Arab J. Math. Sc.
Volume 16, Number 1, June 2010
pp 47-68

ON THE IMPLEMENTATION OF A BLOCK PREDICTOR-CORRECTOR METHOD FOR INITIAL VALUE PROBLEMS

SALMAN H. ABBAS

ABSTRACT. Recently, a class of block predictor-corrector methods [19] for the numerical solution of initial value problems have been proposed. In this paper, we consider some important computational aspects associated with a new second-order block method. These aspects make the proposed method competitive with respect to all known solvers for IVP's.

Keywords. Block methods; Ordinary differential equations; Parallel methods; Predictor-corrector method.

Mathematics Subject Classification (2000). 35K55, 35K05.

1. INTRODUCTION

The numerical methods for parallel solution of initial value problems (IVPS) are well established techniques in literature [1,2]. In the last two decades number of papers have appeared on this topic (see, [3]-[21] and references there in). One such technique is the block method which by means of a single application of a calculation unit, yields a sequence of new estimates for y of the differential equation:

$$y' = f(x, y), \qquad y(x_0) = y_0 \quad \text{and} \quad y, f \in R^s \tag{1.1}$$

Date received. November 11, 2008

If k is the block size, $k \geq 1$, then in simple cases the values of x at which solutions are computed will be evenly separated. In other words each basic cycle of the calculation has the potential to advance the solution by k new points in the x direction. Each such block can therefore be considered as a unit of calculation. Let y_i denotes the approximation to the exact solution $y(x_i)$ at $x = x_i$, and f_i denotes the value of $f(x_i, y_i)$ the approximation for $y(x_i)$. A block of solutions can be represented by the vector

$$\underline{Y} = [y_1, y_2, ..., y_k]^r \tag{1.2}$$

with $y_j (1 \leq j \leq k)$ and the solution generated at $x_j = x_0 + jh$, to the right hand end point of the preceding block with h the uniform spacing between solution values. Such procedures can be formulated either as implicit or predictor-corrector methods [10]. In addition, the underlying formula may only refer to the end point of the previous block, and called one-step [10] and [12]. Otherwise some or all of the points in the previous block could be used [8] and [10]. Multi-step methods or a number of previous blocks [3] in which case the methods are referred to as multi-block methods.

Chu and Hamilton [3] have examined a few possible multi-block methods for which the theory is well founded, the availability of formulae is not well satisfied. On the other hand formulae for different block size are readily calculated using a single preceding block. It is for this reason that we shall consider methods based upon a single preceding block. Adopting the identical notation used by [6] the implicit matrix vector for single block methods can be expressed:

$$\underline{y} = \underline{\alpha} y_0 + h\underline{\beta} f_o + hD\underline{F} \tag{1.3}$$

where $\underline{\alpha}$ and $\underline{\beta}$ are $K - vectors$, D is a matrix, and $\underline{F}$ is a k-vector whose j^{th} entry is $f_j = f(x_0 + jh, y_j), 1 \leq j \leq k$, As equation (1.3) is implicit in $\underline{y}$ it has to be solved iteratively, using in the first instance a set of tentative ("predicted") solution values. The predictor formula can be expressed in the form:

$$\underline{Y^p} = \underline{\alpha} y_0 + h\underline{\beta} f_0 + A\underline{Y_0} + hB\underline{F_0} \tag{1.4}$$

where $\underline{\alpha}$ and $\underline{\beta}$ are K-vectors, A and B are the $k \times k$ matrices, F_0 is a k-vector whose j^{th} entry is $f_j = f(x_0 - jh, y_j)$ $1 \leq j \leq k$, with y_j the solution generated

at $x = x_0 - jh$ and $\underline{Y_0}$ contains previous information that is :

$$\underline{Y_0} = [y_1, y_2, ..., y_k]^T \tag{1.5}$$

The simplest initial estimate for elements of $\underline{Y}$ is obtained by the one step Euler formula with A and B null matrices and hence,

$$\underline{Y}^p = \underline{\alpha} y_0 + h\underline{\beta} f_0 \tag{1.6}$$

Following an application of equation (1.4) or (1.6), equation (1.3) may be applied iteratively through,

$$\underline{Y}^{i+1} = \underline{\alpha} y_0 + h\underline{\beta} f_o + hD\underline{F^i} \tag{1.7}$$

where $\underline{Y}^{i+1} = Y^p$ and F^i a k-vector holds the derivatives $f(x_0 + jh, y_j^p)$, $j = 1, 2, 3, ..., k$

2. Development of Formulae

The k equations represented in equation (1.3) contain $k(k+2)$ free and as yet unspecified parameters; namely the entries of arrays $\underline{\alpha}$ and $\underline{\beta}$ and D. A set of equations that uniquely determine the unknown parameters can be formulated using the method of undetermined coefficients. Specifically it is necessary to impose the condition that equation (1.3) yields the exact results for the solution to equation (1.1) when $y(x) = (x - x_0)^r$ for $r = 0, 1, 2, .., (k+1)$. With these conditions, all elements of $\underline{\alpha}, \underline{\beta}$ and D can be calculated. Thus a set of formulae of order $k+1$ are determined for a method with block length k and local truncation error of order $k+2$.

Likewise the same conditions can be applied to equation (1.4), which has $2k(k+1)$ unknown, and k^2 free variables. Thus the predictor vector is not uniquely determined and there is an infinity of predictor formulae. The best choice is not obvious, Shampine and Watts [11] considered fixing the matrix A with $a_{ij} = \frac{1}{k+1}$ and thus proceeded to find the remaining unknowns. They also showed that the local truncation error at the end points of a block when the block size is even is of order $k+3$ to achieve thus higher accuracy than expected is achieved.

The equations connecting all coefficients in, $\underline{\alpha}, \underline{\beta}, D$ and B are simply determined in matrix-vector form and are listed in [8]. These equations can

exactly be calculated using REDUCE [14].For example with A and B both null matrices yield the following for $k = 2$.

$$\underline{\alpha} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad \underline{\beta} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \quad \underline{\underline{\alpha}} = \begin{pmatrix} 1 \\ 1 \end{pmatrix},$$
$$\underline{\underline{\beta}} = \begin{pmatrix} \frac{5}{12} \\ \\ \frac{1}{3} \end{pmatrix}, D = \begin{pmatrix} \frac{2}{3} & \frac{-1}{12} \\ \\ \frac{4}{3} & \frac{1}{3} \end{pmatrix},$$

whilst for $k = 4$ we find:

$$\underline{\alpha} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \quad \underline{\beta} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} \underline{\underline{\alpha}} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix},$$

$$\underline{\underline{\beta}} = \begin{pmatrix} \frac{251}{720} \\ \\ \frac{29}{90} \\ \\ \frac{27}{80} \\ \\ \frac{14}{45} \end{pmatrix}, D = \begin{pmatrix} \frac{322}{360} & \frac{-11}{30} & \frac{53}{360} & \frac{-19}{720} \\ \\ \frac{62}{45} & \frac{4}{15} & \frac{2}{45} & \frac{-1}{90} \\ \\ \frac{51}{40} & \frac{9}{10} & \frac{21}{40} & \frac{-3}{80} \\ \\ \frac{64}{45} & \frac{8}{15} & \frac{64}{45} & \frac{14}{45} \end{pmatrix},$$

3. Identification of Parallelism in the Method

Inspection of equations (1.3), (1.4) and (1.5) show the potential for parallelism in the calculation of $\underline{Y}^p$ (equation (1.4)), F^i (equation (1.7)) and also in the calculation of the right hand side of equation (1.3). The granularity of the parallelism is variable and depends upon both the block size and the computational effort to calculate the derivative function . When k is small and is a simple expression, the parallelism is fine grained, and when k becomes large, it becomes time expensive to evaluate and the parallelism changes to coarse grained.

To be more specific, k values of $f(x, y)$, can be calculated in parallel across the block, elements in the right hand side of (1.3) or (1.7) can be calculated

in parallel, and the derivative values have to be distributed (as clarified later). Subsequent iterations then require the same sequence of operations to be repeated.

4. Hardware Realisation

The basic calculations for each cycle of iteration have already been identified. The important side is that y_j^p, f_j^p, y_i^j and f_j^i with $j = 1, 2, ., k$ can be calculated independently provided that the sufficient information exchange takes place in each cycle. The simplest network for this calculation is one in which there are as many processors as the block size (Franklin [7]). It is reasonable to anticipate that under certain conditions, more efficient use of the processors may be achieved by dedicating a larger subset of calculations of each block to each processor. Hence the implemented codes allow for fewer processors than block size. The network chosen was a simple linear chain of P processors can be shown as:

Host⇌Master⇄Slave 1⇆Slave 2⇄Slave 3..... ⇄ Slave P

Figure 1 Network configuration for Parallel

The figure shows p slave transputers, a master and host with communication taking place between the networked transputers along the channels shown. The essential tasks of the various transputers are as follows:

(1) The host or root transputer which communicates the problem information for an expert to the network. This processor hosts a driver program.
(2) A master processor which gathers the problem information and also controls the calculations performed by the P slaves of the network. This processor also communicates results back to the host driver program.
(3) $P(1 < P \leq P_{max})$ slave transputers, each hosting the same program but the data for each being determined by the position of each being determined by the position of each slave with respect to the master and the currently chosen block size k.

The hardware used for tests consisted of 8 networked transputers and hence 7 is the maximum block size that can be used when each processor calculates a single y value. (Details of the original coding with $k = p$ can be found in [11].) With the knowledge of position in the linear chain, each processor is programmed to determine how much information it needs to pass up or down the network. For example in each iteration cycle, the jth processor will calculate a set of next estimates before tested for convergence. The result of the convergence tests and the latest estimates for y_j are then passed up the network for the master to determine if all y_j values, $1 \leq j \leq k$, have converged.

5. Block Method

In this section we consider a block method at the cost of the truncation error in which the order is fixed but the block length may be any value. The method developed uses equation (1.6) to predict solutions to the problem, then applies the following equation iteratively

$$\underline{Y^{i+j'}} = \alpha y_0 + hA\underline{F^p}, \tag{5.1}$$

where α is a unit k-vector, A is a $k \times k+1$ matrix and $\underline{F^p}$ is a $(k+1)$ -vector whose j^{th} entry is $f_{j-1} = f(x_0 + (j-1)h, y^p_{j-1})$ with y^p_j the latest estimate for y_j. We consider an implementation of the method with defined by :

$$a_{11} = a_{22} = \frac{1}{2}, \quad a_{22} = \frac{5}{6} \tag{5.2}$$

$$\begin{aligned} a_{r,1} &= a_{r,r+1} = \frac{2}{5}, \ r = 2, ..., k \\ a_{r,2} &= a_{r,r+1} = \frac{11}{10}, \ r = 3, 4, ..., k \end{aligned}$$

$$a_{rj} = 1, r = 4, 5, ..., k \quad \text{and } j = 3, ..., k \tag{5.3}$$

6. Parallelism in the Method

Inspecting the method, it can be seen that, as in section 2, there is a potential for parallelism in the calculation of the predicted y values, i.e. $\underline{Y^p}$ (eq.(1.6)), and hence the predicted derivative values. The right hand side of (5.1) can be splitted into sub-block work packets and framed out to slaves, allowing a high degree of parallelism to be accomplished.

For example, with 2 processors (5.1) can be manipulated : First with A in its complete form:

$$\underline{Y^{i+1}} = \underline{\alpha} y_0 + h \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & & & & & & & \\ \frac{2}{5} & \frac{6}{5} & \frac{2}{5} & & & & & & \\ \frac{2}{5} & \frac{10}{11} & \frac{10}{11} & \frac{2}{5} & & & & & \\ \frac{2}{5} & \frac{11}{10} & \frac{11}{10} & 1 & \frac{2}{5} & & & & \\ \frac{2}{5} & \frac{11}{10} & 1 & 1 & \frac{2}{5} & \dots & \dots & \dots & \\ \vdots & \vdots & \vdots & & & \dots & \dots & \dots & \\ \frac{2}{5} & \frac{11}{10} & 1 & 1 & 1 & 1 & \dots & \frac{11}{10} & \frac{2}{5} \end{pmatrix} \underline{F^p} \tag{6.1}$$

The calculation of $A\underline{F^p}$ is readily parallelized using a splitting

$$A = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & & & & & & & \\ \frac{2}{5} & \frac{6}{5} & \frac{2}{5} & & & & & & \\ \frac{2}{5} & \frac{11}{10} & \frac{11}{10} & \frac{2}{5} & & & & & \\ \frac{2}{5} & \frac{11}{10} & \frac{11}{10} & 1 & \frac{2}{5} & & & & \\ \frac{2}{5} & \frac{11}{10} & 1 & 1 & \frac{2}{5} & \dots & \dots & \dots & \\ \vdots & \vdots & \vdots & & & \dots & \dots & \dots & \\ \frac{2}{5} & \frac{11}{10} & 1 & 1 & 1 & 1 & \dots & \frac{11}{10} & \frac{2}{5} \end{pmatrix} \tag{6.2}$$

If we write A as :

(6.3)

$$A = \left(\begin{array}{ccccccc|c} \frac{1}{2} & \frac{1}{2} & & & & & & \\ \frac{2}{5} & \frac{6}{5} & \frac{2}{5} & & & & & \\ \frac{2}{5} & \frac{11}{10} & \frac{11}{10} & \frac{2}{5} & & & & 0 \\ \frac{2}{5} & \frac{11}{10} & \frac{11}{10} & 1 & \frac{2}{5} & & & \\ \vdots & \vdots & & \vdots & & & & \\ \frac{2}{5} & \frac{11}{10} & 1 & 1 & \cdots & \frac{11}{10} & \frac{2}{5} & \\ \hline \frac{2}{5} & \frac{11}{10} & 1 & 1 & & & \frac{11}{10} & \\ \frac{2}{5} & \frac{11}{10} & 1 & 1 & & & 1 & \\ \frac{2}{5} & \frac{11}{10} & 1 & 1 & & & 1 & 0 \\ \vdots & & & \vdots & & & & \\ \frac{2}{5} & \frac{11}{10} & 1 & 1 & & & 1 & \end{array}\right) + \left(\begin{array}{c|ccccccc} & & & & & & & \\ & & & & & & & \\ 0 & & & 0 & & & & \\ & & & & & & & \\ & & & & & & & \\ & & & & & & & \\ & & & & & & & \\ \hline & \frac{2}{5} & & & & & & \\ & \frac{11}{10} & \frac{2}{5} & 1 & & & & \\ 0 & 1 & \frac{11}{10} & \frac{2}{5} & & & & \\ & & & & & & & \\ & 1 & 1 & 1 & \cdots & & \frac{11}{10} & \frac{2}{5} \end{array}\right)$$

and further split A up to

(6.4)

$$A = \left(\begin{array}{c|cc} \frac{1}{2} & & \\ \frac{2}{5} & & \\ \frac{2}{5} & 0 & 0 \\ \frac{1}{2} & & \\ \vdots & & \\ \frac{2}{5} & & \\ \hline \frac{2}{5} & & \\ \frac{2}{5} & & \\ \frac{2}{5} & 0 & 0 \\ \frac{1}{2} & & \\ \vdots & & \\ \frac{2}{5} & & \end{array}\right) + \left(\begin{array}{c|ccccc|c} & \frac{1}{2} & & & & & \\ & \frac{6}{5} & \frac{2}{5} & & & & \\ 0 & \frac{11}{10} & \frac{11}{10} & \frac{2}{5} & & & 0 \\ & \frac{11}{10} & 1 & \frac{11}{10} & \frac{2}{5} & & \\ & \vdots & & & & & \\ & \frac{11}{10} & 1 & 1 & \cdots & \frac{11}{10} & \\ \hline & & & & & & \\ & \frac{2}{5} & \frac{11}{10} & 1 & 1 & \frac{11}{10} & \\ & \frac{2}{5} & \frac{11}{10} & 1 & 1 & 1 & \\ 0 & \frac{2}{5} & \frac{11}{10} & 1 & 1 & 1 & 0 \\ & \vdots & & \vdots & & & \\ & \frac{2}{5} & \frac{11}{10} & 1 & 1 & 1 & \end{array}\right) + \left(\begin{array}{cc|ccccc} & & & & & & \\ & & & & & & \\ 0 & 0 & & & & & \\ & & & & & & \\ \vdots & & & & & & \\ & & & & & & \\ \hline & & \frac{2}{5} & & & & \\ & & \frac{11}{10} & \frac{2}{5} & & & \\ 0 & 0 & 1 & \frac{11}{10} & \frac{2}{5} & & \\ & & \vdots & & & & \\ & & & & & & \\ & & 1 & 1 & 1 & \frac{11}{10} & \frac{2}{5} \end{array}\right)$$

or $A = A_0 + A_1 + A_2$ then (6.1) could be represented by

$$\underline{Y}^{i+1} = \underline{\alpha} y_0 + h\{A_0 + A_1 + A_2\}\underline{F}^p \quad \text{which is equivalent to}$$

$$\underline{Y}^{i+1} = \underline{\alpha} y_0 + hA_0\underline{F_0}^p + hA_1\underline{F_1}^p + hA_2\underline{F_2}^p$$

where $\underline{F_0}^p = [f_0, 0, 0, ..., 0]^T$, $\underline{F_1}^p = [0, f_1, f_2, ..., f_{k/2}, 0, 0, ..., 0]^T$ and $\underline{F_2}^p = [0, 0_1, ..., f_{k/2}, f_{k/2+2}, ..., f_k]^T$ are $(k+1)$-vectors. Thus by "splitting up" the $\underline{F}^p$ vector and the matrix A over 2 processors with both processors knowing $\frac{2h}{5} f_0$ and processor 2 knowing an estimate of $\sum f_r, r = 1, ..., \frac{k}{2}$ then estimates for y'_r, $r = 1, ..., \frac{k}{2}$ and $y^i_s, s = \frac{k}{2} + 1, ..., k$, can be obtained in parallel. It should now be apparent that A can be splitted across any number of processors. In general for P processors where, $\frac{k}{P} = q$, we follow the procedure

of 'splitting up' as in equation (6.3), i.e.

$$\underline{Y}^{i+1} = \underline{\alpha} y_0 + hA_0\underline{F_0}^p + hA_1\underline{F_1}^p + hA_2\underline{G_2}^p, \tag{6.5}$$

where $\underline{F_0}^p = [f_0, 0, 0, ..., 0]^T$, $\underline{F_1}^p = [0, f_1, f_2, ..., f_{k/2}, 0, 0, ..., 0]^T$ and $\underline{G_2}^p = [0, 0, ..., f_{q+1}, f_{q+2}, ..., f_k]^T$ = are $(k+1)$- vectors. The "splitting up" continues with B_2:

(6.6)

$$B_2 = \left(\begin{array}{c|cccccc|c} 0 & & & 0 & & & 0 & \\ \hline & \frac{1}{2} & & & & & & \\ & \frac{6}{5} & \frac{2}{5} & & & & & \\ 0 & \frac{11}{10} & \frac{11}{10} & \frac{2}{5} & & & 0 & \\ & \frac{11}{10} & 1 & \frac{11}{10} & \frac{2}{5} & & & \\ & \vdots & & & & & & \\ & \frac{11}{10} & 1 & 1 & \ldots & \frac{11}{10} & \frac{2}{5} & \\ \hline & \frac{11}{10} & 1 & 1 & \ldots & & 1 & \frac{11}{10} \\ & \frac{11}{10} & 1 & 1 & 1 & & & 1 \\ 0 & \frac{11}{10} & 1 & 1 & 1 & & & 1 \\ & & \vdots & & & \vdots & & \\ & \frac{11}{10} & 1 & 1 & \ldots & & \ldots & 1 \end{array}\right.$$

$$\left.\begin{array}{c|cccccc} & & & & & & \\ & & & & & & \\ 0 & 0 & & & & & \\ & & & & & & \\ \vdots & & & & & & \\ & & & & & & \\ \hline & & & & & & \\ & \frac{2}{5} & & & & & \\ & \frac{11}{10} & \frac{2}{5} & & & & \\ 0 & 0 & 1 & \frac{11}{10} & \frac{2}{5} & & \\ & & \vdots & & & & \\ & & & & & & \\ & 1 & 1 & 1 & \frac{11}{10} & \frac{2}{5} \end{array}\right)$$

or $B = B_2 + B_3$, then $\underline{F_2}^p = [0, 0, ..., f_{q+1}, f_{2q}, ...0, 0, ..0]^T$ and $\underline{G_2}^p = [0, 0, ..., f_{2q+1}, f_{2q+2}, ..., f_k]^T$. Similarly we split B_3 into A_3 and B_4

and continue until finally:

(6.7)

$$B_{p-1} = \left(\begin{array}{c|ccccccc} 0 & & & 0 & & & 0 & \\ \hline & \frac{1}{2} & & & & & & \\ & \frac{6}{5} & \frac{2}{5} & & & & & \\ 0 & \frac{11}{10} & \frac{11}{10} & \frac{2}{5} & & & 0 & \\ & \frac{11}{10} & 1 & \frac{11}{10} & \frac{2}{5} & & & \\ & \vdots & & & & & & \\ & \frac{11}{10} & 1 & 1 & \ldots & \frac{11}{10} & \frac{2}{5} & \\ \hline & \frac{11}{10} & 1 & 1 & \ldots & & 1 & \frac{11}{10} \\ & \frac{11}{10} & 1 & 1 & 1 & & & 1 \\ 0 & \frac{11}{10} & 1 & 1 & 1 & & & 1 \\ & & \vdots & & & \vdots & & \\ & \frac{11}{10} & 1 & 1 & \ldots & & \ldots & 1 \end{array}\right) + \left(\begin{array}{c|ccccc} & & & & & \\ & & & & & \\ 0 & 0 & & & & \\ & & & & & \\ \vdots & & & & & \\ & & & & & \\ \hline & & \frac{2}{5} & & & \\ & & \frac{11}{10} & \frac{2}{5} & & \\ 0 & 0 & 1 & \frac{11}{10} & \frac{2}{5} & \\ & & \vdots & & & \\ & & & & & \\ & & 1 & 1 & 1 & \frac{11}{10} & \frac{2}{5} \end{array}\right)$$

or

$$B_{p-1} = A_{p-1} + A_p \tag{6.8}$$

Gives us:

$$\underline{Y}^{i+1} = \underline{\alpha} y_0 + h A_0 \underline{F_0}^p + h \sum_{r=1}^{p} A_r \underline{F_r}^p, \tag{6.9}$$

where $\underline{F_r}^p = [0, 0, ..., f_{(r-1)(q+1)} .... f_{rq}, 0, ..., 0]^T$ and the matrices are defined as above. Thus the work of calculating k y-values may be fared out by "splitting up" $\underline{F_r}^p$ and A into $P + 1$ sub-vectors and matrices. As each processor knows the value of y_0 and f_0, processor P needs only to receive

an estimate of $\sum f_r, r = 1, ...(p-1)q$ to calculate its sub-block of y-values, $y'_r,\ r = (p-1)q+1, ..., pq$.

On inspection of the method it was found that increased accuracy could be achieved in the corrector by updating the estimate of f_{j-1} in the calculation of y^i_j through the sub-block. the equations Consider the first point in a sub-block y'_j. Calculations at this point can be written as:

$$
\begin{aligned}
y^i_1 &= y_0 + \frac{h}{2}f_0 + \frac{h}{2}f_1 \\
y^i_1 &= y_0 + \frac{2h}{5}f_0 + \frac{6h}{5}f_1 + \frac{2}{5}hf_2 \\
y^i_j &= y_0 + \frac{2h}{5}f_0 + \frac{11h}{5}f_1 + \sum_{r=2}^{j-2} f^p_r + \frac{11}{10}hf_{j-1} + \frac{2}{5}hf_2, \qquad (6.10)
\end{aligned}
$$

where $\vec{y_0}$,f_0 and f^p_j 's are communicated, before we start to calculate y^i_j. We could find f^i_j and use this in equation (19) to find an ' updated ' solution for y^i_{j+1}. The method can be seen to be coarse grained, the granularity increasing as the derivative function becomes more expensive to calculate and the sub-block size q increases.

7. Implementation

The method is implemented on a linear chain of P slave processors such that $1 \leq p \leq k$, and the block size in question and a master processor (see fig.1). A $P(EC)^2$ execution of the method is chosen.

The algorithm requires run time specification of P, the number of transputers, k, the block size, and N, the number of steps to be used, and hence the step length h can be computed. The derivative equation, $f(x, y)$ has to be precompiled and loaded on to each slave transputer.

(1) Initial values N, K and P are sent down to the slaves followed by y_0, x_0 and h.
(2) The master then undertakes its own work acting in a sense as a slave.
(3) The final resulting y values are then collected from the slaves and returned to the host.

As there is no error checking there is no need for the master to have any greater control over the slaves. When initiated by the master, the slaves follow the following steps:

(1) Pass through the system, the initial values sent down from the master.
(2) The slaves can now start to calculate the predicted values of y, and hence the predicted derivative values, for a particular sub-block. These are summed and sent down through the system to be used in the corrector by each succeeding slave.
(3) The slaves then apply the corrector for the first time on their sub-block. Values obtained are used to find derivative values which are again summed and sent down the linear chain.
(4) The corrector is applied for the second time to give the final estimate for y.
(5) The final estimates are sent back to the master, each slave reinitializing the block start value as the final y estimate of the preceding block.
(6) Steps (ii) to (v) are repeated till completion of the task.

8. Accuracy

With two applications of the corrector, the kth component of the principal local truncation error is given by

$$\begin{aligned} \epsilon_k &= y(t_n+k) - y(t_n) + h\sum_{j=0}^{k} \alpha_{kj} y(t_{n+k}) \\ &= y(t_{n+k}) - y(t_n) - h(\frac{2}{5})\dot{y}(t_n) + \frac{11}{10}\dot{y}(t_{n+1}) + \sum_{j=2}^{k-2}[\dot{y}(t_n+j)] \\ &\quad + \frac{11}{10}\dot{y}(t_{n+k+1}) + \frac{2}{5}\dot{y}(t_{n+k}) \end{aligned} \tag{8.1}$$

Express ϵ_k as a power series in h and the remainder term of Taylor series expansion is used to derive ϵ_k . Therefore,

$$\epsilon_k = \frac{h^3}{60}(k-6)y^m(t_n) = O(kh^3) \tag{8.2}$$

which is similar to Block Trapezoidal rule [14].

9. Stability analysis

The linear stability properties of the block corrector (eq.(5.1)) are determined through application to the test equation

$$\dot{y} = \lambda y, \lambda < 0 \tag{9.1}$$

where,

(9.2)

$$A = \begin{pmatrix}
1-\frac{z}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & & \\
\frac{-6}{5}z & 1-\frac{2}{5}z & 0 & 0 & 0 & 0 & 0 & & \\
-\frac{11}{6}z & -\frac{11}{10}z & 1-\frac{2}{5}z & 0 & 0 & 0 & 0 & 0 & \\
-\frac{11}{10}z & -z & -\frac{11}{10}z & 1 & 1-\frac{2}{5}z & 0 & 0 & 0 & 0 \\
-\frac{11}{10} & -z & -z & -\frac{11}{10}z & 1-\frac{2}{5}z & 0 & 0 & 0 & \\
-\frac{11}{10} & -z & -z & -z & -\frac{11}{10}z & 1-\frac{2}{5}z & 0 & 0 & 0 \\
-\frac{11}{10} & -z & -z & -z & -z & -\frac{11}{10}z & 1-\frac{2}{5}z & 0 & \\
-\frac{11}{10} & -z & -z & -z & -z & -z & -\frac{11}{10}z & 1-\frac{2}{5}z & \\
-\frac{11}{10} & -z & -z & -z & -z & -z & -z & -\frac{11}{10}z & 1-\frac{2}{5}z \\
-\frac{11}{10} & -z & -z & -z & -z & -z & -z & -z & -\frac{11}{10}z \quad 1-\frac{2}{5}z
\end{pmatrix}$$

$$\begin{aligned} b^T &= (1+\frac{1}{2}Z, 1+\frac{2}{5}Z, 1+\frac{2}{5}Z, ...1+\frac{2}{5}z), \\ Y^T &= (y_{n+1}, y_{n+2}, ..., y_{n+10} \end{aligned}$$

Using Cramer's we find that

$$Y_{n+r} = \frac{D_r(z)}{D(z)} y_n, \quad r = 1, 2, ..., 10 \tag{9.3}$$

where $D(z) = det(Q)$ and $D_r(z) = det(Q_r)$ is obtained form Q by replacing r^{th} the column by vector $\vec{b}$. Absolute stability requires

$$|\frac{D_r(z)}{D(z)}| < 1 \tag{9.4}$$

In general, absolute stability properties depend on the predictor and the mode of implementation. Applying equ. or equ (n) with ν corrections to the standard linear test problem equ. or equ (n), yields $Y^{\nu}_{m+1} = T^m(z)Y^{\nu}_0$
with

$$E = \begin{pmatrix} 0 & 0 & \dots & \dots & \dots & 1 \\ 0 & 0 & \dots & \dots & \dots & 1 \\ 0 & 0 & \dots & \dots & \dots & 1 \\ 0 & 0 & \vdots & \vdots & \vdots & 1 \\ 0 & 0 & \dots & \dots & \dots & 1 \end{pmatrix}$$

the stability matrix $T(z)$ is given as [14].

$$T(z) = E + \sum_{j=1}^{\nu} A^j E z^j + z^{\nu+1} A^{\nu} A_p \tag{9.5}$$

Consequently, the stability boundary is the largest number α such that if$z \in (-\alpha, 0)$ then $\rho(T(z)) < 1$ where $\rho(T(z))$ denotes the spectral radius of $T(z)$. Since $A^{\nu}A_p = ue^T_{k+1}$, hence

$$\rho(T(z)) = \left|1 + kz + .. + \frac{(kz)^p}{\nu!} + u_{k+1}(kz)^{\nu+1}\right| \tag{9.6}$$

Taking $\nu = p$ where p is the order of the block method, it is easy to show that

$$\rho(T(z)) = \left|1 + kz + .. + \frac{(kz)^p}{p!} + \Big(\frac{1}{(p+1)!} - \frac{C^k_{p+1}}{K^{p+1}}\Big)(kz)^{p+1}\right| \tag{9.7}$$

where C^k_{p+1} is the error constant of the method at the k^{th} point in the block. In this case, the stability polynomial is simply a perturbation of that corresponding to a $(p+1)^{th}$ order explicit Rung-Kutta method scaled according to block size.

10. Theoretical speed up Model

An assessment of how the method will perform can be judged from a simple speed-up model of the algorithm, where are the variable named as in section 5 and k is the sub-block size.

TABLE 1. Equivalent MFLOPS(TEST II).

k	ν	$\alpha NBPC$	$\alpha TBPC$
4	2	0.6161	0.6161
	4	0.7492	0.67483
	6	0.8613	0.8608
	8	0.9530	0.9644
	∞	∞	∞
10	2	0.3512	0.2505
	4	0.3230	0.3175
	6	0.3991	0.3842
	8	0.4483	0.4477
	∞	∞	∞

Calculation units

Quantity	Sequential	Parallel
f_0	n_f	n_f
$\underline{Y}^p = \underline{\alpha} y_0 + h\underline{\beta} f_0$	$2k$	$2q$
$\underline{F}^p$	kn_f	qn_f
$\sum_{r=0}^{q} \underline{F}^p$		q
$y_1^i = y_0 + \frac{h}{2} f_0 + \frac{h}{2} f_1$		
$y_2^i = y_0 + \frac{2h}{5} f_0 + \frac{6}{5} h f_1 + \frac{2}{5} h f_2$	$3k + kn_f$	$3q - 1 + (q-1)n_f$
$y_j^i = y_0 + \frac{2h}{5} f_0 + \frac{11}{10} h f_1 + \sum_{r=2}^{j-2} f_r^p + \frac{11}{10} h f_{j-1} + \frac{2}{5} h f_1$		
$\underline{F}^i$		
$\sum_{r=0}^{q} \underline{F_r}^i$		q
$y_1^i = y_0 + \frac{h}{2} f_0 + \frac{h}{2} f_1$		
$y_2^i = y_0 + \frac{2h}{5} f_0 + \frac{6}{5} h f_1 + \frac{2}{5} h f_2$	$3k + (k-1)n_f$	$3q - 1 + (q-1)n_f$
$y_j^i = y_0 + \frac{2h}{5} f_0 + \frac{11}{10} h f_1 + \sum_{r=2}^{j-2} f_r^p + \frac{11}{10} h f_{j-1} + \frac{2}{5} h f_1$		
$\underline{F_r}^{i+1}$		

Thus the times for calculation are :

$$T_s = 3kn_f + 8k \tag{10.1}$$

$$T_p = 3qn_f + 10q - 2 \tag{10.2}$$

When assessing communication costs we look at the worst possible case. The time it takes to communicate between the master and last transputer (through $P-1$ links).

Communicated value	Communicated units
$\sum \underline{F^p}$	$P-1$
$\sum \underline{F^i}$	P-1
q-vector $\underline{Y}^{i+1}$	$(P-1)q$

Thus the total time for communication within the system is :

$$T_c = (q+2)(q+1) \tag{10.3}$$

If we define p to be the ratio of communication to floating point arithmetic then our model yields a speed-up S :

$$S = \frac{3kn_f + 8k}{10q - 2 + 3qn_f + (q+2)(p-1)p} \tag{10.4}$$

11. Results and discussion

Consider of the nonlinear IVP

$$\dot{y} = -\frac{y^3}{2}, \quad y(0) = 1 \tag{11.1}$$

for $t \in [0,4]$ with exact solution $y = 1\sqrt{t+1}$,we use NBPC and TBPC methods with block size $k = 4$ and $k = 10$and various step sizes. Letting e_1 and e_2 the maximum absolute errors at $t = 4$ and using step sizes h_1 and h_2 , and assuming that $e_i = Ch_i^p$,Table 7 gives the maximum absolute error as well includes the observed rates of convergence calculated, using $p = ln(e_1/e_2)/ln(h_1/h_2)$

To investigate numerically the linear stability properties of the block predictor-corrector methods we consider the linear IVP

$$y' = -100(y - sin(t)) + cos(t), \quad y(0) = 0 \tag{11.2}$$

TABLE 2. Approximate Rates of Convergence.

k	h	NBPC	P	TBPC	P
4	0.2	0.10002E-013	-	0.1004E-01	-
	0.1	0.1266E-02	3.00	0.1256E-02	3.00
	0.05	0.1832E-03	2.79	0.1824E-03	2.78
10	0.2	0.9996E-01	-	0.9995E-01	-
	0.1	0.1692E-01	2.66	0.1690E-01	2.56
	0.05	0.2032E-03	3.11	0.1824E-03	3.06

for $t \in [0, 1]$ with exact solution $y = sin(t)$ Table 8 contains the maximum absolute errors using the NBPC and TBPC methods with block sizes $k = 10$

TABLE 3. Approximate Rates of Convergence.

k	h	NBPC	P	TBPC	P
4	0.2	0.10002E-013	-	0.1004E-01	-
	0.1	0.1266E-02	3.00	0.1256E-02	3.00
	0.05	0.1832E-03	2.79	0.1824E-03	2.78
10	0.2	0.9996E-01	-	0.9995E-01	-
	0.1	0.1692E-01	2.66	0.1690E-01	2.56
	0.05	0.2032E-03	3.11	0.1824E-03	3.06

TABLE 4. Maximum Absolute Errors.

h	NBPC	TBPC
$\frac{1}{400}$	0.706E-04	0.7039E-04
$\frac{1}{360}$	0.1503E+01	0.1514E+01
$\frac{1}{320}$	0.2726E+06	0.2784E+06
$\frac{1}{300}$	0.3416E+08	0.3504E+08
$\frac{1}{280}$	0.3923E+09	0.3948E+09

The comparison is made between NBPC (new block predictor) and TBPC (Trapezoidal block predictor corrections) in the behavior of global error. The numerical results presented here are those produced by the above methods when applied to the following test problems.

(1)

$$y' = (20y - y^2)/80, \quad y(0) = 1, 0 \leq x \leq 20$$

(2)

$$y' = ycosx, \quad y(0) = 1, 0 \leq x \leq 1$$

TABLE 5. Test Problem(i)

Number of steps	Step size	Globler NBPC	TBPC
100	0.2	$2.31x10^{-6}$	$1.21x10^{-6}$
200	0.1	$4.56x10^{-7}$	$1.79x10^{-7}$
400	0.05	$3.98x10^{-8}$	$2.88x10^{-8}$
800	0.025	$6.27x10^{-9}$	$5.18x10^{-9}$
1000	0.02	$4.45x10^{-10}$	$3.4x10^{-10}$

TABLE 6. Test Problem(ii)

Number of steps	Step size	Globler NBPC	TBPC
100	0.01	$3.291x10^{-6}$	$3.05x10^{-6}$
200	0.005	$4.57x10^{-7}$	$2.03x10^{-7}$
400	0.0025	$5.67x10^{-8}$	$3.98x10^{-8}$
800	0.00125	$6.10x10^{-9}$	$4.55x10^{-9}$

When implemented with the test equations (9.2) and (9.3) the speed-ups shown in Tables (7) and (8) are achieved.

$$\begin{aligned} \text{TEST I} : y' &= y, 0 \leq x \leq 1, y(0) = 1.0 \\ &\quad \text{Analytic solution} y(x) = e^x \text{and } n_f \sim 1 \end{aligned} \tag{11.3}$$

$$\begin{aligned} \text{TEST II} : y' &= 8x^7, 0 \leq x \leq 1, y(0) = 0.0 \\ &\quad \text{with } x^7 \text{ calculated using power function } (n_f \simeq 48 \\ &\quad \text{Analytic solution and } y(x) = x^8 \text{ and } n_f \sim 48 \end{aligned} \tag{11.4}$$

In both cases h was fixed and equal to 0.0001. where (p) is the predicted value from the model
and (a) is the actual value form experiment.

The ratio chosen for p in the speed-up model is the value applicable to the sending of a large array of REAL32's. This was measured to be 2.3. There is a little amount of communication associated with a single REAL32 but this is negligible and hence we would expect reasonable predictions of performance

TABLE 7. Speed -Up results(TEST I).

Bloc size		Number of processors ,P		
k	2	3	4	5
100 (p)	1.574	1.987	2.262	2.629
(a)	1.410	1.882	2.178	2.481
500 (p)	1.478	1.938	2.302	2.589
(a)	1.452	1.907	2.305	2.596
1000 (p)	1.478	1.941	2.308	2.589
(a)	1.473	1.934	2.3020	2.900

TABLE 8. Speed -Up results(TEST II).

Bloc size		Number of processors, P		
k	2	3	4	5
100 (p)	1.944	2.815	3.768	4.634
(a)	1.938	2.799	3.752	4.588
500 (p)	1.945	2.869	3.774	4.652
(a)	1.941	2.855	3.764	4.613
1000 (p)	1.945	2.869	3.778	4.655
(a)	1.939	2.869	3.762	4.696

with this value. The above tables give excellent correlation between the theoretical and practical performances. Table (3) below, shows the performance of the algorithm, measured in equivalent MFLOPS, for TEST I. The values are

TABLE 9. Equivalent MFLOPS(TEST II).

Bloc size		Number of processors, p		
k	2	3	4	5
100	0.48	0.64	0.76	0.84
500	0.49	0.66	0.81	0.93
1000	0.49	0.67	0.81	0.94

to be compared with 0.35, the rate obtained for a sequential implementation and show the parallel algorithm increases the performance of a sequential solution by 2.5 times. The following tables show speed-up estimates for $k = 100$ but are typical for any block length. The figures indicate the effects of varying both the number of processors and the amount of work in the differential equation and the communication ratio p. It can be seen form the speed-up

TABLE 10. Speed-Up(Ratios ($k = 100, P = 2$).

	Communication ratio ρ					
n_f	1/8	1/4	1/2	1	2	4
1	1.68	1.66	1.63	1.57	1.46	1.29
4	1.81	1.80	1.73	1.74	1.66	1.53
16	1.93	1.92	1.92	1.90	1.87	1.80
64	1.98	1.98	1.98	1.97	1.96	1.94

TABLE 11. Speed-Up(Ratios ($K = 100, P = 5$).

	communication ratio ρ					
n_f	1/8	1/4	1/2	1	2	4
1	4.09	3.93	3.64	3.18	2.53	1.80
4	4.45	4.35	4.15	3.80	3.26	2.53
16	4.79	4.75	4.66	4.49	4.20	3.71
64	4.94	4.93	4.90	4.85	4.75	4.56

model, equation (9.1), and the above tables that if n_f increases and p decreases then $S \to k/q = p$ indicating that maximum efficiency could theoretically be achieved.

12. CONCLUSIONS

Models with satisfactory performance have been developed for block method of second-order with variable block length. It is now feasible to propose the performance of an algorithm which can be determined before implementation of the approach. This methodology can be recommended for all algorithm designs and in particular when the granularity is fine.

REFERENCES

[1] C.W.Gear, The Potentieal for Parallelism in Ordinary Differential Equations, Report No. UIUCDS-R-86-1246, Dept. Computer Science, Univ. of Illinois, Urbana-Champaign, Urbana.

[2] NAG Library Mk12, Numerical Algorithms Group Ltd, Oxford (1988).

[3] G. M.T.Chu and H.Hamilton, Parallel Solution of ODE's by Multi-Block Methods, SIAM J. Sci. Stat. Comput. 8, (1987), 342-358.

[4] E. J. Janowski, M.R. S. Kulenovi and Z. Nurkanovi, Stability of the kth order Lyness' equation with aiod -k coefficient, Int. J. of Bifurcation and Chaos , 17 (2007) 143-152.
[5] Li. Xiuling, J. Wei, Stability and Bifurcation analysis on a delayed neural network model,Int. J. of Bifurcation and Chaos , 15 (2005) 2883-2893.
[6] P.B.Worland, Parallel Methods for the Numerical Solution of Ordinary Differential Equations, IEEE Trans. Comput., C-25(1976), 1045-1048.
[7] M.A.Franklin, Parallel Solution of Ordinary Differential Equations, IEEE Trans. Comp., C-27, (1987) 413-420.
[8] L.Birta and O.Abou-Rabia, Parallel Block Predictor-Corrector Methods for ODE's , IEEE Tans. Comp., C-36 (1987), 229-31..
[9] S.Horiguchi, Y.Kawazoe, H. Nara, "A Parallel Algorithm for the Integration of Ordinary Differential Equations" Proc. Int. Conf. on parallel Processing, Aug. (1984) , 465-469.
[10] J.Nievergelt, Recent Progress in Extrapolation Methods for Ordinary Differential Equations, SIAM Review, Vol.27, (1985), 505-536.
[11] L.F.Shampine and H.A.Watts, Block Implicit One Step Methods, Math. Comput., 23 (1969), 731-740.
[12] A.Watts and L.F.Shampine, A-stable Block Implicit One-step Methods, BIT, 12, p 252-266, (1972).
[13] D.B.Clegg, An Occam Program for the Block Inplicit One Setp Method, Report 89.6, School of Information Science and Technology, (1988), Liverpool Polytechnic.
[14] S. Abbas and L.M.Delves, Parallel Solution of ODE's by One-Step Block Methods, CSMR, Internal Report, Dept. of Computational Statistics and Mathematics, Liverpool University (1990).
[15] L.F.Shampine and H.A.Watts, Practical Solution of Ordeinary Differential Equations by Runge-Kutta Methods , Sandia Lab Report SAND 76-0585 Albuquerque, NM (1976).
[16] J.C.Butcher, Coefficients for the Study of Runge-Kutta Intergration Processes, J. Austral. Math. Soc., 3(1963), 185-201.
[17] A. K. Burrage, Parallel and Sequential Methods for Ordinary differential Equations, Oxford University Press Inc., New York, (1995).
[18] S.Abbas, A Parallel Algorithm for Ordinary Differential Equations Proceeding of the Pakistan Academy of Sciences,33 (1996), 159-163.
[19] S.H Abbas, A New Parallel Method for the System of ODE's, Intl. J. of Applied Sc. And Computations, vol. 2, No. 3, (1996), 453-461
[20] D. Voss, and Abbas, S., Block predictor-corrector Schemes for the Parallel Solution of ODE's, Computer Math. Applic. Vol. 33, (1996), 65-72
[21] S. Abbas, Derivation of New block Methods for the Numerical Solution of First-Order IVP's Inter. J. computer Math., 64, (1997) , 235-244 .
[22] E. J. Janowski, M.R. S. Kulenovi and Z. Nurkanovi, Stability of the kth order Lyness' equation with aiod -k coefficient, Int. J. of Bifurcation and Chaos , 17 (2007) 143-152.
[23] Li. Xiuling, J. Wei Stability and Bifurcation analysis on a delayed neural network model,Int. J. of Bifurcation and Chaos , 15 (2005) 2883-2893.

Salman H. Abbas, Department of Mathematics, College of Science, University of Bahrain, Sakhier, P. O. Box 32038 Bahrain.

E-mail address: dr_salman121@hotmail.com