Parallel Block Predictor-Corrector Methods for the Numerical Solution of ODE's

By

Liming Yang

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

Ottawa, Ontario, 1993

ISBN 0-315-89690-6

Canada

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

To L. Ding

for all love she gives me

I hereby declare that I am the sole author of this thesis.

I authorize the University of Ottawa to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the University of Ottawa to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Liming Yang

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 help.

Abstract

Stability and efficiency (i.e. derivative function evaluations per processor) are the two main considerations in deriving good numerical methods for ODE's. The underlying challenge is to increase the stability region while maintaining or even improving efficiency. To achieve this, some extensions of predictor-corrector based methods, which apply a fixed number of corrector iterations, are considered.

This thesis studies two particular members of a family of methods called the Parallel Block Predictor-Corrector Family, which are based on these extensions. These two members are called PBPC/2 and PBPC/3. They are characterized by iterated corrector evaluations carried out in two adjacent blocks. Stability properties of these methods are analyzed and compared with some existing block-based parallel predictor-corrector methods. Performance of the PBPC/2 and PBPC/3 methods and these existing block-based parallel predictor-corrector methods is compared using solution formulas which extend over a range of integration orders and which use various number of processors. The results obtained from a stability analysis and from a collection of numerical experiments indicate that the proposed methods provide a potential opportunity to balance stability properties and efficiency in the parallel computer systems.

Contents

1. Introduction

1.1 Review of Previous Work

We are interested in investigating parallel numerical methods for solving the system of ordinary differential equations (ode's)

$$y(t_0) = y_0$$
$$\frac{dy}{dt} = f(t, y(t)), \quad t_0 \le t \le t_f \qquad (1.1)$$
$$y, f \in R^n$$

Such systems of equations occur frequently in a wide variety of fields. The development of efficient means for their solution has grown in importance with the increasing use of simulation as a problem solving tool. Furthermore the increasing use of multiprocessor systems has established a pressing need for effective methods that properly exploit the computational power of such systems.

Approaches for achieving parallelism in the ode problem fall into two broad categories; namely, the equation segmentation approach (sometimes referred to as parallelism across the system) and the parallel algorithm approach (sometimes referred to as parallelism across the method). In the former case, the given system of ode's is separated into constituent parts and these are allocated to the available processors where conventional ode solution procedures are applied (e.g. Runge-Kutta, or linear multistep). If the number of equations is large, this approach provides a excellent opportunity of parallelism, as long as the computation load can be well balanced across processors. Existing methods in this category are more or less based on the idea of independent integration [1,2]. With independent integration, subsets of equations are integrated independently of one another, with the stepsize and error control of each subset determined locally. The fundamental shortcoming of this kind of approach is the inherent problem of how to carry out the equation separation in a way which equalizes the workload among the available processors. Therefore, the approach is likely to be efficient only when the system has a regular structure such as that which arises from the decomposition of a partial differential equation. In addition, however, significant inter-processor communication problems may be arise.

In the parallel algorithm approach, each processor deals with all equations and parallelism is achieved at the algorithmic level. This approach provides the potential for only a small degree of parallelism, but an outstanding advantage of this approach is its total circumvention of the decomposition problem. Although inter-processor communication problems do arise, these are relatively easy to deal with. In this thesis, our investigation concentrates on this second approach.

The most often proposed methods in the parallel algorithm approach are block-based parallel predictor-corrector methods. With methods of this category, the solution axis; i.e., the t-axis, is divided into a series of adjacent blocks each containing a fixed number of equally spaced points (i.e., s points). Each "step" of the procedure generates a solution value at each of the s points within a block. To achieve this, one of the N available processors is allocated to each point in a block in order to (simultaneously) carry out the necessary computation. Thus all s points in a block can be regarded as being simultaneously "active". The nature of the computation corresponds either to a predictor or to a corrector computation.

Each step of the procedure outlined above generally take place over a sequence of computation periods each of which consumes a fixed amount of (real) time. Each such period is identified with an evaluation, by each processor, of the derivative function, f, in (1.1). The number of such computation periods that occurs during a step is referred to as the number of stages of the method.

The block-based parallel predictor-corrector methods can be formulated from any appropriate class of discrete methods such as Linear multistep, Runge-Kutta, or Multivalue methods and can be used in the solution of non-stiff or stiff equations depending on the nature of the iteration. Since our study in this thesis focuses on the investigation of non-stiff methods, our review of previous research in this section will concentrate on such methods.

We classify the block-based parallel predictor-corrector methods into two families called the Block Predictor-Corrector (BPC) family and the Parallel Block Predictor-Corrector (PBPC) family, respectively. The characteristic feature of the BPC family is that all available processors (N) are assigned to a single block; hence s=N. This, in effect, implies that all processors carry out a generic sequence of computations in a synchronous manner and there is no workload balancing problem to deal with.

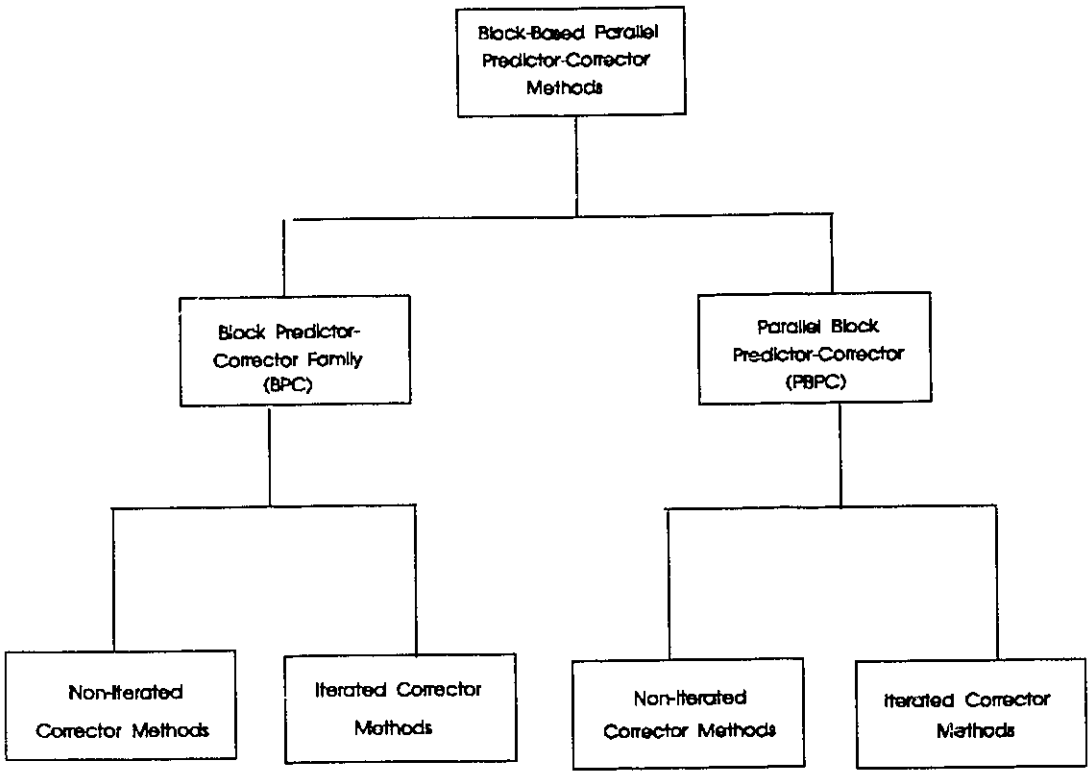

Fig. 1.1

The Classification of Block-Based
Parallel Predictor-Corrector Methods

The distinctive feature of the PBPC family is that the N available processors are distributed over multiple adjacent blocks; hence N=us for some integer value u>1. This, in particular, implies that multiple blocks are "active" during any computation period. Though a common computation takes place for all points within a given block, it is not essential that this computation be identical over all active blocks. However, from the point of view of ensuring that all processors are always "productive" it is important to avoid dissimilar computations within the active blocks.

For each family, we consider two groups of methods; namely, non-iterated corrector methods;

i.e., those methods for which the corrector is only applied once and iterated corrector methods which are those methods where the corrector is applied more than once. Fig. 1.1 shows this classification.

Moulton [3,4] was probably the first investigator to discuss block methods. Milne [5] also proposed a method that generates four new values in a single step to serve as the starter for a multistep method. Rosser [6] extended this idea with the intent of making Runge-Kutta schemes more efficient. He gives methods for block sizes up to eight. A typical 4-point block formula, as produced by Rosser, is given by (note that in the interests of notational simplicity and with no loss of generality, we consider only a single equation throughout our presentation):

$$\begin{bmatrix} y^0_{n+\frac{7}{4}} \\ y^0_{n+\frac{6}{4}} \\ y^0_{n+\frac{5}{4}} \\ y^0_{n+1} \end{bmatrix} = \begin{bmatrix} y_{n+\frac{3}{4}} \\ y_{n+\frac{3}{4}} \\ y_{n+\frac{3}{4}} \\ y_{n+\frac{3}{4}} \end{bmatrix} + H \begin{bmatrix} 1 \\ \frac{3}{4} \\ \frac{2}{4} \\ \frac{1}{4} \end{bmatrix} f(y_{n+\frac{3}{4}}) \qquad (1.2.a)$$

$$\begin{bmatrix} y^{m+1}_{n+\frac{7}{4}} \\ y^{m+1}_{n+\frac{6}{4}} \\ y^{m+1}_{n+\frac{5}{4}} \\ y^{m+1}_{n+1} \end{bmatrix} = \begin{bmatrix} y_{n+\frac{3}{4}} \\ y_{n+\frac{3}{4}} \\ y_{n+\frac{3}{4}} \\ y_{n+\frac{3}{4}} \end{bmatrix} + \frac{H}{4} \begin{bmatrix} \frac{28}{90} \\ \frac{27}{80} \\ \frac{29}{90} \\ \frac{215}{720} \end{bmatrix} f_{n+\frac{3}{4}} + \frac{H}{4} \begin{bmatrix} \frac{128}{90} & \frac{48}{90} & \frac{128}{90} & \frac{28}{90} \\ \frac{102}{80} & \frac{72}{80} & \frac{42}{80} & -\frac{3}{80} \\ \frac{124}{90} & \frac{24}{90} & \frac{4}{90} & -\frac{1}{90} \\ \frac{646}{720} & -\frac{264}{720} & \frac{106}{720} & -\frac{19}{720} \end{bmatrix} \begin{bmatrix} f^m_{n+\frac{7}{4}} \\ f^m_{n+\frac{6}{4}} \\ f^m_{n+\frac{5}{4}} \\ f^m_{n+1} \end{bmatrix} \qquad (1.2.b)$$

where m is an iteration indictor (m=0,1,2,..). Rosser considered the solution values y_{n+1}, $y_{n+5/4}$, $y_{n+6/4}$, $y_{n+7/4}$ as the internal stage values of a Runge-Kutta method with stepsize H. However, it should be noted that each of these values can be regarded as a valid solution point. Tam [12] has stated that since the above stage values can be computed at the same time by a group of four processors the number of function evaluations per step is lower than that of a Runge-Kutta method with the comparable order. Note also that Rosser's approach in (1.2) can be viewed as

a member of the BPC family as shown in Fig. 1.1. It has a first order predictor and multiple corrector iterations.

Shampine and Watts [7,8] and Worland [9] studied the BPC family and suggested new formulas where the predictor and the corrector have the same order. An example of a 4th order formula is given by:

$$\begin{bmatrix} y^p_{n+2} \\ y^p_{n+1} \end{bmatrix} = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix} \begin{bmatrix} y_n \\ y_{n-1} \\ y_{n-2} \end{bmatrix} + H \begin{bmatrix} \frac{79}{6} & -\frac{72}{6} & \frac{29}{6} \\ \frac{13}{3} & -\frac{4}{3} & 1 \end{bmatrix} \begin{bmatrix} f_n \\ f_{n-1} \\ f_{n-2} \end{bmatrix} \qquad (1.3.a)$$

$$\begin{bmatrix} y_{n+2} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} y_n \\ y_n \end{bmatrix} + H \begin{bmatrix} \frac{1}{6} & \frac{4}{6} & \frac{1}{6} \\ -\frac{1}{6} & \frac{8}{6} & \frac{5}{6} \end{bmatrix} \begin{bmatrix} f^p_{n+2} \\ f^p_{n+1} \\ f_n \end{bmatrix} \qquad (1.3.b)$$

The method proposed by Shampine and Watts can be written in the following general form

$$Y_1^p = A^p Y_0 + H B^p F(Y_0) \qquad (1.4.a)$$

$$Y_1 = A^c Y_0 + H B_1^c F(Y_1^p) + H B_2^c F(Y_0) \qquad (1.4.b)$$

where s is the number of points in any block, $Y_1=(y_{n+s}, y_{n+s-1}, \dots, y_{n+1})^T$, y_{n+j}, j=1,2,...,s, denotes the generated solution value at t_{n+j}; $Y_0=(y_n, y_{n-1}, y_{n-2}, \dots, y_{n-s})^T$, y_{n-j}, j=0,1,2,..,s, denotes the generated solution at $t=t_{n-j}$; A^p,B^p and B_2^c are s x (s+1) matrices, A^c is an s x (s+1) matrix whose entries in the first column are set to 1 and all others are 0. B_1^c is an s x s matrix.

The method of Shampine and Watts as presented in [7,8] is called the "equal weight predictor" BPC method by Birta and Abou-Rabia [10] since all entries in A^p are equal (the common value is 1/(s+1)). They proposed an improvement to the method by modifying the predictor formula in a "Null Weight" fashion and referred to it as the NWP/BPC method. More specifically, the entries in the first column of the matrix A^p in (1.4) are set to 1, and all others are 0 (i.e., $A^p=A^c$).

This modification enlarges the stability interval of the method, where the stability interval is the interval on the real axis extending from the origin to the point of intersection of the real axis with the stability region.

Chu and Hamilton [11] also introduced some BPC methods. An example of one of their methods is :

$$\begin{bmatrix} y^P_{n+2} \\ y^P_{n+1} \end{bmatrix} = \begin{bmatrix} -\alpha_6 & -\alpha_5 \\ -\alpha_3 & -\alpha_2 \end{bmatrix} \begin{bmatrix} y_n \\ y_{n-1} \end{bmatrix} + \begin{bmatrix} -\alpha_4 & 1-\alpha_4+\alpha_5+\alpha_6 \\ -\alpha_1 & 1+\alpha_1+\alpha_2+\alpha_3 \end{bmatrix} \begin{bmatrix} y_{n-2} \\ y_{n-3} \end{bmatrix}$$
$$+\frac{H}{96}\begin{bmatrix} 225+\alpha_4+9\alpha_6 & -325-5\alpha_4+8\alpha_5+27\alpha_6 \\ 64+\alpha_1+9\alpha_3 & -32-5\alpha_1+8\alpha_2+27\alpha_3 \end{bmatrix} \begin{bmatrix} f_n \\ f_{n-1} \end{bmatrix}$$
$$+\frac{H}{96}\begin{bmatrix} 275+19\alpha_4+32\alpha_5+27\alpha_6 & -55+9\alpha_4+8\alpha_5+9\alpha_6 \\ 64+19\alpha_1+32\alpha_2+27\alpha_3 & 9\alpha_1+8\alpha_2+9\alpha_3 \end{bmatrix} \begin{bmatrix} f_{n-1} \\ f_{n-3} \end{bmatrix}$$

$$\begin{bmatrix} y_{n+2} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} -\alpha^*_2 & 1+\alpha^*_2 \\ -\alpha^*_1 & 1+\alpha^*_1 \end{bmatrix} \begin{bmatrix} y_n \\ y_{n-1} \end{bmatrix} + \frac{H}{96}\begin{bmatrix} 9+\alpha^*_2 & 27-5\alpha^*_2 \\ \alpha^*_1 & 8-5\alpha^*_1 \end{bmatrix} \begin{bmatrix} f^P_{n+2} \\ f^P_{n+1} \end{bmatrix} + \frac{H}{96}\begin{bmatrix} 27+19\alpha^*_2 & 9+9\alpha^*_2 \\ 32+19\alpha^*_1 & 8+9\alpha^*_1 \end{bmatrix} \begin{bmatrix} f_n \\ f_{n-1} \end{bmatrix}$$

where $\alpha_1 = -1.23$, $\alpha_2 = -0.463$, $\alpha_3 = -0.0277$, $\alpha_4 = -1.33$, $\alpha_5 = -0.77212$, $\alpha_6 = -0.37$, $\alpha^*_1 = -1.01$, and $\alpha^*_2 = -0.0452$. This is a fourth order method.

To improve stability properties, Tam [12] constructed a BPC class of methods that have a predictor order of s and a corrector order of s+1. The order of method is s+1. The stability region of this class of methods does not decrease as s increases. For example, the case for s=2 is:

$$\begin{bmatrix} y^P_{n+2} \\ y^P_{n+1} \end{bmatrix} = \begin{bmatrix} 1-b & b \\ 1-a & a \end{bmatrix} \begin{bmatrix} y_n \\ y_{n-1} \end{bmatrix} + H \begin{bmatrix} 2+\frac{b}{4} & -1+\frac{b}{4} \\ \frac{3}{4}+\frac{a}{4} & -\frac{1}{4}+\frac{a}{4} \end{bmatrix} \begin{bmatrix} f_n \\ f_{n-1} \end{bmatrix}$$

$$\begin{bmatrix} y_{n+2} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} 1-b^* & b^* \\ 1-a^* & a^* \end{bmatrix} \begin{bmatrix} y_n \\ y_{n-1} \end{bmatrix} + H \begin{bmatrix} \frac{7}{18} - \frac{b^*}{72} & 0 \\ 0 & \frac{5}{24} - \frac{a^*}{24} \end{bmatrix} \begin{bmatrix} f^P_{n+2} \\ f^P_{n+1} \end{bmatrix}$$
$$+ H \begin{bmatrix} \frac{5}{6} + \frac{7}{24} b^* & -\frac{2}{9} + \frac{2}{9} b^* \\ \frac{1}{3} + \frac{1}{3} a^* & -\frac{1}{24} + \frac{5}{24} a^* \end{bmatrix} \begin{bmatrix} f_n \\ f_{n-1} \end{bmatrix}$$

where a, b, a^*, b^* are free parameters. The specification of these free parameters is subject to the following "stability condition":

$$\psi(\xi) = (\xi - (\frac{(\lambda H)^2}{2} + (\lambda H) + 1))^2 \qquad (1.6)$$

The specification in (1.6) is the stability polynomial of this method. Its form implies that the stability region of the method is defined by $|(\lambda H)^2/2 + (\lambda H) + 1| \leq 1$ and hence is a unit circle centered at (-1,0).

The general form of (1.6) is

$$\psi(\xi) = (\xi - (\frac{(\lambda H)^2}{2} + (\lambda H) + 1))^s \qquad (1.7)$$

From this it is clear that the stability polynomial is a perfect power and the stability region does not shrink as s (and hence, order), increases.

Unfortunately, because of very large error coefficients, this class of methods performs poorly when the order of the method is greater than four.

Another popular class of BPC methods uses the Euler formula as the predictor and applies a

number of corrector iterations using a relatively high order Runge-Kutta corrector formula. This approach is usually called the Parallel Iterated Runge-Kutta (PIRK) method. Much work with methods of this class has been reported by Van Der Houwen and Sommeijer [13,14,15]. The Runge-Kutta corrector for the m^{th} iteration has the following form:

$$Y_1^m = AY_0 + HB_1F(Y_1^{m-1}) + HB_2F(Y_0) \qquad (1.8)$$

Here A, B_1, B_2 are s x s matrices, Y_1 is as defined earlier (see following (1.4)), and $Y_0=(y_n,y_{n-1},\ldots,y_{n-s+1})^T$. (Note that the size of Y_0 here is different from that in (1.4)). Note also that the scope of the methods encompassed by (1.8) can be significantly extended by taking the size of Y_0 (the number of previous solution values that are used) to be k where k is a parameter.

The PBPC approach was originally proposed by Miranker and Liniger [21] (see Fig. 1.1). In that presentation, second third and fourth order parallel predictor-corrector methods were given for the two processor case. An example of a fourth order formula is:

$$y_{n+1}^P = y_{n-1} + \frac{H}{3}(8f_n^P - 5f_{n-1} + 4f_{n-2} - f_{n-3})$$

$$y_n = y_{n-1} + \frac{H}{24}(9f_n^P + 19f_{n-1} - 5f_{n-2} + f_{n-3})$$

The essential feature of the approach is that the available processors are separated into two equal groups, one of which carries out predictor computations while the other carries out corrector computations. These computations take place in adjacent blocks. The predictor and corrector computations move forward along the solution axis in unison and hence the approach is sometimes referred to as a "frontal" method.

Further work with this approach was carried out by Krosel and Milner [22] who presented second order formulas for two, four and eight processor cases. Katz, Franklin and Sen [23] examined this approach in the context of correctors which have a higher order than predictors. Abou-Rabia, Birta and Chen [24] extended the method of Miranker and Liniger to larger values

of s (number of points in any block) and also considered higher values of integration order, r. The formulas investigated in [24] have the form:

$$y^{p}_{n+i}=y_{n-s}+H\sum_{j=1}^{r}\beta^{p}_{ij}f(y_{n-(r-j)}); \quad for\ i=1,2,..,s$$

$$y^{c}_{n+i-(s-1)}=y_{n-s}+H\sum_{j=1}^{r}\beta^{c}_{ij}f(y_{n+i-(s-1)-(r-j)}) \quad for\ i=0,1,..,(s-1)$$

These formulas can be rewritten in the following more general form:

$$Y_1^{p}=A^{p}Y_{-1}+HB_1^{p}F(Y_0^{p})+HB_2^{p}F(Y_{-1}) \qquad (1.9.a)$$

$$Y_0^{c}=A^{c}Y_{-1}+HB_1^{c}F(Y_0^{p})+HB_2^{c}F(Y_{-1}) \qquad (1.9.b)$$

where $Y_1^{p}=(y_{n+s}^{p},y_{n+s-1}^{p},....,y_{n+1}^{p})^{T}$, Y_0 is an s-vector with $Y_0=(y_n,y_{n-1},...,y_{n-s+1})^{T}$, Y_{-1} is a k-vector with $Y_{-1}=(y_{n-s}, y_{n-s-1},...,y_{n-s-k+1})^{T}$. A^{p} and A^{c} are matrices of size s x k; B^{p}_1 and B^{c}_1 are matrices of size s x s; B^{p}_2 and B^{c}_2 are matrices of size s x k.

The relationships in (1.9) can be written as:

$$\begin{bmatrix}Y_1^{p}\\ Y_0^{c}\end{bmatrix}=\begin{bmatrix}0 & A^{p}\\ 0 & A^{c}\end{bmatrix}\cdot\begin{bmatrix}Y_0^{p}\\ Y_{-1}\end{bmatrix}+H\cdot\begin{bmatrix}B_1^{p} & B_2^{p}\\ B_1^{c} & B_2^{c}\end{bmatrix}\cdot\begin{bmatrix}F(Y_0^{p})\\ F(Y_{-1})\end{bmatrix}$$

Let

$$\overline{Y_1}=(Y_1^P, Y_0^C)^T,\ \overline{Y_0}=(Y_0^P, Y_{-1})^T$$

then we have:

$$\overline{Y_1}=A\overline{Y_0}+HBF(\overline{Y_0}) \qquad (1.10)$$

It should be noted that if N=2s processors are available, there is only one function evaluation per processor per step in the solution procedure represented by (1.10). This feature enhances the efficiency of the approach. However the requirement of computing the predicted solution values in block (n+1) and the corrected solution values in block n at the same time indicates that the predictor values in block (n+1) must be computed from predicted values in block n (rather than from corrected values). This delay of information usage restricts the stability region of the method. Therefore, the method of (1.10) has a relatively small stability region in comparison with members of the BPC family [30,31]. A comparison of the stability behaviour of some members of the BPC family with some special members of (1.10) can be found in [24].

All of the methods mentioned above can be represented in a more general model called Butcher's explicit linear model [12,17,18]. This model encompasses many sequential and parallel ode methods and consequently has broad applicability. Some analysis of the features of this model with respect to stability properties and local error can be found in [19,20,29].

As Burrage [25] has pointed out, the standard parallel approaches, as outlined above, have relatively small stability regions unless a large number of corrector iterations are applied. But a large number of corrector iterations implies an increase in the number of function evaluations which, in turn, means a loss in efficiency. Thus, the underlying challenge in generating an efficient parallel methods is how to achieve efficiency without a sacrifice in stability properties.

1.2 The Main Contributions of the Thesis

In this thesis, we first examine a particular member of the BPC family. This member is the one referred to as the Null-Weight-Predictor (NWP) in [10]. Its stability properties are

investigated when the number of corrector iteration are increased beyond one; i.e. for a $P(EC)^mE$ format. This investigation shows that the stability behaviour, for m=1 and the integration order r sufficiently high (e.g. r≥6) is superior to the Adams Bashforth/Adams Moulton (ABAM) method. (Note that the ABAM method is the special case of the NWP/BPC method which corresponds to s=1; i.e., the single processor case). Furthermore, it is shown that the stability bound will increase as m increases up to a limiting value which is dependent on r. This feature provides an opportunity to improve the stability properties of those parallel methods whose formulation is based on the linear multi-step methods by applying a fixed number (>1) of corrector iterations.

To deal with the problem of inefficiency which results from an increases of the number of corrector iterations, we also investigate the Parallel Block Predictor-Corrector (PBPC) family. In particular, we extend the Parallel Predictor-Corrector (PPC) method described in [24] by incorporating more than one corrector iteration. This, in particular, implies a situation where a corrector iteration simultaneously take place in two adjacent blocks. The methods of this category which we investigate are designated by the notation PBPC/M where M represents the number of function evaluations per processor per step. A simple relationship between M and the number of corrector iteration is formulated. We consider only the cases where M=2 or 3. Also we restrict our attention to the case where only two blocks are simultaneously active.

An analysis of the stability properties of these PBPC methods is carried out and the stability properties are compared with those of NWP/BPC methods as described in [10,24]. The performance of these methods 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 these experiments, it is shown that the PBPC methods provide better results in most cases than the NWP/BPC methods in [10,24], especially when r≥s+1.

It should also be noted that, in this thesis, H denotes the distance between the left-most point and right-most point in any block; i.e., H=sh, where h is the distance between adjacent points in any block. The stability bound used in this thesis is expressed in terms of H as opposed to h. As a consequence of this the stability bound values differ from those presented in [10] and [24] where h is used as the underlying basis for expressing stability bound information.

2. Stability Properties of Some Members of the BPC Family

2.1 Introduction

Stability, efficiency and accuracy are the main considerations in deriving good numerical methods for the solution of ode's. In this chapter, we focus our considerations on the issue of stability for some members of the BPC family which incorporate corrector iterations. These methods are generally applicable only to the solution of non-stiff problems.

First, the stability behaviour of a particular member of the BPC family is analyzed. This is the NWP/BPC method as proposed in [10,24]. Since the formulation of the NWP/BPC method is based on the Adams model, the stability behaviour of the NWP/BPC is compared with that of the Adams-Bashforth/Adams-Moulton (AB/AM) method. The analysis indicates that the NWP/BPC has a bigger stability bound when the integration order is greater than or equal to six. This feature implies that relatively higher order formulas can be used with the NWP/BPC formulation. In order to investigate the possibility of improving the stability properties of Block-Based Parallel Predictor-Corrector methods, an investigation of the effect of using more than one corrector iteration with the NWP/BPC formulas is undertaken. The results of this investigation show the opportunity for increasing the stability bound by applying a fixed number of corrector iterations.

2.2 Structure of the BPC Family

The basic structure of the inherently explicit methods of the BPC family is represented in Fig. 2.1. The integer M in this figure indicates the number of derivative function evaluations per processor per step.

The methods of the BPC family can be written as:

$$Y_1^0 = A_0 Y_0 + H B_0 F(Y_0) \qquad (2.1.a)$$

$$Y_1^m = A_1 Y_0 + B_1 F(Y_1^{m-1}) + H B_2 F(Y_0); \quad m = 1,2,..,M-1 \qquad (2.1.b)$$

where $Y_1=(y_{n+s},y_{n+s-1},\ldots,y_{n+1})^T$ is an s-vector and $Y_0=(y_n,y_{n-1},\ldots,y_{n-k+1})^T$ is a k-vector, A_0, B_0, B_2 are matrices of size s x k, and B_1 of size s x s. Here k denotes the number of saved values to be used. Corrector iterations occur when M > 2.

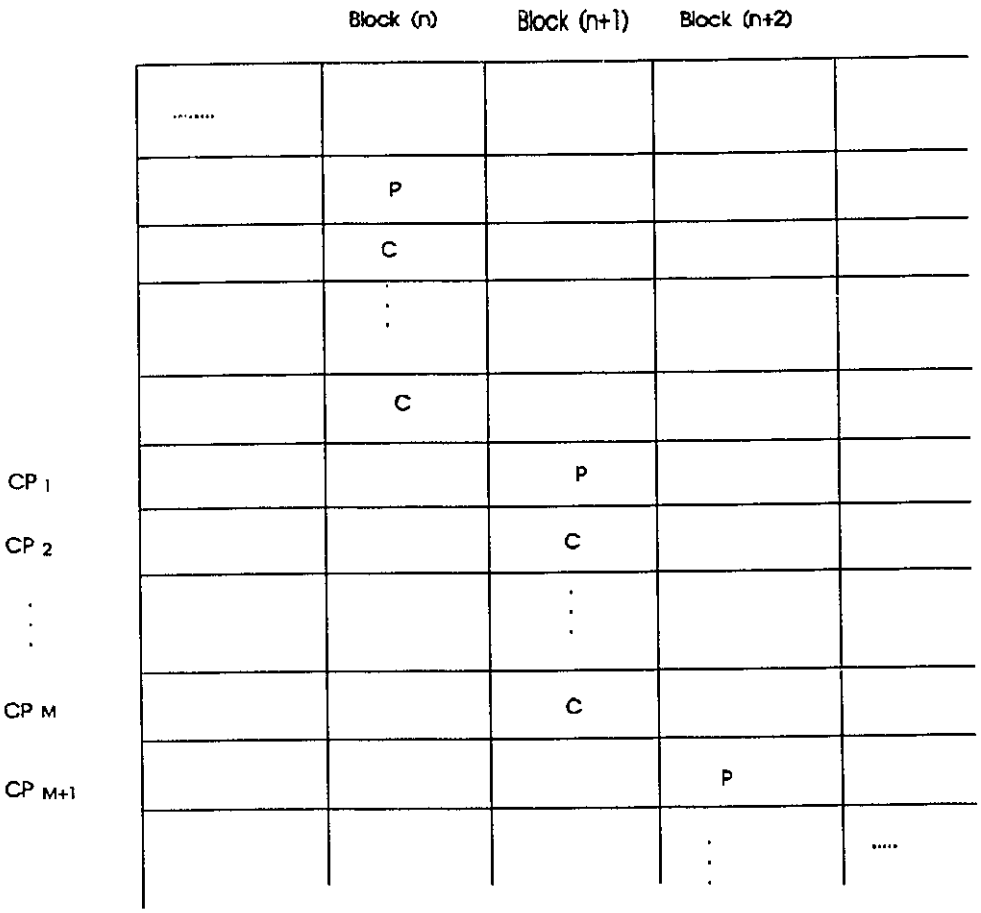

Fig. 2.1

Structure of the BPC Family

2.3 Analysis of the Stability Bound of the NWP/BPC Method

The NWP/BPC method presented in [10,24] is a member of the non-iterated corrector BPC sub-family where $r^p=r=k$, and r^p is the order of predictor. For the NWP/BPC formulation, there are two cases to be considered:

1) if $r \geq s+1$, we have

$$A_0=A_1=\begin{bmatrix} 1 & 0 & \dots & 0 \\ 1 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 1 & 0 & \dots & 0 \end{bmatrix} \qquad B_0=\begin{bmatrix} \beta_{11}^p & \dots & \beta_{1r}^p \\ \dots & \dots & \dots \\ \beta_{s1}^p & \dots & \beta_{sr}^p \end{bmatrix}$$

$$B_1=\begin{bmatrix} \beta_{11}^c & \dots & \beta_{1s}^c \\ \dots & \dots & \dots \\ \beta_{s1}^c & \dots & \beta_{ss}^c \end{bmatrix} \qquad B_2=\begin{bmatrix} \beta_{1,s+1}^c & \dots & \beta_{1,r}^c & \dots \\ \dots & \dots & \dots & O_{sxs}\dots \\ \beta_{s,s+1}^c & \dots & \beta_{s,kr}^c & \dots \end{bmatrix}$$

where $O_{(.x.)}$ is a 0 sub-matrix of the size as described as subscript.

2) if $r \leq s$, we have

$$A_0=A_1=\begin{bmatrix} 1 & 0 & \dots & 0 \\ 1 & 0 & \dots & 0 \\ \dots & \dots & \dots & .. \\ 1 & 0 & \dots & 0 \end{bmatrix} \qquad B_0=\begin{bmatrix} \beta_{11}^p & \dots & \beta_{1r}^p \\ \dots & \dots & \dots \\ \beta_{s1}^p & \dots & \beta_{sr}^p \end{bmatrix}$$

To facilitate the analysis of the stability structure of the NWP/BPC method, we define:

if $r \geq s+1$:

$$B_1=\begin{bmatrix} \beta_{11}^{c} & \cdots & \beta_{1r}^{c} & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & O_{sx(s-r+1)}\cdots \\ \beta_{s1}^{c} & \cdots & \beta_{sr} & \cdots & \cdots \end{bmatrix} \qquad B_2=0$$

$$\overline{Y_0}=(y_n,y_{n-1},\ldots,y_{n-r+1})^T$$

$$\overline{Y_1^0}=(y_{n+s}^0,y_{n+s-1}^0,\ldots,y_{n+1}^0,y_n,\ldots,y_{n+s-r+1})^T$$

$$\overline{Y_1}=(y_{n+s},y_{n+s-1},\ldots,y_{n+1},y_n,\ldots,y_{n+s-r+1})^T$$

if r≤s:

$$\overline{Y_0}=(y_n,y_{n-1},\ldots,y_{n-r+1},\ldots y_s)^T$$

$$\overline{Y_1^0}=(y_{n+s}^0,y_{n+s-1}^0,\ldots,y_{n+1}^0,y_n)^T$$

$$\overline{Y_1}=(y_{n+s},y_{n+s-1},\ldots,y_{n+1},y_n)^T$$

With these definitions, (2.1) can be rewritten as:

$$\overline{Y_1^0}=A\overline{Y_0}+HB^pF(\overline{Y_0}) \qquad (2.2.a)$$

$$\overline{Y_1}=A\overline{Y_0}+HB^cF(\overline{Y_1^0}); \qquad (2.2.b)$$

where A, B^p, and B^c are matrices of size r x r, if r≥s+1, or (s+1) x (s+1), if r≤s, and have the following forms:

if r≥s+1

$$A=\begin{bmatrix} & A_0 \\ I_{(r-s)x(r-s)} & 0_{(r-s)xs} \end{bmatrix}$$

$$B^p=\begin{bmatrix} B_0 \\ 0_{(r-s)xr} \end{bmatrix} \qquad B^c=\begin{bmatrix} B_1 \\ 0_{(r-s)xr} \end{bmatrix}$$

if r≤s

$$A=\begin{bmatrix} A_0 & 0_{rx(s-r+1)} \\ & 0_{1x(s+1)} \end{bmatrix}$$

$$B^p=\begin{bmatrix} B_0 & 0_{sx(s-r+1)} \\ & 0_{1x(s+1)} \end{bmatrix} \qquad B^c=\begin{bmatrix} B_1 & 0_{(sx(s-r+1)} \\ & 0_{1x(s+1)} \end{bmatrix}$$

Using the standard technique and substituting the test problem y'=λy into (2.2), we obtain:

$$\overline{Y_1^0}=A\overline{Y_0}+\lambda HB^p\overline{Y_0} \qquad (2.3.a)$$

$$\overline{Y_1}=A\overline{Y_0}+\lambda HB^c\overline{Y_1^0} \qquad (2.3.b)$$

It then follows that:

$$\overline{Y_1}=\Gamma(\lambda H)\overline{Y_0} \qquad (2.4)$$

where

$$\Gamma(\lambda H)=A+\lambda H B^{c}A+(\lambda H)^2(B^{c})^2B^{p} \qquad (2.5)$$

This matrix is called stability matrix. Its size is r x r if r≥s+1 or (s+1) x (s+1) if r≤s, and its entries are polynomials in λH of degree ≤ 2.

The stability behaviour of interest corresponds to ensuring that the limiting values resulting from repeated application of the equation (2.4) is zero. This can occur if and only if all eigenvalues of the stability matrix (2.5) have modulus less than or equal to one subject to the constraint that any eigenvalue equal to one must be simple. The area on the complex λH-plane which satisfies the above condition is called absolute stability region. In this thesis we restrict our interest to the portion of the real axis that exists within the stability region. This is called the stability interval and the absolute value of its left-most point is the stability bound.

Table 2.1 shows some results of the stability bound evaluations based on the stability matrix (2.5). (Note that the row designated as AB/AM (i.e. Adams Bashforth/Adams Moulton) corresponds to N=1; i.e., the single processor case. Recall also that s=N throughout these considerations. From table 2.1, we can see that the NWP/BPC method has the following stability properties:

1) For all cases, the stability bound is smaller than the AB/AM PECE method if r≤5. This feature indicates that for the stability sensitive problems, there is a limited possibility for speed-up with the NWP/BPC approach.

2) When r≤7, the stability bound increases as the number of processors goes up until the condition r=s+1 is reached. When r≤s, the stability bound remains almost constant.

3) when r≥8, a dramatic increase in the stability bound happens for r=s+1.

This feature suggests that r=s+1 is an optimal match of the order of the method and the number of processors. Note also that on the sub-diagonal that corresponds to r=s+1, the stability bound values decrease only marginally as r increases.

4) When r≥6 and r≤s+1, the stability bounds are larger when s>1 than when s=1 (the single processor case). This property of the NWP/BPC method suggests that the use of multiple processors can provide a speed-up when higher order formulas are used.

The results given in Table 2.1 indicate that the stability bound for this member of BPC family is relatively small. This observation and related analysis in [25] and [26] suggests that the relatively small stability region is a main drawback of the methods in the BPC family.

To increase the stability bound, the application of more than one corrector iteration to the solution procedure is considered in the next section. In particular, the stability properties of such members of the BPC family are investigated.

	r=3	r=4	r=5	r=6	r=7	r=8	r=9
s=1 (AB/AM)	1.73	1.28	0.934	0.696	0.523	0.381	0.284
s=2	1.15	0.825	0.579	0.404	0.281	0.195	0.135
s=3	1.09	0.977	0.837	0.438	0.253	0.161	0.109
s=4	1.06	0.953	0.884	0.481	0.236	0.125	0.100
s=5	1.04	0.939	0.875	0.873	0.390	0.142	0.095
s=6	1.03	0.929	0.867	0.833	0.808	0.223	0.095
s=7	1.02	0.925	0.862	0.829	0.808	0.792	0.148
s=8	1.02	0.920	0.862	0.829	0.804	0.788	0.781
s=9	1.01	0.920	0.858	0.825	0.804	0.788	0.777
s=10	1.01	0.914	0.857	0.823	0.803	0.787	0.779

Table 2.1
Stability Bound Values for the NWP/BPC Methods

2.4 Stability Bound of the NWP/BPC Method with Corrector Iterations

For the single processor PC methods with a number of corrector iterations, the theoretical analysis of the stability structure [27] given by H. J. Stetter shows that the stability region generally does not converge towards that of the corrector with an increasing number of corrector applications. He notes also that for a small number of corrector iterations a uniform increase in stability region is not guaranteed.

In this section, we investigate the stability properties of the NWP/BPC formulas when corrector iterations are introduced (i.e., the case of M>2 in (2.1)). By a procedure similar to that used in obtaining (2.2) from (2.1), we can similarly develop the following results:

$$\overline{Y_1^0}=A\overline{Y_0}+HB^pF(\overline{Y_0}) \qquad (2.6.a)$$

$$\overline{Y_1^m}=A\overline{Y_0}+HB^cF(\overline{Y_1^{m-1}}) \qquad (2.6.b)$$

Again, we substitute the test problem $y'=\lambda y$ into (2.6), and obtain the following stability matrix of the NWP/BPC formula with m corrector iterations:

$$\Gamma(\lambda H)=A+\lambda HB^cA+\ldots+(\lambda H)^mB^cB^p \qquad (2.7)$$

where A, B^p, B^c are as defined in section 2.3.

Some results of an investigation of the stability behaviour for the NWP/BPC formula with a number of corrector iterations are shown in Fig. 2.2. These results show the dependence of the stability bound on the parameter (M-1) in equation (2.1). Four values of order, r, are treated. The single processor case (i.e., s=1) is the Adams-Bashforth/Adams-Moulton method. The multi-processor cases are constrained by the condition r=s+1.

The results show that the stability bounds for the NWP/BPC formula increase with an

increasing number of corrector iterations for r=5,7 and 9. This is quite different from the behaviour for the AB/AM cases which are either erratic (r=3,5,7) or flat (r=9). The results of Fig. 2.2 show that the iterated corrector approach within the BPC context is effective from the point of view of increased stability bound. The challenge then is to maintain efficiency. When the number of corrector iterations goes up, the number of function evaluations per processor will go up correspondingly. The issue is whether or not an increase in cost effectiveness can be achieved in conjunction with the associated increase in stability bound.

In the following chapter, we propose some new methods for the PBPC family that is shown in Fig. 1.1. We refer to these as the PBPC/2 and the PBPC/3 methods. These methods use more than one corrector iteration but, by distributing the processors over multiple blocks, they carry out simultaneous corrector computations. The advantage of these methods is that we can increase the number of corrector iterations while keeping the number of function evaluations per processor per step relatively small. This means that we have the potential to achieve an improvement in efficiency in a multi-processor context.

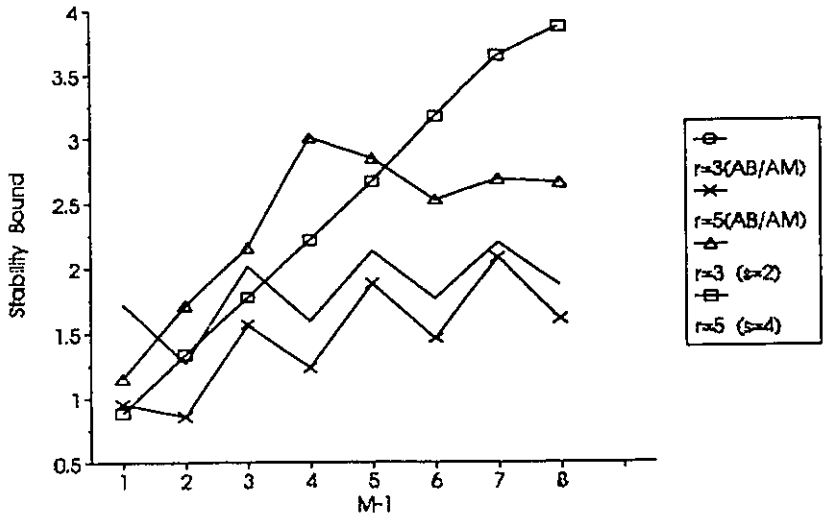

Fig. 2.2.a

The Stability Behaviour of the NWP/BPC Method with Corrector Iterations (r=3,r=5)

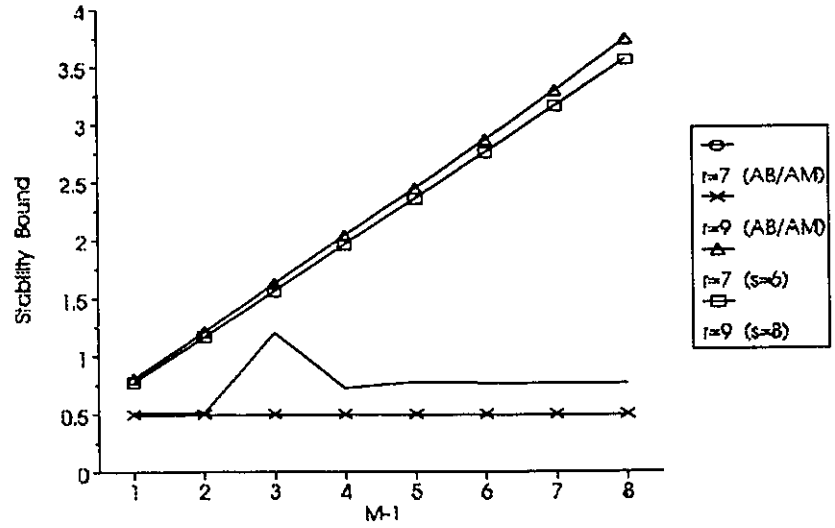

Fig. 2.2.b

The Stability Behaviour of the NWP/BPC Method with Corrector Iterations (r=7,r=9)

3. Parallel Block Predictor-Corrector Methods

3.1 Introduction

The methods discussed in this chapter (PBPC/2 and PBPC/3) can be regarded as extensions of the parallel predictor-corrector (PPC) method as discussed in [24]. The basic feature of these methods is that they simultaneously produce solution values (both predicted and (updated) corrected), in two adjacent blocks during any computational period (see Fig. 3.1).

The effectiveness of the PPC method is limited, in part, by the restriction on its stability bound that results from a delay in information usage. The use of multiple corrector iterations is a promising approach for overcoming this difficulty. The PBPC/2 and PBPC/3 methods have been formulated with this in mind.

The PBPC/2 and PBPC/3 methods apply 3 and 5 corrector iterations respectively and belong to the iterated corrector sub-family of methods in the PBPC family shown in Fig. 1.1. Since the additional corrector iterations are applied in parallel within the two active blocks, the number of function evaluations per processor per step is smaller than the total number of corrector iterations. This feature of PBPC/2 and PBPC/3 provides the potential opportunity to maintain, or even improve, efficiency when the stability bound is increased by increasing the number of corrector iterations.

In this chapter, we first introduce the structure of the PBPC methods. The specification for the parameters of these methods is then given. A stability analysis of the PBPC/2 and the PBPC/3 methods is carried out and results for the cases where order ranges from 3 to 9 are provided. A comparison with the PPC method and some members of the BPC family is also provided.

3.2. The Structure of the PBPC/M Methods

The structure of the PBPC/M method is presented as Fig. 3.1. The underlying idea is to divide the t axis between $t=t_0$ and $t=t_f$ into a sequence of adjacent blocks, each containing s equally spaced values, where $s=N/2$ and N is the number of available processors (N is even). The value

of M is the number of function evaluations per processor per step. The method PBPC/1; i.e., M=1, corresponds to the PPC method as discussed in [24]. Our principal interest is with the cases where M=2 and M=3.

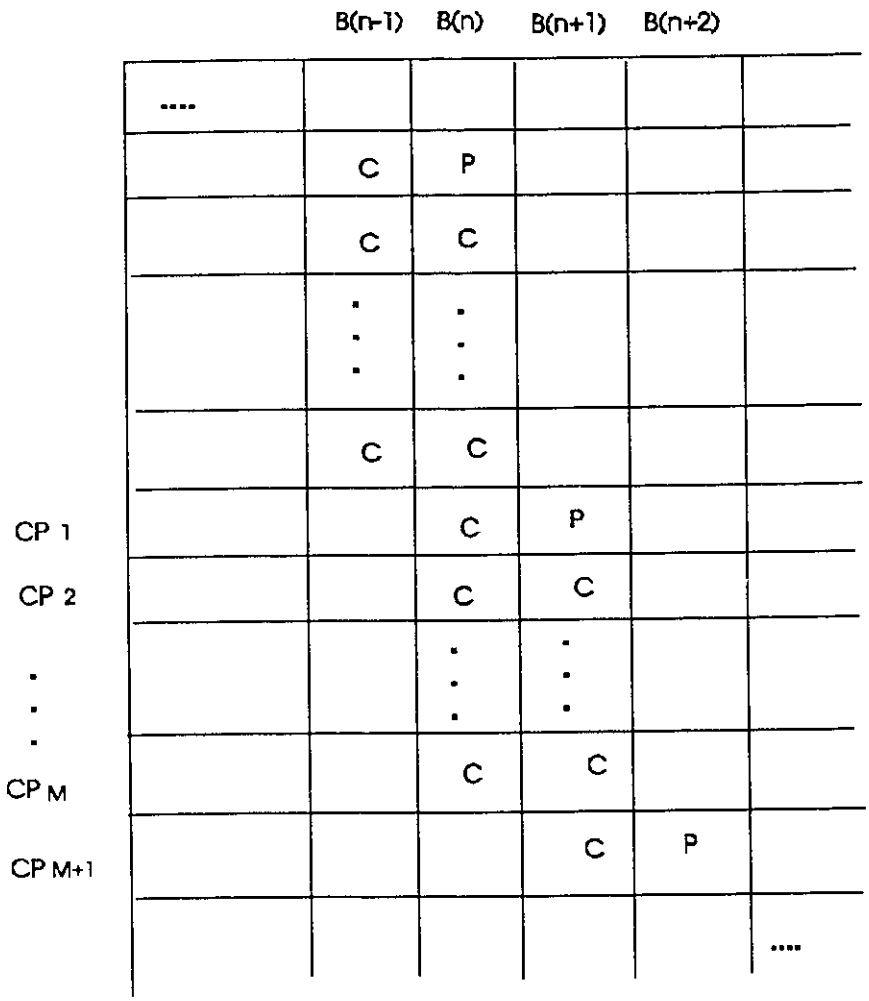

Fig. 3.1

Structure of the PBPC Family

From Fig. 3.1 it can be seen that the task during any step is distributed over M computation periods. There are as follows:

a) During the first computation period:

i) Use the $(M-1)^{st}$ corrected values available in block n and the final solution values from previous blocks to produce predicted solution values in block (n+1).

ii) Use the $(M-1)^{st}$ corrected values available in block n and the final solution values from previous blocks to produce the M^{st} corrected solution values in block n.

b) During the $(m+1)^{st}$ computation period ($1 \leq m \leq M-1$):

i) Use the most recent values available in block (n+1) together with the $(M+m-1)^{th}$ corrected values produced in block n (during the most recent computation period of the current step) and the final solution values from previous blocks to produce the m^{st} corrected solution values in block (n+1). (If m=1, the most recent values available in block (n+1) are the predicted values and if m>1 they are the corrected values from the last computation period).

ii) Use the $(M+m-1)^{st}$ corrected values produced in block n during the most recent computation period of the current step in block n and the final solution values from previous blocks and to produce the $(M+m)^{st}$ corrected solution values in block n. If m=M-1, the $(M+m)^{th}$ corrected solution values will be saved as the final solution values in block n.

This process can be presented by the following groups of equations (we use Y^0 to represent a predicted value):

For the first computation period, we have

$$Y_1^0 = A_1^p Y_{-1} + H[B_1^p F(Y_0^{M-1}) + B_2^p F(Y_{-1})] \qquad (3.1.a)$$

$$Y_0^M = A_1^c Y_{-1} + H[B_1^c F(Y_0^{M-1}) + B_2^c F(Y_{-1})] \qquad (3.1.b)$$

For the $(m+1)^{st}$ computation period ($1 \le m \le M-1$), we have

$$Y_1^m = A_2^c Y_0^{M+m-1} + A_3^c Y_{-1} + H[B_3^c F(Y_1^{m-1}) + B_4^c F(Y_0^{M+m-1}) + B_5^c F(Y_{-1})] \qquad (3.2.a)$$

$$Y_0^{M+m} = A_1^c Y_{-1} + H[B_1^c F(Y_0^{M+m-1}) + B_2^c F(Y_{-1})] \qquad (3.2.b)$$

Combining the above equations, (3.1) and (3.2) can be rewritten as:

$$\begin{bmatrix} Y_1^0 \\ Y_0^M \end{bmatrix} = \begin{bmatrix} 0_{sxs} & A_1^p \\ 0_{sxs} & A_1^c \end{bmatrix} \begin{bmatrix} Y_0^{M-1} \\ Y_{-1} \end{bmatrix} + H \begin{bmatrix} B_1^p & B_2^p \\ B_1^c & B_2^c \end{bmatrix} \begin{bmatrix} F(Y_0^{M-1}) \\ F(Y_{-1}) \end{bmatrix} \qquad (3.3)$$

$$\begin{bmatrix} Y_1^m \\ Y_0^{M+m} \end{bmatrix} = \begin{bmatrix} 0_{sxs} & A_2^c & A_3^c \\ 0_{sxs} & 0_{sxs} & A_1^c \end{bmatrix} \begin{bmatrix} Y_1^{m-1} \\ Y_0^{M+m-1} \\ Y_{-1} \end{bmatrix} + H \begin{bmatrix} B_3^c & B_4^c & B_5^c \\ 0_{sxs} & B_1^c & B_2^c \end{bmatrix} \begin{bmatrix} F(Y_1^{m-1}) \\ F(Y_0^{M+m-1}) \\ F(Y_{-1}) \end{bmatrix} \qquad (3.4)$$

where $Y_{-1} = (y_{n-s}, y_{n-s-1}, \ldots, y_{n-s-k+1})^T$ is a k-vector whose entries are saved solution values from blocks to the left of block n. $Y_1^0 = (y_{n+s}^0, y_{n+s-1}^0, \ldots, y_{n+1}^0)^T$ is an s-vector of the predicted solution values in block (n+1), and $Y_1^q = (y_{n+s}^q, y_{n+s-1}^q, \ldots, y_{n+1}^q)^T$, q=1,2,...,M-1, is an s-vector of the corrected solution values in block (n+1) for the q^{th} iteration. $Y_0^q = (y_n^q, y_{n-1}^q, \ldots, y_{n-s+1}^q)^T$, q=M,M+1,...,2M+1, is an s-vector whose entries are the q^{th} corrected solution values in block n. Also $A_1^p, B_2^p, A_1^c, A_3^c$,

B_2^c, B_5^c are s x k matrices, and A_2^c, B_1^p, B_1^c, B_2^c, B_3^c are s x s matrices.

As noted earlier, each step corresponds to M computation periods. Consequently the generation of the final solution values produced by the PBPC/M method takes place over two steps. One of the important features of this structure is that there are 2M-1 corrector iterations used in producing the final solution, but the number of function evaluations per processor during any step is only M.

3.3 Specification of the PBPC/M Formulas

The formulas of (3.1) and (3.2) fall within the context of linear multistep methods. In developing specific formulas for (3.1) and (3.2), we restrict our considerations to an Adams model and define k = r-s where r is the order of corrector.

(3.1.a) corresponds to the predictor evaluation in block (n+1). Two cases need to be considered and these depend on the relationship between s and r^p which is the order of the predictor (we consider only the cases where $r^p \leq r$). They are:

1) if $r^p \leq s$

$$A_1^p = \begin{bmatrix} 1 & \\ 1 & O_{(k-1)x(k-1)} \\ .. & \\ 1 & \end{bmatrix} \qquad B_1^p = \begin{bmatrix} \beta_{11}^p & ..... & \beta_{1r^p}^p & \\ ... & ... & ... & O_{(s-r^p)x(s-r^p)} \\ \beta_{s1}^p & ..... & \beta_{sr^p}^p & \end{bmatrix} \qquad B_2^p = 0$$

2) if $r^p \geq s+1$

$$A_1^P=\begin{bmatrix}1 & \\ 1 & O_{(k-1)x(k-1)} \\ .. & \\ 1 & \end{bmatrix} \quad B_1^P=\begin{bmatrix}\beta_{11}^P & .... & \beta_{1s}^P \\ .. & ... & ... \\ \beta_{s1}^P & ... & \beta_{ss}^P\end{bmatrix} \quad B_2^P=\begin{bmatrix}\beta_{1,s+1}^P & ..... & \beta_{1,r^P}^P & \\ ... & ... & ... & O_{(k-(r^P-s))x(k-(r^P-s))} \\ \beta_{s,s+1}^P & ..... & \beta_{s,r^P}^P & \end{bmatrix}$$

where $O_{(.x.)}$ is a 0 matrix whose size is specified by its subscripts.

In total, there are (2M-1) corrector iterations. The first (M-1) of them take place using (3.2.a). In this instance, three cases need to be considered and these depend on the relationship between s and r where r is the order of the corrector. They are:

1) if r≤s

$$A_2^c=\begin{bmatrix}1 & \\ 1 & O_{(s-1)x(s-1)} \\ .. & \\ 1 & \end{bmatrix} \quad A_3^c=0$$

$$B_3^c=\begin{bmatrix}\beta_{11}^c & ... & \beta_{1r}^c & \\ ... & ... & ... & O_{(s-r)x(s-r)} \\ \beta_{s1}^c & ... & \beta_{sr}^c & \end{bmatrix} \quad B_4^c=0 \quad B_5^c=0$$

2) if 2s+1≥r≥s+1

A_2^c and A_3^c are as the same as case 1), and

$$B_3^c=\begin{bmatrix}\beta_{11}^c & \cdots & \beta_{1s}^c\\ \cdots & \cdots & \cdots\\ \beta_{s1}^c & \cdots & \beta_{ss}^c\end{bmatrix} \quad B_4^c=\begin{bmatrix}\beta_{1,s+1}^c & \cdots & \beta_{1,r}^c & \\ \cdots & \cdots & \cdots & O_{(2s-r)x(2s-r)}\\ \beta_{s,s+1}^c & \cdots & \beta_{s,r}^c & \end{bmatrix} \quad B_5^c=0$$

3) if r≥2s+1

Again, A_2^c and A_3^c are as the same as case 1), and

$$B_3^c=\begin{bmatrix}\beta_{11}^c & \cdots & \beta_{1s}^c\\ \cdots & \cdots & \cdots\\ \beta_{s1}^c & \cdots & \beta_{ss}^c\end{bmatrix} \quad B_4^c=\begin{bmatrix}\beta_{1,s+1}^c & \cdots & \beta_{1,2s}^c\\ \cdots & \cdots & \cdots\\ \beta_{s,s+1}^c & \cdots & \beta_{s,2s}^c\end{bmatrix} \quad B_5^c=\begin{bmatrix}\beta_{1,2s+1}^c & \cdots & \beta_{1,r}^c & \\ \cdots & \cdots & \cdots & O_{(k-(r-2s))x(k-(r-2s))}\\ \beta_{s,2s+1}^c & \cdots & \beta_{s,r}^c & \end{bmatrix}$$

The second group of corrector iterations (there are M in this group) takes place using (3.1.b) once, and (3.2.b) (M-1) times. These equations however share the same coefficient matrices. Two cases need to be considered. They are:

1) if r≤s

$$A_1^c=\begin{bmatrix}1 & \\ 1 & O_{(k-1)x(k-1)}\\ \cdots & \\ 1 & \end{bmatrix} \quad B_1^c=\begin{bmatrix}\beta_{11}^c & \cdots & \beta_{1r}^c & \\ \cdots & \cdots & \cdots & O_{(s-r)x(s-r)}\\ \beta_{s1}^c & \cdots & \beta_{sr}^c & \end{bmatrix} \quad B_2^c=0$$

2) if r≥s+1

$$A_1^c=\begin{bmatrix}1 & \\ 1 & O_{(k-1)x(k-1)} \\ .. & \\ 1 & \end{bmatrix} \quad B_1^c=\begin{bmatrix}\beta_{11}^c & ..... & \beta_{1s}^c \\ ... & ... & .... \\ \beta_{s1}^c & ..... & \beta_{ss}^c\end{bmatrix} \quad B_2^c=\begin{bmatrix}\beta_{1,s+1}^c & .... & \beta_{1,r}^c \\ ... & ... & ... \\ \beta_{s,s+1}^c & .... & \beta_{s,r}^c\end{bmatrix}$$

To obtain the coefficients β_{ij}^p, i=1,2,...,s; j=1,2,...,r^p,in (3.1.a), the following integration-based formula is used

$$y_{n+1}^0=y_{n-s}+H\int_{t_{n-s}}^{t_{n+1}} P_1(t)dt \qquad (3.5)$$

where, $P_1(t)$ is a Lagrange forward interpolating polynomial which can be written as:

$$P_1(t)=\sum_{j=1}^{r^p}\left(\prod_{l=1,j\neq l}^{r^p}\frac{(t-t_{n-l+1})}{(t_{n-j+1}-t_{n-l+1})}\right)f(y_{n-j+1})$$

Let $t-t_{n-s}=Hv$. Then, the above polynomial can written as follows:

$$P_1(v)=\sum_{j=1}^{r^p}\prod_{l=1,j\neq l}^{r^p}\frac{(sv+(l-s-1))}{(l-j)}f(y_{n-j+1})=\sum_{j=1}^{r^p}w_j(v)f(y_{n-j+1})$$

$$w_j(v)=\prod_{l=1,j\neq l}^{r^p}\frac{(sv+(l-s-1))}{(l-j)}$$

The coefficients in question are then

$$\beta_{ij}^{P}=\int_{0}^{(i+s)/s} w_j(v)dv, \qquad i=1,2,..,s;\ j=1,2,..,r^P \qquad (3.6)$$

In a similar way, we can produce the coefficients of the two group of correctors by an integration-based formulas (Note that the two groups of correctors; i.e. (3.2.a) and (3.1.b) together with (3.2.b), have the same formulation in this specification).

For (3.2.a), we have

$$y_{n+i}=y_n+\int_{t_n}^{t_{n+i}} P_{21}(t)dt \qquad (3.7)$$

where $P_{21}(t)$ is a Lagrange backward interpolating polynomial

$$P_{21}(t)=\sum_{j=1}^{r}\prod_{l=1,j\neq l}^{r}\frac{(t-t_{n+s-l+1})}{(t_{n+s-j+1}-t_{n+s-l+1})}f(y_{n+s-j+1})$$

and for (3.1.b) and (3.2.b), we have

$$y_{n-s+i}=y_{n-s}+\int_{t_{n-s}}^{t_{n-s+i}} P_{22}(t)dt \qquad (3.8)$$

where $P_{22}(t)$ is also a Lagrange backward interpolating polynomial

$$P_{22}(t)=\sum_{j=1}^{r}\prod_{l=1,j\neq l}^{r}\frac{(t-t_{n-l+1})}{(t_{n-j+1}-t_{n-l+1})}f(y_{n-j+1})$$

Letting $t-t_{n+i}=Hv$ for (3.5) and $t-t_{n-s+i}=Hv$ for (3.6), we have

$$P_{21}(v)=\sum_{j=1}^{r}\prod_{l=1,j\neq l}^{r}\frac{(sv+i+l-s-1)}{(l-j)}f(y_{n+s-j+1})$$

and

$$P_{21}(v)=\sum_{j=1}^{r}\prod_{l=1,j\neq l}^{r}\frac{(sv+i+l-s-1)}{(l-j)}f(y_{n-j+1})$$

respectively.

If we let

$$w_j(v)=\prod_{l=1,j\neq l}^{r}\frac{(sv+i+l-s-1)}{(l-j)}$$

then the coefficients of the correctors are given by

$$\beta_{ij}^{c}=\int_{-i/s}^{0}w_j(v)dv \qquad i=1,2,..,s;\ j=1,2,..,r \qquad (3.9)$$

Some important features in the PBPC/M formula should be noted:

a) The lower integration limit (3.5) is t_{n-s}, the right-most time point in the previous block (n-1). Thus the predictor values in block (n+1) directly depend on the right-most value y_{n-s} in block (n-1).

b) The lower integration limit for the first group of corrector formulas (3.7) is t_n (these apply to block (n+1)). This indicates that the updated corrected values in block n are used for producing the first group of corrected values in block (n+1).

c) The lower integration limit for the second group of corrector formulas (3.8) is t_{n-s} (these apply to block n). Thus these correctors use final, rather than intermediate, solution information from block (n-1).

To illustrate the application of the above specifications, we give in Appendix 1, several specific sets of formulas corresponding to different order and number of processors.

3.4 Analysis of the Stability Properties of the PBPC/M Methods

3.4.1 Stability Matrix of the PBPC/M Methods

In this section, we discuss the stability properties of the PBPC/M methods whose formulation is given in section 3.3, in particular, the PBPC/2 and the PBPC/3 methods.

To facilitate the analysis of the stability structure of the PBPC/M methods described in (3.3) through (3.9), we define a collection of M+1 vectors; namely, $\overline{Y_0}$, $\overline{Y_1}^0$, and $\overline{Y_1}^m$, m=1,2,..,M-1. Their specification, however, depends on the relative values of r and s. Two cases are distinguished:

i) If r≤s

$$\overline{Y_0}=(y_n^{M-1},..,y_{n-s+1}^{M-1},y_{N-s},..,y_{n-2s+1},y_{n-2s})^T$$

$$\overline{Y_1^0}=(y_{n+s}^0,\cdots,y_{n+1}^0,y_n^M,\cdots,y_{n-s+1}^M,y_{n-s})^T$$

$$\overline{Y_1^m}=(y_{n+s}^m,\cdots,y_{n+1}^m,y_n^{M+m},\cdots,y_{n-s+1}^{M+m},y_{n-s})^T$$

m=1,2,...,M-1

Each of the above is a (2s+1) vector, and

ii) If r≥s+1

$$\overline{Y_0}=(y_n^{M-1},\cdots,y_{n-s+1}^{M-1},y_{n-s},\cdots,y_{n-r-s+1})^T$$

$$\overline{Y_1^0}=(y_{n+s}^0,\cdots,y_{n+1}^0,y_n^M,\cdots,y_{n-s+1}^M,y_{n-s},\cdots,y_{n-r+1})^T$$

$$\overline{Y_1^m}=(y_{n+s}^m,\cdots,y_{n+1}^m,y_n^{M+m},\cdots,y_{n-s+1}^{M+m},y_{n-s},\cdots,y_{n-r+1})^T$$

m=1,2,...,M-1

Here each of the above is an (s+r) vector.

Let

$$B^P=\begin{bmatrix}\beta_{11}^P & \cdots\cdots & \beta_{1r}^P \\ \cdots & \cdots\cdots & \cdots \\ \beta_{s1}^P & \cdots\cdots & \beta_{sr}^P\end{bmatrix}$$

be an s x r^p matrix whose entries are the coefficients presented in (3.6), and

$$B^c=\begin{bmatrix} \beta_{11}^c & \cdots\cdots & \beta_{1r}^c \\ \cdots & \cdots & \cdots \\ \beta_{s1}^c & \cdots\cdots & \beta_{sr}^c \end{bmatrix}$$

be an s x r matrix whose entries are the coefficients presented in (3.9).

With the above definitions, equation (3.3) can be rewritten as:

$$\overline{Y_1^0}=\overline{A}\ \overline{Y_0}+H\overline{B}F(\overline{Y_0}) \qquad (3.10.a)$$

Similarly equation (3.4) can be rewritten as:

$$\overline{Y_1^m}=A^*\overline{Y_1^{m-1}}+HB^*F(\overline{Y_1^{m-1}}) \qquad (3.10.b)$$

m=1,2,..,M-1

To facilitate our later analysis, we let $\overline{Y}_1 = \overline{Y_1^{M-1}}$.

If r≤s

$$\overline{A}=\begin{bmatrix} O_{sxs} & E_{sxs} & O_{sx1} \\ O_{sxs} & E_{sxs} & O_{sx1} \\ O_{1xs} & 1 & O_{1xs} \end{bmatrix}$$

$$A^* = \begin{bmatrix} O_{sxs} & E_{sxs} & O_{sx1} \\ O_{sxs} & O_{sxs} & 1_{sx1} \\ O_{1xs} & O_{1xs} & 1 \end{bmatrix}$$

$$\overline{B} = \begin{bmatrix} B^p & O_{s\ x(2s+1-r^p)} \\ B^c & O_{s\ x(2s+1-r)} \\ & O_{1\ x(2s+1)} \end{bmatrix}$$

$$B^* = \begin{bmatrix} B^c & O_{sxs} & O_{sx(s-r+1)} \\ O_{sxs} & B^c & O_{sx(s-r+1)} \\ & O_{1\ x(2s+1)} & \end{bmatrix}$$

Here $E_{(.\ x.)}$ is a matrix whose entries in the first column are 1 and all others are 0; its size is specified by its subscripts. 1_{sx1} is an s vector whose entries are all 1.

and if r≥s+1

$$\overline{A} = \begin{bmatrix} O_{s\ x\ s} & E_{sxs} & O_{s\ x(r-s)} \\ O_{s\ x\ s} & E_{sxs} & O_{s\ x(r-s)} \\ O_{(r-s)xs} & I_{(r-s)x(r-s)} & O_{(r-s)xs} \end{bmatrix}$$

$$A^* = \begin{bmatrix} O_{s\ x\ s} & E_{sxs} & O_{s\ x(r-s)} \\ O_{s\ x\ s} & O_{s\ x\ s} & E_{sx(r-s)} \\ O_{(r-s)x\ s} & O_{(r-s)x\ s} & I_{(r-s)x(r-s)} \end{bmatrix}$$

$$\overline{B}=\begin{bmatrix} B^p & O_{s\,x(s+r-r^p)} \\ B^c & O_{s\,x\,s} \\ & O_{(r-s)x(r+s)} \end{bmatrix}$$

$$B^*=\begin{bmatrix} B^c & O_{s\,x\,s} \\ O_{s\,x\,s} & B^c \\ & O_{(r-s)x(r+s)} \end{bmatrix}$$

where $I_{(r-s)x(r-s)}$ is an identity matrix of the size (r-s) x (r-s).

Substitution of the standard test problem y'=λy into (3.10), yields:

$$\begin{aligned}\overline{Y_1}=Y_1^{M-1}&=A^*Y_1^{M-2}+\lambda HB^*Y_1^{M-2}\\ &=(A^*+\lambda HB^*)(A^*+\lambda HB^*)....(A^*+\lambda HB^*)(\overline{A}+\lambda H\overline{B})\overline{Y_0}\\ &=(A^*+\lambda HB^*)^{M-1}(\overline{A}+\lambda H\overline{B})\overline{Y_0}\end{aligned}$$

Therefore the stability matrix for (3.10) is:

$$\Gamma(\lambda H)=(A^*+\lambda HB^*)^{M-1}(\overline{A}+\lambda H\overline{B}) \qquad (3.11)$$

The parameter r^p is an important parameter which can affect both the stability region and the efficiency of the method. Its determination will be discussed in the next section.

3.4.2 Integration Order of the Predictor

It is difficult to give a general statement about how the integration order of the predictor affects the stability properties of the PBPC/M methods. But, an empirical investigation indicates that the cases of a lower order predictor normally increase the stability bound of the method.

Table 3.1 shows some results of this investigation. In this table, r represents the order of the corrector.

		N=4	N=6	N=8	N=10	N=12	N=14	N=16
r=3	r^p=2	1.18	1.09	1.07	1.05	1.05	1.04	1.04
	r^p=3	0.954	0.954	0.935	0.926	0.926	0.917	0.917
r=5	r^p=2	1.18	1.26	1.42	1.09	1.09	1.09	1.09
	r^p=3	0.954	0.963	0.935	0.926	0.917	0.908	0.908
	r^p=4	0.847	1.00	0.881	0.872	0.872	0.864	0.864
	r^p=5	0.766	0.881	0.736	0.838	0.830	0.830	0.830
r=7	r^p=2	1.16	1.26	1.35	1.35	1.29	1.09	1.09
	r^p=3	0.944	0.963	0.935	0.926	0.917	0.908	0.908
	r^p=4	0.838	0.993	0.881	0.872	0.872	0.864	0.864
	r^p=5	0.759	0.881	0.736	0.838	0.830	0.830	0.864
	r^p=6	0.694	0.736	0.552	0.722	0.813	0.805	0.864
	r^p=7	0.643	0.569	0.418	0.471	0.500	0.790	0.864
r=9	r^p=2	0.944	1.24	1.22	1.27	1.26	1.30	1.21
	r^p=3	0.864	0.944	0.917	0.935	0.908	0.917	0.908
	r^p=4	0.797	0.954	0.872	0.864	0.872	0.864	0.864
	r^p=5	0.729	0.890	0.729	0.813	0.830	0.830	0.830
	r^p=6	0.673	0.694	0.552	0.729	0.813	0.797	0.805
	r^p=7	0.616	0.563	0.422	0.471	0.500	0.774	0.790
	r^p=8	0.529	0.471	0.326	0.313	0.307	0.430	0.805
	r^p=9	0.520	0.401	0.259	0.217	0.198	0.235	0.295

Table 3.1

Stability Bounds of the PBPC/2 Method for Different r^p

However, a predictor with too low an order may decrease the order of the overall method if a small number of corrector iterations is applied. However if a large number of corrector iterations is applied, it will increase the number of function evaluations per step, thereby, causing a deterioration in efficiency.

An analysis of the local truncation error for a general method of block-based parallel predictor-corrector schemes made by K. Jackson, S.P. Nϕrsett [28] and Burrage [20] has shown that each application of a corrector increase the order of the overall method by at least one until the order of corrector is reached. Using Butcher's series[31], Burrage [20] shows that when the number of corrections is such that the order cannot increase then the effect of more corrections is to shift the errors due to the predictor further away from the principal error term.

Our objective is to maintain or even improve efficiency when the stability bound increases. To achieve this, the order of the predictor must be sufficiently high so that a given accuracy level can be achieved with a small number of corrector iterations. Therefore, the determination of the order of a predictor formula should be based on the structure of the PBPC/M methods and an analysis of the local truncation error.

From insights arising from the work of Jackson, Nϕrsett and Burrage, a good choice for r^p appears to be r-1. This choice will likely provide a good balance between the achievement of efficiency and stability. We note in particular the advantage that when multiple correctors are applied the impact of the large principal error term of the predictor is reduced.

3.4.3 Stability Analysis of the PBPC/2 and PBPC/3 Methods

In this section, we analyze the stability behaviour of the PBPC/2 and PBPC/3 methods. In particular, values of the stability bound for PBPC/2 are shown in Table 3.2 and those for PBPC/3 are shown in Table 3.3. These results are obtained by setting M=2 and M=3 in equation (3.9) respectively.

For comparison, Table 3.4 and Table 3.5 show the stability bounds of the PBPC/1 method (i.e. the PPC method in [24]) and the NWP/BPC/3 method (i.e. the NWP/BPC method discussed in chapter 2, with 2 corrector iterations), respectively. The stability bounds of the NWP/BPC method (single corrector evaluation) are shown in Table 2.1.

Fig. 3.2 to Fig. 3.5 show that the stability bounds of the PBPC/1, PBPC/2, PBPC/3, NWP/BPC and the NWP/BPC/3 methods for the integration order r=3,5,7 and 9, respectively. It should also be noted that, for the PBPC family the relationship N=2s holds while for the methods in the BPC family N=s. From the comparison, we can note that:

i) For the methods of PBPC family, the additional corrector iterations associated with a

transition from method PBPC/1 to PBPC/2 to PBPC/3 generally has a substantial effect on increasing the stability bound for any particular integration order and number of processors. Only a few exceptions can be observed; i.e., for r=9 and N=8,10 and 12.

ii) When r>s+1, for any order (r) and number of points in one block (s), the stability behaviour of the PBPC/2 and PBPC/3 methods is superior to that of the NWP/BPC and NWP/BPC/3 methods, respectively. When r≤s+1, the NWP/BPC/3 cases have bigger stability bounds than the PBPC/3 cases, and the stability bounds of the NWP/BPC and the PBPC/2 are essentially the same.

	r=3	r=4	r=5	r=6	r=7	r=8	r=9
s=2	1.18	0.945	0.847	0.766	0.694	0.634	0.569
s=3	1.09	0.963	1.00	0.881	0.694	0.569	0.471
s=4	1.07	0.935	0.881	0.736	0.552	0.422	0.362
s=5	1.05	0.926	0.872	0.838	0.722	0.471	0.313
s=6	1.05	0.917	0.872	0.830	0.813	0.500	0.307
s=7	1.04	0.908	0.864	0.830	0.805	0.790	0.430
s=8	1.04	0.899	0.864	0.830	0.805	0.790	0.805
s=9	1.03	0.899	0.864	0.822	0.805	0.790	0.774
s=10	1.03	0.890	0.864	0.822	0.805	0.790	0.774

Table 3.2
Stability Bounds for the PBPC/2

	r=3	r=4	r=5	r=6	r=7	r=8	r=9
s=2	1.70	1.49	1.41	1.35	1.30	1.25	0.964
s=3	1.10	1.37	1.33	1.27	1.22	1.22	1.30
s=4	1.12	1.08	1.26	1.30	1.33	1.27	1.20
s=5	1.14	1.08	1.08	1.20	1.18	1.09	1.00
s=6	1.15	1.08	1.08	1.08	1.18	1.21	1.05
s=7	1.16	1.08	1.07	1.08	1.08	1.15	1.01
s=8	1.16	1.08	1.07	1.08	1.08	1.07	1.14
s=9	1.18	1.08	1.07	1.08	1.08	1.08	1.08
s=10	1.18	1.09	1.07	1.08	1.08	1.08	1.08

Table 3.3
Stability Bounds for the PBPC/3

	r=3	r=4	r=5	r=6	r=7	r=8	r=9
s=2	0.490	0.451	0.429	0.393	0.264	0.185	0.135
s=3	0.482	0.443	0.423	0.409	0.399	0.239	0.136
s=4	0.477	0.440	0.420	0.407	0.397	0.392	0.387
s=5	0.475	0.438	0.417	0.405	0.396	0.391	0.386
s=6	0.473	0.436	0.417	0.405	0.403	0.391	0.387
s=7	0.472	0.436	0.417	0.403	0.403	0.391	0.386
s=8	0.471	0.434	0.416	0.403	0.394	0.389	0.385
s=9	0.470	0.434	0.414	0.402	0.394	0.389	0.385
s=10	0.470	0.433	0.414	0.401	0.393	0.388	0.385

Table 3.4
Stability Bounds for the PBPC/1

	r=3	r=4	r=5	r=6	r=7	r=8	r=9
s=2	1.71	1.71	1.28	1.01	0.807	0.645	0.515
s=3	1.58	1.45	1.14	0.840	0.626	<0.50	<0.50
s=4	1.51	1.41	1.33	1.15	0.688	<0.50	<0.50
s=5	1.45	1.40	1.31	1.26	0.881	<0.50	<0.50
s=6	1.43	1.39	1.30	1.25	1.21	0.705	<0.50
s=7	1.40	1.38	1.28	1.24	1.21	1.19	<0.50
s=8	1.39	1.38	1.28	1.24	1.21	1.18	1.17
s=9	1.38	1.38	1.27	1.23	1.20	1.18	1.17
s=10	1.36	1.38	1.27	1.23	1.20	1.18	1.17

Table 3.5
Stability Bounds for the NWP/BPC/3

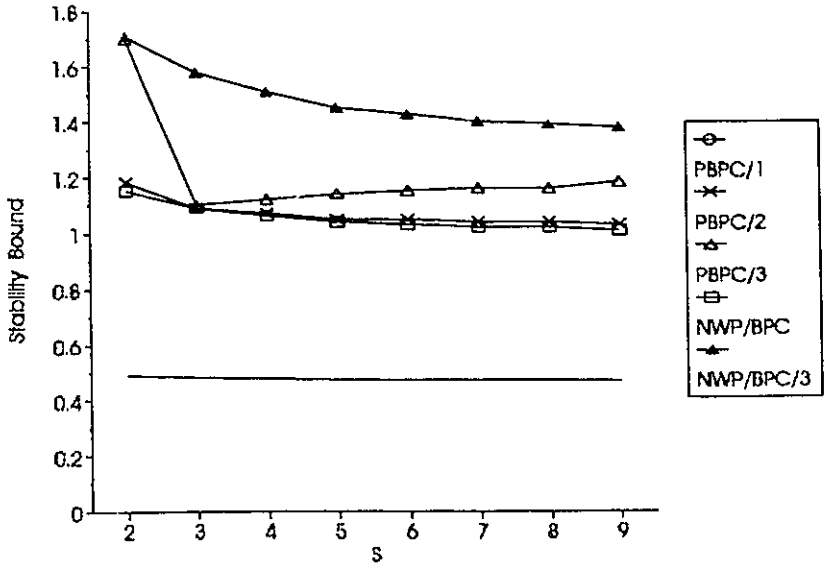

Fig. 3.2
Stability Bounds for the Case of r=3

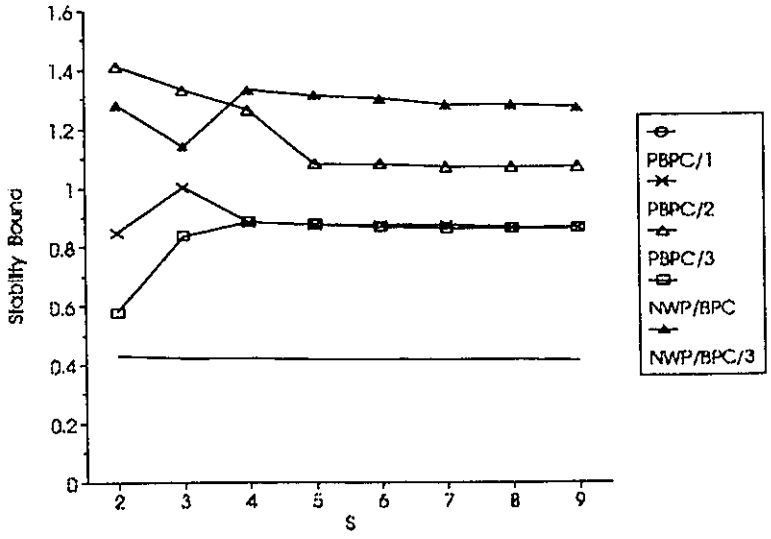

Fig. 3.3
Stability Bounds for the Case r=5

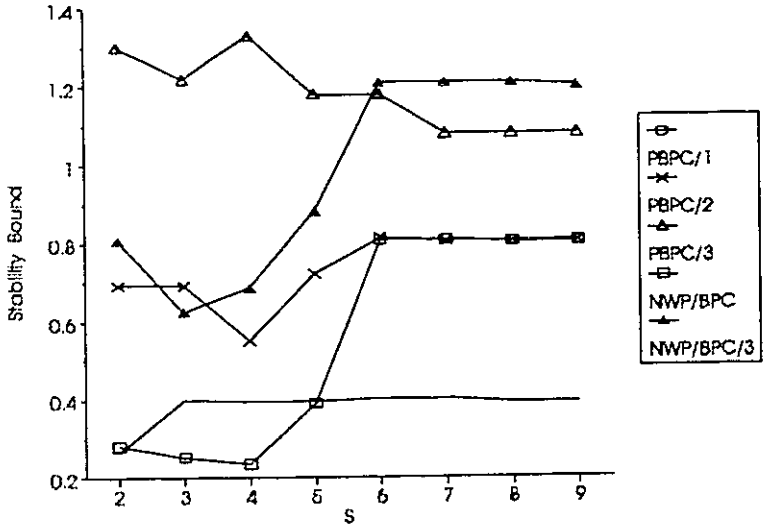

Fig. 3.4
Stability Bounds for the Case r=7

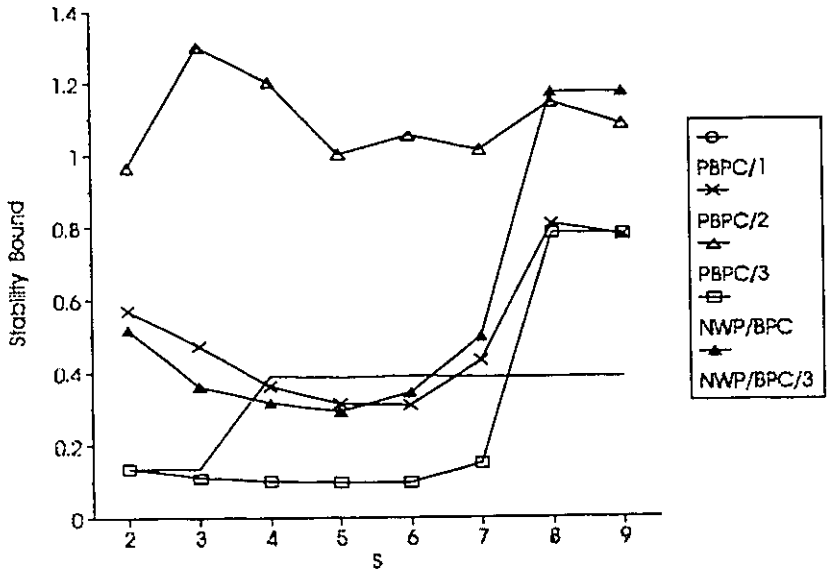

Fig. 3.5
Stability Bounds for the Case r=9

4 Numerical Experiments

4.1 Experiment Framework

A set of numerical experiments was carried out to obtain insight into the performance of the PBPC/2, PBPC/3 and NWP/BPC/3 methods. Also of interest was a comparison of the performance of these methods with that of the previously studied methods PBPC/1 and NWP/BPC in [24]. Two groups of test problems were considered. The first group includes five non-stiff test problems, and the second group is based on one of Krogh's problems [32] with different parameter values which increase the stiffness of the problem. All problems used have a known analytic solution.

The experiments for the first group of problems were organized to examine the behaviour of these methods with respect to three parameters; namely, number of available processors (N), order of the formulas (r) and admissible global solution error (G). The experiments for the second group of problems were organized to examine the behaviour of these methods as the stiffness of the test problems increases. All results were obtained using Fortran programs executing double precision arithmetic on an IBM/39 computer.

Like conventional predictor-corrector methods, all methods proposed and referenced in these experiments require a set of initial values to provide the priming data for the procedure. In these experiments, this data, (which corresponds to values at r^p points within the first block, where r^p is the order of the predictor) was obtained by evaluating the known analytic solution at these points.

The fundamental performance criterion used in the evaluation was solution speed. The measure used for this purpose was taken to be the number of derivative function evaluations per processor. This approach ignores the interprocessor communication time that must be spent in the solution procedure, but the approach does tend to become increasingly more realistic when the complexity of the derivative functions increases and/or the size of the model (the number of ode's) increases.

For the first group of problems, the experiments were organized to examine performance at three different solution accuracies; i.e., $G=10^{-3}$, 10^{-5} and 10^{-7}. Because the testing was restricted

to problems with a known solution, it was feasible to explicitly measure global error over the solution interval. The test results were obtained using an iterative procedure which was designed to provide solutions whose global error was within a prescribed tolerance of the desired value of G. The iteration was based on the adjustment of the stepsize, H, and terminated only when a solution with global error in the interval [0.5G, G] was obtained. Test results were obtained for four integration orders; namely, r=3,5,7,9.

For the second group problems, the experiments were organized to examine the sensitivity of these methods to the stepsize, H. The results of these experiments are provided for three integration orders; namely, r=5, 7 and 9.

4.2 Test Results for the First Group of Problems

The results of the experiments with the PBPC/2 and PBPC/3 methods are presented in Tables 4.1 and 4.2 respectively. For comparison, Tables 4.3 through 4.5 give the results of the same test problems for the PBPC/1, NWP/BPC and NWP/BPC/3 methods, respectively. The table entries show the number of function evaluations per processor (dfe/p).

Test results for the methods in the PBPC family are provided only for the cases where $r \geq s$ (recall s=N/2). This is because relevant solution information in each block is not used when $r<s$ and this significantly deteriorates the performance of these methods.

In Tables 4.4 and 4.5, the values of N which we used range from 4 to 2r (except for the r=3 case). This choice was used primarily to permit a comparison with the data in Tables 4.1 through 4.3. It, however, does incorporate results for cases where the normal constraint $r \geq s$ is violated and hence poor performance can be anticipated because relevant solution information is not used. The r=3 case is special for two reasons. First, the N=2r=6 alternative is not included because the performance is very poor and second, an N=2 case is included because it yields superior performance.

In order to provide additional insight into the relative performance of the proposed methods (i.e., PBPC2 and PBPC/3) and the reference methods (i.e., PBPC/1, NWP/BPC and NWP/BPC/3), most of the data in Tables 4.1 through 4.5 is presented in a graphical format in Figs. 4.1 through

4.9. Each of these figures is associated with a particular value of the global solution accuracy, and in each figure the corresponding test data for values of r=5,r=7 and r=9 is shown for all five methods. The total number of dfe/p over the set of test problems is shown as a function N with N in range 4 through 2r.

The test results for the r=3 case are not included in Figs. 4.1 through 4.9 because the performance of all the methods for this case is unacceptable from a practical point of view. This unacceptable behaviour is apparent from Tables 4.1 to 4.5 where it can be noted that when r=3 the number of dfe/p is, in almost all cases, disproportionately large. Furthermore, for each of the problems, and each value of G, the number of dfe/p generally increases as N increases, implying a decrease (rather than increase) in solution speed with added processing power.

In the performance curves of Figs. 4.1 through 4.9, it can be noted that:

i) The solution speed (as measured in terms of the number of dfe/p) of the PBPC/2 and PBPC/3 methods has no observable improvement from the PBPC/1 method. This implies that the increase in the stability bound which occurs with more corrector iterations does not guarantee improved efficiency; indeed a deterioration in efficiency may occur.

ii) In most cases, the solution speed of the PBPC/2 method is better than that of the NWP/BPC method, and if N>6, the solution speed of the PBPC/3 method is better than that of the NWP/BPC/3 method. This indicates that performance can be improved by applying more corrector iterations and distributing these corrector evaluations over multiple blocks in parallel.

It should be noted that for any particular integration order, the solution speed does not consistently decrease when the number of processors, N, increases. This is an undesirable feature for all the methods mentioned in this thesis. The best case for any particular integration order generally occurs at the condition s=r or s=r-1.

4.3 Test Results for the Second Group of Problems

In this section, we give some insight into the sensitivity of solution accuracy to stepsize as the stiffness of a given problem increases. These experiments are based on the test problems

called Krogh1, Krogh2 and Krogh3 in Appendix 2. Each of these problems is an instance of a problem suggested by Krogh. The stiffness of the problem increases with the value assigned to the parameter β_3.

The curves in Figs. 4.10 through 4.18 show some of the results of these experiments. Each of these figures is associated with a particular integration order, r, for a particular test problem. The vertical co-ordinate in each figure is -Log10(G) where G is the global solution error that was obtained. The horizontal co-ordinate is the stepsize, H. In each of the experiments, the relationship between s and r that was used was s=r-1 (recall that for methods of the BPPC family N=s while for methods of the PBPC family N=2s).

Tables 4.6 through 4.8 present associated data relating to the curves given in Figs. 4.10 through 4.18; e.g. solution speed information (dfe/p).

From the curves in Figs. 4.10 through 4.18, we can note that:

i) In the case of the Krogh2 and Krogh3 problems (low stiffness and medium stiffness), for any particular order and any particular method, the solution accuracy has a dramatic drop when a particular value of H is reached. This value of H depends on the level of stiffness of the test problem. Note also that for the Krogh1 problem (non-stiff), the curves of solution accuracy are significantly smoother than for the Krogh2 and Krogh3 cases. This demonstrates that as the stiffness of the problem increases the solution accuracy is more sensitive to stepsize.

ii) In each figure, the magnitude of the slope of the solution accuracy curves decreases when M, the number of dfe/p per step, increases. This occurs for the methods in both the PBPC family and BPC family. This suggests that increasing the number of corrector iterations can decrease the sensitivity of the solution accuracy to the stepsize.

From the data in Tables 4.6 through 4.8, we can note that when the stiffness of the problem increases, there is no guarantee that the increase of stability bound arising from an increased number of corrector iteration will improve the solution speed (dfe/p). This applies for any given accuracy level.

It also should be noted that all the methods mentioned in this thesis are not suitable for

solving very stiff problems because of their relatively small stability bound; in particular the methods are not A-stable.

		TP1			TP2			TP3			TP4			TP5		
		10^{-3}	10^{-5}	10^{-7}	10^{-3}	10^{-5}	10^{-7}	10^{-3}	10^{-5}	10^{-7}	10^{-3}	10^{-5}	10^{-7}	10^{-3}	10^{-5}	10^{-7}
r=3	N=4	97	275	851	247	767	2235	309	801	2503	605	1203	2607	203	585	1521
	N=6	155	727	3407	341	1389	6259	1059	4837	9897	945	4003	9271	305	1265	5797
r=5	N=4	75	165	433	97	171	527	123	451	1019	245	549	1329	119	227	673
	N=6	57	123	189	85	151	339	139	287	687	203	407	887	109	207	451
	N=8	53	121	229	85	161	321	141	239	443	189	379	739	113	191	375
	N=10	67	125	263	85	165	407	139	197	655	207	415	903	115	207	453
r=7	N=4	73	117	237	99	177	305	107	169	361	165	303	505	133	161	313
	N=6	61	79	159	79	127	205	125	155	221	145	225	425	165	171	221
	N=8	53	87	133	77	107	169	115	143	227	135	211	355	119	175	195
	N=10	57	73	125	69	95	143	119	149	219	143	209	317	155	159	191
	N=12	49	71	105	65	97	141	99	137	207	133	205	311	121	129	187
	N=14	47	73	111	63	97	147	99	139	207	133	195	297	121	129	177
r=9	N=4	63	105	173	83	129	197	107	127	157	125	195	331	151	157	195
	N=6	65	75	115	93	99	147	157	165	179	171	189	261	225	231	237
	N=8	69	87	119	113	121	129	187	195	205	243	261	277	279	285	291
	N=10	79	89	105	135	141	147	235	243	253	303	329	347	333	339	345
	N=12	73	81	103	121	127	131	201	205	213	275	291	307	299	303	311
	N=14	63	73	93	99	105	111	173	181	191	219	235	255	243	251	257
	N=16	53	67	93	59	85	113	91	101	135	113	147	213	121	125	141
	N=18	53	65	85	61	83	113	87	101	125	113	145	211	123	127	139

Table 4.1
Results for the PBPC/2

		TP1			TP2			TP3			TP4			TP5		
		10^{-3}	10^{-5}	10^{-7}	10^{-3}	10^{-5}	10^{-7}	10^{-3}	10^{-5}	10^{-7}	10^{-3}	10^{-5}	10^{-7}	10^{-3}	10^{-5}	10^{-7}
r=3	N=4	145	421	1297	385	1201	3763	373	928	3019	604	2795	5737	274	844	2311
	N=6	250	1099	5182	535	2068	9868	1567	7240	9989	1279	5809	*	424	1987	9235
r=5	N=4	106	247	619	184	397	754	268	613	1549	331	838	1993	154	382	994
	N=6	79	166	415	109	211	505	181	412	1033	223	559	1330	121	256	664
	N=8	85	127	256	91	196	421	136	208	307	187	286	646	130	175	307
	N=10	85	172	424	124	265	634	172	415	1207	226	529	1330	142	238	634
r=7	N=4	91	172	340	133	256	439	142	277	544	217	424	802	121	217	439
	N=6	70	112	229	91	157	268	112	187	355	148	286	535	121	139	295
	N=8	70	100	166	73	112	193	109	148	259	154	214	403	127	139	211
	N=10	67	94	142	67	91	172	106	139	220	136	214	325	130	139	190
	N=12	67	94	139	70	95	160	106	136	193	127	211	301	130	136	196
	N=14	64	91	148	70	112	187	106	136	214	130	199	319	136	142	187
r=9	N=4	91	145	256	103	175	292	109	166	283	160	280	475	127	160	274
	N=6	67	94	166	91	118	187	91	124	190	127	181	319	133	139	184
	N=8	61	94	124	70	97	133	112	124	160	121	178	241	160	166	172
	N=10	67	88	112	70	79	109	94	109	151	115	160	241	154	160	166
	N=12	64	88	115	70	82	106	109	118	145	112	148	223	157	163	166
	N=14	61	82	112	64	79	109	91	109	139	109	151	214	142	148	154
	N=16	61	82	115	64	85	106	97	112	133	109	148	187	121	127	148
	N=18	64	82	109	67	82	106	97	112	133	106	157	190	130	139	148

Fig. 4.2
Results for the PBPC/3

In these Tables, * means that the number of function evaluations per processor is greater than 10,000.

		TP1			TP2			TP3			TP4			TP5		
		10^{-3}	10^{-5}	10^{-7}	10^{-3}	10^{-5}	10^{-7}	10^{-3}	10^{-5}	10^{-7}	10^{-3}	10^{-5}	10^{-7}	10^{-3}	10^{-5}	10^{-7}
r=3	N=4	102	288	802	288	802	2669	360	912	2502	482	1052	4802	202	669	2002
	N=6	85	224	1335	268	668	2668	557	2668	*	535	2001	8002	224	668	3049
r=5	N=4	65	114	252	127	252	574	159	280	475	202	431	926	120	225	502
	N=6	58	105	252	105	225	447	154	259	479	202	366	802	119	193	413
	N=8	58	102	202	86	202	402	141	252	457	169	301	752	120	182	382
	N=10	55	101	179	81	179	401	144	237	418	173	287	687	119	177	366
r=7	N=4	57	86	169	84	136	252	138	345	252	136	233	402	191	197	233
	N=6	54	86	151	69	105	193	110	154	225	136	202	336	140	145	203
	N=8	52	69	136	69	102	156	107	150	218	129	202	302	124	128	195
	N=10	52	68	136	69	102	155	102	145	213	122	194	302	124	128	183
	N=12	52	68	129	69	105	159	104	141	211	117	193	299	124	128	182
	N=14	52	68	135	70	105	158	102	137	197	115	188	298	123	127	174
r=9	N=4	77	90	135	157	164	171	277	286	294	380	400	424	387	394	400
	N=6	73	85	111	147	153	160	252	259	268	369	392	412	365	371	377
	N=8	53	68	98	60	88	120	93	113	146	117	150	221	123	127	142
	N=10	53	67	97	59	82	119	93	110	136	115	148	206	123	128	139
	N=12	53	65	94	61	84	122	94	118	188	134	151	201	124	127	136
	N=14	55	65	94	61	80	110	94	107	250	141	168	247	124	129	142
	N=16	53	63	96	61	86	196	107	120	219	153	188	360	126	130	177
	N=18	53	64	116	63	90	237	121	216	658	159	205	562	127	135	237

Table 4.3

Results for the PBPC/1

		TP1			TP2			TP3			TP4			TP5		
		10^{-3}	10^{-5}	10^{-7}	10^{-3}	10^{-5}	10^{-7}	10^{-3}	10^{-5}	10^{-7}	10^{-3}	10^{-5}	10^{-7}	10^{-3}	10^{-5}	10^{-7}
r=3	N=2	193	150	1667	639	1829	6401	571	1599	4571	961	3071	*	401	1279	3657
	N=4	251	1001	5001	535	1281	8531	1601	6401	*	1281	5121	*	401	1061	8535
r=5	N=4	113	193	417	201	401	915	181	363	727	275	641	1397	133	319	711
	N=6	105	177	371	179	357	777	157	243	889	257	569	1281	119	267	611
	N=8	105	167	471	161	357	753	139	401	1143	241	549	1397	125	267	641
	N=10	101	167	445	161	367	733	161	459	1281	257	615	1537	129	285	733
r=7	N=4	101	177	249	147	199	355	273	287	309	273	307	511	355	365	375
	N=6	71	129	185	97	177	305	111	167	267	143	233	465	121	133	267
	N=8	69	125	179	89	161	291	105	155	251	137	213	427	121	133	201
	N=10	69	119	175	81	143	271	107	147	229	129	221	385	121	135	197
	N=12	71	121	177	77	143	267	103	149	335	129	215	395	123	135	215
	N=14	69	117	169	67	133	245	103	137	2297	123	129	217	123	129	217
r=9	N=4	259	279	363	535	545	555	929	943	968	1483	1529	1561	1311	1321	1331
	N=6	243	259	317	499	509	523	871	883	905	1377	1419	1477	1217	1227	1235
	N=8	73	103	139	65	95	153	93	115	155	119	159	239	121	127	147
	N=10	71	83	137	63	85	135	93	115	179	133	157	237	123	127	145
	N=12	71	85	141	61	79	133	95	145	205	139	163	627	123	127	151
	N=14	67	81	167	61	83	759	125	423	7015	157	213	3217	125	137	505
	N=16	69	251	*	69	137	4421	149	1245	*	183	1525	*	127	165	4307
	N=18	77	2983	*	99	847	*	277	1873	*	213	7279	*	131	611	*

Table 4.4

Results for the NWP/BPC

		TP1			TP2			TP3			TP4			TP5		
		10^{-3}	10^{-5}	10^{-7}	10^{-3}	10^{-5}	10^{-7}	10^{-3}	10^{-5}	10^{-7}	10^{-3}	10^{-5}	10^{-7}	10^{-3}	10^{-5}	10^{-7}
r=3	N=2	199	514	1405	397	1165	3598	457	1291	3643	721	1858	5923	346	793	2392
	N=4	268	1525	7078	1023	3001	*	2212	9988	*	1666	7741	*	538	2500	*
r=5	N=4	115	205	370	172	313	571	193	373	715	313	592	109	169	340	658
	N=6	106	199	487	190	529	1351	199	547	1345	313	697	1714	154	349	760
	N=8	97	256	619	250	673	1714	253	625	1708	358	799	1960	175	400	964
	N=10	106	280	679	271	736	1882	280	688	1876	394	877	2152	193	439	1060
r=7	N=4	97	136	208	121	205	331	139	205	325	181	343	589	205	214	355
	N=6	85	118	196	109	178	289	112	169	283	136	313	511	121	181	313
	N=8	79	109	187	109	154	232	109	157	226	139	280	487	124	163	292
	N=10	79	115	205	103	142	313	103	142	307	145	277	535	124	145	289
	N=12	73	130	235	97	181	358	97	178	349	166	316	550	124	166	313
	N=14	76	139	250	94	190	403	94	190	316	175	334	583	124	175	334
r=9	N=4	106	154	190	172	217	185	274	286	298	379	403	442	433	439	451
	N=6	97	136	169	154	166	187	250	263	274	349	376	409	391	400	409
	N=8	76	103	148	67	115	169	91	115	166	106	175	283	124	127	184
	N=10	76	100	136	91	115	157	91	115	160	115	166	265	124	127	166
	N=12	70	100	136	91	115	148	91	112	181	127	154	463	124	130	154
	N=14	70	97	187	103	121	1033	97	166	1281	145	184	1138	124	130	421
	N=16	70	142	640	112	541	*	115	337	*	157	775	*	127	172	1537
	N=18	76	238	2024	169	4630	*	175	3739	*	277	3010	*	127	391	8185

Table 4.5

Results for the NWP/BPC/3

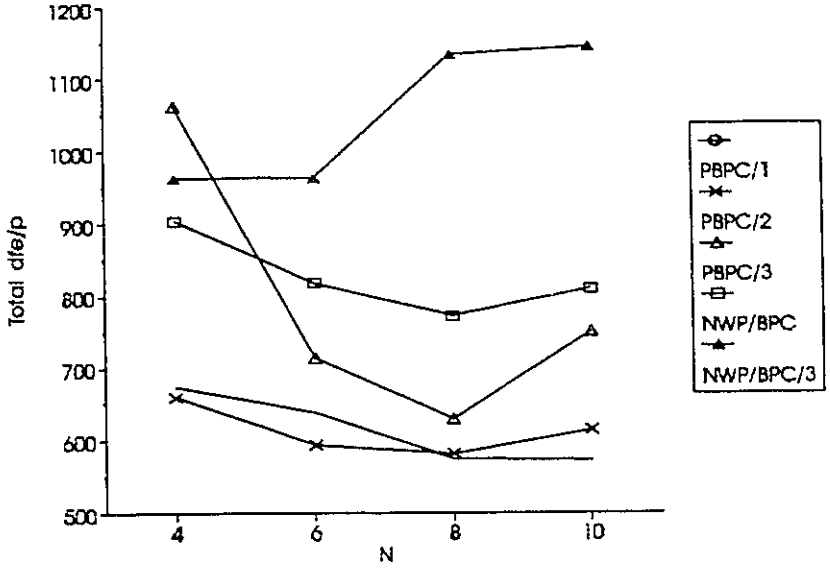

Fig. 4.1
Total dfe/p for r=5 and $G=10^{-3}$

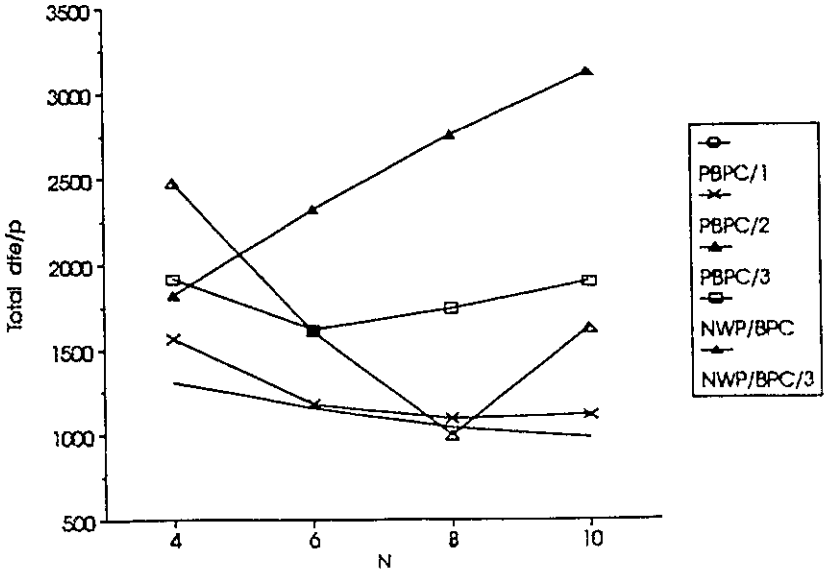

Fig. 4.2
Total dfe/p for r=5 and $G=10^{-5}$

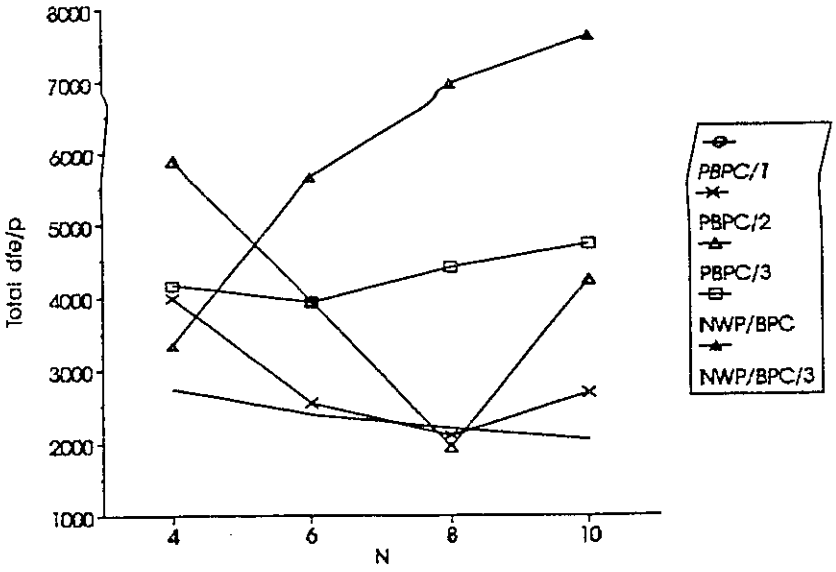

Fig. 4.3
Total dfe/p for r=5 and $G=10^{-7}$

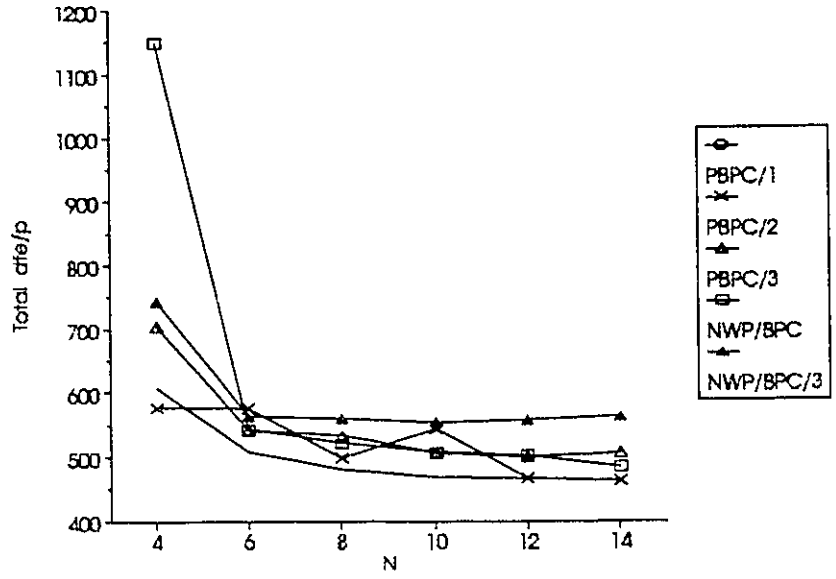

Fig. 4.4
Total dfe/p for r=7 and $G=10^{-3}$

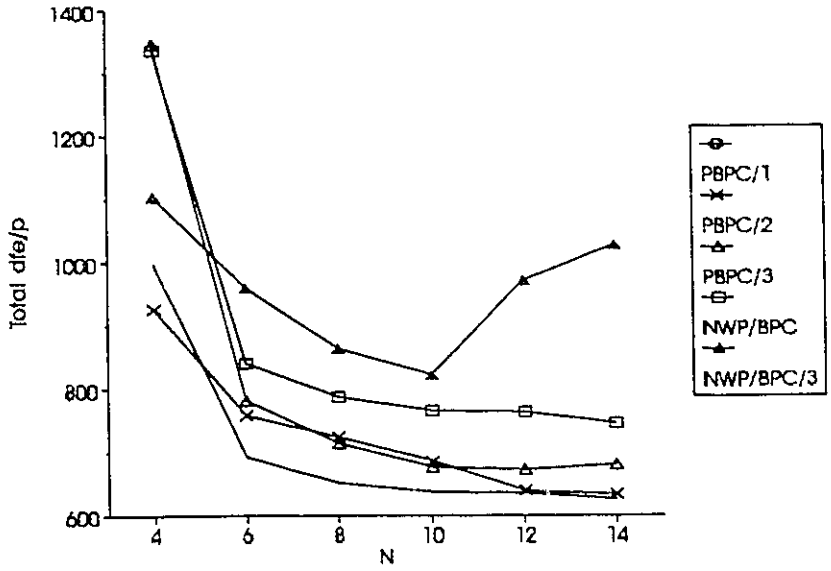

Fig. 4.5
Total dfe/p for r=7 and $G=10^{-5}$

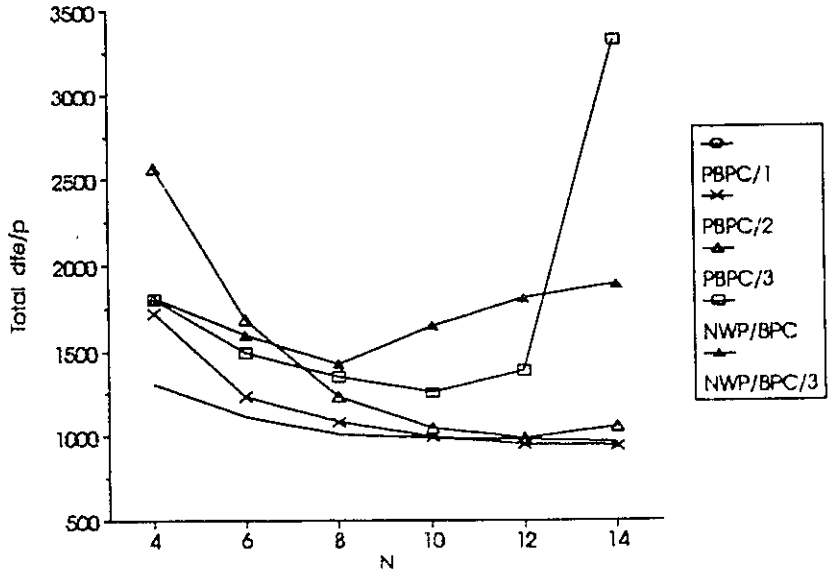

Fig. 4.6
Total dfe/p for r=7 and $G=10^{-7}$

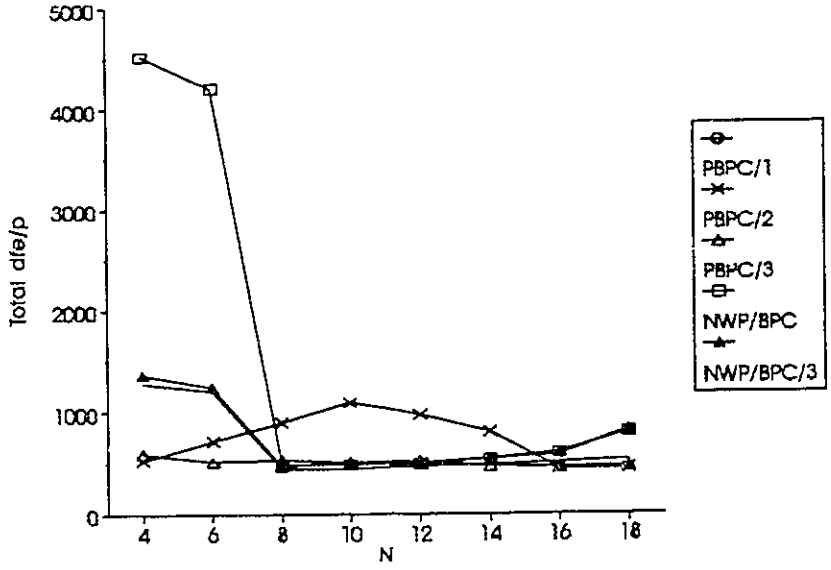

Fig. 4.7
Total dfe/p for r=9 and $G=10^{-3}$

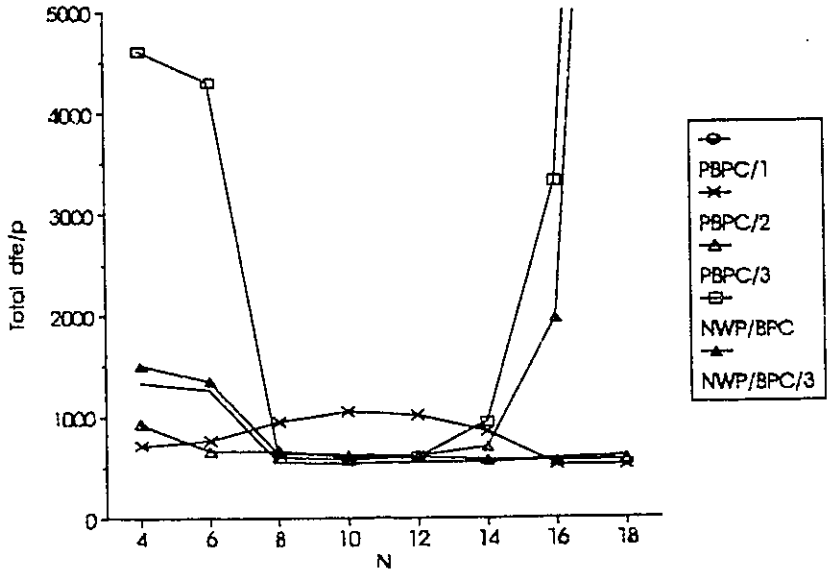

Fig. 4.8
Total dfe/p for r=9 and $G=10^{-5}$

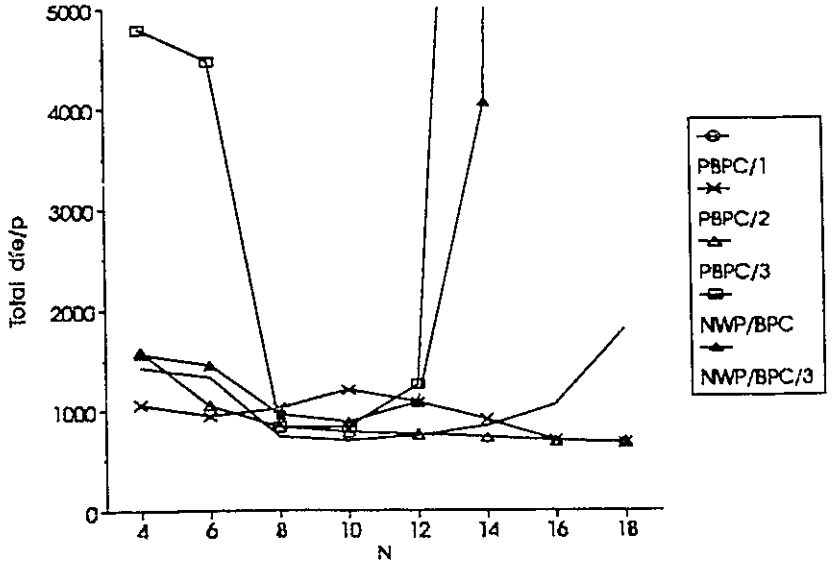

Fig. 4.9
Total dfe/p for r=9 and $G=10^{-7}$

	r=5			r=7			r=9		
	-Log10(G)	H	dfe/p	-Log10(G)	H	dfe/p	-log10(G)	H	dfe/p
PBPC/1	3.14	0.01	102	3.8	0.01	102	4.3	0.01	102
	1.47	0.015	68	1.7	0.015	68	1.7	0.015	68
	0.26	0.02	52	< 0	0.02	52	< 0	0.02	52
	< 0	0.025	42						
PBPC/2	6.3	0.01	203	8.21	0.01	203	9.59	0.01	203
	5.05	0.015	135	6.12	0.015	135	6.94	0.015	135
	3.89	0.02	103	4.78	0.02	103	4.03	0.02	103
	2.98	0.025	83	3.34	0.025	83	1.75	0.025	83
	2.19	0.03	69	2.39	0.03	69	0.23	0.03	69
	1.59	0.035	59	< 0	0.035	59	< 0	0.035	59
PBPC/3	6.16	0.01	304	8.97	0.01	304	11.7	0.01	304
	5.13	0.015	202	7.60	0.015	202	9.09	0.015	202
	4.44	0.02	154	6.33	0.02	154	7.25	0.02	154
	3.83	0.025	124	5.07	0.025	124	5.72	0.025	124
	3.41	0.03	103	3.97	0.03	103	4.52	0.03	103
	2.58	0.035	88	3.11	0.035	88	3.56	0.035	88
	2.06	0.04	79	2.57	0.04	79	2.92	0.04	79
	1.57	0.045	70	1.98	0.045	70	2.31	0.045	70
	1.22	0.05	64	1.35	0.05	64	< 0	0.05	64
NWP/BPC	3.90	0.01	201	5.53	0.01	201	7.07	0.01	201
	2.86	0.015	133	4.04	0.015	133	4.88	0.015	133
	2.11	0.02	101	2.77	0.02	101	3.08	0.02	101
	1.31	0.025	81	1.59	0.025	81	1.61	0.025	81
	0.79	0.03	67	0.64	0.03	67	0.66	0.03	67
	0.00	0.035	57	< 0	0.035	57	< 0	0.035	57
NWP/BPC/3	5.19	0.01	301	6.85	0.01	301	8.45	0.01	301
	3.97	0.015	199	5.26	0.015	199	6.52	0.015	199
	3.16	0.02	151	4.18	0.02	151	5.09	0.02	151
	2.40	0.025	121	3.19	0.025	121	3.83	0.025	121
	1.95	0.03	100	2.47	0.03	100	2.72	0.03	100
	1.40	0.035	85	1.63	0.035	85	1.70	0.035	85
	0.88	0.04	76	0.97	0.04	76	1.04	0.04	76
	0.61	0.045	67	0.33	0.045	67	0.39	0.045	67
	0.07	0.05	61	0.00	0.05	61	< 0	0.05	61

Table 4.6
Results for Krogh1

	r=5			r=7			r=9		
	-Log10(G)	H	dfe/p	-Log10(G)	H	dfe/p	-log10(G)	H	dfe/p
PBPC/1	7.65	0.002	502	9.58	0.002	502	8.38	0.002	502
	4.97	0.004	252	4.65	0.004	252	2.24	0.004	252
	< 0	0.006	202	< 0	0.006	202	< 0	0.006	202
PBPC/2	9.29	0.002	1003	10.3	0.002	1003	10.3	0.002	1003
	7.20	0.004	503	9.16	0.004	503	9.06	0.004	503
	5.86	0.006	335	7.27	0.006	335	7.27	0.006	335
	4.78	0.008	253	5.73	0.008	253	5.73	0.008	253
	0.64	0.01	203	0.92	0.01	203	0.92	0.01	203
	0.21	0.012	169	0.40	0.012	169	0.40	0.012	169
	< 0	0.014	145	< 0	0.014	145	< 0	0.014	145
PBPC/3	9.72	0.002	1504	10.3	0.002	1504	10.3	0.002	1504
	8.12	0.004	754	10.1	0.004	754	10.4	0.004	754
	6.70	0.006	502	8.51	0.006	502	10.0	0.006	502
	5.62	0.008	379	7.19	0.008	379	8.87	0.008	379
	4.78	0.01	304	6.19	0.01	304	7.70	0.01	304
	4.05	0.012	253	4.62	0.012	253	3.56	0.012	253
	< 0	0.014	217	< 0	0.014	217	< 0	0.014	217
NWP/ BPC	7.31	0.002	1001	9.85	0.002	1001	11.4	0.002	1001
	5.39	0.004	501	7.21	0.004	501	8.80	0.004	501
	4.17	0.006	333	5.47	0.006	333	6.55	0.006	333
	3.17	0.008	251	3.86	0.008	251	1.85	0.008	251
	< 0	0.01	201	< 0	0.01	201	< 0	0.01	201
NWP/ BPC/3	8.83	0.002	1501	11.4	0.002	1501	12.3	0.002	1501
	6.71	0.004	751	8.79	0.004	751	10.8	0.004	751
	5.46	0.006	499	7.16	0.006	499	8.81	0.006	499
	4.54	0.008	367	6.01	0.008	367	7.42	0.008	367
	3.83	0.01	301	5.13	0.01	301	6.05	0.01	301
	3.02	0.012	250	4.17	0.012	250	2.10	0.012	250
	< 0	0.014	214	< 0	0.014	214	< 0	0.014	214

Table 4.7
Results for Krogh2

	r=5			r=7			r=9		
	-Log10(G)	H	dfe/p	-Log10(G)	H	dfe/p	-log10(G)	H	dfe/p
PBPC/1	7.81	0.0002	5002	9.96	0.0002	5002	8.90	0.0002	5002
	5.15	0.0004	2502	4.49	0.0004	2502	0.59	0.0004	2502
	< 0	0.0006	2002	< 0	0.0006	2002	< 0	0.0006	2002
PBPC/2	9.34	0.0002	10003	10.2	0.0002	10003	10.3	0.0002	10003
	7.27	0.0004	5003	9.38	0.0004	5003	9.06	0.0004	5003
	5.91	0.0006	3305	7.50	0.0006	3305	7.27	0.0006	3335
	4.83	0.0008	2503	5.90	0.0008	2503	5.73	0.0008	2503
	< 0	0.001	2003	< 0	0.001	2003	< 0	0.001	2003
PBPC/3	9.88	0.0002	15004	10.2	0.0002	15004	10.2	0.0002	15004
	8.19	0.0004	7504	10.2	0.0004	7504	10.3	0.0004	7504
	6.77	0.0006	5002	8.75	0.0006	5002	10.3	0.0006	5002
	5.68	0.0008	3754	7.40	0.0008	3754	9.40	0.0008	3754
	4.82	0.001	3004	6.38	0.001	3004	8.21	0.001	3004
	4.09	0.0012	2503	< 0	0.0012	2503	< 0	0.0012	2503
	< 0	0.0014	2146						
NWP/ BPC	9.30	0.0002	10001	11.5	0.0002	10001	11.4	0.0002	10001
	6.34	0.0004	5001	8.75	0.0004	5001	11.0	0.0004	5001
	4.85	0.0006	3301	6.65	0.0006	3301	8.38	0.0006	3301
	3.77	0.0008	2501	5.02	0.0008	2501	6.17	0.0008	2501
	< 0	0.001	2001	< 0	0.001	2001	< 0	0.001	2001
NWP/ BPC/3	9.95	0.0002	15004	11.8	0.0002	15004	11.8	0.0002	15004
	8.29	0.0004	7504	11.6	0.0004	7504	12.0	0.0004	7504
	7.39	0.0006	5002	9.63	0.0006	5002	11.7	0.0006	5002
	6.37	0.0008	3754	8.15	0.0008	3754	10.2	0.0008	3754
	5.38	0.001	3004	7.05	0.001	3004	8.58	0.001	3004
	4.58	0.0012	2503	1.31	0.0012	2503	< 0	0.0012	2503
	< 0	0.0014	2146	< 0	0.0014	2146			

Table 4.7
Results for Krogh3

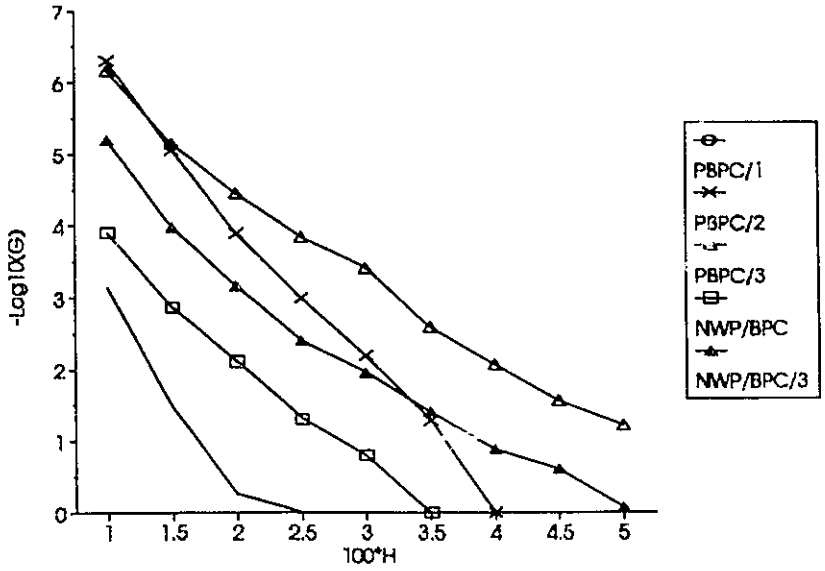

Fig. 4.10
Results for Krogh1 in the Case r=5

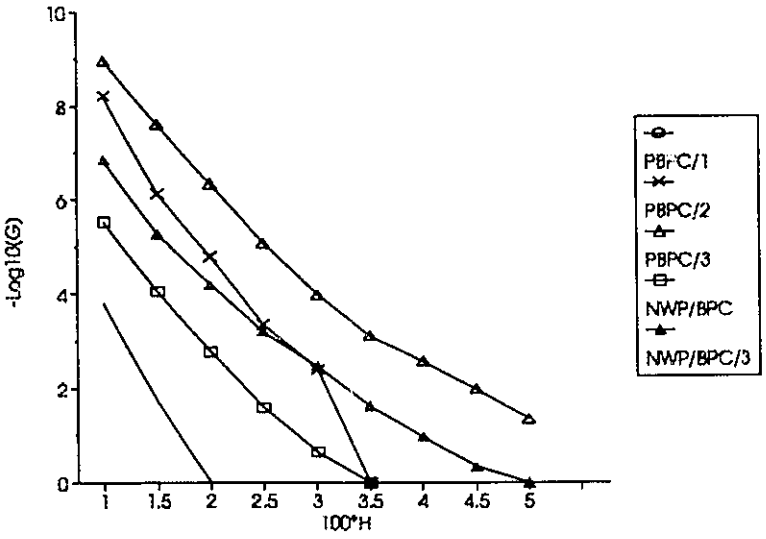

Fig. 4.11
Results for Krogh1 in the Case r=7

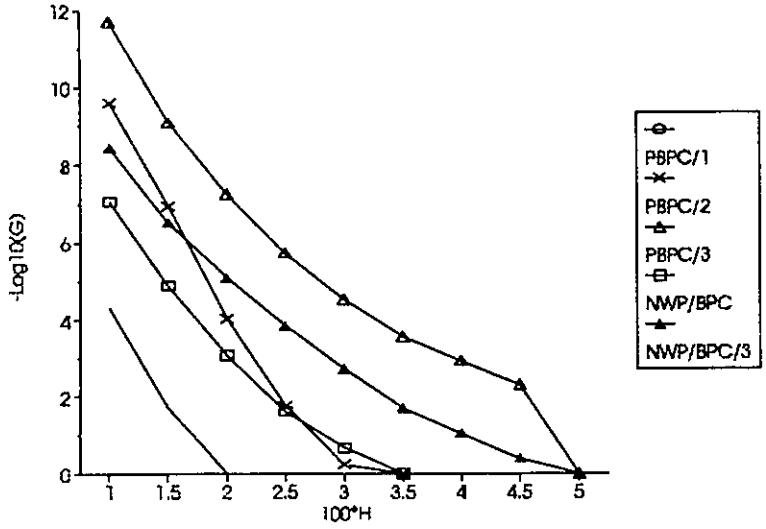

Fig. 4.12
Results for Krogh1 in the Case r=9

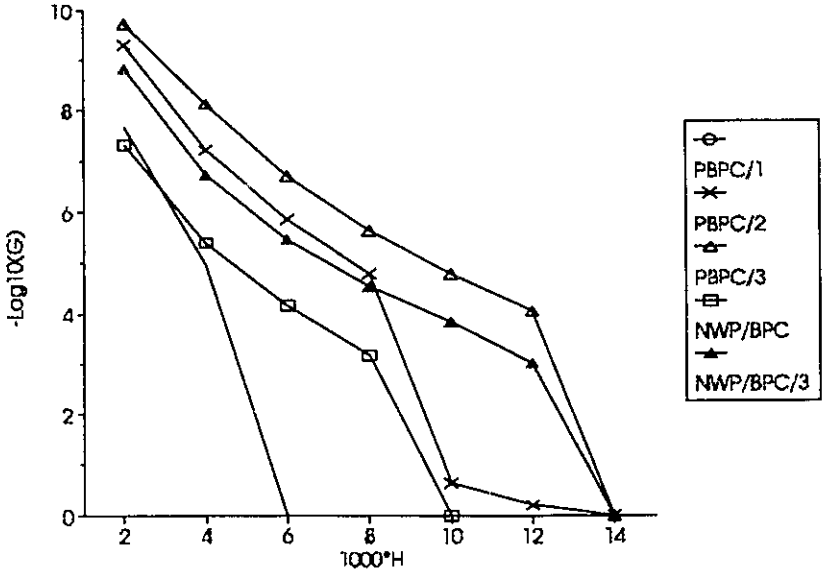

Fig. 4.13
Results for Krogh2 in the Case r=5

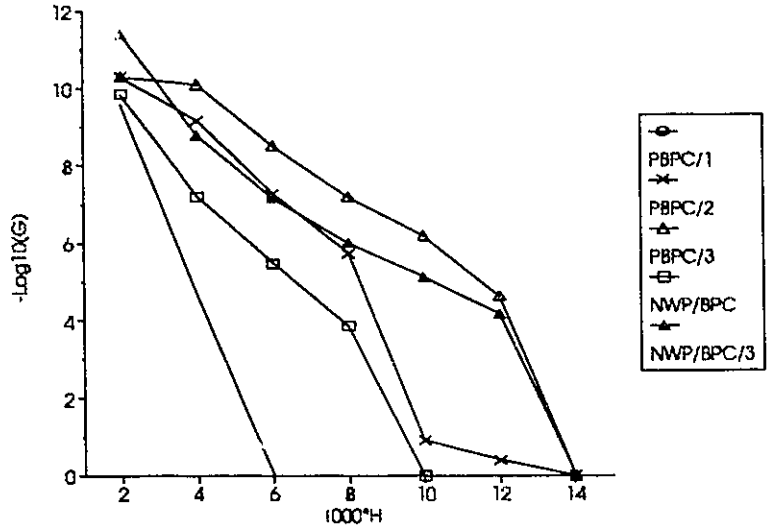

Fig. 4.14
Results for Krogh2 in the Case r=7

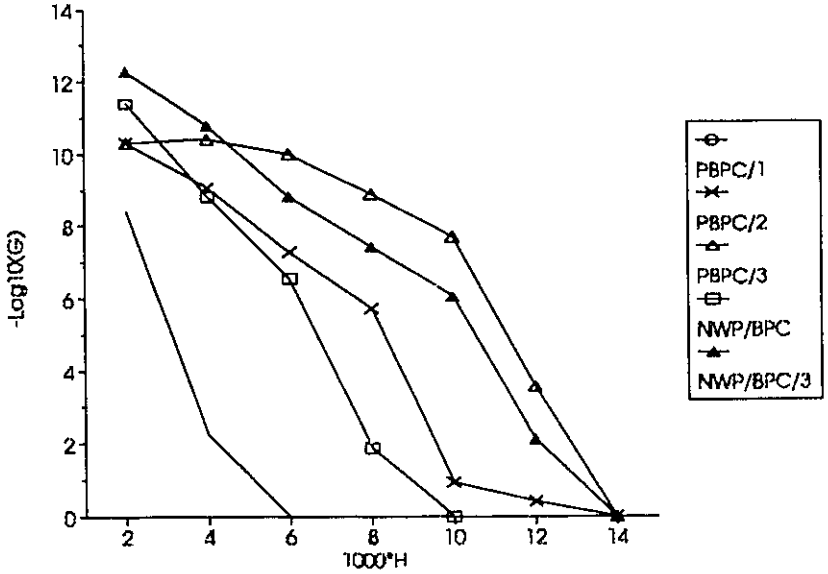

Fig. 4.15
Results for Krogh2 in the Case r=9

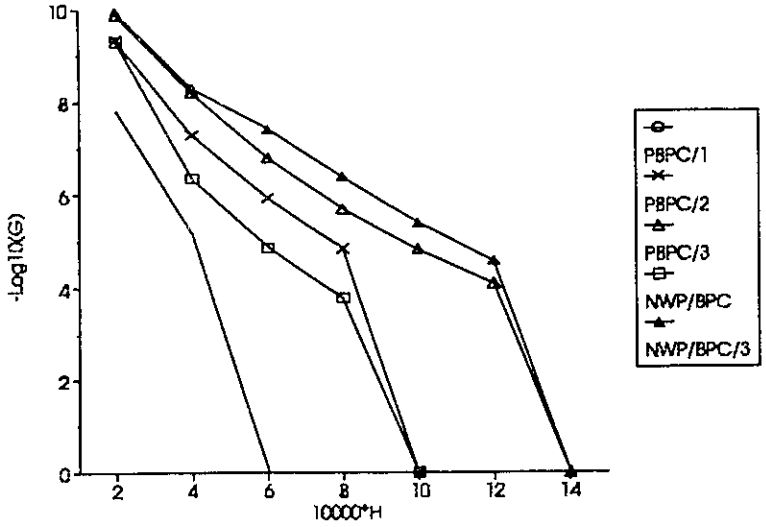

Fig. 4.16
Results for Krogh3 in the Case r=5

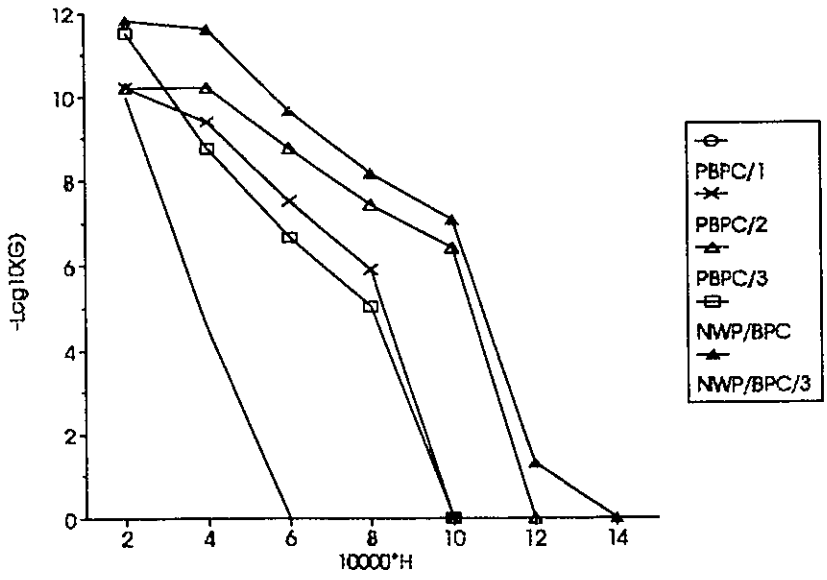

Fig. 4.17
Results for Krogh3 in the Case r=7

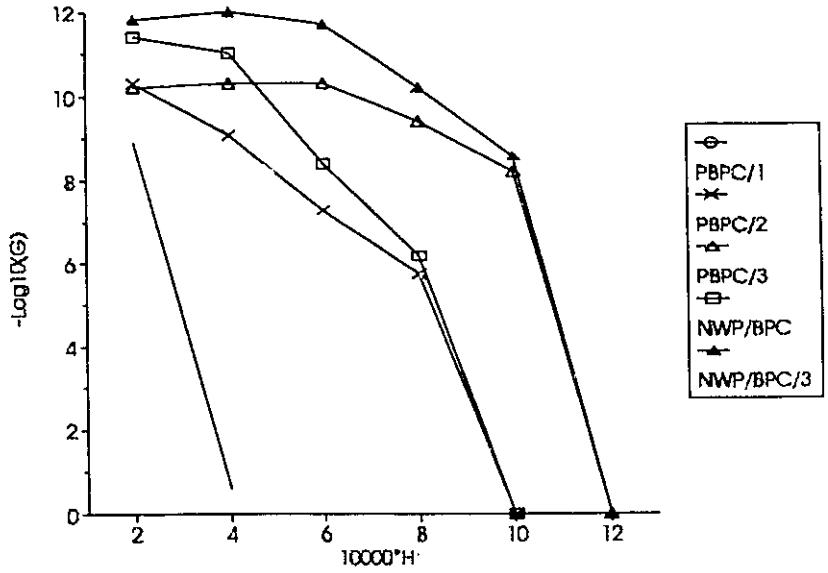

Fig. 4.18
Results for Krogh3 in the Case r=9

5. Conclusions and Future Work

5.1 General Observations

The two new methods proposed in this thesis; namely, the PBPC/2 and PBPC/3 methods, have an increased stability interval relative to the PBPC/1 method (see Tables 3.2, 3.3 and 3.4). But, in the numerical experiments, they do not achieve observable improvement in performance (see Tables 4.1, 4.2 and 4.3). This is because of the increase in the number of function evaluations per processor, per step, which accompanies the increase stability interval. However, the improvement in stability behaviour achieved by increasing the number of corrector iterations in the PBPC/2 and PBPC/3 methods does reduce the sensitivity to the step size; i.e., H. This feature, in practice, improves the confidence with which these methods can be used.

The PBPC/2 and PBPC/3 methods are based on the Adams multi-step model. Methods in the BPC family that are based on this same model can also be formulated; e.g. the NWP/BPC method. These methods generally show inferior performance in test results to the PBPC/2 and PBPC/3 (for cases of comparable order, number of processors and number of function evaluations per processor, per step (i.e., M). This is because the corrector iterations in the PBPC/2 and PBPC/3 cases have a greater influence in reducing the effect of the predictor's contribution to the local truncation error [20].

Like the previously suggested PBPC/1 method, the PBPC/2 and PBPC/3 methods encounter a delay in information usage that occurs because final solution values in block n cannot be used for computing predicted values and intermediate corrected values in block (n+1). This constrains the ability of the additional corrector iterations to extend the stability bounds and they are, in fact, not much larger than for equivalent methods of the BPC family (for same value of M). (see Fig. 3.2, 3.3, 3.4 and 3.5).

Our study of the PBPC/M methods shows that these methods provide a potential opportunity in parallel computer systems to balance stability and efficiency by applying more corrector iterations simultaneously in multiple blocks. These two fundamental properties are often traded-off in generating methods for the numerical solution of ode's.

5.2 Future Work

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

a) Instead of the Adams model, a more sophisticated approach which uses more solution information could be used for the formulation of the PBPC/M methods. This formulation might enlarge the stability region and/or improve efficiency.

b) Further investigation should be undertaken to determine how the order of the predictor affects the stability properties and the efficiency of block-based predictor-corrector methods. This investigation could be worthwhile in the context of both stability and efficiency in generating effective numerical ode methods.

c) Investigation of the strategies for implementing variable stepsize procedures for PBPC/M method should be undertaken to achieve automatic stepsize control.

d) 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 the performance of these solution methods.

Appendix 1

We provide in this Appendix, the predictor and corrector (both groups) formulas for the PBPC/M methods for the cases s=2, r=5 and s=4, r=3.

i) s=2, r=5

Predictor:

$$y_{n+2}^{0}=y_{n-2}+\frac{1}{6}H(28f_n^{M-1}-40f_{n-1}^{M-1}+32f_{n-2}-8f_{n-3})$$
$$y_{n+1}^{0}=y_{n-2}+\frac{1}{16}H(21f_n^{M-1}-9f_{n-1}^{M-1}+15f_{n-2}+3f_{N-3})$$

Corrector (first group):

$$y_{n+2}^{m}=y_n^{M+m-1}+\frac{1}{180}H(29f_{n+2}^{m-1}+124f_{n+1}^{m-1}+24f_n^{M+m-1}+4f_{n-1}^{M+m-1}-f_{n-2})$$
$$y_{n+1}^{m}=y_n^{M+m-1}+\frac{1}{1440}H(-19f_{n+2}^{m-1}+346f_{n+1}^{m-1}+456f_n^{M+m-1}-74f_{n-1}^{M+m-1}+11f_{n-2})$$

where m=1,2,..M-1

Corrector (second group):

$$y_n^{M+m-1}=y_{n-2}\frac{1}{180}H(29f_n^{M+m-1}+124f_{n-1}^{M+m-1}+24f_{n-2}+4f_{n-3}-f_{n-4})$$
$$y_{n-1}^{M+m-1}=y_{n-2}+\frac{1}{1440}H(-19f_n^{M+m-1}+346f_{n-1}^{M+m-1}+456f_{n-2}-74f_{n-3}+11f_{n-4})$$

where m=1,2,..,M.

ii) s=4, r=3

Predictor:

$$y_{n+4}^{0}=y_{n-4}+\frac{1}{4}H8f_{n}^{M-1}$$
$$y_{n+3}^{0}=y_{n-4}+\frac{1}{8}H(7f_{n}^{M-1}+7f_{n-1}^{M-1})$$
$$y_{n+2}^{0}=y_{n-4}+\frac{1}{4}H6f_{n-1}^{M-1}$$
$$y_{n+1}^{0}=y_{n-4}+\frac{1}{8}H(-5f_{n}^{M-1}+15f_{n-1}^{M-1})$$

Corrector (first group):

$$y_{n+4}^{m}=y_{n}^{M+m-1}+\frac{1}{12}H(8f_{n+4}^{m-1}-16f_{n+3}^{m-1}+20f_{n+2}^{m-1})$$
$$y_{n+3}^{m}=y_{n}^{M+m-1}+\frac{1}{16}H(9f_{n+4}^{m-1}-24f_{n+3}^{m-1}+27f_{n+2}^{m-1})$$
$$y_{n+2}^{m}=y_{n}^{M+m-1}+\frac{1}{12}H(7f_{n+4}^{m-1}-20f_{n+3}^{m-1}+19f_{n+2}^{m-1})$$
$$y_{n+1}^{m}=y_{n}^{M+m-1}+\frac{1}{48}H(23f_{n+4}^{m-1}-64f_{n+3}^{m-1}+53f_{n+2}^{m-1})$$

where m=1,2,..,M-1

Corrector (second group):

$$y_{n+4}^{M+m-1} = y_{n-4} + \frac{1}{12} H(8f_n^{M+m-1} - 16f_{n+3}^{M+m-1} + 20f_{n+2}^{M+m-1})$$

$$y_{n+3}^{M+m-1} = y_{n-4} + \frac{1}{16} H(9f_{n+4}^{M+m-1} - 24f_{n+3}^{M+m-1} + 27f_{n+2}^{M+m-1})$$

$$y_{n+2}^{M+m-1} = y_{n-4} + \frac{1}{12} H(7f_{n+4}^{M+m-1} - 20f_{n+3}^{M+m-1} + 19f_{n+2}^{M+m-1})$$

$$y_{n+1}^{M+m-1} = y_{n-4} + \frac{1}{48} H(23f_{n+4}^{M+m-1} - 64f_{n+3}^{M+m-1} + 53f_{n+2}^{M+m-1})$$

where m=1,2,..,M

Appendix 2

In this Appendix we provide a summary of two groups of test problems used in the experimental evaluations carried out in this thesis.

First Group of Test Problems:

TP1:

$dy/dt=y\cos(t);$ $\quad y(0)=1,\ t_f = 20$

Analytic solution:

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

TP2:

$dy_1/dt=-y_2-y_1y_3/r;$ $\quad y_1(0) = 3$
$dy_2/dt=y_1-y_2y_3/r;$ $\quad y_2(0) = 0$
$dy_3/dt=y_1/r;$ $\quad y_3(0) = 0$

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

Analytic solution:

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

TP3:

$dy_1/dt=y_2;$ $\quad y_1(0) = 1$
$dy_2/dt=-y_1/r^3;$ $\quad y_2(0) = 0$
$dy_3/dt=y_4;$ $\quad y_3(0) = 0$
$dy_4/dt=-y_3/r^3;$ $\quad y_4(0) = 1$

with $r=(y_1^2+y_2^2)^{1/2}$; $t_f=25$

Analytic solution:

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

TP4:

$dy_1/dt=y_1/[2(1+t)]-2ty_2$; $y_1(0) = 1$
$dy_2/dt=y_2/[2(1+t)]-2ty_1$; $y_2(0) = 0$

with $t_f=6$

Analytic solution:

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

TP5:

$dy_1/dt=y_2$; $y_1(0) = 0$
$dy_2/dt=-2y_2-101y_1$; $y_2(0) = 1$
$dy_3/dt=y_4$; $y_3(0) = 0$
$dy_4/dt=y_1-4y_4-29y_3$; $y_4(0) = 0$

with $t_f=5$

Analytic solution:

$y_1=0.1e^{-t}\sin(10t)$
$y_2=(1.01)^{1/2}e^{-t}\cos(10t+\theta_1)$
$y_3=(5e^{-t}\sin(10t-\phi)+10e^{-2t}\sin(5t+\phi))/Q$
$y_4=(5e^{-t}(101)^{1/2}\cos(10t-(\phi-\theta_1))+10$

$e^{-2t}(29)^{1/2}\cos(5t-(\Theta-\theta_2)))/Q$

with

$Q=50(6478)^{1/2}$
$\phi=\arctan(-20/74)$
$\Theta=\arctan(-10/76)$
$\theta_1=\arctan(0.1)$
$\theta_2=\arctan(0.4)$

Second Group of Test Problems:

Krogh1:

$dy/dt=-By(t) + g(y); \quad y(0)=[0,-2,-1,-1]^T$
$B=UDU,$

where

$$D=\begin{bmatrix} \beta_1 & \beta_2 & 0 & 0 \\ \beta_2 & \beta_1 & 0 & 0 \\ 0 & 0 & \beta_3 & 0 \\ 0 & 0 & 0 & \beta_4 \end{bmatrix}, \quad U=\frac{1}{2}\begin{bmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{bmatrix}$$

$$g(y)=U\begin{bmatrix} \frac{1}{2}(z_1^2(t)-z_2^2(t)) \\ z_1(t)z_2(t) \\ z_3^2(t) \\ z_4^2(t) \end{bmatrix} \quad z(t)=Uy(t)$$

with $t_f = 1$

Analytic Solution:

$y(t) = U\ v(t), \qquad v(t) = [v_1(t), v_2(t), v_3(t), v(_4(t)]^T$

$v_1(t) = \alpha(t)\ (\beta_1\ w_1(t) - \beta_2\ w_2(t))$
$v_2(t) = \alpha(t)\ (\beta_2\ w_1(t) - \beta_1\ w_2(t))$

$$v_k(t) = \frac{\beta_k}{(1 + c_k e^{\beta_k t})}, \qquad c_k = -(1 + \beta_k) \quad for\ k=3,4$$

where

$$\alpha(t) = 2(w_1^2(t) + w_2^2(t))^{-1}$$

$$w_1(t) = 1 - e^{\beta_1 t}[(1 + \beta_1)\cos(\beta_2 t) - \beta_2 \sin(\beta_2 t)]$$
$$w_2(t) = e^{\beta_1 t}[\beta_2 \cos(\beta_2 t) + (1 + \beta_1)\sin(\beta_2 t)]$$

with $\beta_1 = -10$, $\beta_2 = 10$, $\beta_3 = 10$, $\beta_4 = 0.01$

Krogh2:

As the same as Krogh1, but $\beta_3 = 100$.

Krogh3:

As the same as Krogh1, but $\beta_3 = 1000$.

Bibliography

[1] C. W. Gear, "The Potential for Parallelism in Ordinary Differential Equations", University of Illinois at Urbana-Champaign, Report No. UIUCDCS-R-86-1246, 1986

[2] C. W. Gear and D. R. Wells, "Multirate Linear Multistep Methods", BIT, Vol. 24, pp. 485-502, 1984

[3] F. R. Moulton, "New Methods in Exterior Ballistics", University of Chicago Press, Chicago, 1926

[4] C. W. Gear and R. D. Skeel, "The Development of ODE Methods: A Symbiosis between Hardware and Numerical Analysis", Manuscript, University of Illinois at Urbana-Champaign, 1989

[5] W. E. Milne, "Numerical Solution of Differential Equations", John Wiley, New York, 1953

[6] J. B. Rosser, "A Runge-Kutta for All Seasons", SIAM Review, Vol. 9, pp. 417-452, 1967

[7] L. F. Shampine and H. W. Watts, "Block Implicit One-step Methods", Math. Comp., Vol. 23, pp. 731-740, 1969

[8] H. W. Watts and L. F. Shampine, "A-Stable Block Implicit One-step Methods", BIT, Vol. 12, pp. 252-266, 1972

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

[10] L. G. Birta and O. Abou-Rabia, "Parallel Block Predictor-Corrector Methods for ODE's", IEEE Trans. on Computer, Vol. C-36, No. 3, 1987

[11] M. Chu and H. Hamilton, "Parallel Solution of ODE's by Multi-Block Methods", SIAM J. of Science and Statistics Computing, Vol. 8, No. 3, 1987

[12] H. W. Tam, "Parallel Methods for the Numerical Solution of Ordinary Differential Equations", University of Illinois at Urbana-Champaign, Report No. UIUCDCS-R-89-1516, 1989

[13] P. J. Van Der Houwen and B. P. Sommeijier, "Variable Step Iteration of High Order Runge-Kutta Methods on Parallel Computers", Tech. Report NM-R8817, CWI, Amsterdam, 1988

[14] P. J. Van Der Houwen and B. P. Sommeijier, "Block Runge-Kutta Methods on Parallel Computers", Tech. Report NM-R8906, CWI, Amsterdam, 1989

[15] P. J. Van Der Houwen and B. P. Sommeijier, "Iterated Runge-Kutta Methods on Parallel Computers", SIAM J. of Science and Statistics Computing, Vol. 12, No. 5, pp. 1000-1028, 1991

[16] P. J. Van Der Houwen and B. P. Sommeijier, "Parallel ODE Solvers", Computer Architecture News, Vol. 18, No. 3, 1990

[17] J. C. Butcher, "The Numerical Analysis of Ordinary Differential Equations: Runge-Kutta and General Linear Methods", Wiley, New York, 1987

[18] P. J. Van Der Houwen, "Parallel Step-by-Step Methods", Applied Numerical Mathematics, Vol. 11, pp. 69-81, 1993

[19] K. Burrage, "Order Properties of Implicit Multivalue Methods for Ordinary Differential Equations", IMA J. Numerical Analysis, Vol. 8, pp. 43-69, 1988

[20] K. Burrage, "Error Behaviour of Predictor-Corrector Methods", Applied Numerical Mathematics, Vol. 8, pp. 201-216, 1991

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

[22] S. M. Krosel and E. J. Milner, "Applications of Integration Algorithms in a Parallel Processing Environment for Simulation of Jet Engines", 15th Annual Simulation Symposium, Tampa, Florda, 1982

[23] I. N. Katz, M. A. Franklin and A. Sen, "Optimally Stable Parallel Predictor for Adams-Moulton Correctors", Comput. Math. Appl., Vol. 3, pp. 217-233, 1977

[24] O. Abou-Rabia, L. G. Birta and M. Chen, "A Comparative Evaluation of BPC and PPC Methods for Parallel Solution of ODE's", Trans. of Society for Computer Simulation, Vol. 6, No. 4, pp. 265-290, 1989

[25] K. Burrage, "Parallel Methods for Initial Value Problems", Applied Numerical Mathematics, Vol. 11, pp. 5-25, 1993

[26] R. D. Skeel and H. W. Tam, "Potential for Parallelism in Explicit Linear Methods", Computer Science Dept., University of Illinois at Urbana-Champaign, IL, 1991

[27] H. J. Stetter, "Improved Absolute Stability of Predictor-Corrector Schemes", Computing, Vol. 3, pp. 286-196, 1968

[28] K. Jackson and S. P. Nørsett, "The potential for Parallelism in Runge-Kutta Methods, Part 1: RK Formulas in Standard Form", Tech. Report 239/90, Computer Science Dept., University of Toronto, 1990

[29] K. Burrage, "The Search for Holy Grail, or: Predictor-Corrector Methods for Solving ODE IVP's", Applied Numerical Mathematics, Vol. 11, pp. 125-141, 1993

[30] C. W. Gear, "Parallel Methods for Ordinary Differential Equations", University of Illinois at Urbana-Champaign, Report No. UIUCDCS-R-87-1369, 1987

[31] D. Hutchinson and B. M. S. Khalaf, "Parallel Algorithms for Solving Initial Value Problems: Front Broadening and Embedded Parallelism", Parallel Computing, Vol. 17, pp. 957-968, 1991

[32] Fred T. Krogh, "Numerical Integration of Ordinary Differential Equations", 1973