PARALLEL 3-POINT EXPLICIT BLOCK METHOD FOR SOLVING HIGHER ORDER ORDINARY DIFFERENTIAL DIRECTLY

ZURNI BIN OMAR
School of Information Technology
Universiti Utara Malaysia
06010 UUM, Sintok
Kedah, Malaysia
zurni@webmail.uum.edu.my
and
MOHAMED SULEIMAN
Department of Mathematics
Universiti Putra Malaysia
43400 UPM, Serdang
Selangor, Malaysia
msuleiman@lan.moe.gov.my

ABSTRACT

In this paper, a new method called parallel 3-point explicit block method for solving a single equation of higher order ODE directly using constant step size is developed. This method, which calculates the numerical solution at more than one point simultaneously, is parallel in nature.

The program of the method employed is run on a shared memory Sequent Symmetry S27 parallel computer. Computational advantages are presented comparing the results obtained by the new method with that of conventional 1-point method. The numerical results show that the new method reduces the total number of steps and execution time. The accuracy of the parallel block and the conventional 1-point methods is comparable particularly when finer tolerances are used.

Keywords: Parallel, 3-point, explicit block method, higher order ODEs, directly.

1. INTRODUCTION

Consider the following *d*th order ODE

$$y^d = f(x, y, y', y'', ..., y^{d-1}), \quad y^{(i)}(a) = \eta_i, \quad a \le x \le b. \tag{1.1}$$

Equation (1.1) can be solved by reducing it to the equivalent first order system and then solve it using first order ordinary differentials (ODEs) methods. The disadvantage of these methods is that the system in (1.1) has been enlarged. The other approach is to solve (1.1) directly as discussed in [3], [4], [5], [6], [9] and [10]. There have been quite a number of parallel methods for solving first order ODEs as discussed in [1], [2], [8] and [11]. However, the available methods for solving (1.1) directly are mostly sequential.

In this paper, a new parallel method called 3-point explicit block method is introduced. In this method the interval $[a,b]$ is divided into series of blocks with each block containing three points, i.e x_{n-2}, x_{n-1} and x_n in the first block while x_{n+1}, x_{n+2} and x_{n+3} in the second block (refer to Figure 1) where solutions to (1.1) are to be computed.

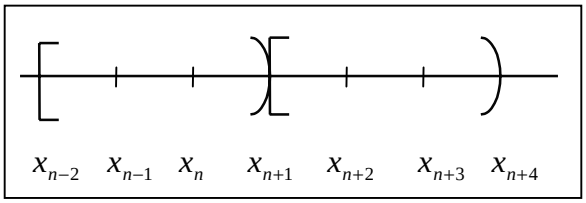

Figure 1: 3-Point Method

The computation which proceeds in blocks is based on the computed values at the earlier blocks. If the computed values at the previous k blocks are used to compute the current block containing 3 points, then the method is called 3-point k-block method. Within a block it is possible to assign the computational tasks at each point to a single processor and therefore the computations can be performed simultaneously.

2. DERIVATION OF THE PARALLEL 3-POINT EXPLICIT BLOCK METHOD

Integrating Equation (1.1) *p* times gives

$$\int_{x_n}^{x_{n+t}} \int_{x_n}^{x} \int_{x_n}^{x} \int_{x_n}^{x} ... \int_{x_n}^{x} y^d(x, y, y', ..., y^{d-1}) dx ... dx = \int_{x_n}^{x_{n+t}} \int_{x_n}^{x} \int_{x_n}^{x} \int_{x_n}^{x} ... \int_{x_n}^{x} f(x, y, y', ..., y^{d-1}) \, dx ... dx \tag{2.1}$$

where

$$x_{n+t} = x_n + th, \; t = 1,2,3.$$

Define $P_{k,n}(x)$ as the interpolation polynomial which interpolates $f(x, y, y', ..., y^{d-1})$ at the k back values namely $\{x_{n-i} \mid i = 0,1,2,..., k-1\}$ as follows

$$P_{k,n}(x) = \sum_{m=0}^{k-1} (-1)^m \binom{-s}{m} \nabla^m f_n$$

where

$$s = \frac{x - x_n}{h}$$

Replacing $f(x, y, y', ..., y^{d-1})$ with $P_{k,n}(x)$ in (2.1) leads to

$$\begin{bmatrix} y_{n+1}^{(n-p)} \\ y_{n+2}^{(n-p)} \\ y_{n+3}^{(n-p)} \end{bmatrix} = \begin{bmatrix} y_n^{(n-p)} \\ y_n^{(n-p)} \\ y_n^{(n-p)} \end{bmatrix} + h \begin{bmatrix} y_n^{(n-p+1)} \\ 2y_n^{(n-p+1)} \\ 3y_n^{(n-p+1)} \end{bmatrix} + \frac{h^2}{2!} \begin{bmatrix} y_n^{(n-p+2)} \\ 2^2 y_n^{(n-p+2)} \\ 3^2 y_n^{(n-p+2)} \end{bmatrix} + ... \frac{h^{p-1}}{(p-1)!} \begin{bmatrix} y_n^{(n-1)} \\ 2^{p-1} y_n^{(n-1)} \\ 3^{p-1} y_n^{(n-1)} \end{bmatrix} + \begin{bmatrix} A_1 \\ A_2 \\ A_3 \end{bmatrix} \tag{2.2}$$

where

$$A_j = \int_{x_n}^{x_{n+j}} \int_{x_n}^{x} \int_{x_n}^{x} \int_{x_n}^{x} ... \int_{x_n}^{x} \sum_{m=0}^{k-1} (-1)^m \binom{-s}{m} \nabla^m f_n \, dx...dx \, .$$

Substituting $dx = hds$ and changing the limit of integration in (2.2) leads to

$$\begin{bmatrix} y_{n+1}^{(n-p)} \\ y_{n+2}^{(n-p)} \\ y_{n+3}^{(n-p)} \end{bmatrix} = \begin{bmatrix} y_n^{(n-p)} \\ y_n^{(n-p)} \\ y_n^{(n-p)} \end{bmatrix} + h \begin{bmatrix} y_n^{(n-p+1)} \\ 2y_n^{(n-p+1)} \\ 3y_n^{(n-p+1)} \end{bmatrix} + \frac{h^2}{2!} \begin{bmatrix} y_n^{(n-p+2)} \\ 2^2 y_n^{(n-p+2)} \\ 3^2 y_n^{(n-p+2)} \end{bmatrix} + ... \frac{h^{p-1}}{(p-1)!} \begin{bmatrix} y_n^{(n-1)} \\ 2^{p-1} y_n^{(n-1)} \\ 3^{p-1} y_n^{(n-1)} \end{bmatrix} + h^p \begin{bmatrix} \sum_{m=0}^{k-1} \alpha_{1,m}^{(p)} \nabla^m f_n \\ \sum_{m=0}^{k-1} \alpha_{2,m}^{(p)} \nabla^m f_n \\ \sum_{m=0}^{k-1} \alpha_{3,m}^{(p)} \nabla^m f_n \end{bmatrix} \tag{2.3}$$

where

$$\alpha_{j,m}^{(p)} = (-1)^m \int_0^j \frac{(1-s)^{p-1}}{(p-1)!} \binom{-s}{m} ds \, .$$

Define the generating functions $G_j^{(p)}(t)$ as follows

$$G_j^{(p)}(t) = \sum_{m=0}^{\infty} \alpha_{j,m}^{(p)} t^m \quad . \tag{2.4}$$

Solving (2.4) leads to the following relationships

$$G_j^{(p)}(t) = \frac{j^{(p-1)} - (p-1)! G_j^{(p-1)}(t)}{(p-1)! \log(1-t)} \text{ for } p = 2,3,...,d \tag{2.5}$$

which can easily be verified using mathematical induction. The solution of (2.5) when is given by:

$$\begin{aligned}
&\alpha_{1,0}^{(1)} = 1,\ \alpha_{1,m+1}^{(1)} = 1 - \sum_{r=0}^{m} \frac{\alpha_{1,r}^{(1)}}{m+2-r} \quad , \\
&\alpha_{2,0}^{(1)} = 2,\ \alpha_{2,m+1}^{(1)} = (m+3) - \sum_{r=0}^{m} \frac{\alpha_{2,r}^{(1)}}{m+2-r} \quad , \\
&\alpha_{3,0}^{(1)} = 3, \alpha_{3,m+1}^{(1)} = \frac{(m+4)(m+3)}{2} - \sum_{r=0}^{m} \frac{\alpha_{3,r}^{(1)}}{m+2-r} \ , \\
&\alpha_{1,0}^{(p)} = \alpha_{1,1}^{(p-1)}, \alpha_{1,m+1}^{(p))} = \alpha_{1,m+2}^{(p-1)} - \sum_{r=0}^{m} \frac{\alpha_{1,r}^{(p)}}{m+2-r} \quad , \\
&\alpha_{2,0}^{(p)} = \alpha_{2,1}^{(p-1)}, \alpha_{2,m+1}^{(p))} = \alpha_{2,m+2}^{(p-1)} - \sum_{r=0}^{m} \frac{\alpha_{2,r}^{(p)}}{m+2-r} \quad , \\
&\alpha_{3,0}^{(p)} = \alpha_{3,1}^{(p-1)}, \alpha_{3,m+1}^{(p))} = \alpha_{3,m+2}^{(p-1)} - \sum_{r=0}^{m} \frac{\alpha_{3,r}^{(p)}}{m+2-r} \text{ for } p = 2,3,...,d.
\end{aligned} \tag{2.6}$$

Note that formula (2.2) can be written in the form

$$\begin{aligned}
\begin{bmatrix} y_{n+1}^{(d-p)} \\ y_{n+2}^{(d-p)} \\ y_{n+3}^{(d-p)} \end{bmatrix} &= \begin{bmatrix} y_n^{(d-p)} \\ y_n^{(d-p)} \\ y_n^{(d-p)} \end{bmatrix} + h \begin{bmatrix} y_n^{(d-p+1)} \\ 2y_n^{(d-p+1)} \\ 3y_n^{(d-p+1)} \end{bmatrix} + \frac{h^2}{2!} \begin{bmatrix} y_n^{(d-p+2)} \\ 2^2 y_n^{(d-p+2)} \\ 3^2 y_n^{(d-p+2)} \end{bmatrix} + ... \\
&\frac{h^{p-1}}{(p-1)!} \begin{bmatrix} y_n^{(d-1)} \\ 2^{p-1} y_n^{(d-1)} \\ 3^{p-1} y_n^{(d-1)} \end{bmatrix} + h^p \begin{bmatrix} \sum_{m=0}^{k-1} \beta_{k-1,m}^{(1,p)} f_{n-m} \\ \sum_{m=0}^{k-1} \beta_{k-1,m}^{(2,p)} \alpha f_{n-m} \\ \sum_{m=1}^{k-1} \beta_{k-1,m}^{(3,p)} f_{n-m} \end{bmatrix}
\end{aligned} \tag{2.7}$$

where

$$\beta_{k-1,m}^{(j,p)} = (-1)^m \sum_{r=m}^{k-1} \binom{r}{m} \alpha_{j,r}^{(p)} . \qquad (2.8)$$

3. TEST PROBLEMS

The following problems were tested on the Sequent Symmetry S27 using the 2-point explicit block method.

Problem 1: $y''' = 2y'' - 4, \quad y(0) = 1,\ y'(0) = 2,\ y''(0) = 6\ ,\ 0 \le x \le 1$

Solution: $y(x) = x^2 + e^{2x}$

Artificial Problem.

Problem 2: $y''' = 8y' - 3y - 4e^x, \quad y(0) = 2,\ y'(0) = -2,\ y''(0) = 10\ ,\ 0 \le x \le 1$

Solution: $y(x) = e^x + e^{-3x}$

Source: Suleiman (1989).

Problem 3: $y^{(iv)} = (x^4 + 14x^3 + 49x^2 + 32x - 12)e^x,$

$y(0) = y'(0) = 0,\ y''(0) = 2,\ y'''(0) = -6\ ,\ 0 \le x \le 1.$

Solution: $y(x) = x^2(1-x)^2 e^x .$

Source: Russel and Shampine (1972).

4. NUMERICAL RESULTS

The numerical tests were performed on the shared memory parallel computer, Sequent S27 which has 6 processors. The programs for E1P and the sequential implementation of the 2PEB method was written in C language whereas parallel C language was used for the parallel implementation. Both languages were supported by the Sequent C library. Each method used 5 back values in its computation. The abbreviations and notations are defined as follows:

h	Step size used
STEPS	Total number of steps taken to obtain the solution
MTD	Method employed
MAXE	Magnitude of the maximum error of the computed solution
TIME	The execution time in microseconds needed to complete the integration in a given range using the parallel computer Sequent S27.
S3PEB	Sequential implementation of the 3-point explicit block method
P3PEB	Parallel implementation of the 3-point explicit block method

The maximum error is defined as follows

$$\text{MAXE} = \max_{1 \le i \le \text{STEPS}} \left(\left| y_i - y(x_i) \right| \right)$$

The comparison of the 3PEB method with the E1P method for solving the test problems in terms of the total number of steps, maximum error and execution times are tabulated in Tables 1-3. Table 4 shows the ratio of steps and times of the 3PEB method to E1P method. The ratios of the two parameters are obtained by dividing the parameters of the latter method with the corresponding parameters of the former methods. Hence, the ratios (also known as speedup) that are greater than one for both parameters indicate the efficiency of the 3PEB method.

Table 1

Comparison Between the E1P and 3PEB Methods for Solving Problem 1

h	MTD	STEPS	MAXE	TIME
10^{-2}	E1P	100	4.41035(-2)	122372
	S3PEB	37	1.75128(-1)	126945
	P3PEB	37	1.75128(-1)	274406
10^{-3}	E1P	1000	4.39119(-3)	1142734
	S3PEB	337	6.28697(-3)	1123455
	P3PEB	337	6.28697(-3)	833854
10^{-4}	E1P	10000	4.38927(-4)	11430453
	S3PEB	3337	4.58567(-4)	11171714
	P3PEB	3337	4.58567(-4)	7986848
10^{-5}	E1P	100000	4.38908(-5)	114082053
	S3PEB	33337	4.40879(-5)	111392883
	P3PEB	33337	4.40879(-5)	78242291

Table 2

Comparison Between the E1P and 3PEB Methods for Solving Problem 2

h	MTD	STEPS	MAXE	TIME
10^{-2}	E1P	100	1.18234(-1)	130511
	S3PEB	37	1.10115(-1)	135708
	P3PEB	37	1.10115(-1)	254185
10^{-3}	E1P	1000	1.17139(-2)	1223205
	S3PEB	337	1.17042(-2)	1209884
	P3PEB	337	1.17042(-2)	913258
10^{-4}	E1P	10000	1.17030(-3)	12235538
	S3PEB	3337	1.17029(-3)	12036763
	P3PEB	3337	1.17029(-3)	8798665
10^{-5}	E1P	100000	1.17019(-4)	122141796
	S3PEB	33337	1.17019(-4)	120063725
	P3PEB	33337	1.17019(-4)	87246831

Table 3

Comparison Between the E1P and 3PEB Methods for Solving Problem 3

h	MTD	STEPS	MAXE	TIME
10^{-2}	E1P	100	1.00778(-2)	210474
	S3PEB	37	1.00779(-2)	201983
	P3PEB	37	1.00779(-2)	325724
10^{-3}	E1P	1000	1.00078(-3)	2006247
	S3PEB	337	1.00078(-3)	1850179
	P3PEB	337	1.00078(-3)	1583939
10^{-4}	E1P	10000	1.00008(-4)	20077275
	S3PEB	3337	1.00008(-4)	18504213
	P3PEB	3337	1.00008(-4)	15212065
10^{-5}	E1P	100000	1.00001(-5)	200307659
	S3PEB	33337	1.00001(-5)	184047416
	P3PEB	33337	1.00001(-5)	151403794

Table 4

The Ratio Steps and Execution Times of the 3PEB Method to the E1P Method for Solving Higher Order ODEs

TOL	MTD	RATIO STEP	RATIO PROB.1	TIME PROB.3	PROB.4
10^{-2}	S3PEB	2.70270	0.96398	0.96171	1.04204
	P3PEB	2.70270	0.44595	0.51345	0.64617
10^{-3}	S3PEB	2.96736	1.01716	1.01101	1.08435
	P3PEB	2.96736	1.37043	1.33939	1.26662
10^{-4}	S3PEB	2.99670	1.02316	1.01651	1.08501
	P3PEB	2.99670	1.43116	1.39061	1.31983
10^{-5}	S3PEB	2.99967	1.02414	1.01731	1.08835
	P3PEB	2.99967	1.45806	1.39996	1.32300

5. COMMENTS ON THE RESULTS AND CONCLUSION

It is apparent from the results that the 3PEB method outperforms the E1P method in term of the total number of steps. As the step size becomes finer, the 3PEB method reduces the number of steps to almost one half. These results are expected since the 3PEB method approximates the numerical solution at three points respectively at the same time, thus reducing the number of steps taken by the method.

In term of accuracy, both E1P and 3PEB methods have the same order of accuracy.

As expected, the execution times taken by the parallel implementation of the 3PEB method are more than those taken by the sequential counterpart and the E1P method at $h = 10^{-2}$. This is because the number of steps taken is small and most of the execution times are dominated by the parallel overheads. However, the timings of the parallel version of the 3PEB method are better then other methods when $h < 10^{-2}$. The reason for these gains is that as the step size gets smaller, more steps are taken to complete the computation. By using 3 processors instead of 1, the computation can be performed quicker. In other words, the parallelism in the 3PEB method could really be exploited. The results also suggest that parallel 3PEB method is recommended for solving second order ODEs directly using finer step sizes.

References

[1] Birta L.G. & Abou-Rabia O. 1987. Parallel Block Predictor-Corrector Methods for ODEs, IEEE Transactions on Computers, Vol C-36, No.3, pp. 299-311.

[2] Chu M.T. & Hamilton H. 1987. Parallel Solution of ODEs by Multi-Block Methods, Siam J. Sci. Stat. Comput., Vol 8, No. 1, pp**.** 342-353.

[3] Gear, C.W. 1966. The Numerical Integration of Ordinary Differential Equations. *Math. Comp*. **21**: 146-156.

[4] Gear, C.W. 1971. *Numerical Initial Value Problems in Ordinary Differential Equations.* New Jersey: Prentice Hall, Inc.

[5] Gear, C.W. 1978. The Stability of Numerical Methods for Second-Order Ordinary Differential Equations. *SIAM J. Numer. Anal*. **15**(1): 118-197.

[6] Hall, G. and Suleiman, M.B. 1981. Stability of Adams-Type Formulae for Second-Order Ordinary Differential Equations. *IMA J. Numer. Anal*. **1**: 427-428.

[7] Russel, R.D. and Shampine, L.F. 1972. A Collocation Method for Boundary Value Problems. *Num. Math* **19**: 1-28.

[8] Shampine L.F. and Watts H.A. 1969. Block implicit one-step methods, Math. Comp., Vol. 23, pp 731-740.

[9] Suleiman, M.B. 1979. Generalised Muiltistep Adams and Backward Differentiation Methods for the Solution of Stiff and Non-Stiff Ordinary Differential Equations. PhD Thesis. University of Manchester.

[10] Suleiman, M.H. 1989. Solving Higher Order ODEs Directly by the Direct Integration Method. *Applied Mathematics and Computation* **33(**3): 197-219.

[11] Tam H.W. 1989. Parallel Methods For The Numerical Solution Of Ordinary Differential Equations, Report No. UIUCDCS-R-89-1516. Department of Computer Science, University of Illinois at Urbana-Champaign.