	2	5	1.2e+01	1.0e-06	3.3e-02	188	298

Stiff Set No. E3 Tolerance= 1.0e-06 System Size=3 Diff'd J's = f

Code	Error	Step	NFE	NG	NJ	K	EndH	EndTol	Time	NBDF	BSOL
STRUT	1.1e-05	155	273	24	13	5	1.8e+01	1.5e-06	5.2e-02	118	306
LSODA	2.2e-05	228	374	27	27	5	1.3e+01	1.0e-06	3.0e-01	177	267
LSODE	2.1e-05	236	309	34	34	4	6.9e+00	1.0e-06	3.7e-02	236	309
DC03A	3.0e-05	188	306	34	2	5	1.2e+01	1.0e-06	3.3e-02	188	306

Stiff Set No. E4 Tolerance= 1.0e-06 System Size=4 Diff'd J's = t

Code	Error	Step	NFE	NG	NJ	K	EndH	EndTol	Time	NBDF	BSOL
STRUT	5.4e-05	572	825	28	7	2	7.2e+01	1.0e-06	1.9e-01	533	974
LSODA	1.1e-05	981	1988	30	30	3	1.1e+02	1.0e-06	2.4e+00	185	253
LSODE	2.5e-05	476	795	59	59	3	1.0e+02	1.0e-06	8.3e-02	476	795
DC03A	5.4e-05	438	660	55	4	5	8.2e+01	1.0e-06	7.9e-02	438	958

Stiff Set No. E4 Tolerance= 1.0e-06 System Size=4 Diff'd J's = f

Code	Error	Step	NFE	NG	NJ	K	EndH	EndTol	Time	NBDF	BSOL
STRUT	5.7e-05	596	786	31	8	0	6.6e+01	1.0e-06	1.9e-01	557	951
LSODA	1.1e-05	981	1988	30	30	3	1.1e+02	1.0e-06	2.3e+00	185	253
LSODE	2.4e-05	476	560	59	59	3	1.0e+02	1.0e-06	8.1e-02	476	560
DC03A	5.4e-05	439	678	56	4	5	6.5e+01	1.0e-06	7.9e-02	439	984

Problems from the Non-Stiff Test Test

We now examine performance on some problems from the non-stiff test set. All the BDF codes are using differenced Jacobians. We present the results of the switching codes STRUT and LSODA which begin the integration using the Adams method. Nothing was done to prevent them from changing to BDF based on their internal logic.

LSODE was run initially as an Adams code. The Harwell code DC03A is only a BDF, so it was decided to examine the performance of LSODE as a BDF, STRUT forced to use BDF after 3 steps and DC03A. The cost is higher when using a code intended for stiff problems on non-stiff problems. We felt this would exercise many of the assumptions made in writing STRUT which applied the $(h\to\infty)$ AAS results to yield algorithms that would have to work on problems with small step size.

A5 represents a scalar problem but the Jacobian is not autonomous. This could pose a problem for the matrix error estimator in STRUT which estimates error in the Jacobian by changes in f assuming that f depends only on y.

Non-Stiff Set No. A5 Tolerance= 1.0e-04 System Size=1 Diffed. J's=t

Code	Error	Step	NFE	NG	NJ	K	EndH	EndTol	Time	NBDF	BSOL
STRUT	1.1e-05	34	70	0	0	8	2.3e+00		5.4e-03	0	0
LSODA	3.4e-04	32	69	0	0	4	1.6e+00		2.5e-02	32	0
LSODE	1.4e-04	31	34	0	0	6	1.8e+00		2.7e-03	31	34
Run as BDF Codes:											
STRUT	1.4e-03	44	79	4	1	4	9.3e-01		1.1e-02	40	101
LSODE	3.0e-04	45	65	9	9	4	9.8e-01		4.4e-03	45	65
DC03A	8.3e-04	46	60	0	0	4	9.5e-01		4.0e-03	46	60

```
Non-Stiff Set No. A5 Tolerance=1.0e-06   System Size= 1  Diffed. J's=t
 Code      Error  Step    NFE   NG   NJ  K  EndH      EndTol    Time   NBDF  BSOL
 STRUT 1.0e-06     54    110    0    0  8  1.0e+00            7.8e-03     0     0
 LSODA 2.6e-05     60    125    0    0  4  4.9e-01            4.0e-02    60     0
 LSODE 1.2e-06     53     58    0    0  7  7.2e-01            4.6e-03    53    58

 Run as BDF Codes:
 STRUT 3.0e-05     73    117    6    1  4  4.0e-01            1.8e-02    69   139
 LSODE 2.7e-05     80    104   10   10  5  4.7e-01            7.2e-03    80   104
 DC03A 3.3e-05     84    104    0    0  5  4.8e-01            6.7e-03    84   104
```

C3 and C4 represent the discretization of the heat equation using finite differences with grid sizes of 10 and 51 points, respectively. This yields a problem with a smooth variation in the spectrum of eigenvalues, not isolated large or small eigenvalues. This should be a tough test on the matrix error estimator using power method results for only the largest and smallest eigenvalues.

```
Non-Stiff Set No.  C3 Tolerance=1.0e-04  System Size=10  Diffed. J's=t
Code      Error  Step    NFE   NG   NJ  K  EndH      EndTol    Time   NBDF  BSOL
STRUT 2.6e-04     39     91    3    1  2  2.5e+00            1.7e-02    15    30
LSODA 1.1e-04     89    186   16   16  2  2.3e-01            7.7e-02     6    46
LSODE 4.2e-04    129    218    0    0  1  2.2e-01            1.5e-02   129   218

Run as BDF Codes:
STRUT 3.5e-04     40     64    4    1  2  2.8e+00            2.4e-02    36    63
LSODE 2.2e-04     51    161   10   10  4  2.2e+00            2.0e-02    51   161
DC03A 2.6e-04     59     88   13    1  4  2.0e+00            2.5e-02    59    88
```

```
Non-Stiff Set No.  C3 Tolerance=1.0e-06  System Size=10  Diffed. J's=t
Code      Error  Step    NFE   NG   NJ  K  EndH      EndTol    Time   NBDF  BSOL
STRUT 2.7e-06     80    168    3    1  3  1.2e+00            2.5e-02    19    39
LSODA 1.6e-06    109    232    0    0  2  3.0e-01            1.0e-01   109     0
LSODE 1.4e-05    206    222    0    0  2  1.3e-01            2.2e-02   206   222

Run as BDF Codes:
STRUT 3.2e-06     83    107    6    1  3  1.3e+00            4.8e-02    79   141
LSODE 5.3e-06     89    245   14   14  5  1.3e+00            3.4e-02    89   245
DC03A 2.7e-06    100    136   19    1  5  1.3e+00            4.0e-02   100   136
```

```
Non-Stiff Set No.  C4 Tolerance=1.0e-04  System Size=51  Diffed. J's=t
Code      Error  Step   NFE  NG  NJ K  EndH    EndTol   Time   NBDF BSOL
STRUT 5.5e-04    35    125   3   1 2 2.1e+00          6.9e-02   15   28
LSODA 1.1e-04    89    180  16  16 2 1.7e+00          3.9e-01    1   40
LSODE 8.8e-04   109    195   0   0 1 1.3e+00          2.7e-02  109  195

Run as BDF Codes:
STRUT 5.8e-04    36    101   5   1 2 4.0e+00          1.2e-01   32   59
LSODE 4.3e-04    45    512   9   9 4 2.7e+00          1.2e-01   45  512
DC03A 2.6e-04    59    129  13   1 4 2.0e+00          2.0e-01   59  217
```

```
Non-Stiff Set No.  C4 Tolerance=1.0e-06  System Size=51  Diffed. J's=t
Code      Error  Step   NFE  NG  NJ K  EndH    EndTol   Time   NBDF BSOL
STRUT 6.2e-06    65    190   3   1 3 1.5e+00          9.7e-02   21   43
LSODA 1.6e-06   116    238   9   9 2 1.8e-01          3.6e-01   17   26
LSODE 3.8e-05   433    487   0   0 4 4.7e-02          1.4e-01  433  487

Run as BDF Codes:
STRUT 7.1e-06    71    135   5   1 3 1.5e+00          1.9e-01   67  123
LSODE 7.6e-06    81    760  13  13 5 1.1e+00          2.0e-01   81  760
DC03A 2.7e-06   100    177  19   1 5 1.3e+00          3.1e-01  100  313
```

D3 is the middle range orbit problem. D4 is a slightly more eccentric orbit. Orbit problems are good for testing the step size and order selection algorithms of a variable order, variable step size code.

```
Non-Stiff Set No. D3 Tolerance=1.0e-04   System Size=4  Diffed. J's=t
Code      Error  Step   NFE  NG  NJ K  EndH    EndTol   Time   NBDF BSOL
STRUT 8.8e-02   154    329   0   0 7 1.4e-01          2.3e-02    0    0
LSODA 4.5e-02   169    387   0   0 4 9.5e-02          8.7e-02  169    0
LSODE 2.6e-01   170    214   0   0 3 9.5e-02          1.7e-02  170  214

Run as BDF Codes:
STRUT 6.9e-02   177    255   4   3 4 9.2e-02          6.5e-02  173  350
LSODE 8.8e-02   220    432  34  34 4 7.1e-02          4.1e-02  220  432
DC03A 9.8e-02   240    360  42   2 5 7.5e-02          4.4e-02  240  360
```

Non-Stiff Set No. D3 Tolerance=1.0e-06 System Size=4 Diffed. J's=t

Code	Error	Step	NFE	NG	NJ	K	EndH	EndTol	Time	NBDF	BSOL
STRUT	3.1e-04	261	540	0	0	8	6.0e-02		3.8e-02	0	0
LSODA	1.4e-04	280	613	0	0	6	6.2e-02		1.4e-01	280	0
LSODE	1.3e-03	335	366	0	0	6	5.7e-02		3.2e-02	335	366
Run as BDF Codes:											
STRUT	4.8e-03	389	592	3	3	4	5.3e-02		1.4e-01	385	639
LSODE	2.3e-03	404	626	35	35	5	4.5e-02		6.8e-02	404	626
DC03A	1.2e-03	463	696	64	1	5	4.6e-02		8.4e-02	463	696

Non-Stiff Set No. D4 Tolerance=1.0e-04 System Size=4 Diffed. J's=t

Code	Error	Step	NFE	NG	NJ	K	EndH	EndTol	Time	NBDF	BSOL
STRUT	1.7e-01	210	444	0	0	7	8.5e-02		3.1e-02	0	0
LSODA	4.2e-02	225	524	0	0	4	9.9e-02		1.2e-01	225	0
LSODE	1.6e-01	213	275	0	0	4	9.4e-02		2.2e-02	213	275
Run as BDF Codes:											
STRUT	1.4e-01	236	376	9	8	2	5.0e-02		8.9e-02	232	478
LSODE	9.9e-02	292	600	51	51	4	9.2e-02		5.5e-02	292	600
DC03A	1.2e-01	322	488	63	2	5	1.1e-01		6.0e-02	322	848

Non-Stiff Set No. D4 Tolerance=1.0e-06 System Size=4 Diffed. J's=t

Code	Error	Step	NFE	NG	NJ	K	EndH	EndTol	Time	NBDF	BSOL
STRUT	7.9e-04	355	734	0	0	8	4.9e-02		5.2e-02	0	0
LSODA	7.6e-04	376	831	0	0	6	5.1e-02		1.9e-01	376	0
LSODE	2.5e-03	393	439	0	0	5	6.2e-02		4.0e-02	393	439
Run as BDF Codes:											
STRUT	9.0e-04	513	749	3	3	4	5.2e-02		1.8e-01	509	772
LSODE	3.7e-03	534	863	52	52	5	4.3e-02		9.3e-02	534	863
DC03A	2.4e-03	609	791	85	2	5	4.6e-02		1.0e-01	609	1487

E2 and E3 are examples of problems occurring originally as single second order equations; both are nonlinear. E2 is the Van der Pol equation. E3 is Duffing's equation and is one of the few nonautonomous systems in the problem set.

```
Non-Stiff Set No.  E2 Tolerance=1.0e-04  System Size=2  Diffed. J's=t
Code      Error   Step   NFE   NG   NJ K  EndH     EndTol    Time   NBDF BSOL
STRUT 2.2e-04   201    413    0    0 8 7.4e-02             2.6e-02    0    0
LSODA 4.8e-04   283    623    0    0 4 6.7e-02             1.0e-01  283    0
LSODE 1.0e-03   279    325    0    0 4 6.5e-02             2.3e-02  279  325

Run as BDF Codes:
STRUT 3.0e-03   294    487   12   12 3 5.4e-02             8.7e-02  290  639
LSODE 5.2e-04   308    464   32   32 5 5.3e-02             3.7e-02  308  464
DC03A 1.2e-03   331    458   61    1 5 6.4e-02             3.7e-02  331  458


Non-Stiff Set No.  E2 Tolerance=1.0e-06  System Size=2  Diffed. J's=t
Code      Error   Step   NFE   NG   NJ K  EndH     EndTol    Time   NBDF BSOL
STRUT 3.6e-06   346    703    0    0 8 4.3e-02             4.4e-02    0    0
LSODA 1.0e-05   434    943    0    0 5 4.4e-02             1.5e-01  434    0
LSODE 2.8e-05   495    544    0    0 6 4.8e-02             4.2e-02  495  544

Run as BDF Codes:
STRUT 7.3e-05   581    715    1    1 4 2.8e-02             1.5e-01  577  788
LSODE 1.8e-05   604    823   49   49 5 2.8e-02             6.8e-02  604  823
DC03A 4.0e-05   653    791   72    1 5 2.9e-02             6.7e-02  653  791


Non-Stiff Set No.  E3 Tolerance=1.0e-04  System Size=2  Diffed. J's=t
Code      Error   Step   NFE   NG   NJ K  EndH     EndTol    Time   NBDF BSOL
STRUT 8.2e-05   142    289    0    0 8 1.5e-01             1.9e-02    0    0
LSODA 1.5e-03   156    331    0    0 6 1.5e-01             9.0e-02  156    0
LSODE 3.4e-03   173    180    0    0 5 1.4e-01             1.5e-02  173  180

Run as BDF Codes:
STRUT 4.5e-03   201    330   22    4 4 1.0e-01             5.9e-02  197  414
LSODE 7.2e-04   207    266   16   16 5 1.1e-01             2.3e-02  207  266
DC03A 1.0e-03   224    268   25    1 5 1.1e-01             2.5e-02  224  726


Non-Stiff Set No.  E3 Tolerance=1.0e-06  System Size=2  Diffed. J's=t
Code      Error   Step   NFE   NG   NJ K  EndH     EndTol    Time   NBDF BSOL
STRUT 2.3e-06   220    445    0    0 7 9.4e-02             2.9e-02    0    0
LSODA 9.9e-06   218    449    0    0 8 1.0e-01             1.1e-01  218    0
LSODE 5.9e-05   306    321    0    0 6 7.0e-02             2.7e-02  306  321

Run as BDF Codes:
STRUT 2.0e-05   479    630    4    3 4 4.1e-02             1.3e-01  475  627
LSODE 1.6e-05   427    536   27   27 5 5.2e-02             4.8e-02  427  536
DC03A 1.8e-05   480    631   50    1 5 5.3e-02             5.4e-02  480 1422
```

Appendix B. μ-Plane Stability Plots for $h=\infty$

The following plots were presented originally in [35] and are repeated here for completeness. See Section 4.d for development and discussion.

Figure B1. Asymptotic ($h\to\infty$) Absolute Stability
1 Step Predictor (Order 0)
Stability Limitation in the μ-plane

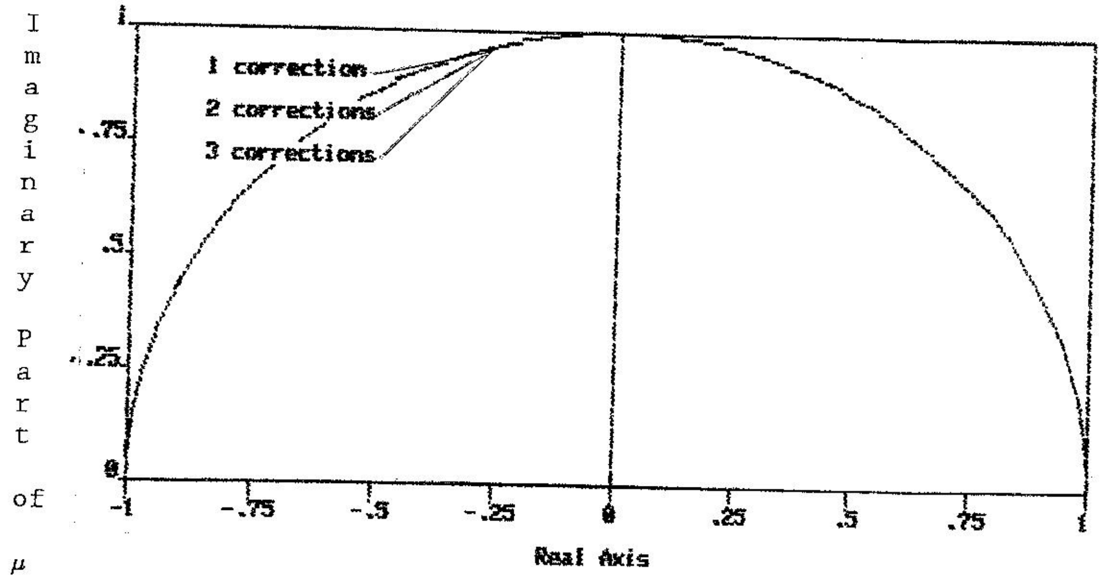


Figure B2. Asymptotic ($h\to\infty$) Absolute Stability
2 Step Predictor (Order 1)
Stability Limitation in the μ-plane

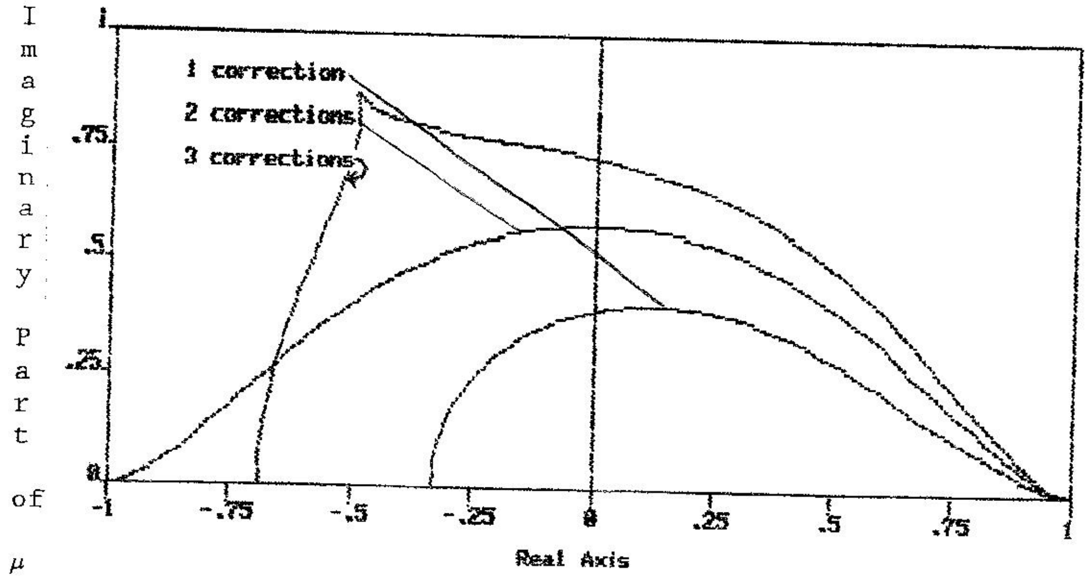


Figure B3. Asymptotic ($h \to \infty$) Absolute Stability
3 Step Predictor (Order 2)
Stability Limitation in the μ-plane

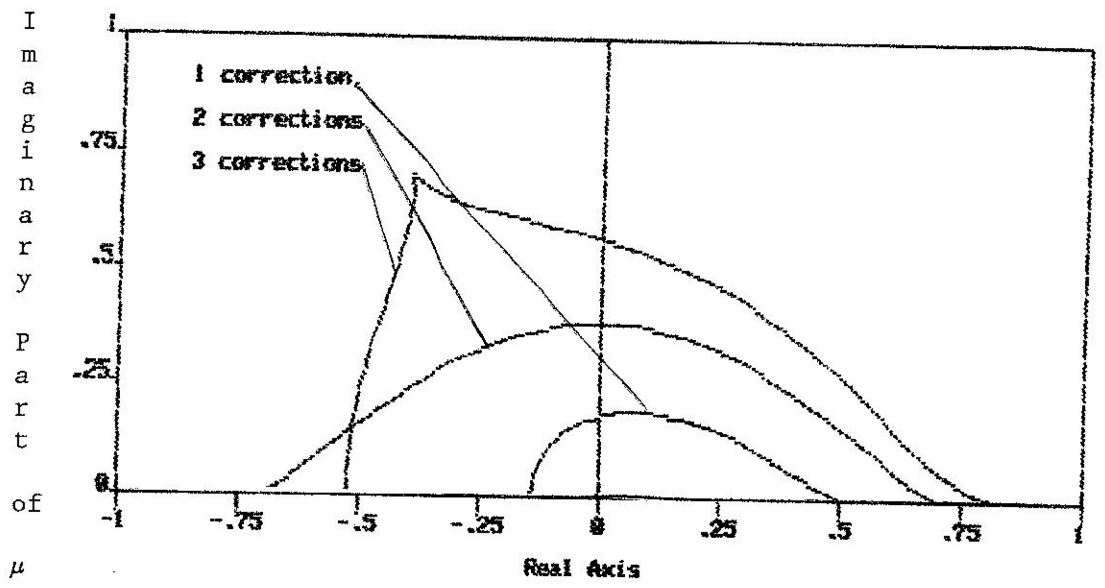


Figure B4. Asymptotic ($h \to \infty$) Absolute Stability
4 Step Predictor (Order 3)
Stability Limitation in the μ-plane

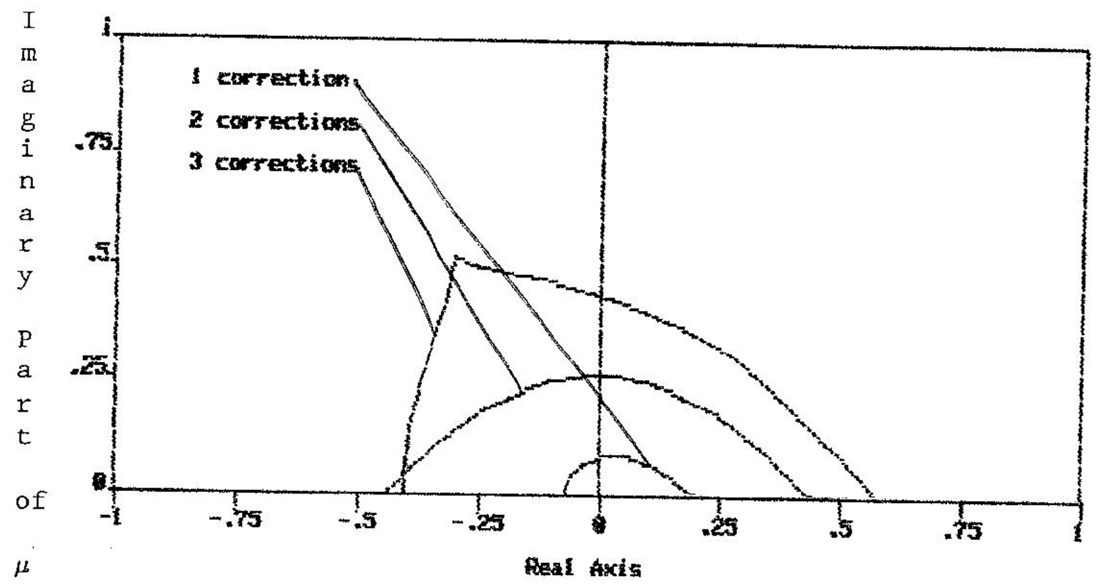


Figure B5. Asymptotic ($h \to \infty$) Absolute Stability
5 Step Predictor (Order 4)
Stability Limitation in the μ-plane

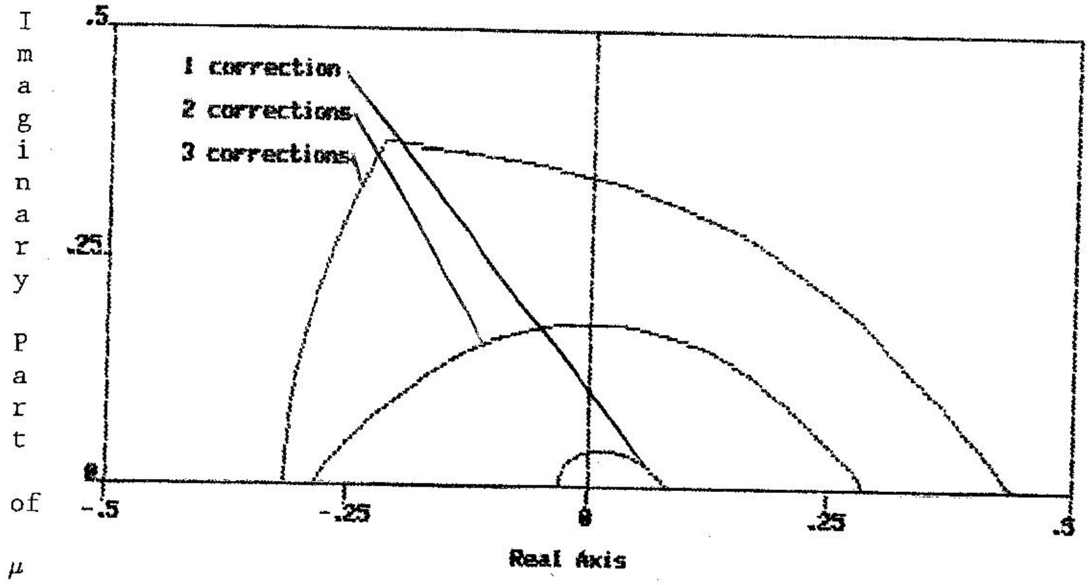
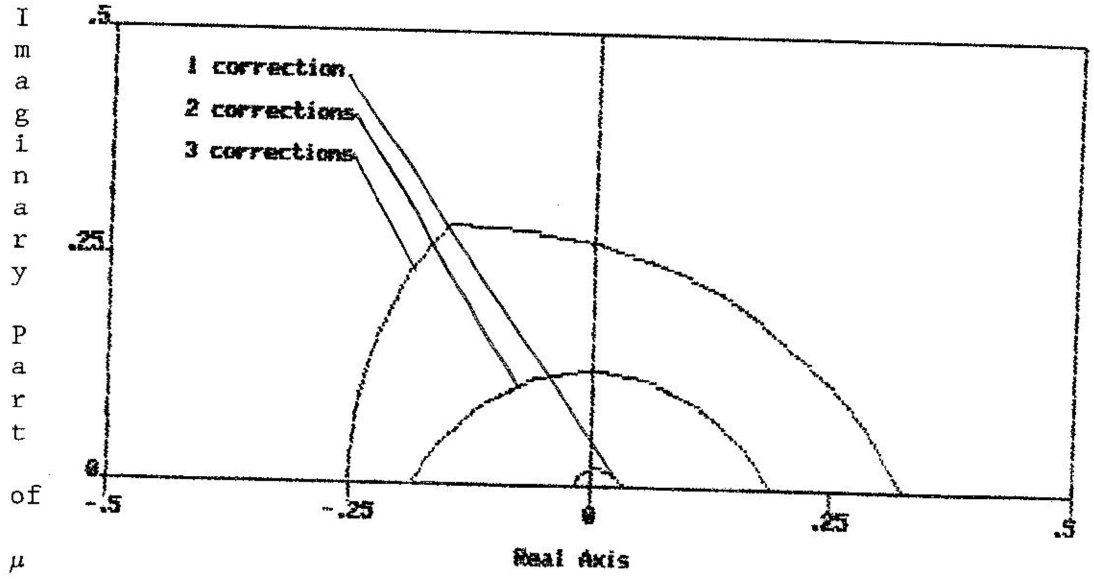


Figure B6. Asymptotic ($h \to \infty$) Absolute Stability
6 Step Predictor (Order 5)
Stability Limitation in the μ-plane



Appendix C. μ-Plane Stability Plots for Finite $h\lambda$

As discussed in Section 4.e, only the scalar generalized difference equation (SGDE) can be easily analyzed for the case of nonzero matrix error coupled with finite, nonzero values of the step size. Still, the behavior of Absolute Stability for the BDF is mimicked by the SGDE. For example, the lower order P_kEC_{k+1} methods contain the origin within the region of stability. The origin in the μ-plane corresponds to the BDF solved exactly, and the low order BDFs are A-stable.

We begin with the two step corrector (of order one) using the one step predictor (of order zero), P_1EC_2.

Figure C1. P_1EC_2 Scalar SIBDF Method with $|h\lambda|=1$ Stability Limitation in the μ-plane

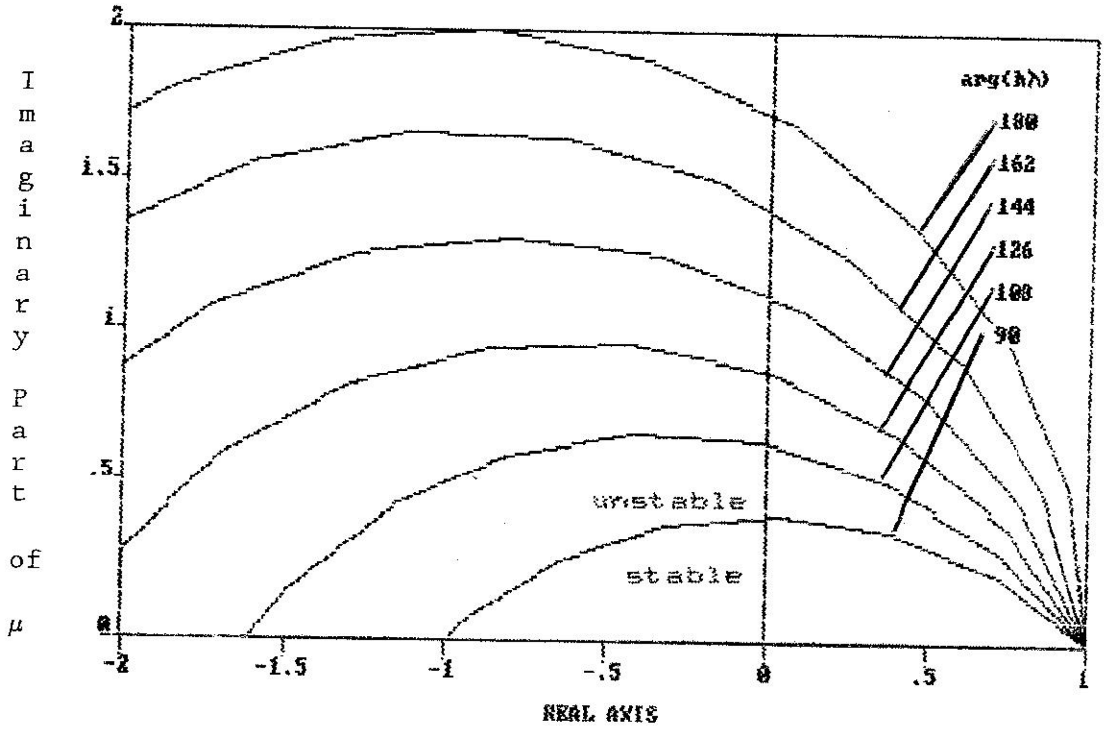


Figure C2. P_1EC_2 Scalar SIBDF Method with $|h\lambda|=5$
Stability Limitation in the μ-plane

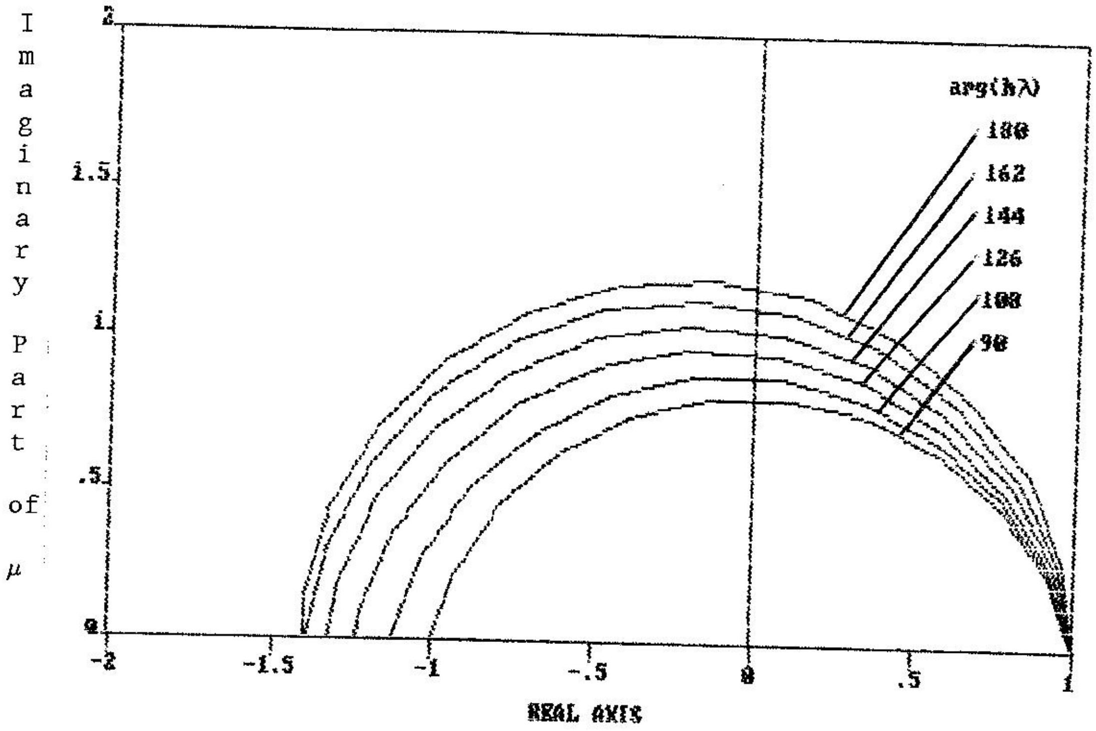
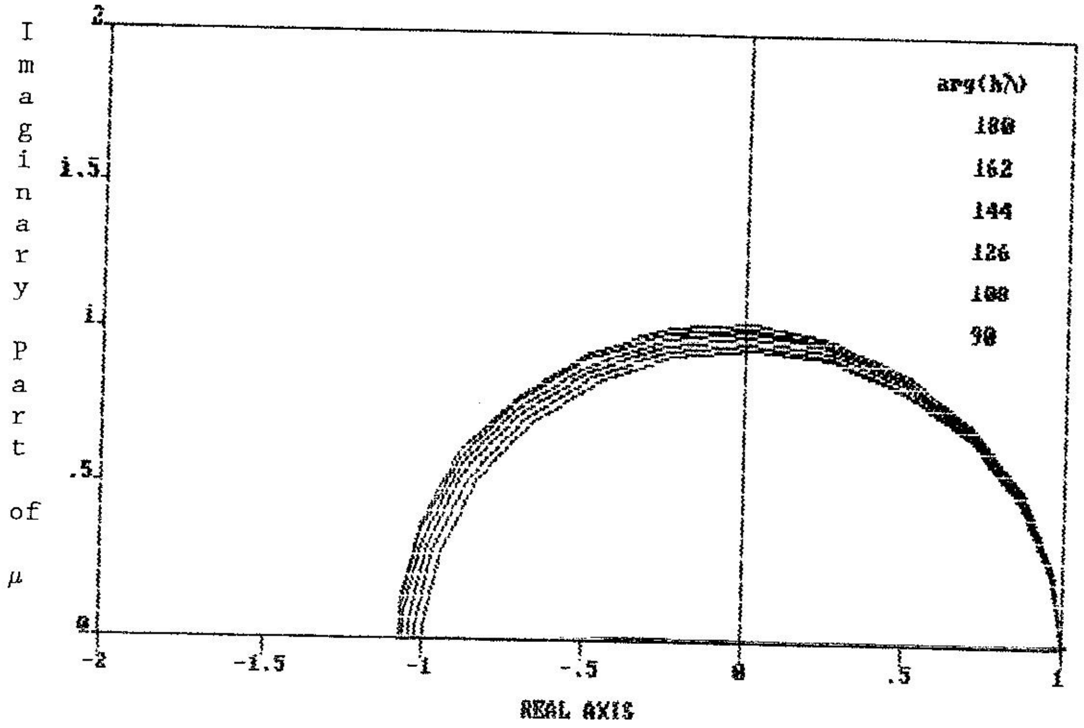


Figure C3. P_1EC_2 Scalar SIBDF Method with $|h\lambda|=25$
Stability Limitation in the μ-plane



The second order corrector, P_2EC_3, reveals stability when the corrector is solved exactly as predicted by the BDF analysis.

Figure C4. P_2EC_3 Scalar SIBDF Method with $|h\lambda|=1$ Stability Limitation in the μ-plane

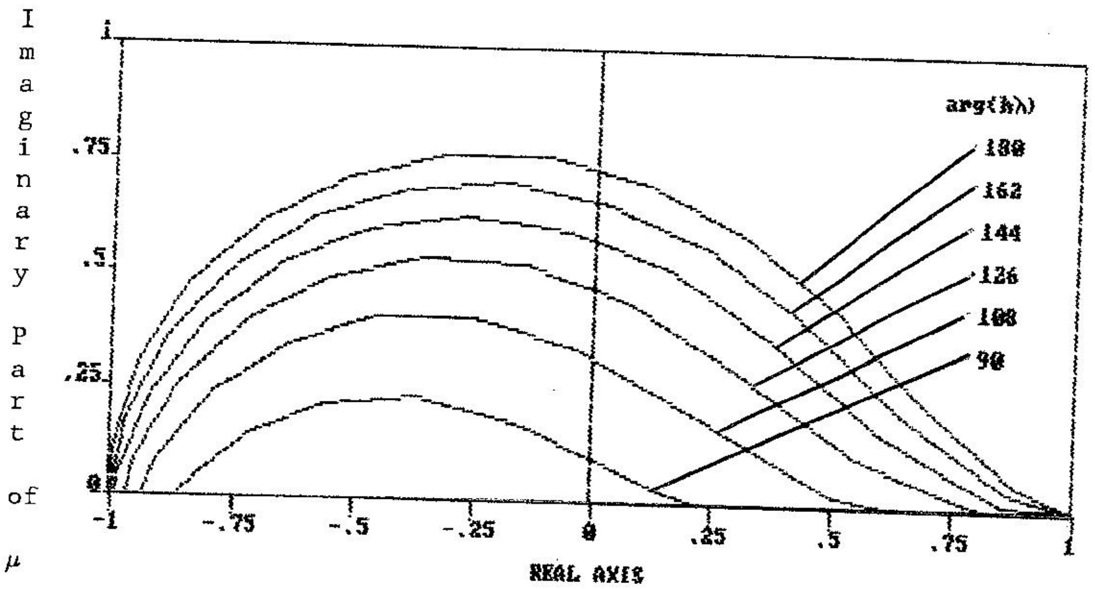


Figure C5. P_2EC_3 Scalar SIBDF Method with $|h\lambda|=5$ Stability Limitation in the μ-plane

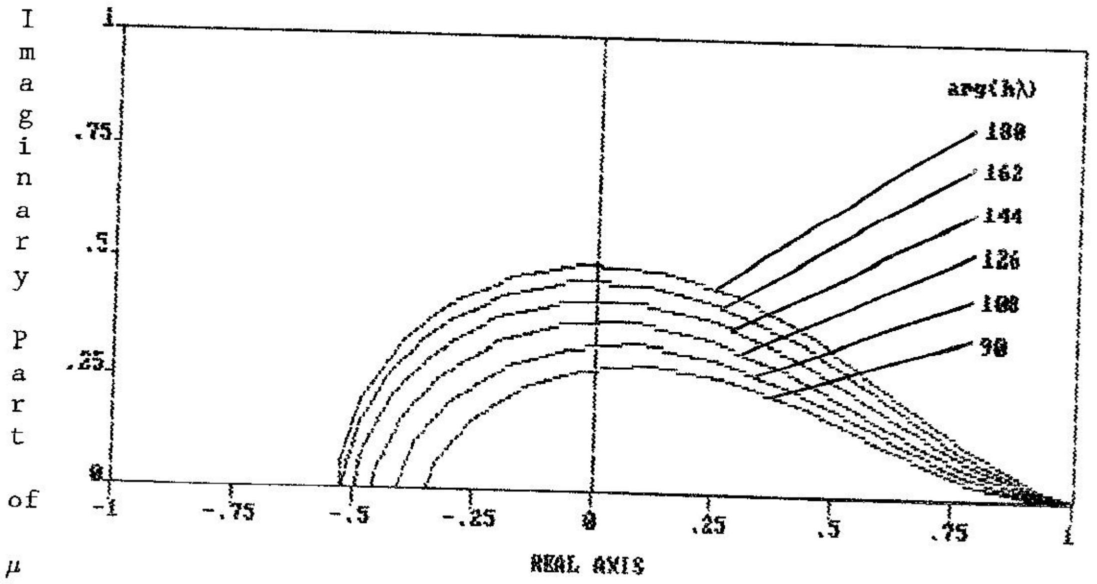
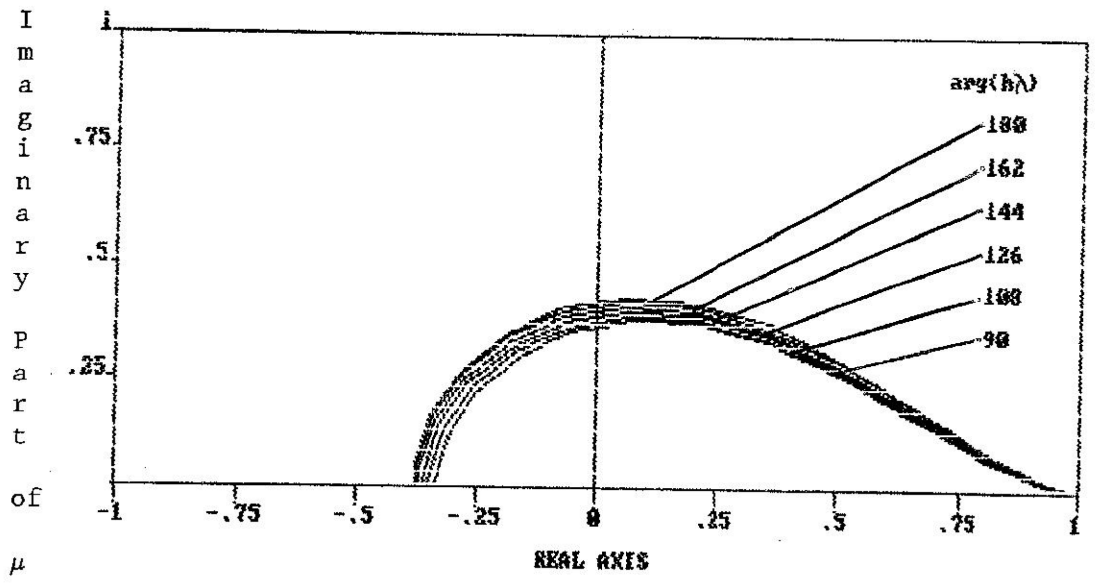


Figure C6. P_2EC_3 Scalar SIBDF Method with $|h\lambda|=25$
Stability Limitation in the μ-plane



With the P_3EC_4 method, we see for $|h\lambda|=1$, that the origin is not within the stability region when $\arg(\lambda)=90^o$, i.e. the purely imaginary eigenvalue. This corresponds to the lobe in the left half plane that is carved out for the third order BDF corrector.

Figure C7. P_3EC_4 Scalar SIBDF Method with $|h\lambda|=1$
Stability Limitation in the μ-plane

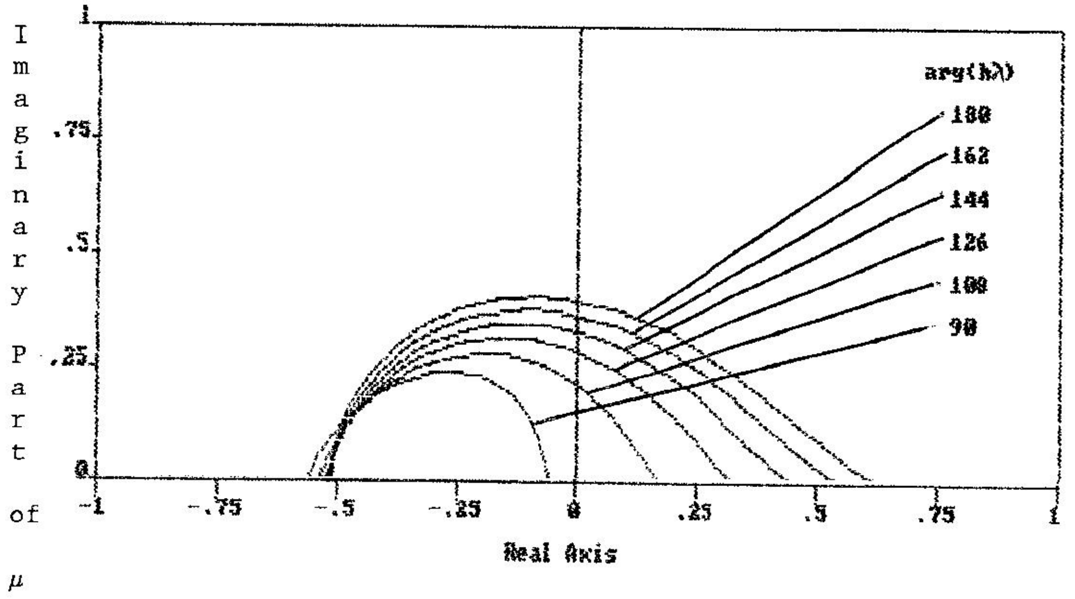


Figure C8. P_3EC_4 Scalar SIBDF Method with $|h\lambda|=5$
Stability Limitation in the μ-plane

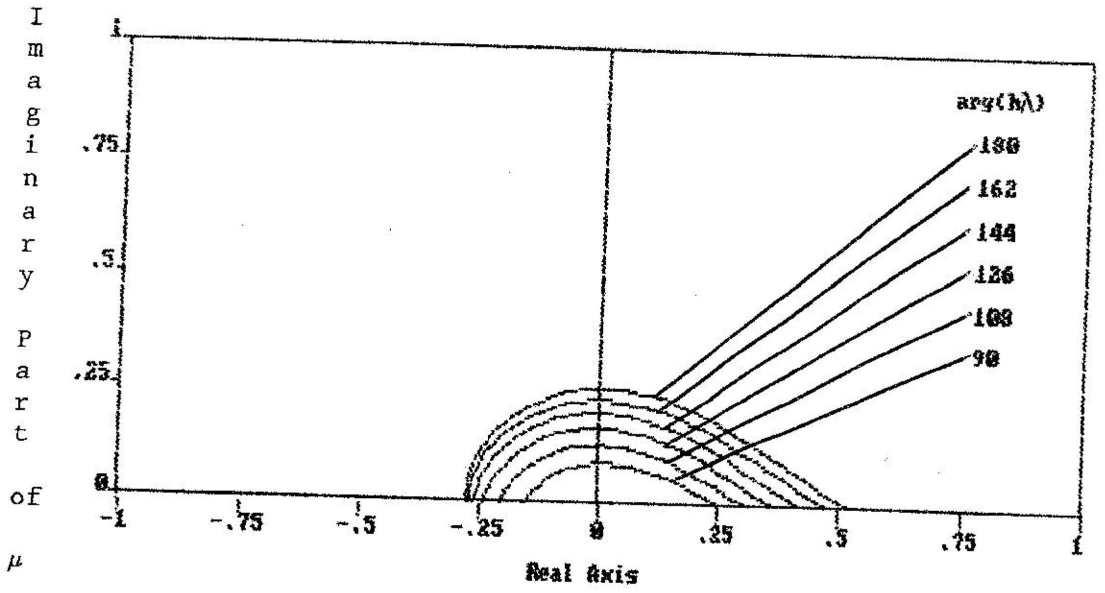
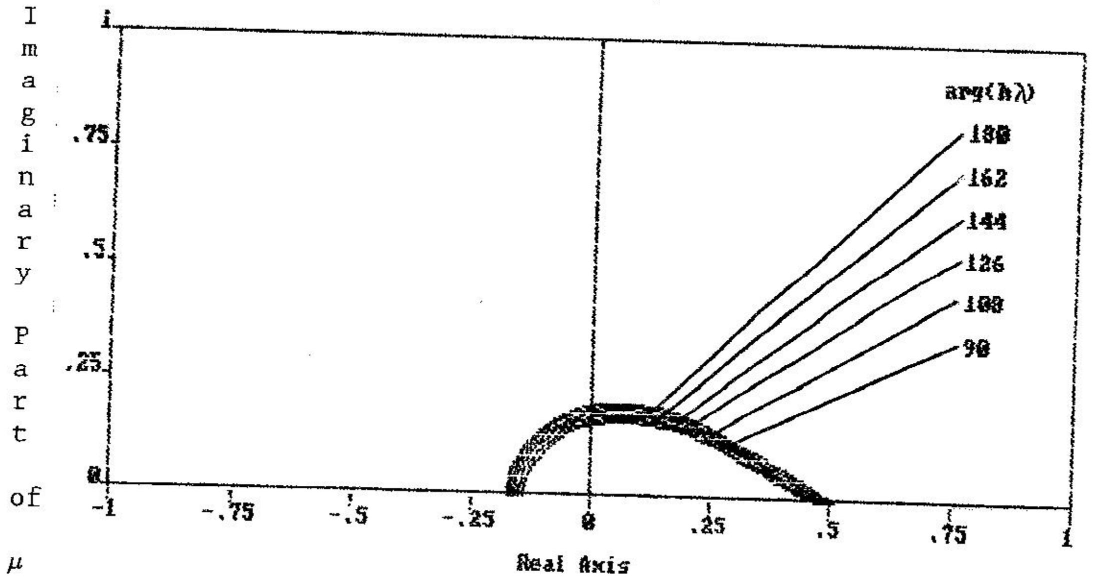


Figure C9. P_3EC_4 Scalar SIBDF Method with $|h\lambda|=25$
Stability Limitation in the μ-plane



For small $h\lambda$, when λ is purely imaginary, i.e. $|h\lambda| = 1$ and $\arg(h\lambda) = 90^{\circ}$, we have no region of stability at all with the fourth order corrector. As the order of the corrector continues to increase, the lack of μ-stability corresponding to small values of $h\lambda$ becomes more dramatic.

Figure C10. P_4EC_5 Scalar SIBDF Method with $|h\lambda|=1$
Stability Limitation in the μ-plane

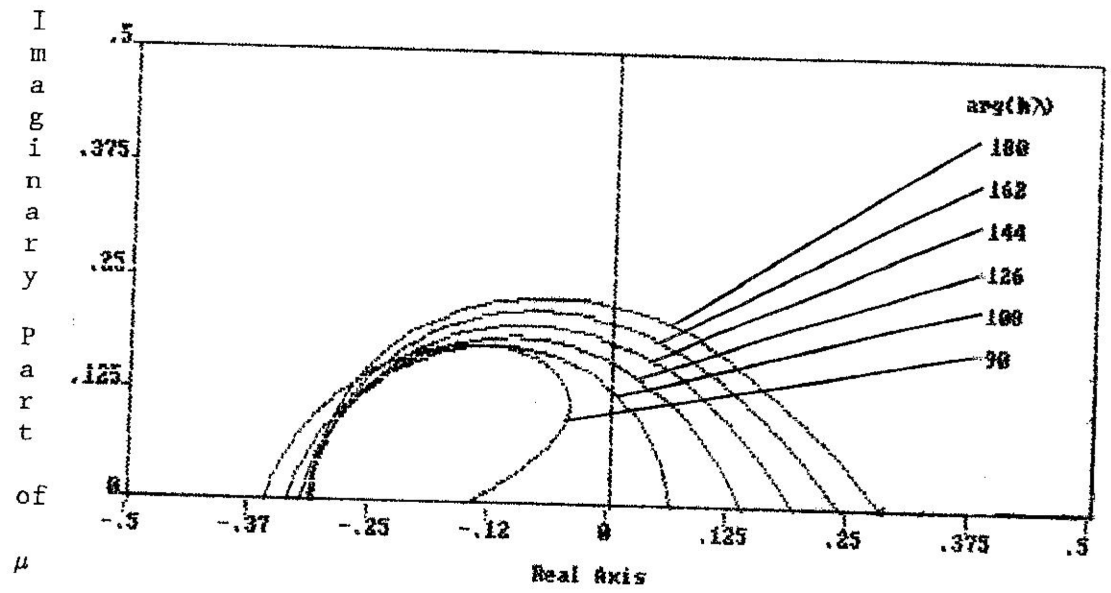


Figure C11. P_4EC_5 Scalar SIBDF Method with $|h\lambda|=5$

Stability Limitation in the μ-plane

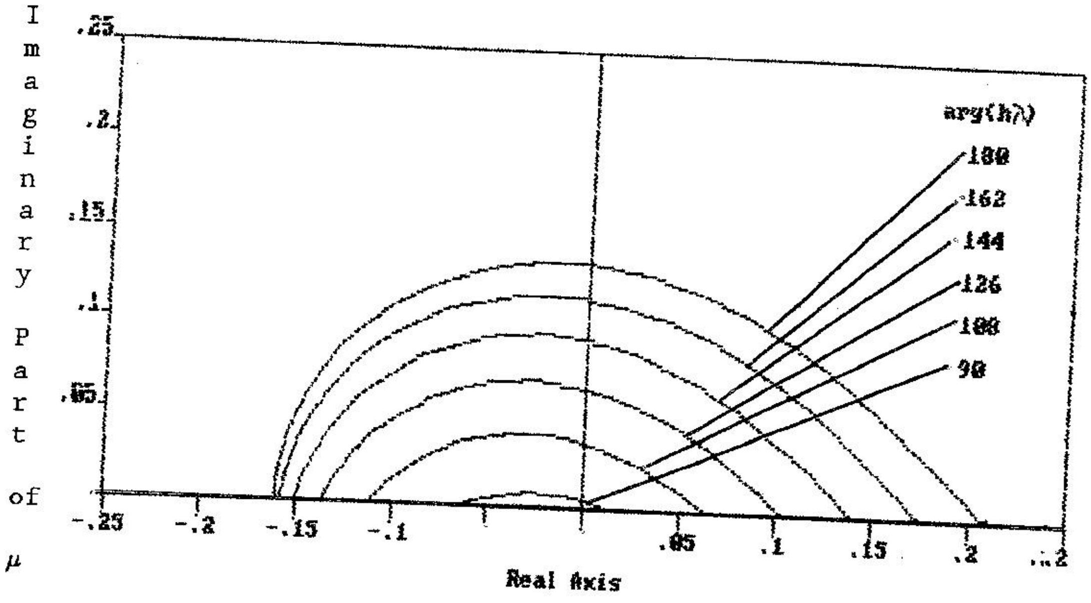
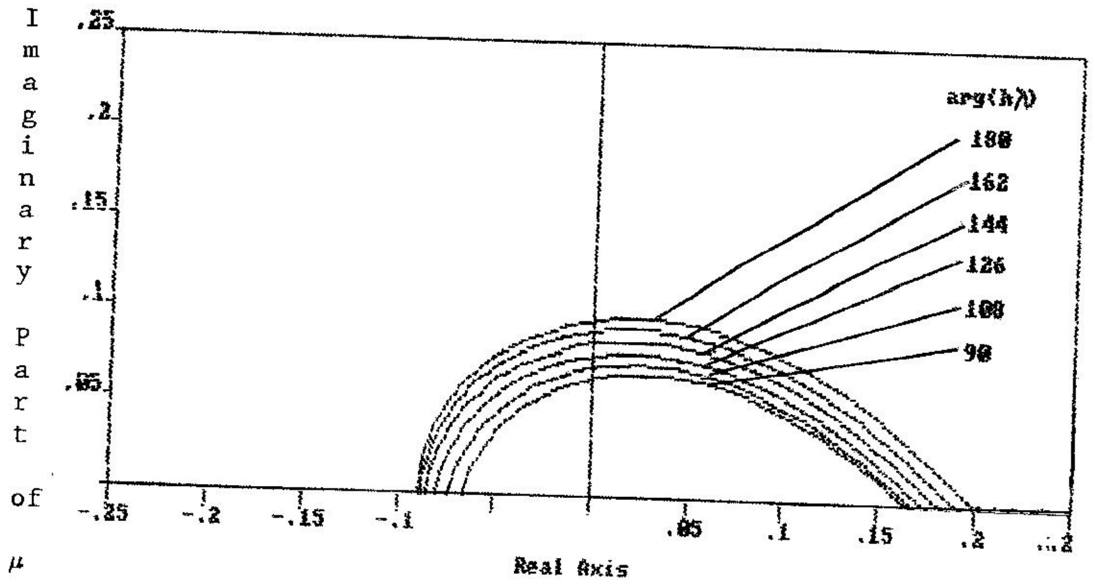


Figure C12. P_4EC_5 Scalar SIBDF Method with $|h\lambda|=25$

Stability Limitation in the μ-plane



For small $h\lambda$, when λ is purely imaginary, i.e. $|h\lambda| = 1$ and $\arg(h\lambda) = 90^{o}$, we have no region of stability at all with the fifth order corrector. When $\arg(h\lambda) = 108^{o}$, near the imaginary axis, we do not have a stable method when $\mu = 0$, the case of the BDF. As with the third order corrector, though, nonzero values of matrix error will yield a stable P_5EC_6 method.

Figure C13. P_5EC_6 Scalar SIBDF Method with $|h\lambda|=1$
Stability Limitation in the μ-plane

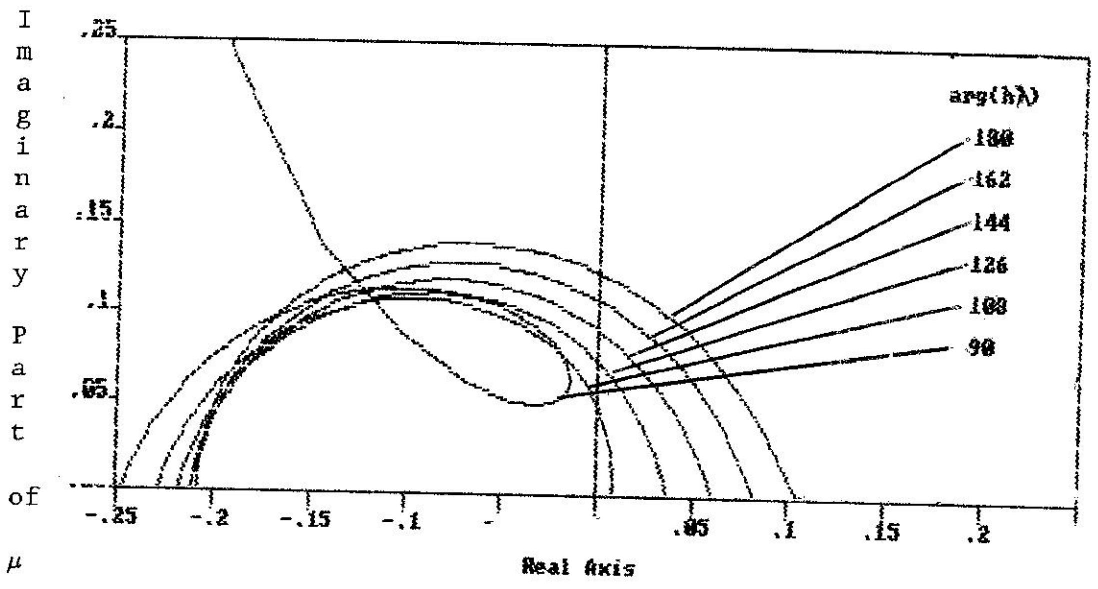


Figure C14. P_5EC_6 Scalar SIBDF Method with $|h\lambda|=5$
Stability Limitation in the μ-plane

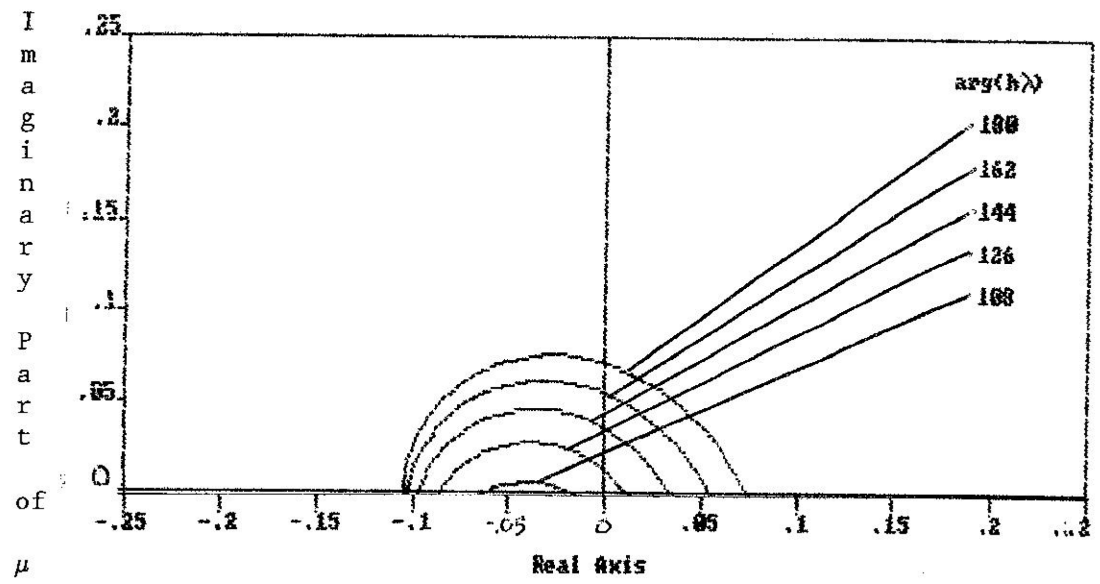
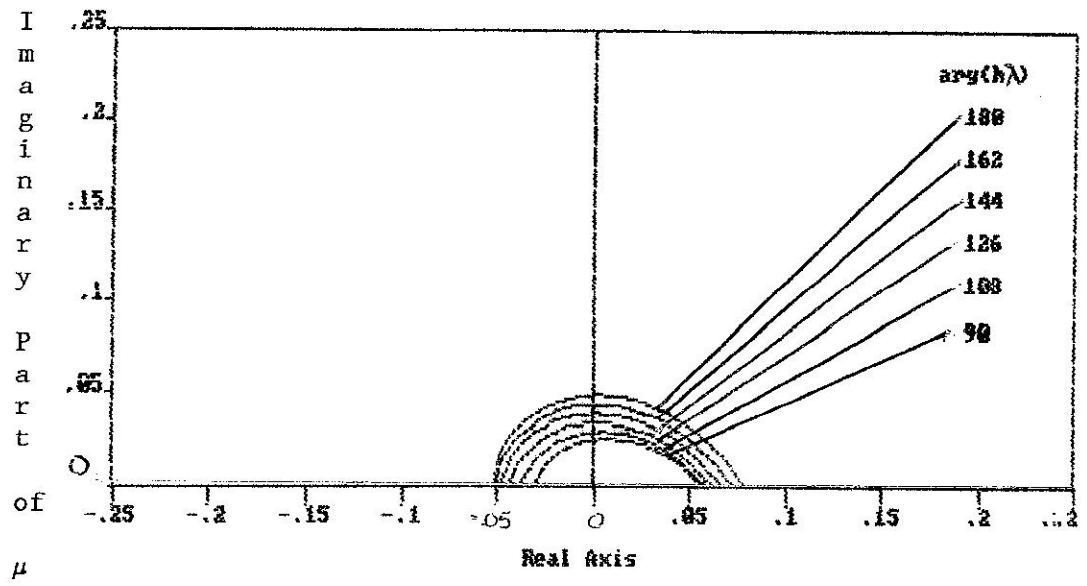


Figure C15. P_5EC_6 Scalar SIBDF Method with $|h\lambda|=25$
Stability Limitation in the μ-plane



References

[1] Brayton, Robert K., Gustavson, Fred G., and Hachtel, Gary D., 'A New Efficient Algorithm for Solving Differential-Algebraic Systems Using Implicit Backward Differentiation Formulas', Proc. of the IEEE 60 (1972) 98-107.

[2] Byrne, G.D. and Hindmarsh, A.C., 'A Polyalgorithm for the Numerical Solution of Ordinary Differential Equations', ACM Trans. on Math. Softw. 1 (1975) 71-96.

[3] Byrne, George D., Hindmarsh, Alan C., Jackson, Kenneth R., Brown, H. Gordon, 'A Comparison of Two ODE Codes: GEAR and EPISODE', Computers and Chem. Engineering 1 (1977) 133-147.

[4] Chan, Tony F. and Jackson, Kenneth R., 'The Use of Iterative Linear-Equation Solvers in Codes for Large Systems of Stiff IVPs for ODEs', SIAM J. Sci. Stat. Comput. 7 (1986) 378-417.

[5] Cryer, Colin W., 'On the Instability of High Order Backward-Difference Multistep Methods', B.I.T. 12 (1972) 17-25.

[6] Curtis, A.R., 'Solution of Large, Stiff Initial Value Problems - The State of the Art', in Numerical Software - Needs and Availability, ed. D. Jacobs, Academic Press (1978) 257-277.

[7] Curtis, A.R., 'The FACSIMILE Numerical Integrator for Stiff Initial Value Problems', in Computational Techiques for Ordinary Differential Equations, ed. I. Gladwell and D. Sayers, Academic Press, 1980.

[8] Curtis, A.R., 'Jacobian Matrix Properties and Their Impact on Choice of Software for Stiff ODE Systems', IMA J. Numerical Analysis 3 (1983) 397-415.

[9] Curtis, A.R., DC03A/AD Stiff ODE Software from 1985 Harwell Library, Computer Science and Systems Division, A.E.R.E. Harwell, Oxon

[10] Curtiss, C.F. and Hirschfelder, J.O., 'Integration of Stiff Equations', Proc. National Academy of Sciences USA 38 (1952) 235-243.

[11] Dennis, J.E., Jr., and Schnabel, Robert B., Numerical Methods for Unconstrained Optimization and Nonlinear Equations, Prentice-Hall, Inc., 1983.

[12] Enright, W.H., Hull, T.E. and Lindberg, B., 'Comparing Numerical Methods for the Solution of Stiff Systems of Equations', B.I.T. 15 (1975) 10-48.

[13] Forsythe, George E. and Moler, Cleve B., Computer Solution of Linear Algebraic Systems, Prentice-Hall, Inc., 1967.

[14] Franklin, J.N., Matrix Theory, Prentice-Hall, 1968.

[15] Gear, C.W., 'The Automatic Integration of Stiff Ordinary Differential Equations', Information Processing 68, North-Holland Pub. Co., 1969.

[16] Gear, C.W., 'Algorithm 407, DIFSUB for Solution of Ordinary Differential Equations', Communications of the ACM 14 (1971) 185-190.

[17] Gear, C.W., Numerical Intial Value Problems in Ordinary Differential Equations, Prentice-Hall, 1971.

[18] Gohberg, I., Lancaster, P., and Rodman L., Matrix Polynomials, Academic Press, 1982.

[19] Hall, G. and Watt, J.M., Modern Numerical Methods for Ordinary Differential Equations, Clarendon Press, Oxford, 1976.

[20] Hall, G, 'Numerical Solution of Stiff Systems of Ordinary Differential Equations', Numerical Analysis Report No. 64, University of Manchester (August 1981) 12 pp.

[21] Henrici, P., Discrete Variable Methods for Ordinary Differential Equations, John Wiley, 1962.

[22] Hindmarsh, A.C., 'Construction of Mathematical Software Part III: The Control of Error in the GEAR Package for Ordinary Differential Equations', UCID-30050 Part 3, Lawrence Livermore Laboratory (1972) 15 pp.

[23] Hindmarsh, A.C., 'GEAR: Ordinary Differential Equation System Solver', UCID-30001, Rev. 3, Lawrence Livermore Laboratory, December, 1974.

[24] Hindmarsh, A.C., 'On Error Estimation in the BDF Method for Ordinary Differential Equations', UCRL-80247, Lawrence Livermore Laboratory (1977) 14 pp.

[25] Hindmarsh, A.C., 'LSODE and LSODI, Two New Intial Value Ordinary Differential Equation Solvers', ACM/SIGNUM Newsletter 15, No. 4 (1980) 10-11.

[26] Hull, T.E., Enright, W.H., Fellen, B.M. and Sedgwick, A.E., 'Comparing Numerical Methods for Ordinary Differential Equations', SIAM J. Numerical Analysis 9 (1972) 603-637.

[27] Isaacson, E., and Keller, H.B., Analysis of Numerical Methods, John Wiley, 1966.

[28] Jackson, K.R. and Sacks-Davis, R., 'An Alternative Implementation of Variable Step-Size Multistep Formulas for Stiff ODES', ACM Trans. on Math. Softw. 6 (1980) 295-318.

[29] Klopfenstein, R.W., 'Numerical Differentiation Formulas for Stiff Systems of Ordinary Differential Equations', RCA Review 32 (1971) 447-462.

[30] Klopfenstein, R.W., Davis, C.B., 'PECE Algorithms for the Solution of Stiff Systems of Ordinary Differential Equations', Math. Comp. 25 (1971) 457-473.

[31] Krogh, F.T., 'The Numerical Integration of Stiff Differential Equations', TRW Systems Group Report 99900-6573-R000, 28 March 1968.

[32] Krogh, F.T., 'A Variable Order Variable Step Multistep Method for the Numerical Solution of Ordinary Differential Equations', Information Processing 68, North-Holland Pub. Co., 1969.

[33] Krogh, F.T., 'Algorithms for Changing the Stepsize', SIAM Journal of Numerical Analysis (1973) 949-965.

[34] Krogh, F.T., 'Changing Stepsize in the Integration of Differential Equations Using Modified Divided Differences', Proc. Conf. on Numerical Solution of Ordinary Differential Equations, Lecture Notes in Mathematics Vol. 362, Springer-Verlag, 1974.

[35] Krogh, F.T. and Stewart, K., 'Asymptotic ($h \to \infty$) Absolute Stability for BDFs Applied to Stiff Differential Equations', ACM Trans. on Math. Softw. 10 (1984) 45-57.

[36] Lambert, J.D., Computational Methods in Ordinary Differential Equations, John Wiley, 1973.

[37] Moler, Cleve and Van Loan, Charles, 'Nineteen Dubious Ways to Compute the Exponential of a Matrix', SIAM Review 20 (1978) 801-836.

[38] Norsett, Syvert P., 'A Criterion for $A(\alpha)$-Stability of Linear Multistep Methods', B.I.T. 9 (1969) 259-263.

[39] Ortega, J.M., Numerical Analysis: A Second Course, Academic Press, 1972.

[40] Petzold, L.R., 'Differential/Algebraic Equations are not ODE's', SIAM J. Sci. Stat. Comput. 3 (1982) 367-384.

[41] Petzold, L.R., 'Automatic Selection of Methods for Solving Stiff and Nonstiff Systems of Ordinary Differential Equations', SIAM J. Sci. Stat. Comput. 4 (1983) 136-148.

[42] Rubner-Petersen, Thomas, 'An Efficient Algorithm Using Backward Time-Scaled Differences for Solving Stiff Differential-Algebraic Systems', Inst. of Circuit Theory and Telecommunication Technical Report (Sept. 1973) Technical University of Denmark.

[43] Sacks-Davis, R., 'Error Estimates for a Stiff Differential Equation Procedure', Math. of Comp. 31 No. 140 (1977) 939-953.

[44] Shampine, L.F., 'Lipschitz Constants and Robust ODE Codes', in Computational Methods in Nonlinear Mechanics, J.T. Oden, Ed., North Holland, 1980, 424-449.

[45] Shampine, L.F., 'Evaluation of Implicit Formulas for the Solution of ODE's', B.I.T. 19 (1979) 495-502.

[46] Shampine, L.F., 'Implementation of Implicit Formulas for the Solution of ODE's', SIAM J. Sci. Stat. Comput. 1 (1980) 103-118.

[47] Shampine, L.F. and Gear, C.W., 'A User's View of Solving Stiff Ordinary Differential Equations', SIAM Review 21 (1979) 1-17.

[48] Shampine, L.F. and Gordon, M.K., Computer Solution of Ordinary Differential Equations: The Initial Value Problem, W.H. Freeman and Co., 1975.

[49] Shampine, L.F. and Gordon, M.K., 'Local Error and Variable Order Adams Codes', Appl. Math. and Comp. 1 (1975) 47-66.

[50] Skelboe, Stig, 'The Control of Order and Steplength for Backward Differentiation Formulas', B.I.T. 17 (1977) 91-107.

[51] Soderlind, Gustaf, 'On the Efficient Solution of Nonlinear Equations in Numerical Methods for Stiff Differential Systems', TRITA-NA-8114, Royal Inst. of Technology, Stockholm, (1984).

[52] Stewart, K. and Krogh, F.T., 'Preliminary Comparison Test Results for STRUT, LSODE and LSODA', Section 366 Internal Memorandum No. 499, Jet Propulsion Laboratory, Pasadena, July 1983.

[53] Wilkinson, J.H., The Algebraic Eigenvalue Problem, Clarendon Press, Oxford, 1965.