PARALLEL EXECUTION OF BLOCK RUNGE-KUTTA METHODS FOR SOLVING ORDINARY DIFFERENTIAL EQUATIONS

Fudziah Ismail [1], Zailan Siri [2], Mohamad Othman [3] and Mohamed Suleiman [1]

[1]Department of Mathematics, University Putra Malaysia,
43400 UPM Serdang, Selangor, Malaysia

[2]Institute of Mathematical Sciences, University of Malaya,
50603 Lembah Pantai, Kuala Lumpur, Malaysia

[3] Department of Communication Technology and Network
University Putra Malaysia, 43400 UPM Serdang, Selangor, Malaysia

ABSTRACT

The objective of this paper is to exploit the favourable characteristics of block explicit Runge-Kutta and block diagonally implicit Runge-Kutta methods for sequential machines to parallel ones. Both methods are used to solve ordinary differential equations, codes based on the methods are execute in sequential and parallel. Numerical results based on the two modes of executions are tabulated and compared.

Keywords: Block Explicit Runge-Kutta , Block Diagonally Implicit Runge-Kutta, sequential, parallel.

Makalah diterima 1 Juli 2005

1. INTRODUCTION

Parallelism in ODE (ordinary differential equation) software can be divided into three categories: in coding the method so that it can be executed simultaneously on several processors, in splitting variables in a multivariable ODE system between processors and lastly in exploiting parallelism in solving the algebraic system involved. This paper focuses on the parallel execution of the method.

Work on parallel Runge-Kutta methods for solving first order ODEs have been proposed by a number of researchers as can be seen in [1 - 4]. Iserles and Norsett [5] proposed diagonally implicit Runge-Kutta method which is designed specifically for parallel execution. Cash [6,7] derived explicit and diagonally implicit block Runge-Kutta method which can be exploited for the purpose of parallel implementation. We hope by parallelizing the algorithms a more effective codes can be developed.

2. BLOCK EXPLICIT RUNGE-KUTTA METHODS

Cash [6] derived a family of block explicit Runge-Kutta (BERK) methods of order two. At the first point (x_{n+1}) the formula is given by

$$\begin{array}{c|cc} 0 & 0 & \\ 1 & 1 & 0 \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array} \qquad (1)$$

And at the second point (x_{n+2}) after normalizing the method in (1) and adding one more step, the formula is given by

$$\begin{array}{c|cc} 0 & 0 & \\ 1 & 1 & 0 \\ \hline & 0 & 2 \end{array} \qquad (2)$$

where

$$k_1 = f(x_n, y_n),$$

$$k_2 = f(x_{n+1}, y_n + hk_1)$$

$$y_{n+1}^{(1)} = y_n + hk_1,$$

$$y_{n+1}^{(2)} = y_n + \tfrac{h}{2}(k_1 + k_2)$$

$$y_{n+2}^{(1)} = y_n + h(k_1 + k_2)$$

$$y_{n+2}^{(2)} = y_n + 2hk_2$$

$k^{(m)}$ denotes the *m*th iteration of *k*. Formula (1) and (2) produces second order approximations at both x_{n+1} and x_{n+2} and estimate of the local truncation error (LTE) in $y_{n+j}^{(1)}$ is

$$y_{n+j}^{(2)} - y_{n+j}^{(1)}$$

for $j = 1, 2$.

To investigate parallelism in (1) and (2), we produce a digraph in Figure 1.

From Figure 1, on S2; it can be seen that all $y_{n+j}^{(m)}$ for $j,m = 1, 2$ are independent of each other but not for k_1 and k_2 on S1. Meaning, it is possible to calculate $y_{n+j}^{(m)}$

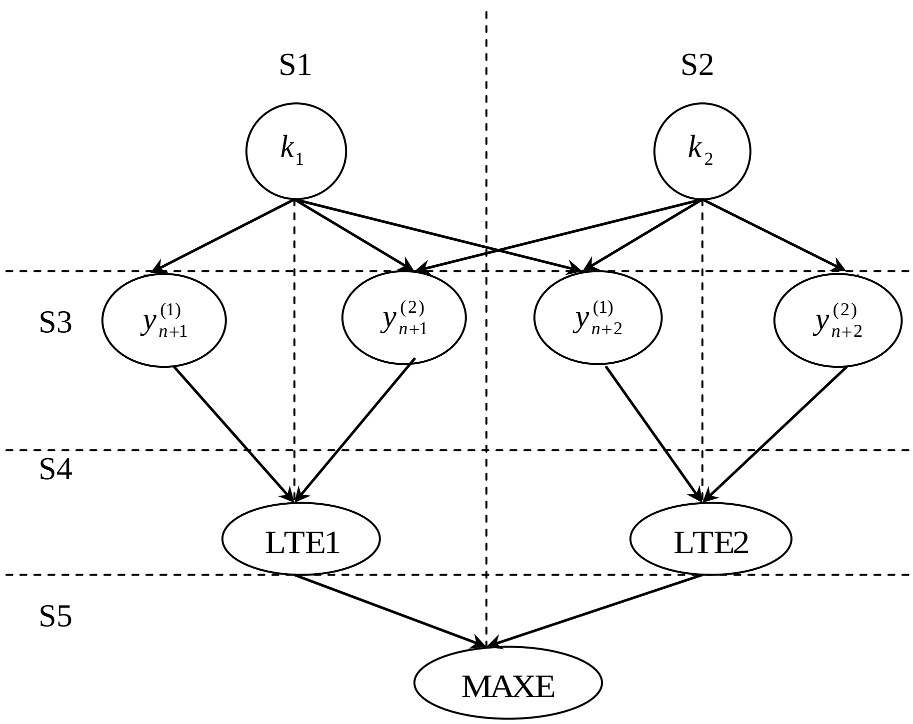

Figure 1. Illustration of Second Order BERK Methods on Parallel Machine

for m, j = 1, 2 in parallel with four processors after we compute k_1 and k_2. On S4, calculate both LTE in parallel using two processors and then find the maximum error of the LTE.

Another second order BERK given in Cash [6] is as follows:
At the point (x_{n+1}), the formula is given by

$$\begin{array}{c|cc} 0 & 0 & \\ 1 & 1 & 0 \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

And at (x_{n+2}) the formulae is given by

$$\begin{array}{c|ccc} 0 & 0 & & \\ 1 & 1 & 0 & \\ 2 & 1 & 1 & 0 \\ \hline & \frac{1}{2} & 1 & \frac{1}{2} \end{array}$$

where
$$k_1 = f(x_n, y_n)$$
$$k_2 = f(x_{n+1}, y_n + hk_1)$$
$$k_3 = f(x_{n+2}, y_n + h(k_1 + k_2))$$
$$y_{n+1}^{(1)} = y_n + hk_1$$
$$y_{n+1}^{(2)} = y_n + \tfrac{h}{2}(k_1 + k_2)$$
$$y_{n+2}^{(1)} = y_n + h(k_1 + k_2)$$
$$y_{n+2}^{(2)} = y_n + \tfrac{h}{2}(k_1 + 2k_2 + k_3)$$

The following diagraph is shown to make it easier to visualize the parallelism in this method.

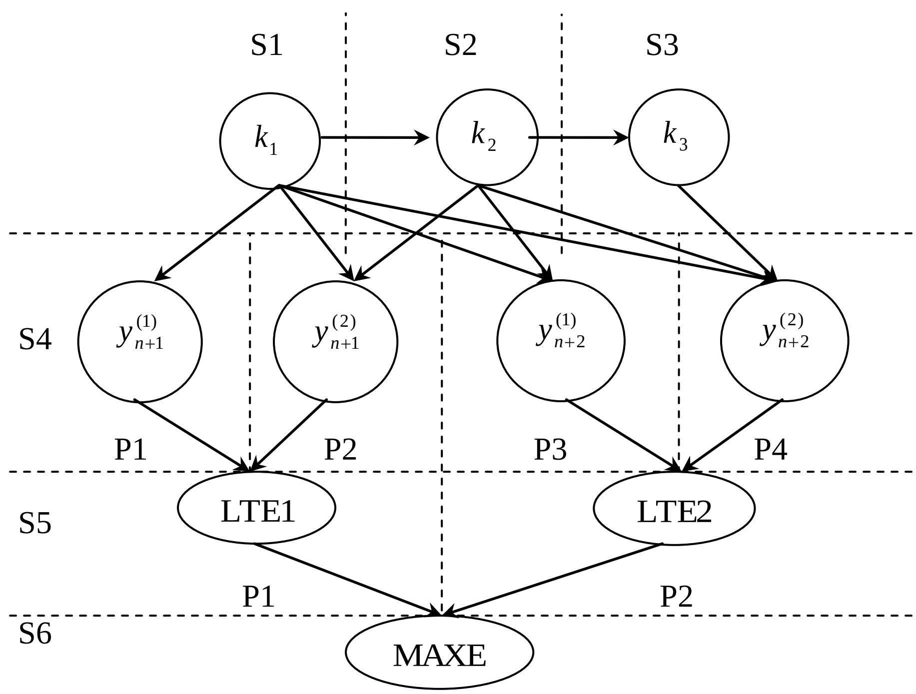

Figure 2. Illustration of Second Order BERK Methods on Parallel Machine

This method is similar to the previous method; parallelism arises only on $y_{n+j}^{(m)}; m.j = 1,2$ because they are independent of each other. In this method, calculate

k_1 first followed by k_2 and then k_3, after k_i; $i = 1,2,3$ have been computed, calculate $y_{n+j}^{(m)}; i.j = 1,2$ simultaneously using four processors. The following is the parallel algorithms for second order BERK methods.

Step 1:
Sequentially compute $h = \dfrac{x_n - x_0}{n}$ is the step-size of the method, k_1 and k_2 on P1.

Step 2:
Calculate $y_{n+1}^{(1)}$, $y_{n+1}^{(2)}$, $y_{n+2}^{(1)}$ and $y_{n+1}^{(2)}$ on P1, P2, P3 and P4 respectively.

Step 3:
By using two processors; calculate LTE1 and LTE2 in parallel on P1 and P2. Then, find the maximum error of these two LTEs.

Step 4:
Repeat Step 1- Step 3 until the end of the integration interval.

3. PARALLELISM IN BDIRK METHODS

In this section, the execution of block diagonally implicit Runge-Kutta (BDIRK) methods in Cash [7] on parallel computer will be presented. The method is given by the following tableau

$$
\begin{array}{c|ccccc}
1 & 1 & & & & \\
2 & 1 & 1 & & & \\
1 & \frac{1}{2} & -\frac{1}{2} & 1 & & \\
2 & 1 & -1 & 1 & 1 & \\
3 & \frac{3}{2} & -\frac{3}{2} & 1 & 1 & 1 \\
\hline
 & \frac{3}{2} & -\frac{3}{2} & 1 & 1 & 1
\end{array}
\qquad (3)
$$

Sequentially the method can be implemented as follows:

At x_{n+1} we have : $y_{n+1}^{(1)} = y_n + hk_1$

At x_{n+2} we have : $y_{n+2}^{(1)} = y + h(k_1 + k_2)$

At x_{n+1} we have :

$$y_{n+1}^{(2} = y_n + \frac{h}{2}\left(k_1 - k_2 + k_3\right)$$

At x_{n+2} we have :

$$y_{n+2}^{(2)} = y + h(k_1 - k_2 + k_3 + k_4)$$

And at x_{n+3} :

$$y_{n+3} = y_n + h(\frac{3}{2}k_1 - \frac{3}{2}k_2 + k_3 + k_4 + k_5)$$

BDIRK method with Butcher array as in (3) provides second order solution at x_{n+3} and x_{n+2} and first order solution at x_{n+1}.

The digraph of this method is given below

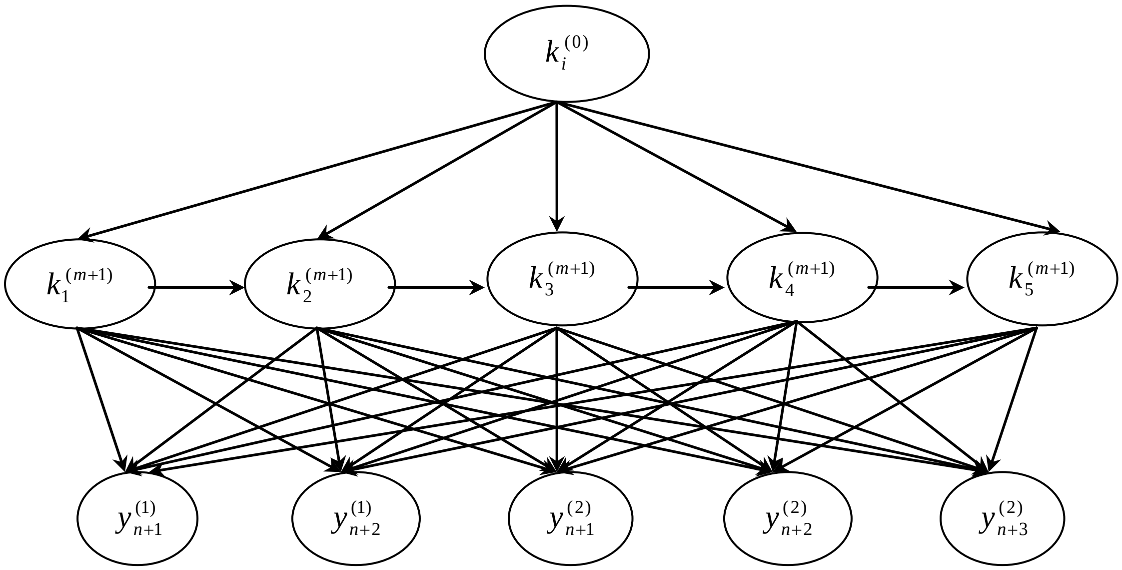

Figure 3. Illustration of BDIRK Methods on Parallel Machine

The digraph of second-order BDIRK method above, clearly showed that every $y_{n+i}^{(1)}$ and $y_{n+j}^{(2)}$ for i = 1, 2; j = 1, 2, 3 are independent of each other. So, we can calculate them simultaneously using five processors.

4. NUMERICAL RESULTS

Before presenting the numerical results, let us introduce the metric for measuring the performance of parallel programs:

1. The number of processors, p used.
2. Parallel time, t_p that is the time period elapsed between the beginning of the first processor and the end of the last processor during the execution of the algorithm.
3. Speed-up, S_p compares the parallel running time, t_p of an algorithm that uses p processors to solve a particular problem, to the sequential running time, t_s of an algorithm for the same problem, it is given by:

$$S_p = \frac{t_s}{t_p} .$$

Or it can be defined as the ratio of the execution time of the parallel algorithm on a single processor and the execution time of the parallel algorithm on p processors, that is:

$$S_p = \frac{t_{p=1}}{t_p} .$$

4. $$E_p = \frac{S}{p} = \frac{t_s}{pt_p} = \frac{t_{p=1}}{pt_p}$$

E_p is the efficiency of the parallel algorithm and it must be less or equal to one ($E_p \le 1$). If E_p = 1, the speed-up is said to be perfect. Perfect speed-up is rarely ever achievable and it can be multiplied by 100 to get the percentage.

5. $T = \frac{1}{t_p}$. Temporal Performance of the method

Given below are the test problems used, they are solved using BERK and BDIRK methods and the programs are run on Sequent 30 which is available at University Putra Malaysia for various values of step-size.

Problem 1:

$$y' = \frac{1}{4} y(1 - \frac{1}{20} y)$$
$$y(0) = 1$$
$$0 \le t \le 5$$

Exact solution:

$$y(t) = \frac{20}{1 + 19e^{-\frac{1}{4}t}}$$

Problem 2:

$$y' = y + 2te^t$$
$$y(0) = 1$$
$$0 \le t \le 1$$

Exact solution:

$$y(t) = (t^2 + 1)e^t$$

Problem 3:

$$y' = te^{2t} + 2y$$
$$y(0) = 0$$
$$0 \le t \le 1$$

Exact solution:

$$y(t) = \frac{1}{2} t^2 e^{2t}$$

Problem 4:

$$y' = t\cos(t)$$
$$y(0) = 1$$
$$0 \le t \le 1$$

Exact solution:

$$y(t) = \cos(t) + t\sin(t)$$

Problem 5:

$$y' = t^2 - y$$
$$y(0) = 1$$
$$0 \le t \le 0.4$$

Exact solution:

$$y(t) = 2 - e^{-t} - 2t + t^2$$

Numerical results obtained are given in Tables 2 - 11 and the notations used are as follows:

Table 1. Notations are used in the Numerical Results Tables

Notation	Description
BERK1	BERK method for Butcher array (2.1)
BERK2	BERK method for Butcher array (2.2)
BDIRK	BDIRK method for Butcher array (4.1)
h	Step-size used
METHOD	Method employed
t_{seq}	The execution sequential time (in microseconds)
t_{par}	The execution parallel time (in microseconds)
MAXE	Magnitude of the maximum error of the computed solution
S	Speed-up of the method
E	Efficiency of the method
C	Cost of the method
T	Temporal Performance of the method

Table 2. Numerical Results for Problem 1

h	METHOD	t_{seq}	t_{par}	MAXE
1.0×10^{-1}	BERK1	3212	3048	7.44555×10^{-4}
	BERK2	3304	3078	7.50849×10^{-4}
	BDIRK	4888	3892	6.54178×10^{-2}
1.0×10^{-2}	BERK1	33359	31222	7.68991×10^{-6}
	BERK2	34081	31960	7.69632×10^{-6}
	BDIRK	51292	36932	6.74254×10^{-3}
1.0×10^{-3}	BERK1	332486	310530	7.71452×10^{-8}
	BERK2	357018	318595	7.71516×10^{-8}
	BDIRK	377545	366810	6.76280×10^{-4}
1.0×10^{-4}	BERK1	4151072	2975945	7.71698×10^{-10}
	BERK2	4304168	3038146	7.71705×10^{-10}
	BDIRK	3785277	3655726	6.76483×10^{-5}
1.0×10^{-5}	BERK1	4651527	2922069	7.71700×10^{-12}
	BERK2	49952251	30380257	7.71700×10^{-12}
	BDIRK	44187934	38357113	6.76503×10^{-6}

Table 3. Numerical Results for Problem 2

h	METHOD	t_{seq}	t_{par}	MAXE
1.0×10^{-1}	BERK1	3215	3122	2.68265×10^{-1}
	BERK2	3431	3163	2.91612×10^{-1}
	BDIRK	4396	3956	6.45222×10^{0}
1.0×10^{-2}	BERK1	33843	30886	3.10373×10^{-3}
	BERK2	34733	30943	3.12764×10^{-3}
	BDIRK	39476	37973	6.51413×10^{-1}
1.0×10^{-3}	BERK1	336528	310980	3.14617×10^{-5}
	BERK2	354034	311464	3.14856×10^{-5}
	BDIRK	396133	379053	6.51290×10^{-2}
1.0×10^{-4}	BERK1	3338592	2548952	3.15040×10^{-7}
	BERK2	3454525	2644023	3.15064×10^{-7}
	BDIRK	3901977	3795640	6.51274×10^{-3}
1.0×10^{-5}	BERK1	34736209	25689459	3.15083×10^{-9}
	BERK2	36534227	26949736	3.15085×10^{-9}
	BDIRK	40482528	39443653	6.51274×10^{-4}

Table 4. Numerical Results for Problem 3

h	METHOD	t_{seq}	t_{par}	MAXE
1.0×10^{-1}	BERK1	3358	2964	7.51716×10^{-1}
	BERK2	3480	2995	8.89355×10^{-1}
	BDIRK	4446	4161	2.39502×10^{1}
1.0×10^{-2}	BERK1	37934	31214	9.93971×10^{-3}
	BERK2	39340	32423	1.00743×10^{-2}
	BDIRK	39560	38434	2.22956×10^{0}
1.0×10^{-3}	BERK1	415294	310561	1.01573×10^{-4}
	BERK2	429194	310738	1.01705×10^{-4}
	BDIRK	387679	381531	2.17819×10^{-1}
1.0×10^{-4}	BERK1	4060980	3006762	1.01784×10^{-6}
	BERK2	4096098	3178707	1.01797×10^{-6}
	BDIRK	3887938	3824693	2.17290×10^{-2}
1.0×10^{-5}	BERK1	39278591	28757755	1.01805×10^{-8}
	BERK2	41553138	29857012	1.01806×10^{-8}
	BDIRK	38907232	37979786	2.17236×10^{-3}

Table 5. Numerical Results for Problem 4

h	METHOD	t_{seq}	t_{par}	MAXE
1.0×10^{-1}	BERK1	3231	3015	1.91467×10^{-3}
	BERK2	3379	3094	3.02466×10^{-3}
	BDIRK	4338	3862	8.37090×10^{-2}
1.0×10^{-2}	BERK1	33549	30699	2.90066×10^{-5}
	BERK2	35358	31099	3.01182×10^{-5}
	BDIRK	45123	35472	8.41596×10^{-3}
1.0×10^{-3}	BERK1	333609	317679	3.00057×10^{-7}
	BERK2	376714	329301	3.01169×10^{-7}
	BDIRK	449193	348015	8.41644×10^{-4}
1.0×10^{-4}	BERK1	3592339	2607253	3.01058×10^{-9}
	BERK2	3617280	2618477	3.01169×10^{-9}
	BDIRK	4375188	3469182	8.41645×10^{-5}
1.0×10^{-5}	BERK1	3375981	2122692	3.01160×10^{-11}
	BERK2	36370391	23107885	3.01170×10^{-11}
	BDIRK	48596000	36909709	8.41645×10^{-6}

Table 6. Numerical Results for Problem 5

h	METHOD	t_{seq}	t_{par}	MAXE
1.0×10^{-1}	BERK1	3274	3024	1.83728×10^{-3}
	BERK2	3368	3060	1.86484×10^{-3}
	BDIRK	4034	3617	5.66906×10^{-2}
1.0×10^{-2}	BERK1	33431	31143	1.80315×10^{-5}
	BERK2	33707	31223	1.80594×10^{-5}
	BDIRK	35205	34617	5.96789×10^{-3}
1.0×10^{-3}	BERK1	480013	308860	1.79983×10^{-7}
	BERK2	487536	309071	1.80011×10^{-7}
	BDIRK	355176	345497	5.99680×10^{-4}
1.0×10^{-4}	BERK1	5845446	3455051	1.79950×10^{-9}
	BERK2	5953279	3564413	1.79953×10^{-9}
	BDIRK	3559931	3461494	5.99968×10^{-5}
1.0×10^{-5}	BERK1	65757617	36441696	7.71698×10^{-10}
	BERK2	66702946	37187850	1.79950×10^{-11}
	BDIRK	44137409	35645602	5.99997×10^{-6}

Table 7. Results on the efficiency of the methods for Problem 1

h	METHOD	$S = \frac{t_{seq}}{t_{par}}$	$E = \frac{S}{p}$	$C = pt_{par}$	$T = \frac{1}{t_{par}}$
1.0×10^{-1}	BERK1	1.05381	0.52690	6096	3.28084×10^{-4}
	BERK2	1.07342	0.53671	6156	3.24886×10^{-4}
	BDIRK	1.25591	0.25118	19460	2.56937×10^{-4}
1.0×10^{-2}	BERK1	1.06845	0.53422	62444	3.20287×10^{-5}
	BERK2	1.06636	0.53318	63920	3.12891×10^{-5}
	BDIRK	1.38882	0.27776	184660	2.70768×10^{-5}
1.0×10^{-3}	BERK1	1.07070	0.53535	621060	3.22030×10^{-6}
	BERK2	1.12060	0.56030	637190	3.13878×10^{-6}
	BDIRK	1.02927	0.20585	1834050	2.72621×10^{-6}
1.0×10^{-4}	BERK1	1.39488	0.69744	5951890	3.36028×10^{-7}
	BERK2	1.41671	0.70835	6076292	3.29148×10^{-7}
	BDIRK	1.03544	0.20709	18278630	2.73543×10^{-7}
1.0×10^{-5}	BERK1	1.59186	0.79593	5844138	3.42223×10^{-7}
	BERK2	1.64423	0.82212	60760514	3.29161×10^{-8}
	BDIRK	1.23823	0.24765	178228010	2.80540×10^{-8}

Table 8. Results on the efficiency of the methods for Problem 2

h	METHOD	$S = \frac{t_{seq}}{t_{par}}$	$E = \frac{S}{p}$	$C = pt_{par}$	$T = \frac{1}{t_{par}}$
1.0×10^{-1}	BERK1	1.02979	0.51489	6244	3.20307×10^{-4}
	BERK2	1.08473	0.54236	6326	3.16156×10^{-4}
	BDIRK	1.11122	0.22224	19780	2.52781×10^{-4}
1.0×10^{-2}	BERK1	1.09574	0.54787	61772	3.23771×10^{-5}
	BERK2	1.12248	0.56124	61886	3.23175×10^{-5}
	BDIRK	1.03958	0.20792	189865	2.63345×10^{-5}
1.0×10^{-3}	BERK1	1.08215	0.54108	621960	3.21564×10^{-6}
	BERK2	1.13668	0.56834	622928	3.21064×10^{-6}
	BDIRK	1.04506	0.20901	1895265	2.63815×10^{-6}
1.0×10^{-4}	BERK1	1.30979	0.65490	5097904	3.92318×10^{-7}
	BERK2	1.30654	0.65327	5288046	3.78212×10^{-7}
	BDIRK	1.02802	0.20560	18978200	2.63460×10^{-7}
1.0×10^{-5}	BERK1	1.35216	0.67608	51378918	3.89265×10^{-8}
	BERK2	1.35564	0.67782	53899472	3.71061×10^{-8}
	BDIRK	1.02634	0.20527	197218265	2.53526×10^{-8}

Table 9. Result on the efficiency of the methods for Problem 3

h	METHOD	$S = \frac{t_{seq}}{t_{par}}$	$E = \frac{S}{p}$	$C = pt_{par}$	$T = \frac{1}{t_{par}}$
1.0×10^{-1}	BERK1	1.13293	0.56646	5928	3.37382×10^{-4}
	BERK2	1.16194	0.58097	5990	3.33890×10^{-4}
	BDIRK	1.06849	0.21370	20805	2.40327×10^{-4}
1.0×10^{-2}	BERK1	1.21529	0.60764	62428	3.20369×10^{-5}
	BERK2	1.21334	0.60667	64846	3.08423×10^{-5}
	BDIRK	1.02930	0.20586	192170	2.60186×10^{-5}
1.0×10^{-3}	BERK1	1.33724	0.66862	621122	3.21998×10^{-6}
	BERK2	1.38121	0.69060	621476	3.21815×10^{-6}
	BDIRK	1.01611	0.20322	1907655	2.62102×10^{-6}
1.0×10^{-4}	BERK1	1.35062	0.67531	6013524	3.32584×10^{-7}
	BERK2	1.28860	0.64430	6357414	3.14593×10^{-7}
	BDIRK	1.01654	0.20331	19123465	2.61459×10^{-7}
1.0×10^{-5}	BERK1	1.36584	0.68292	57515510	3.47732×10^{-8}
	BERK2	1.39174	0.69587	59714024	3.34930×10^{-8}
	BDIRK	1.02442	0.20488	189898930	2.63298×10^{-8}

Table 10. Result on the efficiency of the methods for Problem 4

h	METHOD	$S=\frac{t_{seq}}{t_{par}}$	$E=\frac{S}{p}$	$C=pt_{par}$	$T=\frac{1}{t_{par}}$
1.0×10^{-1}	BERK1	1.07164	0.53582	6030	3.31675×10^{-4}
	BERK2	1.09211	0.54606	6188	3.23206×10^{-4}
	BDIRK	1.12325	0.22465	19310	2.58933×10^{-4}
1.0×10^{-2}	BERK1	1.09284	0.54642	61398	3.25744×10^{-5}
	BERK2	1.13695	0.56847	62198	3.21554×10^{-5}
	BDIRK	1.27207	0.25441	177360	2.81912×10^{-5}
1.0×10^{-3}	BERK1	1.05014	0.52507	635358	3.14783×10^{-6}
	BERK2	1.14398	0.57199	658602	3.03674×10^{-6}
	BDIRK	1.29073	0.25815	1740075	2.87344×10^{-6}
1.0×10^{-4}	BERK1	1.37783	0.68891	5214506	3.83545×10^{-7}
	BERK2	1.38144	0.69072	5236954	3.81901×10^{-7}
	BDIRK	1.26116	0.25223	17345910	2.88252×10^{-7}
1.0×10^{-5}	BERK1	1.59042	0.79521	4245384	4.71100×10^{-7}
	BERK2	1.57394	0.78697	46215770	4.32753×10^{-8}
	BDIRK	1.31662	0.26332	184548545	2.70931×10^{-8}

Table 11. Result on the efficiency of the methods for Problem 5

h	METHOD	$S=\frac{t_{seq}}{t_{par}}$	$E=\frac{S}{p}$	$C=pt_{par}$	$T=\frac{1}{t_{par}}$
1.0×10^{-1}	BERK1	1.08267	0.54134	6048	3.30688×10^{-4}
	BERK2	1.10065	0.55033	6120	3.26797×10^{-4}
	BDIRK	1.11529	0.22306	18085	2.76472×10^{-4}
1.0×10^{-2}	BERK1	1.07347	0.53673	62286	3.21099×10^{-5}
	BERK2	1.07956	0.53978	62446	3.20277×10^{-5}
	BDIRK	1.01699	0.20340	173085	2.88875×10^{-5}
1.0×10^{-3}	BERK1	1.55414	0.77707	617720	3.23771×10^{-6}
	BERK2	1.57742	0.78871	618142	3.23550×10^{-6}
	BDIRK	1.02801	0.20560	1727485	2.89438×10^{-6}
1.0×10^{-4}	BERK1	1.69186	0.84593	6910102	2.89431×10^{-7}
	BERK2	1.67020	0.83510	7128826	2.80551×10^{-7}
	BDIRK	1.02844	0.20569	17307470	2.88893×10^{-7}
1.0×10^{-5}	BERK1	1.80446	0.90223	72883392	2.74411×10^{-8}
	BERK2	1.79368	0.89684	74375700	2.68905×10^{-8}
	BDIRK	1.23823	0.24765	178228010	2.80540×10^{-8}

5. CONCLUSION

From the results we observed that

1. Parallel executions of all the methods performed better in terms of execution time

compared to their sequential counterparts. This is more obvious when the stepsize is smaller.

2. Comparing BERK and BDIRK method on parallel machines; we observed that BERK method performed better in terms of speed up, efficiency, cost and temporal performance compared to BDIRK. BDIRK method gives less than 30% efficiency compared to 60% efficiency in BERK method. For all the methods the efficiency increases as the stepsizes decreases. It is noted too that as the efficiency increases the speed up also increases, the cost decreases and the temporal performance increases. The reason why BERK method perform better is that in BDIRK method there are iterations on the k_i which have to be performed sequentially and this consumed a lot of time.
3. It is also observed that in BERK method BERK2 method performed slightly better compared to BERK1 method, this is expected because in BERK2 method the values of y 's at x_{n+1}, x_{n+2} and at x_{n+3} can be computed in parallel compared to only values of y 's at x_{n+1}, x_{n+2} in BERK1 method.

As a conclusion, before any assumption is made, more experiment should be carried out, such as test problems which include bigger systems of equations, so that the superiority of the parallel execution as well as the method is more obvious.

REFERENCES

[1] Gear, C. W. and Xu, X., 1993. Parallelism Across Time in ODEs. *Applied Numerical Mathematics* **11,** 45-68.

[2] Amodio, P. and Trigianete, D., 1993. A Parallel Direct Method for Solving IVP for ODEs. *Applied Numerical Mathematics* **11,** 85 - 93.

[3] Jackson, K. R. and Nørsett, S. P., 1990. *The Potential for Parallelism in Runge-Kutta Methods, Part I: RK Formulas in Standard Form.* Natural Science and Engineering Research Council of Canada, The Information Technology Research Centre of Ontario and The Norwegian Research Council for Science and Humanities.

[4] Burrage, K., 1993. Parallel Methods for Initial Value Problems. *Applied Numerical Mathematics* **11,** 5 - 25.

[5] Iserles, A. and Nørsett, S. P., 1990. On the Theory of Parallel Runge-Kutta Methods. *A Journal of Numerical Analysis* **10,** 463-488.

[6] Cash, J. R., 1983. Block Runge-Kutta Methods for the Numerical Integration of Initial Value Problems in Ordinary Differential Equations, Part I: The Nonstiff Case. *Mathematics of Computation,* **40(161),** 175-191.

[7] Cash, J. R., 1985. Block embedded explicit Runge-Kutta methods. *Comput. Math. Appl ,* 395-409.