International Journal of Engineering, Mathematical and Physical Sciences
ISSN: 2517-9934
Vol:4, No:4, 2010

Parallel Block Backward Differentiation Formulas For Solving Large Systems of Ordinary Differential Equations

Zarina Bibi, I., Khairil Iskandar, O.

Abstract**—**In this paper, parallelism in the solution of Ordinary Differential Equations (ODEs) to increase the computational speed is studied. The focus is the development of parallel algorithm of the two point Block Backward Differentiation Formulas (PBBDF) that can take advantage of the parallel architecture in computer technology. Parallelism is obtained by using Message Passing Interface (MPI). Numerical results are given to validate the efficiency of the PBBDF implementation as compared to the sequential implementation.

Keywords**—**Ordinary differential equations, parallel.

I. INTRODUCTION

WE shall consider parallel Block Backward Differentiation Formulas (PBBDF) for the numerical solution of initial value problems (IVPs) for the first order Ordinary Differential Equations (ODEs) of the form

$$\begin{cases} \dfrac{dy}{dx} = f(x, y), x \in [0, X] \\ y(0) = y_0 \end{cases} \tag{1}$$

where $f : R \times R^m \to R^m$. For all $x \in [0, X]$, $\|f(x, y) - f(x, z)\| \le L\|y - x\|$, L is a Lipschitz constant.

Most of the existing numerical methods for solving (1) are sequential in nature. Various approaches to solve (1) using multiple processor computer system with emphasis on reduction the computation time is due to the recent advances in computer technology. Many researchers develop or modified existing numerical methods to fully utilize the parallel architecture so that some of the computations can be executed simultaneously on multiple processor computer system. Generally, parallelism can be achieved by partitioning the tasks across the methods or across the system of equations. Parallel block methods have been proposed by many researchers to speed up the integration of (1). Some of the earlier works on parallelism on ODEs are found in Gear [3], Bellen and Zennaro [1], Franklin [4], and Chu and Hamilton [2] to name a few. In [3] parallelism is classified as parallellism by partitioning the tasks either "across the method" or "across the systems of equations". [1] introduced parallelism across the time which means that each processor evaluates *f* for different values of *x*. The paper is organized as follows. The PBBDF method is presented in Section II. In Section III, a detailed implementation of the PBBDF method using Message Passing Interface (MPI) is given. Section IV provides numerical result to validate the efficiency of the parallel algorithm of the PBBDF method. The conclusions are given in Section V.

Zarina Bibi, I. is with the Institute of Mathematical Research, Universiti Putra Malaysia, 43400 UPM Serdang, Selangor, MALAYSIA (corresponding author phone: 603-8946-6861; fax: 603-8943-7958; e-mail: zarinabb@science.upm.edu.my).

Khairil Iskandar, O. is with the University Technology MARA, 40450 Shah Alam, Selangor, MALAYSIA. (e-mail: khairil@tmsk.uitm.edu.my).

II. THE BLOCK BACKWARD DIFFERENTIATION FORMULAS

In this section, we reviewed a class of block multistep methods proposed by Ibrahim *et. al* in [6] which is called Block Backward Differentiation Formulas for solving stiff ODEs. The method given in [6] will compute the solutions of Initial Value Problems (IVPs) at two points simultaneously on the *x*-axis ,*i.e.* y_{n+1} and y_{n+2}. The solver start with constant step size which is formulated as

$$\begin{aligned} -\frac{1}{10}y_{n-2} + \frac{3}{5}y_{n-1} - \frac{9}{5}y_n + y_{n+1} + \frac{3}{10}y_{n+2} &= \frac{6}{5}hf_{n+1} \\ \frac{3}{25}y_{n-2} - \frac{16}{25}y_{n-1} + \frac{36}{25}y_n - \frac{48}{25}y_{n+1} + y_{n+2} &= \frac{12}{25}hf_{n+2} \end{aligned} \tag{2}$$

The step size choosing strategy is based on the estimate of the local truncation error (LTE). The step is accepted if the LTE compared with the prescribed tolerance limit, TOL satisfy LTE < TOL and rejected otherwise. Denoting tolerance by ε, the next step size h_{new} is computed by

$$h_{new} = c \times h_{old} \times \left(\varepsilon \left| y_{n+2}^{(p+1)} - y_{n+2}^{(p)} \right|^{-1} \right)^{1/p}$$

where *c* is the safety factor and *p* is the order of the PBBDF method. For our code, we take the safety factor as 0.8.

The next step size is increased by a factor of 1.6 to speed up the computation. The PBBDF solver with the increased step size 1.6*h* is formulated as

$$-\frac{208}{775}y_{n-2}+\frac{6912}{5425}y_{n-1}-\frac{13689}{6200}y_n+y_{n+1}+\frac{351}{1736}y_{n+2}=\frac{117}{124}hf_{n+1}$$
$$\frac{12544}{29875}y_{n-2}-\frac{53248}{29875}y_{n-1}+\frac{74529}{29875}y_n-\frac{2548}{1195}y_{n+1}+y_{n+2}=\frac{546}{1195}hf_{n+2} \quad (3)$$

If the LTE > TOL, the step is rejected, the previous step will be repeated and computed with halved the step size. The PBBDF method when the step size is halved is given by

$$-\frac{3}{128}y_{n-2}+\frac{25}{128}y_{n-1}-\frac{225}{128}y_n+y_{n+1}+\frac{75}{128}y_{n+2}=\frac{15}{8}hf_{n+1}$$
$$\frac{2}{115}y_{n-2}-\frac{3}{23}y_{n-1}+\frac{18}{23}y_n-\frac{192}{115}y_{n+1}+y_{n+2}=\frac{12}{23}hf_{n+2} \quad (4)$$

The formulas given in (2), (3) and (4) are in the similar form of the standard Backward Differentiation Formula (BDF). The clear advantage of PBBDF method is that all the coefficients will be stored with automatic control of the step size for the purpose of optimizing performance in terms of precision and computation time but yet preserving the characteristic of the variable step size. No differentiation coefficients need to be calculated at each step since the coefficients of the *y* values are stored. Furthermore, in the PBBDF method, two solution i.e. y_{n+1} and y_{n+2} values are computed simultaneously. This will lead to a quicker execution time. See [6] for the details of the derivation of the PBBDF and verification of the method.
We rewrite formulas (2), (3) and (4) in the general form

$$\left.\begin{aligned} y_{n+1} &= \theta_1 y_{n+2}+\alpha_1 hf_{n+1}+\psi_1 \\ y_{n+2} &= \theta_2 y_{n+1}+\alpha_2 hf_{n+2}+\psi_2 \end{aligned}\right\} \quad (5)$$

with ψ_1 and ψ_2 are the backvalues.

Equation (5) in matrix-vector form is given by,

$$\left(\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}-\begin{bmatrix}0 & \theta_1\\ \theta_2 & 0\end{bmatrix}\right)\begin{bmatrix}y_{n+1}\\ y_{n+2}\end{bmatrix}=h\begin{bmatrix}\alpha_1 & 0\\ 0 & \alpha_2\end{bmatrix}\begin{bmatrix}f_{n+1}\\ f_{n+2}\end{bmatrix}+\begin{bmatrix}\psi_1\\ \psi_2\end{bmatrix} \quad (6)$$

From (6)

$$(I-A)Y=hBF+\xi$$

where

$$I=\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix},\ Y=\begin{bmatrix}y_{n+1}\\ y_{n+2}\end{bmatrix},\ A=\begin{bmatrix}0 & \theta_1\\ \theta_2 & 0\end{bmatrix},$$
$$B=\begin{bmatrix}\alpha_1 & 0\\ 0 & \alpha_2\end{bmatrix},\ F=\begin{bmatrix}f_{n+1}\\ f_{n+2}\end{bmatrix} \text{ and } \xi=\begin{bmatrix}\psi_1\\ \psi_2\end{bmatrix}$$

Applying Newton iteration to the matrix above by letting

$$\hat{F}=(I-A)Y-hBF-\xi=0$$

Therefore, the Newton-iteration form for the BBDF method is given as

$$Y_{n+1,n+2}^{(i+1)}-Y_{n+1,n+2}^{(i)}=-\left[(I-A)-hB\frac{\partial F}{\partial Y}\left(Y_{n+1,n+2}^{(i)}\right)\right]^{-1}\left[(I-A)Y_{n+1,n+2}^{(i)}-hBF\left(Y_{n+1,n+2}^{(i)}\right)-\xi\right]$$

where $(I-A)-hB\frac{\partial F}{\partial Y}\left(Y_{n+1,n+2}^{(i)}\right)$ is the Jacobian matrix of $\hat{F}$ with respect to *y*. To reduce the amount of computations, the Jacobian matrix is updated when there is a consecutive step failure in the integration *i.e.* LTE > ε. The starting values were computed from the exact solution if available or by using the Euler method.

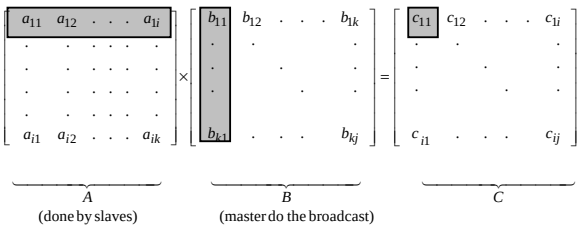
III. PARALELL IMPLEMENTATION OF BBDF

In this section, we discuss the parallel implementation of the BBDF method which allows the distribution of tasks amongst the available processors in order to reduce the execution time. Simultaneous approximations for several steps are obtained using the Message Passing Interface (MPI) library which runs on a High Performance Computer (HPC). These parallel implementations are based on the master – slave approach. The computation occurs only in the slaves while the master broadcast all the data needed by the slaves. The subprogram, JAC1, calculates the Newton-iteration for the PBBDF method. The matrix multiplication is given as

$$\left[hB\frac{\partial F}{\partial Y}\left(Y_{n+1,n+2}^{(i)}\right)\right].$$

One way of performing the multiplication in parallel is to have each processor compute different parts of the product. Consider matrix A_{ik}, B_{kj} and $C_{ij}=\sum_{k=1}^{n_p}A_{ik}\,B_{kj}$.

i) First, the matrix *A* is partition by rows and *B* by columns:



ii) The matrix *A* is striped row-wise among the slaves $P_1, P_2, ..., P_n$ so that each processor is assigned to one row. In order, to avoid any processor been idle, the processor that finish the computation early, will automatically take the next row.
Each process in row *i* will need all the values in column *j*. Therefore, processor P_0, referred as master will broadcast the entire matrix *B* of size $k \times j$ to all the slaves $P_1, P_2, ..., P_n$ as needed prior to the start of the multiplication. Take note that the computation occurs only in the slaves while the master broadcast all the data needed by the slaves.

IV. NUMERICAL RESULTS

Problem 1 : Brusselator systems

Brusselator systems is a nonlinear partial differential equation which arise in the modeling of chemical reaction-diffusion which is of the form

$$\frac{\partial u}{\partial t} = A + u^2 v - (B+1)u + \alpha \frac{\partial^2 u}{\partial x^2}$$
$$\frac{\partial v}{\partial t} = Bu - u^2 v + \alpha \frac{\partial^2 v}{\partial x^2}$$

with $x \in [0,1]$, $\alpha \geq 0$, A and B are the constant parameters. In this paper, we consider A = 1, B = 3, α =1/50 and boundary conditions for u and v which are given by

$$u(0,t) = 1 = u(1,t),\ v(0,t) = 3 = v(1,t)$$
$$u(x,0) = 1 + \sin(2\pi x),\ v(x,0) = 3.$$

By applying the method of lines, we obtain a system of differential equations to be solved on the interval $0 \leq x \leq 10$.

$$\frac{du_i}{dt} = 1 + u_i^2 v_i - 4u_i + \alpha(N+1)^2 (u_{i-1} - 2u_i + u_{i+1})$$
$$\frac{dv_i}{dt} = 3u_i - u_i^2 v_i + \alpha(N+1)^2 (v_{i-1} - 2v_i + v_{i+1}) \quad (7)$$

with $u_0(t) = 1 = u_{N+1}(t), v_0(t) = 3 = v_{N+1}(t),$
$u_i(0) = 1 + \sin(2\pi x_i), v_i(0) = 3,\ x_i = \frac{i}{N+1},\ i = 1,...,N.$

Equation (7) is use to illustrate the performance of the PBBDF method. For more details of this example see Nicolis and Prigogine [7] and Prigogine and Lefever [8].

Problem 2:

$$\begin{bmatrix} y_1' \\ y_2' \\ \vdots \\ y_N' \end{bmatrix} = \begin{bmatrix} -1 & & & & & & \\ 1 & -2 & . & . & . & 0 & \\ & 2 & -3 & & & & \\ & & . & . & & & \\ & & & . & . & & \\ & & & & . & . & \\ & & & & & . & \\ & 0 & & & & & -N+1 \\ & & & & & N-1 & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix},$$

N = number of equations, Initial values: $y(0) = (1,0,...0)^t$
Interval : $0 \leq x \leq 20$
Source: Hull, T.E. *et al*. [5].

The numerical results are performed on Sunfire V1280 HPC. Parameters evaluated are the execution time, speedup and efficiency. The notations used in the tables take the following meaning,

S_p : Speedup
E_p : Efficiency
EQN : Number of equations
TIM E : The execution time in seconds

The speedup of a parallel algorithm is defined as

$$S_p = \frac{T_s}{T_p}$$

where T_s is the execution time of sequential algorithm using one processor and T_p is the executime time by a parallel algorithm using p processors. The efficiency of the parallel algorithm denoted by E_p is defined as the ratio of speedup to the number of processors

$$E_p = \frac{S_p}{p}$$

Theoretically, the value of efficiency is $0 < E_p \leq 1$. Table 1 shows the speedup and efficiency for Problem 1 when run with different number of processors.

TABLE I
2P= 2 processor, 4P=4 processor, 6P= 6 processor
8P= 8 processor

	EQN	2P	4P	6P	8P
S_p	20	0.876	2.151	2.987	3.245
	60	1.146	3.381	5.556	7.434
	100	1.158	3.454	5.712	7.259
E_p	20	0.438	0.538	0.498	0.406
	60	0.573	0.845	0.926	0.929
	100	0.579	0.863	0.952	0.907

Note that the speed up is approaching the linear speedup as the number of equations increased.

Table 2a and 2b shows the speedup and efficiency for Problem 2 when run with different number of processors at difference tolerance.

TABLE IIa: $\varepsilon = 10^{-2}$

	N	NP			
		2	4	6	8
S_p	30	0.976	2.275	3.045	3.262
	50	1.094	3.149	5.052	6.621
	100	1.116	3.303	5.409	6.868
	150	1.124	3.342	5.546	7.460
	200	1.130	3.384	5.604	7.295
	300	0.976	2.275	3.045	3.262
E_p	30	0.488	0.569	0.508	0.408
	50	0.547	0.787	0.842	0.828
	100	0.558	0.826	0.902	0.859
	150	0.562	0.836	0.924	0.933
	200	0.565	0.846	0.934	0.912
	300	0.488	0.569	0.508	0.408

TABLE IIb: $\varepsilon = 10^{-6}$

	N	NP			
		2	4	6	8
Sp	30	0.985	2.296	3.135	3.343
	50	1.095	3.162	5.106	6.257
	100	1.116	3.319	5.425	6.902
	150	1.120	3.343	5.549	7.190
	200	1.130	3.375	5.604	7.004
	300	0.985	2.296	3.135	3.343
E_p	30	0.492	0.574	0.523	0.418
	50	0.547	0.790	0.851	0.782
	100	0.558	0.830	0.904	0.863
	150	0.560	0.836	0.925	0.899
	200	0.565	0.844	0.934	0.876
	300	0.492	0.574	0.523	0.418

V. CONCLUSION

In this paper, we have presented the parallel implementation of the PBBDF method for solving large systems of ordinary differential equations. The parallel implementation of the PBBDF method show significance gains over the sequential implementation. The resulting speed up validates the efficiency of the PBBDF method as the number of equations increased.

ACKNOWLEDGMENT

This research is supported by Institute of Mathematical Research (INSPEM, University Putra Malaysia (UPM) under Fundamental Research Grant Scheme (FRGS).

REFERENCES

[1] Bellen, A. and Zennaro, M. (1989), *Parallel algorithms for initial value problems*, J. Comput. Appl. Math., 25,pp. 341-350.
[2] Chu, M. & Hamilton, H. (1987), *Parallel solution of ODEs by multi-block methods*, SIAM J. Sci. Statist. Comput., 8, pp. 342-353.
[3] Gear, C.W. (1987), *Parallel Methods For Ordinary Differential Equations*, Report No. UIUCDCS-R-87-1369.
[4] Franklin, M., (1978). *Parallel solution of ordinary differential equations*, IEEE Trans. Comput., C-27, pp. 413-420.
[5] Hull, T.E., Enright, W.H., Fellen, B.M. and Sedgwick, A.E. (1972), *Comparing Numerical Methods for Ordinary Differential Equations*. Siam J. Num. Anal. 9(4):603-637.
[6] Ibrahim,Z.B., Suleiman, M.B., Othman, K.I., (2008). *Fixed Coefficients Block Backward Differentiation Formulas for the Numerical Solution of Stiff Ordinary Differential Equations*, European Journal of Scientific Research, Vol. 21 No.3, pp. 508-520.
[7] Nicolis G, Prigogine I. (1977), *Self-organization in non-equilibrium systems*. New York: Wiley-Interscience.
[8] Prigogine I, Lefever R. (1968), *Symmetries breaking instabilities in dissipative systems II*. Journal of Physical Chemistry; 48: 1695-1700.