World Academy of Science, Engineering and Technology
International Journal of Mathematical and Computational Sciences
Vol:2, No:4, 2008

Parallel Direct Integration Variable Step Block Method for Solving Large System of Higher Order Ordinary Differential Equations

Zanariah Abdul Majid, and Mohamed Suleiman

***Abstract*—**The aim of this paper is to investigate the performance of the developed two point block method designed for two processors for solving directly non stiff large systems of higher order ordinary differential equations (ODEs). The method calculates the numerical solution at two points simultaneously and produces two new equally spaced solution values within a block and it is possible to assign the computational tasks at each time step to a single processor. The algorithm of the method was developed in C language and the parallel computation was done on a parallel shared memory environment. Numerical results are given to compare the efficiency of the developed method to the sequential timing. For large problems, the parallel implementation produced 1.95 speed-up and 98% efficiency for the two processors.

***Keywords*—**Numerical methods, parallel method, block method, higher order ODEs.

I. Introduction

THE ever-increasing advancement in computer technology has enabled many in science and engineering sectors to apply numerical methods using parallel computation to solve mathematical models arising from physical phenomena. The numerical solution of large ODEs systems requires a large amount of computing power. Users of parallel computing tend to be those with large mathematical problems to solve with the desire to obtain faster and more accurate results.

In this paper, we consider solving directly the higher order non stiff IVPs (Initial Value Problems) for system of ODEs of the form,

$$y'' = f(x, y, y'),\ y(a) = y_0,\ y'(a) = y_0',\ x \in [a, b]. \quad (1)$$

Equation (1) can be reduced to the equivalent first-order system of twice the dimension equations and then solved using any numerical method. This approach is very well established but it obviously will enlarge the dimension of the equations. The approach for solving the system of higher order ODEs directly has been suggested by several researchers such as in [1] – [8]. In previous work of [8], a general *r* block method of multistep method for solving problems of the form (1) has been investigated. The code in [8] used a repetitive computation of the divided differences and integration coefficients that can be very costly. The worked in [7] has presented a direct block method (2PFDIR) for solving higher order ODEs in variable step size which is faster in terms of timing and comparable or better in terms of accuracy to the existence direct non block method in [8]. The 2PFDIR method will store the coefficients in the code and there will be no calculations that involved the divided difference and integration coefficients. In this paper, we would like to extend the discussions in [7] on the performance of 2PFDIR method using parallel environment particularly focus on the cost of time computation by comparing the execution time of sequential and parallel implementation for solving large problem.

II. Formulation of the Method

In Fig. 1, the two values of y_{n+1} and y_{n+2} are simultaneously computed in a block using the same back values. The block has the step size h and the previous back block has the step size rh. The idea of having the ratio *r* is for variable step size implementation.

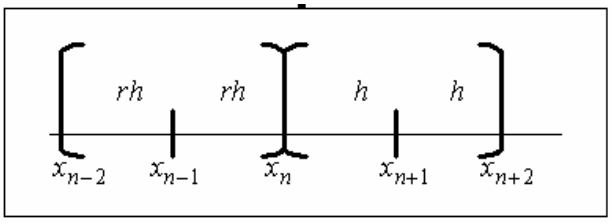

Fig. 1 Two point block method

In Equation (1), the $f(x, y, y')$ will be replaced with Lagrange interpolation polynomial and the interpolation points involved were $(x_{n-2}, f_{n-2}), \ldots, (x_{n+2}, f_{n+2})$. These polynomial will be integrate once and twice over the interval $[x_n, x_{n+1}]$ and $[x_n, x_{n+2}]$ respectively, and the following corrector formulae will be obtained,

Manuscript received May 23, 2008.

Zanariah Abdul Majid is with the Mathematics Department, Faculty Science, Universiti Putra Malaysia, 43400 UPM, Serdang, Selangor Darul Ehsan, Malaysia (phone: 603-89467959; fax: 603-89437958; e-mail: zanariah@math.upm.edu.my).

Mohamed Suleiman is with the Mathematics Department, Faculty Science, Universiti Putra Malaysia, 43400 UPM, Serdang, Selangor Darul Ehsan, Malaysia. (e-mail: m1suleiman@yahoo.com.my).

Integrate Once

First point:

$$y'(x_{n+1}) = y'(x_n) + \frac{h}{240(r+1)(r+2)(2r+1)r^2}\Big[\cdot\left(-(2r+1)r^2\left(3+15r+20r^2\right)f_{n+2} + 4r^2(r+2)\left(18+75r+80r^2\right)f_{n+1} + (r+1)(r+2)(2r+1)\left(7+45r+100r^2\right)f_n - 4(2r+1)(7+30r)f_{n-1} + (r+2)(7+15r)f_{n-2}\right)\Big]. \quad (2)$$

Second point:

$$y'(x_{n+2}) = y'(x_n) + \frac{h}{15r^2(2r+1)(r+2)(r+1)}\Big[\cdot r^2(2r+1)\left(5r^2+15r+9\right)f_{n+2} + 4r^2(r+2)\left(10r^2+15r+6\right)f_{n+1} + (r+2)(r+1)(2r+1)\left(5r^2-1\right)f_n + 4(2r+1)f_{n-1} - (r+2)f_{n-2}\Big]. \quad (3)$$

Integrate twice

First point:

$$y(x_{n+1}) - y(x_n) - hy'(x_n) = \frac{h^2}{240r^2(r+1)(r+2)(2r+1)}\Big[\cdot -r^2(2r+1)\left(1+6r+10r^2\right)f_{n+2} + 4r^2(r+2)\left(4+21r+30r^2\right)f_{n+1} + (r+1)(r+2)(2r+1)\left(3+24r+70r^2\right)f_n - 4(2r+1)(3+16r)f_{n-1} + (r+2)(3+8r)f_{n-2}\Big] \quad . \quad (4)$$

Second point:

$$y(x_{n+2}) - y(x_n) - 2hy'(x_n) = \frac{h^2}{15r(r+1)(r+2)(2r+1)}\Big[\cdot r(2+3r)(2r+1)f_{n+2} + 8r\left(2+6r+5r^2\right)(r+2)f_{n+1} + (3+10r)(r+1)(r+2)(2r+1)f_n - 8(2r+1)f_{n-1} + (r+2)f_{n-2}\Big]. \quad (5)$$

remain constant for at least two blocks before considered it to be doubled. This step size strategy helps to minimize the choices of the ratio *r*. In the code developed, when the next successful step size is doubled, the ratio *r* is 0.5 and if the next successful step size remain constant, *r* is 1.0. In case of step size failure, *r* is 2.0. In Equation (2) – (5), substituting the ratios of *r* will give the corrector formulae for the two point block direct integration method. For detail see [7].

III. Parallelism Implementation

Within a block in the parallel two point direct block method for two processors (P2PFDIR), it is possible to assign both the predictor and the corrector computations to a single processor and to perform the computations simultaneously in parallel. Each application of the block method generates a collection of approximation to the solution within the block. In a parallel environment individual processor could compute independently the approximation values to the solution within the blocks.

The sequential programs were executed on DYNIX/ptx operating system. The parallel programs of the methods employed were run on a shared memory Sequent Symmetry parallel computer at the Faculty of Computer Science and Information Technology, Universiti Putra Malaysia. The choice of an implementation on a shared memory parallel computer is due to the fact that such a computer can consists of several processors sharing a common memory with fast data access and requiring less communication times, which is suited to the features of the P2PFDIR method.

Below are given the general idea of the parallelism of P2PFDIR in Fig. 2:

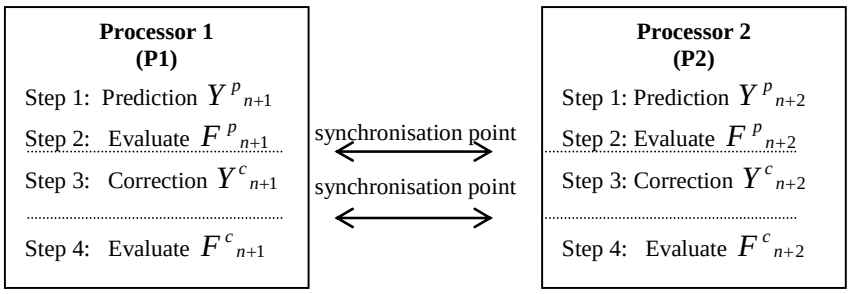

Fig. 2. The parallel process of P2PFDIR

In Equation (2) – (5), this block method was applied in a predictor and corrector mode, and the method is a combination of predictor of order 4 and the corrector of order 5. Each block consists of two steps, i.e *n+1* and *n+2*. The corrector equation values depend on the current blocks *n+1* and *n+2*. The predictor formulae were derived similarly as the corrector formulae and the interpolation points involved are $x_{n-3},\ldots,x_n$. The predictor equations are dependent on values taken from the previous block *n*, *n-1*, *n-2* and *n-3*.

During the implementation of the method, the choices of the next step size will be restricted to half, double or the same as the previous step size and the successful step size will

The predictor and corrector equations at each point are independent of each other. Thus the equation can be easily mapped onto two processors. In the shared memory machine this synchronisation point takes the form of a barrier. For example, all the processors have to exchange information after the evaluation of the terms $F^p{}_{n+1}$ and $F^p{}_{n+2}$ before continue to *Step* 3. The same process happen at *Step* 4 after the evaluation of the terms $F^c{}_{n+1}$ and $F^c{}_{n+2}$. The parallelism is achieved when the code computes at *Step* 3**,** particularly the $Y^c{}_{n+m}$ and $F^c{}_{n+m}$, *m*=1,2. *Step* 1 – 2, *Step* 3 and *Step* 4

can be done concurrently as they are independent to each other. Parallelization in P2PFDIR is achieved by sharing the *f*-evaluations.

The algorithm for P2PFDIR is executed in C language. In order to see a possible speed up of the parallel code, the test problems in Section IV should be expensive. Therefore, the relatively small problems have been enlarge by scaling. The computation cost increased when solving large systems of higher order ODEs because the function evaluations continue to increase. Using two processors to do the work simultaneously can help to reduce the computation time when solving large problem.

IV. Results and Discussion

The following two problems were tested using S2PFDIR and P2PFDIR and compare the sequential and parallel timing for N=1000, 2000 and 4000 for Problem 1 and N=101 when interval [0, 20] and [0, 40] for Problem 2.

Problem 1: (Lagrange equation for the hanging string)

$$y_1'' = K^2(-y_1 + y_2)$$
$$y_2'' = K^2(y_1 - 3y_2 + 2y_3)$$
$$y_3'' = K^2(2y_2 - 5y_3 + 3y_4)$$
$$\vdots$$
$$y_N'' = K^2((N-1)y_{N-1} - (2N-1)y_N)$$

N = number of equations, $0 \le x \le b$, b = end of the interval.

$K = 1$, the initial values $y_i(0) = y_i'(0) = 0$

except $y_{N-2}(0) = y_{N-2}'(0) = 1$,

Source: [9]

Problem 2: (Moon – the second celestial mechanics problem.)

$$x_i'' = \gamma \sum_{j=0, j\neq i}^{N} m_j \frac{(x_j - x_i)}{r_{ij}^3}$$

$$y_i'' = \gamma \sum_{j=0, j\neq i}^{N} m_j \frac{(y_j - y_i)}{r_{ij}^3} \quad \text{where } i = 0,\ldots,N$$

$$r_{ij} = \left((x_i - x_j)^2 + (y_i - y_j)^2\right)^{\frac{1}{2}},\ i, j = 0,\ldots,N$$

$$\gamma = 6.672,\ m_0 = 60,\ m_i = 7\times 10^{-3}\ \ i = 1,\ldots,N$$

Initial data: $x_0(0) = y_0(0) = x_0'(0) = y_0'(0) = 0$

$$x_i(0) = 30\cos\left(\frac{2\pi}{100i}\right) + 400,\ x_i'(0) = 0.8\sin\left(\frac{2\pi}{100i}\right)$$

$$y_i(0) = 30\sin\left(\frac{2\pi}{100i}\right),\ y_i'(0) = -0.8\cos\left(\frac{2\pi}{100i}\right) + 1$$

$N = 101,\ 0 \le t \le b$, b = end of the interval.

Source: [10]

The performance of the sequential and parallel execution times for every problem is shown in Table I– IV while Table V shows the speed-up and efficiency performance for the problem. The notations are defined as follows:

TOL	Tolerances
MTD	Method employed
TS	Total number of steps
FS	Failure steps
FCN	Total function calls
MAXE	Magnitude of the global error $(\max\lvert y_n - y(x_n)\rvert$
TIME	The execution time.
S2PFDIR	Sequential implementation of the two point implicit block method
P2PFDIR	Parallel implementation of the two point implicit block method

In the code, we iterate the corrector to convergence. The convergence test employed were

$$abs\left(y^{(s+1)}_{n+2} - y^{(s)}_{n+2}\right) < 0.1 \times \text{TOL},\quad s = 0,1,2\ldots \qquad (6)$$

where s is the number of iteration. After the successful convergence test of (6), local errors estimated at the point x_{n+2} will be performed to control the error for the block. The error controls were at the second point in the block because in general it had given us better results. The local errors estimates will be obtain by comparing the absolute difference of the corrector formula derived of order k and a similar corrector formula of order k-1.

In these problems we recall that *speedup* is a measure of the relative benefits of parallelising a given application over sequential implementation. The speedup ratio on two processors that we use is defined as $S_2 = \frac{T_0}{T_2}$ where T_0 is the time for the fastest serial algorithm for a given problem and T_2 is the execution time of a parallel program on two processors.

Efficiency of a parallel algorithm is defined as the ratio of speedup compared to the number of processors used. It can be defined as $E_2 = \frac{S_2}{2}\times 100$. In an ideal parallel system, speed-up is equal to the number of processors (P) being used and efficiency is equal to 100%. In practice, speedup is less than P and efficiency is between 0% and 100%, depending on the degree of effectiveness with which the processors are utilised. The speed-up shows the speed gain of the parallel computation and it can describe the increase of performance in the parallel system.

The two problems above were run without exact reference solution in a closed form, so we used the reference solution obtained by the same program using tolerance at two order higher from the current tolerance. The tested problems were run without calculating the maximum error for the execution time of the sequential and parallel execution time. The values maximum errors were computed in a separate program. In Table I – III, without loss of generality, we only compute the MAXE at TOL = 10^{-2} since the execution time is grossly increased with a finer tolerance.

TABLE I
NUMERICAL RESULTS OF 2PFDIR FOR SOLVING PROBLEM 1 WHEN N=1000

TOL	MTD	N=1000, [0, 5]			
		TS	FS	FCN	TIME(min)
10^{-2}	S2PFDIR	239	0	1762	0.195781
	P2PFDIR			883	0.179703
MAXE=1.15083(-2)					
10^{-4}	S2PFDIR	570	1	3384	0.381467
	P2PFDIR			1698	0.354987
10^{-6}	S2PFDIR	714	0	5584	0.609685
	P2PFDIR			2797	0.601066
10^{-8}	S2PFDIR	1743	0	10402	1.167327
	P2PFDIR			5207	1.087472
10^{-10}	S2PFDIR	4298	0	25722	2.882636
	P2PFDIR			12867	2.682821

TABLE II
NUMERICAL RESULTS OF 2PFDIR FOR SOLVING PROBLEM 1 WHEN N=2000

TOL	MTD	N=2000, [0, 5]			
		TS	FS	FCN	TIME(min)
10^{-2}	S2PFDIR	329	0	2300	0.649994
	P2PFDIR			1150	0.436222
MAXE=2.36506(-2)					
10^{-4}	S2PFDIR	796	0	4734	1.360806
	P2PFDIR			2373	0.936770
10^{-6}	S2PFDIR	997	0	7642	2.128587
	P2PFDIR			3801	1.449660
10^{-8}	S2PFDIR	2451	0	14648	4.082667
	P2PFDIR			7330	2.874696
10^{-10}	S2PFDIR	6066	0	36330	10.672470
	P2PFDIR			18171	7.236281

TABLE III
NUMERICAL RESULTS OF 2PFDIR FOR SOLVING PROBLEM 1 WHEN N=4000

TOL	MTD	N=4000, [0, 5]			
		TS	FS	FCN	TIME(min)
10^{-2}	S2PFDIR	457	0	3070	1.278904
	P2PFDIR			1530	0.683906
MAXE=4.23114(-2)					
10^{-4}	S2PFDIR	1116	0	6654	2.812752
	P2PFDIR			3323	1.488229
10^{-6}	S2PFDIR	1397	0	10032	4.127480
	P2PFDIR			5012	2.195468
10^{-8}	S2PFDIR	3459	0	20644	8.778264
	P2PFDIR			10318	4.524878
10^{-10}	S2PFDIR	8566	0	51328	21.953364
	P2PFDIR			25658	11.258135

TABLE IV
NUMERICAL RESULTS OF 2PFDIR FOR SOLVING PROBLEM 2 WHEN N=101, INTERVAL [0, 20]

TOL	MTD	N=101, [0,20]			
		TS	FS	FCN	TIME(sec)
10^{-2}	S2PFDIR	29	0	128	2.079400
	P2PFDIR			70	1.329228
MAXE=3.78992(-4)					
10^{-4}	S2PFDIR	36	0	158	2.542150
	P2PFDIR			85	1.543892
MAXE=2.40610(-6)					
10^{-6}	S2PFDIR	44	0	198	3.175919
	P2PFDIR			105	1.902361
MAXE=8.76138(-7)					
10^{-8}	S2PFDIR	52	0	238	3.808737
	P2PFDIR			125	2.260861
MAXE=4.36732(-9)					
10^{-10}	S2PFDIR	70	0	336	5.360910
	P2PFDIR			174	3.137757
MAXE=6.78177(-11)					

TABLE V
NUMERICAL RESULTS OF 2PFDIR FOR SOLVING PROBLEM 2 WHEN N=101, INTERVAL [0, 40]

TOL	MTD	N=101, [0,40]			
		TS	FS	FCN	TIME(sec)
10^{-2}	S2PFDIR	31	0	136	2.201868
	P2PFDIR			74	1.395546
MAXE=1.77673(-4)					
10^{-4}	S2PFDIR	39	0	176	2.835758
	P2PFDIR			94	1.704369
MAXE=1.53896(-6)					
10^{-6}	S2PFDIR	48	0	224	3.674788
	P2PFDIR			118	2.133841
MAXE=1.39440(-7)					
10^{-8}	S2PFDIR	62	0	296	4.720133
	P2PFDIR			154	2.779889
MAXE=1.29065(-8)					
10^{-10}	S2PFDIR	94	0	478	7.599380
	P2PFDIR			245	4.409488
MAXE=1.54553(-10)					

In Table I - V show the numerical results for the tested problems. For sequential S2PFDIR only one processor was used and two processors were employed for the parallel algorithms of P2PFDIR. The numerical results show that the parallel execution time is faster than the sequential execution time for large ODEs systems.

TABLE VI
SPEED-UP AND EFFICIENCY OF 2PFDIR FOR SOLVING PROBLEM 1 AND 2

TOL	PROB 1			PROB 2	
	N=1000 [0, 5]	N=2000 [0, 5]	N=4000 [0, 5]	N=101 [0, 20]	N=101 [0, 40]
10^{-2}	1.09 [55]	1.49 [75]	1.87 [94]	1.56 [78]	1.58 [79]
10^{-4}	1.07 [54]	1.45 [73]	1.89 [95]	1.65 [83]	1.66 [83]
10^{-6}	1.01 [51]	1.47 [74]	1.88 [94]	1.67 [84]	1.72 [86]
10^{-8}	1.07 [54]	1.42 [71]	1.94 [97]	1.68 [84]	1.70 [85]
10^{-10}	1.07 [54]	1.52 [76]	1.95 [98]	1.71 [86]	1.72 [86]

Note: For each tolerance the values in the square brackets give the results of the efficiency in percentage.

In Table VI, the speed-up ranging between 1.87 and 1.95 for solving Problem 1 when $N = 4000$ and the efficiency is between 94% and 98%. Better speed-up and efficiency can be achieved by increasing the dimension of the ODEs in Problem 1. In Problem 2, the speed up ranging between 1.58 and 1.72 as the interval increased at the same number of equations. The number of function evaluations is almost half in the parallel mode compared to the sequential mode.

In term of accuracy, numerical results are within the given tolerances. The performance of parallel implementation of an integration method depends heavily on the machine, the size of the problem and the costs of the function evaluation. The results suggest that P2PFDIR method be highly recommended for solving large systems of higher order ODEs.

REFERENCES

[1] P.C. Chakravarti, P.B. Worland. A class of self starting methods for the numerical solution of $y'' = f(x, y)$, *BIT* 11, pp 368-383, 1971.

[2] M.B.Suleiman, Solving higher order odes directly by the direct integration method, *Applied Mathematics and Computation* 33, pp 197-219., 1989.
[3] S O.Fatunla, Block methods for second order odes. *Intern. J. Computer Math* 40, pp 55-63, .1990.
[4] Z. Omar, M. Suleiman, Parallel two-point explicit block method for solving high-order ordinary differential equations. *Int. J. of Simulation and Process Modelling.* Vol. 2, No.3/4 pp. 227 - 231, 2006.
[5] Z. Omar and M. Suleiman, Parallel *r*-point implicit block method for solving higher order ordinary differential equations directly, *Journal of ICT*, 3(1), pp 53-66, 2004.
[6] N.H. Cong, K. Strehmel, R. Weiner, and H. Podhaisky, Runge–Kutta-Nystrom-type parallel block predictor-corrector methods. *Advances in Computational Mathematics* 10, pp 115–133, 1999.
[7] Z. A. Majid and M. Suleiman, Two point block direct integration implicit variable steps method for solving higher order systems of ordinary differential equations. *International Conference of Applied and Engineering Mathematics. WCE* (London). *Proceeding of the World Congress on Engineering 2007*, WCE 2007, Volume II, pp 812-815, 2007.
[8] Z. Omar, Developing parallel block methods for solving higher order odes directly, Ph.D. Thesis, University Putra Malaysia, Malaysia, 1999.
[9] Hairer.E., Norsett. S.P. and Wanner. G., *Solving Ordinary Differential Equations I: Nonstiff Problems.* Berlin: Springer-Verlag. pp 26, 1993.
[10] Cong, N.H., Podhaisky, H. and Weiner, R., Performance of explicit pseudo two-step RKN methods on a shared memory computer, 2001. (http://www.mathematik.uni-halle.de/reports/rep-num.html)