International Journal of Engineering, Mathematical and Physical Sciences
ISSN: 2517-9934
Vol:2, No:4, 2008

Parallel Block Backward Differentiation Formulas for Solving Ordinary Differential Equations

Khairil Iskandar Othman, Zarina Bibi Ibrahim, and Mohamed Suleiman

Abstract**—**A parallel block method based on Backward Differentiation Formulas (BDF) is developed for the parallel solution of stiff Ordinary Differential Equations (ODEs). Most common methods for solving stiff systems of ODEs are based on implicit formulae and solved using Newton iteration which requires repeated solution of systems of linear equations with coefficient matrix, $I - h\beta J$. Here, J is the Jacobian matrix of the problem. In this paper, the matrix operations is paralleled in order to reduce the cost of the iterations. Numerical results are given to compare the speedup and efficiency of parallel algorithm and that of sequential algorithm.

Keywords**—**Backward Differentiation Formula, block, ordinary differential equations.

I. Introduction

WE consider block method for the parallel solution of Ordinary Differential equations (ODEs)

$$y' = f(x, y) \tag{1}$$

with initial values $y(a) = y_0$ in the interval $x \in [a, b]$.

Various parallel block methods have been proposed for the parallel solution of (1). Watts and Shampine [5], Worland [6], Birta and Abou-Rabia [1], Chu and Hamilton [2] to name a few. Earlier work on parallelism in ODE are found in Rosser [4], Watts [5] and Gear [3]. Gear classifies parallelism into two categories: (i) parallelism across time which is also referred as "parallelism across the method" and (ii) parallelism across the systems. In parallelism across time, each processor executes a different part of the method. Parallelism across the systems where the systems are divided into a set of subsystems and each subsystem is assigned to a different processor.

In the next section, we reviewed a class of block methods proposed by Zarina *et. al* in [7] which are based on Backward Differentiation Formulas (BDF) for solving stiff ODEs. Such methods are called Block Backward Differentiation Formulas (BBDF).

II. The BBDF Method

Traditionally, the BDF computation proceeds to an approximation y_{n+1} of $y(x_{n+1})$ one step at a time while in 2-point BBDF, the approximation solutions y_{n+1} and y_{n+2} are obtained simultaneously in every step. The simultaneous sequence of computation symbolized as *PECE.*

The first point approximation is

$$\rightarrow y^{p}_{n+1}(Predict:P) \rightarrow f^{p}_{n+1}(Evaluate:E) \rightarrow y^{c}_{n+1}(Correct:C) \rightarrow f^{c}_{n+1}(Evaluate:E)$$

and the second point approximation is

$$\rightarrow y^{p}_{n+2}(Predict:P) \rightarrow f^{p}_{n+2}(Evaluate:E) \rightarrow y^{c}_{n+2}(Correct:C) \rightarrow f^{c}_{n+2}(Evaluate:E)$$

In 2 point BBDF method, the interval [*a*,*b*] is divided into series of blocks with each block containing two equally spaced points (see Fig. 1).

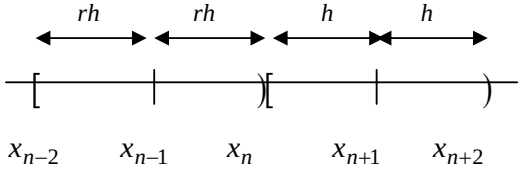


Fig. 1 2-point block method

Let the step size, h of the computed block be $2h$ and the step size of the previous block be $2rh$ where r is the step size ratio. In the step size selection, the decrease of the step size when there is a step failure is limited to halving while the increment of the step size is increased by a factor of 1.6h to ensure zero stability. The BBDF method expressed in the general form is giveb by

$$\left.\begin{aligned} y_{n+1} &= \theta_1 y_{n+2} + \alpha_1 h f_{n+1} + \psi_1 \\ y_{n+2} &= \theta_2 y_{n+1} + \alpha_2 h f_{n+2} + \psi_2 \end{aligned}\right\} \tag{2}$$

with ψ_1 and ψ_2 are the backvalues.

Equation (2) written in matrix-vector form is equivalent to $(I - A)Y_{n+1,n+2} = hBF_{n+1,n+2} + \xi_{n+1,n+2}$ with

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, Y_{n+1,n+2} = \begin{bmatrix} y_{n+1} \\ y_{n+2} \end{bmatrix}, A = \begin{bmatrix} 0 & \theta_1 \\ \theta_2 & 0 \end{bmatrix} B = \begin{bmatrix} \alpha_1 & 0 \\ 0 & \alpha_2 \end{bmatrix}$$

$$F_{n+1,n+2} = \begin{bmatrix} f_{n+1} \\ f_{n+2} \end{bmatrix} \text{ and } \xi_{n+1,n+2} = \begin{bmatrix} \psi_1 \\ \psi_2 \end{bmatrix}.$$

Let $\hat{F}_{n+1,n+2} = (I - A)Y_{n+1,n+2} - hBF_{n+1,n+2} - \xi_{n+1,n+2} = 0$. To approximate this solution, select $Y^{(i)}_{n+1,n+2}$ and generate $Y^{(i+1)}_{n+1,n+2}$ by applying Newton's Iteration to the system (2) to obtain,

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

where $J_{n+1,n+2} = \left(\frac{\partial F}{\partial Y}\right)\left(Y^{(i)}_{n+1,n+2}\right)$ is the Jacobian matrix of F with respect to Y

III. Parallel Implementation of BBDF

This section describes the parallel implementation on the matrix multiplication. Parallelism is obtained using the Message Passing Interface (MPI) library which runs on a High Performance Computer (HPC).

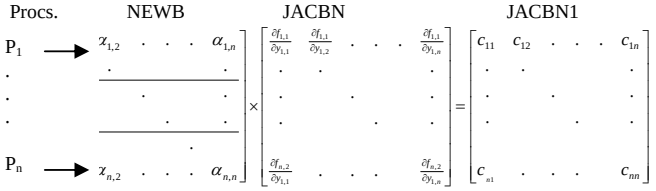
In BBDF code, the matrix multiplication is

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

In order to parallel the matrix multiplication, the matrices involved must be assigned to different processors. This is done by first distributing the matrix JACBN(calculates the Jacobian) to all processors using the command *MPI_BCast(JACBN)*. Next, rows of the matrix can be formed independently and in parallel by partitioning the rows in matrix NEWB to all the processors available. This is done by the formula,

$$\text{range} = \frac{\text{int}(\text{number of rows})}{\text{int}(\text{number of processors})}$$

This will divide evenly the number of rows to the number of processors. If it cannot be divided evenly, the above operation which is an integer division will truncate the value and assigned it to the processors. Any remainder rows will be assigned to the last processor. The master will assign each partition rows of the matrix NEWB to each processor using the command *MPI_Send( )* and *MPI_Recv( )*. The multiplication is done by multiplying the processors to each column of matrix JACBN, and the results are transferred back to master into the matrix JACBN1. This process is done simultaneously. The setting up of the matrix multiplications is done as follows,



IV. Numerical Results

Two test problems are considered. Problem 1 is given to validate the method BBDF works. Problem 2 is a reaction-diffusion problem, the Brusselator system.

Problem 1:

ODEs: $y_1' = -20y_1 - 19y_2$, $y_2' = -19y_1 - 20y_2$

Initial values: $y_1(0) = 2, y_2(0) = 0$

Interval: $0 \le x \le 20$

Solution: $y_1(x) = e^{-39x} + e^{-x}$, $y_2(x) = e^{-39x} - e^{-x}$

Problem 2: Brusselator system

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

where *A*, *B* are constant parameters, $\alpha \ge 0$. We consider parameters $A = 1$, $B = 3$, $\alpha = 0.02$

ODEs:
$$u_i' = 1 + u_i^2 v_i - 4u_i + \alpha(N+1)^2(u_{i-1} - 2u_i + u_{i+1})$$
$$v_i' = 3u_i - u_i^2 v_i + \alpha(N+1)^2(v_{i-1} - 2v_i + v_{i+1})$$

where
$$u_0(t) = 1 = u_{N+1}(t)$$
$$v_0(t) = 3 = v_{N+1}(t)$$
$$u_i(0) = 1 + \sin(2\pi x_i)$$
$$v_i(0) = 3$$

with $x_i = \frac{i}{N+1}$, $i = 1,...,N$ and is solved on the time interval $0 \le x \le 10$.

The notations used in the tables take the following meaning:

TOL	:	Tolerance used
TS	:	Total steps used
FA	:	Total number of rejected steps
IST	:	Total number of accepted steps
MAXE	:	Magnitude of the maximum error of the computed solution
BDF	:	Backward Differentiation Formulas
BBDF	:	Block Backward Differentiation Formulas
S_p	:	Speedup
E_p	:	Efficiency
EQN	:	Number of equations
TIME	:	The execution time in seconds

The numerical results are tabulated in Table I and II.

TABLE I
NUMERICAL RESULT FOR PROBLEM 1

TOL	MTD	FA	IST	TS	MAXE	TIME
10^{-2}	BDF	11	60	71	1.9588e-01	19064
	BBDF	0	32	32	2.7778e-04	6083
10^{-4}	BDF	21	120	141	5.5911e-03	25568
	BBDF	2	63	65	3.5710e-06	9866
10^{-6}	BDF	26	197	223	3.2120e-05	34418
	BBDF	0	170	170	2.6545e-08	19231

Tables II shows the speedup and efficiency for the Brusellator problem when run with different number of processors.

TABLE II
NUMERICAL RESULT FOR PROBLEM 2

	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

1P= 1 processor, 2P= 2 processor, 4P=4 processor,
6P= 6 processor, 8P= 8 processor.

V. CONCLUSION

The numerical results showed that the speedup improves as the problem size increases. In fact, the speed up is approaching the linear speedup as the number of equations increased. Therefore, the parallel implementation of the BBDF methods shows significance gains over the sequential BDF.

ACKNOWLEDGEMENT

This research was supported by Universiti Technology MARA under Fundamental Research Grant Scheme (FRGS).

REFERENCES

[1] Birta, L.G. and Abou-Rabia, O. (1987), *Parallel Block Predictor Corrector Methods for ODEs*, IEEE Transactions on Computers, c-36(3):299-311.
[2] Chu, M.T. and Hamilton, H. (1987), *Parallel solution of ODE's by multi-block methods*,SIAM J. Sci. Stat. Comput. 8: 342-353.
[3] Gear, C.W. (1971), *Numerical Initial Value Problems in Ordinary Differential Equations*, New Jersey: Prentice Hall, Inc.
[4] Rosser, J.B. 1967. A Runge-Kutta for All Seasons. *Siam Review* 9(3): 417-452.
[5] Watts, H.A. and Shampine, L.F. 1972. A-Stable Block Implicit One-Step Methods. *BIT* 12:252-266.
[6] Worland, P.B., (1976). *Parallel Methods for the Numerical Solution of Ordinary Differentials Equations*, IEEE Transactions on Computers C-25:1045-1048.
[7] Zarina Bibi, I., Khairil Iskandar, O.,Suleiman, M., 2007. *Variable Stepsize Block Backward Differentiation Formula For Solving Stiff ODEs*, Proceedings of World Congress on Engineering 2007, London, U.K. Vol II: pg 785-789. ISBN: 978-988-98671-2-6.