MATEMATIKA, 2005, Jilid 21, Bil. 1, hlm. 15–23
©Jabatan Matematik, UTM.

Solving Higher Order Ordinary Differential Equations Using Parallel 2-Point Explicit Block Method

[1]**Zurni Omar & [2]Mohamed Suleiman**

[1]Faculty of Quantitative Sciences, Universiti Utara Malaysia,
06010 UUM, Sintok, Kedah. zurni@uum.edu.my

[2]National Accreditation Board Floor 14B, Menara PKNS-PJ No.17, Jalan Yong Shook Lin,
46050 Petaling Jaya, Selangor. msuleiman@lan.moe.gov.my

Abstract The derivation of a new parallel 2-point explicit block (2PEB) method for solving dth order ordinary differential equations (ODEs) directly is made. Computational advantages are presented comparing the results obtained by the new method with that of conventional 1-point method. Numerical results suggest that the parallel 2PEB method is recommended for solving second order ODEs directly using finer step sizes.

Keywords Parallel, block method, ODEs.

1 Introduction

We shall consider the following dth order ODE

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

Equation (1) can be solved by using either direct or first order methods. In order to use the latter methods, Equation (1) must be reduced to the equivalent first order system. The direct methods available are mostly sequential in nature, meaning that the numerical solution to (1) is computed at one point at a time. Some of these methods can be found in [4], [5], [6], [7], [12] and [13]. There have been quite a number of parallel methods for solving first order ODEs. One of the methods is parallel block method as discussed in [1], [2], [11] and [14].

In a parallel block method, a set of new values that are obtained by each application of the formula is referred as "block". For instance, in a r-point block method, r new equally spaced solution values i.e

$$y_{n+1}, y_{n+2}, ..., y_{n+r}$$

are obtained simultaneously at each iteration of the algorithm. There are two types of block methods; *one-step block* and *multi-block.* In one-step block method, the new block

$$y_{n+i}, i = 1, 2, ..., r,$$

is computed from the value y_n. On the other hand, in multi-block method, the information from one or more previous blocks is used in the computation of the next block.

Though there has been much effort and discussion on solving first order ODEs in parallel, not much attention is given on solving higher order ODES directly using parallel methods. The general theory of special second order initial value problems has been discussed by Fatunla [3].

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

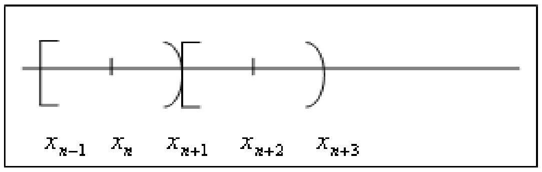


Figure 1: 2-Point Method

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 r points, then the method is called r-point k-block method. For example, in Figure 2 the values used at the previous three blocks are

$$x_{n-5}, x_{n-4}, x_{n-3}, x_{n-2}, x_{n-1}, x_n$$

to compute the solutions of (1) at x_{n+1} and x_{n+2}. This method is known as a 2-point 3-block method.

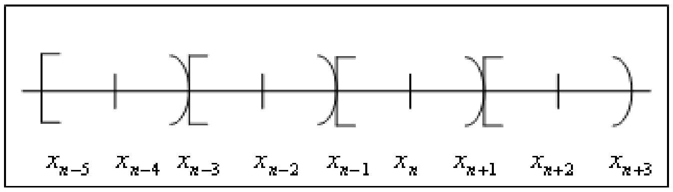


Figure 2: 2-Point 3-Block Method

Since the computational tasks at each point within a block are independent, it is possible to assign the tasks to different processors so that the computations can be performed simultaneously.

2 Derivation of the Parallel 2-Point Explicit Block Method

Let $x_{n+t} = x_n + th,\ t = 1, 2$. Integrating Equation (1) p times gives

$$\underbrace{\int_{x_n}^{x_{n+t}} \int_{x_n}^{x} \int_{x_n}^{x} \int_{x_n}^{x} \ldots \int_{x_n}^{x}}_{\leftarrow p \text{ times} \rightarrow} y^d(x, y, y', ..., y^{d-1}) dx...dx = \int_{x_n}^{x_{n+t}} \int_{x_n}^{x} \int_{x_n}^{x} \int_{x_n}^{x} \ldots \int_{x_n}^{x} f(x, y, y', ..., y^{d-1}) dx...dx$$

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

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

where $s=\dfrac{x-x_n}{h}$

Approximating $f(x,y,y')$ with $P_{k-1,n}(x)$, we then have

$$\begin{aligned}&\begin{bmatrix}y_{n+1}^{(n-p)}\\y_{n+2}^{(n-p)}\end{bmatrix}=\begin{bmatrix}y_n^{(n-p)}\\y_n^{(n-p)}\end{bmatrix}+h\begin{bmatrix}y_n^{(n-p+1)}\\2y_n^{(n-p+1)}\end{bmatrix}+\tfrac{h^2}{2!}\begin{bmatrix}y_n^{(n-p+2)}\\2^2y_n^{(n-p+2)}\end{bmatrix}+...\\&\tfrac{h^{p-1}}{(p-1)!}\begin{bmatrix}y_n^{(n-1)}\\2^{p-1}y_n^{(n-1)}\end{bmatrix}+\begin{bmatrix}A_1\\A_2\end{bmatrix}\end{aligned}\tag{2}$$

where

$$A_1=\int_{x_n}^{x_{n+1}}\int_{x_n}^{x}\int_{x_n}^{x}\int_{x_n}^{x}...\int_{x_n}^{x}\sum_{m=0}^{k-1}(-1)^m\begin{pmatrix}-s\\m\end{pmatrix}\nabla^m f_n dx...dx$$

and

$$A_2=\int_{x_n}^{x_{n+2}}\int_{x_n}^{x}\int_{x_n}^{x}\int_{x_n}^{x}...\int_{x_n}^{x}\sum_{m=0}^{k-1}(-1)^m\begin{pmatrix}-s\\m\end{pmatrix}\nabla^m f_n dx...dx$$

Replacing $dx=h\,ds$ and changing the limit of integration in (2) leads to

$$\begin{aligned}&\begin{bmatrix}y_{n+1}^{(n-p)}\\y_{n+2}^{(n-p)}\end{bmatrix}=\begin{bmatrix}y_n^{(n-p)}\\y_n^{(n-p)}\end{bmatrix}+h\begin{bmatrix}y_n^{(n-p+1)}\\2y_n^{(n-p+1)}\end{bmatrix}+\tfrac{h^2}{2!}\begin{bmatrix}y_n^{(n-p+2)}\\2^2y_n^{(n-p+2)}\end{bmatrix}+...\\&\tfrac{h^{p-1}}{(p-1)!}\begin{bmatrix}y_n^{(n-1)}\\2^{p-1}y_n^{(n-1)}\end{bmatrix}+h^p\begin{bmatrix}\sum\limits_{m=0}^{k-1}\gamma_m^{(p)}\nabla^m f_n\\\sum\limits_{m=0}^{k-1}\delta_m^{(p)}\nabla^m f_n\end{bmatrix}\end{aligned}\tag{3}$$

where

$$\gamma_m^{(p)}=(-1)^m\int_0^1\frac{(1-s)^{p-1}}{(p-1)!}\begin{pmatrix}-s\\m\end{pmatrix}ds$$

and

$$\delta_m^{(p)}=(-1)^m\int_0^2\frac{(2-s)^{p-1}}{(p-1)!}\begin{pmatrix}-s\\m\end{pmatrix}ds$$

Let the generating functions $G^{(p)}(t)$ and $S^{(p)}(t)$ defined as follows

$$G^{(p)}(t)=\sum_{m=0}^{\infty}\gamma_m^{(p)}t^m\tag{4}$$

$$S^{(p)}(t)=\sum_{m=0}^{\infty}\delta_m^{(p)}t^m\tag{5}$$

which leads to the following relationships

$$G^{(p)}(t) = \frac{1^{(p-1)} - (p-1)!G^{(p-1)}(t)}{(p-1)!\log(1-t)} \tag{6}$$

$$S^{(p)}(t) = \frac{2^{(p-1)} - (p-1)!S^{(p-1)}(t)}{(p-1)!\log(1-t)},\ p = 2, 3, ..., d. \tag{7}$$

The above relationships can easily be verified using mathematical induction. Solving (6) and (7) gives the following solution

$$\begin{array}{l} \gamma_0^{(p)} = \gamma_1^{(p-1)} \\ \gamma_{m+1}^{(p))} = \gamma_{m+2}^{(p-1)} - \sum\limits_{r=0}^{m} \frac{\gamma_r^{(p)}}{m+2-r} \\ \delta_0^{(p)} = \delta_1^{(p-1)} \\ \delta_{m+1}^{(p))} = \delta_{m+2}^{(p-1)} - \sum\limits_{r=0}^{m} \frac{\delta_r^{(p)}}{m+2-r} \end{array} \tag{8}$$

The solution of (3) when $p = 1$ is given by (refer to [9])

$$\begin{array}{l} \gamma_0^{(1)} = 1, \gamma_{m+1}^{(1)} = 1 - \sum\limits_{r=0}^{m} \frac{\gamma_r^{(1)}}{m+2-r} \\ \delta_0^{(1)} = 2, \delta_{m+1}^{(1)} = (m+3) - \sum\limits_{r=0}^{m} \frac{\delta_r^{(1)}}{m+2-r} \end{array} \tag{9}$$

Note that formula (3) can be written in the form

$$\begin{array}{l} \left[\begin{array}{c} y_{n+1}^{(n-p)} \\ y_{n+2}^{(n-p)} \end{array}\right] = \left[\begin{array}{c} y_n^{(n-p)} \\ y_n^{(n-p)} \end{array}\right] + h \left[\begin{array}{c} y_n^{(n-p+1)} \\ 2y_n^{(n-p+1)} \end{array}\right] + \frac{h^2}{2!} \left[\begin{array}{c} y_n^{(n-p+2)} \\ 2^2 y_n^{(n-p+2)} \end{array}\right] + ... \\ \frac{h^{p-1}}{(p-1)!} \left[\begin{array}{c} y_n^{(n-1)} \\ 2^{p-1} y_n^{(n-1)} \end{array}\right] + h^p \left[\begin{array}{c} \sum\limits_{m=0}^{k-1} \beta_{k-1,m}^{(p)} f_{n-m} \\ \sum\limits_{m=0}^{k-1} \alpha_{k-1,m}^{(p)} \alpha f_{n-m} \end{array}\right] \end{array} \tag{10}$$

where

$$\beta_{k-1,m}^{(p)} = (-1)^m \sum_{r=m}^{k-1} \left(\begin{array}{c} r \\ m \end{array}\right) \gamma_r^{(p)} \quad \text{and} \quad \alpha_{k-1,m}^{(p)} = (-1)^m \sum_{r=m}^{k-1} \left(\begin{array}{c} r \\ m \end{array}\right) \delta_r^{(p)} \tag{11}$$

3 Test Problems

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

Problem 1:	$y''' = 2y'' - 4, y(0) = 1, y'(0) = 2, y''(0) = 6, 0 \leq x \leq 1$ Solution: $y(x) = x^2 + e^{2x}$ Artificial Problem.
Problem 2:	$y''' = y + 3e^x, y(0) = 0, y'(0) = 1, y''(0) = 2,\ 0 \leq x \leq 1$ Solution: $y(x) = xe^x$ Source: Krogh [8].
Problem 3:	$y''' = 8y' - 3y - 4e^x, y(0) = 2, y'(0) = -2, y''(0) = 10, 0 \leq x \leq 1$ Solution: $y(x) = e^x + e^{-3x}$ Source: Suleiman [13].
Problem 4:	$y^{(iv)} = (x^4 + 14x^3 + 49x^2 + 32x - 12)e^x,$ $y(0) = y'(0) = 0, y''(0) = 2, y'''(0) = -6, 0 \leq x \leq 1$ Solution: $y(x) = x^2(1-x)^2e^x$. Source: Russel and Shampine [10].

4 Numerical Results

The numerical tests were performed on the shared memory parallel computer, Sequent S27 which has 6 processors. The programs for explicit 1-point (E1P) method 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.
S2PEB	Sequential implementation of the 2-point explicit block method
P2PEB	Parallel implementation of the 2-point explicit block method.

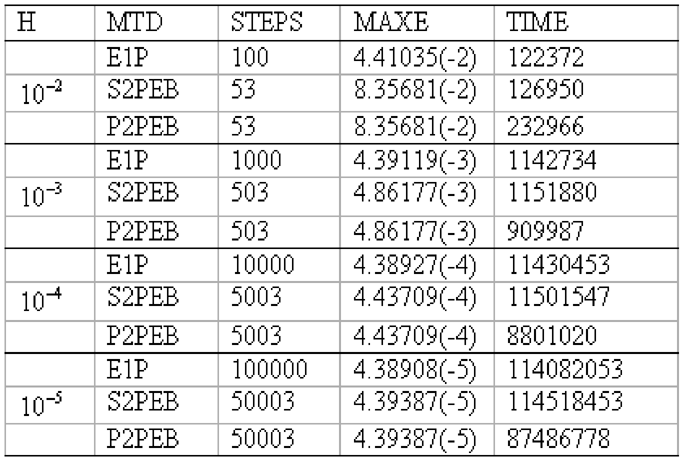
The maximum error is defined as follows

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

The comparison of the 2PEB 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 1-4. Table 5 shows the ratio of steps and times of the 2PEB 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) which are greater than one for both parameters indicate the efficiency of the 2PEB method.

Table 1

Comparison Between the E1P and 2PEB Methods for Solving Problem 1



H	MTD	STEPS	MAXE	TIME
	E1P	100	4.41035(-2)	122372
10^{-2}	S2PEB	53	8.35681(-2)	126950
	P2PEB	53	8.35681(-2)	232966
	E1P	1000	4.39119(-3)	1142734
10^{-3}	S2PEB	503	4.86177(-3)	1151880
	P2PEB	503	4.86177(-3)	909987
	E1P	10000	4.38927(-4)	11430453
10^{-4}	S2PEB	5003	4.43709(-4)	11501547
	P2PEB	5003	4.43709(-4)	8801020
	E1P	100000	4.38908(-5)	114082053
10^{-5}	S2PEB	50003	4.39387(-5)	114518453
	P2PEB	50003	4.39387(-5)	87486778

5 Comments on the Results and Conclusion

It is apparent from the results that the 2PEB method outperforms the E1P method in term of the total number of steps. As the step size becomes finer, the 2PEB method reduces the number of steps to almost one half. These results are expected since the 2PEB method approximate the numerical solution at two points respectively at the same time, thus reducing the number of steps taken by the method.

In terms of accuracy, both E1P and 2PEB methods have the same order. Except for Problem 2, the execution times taken by the sequential implementation of 2PEB method for solving other problems are less encouraging compared to the parallel counterpart and E1P method. The extra time could be resulted from calculating the extra integration coefficients. As expected, the execution times taken by the parallel implementation of the 2PEB 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 2PEB 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 2 processors instead of 1, the computation can be performed quicker. In other words, the parallelism in the 2PEB method could really be exploited. The results also suggest that parallel 2PEB method is recommended for solving second order ODEs directly using finer step sizes.

Table 2

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

H	MTD	STEPS	MAXE	TIME
	E1P	100	7.63447(-3)	121066
10^{-2}	S2PEB	53	7.63448(-3)	106065
	P2PEB	53	7.63448(-3)	241380
	E1P	1000	7.62628(-4)	1126827
10^{-3}	S2PEB	503	7.62628(-4)	951148
	P2PEB	503	7.62628(-4)	941787
	E1P	10000	7.62546(-5)	11270476
10^{-4}	S2PEB	5003	7.62546(-5)	9486304
	P2PEB	5003	7.62546(-5)	9129636
	E1P	100000	7.62539(-6)	112391084
10^{-5}	S2PEB	50003	7.62538(-6)	94637449
	P2PEB	50003	7.62538(-6)	90659201

Table 3

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

H	MTD	STEPS	MAXE	TIME
	E1P	100	1.18234(-1)	130511
10^{-2}	S2PEB	53	1.16165(-1)	135853
	P2PEB	53	1.16165(-1)	256096
	E1P	1000	1.17139(-2)	1223205
10^{-3}	S2PEB	503	1.17115(-2)	1240065
	P2PEB	503	1.17115(-2)	1034676
	E1P	10000	1.17030(-3)	12235538
10^{-4}	S2PEB	5003	1.17029(-3)	12371365
	P2PEB	5003	1.17029(-3)	10051687
	E1P	100000	1.17019(-4)	122141796
10^{-5}	S2PEB	50003	1.17019(-4)	123347394
	P2PEB	50003	1.17019(-4)	99941712

Table 4

Comparison Between the E1P and 2PEB Methods for Solving Problem 4

H	MTD	STEPS	MAXE	TIME
10^{-2}	E1P	100	1.00778(-2)	210474
	S2PEB	53	1.00778(-2)	222259
	P2PEB	53	1.00778(-2)	296038
10^{-3}	E1P	1000	1.00078(-3)	2006247
	S2PEB	503	1.00078(-3)	2076058
	P2PEB	503	1.00078(-3)	1623001
10^{-4}	E1P	10000	1.00008(-4)	20077275
	S2PEB	5003	1.00008(-4)	20727420
	P2PEB	5003	1.00008(-4)	15922113
10^{-5}	E1P	100000	1.00001(-5)	200307659
	S2PEB	50003	1.00001(-5)	206611648
	P2PEB	50003	1.00001(-5)	158493054

Table 5
The Ratio Steps and Execution Times of the 2PEB Method to the E1P Method for Solving Higher Order ODEs

TOL	MTD	RATIO		RATIO	TIME	
		STEP	PROB.1	PROB.2	PROB.3	PROB.4
10^{-2}	S2PEB	1.88679	0.96394	1.14143	0.96068	0.94698
	P2PEB	1.88679	0.52528	0.50156	0.50962	0.71097
10^{-3}	S2PEB	1.98807	0.99206	1.18470	0.98640	0.96637
	P2PEB	1.98807	1.25577	1.19648	1.18221	1.23613
10^{-4}	S2PEB	1.99880	0.99382	1.18808	0.98902	0.96863
	P2PEB	1.99880	1.29877	1.23449	1.21726	1.26097
10^{-5}	S2PEB	1.99988	0.99619	1.18760	0.99023	0.96949
	P2PEB	1.99988	1.30399	1.23971	1.22213	1.26383

References

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

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

[3] S.O. Fatunla, *Block Methods for Second Order ODEs,* Intern. J. Computer Math 40(1990), 55-63.

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

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

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

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

[8] F.T. Krogh, *A Variable Step, Variable Order Multistep Method for the Numerical Solution of Ordinary Differential Equation,* Proceedings of the IFIP Congress in Information Processing 68(1968), 194-199.

[9] Z.B. Omar and M.B. Suleiman, *Solving Second Order ODEs Directly Using Parallel 2-Point Explicit Block Method,* Prosiding Kolokium Kebangsaan Pengintegrasian Teknologi Dalam Sains Matematik, Universiti Sains Malaysia, 1999, pp 390-395.

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

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

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

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

[14] H.W. Tam, *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, 1989.