Parallel Two-Processor Fifth Order Diagonally Implicit Runge-Kutta Method

Ummul Khair Salma Din¹, Fudziah Ismail²

¹School of Mathematical Sciences, Faculty of Science and Technology, Universiti Kebangsaan Malaysia, 43600 Bangi, Selangor
²Department of Mathematics, Faculty of Science, Universiti Putra Malaysia, 43400 UPM Serdang, Selangor
e-mail: ¹ummul@.ukm.my, ²fudziah_i@yahoo.com.my

ABSTRACT
The reduction of time on parallel computation has spurred interests on the development of numerical methods with parallel capability. This paper presents a derivation of a fifth order Runge-Kutta method which is suitable for parallel implementation. The sparsity structure in the development of fourth order parallel Runge-Kutta methods has inspired the construction of this method. Numerical results in terms of accuracy are compared to three established fourth order parallel Runge-Kutta method. The results based on sequential computations suggest that the new method is more accurate compared to the three methods. On the parallel performance of the method an algorithm was developed using C language and the parallel computation was done in Message Passing Interface (MPI) environment. The execution times of the method by sequential and parallel implementation on two large problems are presented. Results show that time taken for the parallel computation gave significant reduction compared to the sequential implementation.

Keywords: Fifth order Runge-Kutta method, initial value problem, parallel computation, two processors

INTRODUCTION
Runge-Kutta method is a famous one-step method for solving initial value problems (IVP). Developed from Euler’s Rule, Runge-Kutta methods are able to achieve higher order without sacrificing the one-step form. The classical fourth order Runge-Kutta method often emerged as one of numerical methods introduced mainly for students studying applied mathematics and engineering. Traditionally, Runge-Kutta methods are all explicit however Butcher (1987) has listed six basic reasons for taking a serious interest in implicit Runge-Kutta methods. The main reason is because of the higher orders of accuracy can be obtained than for explicit methods. Due to the excessive cost in evaluating the stages in a fully implicit Runge-Kutta method, many researchers have opted for the diagonally implicit Runge-Kutta (DIRK) method, as called by Alexander (1977). For these methods, the coefficient matrix A has a lower triangular structure with equal elements on the diagonal. Sometimes these methods are referred to as singly diagonally implicit Runge-Kutta (SDIRK), with DIRK methods not necessarily having equal diagonals.

Latest development has shown that attentions are now given to parallel implementation in numerical computation. According to Jackson (1991), the desire for parallel IVP solvers arises from
the need to solve many important problems more rapidly than is currently possible. The reduction in cost particularly time, is undeniably give great motivation in developing this idea.

Iserles and Nørsett’s (1990) idea in presenting parallel Runge-Kutta methods through sparsity structure is considered as one of the best parallel designs for Runge-Kutta methods. The structure allows the evaluation of the functions with independent arguments to be computed on different processors at the same time as one single evaluation on one processor. Previous methods using the sparsity structure are fourth order DIRK methods suitable to be implemented on two processors (Nørsett & Simonsen (1987), Jackson (1991), van der Houwen et al. (1992), Jackson & Nørsett (1995) and Burrage (1995)).

In this paper, we present a fifth order Runge-Kutta method (which later will be referred as P2DIRK5), where a pattern of DIRK is developed to permit the implementation of parallel computation. Comparison of numerical results based on sequential computations to established parallel fourth order Runge-Kutta methods will be discussed. The performance of P2DIRK5 particularly on the cost of computation time in solving two large systems of ordinary differential equations (ODEs) will be presented.

THE PARALLELISM OF RUNGE-KUTTA METHODS

The IVP for a system of first order ordinary differential equations (ODEs) is defined by

\[ y'(x) = f(x, y), \quad x \in [a, b], \quad y(a) = y_0 \]  \hspace{1cm} (1)

The general \( s \)-stage Runge-Kutta method for problem as in Eq. (1) is defined by

\[
\begin{aligned}
y_{n+1} &= y_n + h \sum_{i=1}^{s} b_i k_i \\
k_i &= f \left( x_n + c_i h, y_n + h \sum_{j=1}^{s} a_{ij} k_j \right), \quad i = 1, 2, ..., s,
\end{aligned}
\]  \hspace{1cm} (2)

assuming the following holds:

\[
c_i = \sum_{j=1}^{s} a_{ij}, \quad i = 1, 2, ..., s.
\]  \hspace{1cm} (3)

The coefficients in Eqs. (2) can also be displayed by using Butcher’s array or Butcher’s tableau (Butcher, 1987) as shown in Figure 1.

\[
\begin{array}{c|c}
c & A \\ \hline & b^T \\
\end{array}
\]

Figure 1. Butcher’s array or Butcher’s tableau

The \( s \)-dimensional vectors \( c \) and \( b \) and the \( s \times s \) matrix \( A \) in Figure 1 are \([c_1, c_2, \ldots, c_s]^T\), \([b_1, b_2, \ldots, b_s]^T\) and \([a_{ij}]\), respectively.
Iserles and Nørsett (1990) have explored the parallelism in Runge-Kutta methods by analyzing directed graph for matrix of sparsity structure as shown in Figure 2 (also in Nørsett and Simonsen, 1987).

In Runge-Kutta formulation it is written as,

$$y_{n+1} = y_n + h \sum_{j=1}^{4} b_j k_j$$

where

$$k_1 = f(x_n + c_1 h, y_n + h(a_{11} k_1)),$$

$$k_2 = f(x_n + c_2 h, y_n + h(a_{22} k_2)),$$

$$k_3 = f(x_n + c_3 h, y_n + h(a_{31} k_1 + a_{32} k_2 + a_{33} k_3)),$$

$$k_4 = f(x_n + c_4 h, y_n + h(a_{41} k_1 + a_{42} k_2 + a_{44} k_4)),$$

which is a fourth order diagonally implicit Runge-Kutta method. Clearly from the method, does not depend on and does not depend on $k_3$. A digraph is used to model the sparsity pattern of the matrix as shown in Figure 3.

$$\begin{array}{c|cccc}
  c_1 & a_{11} \\
  c_2 & 0 & a_{22} \\
  c_3 & a_{31} & a_{32} & a_{33} \\
  c_4 & a_{41} & a_{42} & 0 & a_{44} \\
\end{array}$$

$$b_1 \quad b_2 \quad b_3 \quad b_4$$

**Figure 2.** Sparsity structure for fourth order Runge-Kutta methods.

In Runge-Kutta formulation it is written as,

$$y_{n+1} = y_n + h \sum_{j=1}^{4} b_j k_j$$

where

$$k_1 = f(x_n + c_1 h, y_n + h(a_{11} k_1)),$$

$$k_2 = f(x_n + c_2 h, y_n + h(a_{22} k_2)),$$

$$k_3 = f(x_n + c_3 h, y_n + h(a_{31} k_1 + a_{32} k_2 + a_{33} k_3)),$$

$$k_4 = f(x_n + c_4 h, y_n + h(a_{41} k_1 + a_{42} k_2 + a_{44} k_4)),$$

which is a fourth order diagonally implicit Runge-Kutta method. Clearly from the method, does not depend on and does not depend on $k_3$. A digraph is used to model the sparsity pattern of the matrix as shown in Figure 3.

**Figure 3.** Digraph of sparsity structure for fourth order Runge-Kutta methods.
Methods of this kind are:

\[
\begin{array}{c|cccc}
1 & 1 & 0 & 0 & 0 \\
\frac{2}{3} & 0 & \frac{2}{3} & 0 & 0 \\
0 & \frac{121}{44} & -\frac{215}{44} & 1 & 0 \\
\frac{2}{5} & -\frac{43}{20} & \frac{39}{20} & 0 & \frac{3}{5} \\
\hline
\frac{11}{72} & \frac{23}{72} & \frac{11}{72} & \frac{23}{72}
\end{array}
\]

**Figure 4.** Method by Jackson and Norsett (DIRK1)

\[
\begin{array}{c|cccc}
\frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\
\frac{2}{3} & 0 & \frac{2}{3} & 0 & 0 \\
\frac{1}{2} & -\frac{5}{2} & \frac{5}{2} & \frac{1}{2} & 0 \\
\frac{1}{3} & -\frac{5}{3} & \frac{4}{3} & 0 & \frac{2}{3} \\
\hline
-1 & \frac{3}{2} & -1 & \frac{3}{2}
\end{array}
\]

**Figure 5.** Method by Iserles and Norsett (DIRK2)

\[
\begin{array}{c|cccc}
\frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 \\
\frac{1}{2} & \frac{3}{2} & -\frac{3}{2} & \frac{1}{2} & 0 \\
0 & -3 & 2 & 0 & 1 \\
\hline
\frac{1}{3} & \frac{1}{6} & \frac{1}{3} & \frac{1}{6}
\end{array}
\]

**Figure 6.** Method by Iserles and Norsett (DIRK3).

Figure 4 is a method by Jackson and Norsett (1995) and through the discussion it will be referred as DIRK1. Meanwhile Figures 5 and 6 which are methods by Iserles and Norsett (1990) will be referred as DIRK2 and DIRK3. These three methods are suitable for parallel implementation using two processors.

We extend the method to a higher order Runge-Kutta method which is a fifth order with six stages having the same sparsity structure (see Figure 7).
The derivation of the method can be found in Din et al. (2006). Following is the coefficients obtained for the method.

$$\gamma = 0.5111360444707436, \quad a_{11} = 0.0$$
$$\beta = 0.1200913622711499, \quad a_{10} = \beta,$$
$$a_{22} = \beta, \quad c_i = \gamma,$$
$$a_{32} = 0.3304320730047836, \quad c_2 = \beta,$$
$$a_{33} = 0.5083725832456476, \quad c_3 = 0.5083725832456476,$$
$$a_{42} = -0.001764317609504879, \quad c_4 = 0.1048607614397692,$$
$$a_{33} = 0.0, \quad c_5 = 1 - \gamma,$$
$$a_{44} = \beta, \quad c_6 = 1 - \beta,$$
$$a_{52} = 0.09184637904240046, \quad b_1 = 0.0,$$
$$a_{53} = 0.2402905535606616, \quad b_2 = 0.0,$$
$$a_{54} = 0.206131006694386, \quad b_3 = -0.17967989773612,$$
$$a_{55} = \gamma, \quad b_4 = 0.2614074344151084,$$
$$a_{62} = 0.008530162133924786, \quad b_5 = 0.6241305207277307,$$
$$a_{63} = 0.5217820185104087, \quad b_6 = 0.2941419426307729,$$
$$a_{64} = 0.251411354949371,$$

The values of \( a_i \) for \( i = 1, 2, \ldots, 6 \) are obtained by following the relationship,

$$a_i = c_i - \sum_{j=2}^{i-1} a_{ij}$$
STABILITY OF THE METHOD

The stability function or stability polynomial \( R(\hat{h}) \) as given in Lambert (1991)

\[
R(\hat{h}) = 1 + \hat{h} b^T (I - \hat{h}A)^{-1} e
\]

is obtained when the method is applied to the linear test equation

\[
y' = \lambda y, \quad \lambda \in \mathbb{C}, \quad \text{Re}(\lambda) < 0
\]

\( \hat{h} \) in Eq. (5) represents \( k \gamma \) and \( e = [1, 1, \ldots, 1]^T \). An alternative expression for Eq. is given in the following lemma. The proof can be seen in Butcher (2003).

**Lemma:** Let \((A,b,c)\) denote a Runge-Kutta method. Then its stability function is given by

\[
R(\hat{h}) = \frac{\det(I + \hat{h}(eb^T - A))}{\det(I - \hat{h}A)}.
\]

Using Mathematica, we obtained the stability polynomial for the method as

\[
R(\hat{h}) = \frac{0.890628(\hat{h} - 19.1201)(\hat{h} - 8.52181)(\hat{h}^2 - 3.11507\hat{h} + 2.53164)(\hat{h}^2 + 5.96002\hat{h} + 11.7689)}{(\hat{h} - 8.32693)(\hat{h} - 1.95644)(\hat{h}^2 - 16.6541\hat{h} + 69.3394)(\hat{h}^2 - 3.91283\hat{h} + 3.82757)}
\]

while the stability region where the region lies outside the closed boundary is shown in Figure 8. P2DIRK5 is said to have an \( A(\alpha) \)-stability with \( \alpha \approx 88.58^\circ \).

![Figure 8. Stability region for PDIRK5](image-url)
THE PERFORMANCE OF THE PARALLEL PROCESSING

In comparing the performance between a multiprocessor system and a single processor system we shall use the speedup factor and efficiency denoted by $S_p$ and $E_p$ respectively. $S_p$ is defined as,

$$S_p = \frac{t_p}{t_p^s}$$

where $t_p$ is the execution time of the best sequential algorithm running on a single processor while $t_p^s$ is the execution time for solving the same problem on a multiprocessor. The maximum speedup possible is usually $p$ with processors $p$, normally referred to as linear speedup. However several factors will appear as overhead in the parallel version and limit the speedup making it rare in achieving the linear result. Among common overheads are (Wilkinson & Allen, 2005):

1. Periods when not all processors can be performing useful work and are simply idle.
2. Extra computations in the parallel version not appearing in the sequential version.
3. Communication time between processes.

Due to these factors the speedup is often less than $p$.

The efficiency $E_p$ is defined as,

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

which is the ratio of speedup to the number of processors. It is used to measure the processor utilization. If given as a percentage, the efficiency of 100% happens when all the processors are being used on the computation at all times and $S_p$ equal to $p$.

The algorithm for P2DIRK5 is executed in C language. For the parallel implementation it is supported by Message Passing Interface (MPI) which is a message passing library standard and is extensively used to write message passing programs on high performance computing (HPC) platforms. Both sequential and parallel algorithms were run on Sunfire V1280 with eight homogenous processors located at Institute of Mathematical Research (INSPEM), Universiti Putra Malaysia, Serdang.

NUMERICAL EXPERIMENTS

Accuracy of the Method

All problems that were tested are non-stiff problems. The codes for the algorithm are written and compiled in Microsoft Visual C++. Table 1 gives the performance comparison between P2DIRK5, DIRK1, DIRK2 and DIRK3 in term of maximum error. The step sizes used are 0.01, 0.001, 0.0001 and 0.00001. The maximum error is defined as

$$\max_{151.5 \text{ steps}} \left| y_i - y(x_i) \right|$$

where $y_i$ is the computed value and $y(x_i)$ is the true solution of the problems.
A. Tested problems:

Problem S1:

\[ y' = \frac{1}{2} (x - y) \]

\[ y(0) = 1, \ 0 \leq x \leq 20 \]

Exact solution: \[ y(x) = e^x \]

Source: Artificial problem

Problem S2:

\[ y' = \frac{1}{2} (x - y) \]

\[ y(0) = 1, \ 0 \leq x \leq 20 \]

Exact solution: \[ y(x) = -2 + x + 3e^{-\frac{1}{2}x} \]

Source: Mathews (1992)

Problem S3:

\[ y_1' = -y_1 - \sqrt{3} y_2, \ y_2 = \sqrt{3} y_1 - y_3 \]

\[ y_1(0) = 1, \ y_2(0) = 0, \ 0 \leq x \leq 20 \]

Exact solutions: \[ y_1(x) = e^{-x} \cos \sqrt{3} x, \ y_2(x) = e^{-x} \sin \sqrt{3} x \]

Source: Tam (1992)

Problem S4:

\[ y_1 = -y_1 + y_2, \ y_2 = y_1 - 2y_2 + y_3, \ y_3 = y_2 - y_3 \]

\[ y_1(0) = 2, \ y_2(0) = 0, \ y_3(0) = 1, \ 0 \leq x \leq 20 \]

Exact solutions: \[ y_1(x) = \frac{1}{2} e^{-3x} + 1 + \frac{1}{2} e^{-x}, \]

\[ y_2(x) = 1 - e^{-3x}, \ y_3(x) = 1 + \frac{1}{2} e^{-3x} - \frac{1}{2} e^{-x} \]

Source: Shampine (1980)

Problem S5:

\[ y_1 = y_2, \ y_2 = -y_1, \ y_3 = y_4, \ y_4 = y_2 + 2e^x \]

\[ y_1(0) = 0, \ y_2(0) = -2, \ y_3(0) = 0, \ y_4(0) = -2, \ 0 \leq x \leq 10 \]

Exact solutions: \[ y_1(x) = -e^x + e^{-x}, \ y_2(x) = -e^{-x} - e^{-x}, \]

\[ y_3(x) = e^x - e^{-x}, \ y_4(x) = e^x + e^{-x} \]

Source: Bronson (1973)
B. Results

Table 1. Numerical Results for Test Problems S1 - S5 using DIRK1, DIRK2, DIRK3 and P2DIRK5

<table>
<thead>
<tr>
<th>Problem</th>
<th>Method</th>
<th>h=0.01</th>
<th>h=0.001</th>
<th>h=0.0001</th>
<th>h=0.00001</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DIRK1</td>
<td>7.3605888(-08)</td>
<td>7.3612702(-08)</td>
<td>7.3579575(-08)</td>
<td>7.3576266(-08)</td>
</tr>
<tr>
<td></td>
<td>DIRK2</td>
<td>6.4177323(-09)</td>
<td>1.1036538(-10)</td>
<td>1.1032453(-11)</td>
<td>1.1093904(-12)</td>
</tr>
<tr>
<td></td>
<td>DIRK3</td>
<td>7.3685400(-08)</td>
<td>7.3612690(-08)</td>
<td>7.3579575(-08)</td>
<td>7.3576266(-08)</td>
</tr>
<tr>
<td></td>
<td>P2DIRK5</td>
<td>6.0281913(-10)</td>
<td>6.0285110(-14)</td>
<td>4.2743586(-15)</td>
<td>7.3274720(-15)</td>
</tr>
<tr>
<td>2</td>
<td>DIRK1</td>
<td>4.0079530(-07)</td>
<td>3.9980949(-07)</td>
<td>3.9971949(-07)</td>
<td>3.9970999(-07)</td>
</tr>
<tr>
<td></td>
<td>DIRK2</td>
<td>1.4971420(-09)</td>
<td>1.4979307(-10)</td>
<td>1.4992452(-11)</td>
<td>1.8207658(-12)</td>
</tr>
<tr>
<td></td>
<td>DIRK3</td>
<td>4.0079532(-07)</td>
<td>3.9980949(-07)</td>
<td>3.9971948(-07)</td>
<td>3.9970999(-07)</td>
</tr>
<tr>
<td></td>
<td>P2DIRK5</td>
<td>3.7505332(-11)</td>
<td>7.1054274(-14)</td>
<td>1.1368684(-13)</td>
<td>6.2527761(-13)</td>
</tr>
<tr>
<td>3</td>
<td>DIRK1</td>
<td>1.5551270(-07)</td>
<td>1.3570843(-07)</td>
<td>1.3742278(-07)</td>
<td>1.3741428(-07)</td>
</tr>
<tr>
<td></td>
<td>DIRK2</td>
<td>8.8400030(-08)</td>
<td>4.4003146(-10)</td>
<td>4.3480053(-11)</td>
<td>4.3480497(-12)</td>
</tr>
<tr>
<td></td>
<td>DIRK3</td>
<td>1.5264842(-07)</td>
<td>1.3750830(-07)</td>
<td>1.3742278(-07)</td>
<td>1.3741428(-07)</td>
</tr>
<tr>
<td></td>
<td>P2DIRK5</td>
<td>8.3484779(-09)</td>
<td>8.3568894(-13)</td>
<td>2.1371793(-15)</td>
<td>5.7731597(-15)</td>
</tr>
<tr>
<td>4</td>
<td>DIRK1</td>
<td>3.6288467(-07)</td>
<td>7.3683759(-08)</td>
<td>7.3586936(-08)</td>
<td>7.3577007(-08)</td>
</tr>
<tr>
<td></td>
<td>DIRK2</td>
<td>5.1755977(-07)</td>
<td>3.3248782(-10)</td>
<td>3.3109404(-11)</td>
<td>3.3071323(-12)</td>
</tr>
<tr>
<td></td>
<td>DIRK3</td>
<td>3.3138603(-07)</td>
<td>7.3684274(-08)</td>
<td>7.3586934(-08)</td>
<td>7.3577007(-08)</td>
</tr>
<tr>
<td></td>
<td>P2DIRK5</td>
<td>4.8777528(-08)</td>
<td>4.8845324(-12)</td>
<td>2.0250468(-13)</td>
<td>1.7443824(-12)</td>
</tr>
<tr>
<td>5</td>
<td>DIRK1</td>
<td>1.0926560(-02)</td>
<td>1.1000071(-02)</td>
<td>1.1003057(-02)</td>
<td>1.1003246(-02)</td>
</tr>
<tr>
<td></td>
<td>DIRK2</td>
<td>1.3619845(-04)</td>
<td>1.3223398(-05)</td>
<td>1.3228455(-06)</td>
<td>1.3214958(-07)</td>
</tr>
<tr>
<td></td>
<td>DIRK3</td>
<td>1.0929159(-02)</td>
<td>1.1000069(-02)</td>
<td>1.1003057(-02)</td>
<td>1.1003246(-02)</td>
</tr>
<tr>
<td></td>
<td>P2DIRK5</td>
<td>1.9870018(-05)</td>
<td>1.7280399(-09)</td>
<td>1.7826096(-10)</td>
<td>4.1836756(-10)</td>
</tr>
</tbody>
</table>

Execution time

Two large problems were tested using P2DIRK5 to compare the sequential and parallel time taken to solve the problems for N=5000, 10000, 15000, 20000 and 30000 for step sizes 0.01, 0.001, 0.0001 and 0.00001. Table 2 shows the total steps for every step size considered.

Table 2. Step size and its total steps

<table>
<thead>
<tr>
<th>Step size</th>
<th>Total steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>100</td>
</tr>
<tr>
<td>0.001</td>
<td>1000</td>
</tr>
<tr>
<td>0.0001</td>
<td>10000</td>
</tr>
<tr>
<td>0.00001</td>
<td>100000</td>
</tr>
</tbody>
</table>
A. Tested problems

Problem L1 (A Parabolic Partial Differential Equations):

\[
\begin{pmatrix}
y_1 \\
y_2 \\
\vdots \\
y_N
\end{pmatrix}
= 
\begin{pmatrix}
-2 & 1 \\
1 & -2 & 1 \\
\vdots & \vdots & \vdots \\
1 & -2 & 1
\end{pmatrix}
\begin{pmatrix}
y_1 \\
y_2 \\
\vdots \\
y_N
\end{pmatrix}
+ 
\begin{pmatrix}
1 \\
0 \\
\vdots \\
0
\end{pmatrix}
, \quad y(0) = \begin{pmatrix}
1 \\
0 \\
\vdots \\
0
\end{pmatrix}
\]

N = Number of equations

0 ≤ x ≤ 1

Source: Hull et al. (1972)

Problem L2 (Lagrange Equation for the hanging string):

\[
\begin{align*}
y_1 &= y_2 \\
y_2 &= -y_1 + 2y_3 \\
y_3 &= y_4 \\
y_4 &= y_1 - 5y_2 + 4y_3 \\
&\vdots \\
y_{N-1} &= y_N \\
y_N &= (N - 3)y_{N-3} - (2N - 3)y_{N-1}
\end{align*}
\]

N = Number of equations

0 ≤ x ≤ 1

Initial values: \( y_i(0) = 0 \) except \( y_{N-2}(0) = 1 \)


B. Results

The performance of the sequential and parallel execution times for every problem is shown in Figure 9(a) and 10(a) respectively while Figure 9(b) and 10(b) show the speedup performance for the problems.
Parallel Two-Processor Fifth Order Diagonally Implicit Runge-Kutta Method

(a) Time taken for sequential and parallel executions
(b) Speedup

Figure 9. Results for Problem L1

(a) Time taken for sequential and parallel executions
(b) Speedup

Figure 10. Results for Problem L2

The efficiency of the method is given in Table 3.

Table 3. The efficiency of P2DIRK5

<table>
<thead>
<tr>
<th>No. of equations</th>
<th>Problem</th>
<th>Efficiency (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>h=0.01</td>
</tr>
<tr>
<td>5000</td>
<td>L1</td>
<td>65.13</td>
</tr>
<tr>
<td></td>
<td>L2</td>
<td>65.81</td>
</tr>
<tr>
<td>10000</td>
<td>L1</td>
<td>69.22</td>
</tr>
<tr>
<td></td>
<td>L2</td>
<td>69.35</td>
</tr>
<tr>
<td>15000</td>
<td>L1</td>
<td>79.53</td>
</tr>
<tr>
<td></td>
<td>L2</td>
<td>85.49</td>
</tr>
<tr>
<td>20000</td>
<td>L1</td>
<td>85.51</td>
</tr>
<tr>
<td></td>
<td>L2</td>
<td>85.07</td>
</tr>
<tr>
<td>30000</td>
<td>L1</td>
<td>85.62</td>
</tr>
<tr>
<td></td>
<td>L2</td>
<td>85.32</td>
</tr>
</tbody>
</table>
DISCUSSION AND CONCLUSION

From the results in Table 1, it is found that among the three established methods, DIRK2 shows the most accurate result. An interesting finding can be seen for DIRK1 and DIRK3 where they reach to a same performance when we used the step sizes 0.0001 and 0.00001. For all methods, the smaller the step size used, the results became more accurate. Finally, it is clearly shown that PDIRK5 is the best method among the four methods in term of accuracy.

As for the execution time, Figure 9(a) and 10(a), showed that for every step sizes the parallel execution times are better than the sequential execution times. The time reduction is more apparent as the number of equations increased and the results are consistent for every step size. In Figure 9(b) and 10(b), the results show that as the number of equations gets larger, the speedup increased. In Problem L2, for step sizes 0.0001 and 0.00001 the speedup have reached up to 1.83 and 1.85 respectively, approaching to the ideal speedup for two processors execution. The processors utilization for that particular execution are 91.53% and 92.42% respectively, as shown in Table 3. We can say that during the execution, the processors are almost fully utilized. We can also observed in Table 3 that as the problems get larger, the processors utilization improved.

P2DIRK5 is a newly derived fifth order DIRK method with parallel capability. The parallel structure is based on the sparsity structure introduced by Iserles and Norsett (1990) which has been used in deriving efficient fourth order DIRK methods. Din et. al (2006), have shown that P2DIRK5. In this paper, we showed that P2DIRK has better accuracy than the fourth order methods of its type and the ability to reduce the computation time by the parallel implementation on two processors. This contribution is significant when large problems come in hands because time reduction means cheaper cost in the computation processes. This makes P2DIRK5 as an efficient method for solving ODEs.

REFERENCES


