NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

ISBN 0-315-53229-7

Canada

AN INVESTIGATION OF THE PARALLEL PREDICTOR-CORRECTOR CLASS OF METHODS FOR THE SOLUTION OF ODE'S

by

MIN CHEN

A thesis submitted to the School of Graduate Studies
and Research of the University of Ottawa
in partial fulfillment of the requirements of
Master of Science in Systems Science

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

To my grandmother
for all the love she
gave to me

ACKNOWLEDGEMENTS

I would like to express my most sincere appreciation to my supervisor Dr. L. G. Birta, for his advice, valuable guidance, encouragement in this research work and great effort in helping me to prepare this thesis. Profound thanks are also extended to Dr. O. Abou-Rabia and Dr. R. Vaillancourt for their helpful suggestions and discussions.

Abstract

The problem of distributing, over a set of processors, the computational workload involved in carrying out a continuous system simulation study is essentially equivalent to the problem of achieving parallelism in the solution of ordinary differential equations (ode's). Block oriented methods having a predictor-corrector basis have emerged as a feasible means of pursuing the goal of parallelism in the ode context. Such methods offer the important advantage of yielding algorithmic procedures that map the required computation workload onto the available processors without any need to decompose the underlying ode's.

This thesis examines the Parallel Predictor-Corrector Method (PPC) proposed orginally by Miranker and Liniger. It provides a comprehensive evaluation of the performance of both the originally proposed PPC formulas and a newly formulated variation. Also, a new variable stepsize implementation of the PPC approach is formulated and its performance is evaluated. Stability analysis of the methods considered in the study are presented together with the results of extensive numerical experiments.

Contents

Chapter 1

Introduction

1.1 Review of Past Work

Development of algorithms for continuous system simulation which are parallel, rather than serial, in nature has been prompted by two complementary developments. On one hand, there is the need to accommodate simulation problems of ever increasing complexity and on the other hand, there is the increasing availability of multiprocessor computer systems. Also micro-processors have substantially decreased in cost and at the same time, increased in power. These processors can now be used to build multi-processor computing systems whose effectiveness in systems simulation will be largely determined by the success with which parallel algorithms can be developed to exploit their multi-processing capability.

In a continuous system simulation study, the system is typically described by a set of ordinary differential equations (ode's). (A common alternative is a system described by partial differential equations; however, for solution purposes such a model is often transformed, through approximation procedures, to one based on ode's). The fundamental step in the simulation activity consists of integrating these equations numerically. Parallel methods for carrying out this numerical task have been proposed by various researchers.

Approaches for achieving parallelism in solving a set of ode's fall into two broad categories; namely, the problem partitioning approach and the parallel algorithm

approach. In the former case, a classical integration procedure is used to progress across the solution axis and parallelism is achieved by distributing the necessary computational tasks over the available processors. A wide variety of options are available for carrying out this segmentation. For example, it can occur at the level of individual equations, which is called the equation segmentation approach, or at finer levels of granularity, depending on the number of available processors. In order to achieve this approach, we have to deal with two inherent problems; namely, work-load balance among the processors and development of a high bandwidth data exchange mechanism to support the (frequently) substantial data exchange requirements among processors. In [15], O'Grady and Wong have suggested that the problem partitioning approach is best suited for problems which have a short critical path and a large computational requirement per step.

In the parallel algorithm approach, parts of the integration procedure itself (rather than parts of the problem to be solved) are distributed over the set of available processors. Most of the parallel algorithm methods proposed in the literature belong to two families, i.e., Block Predictor-Corrector (BPC) methods and Parallel Predictor-Corrector (PPC) methods. Methods in both of these two categories can be viewed as an extension of the conventional predictor-corrector approach in which each of the available processor carries out a computation that relates to a distinct point along the solution axis. The computation corresponds either to a computation of a predicted or corrected value or to a derivative function evaluation. The difference between the BPC and PPC approaches relates to whether predicted and corrected values are generated sequentially (BPC) or simultaneously (PPC). In the parallel algorithm approach, all processors carry out a sequence of computationally identical tasks and thus the work-load balancing problem is entirely circumvented.

An outline of the equation segmentation method can be found in Franklin [8]. In this approach, the set of ode's is partitioned into a group of disjoint subsets which are then distributed over the set of available processors. Each processor is then responsible for carrying out the solution of its assigned set of equations. The two obvious problems which arise with this approach are:

(a) how to decompose the given system model into subsets which will equalize the computational load over the processors

(b) how to efficiently carry out the communication of values between processors which is necessary because of current coupling between the equation subsets.

The BPC method is an approach which was first proposed by Shampine and Watts in [16]. Further work with the BPC method was carried out by Birta and Abou-Rabia [4] who, in particular, developed a variable stepsize procedure for the method. In this method, time (the assumed independent variable of the problem) is divided into a series of blocks with each block containing a number of points at which the solution to the system of equations is to be generated. Blocks values are all obtained simultaneously in a single block advance and each such calculation can be considered as the basic unit of the under computation. In the BPC method, the solution values within a block are computed simultaneously based on the existing values within a previous block and the computation proceeds block by block. Within a block it is possible to assign both the predictor and corrector computations at each point to a single processor, and to perform the computations simultaneously (i.e., in parallel).

The PPC method was originally presented by Miranker and Liniger [14]. In this approach, the processors are divided into two distinct sets each having s processors. Those in one set consistently carry out predictor calculations while those in the other set carry out corrector calculations. Each underlying cycle of the procedure advances the solution by s new values. Normally with predictor-corrector methods, the computation of a predicted value is based on corrected values at some previous points. The predictor formulas for the PPC method are distinctive inasmuch as they do not follow this general approach. More specifically, the predictor calculations use predicted values, rather than corrected values, in the previous block. This makes possible the simultaneous predictor and corrector computation that are characteristics of the method.

In [3], Abou-Rabia and Birta presented a hardware implementation of a multimicrocomputer system for continuous system simulation based on the use of Intel

ISBC 86/05 single board computers. The system architecture is designed for a class of parallel integration algorithms and hence provides the means for solving the ordinary differential equations of the system model in a parallel way but without partitioning these equations. The microcomputer communications are carried out through an array of replicated memories which provides for instantaneous data exchange among the processors.

In [6], the two-processor implementation of the BPC the method was examined on a multi-processor machine. The machine used was the Denelcor HEP at Argonne National Laboratory. The HEP (Heterogeneous Element Processor) is a large-scale scientific multi-processor system which can execute a number of parallel programs simultaneously. The results show that the theoretical speed-up can almost be achieved when the cost of derivative evaluation is high.

1.2 Main Contributions of the Thesis

In this thesis, we develop a set of general formulas for the parallel predictor-corrector procedure based on the ideas originally presented in [14]. Then we propose a new member of the general class which uses corrector formulas that are different from those originally suggested in [14]. The performance of both realizations is examined in the context of experiments with a set of test problems. These experiments extend over a sequence of formulas of increasing order used with various numbers of processors. The impact of varying the desired level of solution accuracy is also investigated. From those experiments, it is shown that the new method provides better results in most cases, especially when the number of processors is large and the order is high. Stability analysis with both methods is also carried out and the new method is shown to have superior absolute stability boundaries over the range of available processors and integration orders that are considered in the study.

In this thesis, we also formulate a variable stepsize procedure for the new method. The performance of the variable stepsize method is also evaluate using a set of test problems over a range of integration orders and available processors.

Chapter 2

The Parallel Predictor-Corrector Formulas

2.1 Introduction

A fundamental requirement in continuous system simulation is that of generating the solution to a set of ordinary differential equations (ode's) of the form :

$$dy/dt = f(t, y(t)) \qquad y(t_0) = y_0 \tag{2.1}$$

where y and f are both vectors and t_0 and y_0 are given. In fact, we assume that the model of the system being investigated in the simulation study is represented by such a set of equations. A simulation study will typically require many solutions to (2.1) in order to gain the desired insight into the system's behavior. This task can be computationally intensive when the number of equations is large and/or the derivative function, f, is "complex". In such circumstances, the achievable solution rate in a particular single processor environment may be unacceptably low. The use of multiple processors can, at least in principle, provide a means for circumventing this difficulty.

The basic condition which guarantees the existence of a unique solution of the initial value problem (2.1) is known as a Lipschitz condition which has the following form :

Let $f(t,y)$ be defined and continuous for all points (t,y) in the region D defined by $a \leq t \leq b$, $-\infty \leq y \leq \infty$, a and b finite, and let there exist a constant L (a Lipschitz constant) such that, for every t, y, y^* for which (t,y) and (t,y^*) are both in D, it is true that

$$|f(t,y) - f(t,y^*)| \leq L|y - y^*| \tag{2.2}$$

Then, for any given number y_0, there exists a unique solution $y(t)$ of the initial value problem (2.1), where $y(t)$ is continuous and differentiable for all (t,y) in D. Throughout our considerations, this condition is assumed to be satisfied.

Each iteration of a serial integration method for the solution of ode's generates one new solution value by means of an integration formula that uses one or more past solution values. Each iteration of a parallel integration method, on the other hand , simultaneously generates a set of solution values, at different points in time. With such a method, the computation at each time point is based on a different formula and is assigned to a particular different processor in a multiprocessor environment.

Integration formulas having the desired feature of parallelism generally fall into the category of block oriented methods. Such methods can be viewed as an extension of the conventional predictor-corrector approach in which each of the available collection of processors carries out a computation that relates to a distinct point along the solution axis. The processors operate in synchronism and consequently each underlying cycle of the procedure advances the solution by a distance sh where h is the spacing between the solution values within a block and s is the number of points in the block.

The Parallel Predictor-Corrector (PPC) method was originally proposed by Miranker and Liniger [14]. In that presentation, second, third and fourth order formulas are given for the two and four processor cases. In addition some performance results are given based on tests with two simple linear problems. Some further work with this approach was carried out by Krosel and Milner [11] who presented second order formulas for two, four and eight processor cases and also provided some limited test results. Katz and Franklin [10] provide a comprehensive analysis of the stability properties of the PPC method for the two processor case.

In this chapter, we begin first by providing a general formulation of the class of PPC methods. A description is then provided of a means for establishing values for the parameters within the general formulas in order to obtain methods of arbitrary order. These parameter values are dependent on the prescribed number of available processors. An analysis of stability properties is also included in this chapter.

It should be noted that in the interest of notational simplicity (but without loss of generality) the considerations in the following sections are based on the assumption that the systems model described by equation (2.1) contains only a single ode.

2.2 The Parallel Predictor-Corrector (PPC) Family

We assume that the t axis between $t = t_1$ and $t = t_{(n+1)s}$ is divided into a sequence of $(n+1)$ adjacent blocks, each containing s equally spaced values, where $s = N/2$ and N is the number of available processors (assumed to be even). In the fixed stepsize case which we initially consider, $t_i = ih$ where h is stepsize and we use y_i to denote a computed solution to (2.1) at point t_i. Let Y_j be the s vector, $Y_j = (y_{js}, y_{js-1}, y_{js-2}, \ldots, y_{(j-1)s+1})^T$, where $j \geq 1$, and similarly let F_j be the s-vector, $F_j = (f_{js}, f_{js-1}, f_{js-2}, \ldots, f_{(j-1)s+1})^T$, where $f_i = f(t_i, y_i)$.

The solution to (2.1) is assumed to already have been produced at the points within the first $(n-1)$ of these blocks; i.e., up to and including $t = t_{(n-1)s}$. Furthermore within block n we assume there exists a "tentative" solution vector Y_n^P (i.e. a set of s "predicted" values). Our particular concern is with blocks n and $(n+1)$ as shown in Figure 1. The computational task at this point has two objectives, namely,

(i) Use the existing solution values in blocks $(n-1)$, $(n-2)$, ..., and the "predicted" values in block n to produce "predicted" (i.e. tentative) solution values at the s points within block $(n+1)$.

(ii) Use the existing solution values in blocks $(n-1)$, $(n-2)$, ..., and the "predicted" values in block n to produce "corrected" (i.e., final) solution values at the s points within block n.

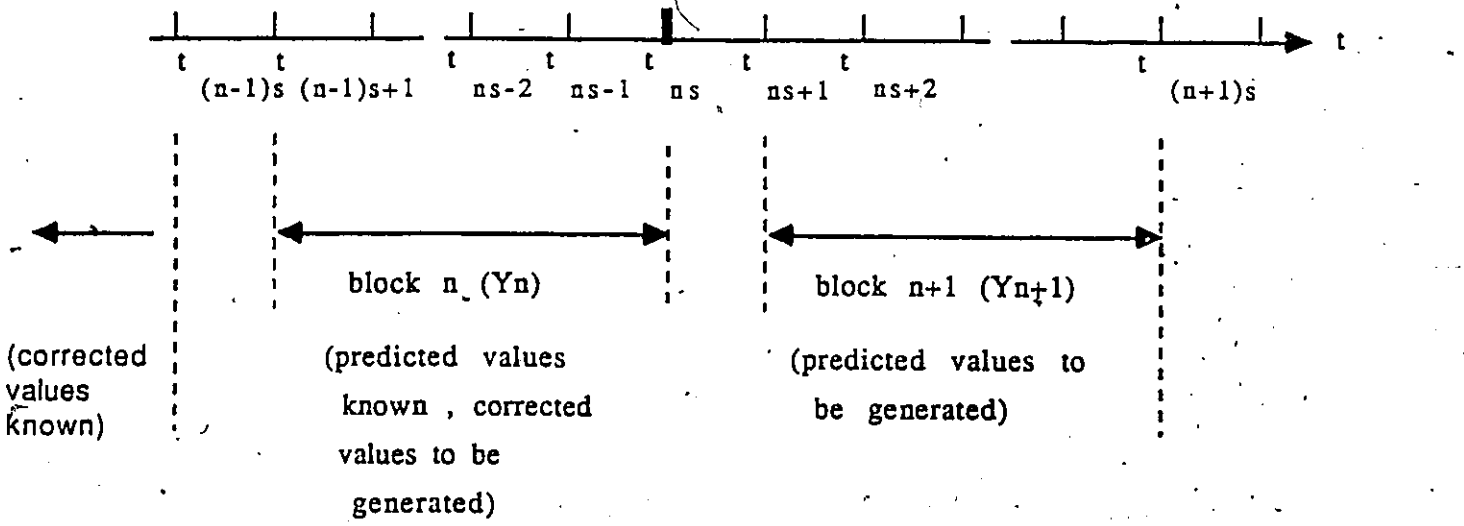

Figure 1

Configuration of Blocks n and (n+1)

These objectives can be expressed in the form of a k-block linear systems which can be written as:

$$A_0^P Y_{n+1}^P + A_1^P Y_n^P + \sum_{j=2}^{k} A_j^P Y_{n+1-j} + h(B_1^P F_n^P + \sum_{j=2}^{k} B_j^P F_{n+1-j}) = 0 \qquad (2.3)$$

$$A_0 Y_n + A_0^* Y_n^P + \sum_{j=1}^{k} \tilde{A}_j Y_{n-j} + h(B_0 F_n^P + \sum_{j=1}^{k} B_j F_{n-j}) = 0 \qquad (2.4)$$

$$k \leq n-1$$

where A_0^*, A_j, A_j^P, B_j, B_j^P, for $j = 0, 1, \ldots, k$ are $s \times s$ matrices. Furthermore we assume $|A_0^P| \neq 0$ and $|A_0| \neq 0$. (Throughout the presentation, we use $|M|$ to denote

the determinant of the square matrix, M). Equation (2.3) provides the mechanism for achieving objective (i) listed above, while (2.4) provides the mechanism for achieving objective (ii).

To obtain a more compact representation of (2.3) and (2.4) we introduce the $N = (2s)$-vectors:

$$\bar{Y}_j = (Y_{j+1}^P, Y_j)^T \quad \text{and} \quad \bar{F}_j = (F_{j+1}^P, F_j)^T$$

and let

$$\bar{A}_0 = \begin{pmatrix} A_0^P & 0 \\ 0 & A_0 \end{pmatrix} \quad \bar{A}_1 = \begin{pmatrix} A_1^P & A_2^P \\ A_0^* & A_1 \end{pmatrix} \quad \bar{B}_1 = \begin{pmatrix} B_1^P & B_2^P \\ B_0 & B_1 \end{pmatrix}$$

$$\bar{A}_j = \begin{pmatrix} 0 & A_{j+1}^P \\ 0 & A_j \end{pmatrix} \quad \bar{B}_j = \begin{pmatrix} 0 & B_{j+1}^P \\ 0 & B_j \end{pmatrix} \qquad j = 2, 3, \ldots, k$$

be $(N \times N)$-matrices with $A_{k+1}^P = B_{k+1}^P = 0$. Then equations (2.3) and (2.4) can be written in a consolidated form as :

$$\sum_{j=0}^{k} \bar{A}_j \bar{Y}_{n-j} + h \sum_{j=1}^{k} \bar{B}_j \bar{F}_{n-j} = 0 \tag{2.5}$$

where $|\bar{A}_0| \neq 0$. Equation (2.5) provides the general specification of the k-block parallel predictor-corrector (PPC) class of methods.

2.3 Parameter Specification for the PPC Formulas

The specification of the parameters in (2.5) is based on two considerations; namely, convergence and the requirements that arise from the designation of a specific order for the integration process. We consider each of these in turn.

2.3.1 Convergence

An important notion associated with any linear multistep methods is that of convergence. This notion relates to the behavior of the method relative to the true solution of the underlying ode's to be solved, when the integration stepsize $h \to 0$, $n \to \infty$ and nh is constant. As a starting point for exploring more precisely this notion in the context of the general PPC system given by (2.5) we introduce the matrix polynomials $P(z)$ and $Q(z)$ in the complex variable, z. The matrix coefficients of those polynomials are obtained from the specification of (2.5), namely,

$$P(z) = \sum_{j=0}^{k} \bar{A}_j z^{k-j} \tag{2.6}$$

$$Q(z) = \sum_{j=1}^{k} \bar{B}_j z^{k-j} \tag{2.7}$$

The necessary and sufficient conditions for a linear multistep method to be convergent are that it be consistent and zero-stable. Since the condition of consistency is automatically satisfied if the integration order $r \geq 1$, so here we only study the condition of zero-stability.

The general PPC method given by (2.5) is zero-stable if all Nk roots of $|P(z)|$ are in the closed unit disk and those of modulus one are simple. As an initial simplification in (2.5), we take $\bar{A}_j = 0$ for $2 \leq j \leq k$. As a result

$$P(z) = \bar{A}_0 z^k + \bar{A}_1 z^{k-1} = z^{k-1}(\bar{A}_0 z + \bar{A}_1)$$

and consequently

$$|P(z)| = z^{N(k-1)} |\bar{A}_0 z + \bar{A}_1|$$

Clearly $|P(z)|$ has the root $z = 0$ with multiplicity at least $N(k-1)$. The remaining N roots of $|P(z)|$ are those of $|\bar{A}_0 z + \bar{A}_1|$.

By way of further simplification, we choose $A_0^P = A_0 = -I_s$ (here the notation I_m is used to denote the $m \times m$ identity matrix). As a consequence $\bar{A}_0 = -I_N$. Further we choose $A_0^* = 0 = A_1^P$ and thus:

$$\bar{A}_0 z + \bar{A}_1 = \begin{pmatrix} -I_s & 0 \\ 0 & -I_s \end{pmatrix} z + \begin{pmatrix} 0 & A_2^P \\ 0 & A_1 \end{pmatrix} = \begin{pmatrix} -I_s z & A_2^P \\ 0 & A_1 - I_s z \end{pmatrix}$$

Since $\bar{A}_0 z + \bar{A}_1$ is block upper triangular, it follows that

$$|\bar{A}_0 z + \bar{A}_1| = |-I_s z||A_1 - I_s z| = z^s |I_s z - A_1|$$

Consequently

$$|P(z)| = z^{(Nk-s)} |I_s z - A_1|$$

The remaining s roots of $|P(z)|$ are clearly controlled by the choice of the $s \times s$ matrix A_1.

One simple choice for A_1 which ensures that the roots of $|P(z)|$ satisfy the root condition is (we use * simply to denote an arbitrary value within the array structure):

$$A_1 = A_1^a = \begin{pmatrix} 0 & * & \dots & \dots & * \\ 0 & 0 & * & \dots & * \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & \dots & * \\ 0 & 0 & \dots & 0 & 1 \end{pmatrix}$$

With this choice, (which was originally suggested by Miranker and Liniger [14]) we have

$$|I_s z - A_1| = \begin{pmatrix} z & * & \dots & \dots & * \\ 0 & z & * & \dots & * \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & z & * \\ 0 & 0 & \dots & 0 & z-1 \end{pmatrix} = z^{s-1}(z-1)$$

and so $|P(z)| = z^{Nk-1}(z-1)$; i.e. a multiple root at $z = 0$ and a simple root at $z = 1$.

An alternate, and equally valid choice for A_1 is:

$$A_1 = A_1^b = \begin{pmatrix} 1 & 0 & \dots & 0 & 0 \\ * & 0 & \dots & 0 & 0 \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ * & \dots & * & 0 & 0 \\ * & \dots & \dots & * & 0 \end{pmatrix}$$

With this choice it again follows that

$$|I_s z - A_1| = z^{s-1}(z-1)$$

and consequently the roots of $|P(z)|$ are identically located as before.

The specific form for A_1^a suggested by Miranker and Liniger [14] is

$$A_1 = A_1^a = \begin{pmatrix} 0 & 1 & 0 & \dots & \dots & 0 \\ 0 & 0 & 1 & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & 0 & \dots & 0 & 1 \\ 0 & 0 & 0 & \dots & 0 & 1 \end{pmatrix}$$

In our alternate proposal we take

$$A_1 = A_1^b = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 1 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & \dots & 0 \end{pmatrix}$$

With the choice of A_1^b, only the most recent solution value (apart from derivative values) is used in (2.4). One of our concerns in the sequel is with evaluating the relative merits of the procedures resulting from the use of A_1^a and A_1^b. We refer to these two methods as Method A and Method B respectively.

2.3.2 Integration Order

With the various assumptions made in section 2.3.1, equation (2.3) and (2.4) can be written simply as:

$$-I_s Y_{n+1}^P + A_2^P Y_{n-2} + h(B_1^P F_n^P + \sum_{j=2}^{k} B_j^P F_{n+1-j}) = 0 \qquad (2.8)$$

$$-I_s Y_n + A_1 Y_{n-1} + h(B_0 F_n^P + \sum_{j=1}^{k} B_j F_{n-j}) = 0 \qquad (2.9)$$

The specification of the remaining parameters within these formulas can be obtained from the consideration of a generalization of the classical linear multistep method. In particular, we examine the method defined by:

$$y_u = \sum_{j=1}^{k} \alpha_j y_{u-m_j} + h \sum_{j=0}^{\ell} \beta_j f_{u-n_j} \tag{2.10}$$

where the m_j, n_j are non-negative integers which satisfy:

(i) $m_i < m_j$ for $i < j$

(ii) $n_i < n_j$ for $i < j$

(iii) $m_1 \neq 0$, $n_1 \neq 0$

(iv) $\beta_0 n_0 = 0$

The method defined by (2.10) is said to be explicit if $\beta_0 = 0$ and implicit if $\beta_0 \neq 0$.

The free parameters in (2.10) can be represented by the vectors

$$\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_k)^T$$

$$\beta = (\beta_\mu, \beta_{\mu+1}, \ldots, \beta_k)^T$$

where $\mu = 1$ if (2.10) is explicit and $\mu = 0$ if (2.10) is implicit. For (2.10) to represent an r^{th} order formula, we must have (2.10) exactly satisfied for the cases where

(a) $y(t) = 1$, (and hence: $y'(t) = 0$)

(b) $y(t) = t^{m+1}$, (and hence: $y'(t) = (m+1)t^m$), $m = 0, 1, \cdots, (r-1)$.

From (a) we get

$$\sum_{j=1}^{k} \alpha_j = 1$$

or equivalently $v_0^T \alpha = 1$ where v_0 is a k-vector whose entires are all 1's.

From (b): if $m = 0$, i.e. $y(t) = t$, $y'(t) = 1$, we get:

$$y(t_u) = \sum_{j=1}^{k} \alpha_j y(t_u - m_j h) + h \sum_{j=0}^{k} \beta_j y'(t_u - n_j h)$$

i.e.,

$$t_u = \sum_{j=1}^{k} \alpha_j (t_u - m_j h) + h \sum_{j=0}^{k} \beta_j$$

Setting $t_u = 0$, yields:

$$\sum_{j=1}^{k} \alpha_j m_j - \sum_{j=0}^{k} \beta_j = 0$$

Consider now the case where $1 \leq m \leq (r-1)$, then we get:

$$y(t_u) = \sum_{j=1}^{k} \alpha_j y(t_u - m_j h) + h \sum_{j=0}^{k} \beta_j y'(t_u - n_j h)$$

i.e.,

$$t_u^{m+1} = \sum_{j=1}^{k} \alpha_j (t_u - m_j h)^{m+1} + h \sum_{j=0}^{k} \beta_j (m+1)(t_u - n_j h)^m$$

Setting $t_u = 0$ gives:

$$\sum_{j=1}^{k} \alpha_j (-m_j h)^{m+1} + h \sum_{j=0}^{k} \beta_j (m+1)(-n_j h)^m = 0$$

i.e.,

$$\sum_{j=1}^{k} \alpha_j m_j^{m+1} - (m+1) \sum_{j=0}^{k} \beta_j n_j^m = 0 \quad for \quad m = 1, 2, \ldots, (r-1)$$

We can write this formula in a matrix form:

$$V\alpha - D\bar{V}\beta = 0$$

where .

$$V = (v_1, v_2, \ldots, v_k), \qquad \bar{V} = (\bar{v}_\mu, \bar{v}_{\mu+1}, \ldots, \bar{v}_k)$$

D is a diagonal matrix with $D_{jj} = j$, and

$$v_j = (m_j, m_j^2, \ldots, m_j^r)^T \qquad 1 \le j \le k$$
$$\bar{v}_j = (1, n_j, n_j^2, \ldots n_j^{r-1})^T \qquad 0 \le j \le k$$

Thus for (2.10) to represent an r^{th} order formula, α and β must satisfy the following equations:

$$v_0^T \alpha = 1 \tag{2.11}$$

$$V\alpha - D\bar{V}\beta = 0 \tag{2.12}$$

In particular, it should be observed that if α is specified, and (2.10) is explicit (i.e., $\mu = 1$), a unique value for β can be obtained by taking $k = r$, (this follows because then $\bar{V}$ is square and non-singular). Similarly when (2.10) is implicit (i.e., $\mu = 0$), a unique value for β can be obtained with $k = r - 1$.

A convenient choice for A_2^P in (2.8) is the $s \times s$ matrix whose entries are zero except for those in the first column which are 1's. With this choice, the i^{th} predictor equation ($1 \le i \le s$) in (2.8) can be written as:

$$y_u = y_{u-m_1} + h(\sum_{j=1}^{k} \beta_j f_{u-n_j}) \tag{2.13}$$

where

$$u = (n+1)s - i + 1$$
$$m_1 = 2s - i + 1$$

$$n_j = s + j - i, \qquad 1 \leq j \leq k$$

(The superscript P denoting predicted values has been surpressed since this distinction is not relevant in order-related considerations). Equation (2.13) clearly corresponds to the explicit form of (2.10) with a specific value for α which statisfies (2.11). The β_j coefficients in (2.13), for any order r, can therefore be obtained using (2.12).

The specific form of the s equations in (2.9) depends on the choice for A_1. With $A_1 = A_1^a$; i.e., Method A, the i^{th} (corrector) equation ($1 \leq i \leq s$) in (2.9) can be written as :

$$y_u = y_{u-m_1} + h(\sum_{j=0}^{k} \beta_j f_{u-n_j}) \tag{2.14}$$

where

$$u = ns - i + 1$$

$$m_1 = \begin{cases} s+1 & \text{for } i < s; \\ s & \text{for } i = s. \end{cases}$$

$$n_j = j$$

With the choice $A_1 = A_1^b$, i.e., Method B, the i^{th} equation ($1 \leq i \leq s$) in (2.9) also has the form given by (2.14), but in this case $m_1 = (s - i + 1)$. Equation (2.14) clearly corresponds to the implicit form of (2.10) with a specific value for α which satisfies (2.11). The β_j coefficients in (2.14), for any order r, can therefore be obtained from (2.12).

To illustrate the application of these general results, we give in Appendix 1, several specific sets of formulas corresponding to different orders and numbers of processors.

2.4 Procedure Summary and an Observation

The solution procedure for the PPC method is based on the use of two distinct sets of $s \geq 1$ processors which we call the P-set and the C-set respectively. The processors in the P-set are committed to predictor calculations while those in the

C-set are committed to corrector calculations. During each cycle of the procedure the processors in each set simultaneously carry out two separate computations and thereby advance the solution by sh.

During the execution of the n^{th} cycle, the processors in the P-set use formulas of a prescribed order, r, derived from equation (2.11) and (2.12) to compute predicted solution values at the s points in block $(n+1)$. Then, as a second step, the derivative values at these same points are computed. In parallel (i.e., simultaneously) with these activities, the processors in the C-set first compute corrected values at the s points within block n using formulas of order r also derived from equation (2.11) and (2.12) , and then compute derivative values at these same points. Since the formulas used to obtain solution values are structurally identical, all processors in both sets remain naturally synchronized.

It is interesting to observe that with the assumed structure for the corrector formulas (for both Method A and Method B), the r derivative values required to produce the y_{ns-j} value $(0 \leq j \leq s-1)$ are chosen to be the values at t_{ns-j} together with the $(r-1)$ values immediately to the left of t_{ns-j}. In particular $(r-j)$ of those values extend into block $(n-1)$ and possibly even "further left". The case which requires the left-most derivative value is the computation of $y_{(n-1)s+1}$. However if $(r-1) < s$ then $(s-r+1)$ derivative values within block $(n-1)$ are not used in any calculation. The implication here is that $(s-r+1)$ of the N processors do not do useful work during the derivative function evaluation phase of each cycle. In other words, when $(r-1) < s$, the full parallelism of the PPC process, as formulated in this study and as conventionally described in the literature is partially undermined.

2.5 Absolute Stability Bounds

In section 2.3.1, the requirement for zero stability was used to formulate certain structural constaints on both Method A and Method B. Recall, however, in that context, our concern was with the limiting situation where $h \to 0$, $n \to \infty$, and nh fixed. In the present discussion we focus our attention on error propogation when h is a fixed positive value and $n \to \infty$. These considerations fall in the domain of

absolute stability theory which, in turn, is one part of the broader domain of weak stability theory.

The absolute stability of any integration method relates to its behavior when applied to the solution of the single equation

$$dy/dt = cy(t) \tag{2.15}$$

When the parameter $c < 0$, any solution of this equation approaches zero as t increases, independent of the initial value. A fundamental requirement on any numerically generated solution, therefore, is that it too approach zero as the iterations progress.

In general terms, the analysis is based on the stability polynomial which is obtained by applying the formulas of the solution method under investigation to (2.15). This polynomial has coefficients which depend upon $\sigma = ch$, where h is a fixed stepsize. The stability region in the complex $\bar{\sigma}$ plane consists of those σ values for which the roots of the stability polynomial lie within the unit circle. Note also that our concern is restricted to the case where $\sigma < 0$ (because $h > 0$ and $c < 0$).

With the parallel predictor-corrector method, the formulation of the conditions necessary to ensure this "stability property", can be established by representing each cycle of the PPC process in the form:

$$Y_{new} = \Phi Y_{old} \tag{2.16}$$

i.e., as a linear transformation from old to new data values. The transformation matrix Φ in this representation is dependent on $\sigma = ch$. The stability behavior of interest therefore corresponds to ensuring that the limiting value resulting from repeated applications of this equation is zero. This can occur only if all eigenvalues of Φ have modulus less than one. Thus we want to find the largest values of $\sigma^* > 0$ such that for all σ in the range $(-\sigma^*, 0)$ this condition on the eigenvalues of Φ is satisfied. The value of σ^* represents the stability bound for the integration formula.

In order to formulate the matrix Φ, we first define the vectors Y_{new} and Y_{old}. For Method A; from (2.13), we have

$$y_{(n+1)s-i+1} = y_{ns-s} + h \sum_{j=1}^{r} \beta_j^i f_{ns-j+1} \tag{2.17}$$

where $1 \leq i \leq s$

From (2.14), we have:

$$y_{ns-i+1} = y_{ns-s-i} + h(\sum_{j=0}^{r-1} \beta_j^i f_{ns-i-j+1}) \quad \text{for } i < s \tag{2.18}$$

$$y_{ns-i+1} = y_{ns-s-i+1} + h(\sum_{j=0}^{r-1} \beta_j^i f_{ns-i-j+1}) \quad \text{for } i = s \tag{2.19}$$

Two separate cases have to be considered depending on whether $r < s+1$ or $r \geq s+1$. In the first case; i.e., $r < s+1$, the left-most value that is required is $f_{ns-2s+1}$, so we define Y_{new} and Y_{old} in the following way:

$$Y_{new} = (y^P_{sn+s}, y^P_{sn+s-1}, \cdots, y^P_{sn+1}, y_{sn}, y_{sn-1}, \cdots, y_{sn-s+1})^T$$

$$Y_{old} = (y^P_{sn}, y^P_{sn-1}, \cdots, y^P_{sn-s+1}, y_{sn-s}, y_{sn-s-1}, \cdots, y_{sn-2s+1})^T$$

Y_{old} and Y_{new} are both $2s$-vectors and Φ is a $2s \times 2s$ matrix which can be written as:

$$\left(\begin{array}{c|c|c} \sigma D_1 & E_1 & F_1 \\ \hline \multicolumn{3}{c}{\sigma G_1 + Z_1} \end{array}\right)$$

where:

D_1 is an $s \times r$ matrix where the i^{th} row consists of the coefficients in the i^{th} formula of (2.13); i.e.,

$$D_1 = \begin{pmatrix} \beta_1^1 & \beta_2^1 & \cdots & \beta_r^1 \\ \vdots & \vdots & & \vdots \\ \beta_1^s & \beta_2^s & \cdots & \beta_r^s \end{pmatrix}$$

E_1 is an $s \times (s-r)$ zero matrix,

F_1 is an $s \times s$ matrix whose entries are zero except for those in the first column,

which are 1's,

G_1 is a $s \times 2s$ matrix having the following form:

$$G_1(i,j) = \begin{cases} \beta^i_{j-1} & \text{for } 1 \le i \le s, \quad i \le j \le r; \\ 0 & \text{otherwise .} \end{cases}$$

where β^i is the vector of coefficients associated with the derivative values in the i^{th} formula of (2.14), i.e.,

$$\beta^i = (\beta^i_1, \beta^i_2, \ldots, \beta^i_r)$$

Z_1 is an $s \times 2s$ matrix having the following form:

$$Z_1(i,j) = \begin{cases} 1 & \text{for } 1 \le i \le s-1, \quad j = i+1; \\ 1 & \text{for } i = s, \quad j = 2s; \\ 0 & \text{otherwise .} \end{cases}$$

In the second case; i.e., $r \ge s+1$, the left-most value that is required is $f_{sn-s-r+2}$; consequently we define Y_{new} and Y_{old} as follows:

$$Y_{new} = (y^P_{sn+s}, y^P_{sn+s-1}, \cdots, y^P_{sn+1}, y_{sn}, y_{sn-1}, \cdots, y_{sn-s+1}, y_{sn-s}, \cdots, y_{sn-r+2})^T$$

$$Y_{old} = (y^P_{sn}, y^P_{sn-1}, \cdots, y^P_{sn-s+1}, y_{sn-s}, y_{sn-s-1}, \cdots, y_{sn-2s+1}, y_{sn-2s}, \cdots, y_{sn-s-r+2})^T$$

In this case Y_{new} and Y_{old} are both $(s+r-1)$-vectors and Φ is an $(s+r-1)\times(s+r-1)$ matrix which has the following form:

$$\Phi = \begin{pmatrix} Q + \sigma D_2 & & E_2 \\ \hline & \sigma G_2 + Z_2 & \\ \hline O & R & O \end{pmatrix}$$

where :

Q is an $s \times r$ matrix whose entries in the $(s+1)^{th}$ column are 1's and in the others

are zero,

D_2 is an $s \times r$ matrix whose i^{th} row consists of the coefficients in the i^{th} formula in (2.13),

E_2 is an $s \times (s-1)$ zero matrix,

G_2 is an $s \times (s+r-1)$ matrix which has the same structure as G_1, i.e.,

$$G_2(i,j) = \begin{cases} \beta^i_{j-1} & \text{for } 1 \le i \le s, \quad i \le j \le r; \\ 0 & \text{otherwise .} \end{cases}$$

Z_2 is a $s \times (s+r-1)$ matrix which has the same structure as Z_1, i.e.,

$$Z_2(i,j) = \begin{cases} 1 & \text{for } 1 \le i \le s-1, \quad j = i+1; \\ 1 & \text{for } i = s, \quad j = 2s; \\ 0 & \text{otherwise .} \end{cases}$$

R is a $(r-s-1) \times (r-s-1)$ identity matrix.

Q is a $(r-s-1) \times s$ zero matrix.

The definitions of Y_{new} and Y_{old} for Method B are the same as those used in Method A. Furthermore, the general structure of the Φ matrices is also the same in both cases. However the specifications for the entries in Φ do differ somewhat. The differences occur in the specifications for the submatrices Z_1 and Z_2. In particular, in Method B, the only non-zero entries in these submatrices occur in column $(s+1)$ which contains all 1's.

The results of a stability bound evaluation for both Methods A and B, based on these specifications for the respective Φ matrices, are provided in Figures 3, 4, and 5. In Fig.3 the dependence of σ^* on N (the number of processors) is displayed for each of three different integration orders, for the case of Method A. The equivalent behavior for Method B is shown in Fig.4. These figures reveal the tendency of σ^* to diminish with both increasing N and increasing r for both methods. Fig.5 shows the relative size of σ^* for the two methods (σ^* for Method B divided by σ^* for Method A). The superiority of Method B with repect to Method A can be observed.

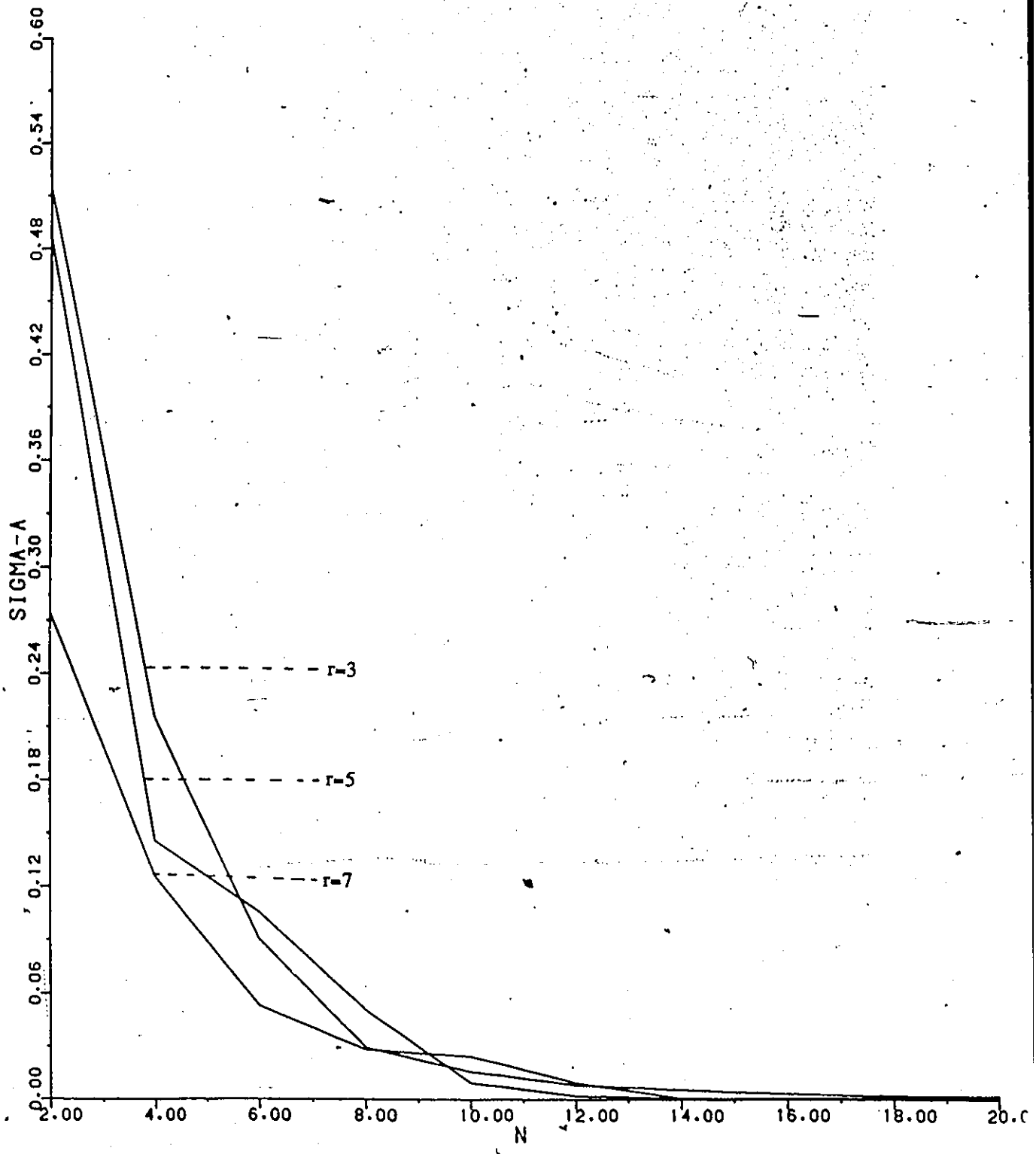
SIGMA-A
0.00
0.06
0.12
0.18
0.24
0.30
0.36
0.42
0.48
0.54
0.60
r=3
r=5
r=7
2.00
4.00
6.00
8.00
10.00
12.00
14.00
16.00
18.00
20.0
N

Figure 3

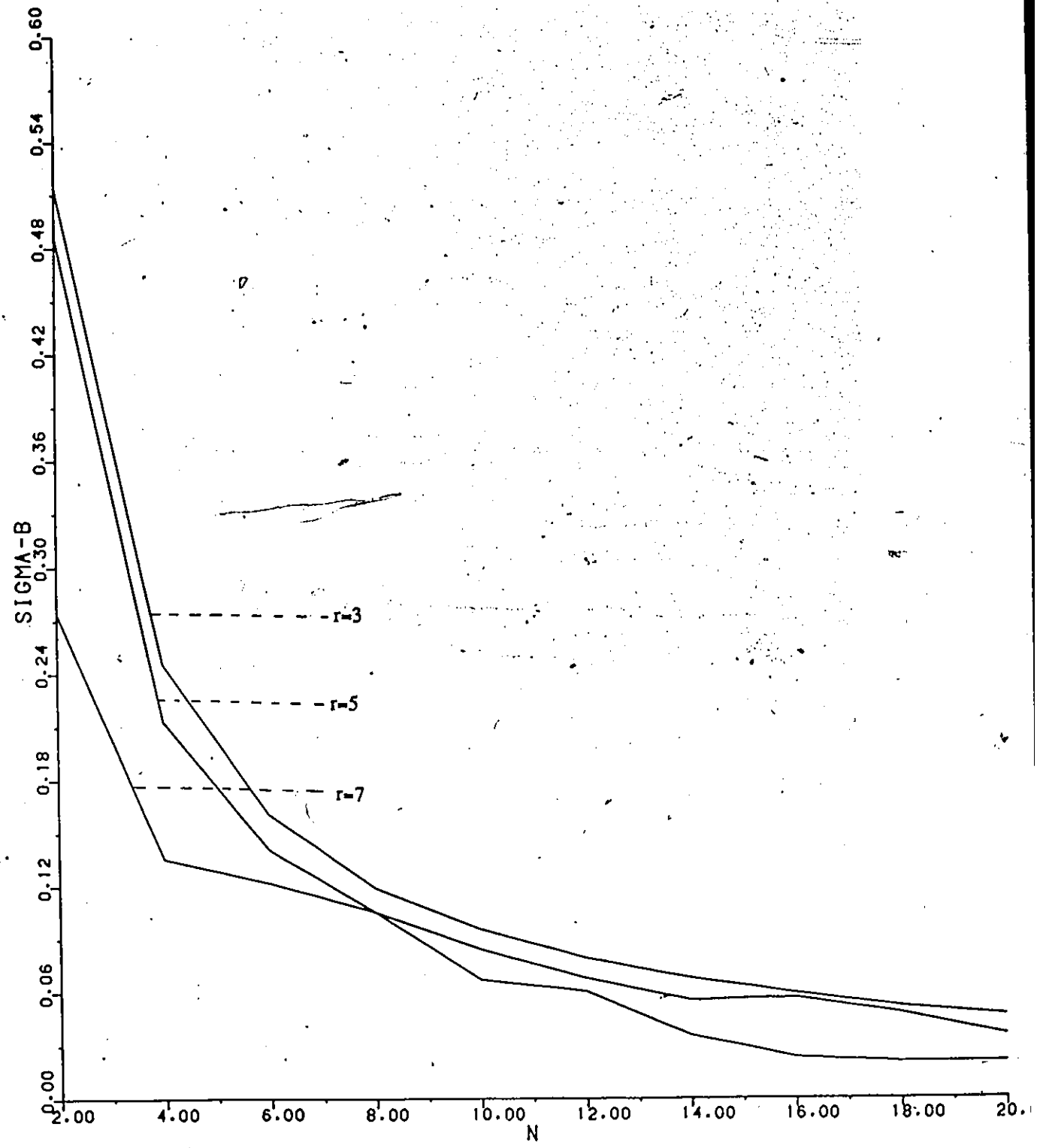
SIGMA-B
0.00
0.06
0.12
0.18
0.24
0.30
0.36
0.42
0.48
0.54
0.60
r=3
r=5
r=7
2.00
4.00
6.00
8.00
10.00
12.00
14.00
16.00
18.00
20.
N

Figure 4

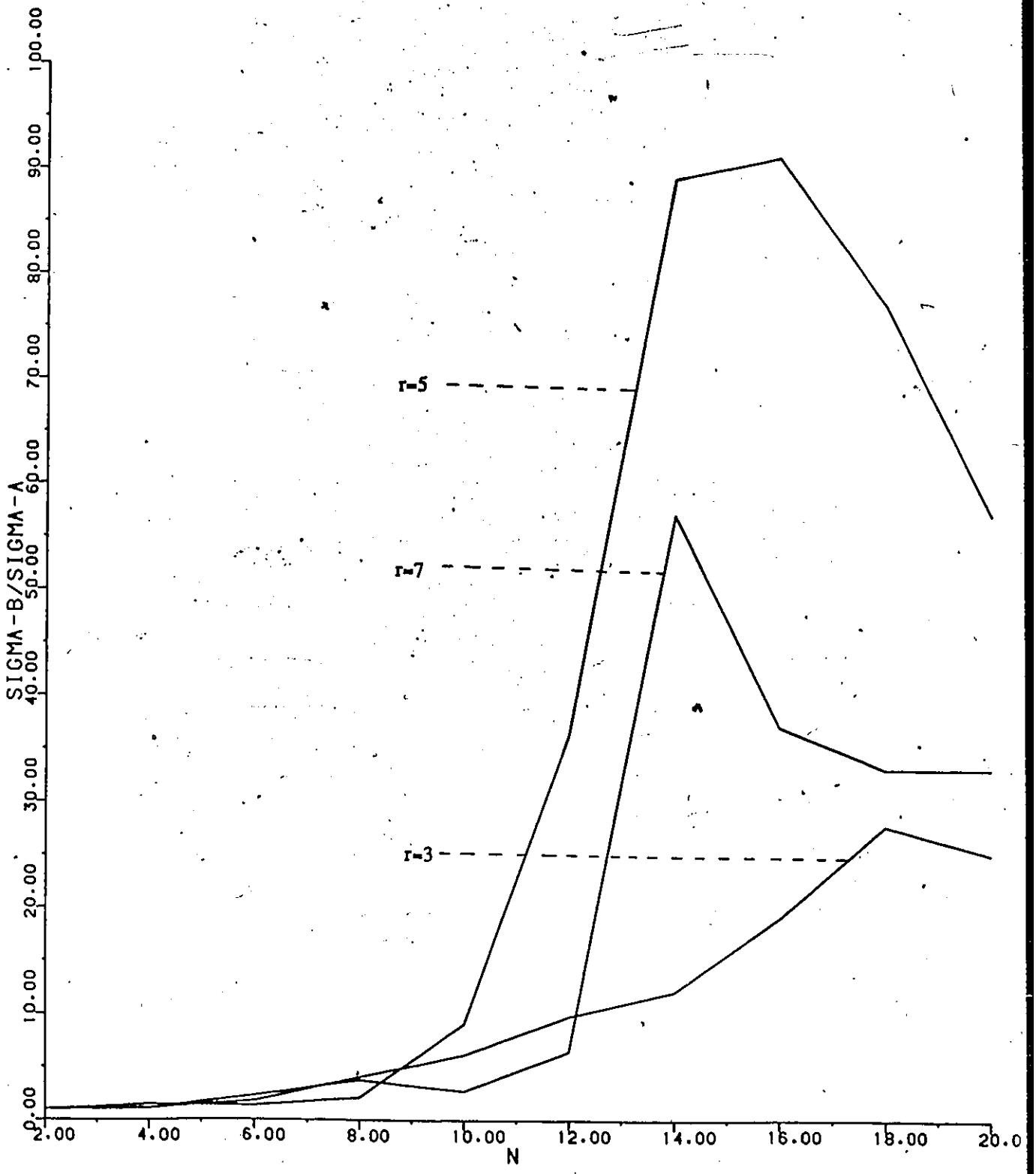
SIGMA-B/SIGMA-A
0.00
10.00
20.00
30.00
40.00
50.00
60.00
70.00
80.00
90.00
100.00
2.00
4.00
6.00
8.00
10.00
12.00
14.00
16.00
18.00
20.0
N
r=5
r=7
r=3

Figure 5

Chapter 3

Variable Stepsize Approach

3.1 Local Truncation Error

In this chapter we examine further the procedure of Method B from the point of view of formulating a variable stepsize approach. We begin by considering the local truncation error.

The predictor and corrector formulas (2.13) and (2.14) are of order r if they provide exact results for the case where the problem solution is a polynomial of degree not exceeding r. This can be achieved by setting $k = r$ in (2.13) and $k = r-1$ in (2.14) . When $y(t)$ is not a polynomial or is a polynomial of degree greater then r, then the formulas yield inexact results. The dominant error is related to the first neglected polynomial $y(t) = t^{r+1}$. On this basis, we can write

$$y(t_i) - y_i^P = T_i^P(h^{r+1}) + O(h^{r+2}) \tag{3.1}$$

$$y(t_i) - y_i = T_i(h^{r+1}) + O(h^{r+2}) \tag{3.2}$$

where y_i^P and y_i are given by the i^{th} equation of (2.13) and (2.14) respectively. The terms T_i^P and T_i are called the principal local truncation errors of (2.13) and (2.14) respectively. (The truncation error is referred to as being local because we assume that past data are exact).

From [13], we know T_i^P and T_i have the form

$$T_i^P = C_{r+1}^P h^{r+1} y^{(r+1)}(t_i) \tag{3.3}$$

$$T_i = C_{r+1} h^{r+1} y^{(r+1)}(t_i) \tag{3.4}$$

where the constants C^P_{r+1}, C_{r+1} can be established in terms of the coefficients of the integration formulas and such considerations are carried out in the following section. (We assume that the solution $y(t)$ of the initial value problem is sufficiently differentiable for (3.3) and (3.4) to be meaningful.)

From [13 pp 89–92], we know that if both the predictor and the corrector formulas have the same order, then the principal local truncation error of a predictor-corrector procedure is that of the corrector alone; i.e., T_i as given by (3.4). Furthermore it is possible to estimate this local truncation error without requiring any derivatives of $y(t)$.

By subtracting (3.1) from (3.2) and using (3.3) and (3.4) to replace T^P_i and T_i we obtain:

$$(C^P_{r+1} - C_{r+1}) h^{r+1} y^{(r+1)}(t_i) = y_i - y^P_i + O(h^{r+2})$$

By ignoring the $O(h^{r+2})$ term, it can then be established that :

$$T_i = C_{r+1} h^{r+1} y^{(r+1)}(t_i) = \frac{C_{r+1}}{C^P_{r+1} - C_{r+1}} (y_i - y^P_i) \tag{3.5}$$

This particular approximation for the local truncation error, T_i , is generally referred to as Milne's device.

Note that the expression for the principal local truncation error given by (3.5) depends on the localizing assumption. In practice, the past values of y will not be exact, and consequently it must be appreciated that, in general, (3.5) gives only an approximation to the principal local truncation error.

3.2 Stepsize Control Policy

In conventional integration procedures, changes in the integration stepsize are based on a solution quality criterion that is typically formulated in terms of the ratio

$$R_i = |T_i / \hat{T}_i| \tag{3.6}$$

where T_i is the estimation of local truncation error at t_i and $\hat{T}_i$ is user-specified local truncation error tolerance. If $R_i \leq 1$, the step can be considered to be successful; however, if $R_i > 1$, the step must be rejected and repeated with a reduced intergration stepsize. From an analysis of local truncation error behavior, it can be demonstrated that the "best" stepsize to achieve the required tolerance on the next iteration (independent of whether R_i is greater than or less than unity) is:

$$h_{new} = \sigma h_{old} \tag{3.7}$$

with $\sigma = (1/r)^\gamma$ and $\gamma = \frac{1}{r+1}$, where r is the order of the formula.

The basic requirements for the implementation of this strategy are values for T_i and $\hat{T}_i$. Milne's device, as given by eq'n (3.5), can be used to obtain T_i. A typical specification for $\hat{T}_i$ has the form:

$$\hat{T}_i = RELERR * |y_i| + ABSERR$$

where the values for the parameters $RELERR$ and $ABSERR$ are user-specified and provide the basic means for controlling solution accuracy.

A variable stepsize control strategy for the PPC method can be formulated by extending the ideas outlined above. This procedure is described in terms of three adjacent blocks, designated as B_0, B_1 and B_2, as shown in Figure 2 below:

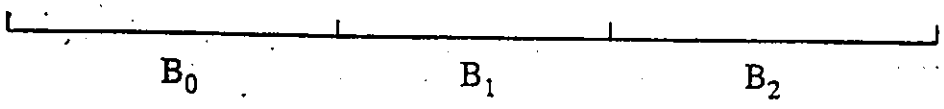

Figure 2

The stepsizes separating the s points in block B_0 are denoted by h_0, while the stepsizes in blocks B_1 and B_2 are denoted as h_1 and h_2 respectively.

We consider a state in the computation where accepted corrected values have been generated in block B_0 (i.e., the local truncation error is within acceptable

bounds) and a set of predicted values has been generated in block B_1. The next steps in the computation are as follows:

1. Compute (simultaneously) the corrected values at the points within B_1 and the predicted values at the points within B_2 using the appropriate formulas and the stepsize h_1 and h_2 repectively.

2. Using the predicted and corrected values in B_1, obtain local truncation error estimates at the s points within B_1 using (3.5) and then compute the quality criterion, R, for block B_1 as $R = \max_{1 \le i \le s} R_i$, where R_i is given by (3.6). If

   (a) $R \le 1$, the corrected values in B_1 are considered acceptable and eq'n (3.7) is used to compute a stepsize h_3 that will be used as the spacing within block B_3 which will occur to the right of B_2.

   (b) $R > 1$, the corrected values in B_1 are considered unacceptable and equation (3.7) is used to compute a stepsize $h_1' < h_1$ which then replaces the current value of h_1. A new set of predicted values is generated in block B_1, using this updated value of h_1. The procedure then returns to step 1 but with $h_2 = h_1'$. Since whenever we get new solution values, we have to compute the derivative values at these solution points, so here we have to compute the derivative values at these s new predicted values. In the meantime, the other s processors do not do useful work.

3.3 Formulation of the Variable Stepsize PPC Method

The general formula for a variable stepsize linear method can be written as follows:

$$y_u = \sum_{j=1}^{k} \alpha_j y_{u-m_j} + \sum_{j=0}^{k} h_j \beta_j f_{u-n_j} \tag{3.8}$$

where the m_j, n_j, are non-negative integers which satisfy:
(i) $m_i < m_j$ for $i < j$

(ii) $n_i < n_j$ for $i < j$

(iii) $m_1 \neq 0$, $n_1 \neq 0$

(iv) $\beta_0 n_0 = 0$

and h_j is the stepsize which is generally different from one block to another. More specifically, h_j is defined as follows:

$$h_j = t_{u-n_j+1} - t_{u-n_j}$$

The method defined by (3.8) is said to be explicit if $\beta_0 = 0$ and implicit if $\beta_0 \neq 0$.

The free parameters in (3.8) are represented by the vectors

$$\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_k)^T$$

$$\beta = (\beta_\mu, \beta_{\mu+1}, \ldots, \beta_k)^T$$

We define $\bar{h}$ to be an s-vector as follow:

$$\bar{h} = (h_\mu, h_{\mu+1}, \ldots, h_k)$$

where $\mu = 1$ if (3.8) is explicit and $\mu = 0$ if (3.8) is implicit. For (3.9) to represent an r^{th} order formula, we must have it exactly satisfied for:

(i) $y(t) = 1$, (and hence : $y'(t) = 0$)

(ii) $y(t) = t^{m+1}$, (and hence: $y'(t) = (m+1)t^m$), $m = 0, 1, \ldots, (r-1)$

From (i) we get:

$$\sum_{j=1}^{k} \alpha_j = 1$$

or equivalently,

$$v_0^T \alpha = 1 \tag{3.9}$$

where v_0 is a k-vector whose entries are all 1's.

From(ii): if $m = 0$, i.e., $y(t) = t, y'(t) = 1$, then we get

$$y(t_u) = \sum_{j=1}^{k} \alpha_j y(t_u - H_{m_j}) + \sum_{j=0}^{k} h_j \beta_j y'(t_u - H_{n_j})$$

$$t_u = \sum_{j=1}^{k} \alpha_j (t_u - H_{m_j}) + \sum_{j=0}^{k} h_j \beta_j$$

Setting $t_u = 0$, yields:

$$\sum_{j=1}^{k} \alpha_j H_{m_j} - \sum_{j=0}^{k} h_j \beta_j = 0$$

and H_i is the length between t_i and t_u, i.e.,

$$H_i = \sum_{j=i}^{u-1} h_j$$

Considering now the case where $1 \leq m \leq (r-1)$, we get:

$$y(t_u) = \sum_{j=1}^{k} \alpha_j (t_u - H_{m_j})^{m+1} + \sum_{j=0}^{k} h_j \beta_j (m+1)(t_u - H_{n_j})^m$$

Setting $t_u = 0$ gives:

$$\sum_{j=1}^{k} \alpha_j H_{m_j}^{m+1} - (m+1) \sum_{j=0}^{k} h_j H_{n_j}^{m} \beta_j = 0 \quad for \quad m = 1, 2, \ldots, (r-1)$$

We can write this formula in matrix form:

$$V\alpha - D\bar{V}W\beta = 0 \tag{3.10}$$

where

$$V = (v_1, v_2, \ldots, v_k)$$
$$\bar{V} = (\bar{v}_\mu, \bar{v}_{\mu+1}, \ldots, \bar{v}_k)$$

D is a diagonal $r \times r$ matrix with $D_{jj} = j$, and W is also a diagonal $r \times r$ matrix with $W_{jj} = \bar{h}_j$ and

$$v_j = (H_{m_j}, H_{m_j}^2, \ldots, H_{m_j}^r)^T \quad 1 \leq j \leq k$$

$$\bar{v}_j = (1, H_{n_j}, H_{n_j}^2 \ldots, H_{m_j}^{r-1})^T \quad 0 \leq j \leq k$$

From the discussion in section (2.3.2) we know the unique value for β can be obtained by taking $k = r$ if the method is explicit (i.e., $\mu = 1$) or by taking $k = r-1$ if the method is implicit (i.e., $\mu = 0$).

According to (2.3.2) and equation (3.8), the predictor formula can be written as:

$$y_u^P = y_{u-m_1} + \sum_{j=1}^{k} h_j \beta_j f_{u-n_j} \tag{3.11}$$

where

$$u = (n+1)s - i + 1$$

$$m_1 = 2s - i + 1$$

$$n_j = s + j - i$$

and

$$1 \leq i \leq s$$

The corrector formula can be written as:

$$y_u = y_{u-m_1} + \sum_{j=0}^{k} h_j \beta_j f_{u-n_j} \tag{3.12}$$

where

$$u = ns - i + 1$$

$$m_1 = s - i + 1$$

$$n_j = j$$

and

$$1 \leq i \leq s$$

Recall that according to [13 pp. 23], the constant C_{r+1}^P for an individual equation in 3.11 can be written as follow:

$$C_{r+1}^P = \frac{1}{r!} H_{m_1}^r - \frac{1}{(r-1)!} \sum_{j=1}^{k} h_j H_{n_j}^{r-1} \beta_j$$

and the constant C_{r+1} for an individual equation in (3.12) can be written as:

$$C_{r+1} = \frac{1}{r!} H_{m_1}^r - \frac{1}{(r-1)!} \sum_{j=0}^{k} h_j H_{n_j}^{r-1} \beta_j$$

Note the difference between the above two formulas and the fixed stepsize formulas (2.13) and (2.14). In (3.11) and (3.12) the stepsize h is subscripted because it is generally different in different intervals. Recall that in this approach, the stepsize within a block is always the same, but the stepsize in different blocks can be different. The β coefficients in (3.11) and (3.12), for any order r, can be obtained from (3.10).

Note also that in (3.10), h_j and H_j are involved and their values can be different in different blocks, but because H_j is the length between t_j and t_u, it can be expressed in terms of the entries in the vector $\bar{h}$. Consequently the β-coefficients are functions only of $\bar{h}$, i.e., $\beta_j = \beta_j(\bar{h})$.

Chapter 4

Experimental Evaluation

4.1 Scope of the Experiments

The testing activity which was carried out was concerned with evaluating the performance of Methods A and B operating in a fixed stepsize mode and with the performance of Method B operating in a variable stepsize mode. To facilitate the subsequent discussion, we denote these cases as Methods A/F, B/F and B/V respectively. The variable stepsize procedure outlined in Chapter 3 was used in the latter case. Experiments were carried out over a range of values for N (number of available processors), r (integration order) and G (admissible global solution error).

The experiments were carried out on five test problems which are listed in Appendix 3. These problems share the feature of having an explicit closed form solution which provided the basis for determining global solution error in the experiments.

All results were obtained using a suite of Fortran programs running on an Amdahl 470/V7 using double precision arithmetic.

4.2 Comparison Criteria

The focus of the fixed stepsize experiments was on speedup and efficiency. The speedup, $S(N)$, of an N processor configuration is taken to be the relative solution

speed provided by that configuration as compared to the solution speed of a single processor solution. Several important factors influence the value of $S(N)$. These include:

(a) the criterion used to measure solution speed

(b) the reference single processor method used for comparison

(c) the order of the formulas being used

(d) the required global solution accuracy

The efficiency, $E(N)$, of an N processor configuration corresponds to a normalization of $S(N)$; i.e. $E(N) = S(N)/N$.

The measure of solution speed used in the experiments was taken to be the number of derivative function evaluations (dfe's) per processor over the prescribed solution interval. This approach offers several advantages which include: implementation simplicity, circumvention of distortions caused by differences in coding style and repeatability of results over different computer hardware. We note also that this approach does tend to become increasingly more realistic when the complexity of the derivative functions increases and/or the size of the model (as measured by the number of ode's) increases. In terms of this solution speed measure, the specification for speedup, $S(N)$, becomes:

$$S(N) = \frac{\text{def's for single processor solution}}{\text{def's, per processor, for parallel method}}$$

The single processor reference solution required for the evalution of $S(N)$ was obtained using an Adams-Bashforth/Adams-Moulton predictor-corrector method of order equal to that of the parallel method being used. For this purpose, the family of formulas presented by Lapidus [12 pp. 180–181] was used. Six different integration orders were considered in the experiments; namely $r = 3, 4, 5, 6, 7$ and 8 and the corresponding single processor formulas used in the reference experiments, are given in Appendix 2.

The experiments were organized to examine performance at three different solution accuracies corresponding roughly to low, moderate and high. Because the

testing was restricted to problems with a known closed form solution, it was feasible to explicitly measure global error over the solution interval. Test results were obtained using an iterative procedure which was designed to provide solutions whose global error was within a prescribed tolerance of some specified value, G. The iteration was based on the adjustment of some parameter and terminated only when a solution with global error in the interval $[G/2, G]$ was obtained. In the case of the fixed stepsize experiments, the relevant parameter was the integration stepsize h while in the variable stepsize experiments the parameter was the parameter $RELERR$ in the specification of the local truncation error tolerance $\hat{T}_i$ (with $ABSERR = RESERR$). This approach provided a meaningful way for ensuring the comparability of the data used in the performance comparisons. The three values used for G were: $10^{-3}, 10^{-5}$ and 10^{-7}.

4.3 Test Results

The results of the experiments are summarized in Tables 1–3 and Figures 6–41. These tables show the number of derivative function evalutions per processor over five test problems for the three different procedures (i.e., Method A/F, B/F, and B/V). Results for each test problem are presented for 6 different integration orders, 6 different numbers of processors (in the range 2 through 12) and 3 different solution accuracies. Table 4 shows the results of the performance of the single processor reference cases.

Each of the Figures 6 through 41 shows average behavior over the set of test problems displayed as a function of the number of processors being used (N). Figures 6–23 provide average $S(N)$ values obtained as the average of the $S(N)$ values for the five individual test problems. The values shown in Figure 24–41 are average $E(N)$ values. Each of the Figures shows two curves designated respectively with the symbol Δ and x. The former indicates results from experiments with Method A/F while the latter indicates results from experiments with Method B/F. Each of these Figures consolidates results from a collection of experiments that have a common value for order (r) and global target error specification (G). The values for

these parameters are indicated on each figure in the format (e, r) where $G = 10^{-e}$.

4.4 Discussion of the Results

We first discuss the results obtained in the fixed stepsize experiments. (i.e., with methods A/F and B/F).

From Tables 1 and 2, the effect of changes in integration order can be observed. As order increases from 3 to 6, the number of dfe's per processor usually decreases for both Methods A/F and B/F. But there are some exceptions for Method A/F. From Table 1, it can be observed that when $N = 10$ or 12, the number of dfe's increases as the order increases from 3 to 5. For Method A/F, the results tend to be worse when the order goes higher than 6. In constrast, the number of dfe's with Method B/F continuously decreases even when the order goes to 8.

Figures 6 through 41 provide a particularly useful indication of the relative performance of Methods A/F and B/F, both in terms of speed-up and in terms of efficiency. It is interesting, furthermore, to observe some correlation between the relative size of the stability bounds for these two methods (as given in Figure 4) and their (average) behavior on the test problems. In particular, Figure 4 indicates the general superiority of Method B over Method A and this superiority is particularly pronounced when N is in the range 10 to 16. From the test results in Figures 6-41, we note that Method A/F sometimes performs better than Method B/F for small values of N. However when N is large ($N \geq 8$) Method B/F always provides superior performance and this feature correlates with the superior stability bound for large N.

A particular anomaly that can be observed in the test results appears in Figures 24-41; namely, the frequent occurrence of an efficiency value that is greater than one (see, in particular, Fig.30, 36). Since this tends to occur for low integration order, this phnomenon could suggest that the reference single processor predictor-corrector method being used for the $r = 3$ case, is fundamentally inefficient. We note also that, for both Methods A/F and B/F the efficiency decreases as N increases suggesting a "diminishing return" phenomenon.

The speedup characteristics for both Methods A and B generally tend to increase initially and then undergo a relatively severe decrease. The speedup of Method A usually decreases after N goes higher than 6, and sometimes even when N exceeds 4. For method B, if the order is greater than 4, the speedup keeps increasing over the whole range of N from 2 to 12, but the relative amount of this increase becomes very small after N exceeds 8.

For each value of r, the speedup curves for increasing accuracy (decreasing G) are remarkably similar. A generally valid conclusion seems to be that Method B provides better speedup for larger values of r and N, independent of solution accuracy.

Table 3 shows the results obtained in the variable stepsize experiments (i.e., using Method B/V). If we compare these results with those obtained with Method B/F, it can be seen that in some cases, Method B/V can perform better than Method B/F. However in many cases, it performs worse than Method B/F. This may be because whenever a step is rejected, the variable procedure must recompute all N values, and also waste s derivative function evalutions as was explained in section 3.2. The relative behavior of the fixed and variable stepsize approach is different for different test problems and different cases. A comparision of the data in Tables 2 and 3 shows that the best performance of Method B/V relative to Method B/F occurs with problem TP4.

	N \ G	r=3			r=4			r=5		
		10^{-3}	10^{-5}	10^{-7}	10^{-3}	10^{-5}	10^{-7}	10^{-3}	10^{-5}	10^{-7}
TP1	2	127	574	3202	115	289	803	87	170	448
	4	86	252	669	74	145	310	58	108	202
	6	85	335	1335	69	151	336	58	105	169
	8	101	401	2001	94	168	501	52	98	249
	10	125	459	2135	230	275	401	210	230	249
	12	179	535	2135	537	567	594	798	826	855
TP2	2	336	2004	6402	253	1606	2289	170	367	803
	4	202	669	2002	127	288	802	86	145	336
	6	268	890	3557	105	245	669	91	151	256
	8	251	1001	4001	227	251	801	93	156	288
	10	268	1068	5121	604	622	1068	555	570	587
	12	397	1068	5335	1247	1270	1299	1916	1938	1958
TP3	2	1252	5002	26669	316	837	2003	155	559	1432
	4	252	457	1669	159	360	836	141	230	502
	6	557	2668	13335	169	373	955	169	225	336
	8	835	3335	16001	406	436	1001	147	211	336
	10	801	4001	20410	1037	1057	1163	1017	1032	1048
	12	861	4446	20195	2112	2138	2195	3342	3361	3381
TP4	2	962	3431	16002	403	1337	4003	303	670	1718
	4	402	1202	4002	242	548	1502	187	313	752
	6	459	1601	8001	180	402	1145	162	288	502
	8	501	2401	9601	446	501	1501	160	302	548
	10	481	2401	12801	1464	1537	1921	1424	1475	1525
	12	713	2668	12801	3330	3392	3513	5425	5507	5655
TP5	2	336	1336	6402	203	575	1603	157	337	803
	4	184	502	1602	140	269	669	163	184	402
	6	224	668	3049	135	193	536	201	207	299
	8	394	835	3848	549	563	623	167	178	306
	10	655	1001	4925	1472	1490	1508	1347	1363	1379
	12	977	1113	5130	2975	2999	3021	4950	4608	4632

Table 1 Behavior of Method A/F

	N \ G	r=6			r=7			r=8		
		10^{-3}	10^{-5}	10^{-7}	10^{-3}	10^{-5}	10^{-7}	10^{-3}	10^{-5}	10^{-7}
TP1	2	81	158	290	76	129	227	77	123	205
	4	55	86	169	58	86	145	70	80	128
	6	59	81	151	75	82	124	98	103	118
	8	58	77	145	82	89	120	119	127	133
	10	59	75	136	91	95	113	158	161	162
	12	567	574	598	89	103	124	139	147	151
TP2	2	116	254	576	122	215	368	105	154	255
	4	93	145	269	93	108	169	143	150	163
	6	117	124	208	155	162	169	217	224	231
	8	116	132	184	178	185	195	264	270	278
	10	688	154	167	199	207	214	363	371	379
	12	1392	1415	1436	245	261	280	336	343	354
TP3	2	161	317	560	136	190	389	133	172	283
	4	114	159	280	156	175	230	246	254	263
	6	212	223	269	283	291	302	395	402	411
	8	191	199	207	304	311	319	462	469	477
	10	274	289	304	404	413	422	718	726	734
	12	2557	2576	2593	541	558	570	567	571	580
TP4	2	204	433	862	204	357	671	165	273	467
	4	144	275	502	168	202	318	313	339	362
	6	231	249	366	354	372	402	547	574	599
	8	253	269	355	426	443	473	694	724	755
	10	272	308	355	506	536	563	1002	1040	1069
	12	3912	4017	4131	623	669	730	933	972	1018
TP5	2	145	255	504	167	200	368	188	194	291
	4	159	164	296	219	225	230	343	349	355
	6	274	281	853	371	378	384	523	530	537
	8	271	278	371	426	433	524	637	646	653
	10	387	399	412	477	483	490	864	874	879
	12	3310	3328	3347	689	708	723	818	827	835

Table 1 Behavior of Method A/F (continued)

	N \ G	r=3			r=4			r=5		
		10^{-3}	10^{-5}	10^{-7}	10^{-3}	10^{-5}	10^{-7}	10^{-3}	10^{-5}	10^{-7}
TP1	2	127	574	3202	115	289	803	87	170	448
	4	102	288	802	86	202	502	65	114	252
	6	85	224	1335	77	169	447	58	105	225
	8	73	335	2001	73	168	401	58	102	202
	10	74	401	2135	68	161	401	55	101	179
	12	85	446	2135	68	150	382	57	97	192
TP2	2	336	2004	6402	253	1606	2289	170	367	803
	4	288	802	2669	145	336	891	127	252	574
	6	268	668	2668	136	299	764	105	225	447
	8	251	668	4001	126	251	668	86	202	402
	10	230	801	4268	116	184	1068	81	179	401
	12	224	890	5335	113	335	1187	80	168	382
TP3	2	1252	5002	26669	316	837	2003	155	559	1432
	4	360	912	2502	230	558	1431	159	280	457
	6	557	2668	13335	211	479	1336	154	259	479
	8	835	3335	16001	180	456	1113	141	252	457
	10	801	4001	16668	183	401	1113	144	237	418
	12	835	4446	20835	168	418	1030	140	224	349
TP4	2	962	3431	16002	403	1337	4003	303	670	1718
	4	482	1502	4802	269	548	1502	202	431	926
	6	535	2001	8001	269	402	1336	202	366	802
	8	601	2401	9601	251	501	1501	169	301	752
	10	601	2401	12801	220	401	1921	173	287	687
	12	668	2668	12801	224	573	2001	163	287	668
TP5	2	336	1336	6402	203	575	1603	157	337	803
	4	202	669	2002	145	336	802	120	225	502
	6	224	668	3049	136	299	754	119	193	413
	8	221	835	3848	119	271	623	120	182	382
	10	236	1001	4925	119	239	685	119	177	366
	12	241	1113	5130	119	221	865	119	181	334

Table 2 Behavior of Method B/F

	N \ G	r=6			r=7			r=8		
		10^{-3}	10^{-5}	10^{-7}	10^{-3}	10^{-5}	10^{-7}	10^{-3}	10^{-5}	10^{-7}
TP1	2	81	158	290	76	129	227	77	123	205
	4	55	102	202	57	86	169	67	94	146
	6	53	86	180	54	86	151	58	77	124
	8	52	86	169	52	69	136	54	74	114
	10	52	79	162	52	68	136	54	69	109
	12	49	76	150	52	68	129	54	69	105
TP2	2	116	254	576	122	215	368	105	154	255
	4	93	169	288	84	136	252	113	121	185
	6	91	151	269	69	105	193	96	224	159
	8	86	145	252	69	102	156	69	102	151
	10	82	136	249	69	102	155	66	97	142
	12	80	135	244	69	105	159	66	91	136
TP3	2	161	317	560	136	190	389	133	172	283
	4	116	211	387	138	345	252	189	198	221
	6	107	169	336	110	154	225	158	165	183
	8	107	169	287	107	150	218	91	111	159
	10	108	156	288	102	145	213	91	110	136
	12	106	153	258	104	141	211	91	110	136
TP4	2	204	433	862	204	357	671	165	273	467
	4	174	302	548	136	233	402	232	253	303
	6	155	269	473	136	202	336	202	225	288
	8	144	252	431	129	202	302	114	152	252
	10	140	242	439	122	194	302	115	157	242
	12	125	237	423	117	193	299	121	156	225
TP5	2	145	265	504	167	200	368	188	194	291
	4	135	178	310	191	197	233	272	278	184
	6	122	151	269	140	145	203	233	238	244
	8	122	149	256	124	128	195	141	146	161
	10	122	146	246	124	128	183	125	129	150
	12	121	140	234	124	128	182	124	129	149

Table 2 Behavior of Method B/F (continued)

	N \ G	r=3			r=4			r=5		
		10^{-3}	10^{-5}	10^{-7}	10^{-3}	10^{-5}	10^{-7}	10^{-3}	10^{-5}	10^{-7}
TP1	2	121	645	2927	138	269	661	123	287	618
	4	174	455	1313	112	261	535	86	142	284
	6	132	428	2167	104	371	832	112	139	250
	8	96	482	1788	101	190	414	105	158	368
	10	119	507	2011	73	233	419	80	127	213
	12	116	501	2414	107	209	353	72	116	200
TP2	2	376	1012	5741	241	759	2293	204	362	744
	4	503	1658	5621	157	374	925	156	312	554
	6	341	863	2883	181	365	1416	167	262	497
	8	257	625	4114	149	277	589	151	314	722
	10	289	745	4125	151	417	1195	130	222	439
	12	257	958	4514	133	351	1096	131	228	442
TP3	2	1101	5186	24623	335	830	2054	223	547	1443
	4	533	1454	4327	257	605	1429	206	294	474
	6	713	3284	15663	332	975	2389	232	287	524
	8	801	3726	14853	223	439	1096	181	376	502
	10	927	4242	17653	219	651	1268	162	281	502
	12	904	3865	20477	192	392	973	152	255	521
TP4	2	664	2611	12405	290	860	2665	243	489	1113
	4	601	1840	5407	207	463	1129	204	327	689
	6	473	1718	7552	305	783	1807	219	349	638
	8	409	1864	7463	184	366	939	157	371	763
	10	498	2087	12006	198	426	1264	155	283	555
	12	456	2117	8665	182	444	1317	143	300	533
TP5	2	299	1139	5043	225	485	1341	181	350	778
	4	261	799	2849	199	388	884	201	295	544
	6	247	719	3041	198	396	1048	280	334	604
	8	230	735	3088	153	282	581	185	257	528
	10	250	828	3771	160	273	722	135	231	391
	12	283	851	3578	153	242	704	137	213	413

Table 3 Behavior of Method B/V

	N \ G	r=6			r=7			r=9		
		10^{-3}	10^{-5}	10^{-7}	10^{-3}	10^{-5}	10^{-7}	10^{-3}	10^{-5}	10^{-7}
TP1	2	96	216	372	104	176	312	93	180	256
	4	77	137	260	91	117	200	89	125	189
	6	115	143	242	125	150	194	133	166	178
	8	146	166	406	175	216	284	172	201	284
	10	79	128	230	126	219	334	200	237	308
	12	64	94	180	97	131	204	237	257	359
TP2	2	144	254	574	151	233	379	115	177	231
	4	101	187	338	106	164	269	138	143	202
	6	157	153	298	158	232	267	197	212	280
	8	138	193	425	177	243	391	231	239	310
	10	118	190	351	202	299	436	293	314	376
	12	88	160	259	144	165	273	352	373	406
TP3	2	190	313	621	148	212	428	161	205	317
	4	146	240	414	175	206	323	237	243	269
	6	198	202	379	208	226	275	383	338	434
	8	244	372	410	263	320	445	385	378	372
	10	178	198	425	330	312	446	457	490	579
	12	94	187	295	239	260	310	532	439	666
TP4	2	186	318	564	136	285	467	148	224	342
	4	161	236	405	173	169	319	235	233	243
	6	202	241	360	230	322	377	378	362	350
	8	212	347	535	290	355	447	365	362	427
	10	183	224	382	368	379	501	478	529	575
	12	99	199	319	275	288	300	551	600	597
TP5	2	154	269	470	181	246	372	266	274	317
	4	209	217	355	272	283	288	356	353	355
	6	299	319	395	320	359	450	631	589	576
	8	374	387	573	402	460	443	566	571	521
	10	281	274	365	489	448	523	704	644	658
	12	128	162	288	408	386	405	759	720	759

Table 3 Behavior of Method B/V (continued)

	r=3			r=4			r=5		
G	10^{-3}	10^{-5}	10^{-7}	10^{-3}	10^{-5}	10^{-7}	10^{-3}	10^{-5}	10^{-7}
TP1	297	1115	6411	212	584	1612	181	377	903
TP2	733	2015	12815	428	1016	4588	377	745	1617
TP3	2511	10011	53345	638	1680	4012	431	1125	2817
TP4	1615	7695	30735	816	2760	7696	617	1389	3217
TP5	607	2691	12823	394	1078	3224	307	663	1541
	r=6			r=7			r=8		
G	10^{-3}	10^{-5}	10^{-7}	10^{-3}	10^{-5}	10^{-7}	10^{-3}	10^{-5}	10^{-7}
TP1	158	300	586	159	265	487	160	266	416
TP2	280	464	686	225	419	747	226	328	466
TP3	292	570	1068	225	329	849	210	350	572
TP4	456	892	1618	363	705	1299	364	586	936
TP5	262	492	904	273	415	727	352	366	588

Table 4 Behavior of Reference Method (single processor, fixed stepsize)

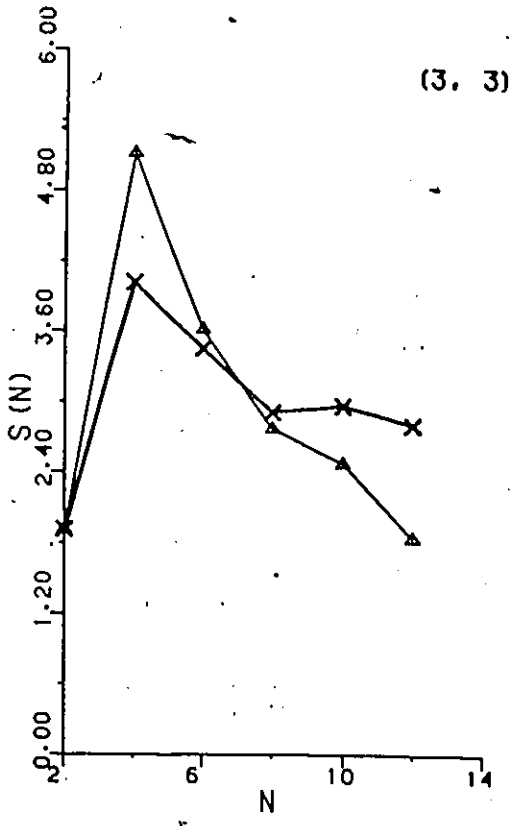

Figure 6

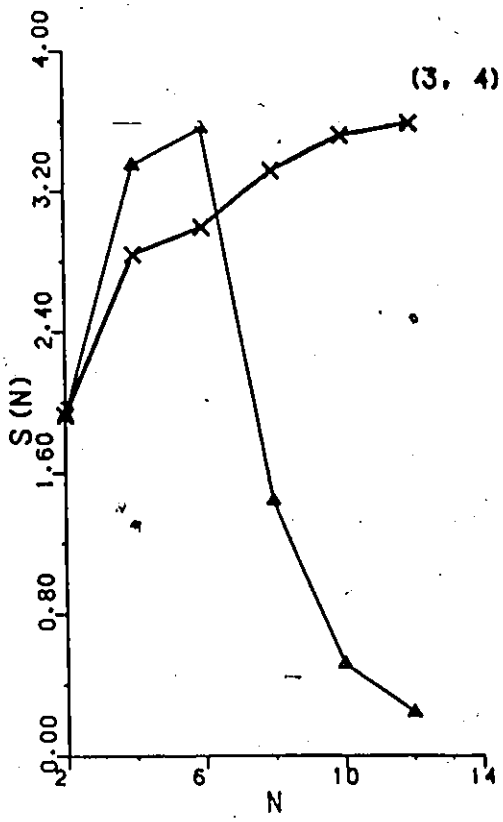

Figure 7

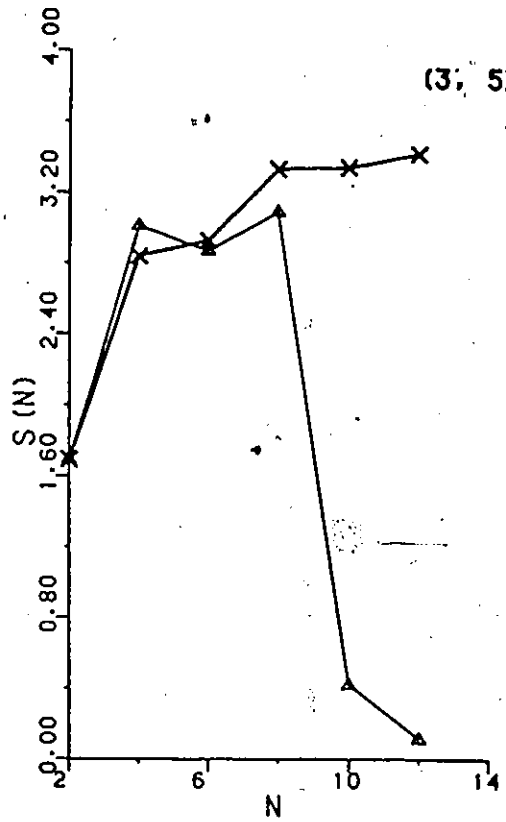

Figure 8

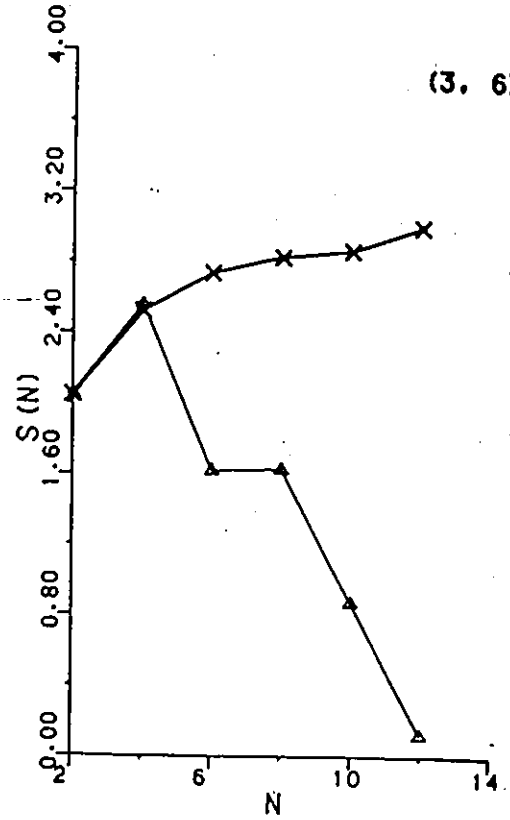

Figure 9

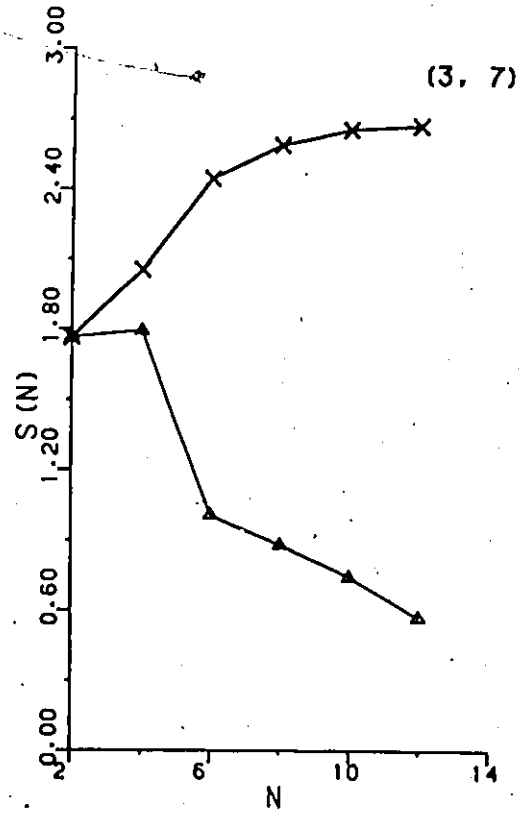

Figure 10

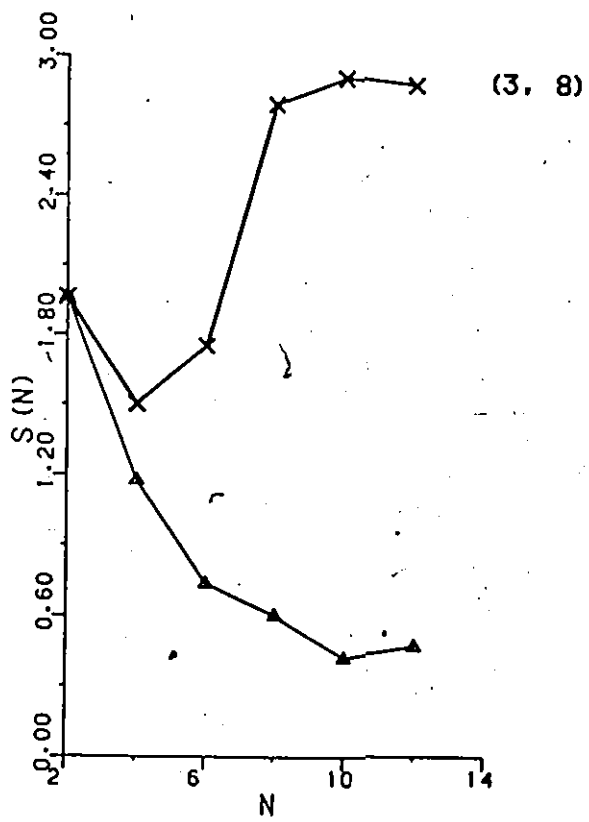

Figure 11

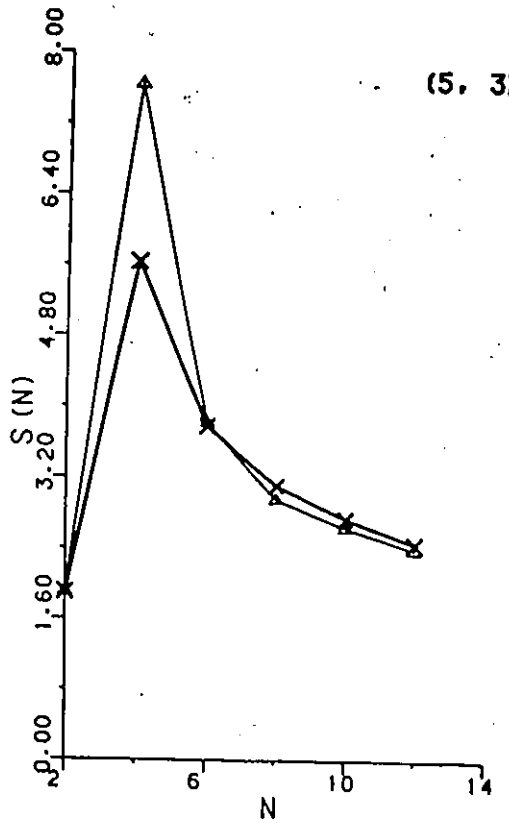

Figure 12

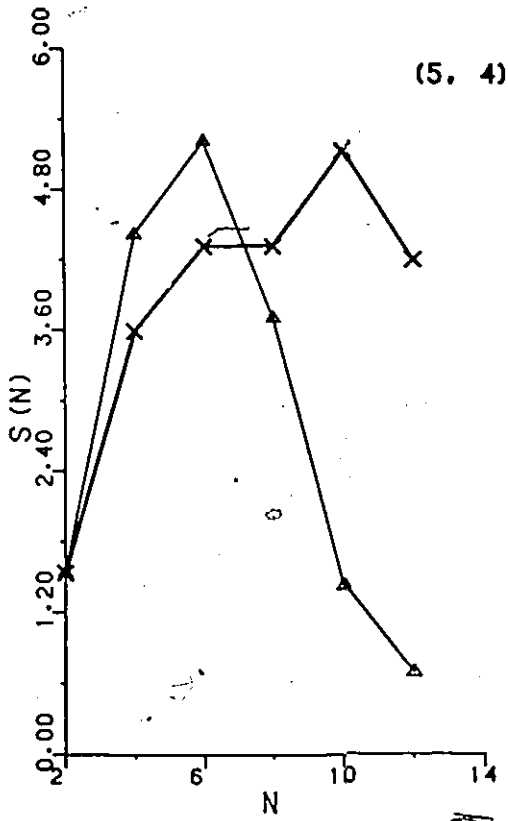

Figure 13

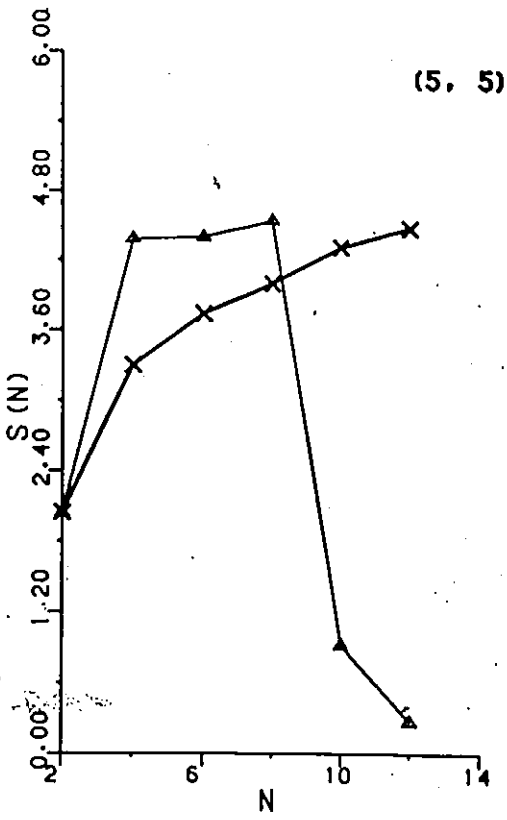

Figure 14

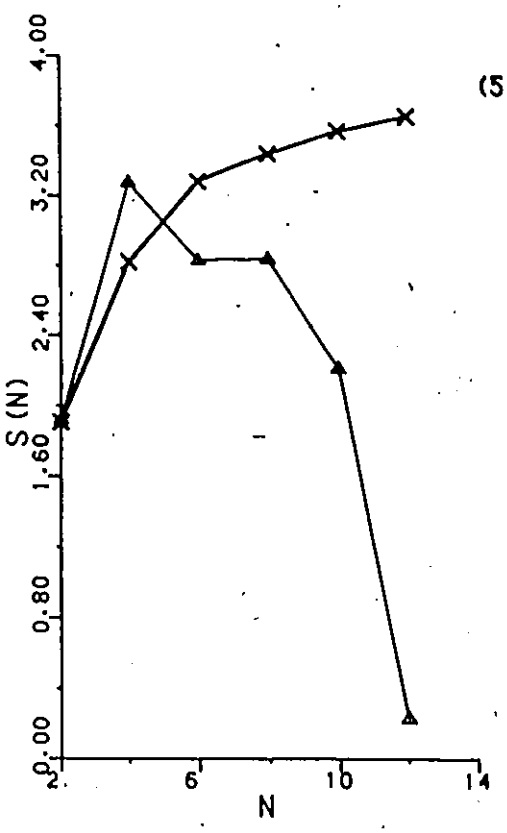

Figure 15

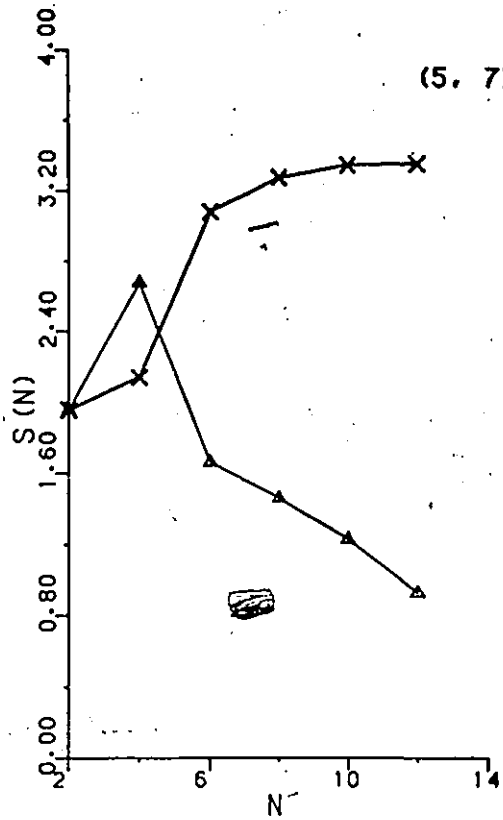

Figure 16

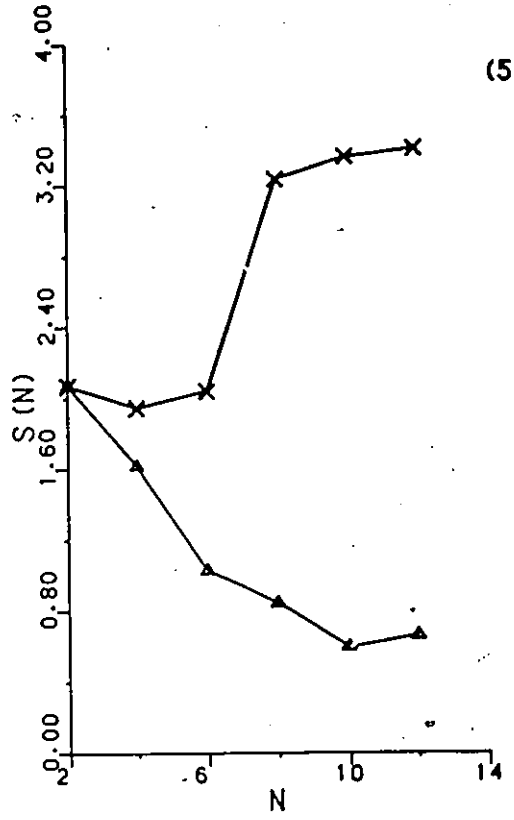

Figure 17

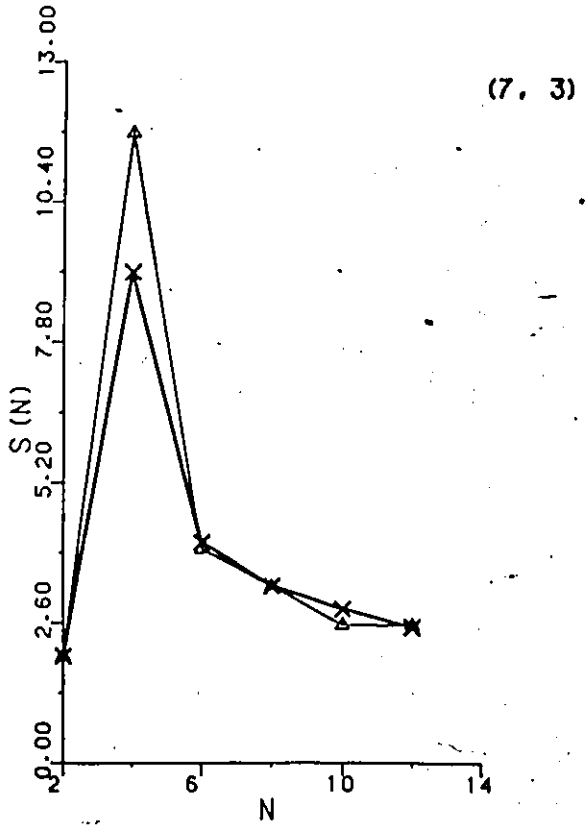

Figure 18

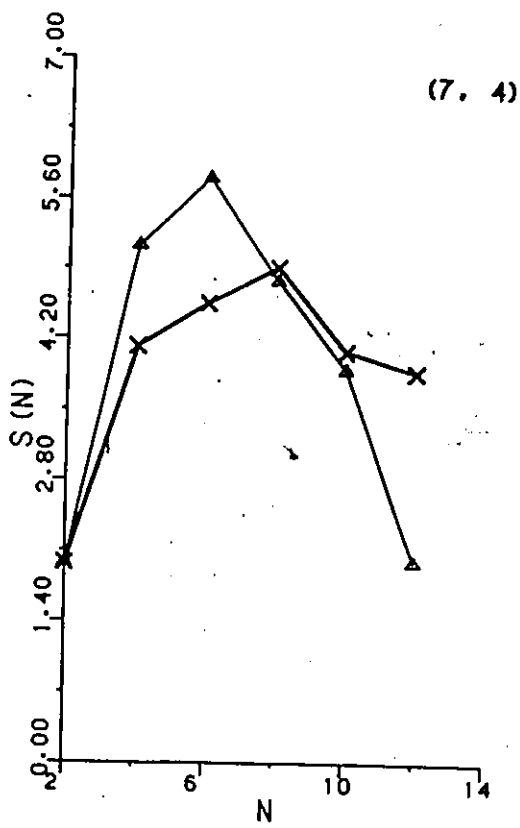

Figure 19

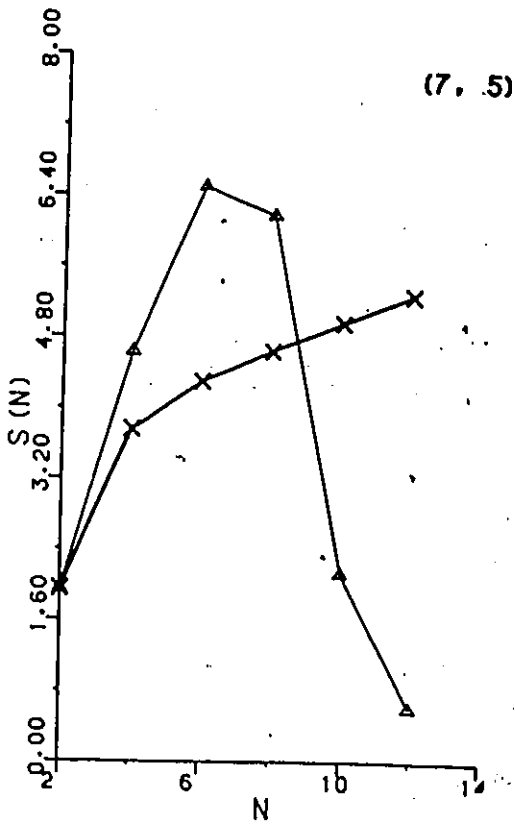

Figure 20

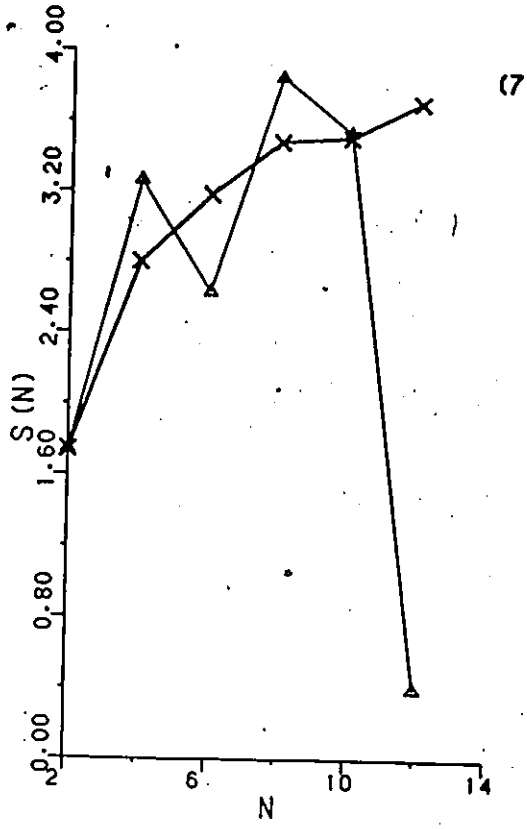

Figure 21

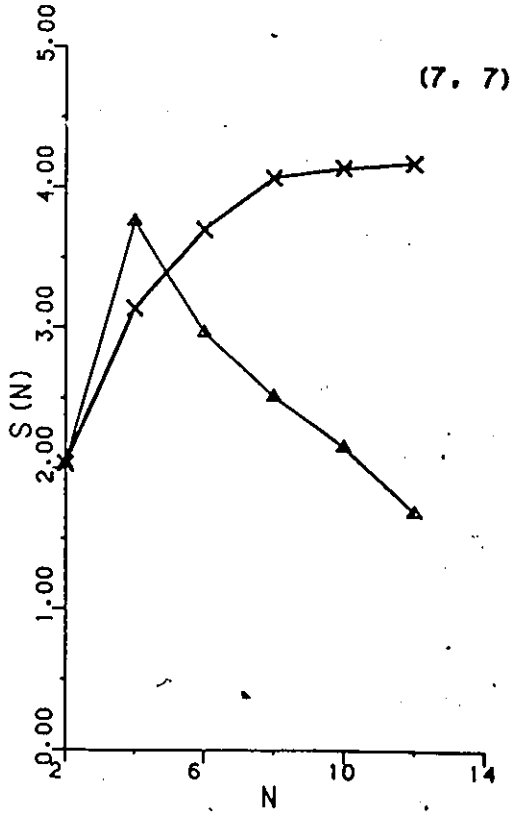

Figure 22

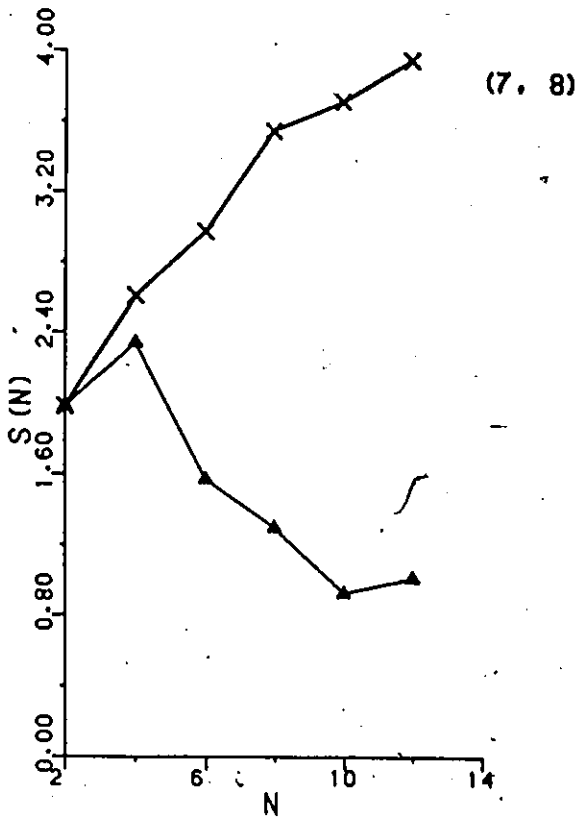

Figure 23

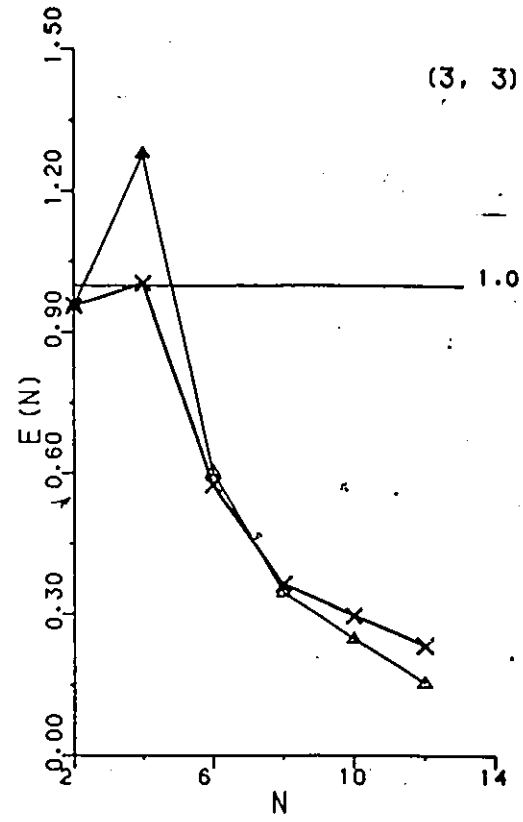

Figure 24

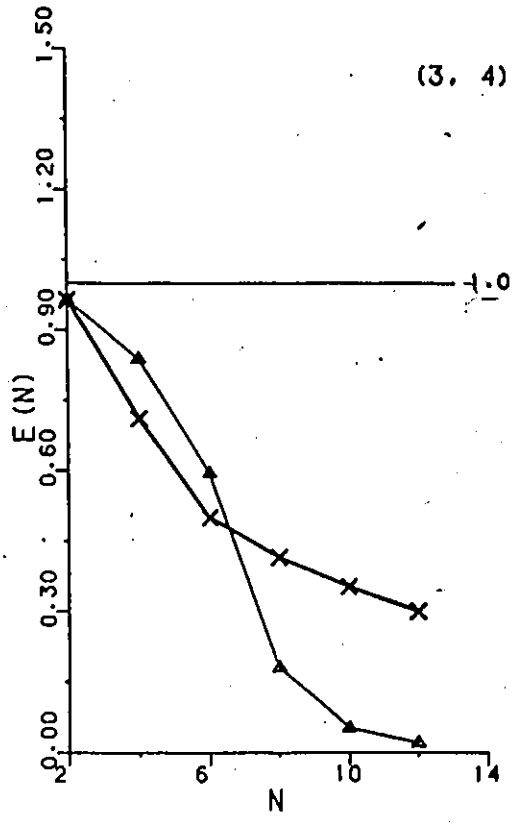

Figure 25

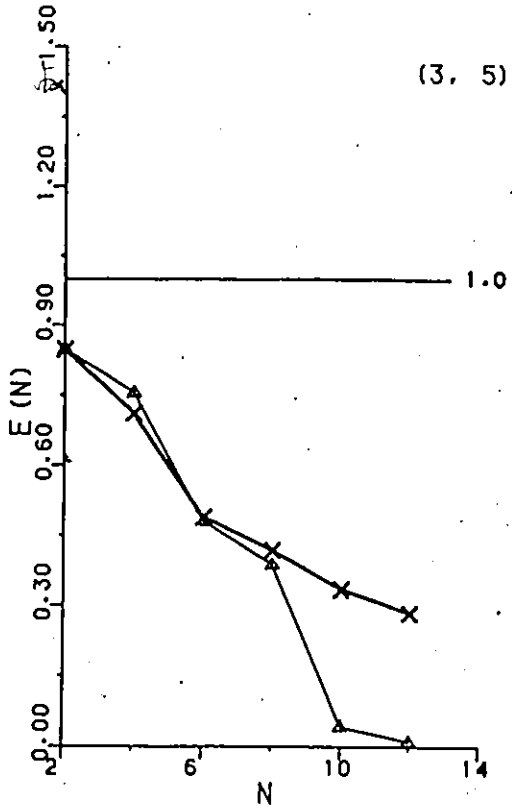

Figure 26

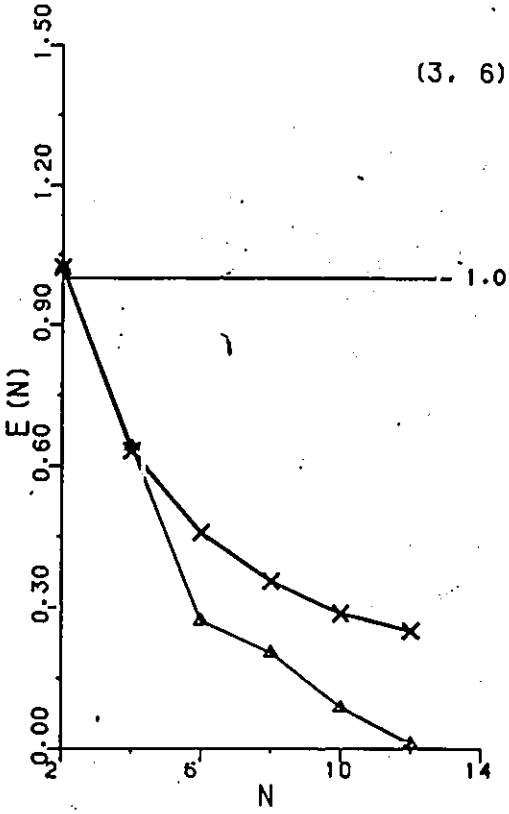

Figure 27

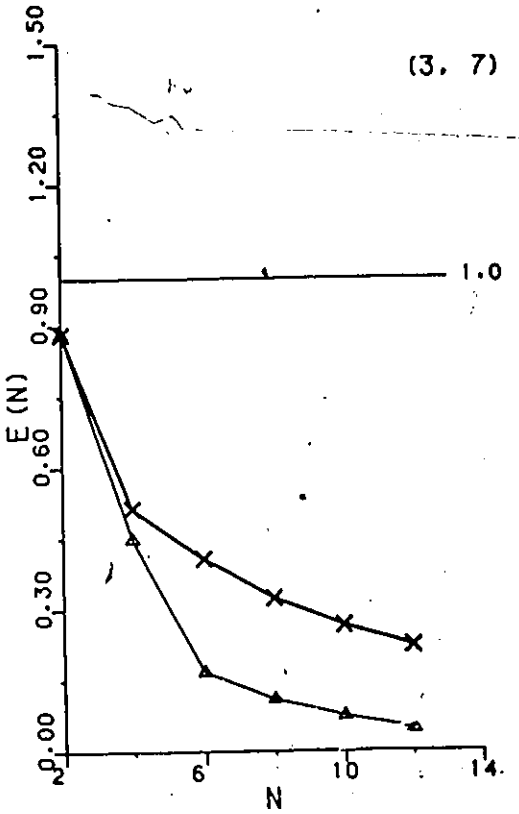

Figure 28

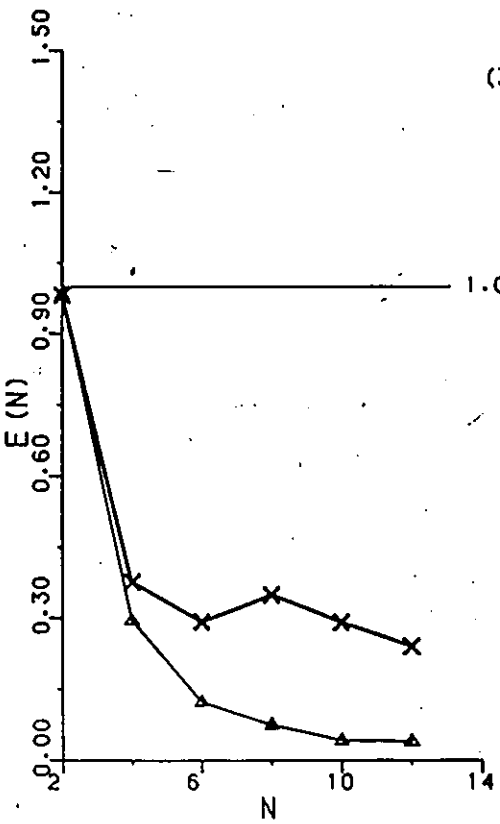

Figure 29

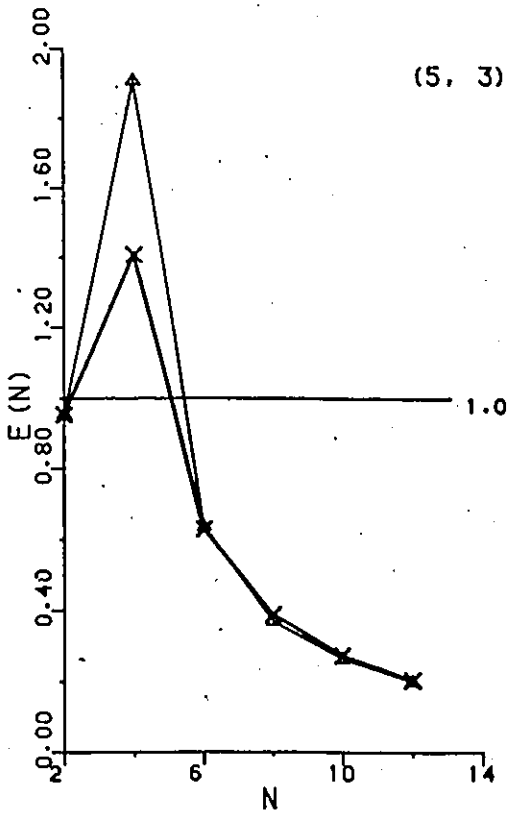

Figure 30

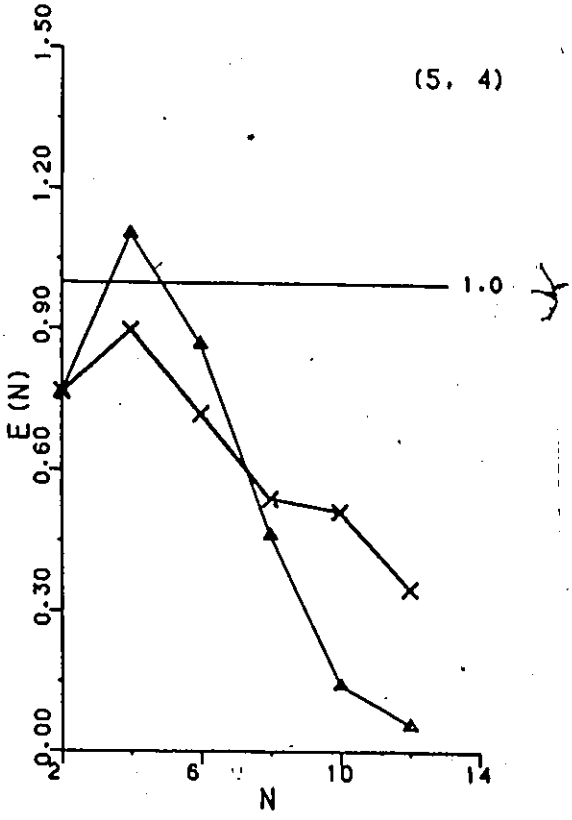

Figure 31

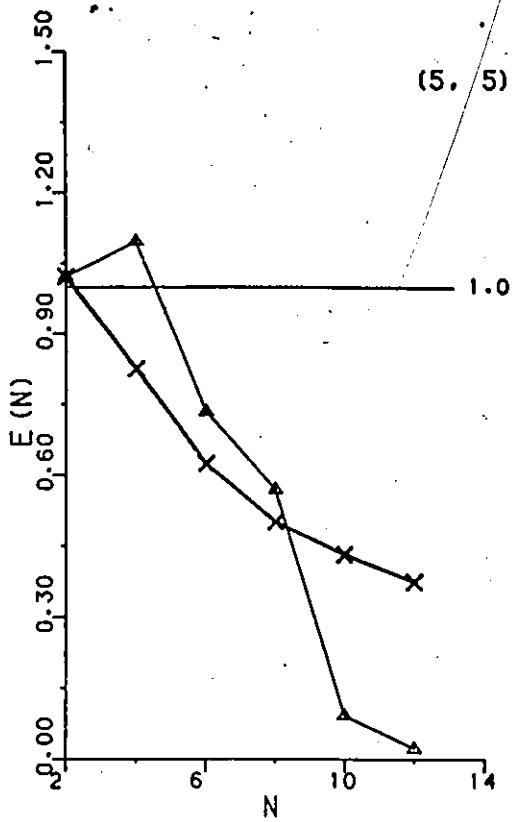

Figure 32

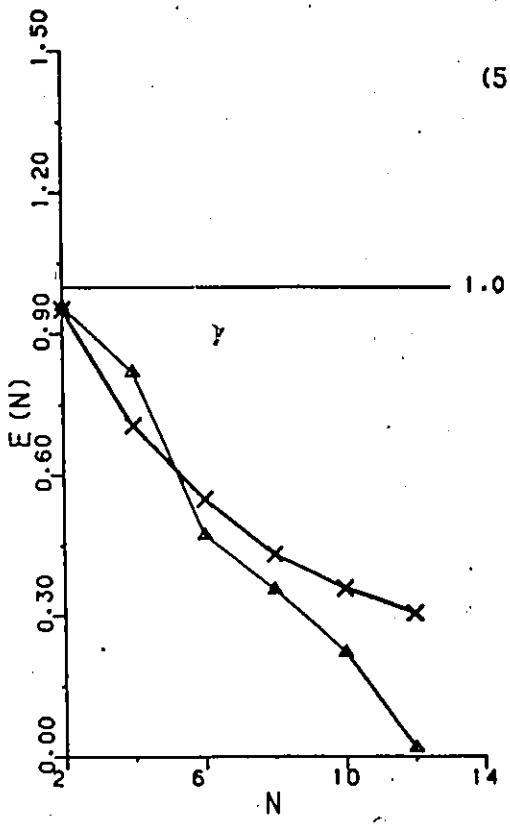

Figure 33

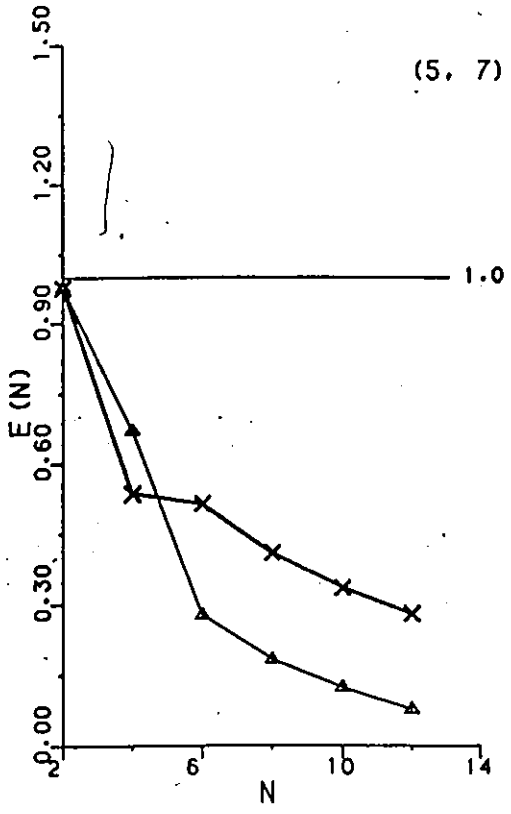

Figure 34

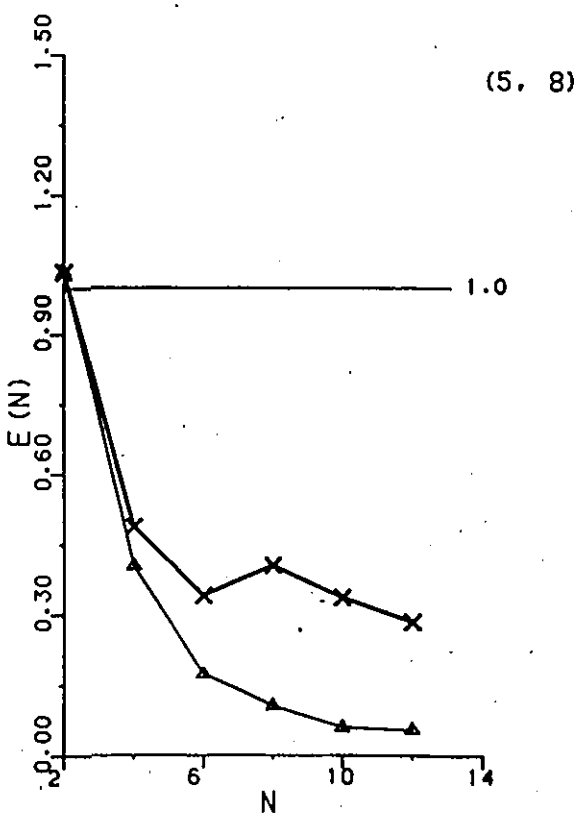

Figure 35

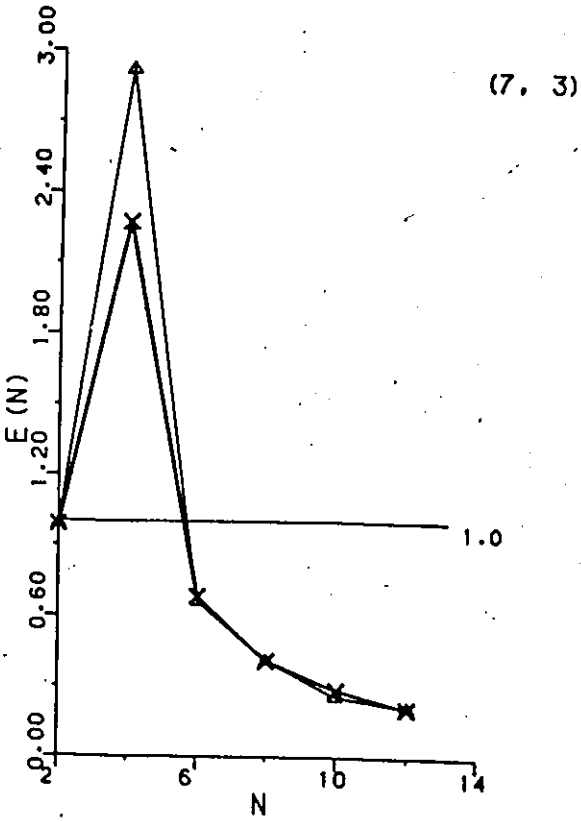

Figure 36

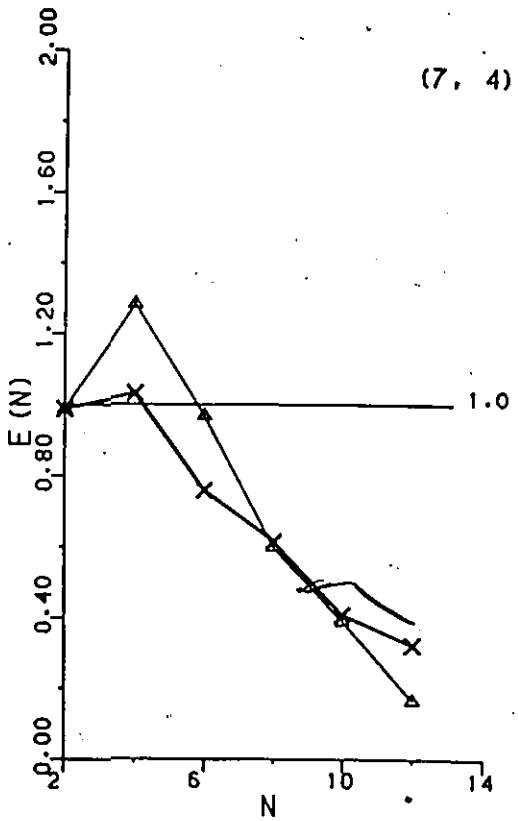

Figure 37

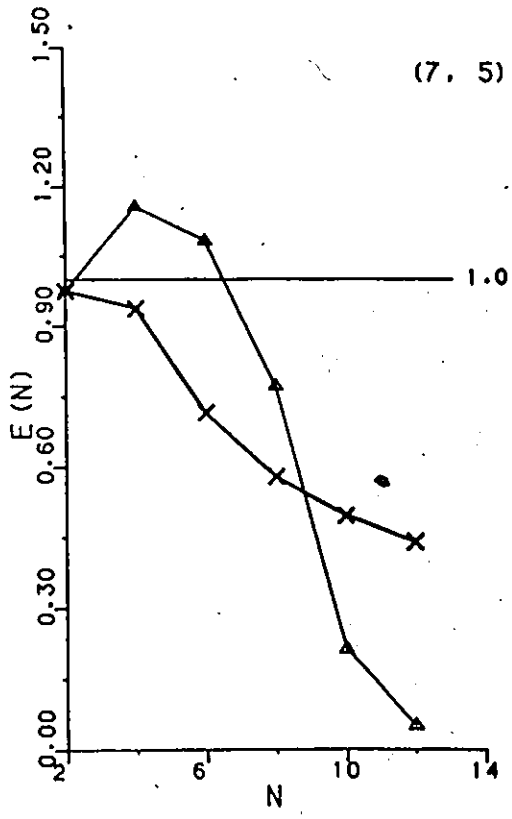

Figure 38

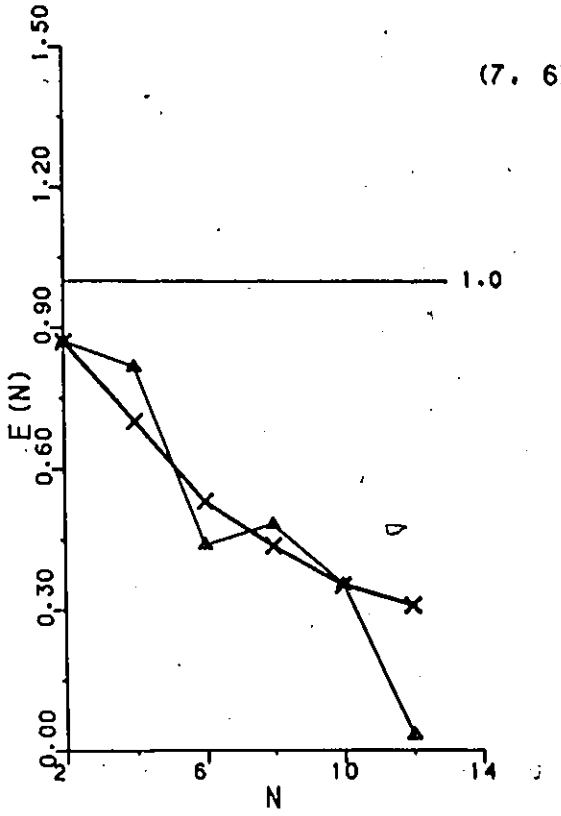

Figure 39

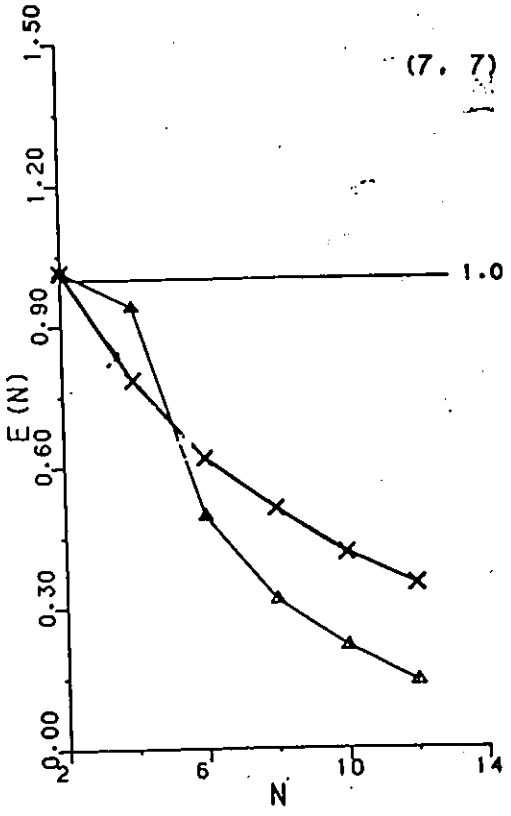

Figure 40

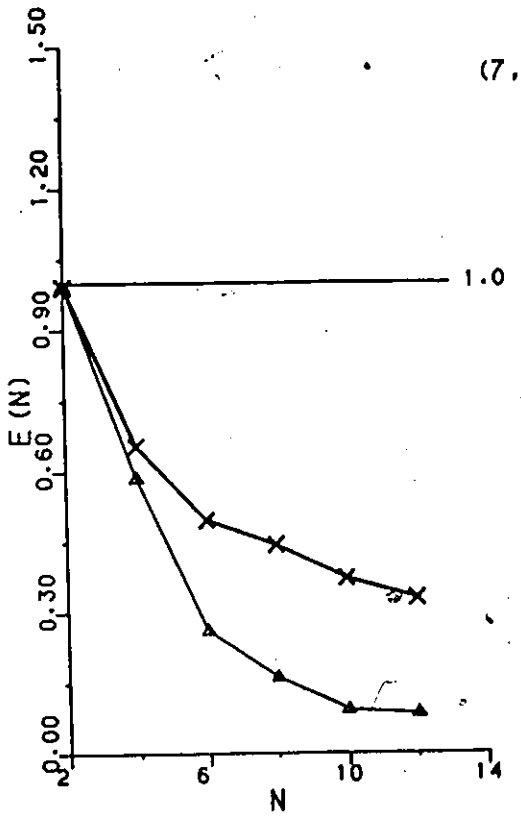

Figure 41

Chapter 5

Concluding Remarks

5.1 General Observations

The fixed stepsize test results as shown in Tables 1 and 2 and Figures 6 through 41 provide adequate evidence to conclude that the variation on the orginal PPC formulas as proposed in this thesis (i.e., Method B) provides superior performance than the orginal formulas (i.e., Method A).

A fundamental weakness in the PPC approach which has been clearly identified in this study, is the dramatic reduction in the absolute stability region which occurs with increasing the number of processors (N). This behavior appears to reflect into the performance results which show a significant reduction in the efficiency measure with increasing N. In particular, this suggests that the use of more than about 12 processors serves little purpose from the point of view of incremental gain. Thus it would seem that the PPC approach cannot be viewed as a practical alternative for implementation with a very large number of processors.

The results obtained with the variable stepsize implementation of Method B are somewhat disappointing inasmush as performance increases were not consistently obtained over all test problems. Furthermore, the incorporation of the variable stepsize has no apparent effect in moderating the decrease in effectiveness as the number of processors increases.

5.2 Future Work

The work completed in this thesis study is a contribution to the evolution of effective methods for the parallel solution of ode's. Much work remains to be done in this relatively new area and some possible projects are given below:

(a) More extensive testing of the various parallel approaches that have appeared in the literature using a broad range of test problems. The intent here would be to try and obtain a ranking of the effectiveness of these approaches and/or a characterization of the problem categories for which particularly good (or poor) behavior is likely.

(b) The testing activity in the present study was carried out in a single processor environment and relied on an idealized measure of solution speed; i.e., count of derivative function evaluations per processor. Testing using actual multiprocessor hardware should be undertaken to more correctly evaluate performance of the various solution methods.

(c) Investigation of alternate strategies for implementing variable stepsize procedures for the PPC approach should be undertaken to try and achieve a more consistent level of performance.

(d) The observed sensitivity in performance to the order of the integration formulas, suggests that an investigation into the development of a variable order parallel method could be worthwile.

(e) Stiff systems frequently occur in continuous system simulation studies and parallel methods specifically designed for such systems need to be developed.

Appendix 1

In this Appendix we give 2 sets of fixed stepsize PPC formulas corresponding to $s = 2, r = 4$ and $s = 4, r = 3$.

$\underline{s = 2, \quad r = 4}$

Predictor (Method A/F and B/F)

$$y^P_{2n+2} = y_{2n-2} + \tfrac{1}{3}h(28f^P_{2n} - 40f^P_{2n-1} + 32f_{2n-2} - 8f_{2n-3})$$
$$y^P_{2n+1} = y_{2n-2} + \tfrac{1}{8}h(21f^P_{2n} - 9f^P_{2n-1} + 15f_{2n-2} - 3f_{2n-3})$$

Corrector (Method A/F)

$$y_{2n} = y_{2n-3} + \tfrac{1}{8}h(3f^P_{2n} + 9f^P_{2n-1} + 9f_{2n-2} + 3f_{2n-3})$$
$$y_{2n-1} = y_{2n-3} + \tfrac{1}{3}h(f^P_{2n-1} + 4f_{2n-2} + f_{2n-3})$$

Corrector (Method B/F)

$$y_{2n} = y_{2n-2} + \tfrac{1}{3}h(f^P_{2n} + 4f^P_{2n-1} + f_{2n-2})$$
$$y_{2n-1} = y_{2n-2} + \tfrac{1}{24}h(9f^P_{2n-1} + 19f_{2n-2} - 5f_{2n-3} + f_{2n-4})$$

$\underline{s = 4, \quad r = 3}$

Predictor (Method A/F and B/F)

$$y^P_{4n+4} = y_{4n-4} + \tfrac{1}{3}h(88f^P_{4n} - 128f^P_{4n-1} + 64f^P_{4n-2})$$
$$y^P_{4n+3} = y_{4n-4} + \tfrac{1}{12}h(329f^P_{4n} - 532f^P_{4n-1} + 329f^P_{4n-2})$$
$$y^P_{4n+2} = y_{4n-4} + 3h(3f^P_{4n} - 4f^P_{4n-1} + 3f^P_{4n-2})$$
$$y^P_{4n+1} = y_{4n-4} - \tfrac{1}{12}h(8f^P_{4n} - 16f^P_{4n-1} + 85f^P_{4n-2})$$

Corrector (Method A/F):

$$y_{4n} = y_{4n-5} + \tfrac{1}{12}h(85f^P_{4n} - 200f^P_{4n-1} + 175f^P_{4n-2})$$
$$y_{4n-1} = y_{4n-6} - \tfrac{1}{12}h(85f^P_{4n-1} - 200f^P_{4n-2} + 175f^P_{4n-3})$$
$$y_{4n-2} = y_{4n-7} - \tfrac{1}{12}h(85f^P_{4n-2} - 200f^P_{4n-3} + 175f^P_{4n-4})$$
$$y_{4n-3} = y_{4n-7} - \tfrac{1}{3}h(8f^P_{4n-3} - 16f_{4n-4} + 20f_{4n-5})$$

Corrector (Method B/F)

$$y_{4n} = y_{4n-4} + \tfrac{1}{3}h(8f^P_{4n} - 16f^P_{4n-1} + 20f^P_{4n-2})$$
$$y_{4n-1} = y_{4n-4} - \tfrac{3}{4}h(f^P_{4n-1} + 3f^P_{4n-2})$$
$$y_{4n-2} = y_{4n-4} - \tfrac{1}{3}h(f^P_{4n-2} + 4f^P_{4n-3} + f_{4n-4})$$
$$y_{4n-3} = y_{4n-4} + \tfrac{1}{12}h(5f^P_{4n-3} + 8f_{4n-4} - f_{4n-5})$$

Appendix 2

In this Appendix we give a set of variable stepsize PPC formulas corresponding to $s = 4$, $r = 3$.

Recall in Fig.2 the stepsizes within blocks B_0, B_1 and B_2 are h_0, h_1 and h_2 respectively and let $\sigma_1 = h_1/h_0$ and $\sigma_2 = h_2/h_1$.

Predictor (Method B/V)

E1: $y^P_{4n+4} = y_{4n-4} + \beta_4 f4n^P + \beta_3 f^P_{4n-1} + \beta_2 f^P_{4n-2}$

E2: $y^P_{4n+3} = y_{4n-4} + \beta_4 f^P_{4n} + \beta_3 f^P_{4n-1} + \beta_2 f^P_{4n-2}$

E3: $y^P_{4n+2} = y_{4n-4} + \beta_4 f^P_{4n} + \beta_3 f^P_{4n-1} + \beta_2 f^P_{4n-2}$

E4: $y^P_{4n+1} = y_{4n-4} + \beta_4 f^P_{4n} + \beta_3 f^P_{4n-1} + \beta_2 f^P_{4n-2}$

where

$\beta_2 = \frac{1}{6}a^3 - \frac{7}{4}a^2 + 6a$

$\beta_3 = \frac{1}{3}a^3 + 3a^2 - 8a$

$\beta_4 = \frac{1}{\sigma_2}(\frac{1}{6}a^3 - \frac{5}{4}a^2 + 3)$

and

$a = 4 + 4\sigma_2$ for E1

$a = 4 + 3\sigma_2$ for E2

$a = 4 + 2\sigma_2$ for E3

$a = 4 + \sigma_2$ for E4

Corrector (Method B/V)

$y_{4n} = y_{4n-4} + \frac{8}{3\sigma_2} f^P_{4n} - \frac{16}{3} f^P_{4n-1} + \frac{40}{6} f^P_{4n-2}$

$y_{4n-1} = y_{4n-4} - \frac{3}{4}h(f^P_{4n-1} + 3f^P_{4n-2})$

$y_{4n-2} = y_{4n-4} - \frac{1}{3}h(f^P_{4n-2} + 4f^P_{4n-3} + f_{4n-4})$

$y_{4n-3} = y_{4n-4} + \beta_2 f^P_{4n-3} + \beta_1 f_{4n-4} - \beta_0 f_{4n-5}$

where

$\beta_0 = -\sigma_1 - \frac{1}{(a^2-a)}(\frac{1}{3}(1-a^3) - \frac{1}{2}(1-a^2)) - \frac{1}{a-1}(\frac{1}{2}(a(1-a^2) - \frac{1}{3}(1-a^3))$

$$\beta_1 = \frac{1}{\sigma_1(a-1)}\left(\frac{a}{2}(1-a^2) - \frac{1}{3}(1-a^3)\right)$$
$$\beta_2 = \frac{1}{\sigma_1(a^2-a)}\left(\frac{1}{3}(1-a^3) - \frac{1}{2}(1-a^2)\right)$$

and

$$a = \sigma_1 + 1$$

Appendix 3

In this Appendix, we give a list of Adams predictor-corrector formulas with order ranging from 3 to 8. We use "P" to indicate the predictor and "C" to indicate the corrector .

$\underline{r=3}$

P: $y_{n+1}^P = y_n + \frac{1}{12}h(23f_n - 16f_{n-1} + 5f_{n-2})$

C: $y_{n+1} = y_n + \frac{1}{12}h(5f_{n+1}^P + 8f_n - f_{n-1})$

$\underline{r=4}$

P: $y_{n+1}^P = y_n + \frac{1}{24}h(55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3})$

C: $y_{n+1} = y_n + \frac{1}{24}h(9f_{n+1}^P + 19f_n - 5f_{n-1} + f_{n-2})$

$\underline{r=5}$

P: $y_{n+1}^P = y_n + \frac{1}{720}h(1901f_n - 2774f_{n-1} + 2616f_{n-2} - 1274f_{n-3} + 251f_{n-4})$

C: $y_{n+1} = y_n + \frac{1}{720}h(251f_{n+1}^P + 646f_n - 264f_{n-1} + 106f_{n-2} - 19f_{n-3})$

$\underline{r=6}$

P: $y_{n+1}^P = y_n + \frac{1}{1440}h(4277f_n - 7673f_{n-1} + 9482f_{n-2} - 6798f_{n-3} + 2627f_{n-4} - 425f_{n-5})$

C: $y_{n+1} = y_n + \frac{1}{1440}h(475f_{n+1}^P + 1427f_n - 798f_{n-1} + 482f_{n-2} - 173f_{n-3} + 27f_{n-4})$

$\underline{r=7}$

P: $y_{n+1}^P = y_n + \frac{1}{60480}h(198721f_n - 447288f_{n-1} + 705549f_{n-2} - 688256f_{n-3} + 407139f_{n-4}$
$-134472f_{n-5} + 19087f_{n-6})$

C: $y_{n+1} = y_n + \frac{1}{60480}h(19087f_{n+1}^P + 65112f_n - 46461f_{n-1} + 37504f_{n-2} - 20211f_{n-3}$
$+6312f_{n-4} - 863f_{n-5})$

$\underline{r=8}$

P: $y^P_{n+1} = y_n + \frac{1}{120960}h(434241f_n - 1152169f_{n-1} + 2183877f_{n-2} - 2664477f_{n-3}$
$+2102243f_{n-4} - 1041723f_{n-5} + 295767f_{n-6} + 36799f_{n-7})$

C: $y_{n+1} = y_n + \frac{1}{120960}h(36799f^P_{n+1} + 139849f_n - 121797f_{n-1} + 123133f_{n-2} - 88547f_{n-3}$
$+41499f_{n-4} - 11351f_{n-5} + 1375f_{n-6})$

Appendix 4

In this Appendix we provide a summary of the five test problems used in the experimental evaluations carried out in this study.

$TP1$:

$$y' = y\cos t; \qquad y(0) = 1, \quad t_f = 20$$

Analytic solution:

$$y = e^{\sin t}$$

$TP2$:

$$y_1' = -y_2 - y_1 y_3/r; \; y_1(0) = 3$$
$$y_2' = y_1 - y_2 y_3/r; \quad y_2(0) = 0$$
$$y_3' = y_1/r; \qquad\quad y_3(0) = 0$$

$$r = (y_1^2 + y_2^2)^{1/2}; \qquad t_f = 20$$

Analytic solution:

$$y_1 = (2 + \cos t)\cos t$$
$$y_2 = (2 + \cos t)\sin t$$
$$y_3 = \sin t$$

$TP3$:

$$y_1' = y_2; \qquad y_1(0) = 1$$
$$y_2' = -y_1/r^3; \quad y_2(0) = 0$$
$$y_3' = y_4; \qquad y_3(0) = 0$$
$$y_4' = -y_3/r^3; \quad y_4(0) = 1$$

with $r = (y_1^2 + y_3^2)^{1/2}; \qquad t_f = 25$

Analytic solution

$$y_1 = \cos t$$
$$y_2 = -\sin t$$
$$y_3 = \sin t$$
$$y_4 = \cos t$$

$TP4$:

$$y_1' = \frac{y_1}{2(1+t)} - 2ty_2;\ y_1(0) = 1$$
$$y_2' = \frac{y_2}{2(1+t)} + 2ty_1;\ y_2(0) = 0$$

with $t_f = 6$

Analytic solution:

$$y_1 = \sqrt{(1+t)}\cos t^2$$
$$y_2 = \sqrt{(1+t)}\sin t^2$$

$TP5$:

$$\begin{aligned} y_1' &= y_2; & y_1(0) &= 0 \\ y_2' &= -2y_2 - 101y_1; & y_2(0) &= 1 \\ y_3' &= y_4; & y_3(0) &= 0 \\ y_4' &= y_1 - 4y_4 - 29y_3; & y_4(0) &= 0 \end{aligned}$$

with $t_f = 5$

Analytic solution

$$y_1 = 0.1e^{-t}\sin 10t$$
$$y_2 = \sqrt{1.01}e^{-t}\cos(10t + \theta_1)$$
$$y_3 = (5e^{-t}\sin(10t - \varphi) + 10e^{-2t}\sin(5t + \phi))/Q$$
$$y_4 = (5e^{-t}\sqrt{101}\cos(10t - (\varphi - \theta_1)) + 10e^{-2t}\sqrt{29}\cos(5t - (\Theta - \theta_2)))/Q$$

with

$$Q = 50\sqrt{6476}$$
$$\varphi = \arctan(-20/74)$$
$$\Theta = \arctan(-10/76)$$
$$\theta_1 = \arctan(0.1)$$
$$\theta_2 = \arctan(0.4)$$

Bibliography

[1] Abou-Rabia, O., "Extending the General Parallel Block Predictor- Corrector Formula to Variable Stepsize", Proc. *1988 Summer Computer Simulation Conference,* Seattle, Washington, pp. 21–23.

[2] Abou-Rabia, O. and L. G. Birta, "Some Variations on the BPC Parallel Integation Method", Proc. *1987 Summer Computer Simulation Conference,* Montreal, Quebec, pp. 37–42.

[3] Abou-Rabia, O. and L. G. Birta, "The Design of a Multimicrocomputer System for Continuous System Simulation", *Journal of Microcomputer Applications,* Vol. 10, No. 2, 1987, pp. 137–151.

[4] Birta, L. G. and O. Abou-Rabia, "Parallel Block Predictor-Corrector Methods for ODE's", *IEEE Trans. on Computers,* Vol. C-36, No. 3, 1987, pp. 299–311.

[5] Buehrer, R. E., H. J. Brundiers, H. Benz, B. Bron, H. Friess, W. Haelg, H. J. Halin, A. Isacson and M. Tadian, "The ETH-Multiprocessor EMPRESS: A Dynamically Configurable MIMD System", *IEEE Trans. on Computers,* Vol. C–31, No. 11, November, 1982, pp. 1035–1044.

[6] Chu, Moody T. and Hans Hamilton, "Parallel Solution of Ode's by Multi-Block Methods", *Siam J. Sci. Stat. Comput.,* Vol. 8, No. 3, May 1987.

[7] Cyre, W. R., C. J. Davis, A. A. Frank, L. Jedynak, M. J. Redmond and V. C. Rideout, "WISPAC: A Parallel Array Computer for Large Scale System Simulation", *Simulation,* Vol. 29, No. 4, November, 1977, pp. 165–172.

[8] Franklin, Mark A., "Parallel Solution of Ordinary Differential Equations", *IEEE Trans. Comput.,* Vol. C-27, No.5, May, 1978.

[9] Halin, H. J., R. Buhrer, W. Haleg, H. Benz, B. Bron, H. J. Brundiers, A. Isacson and M. Tadian, "The ETH Multiprocessor Project: Parallel Simulation of Continuous Systems", *Simulation,* Vol. 35, No. 4, October, 1980, pp. 109-123.

[10] Katz, Norman, Mark A. Franklin and A. Sen, "Optimally Stable Parallel Predictors for Adams-Moulton Correctors", *Comp. & Math. with Appls.,* Vol. 3, pp. 217-233.

[11] Krosel, Susan M. and Edward J. Milner, "Applications of Integration Algorithms in a Parallel Processing Environment for the Simulation of Jet Engines", 15^{th} *Annual Simulation Symposium,* Tampa, Florida, 1982.

[12] Lapidus, L. and J. H. Seinfield, "Numerical Solutions of Ordinary Differential Equations", Academic Press, New York, 1971

[13] Lambert, J. D., "Computational Methods in Ordinary Differential Equations", John Wiley & Sons, New York.

[14] Miranker, W. L. and W. M. Liniger, "Parallel Methods for Numerical Integration of Ordinary Differential Equations", *Math. Comput.,* Vol. 23, 1969, pp. 731-740.

[15] O'Grady, E. P. and C. H. Wang, "Performance Limitations in Parallel Processor Simulations", *Transactions of SCS,* vol.4, Oct., 1987, pp. 311–330.

[16] Shampine, L.F. and H.A. Watts, "Block Implicit One-step Methods", *Math. Comput.,* Vol. 12, 1972, pp. 252–266.

[17] Watts, H. A. and L. F. Shampine, "A Stable Block Implicit One-Step Methods", *BIT,* Vol.12, 1972, pp. 252–266.

[18] Worland, P. B., "Parallel Methods for the Numerical Solution of Ordinary Differential Equations", *IEEE Trans. on Computers,* Vol. C-25, 1976, pp. 1045–1048.

[19] Yoshikowa:et al., "A Multi-Microprocessor Approach to High-Speed and Low-Cost Continuous System Simulation", Proc. *National Computer Conference, AIFIP Press,* 1977, pp. 931–936.