The Parallel Three-Processor Fifth Order Diagonally Implicit Runge-Kutta Methods for Solving Ordinary Differential Equations

UMMUL KHAIR SALMA DIN
Universiti Kebangsaan Malaysia
Faculty of Science & Technology
School of Mathematical Sciences
43600 UKM Bangi, Selangor
MALAYSIA

FUDZIAH ISMAIL[1]
MOHAMED SULEIMAN[2]
ZANARIAH ABDUL MAJID[3]
Universiti Putra Malaysia
Faculty of Science
Department of Mathematics
43400 UPM, Serdang, Selangor
MALAYSIA

MOHAMED OTHMAN
Universiti Putra Malaysia
Faculty of Computer Science
& Information Technology
Department of Communication
Technology & Network
43400 UPM, Serdang, Selangor
MALAYSIA

Abstract: Fifth order diagonally implicit Runge-Kutta methods with a modified sparsity structure suitable for parallel implementations on three processors are developed. The efficiency of the methods in terms of accuracy to solve a standard set of problems is compared to an established method. From the results we can conclude the new methods are comparable to the existing method.

Key–Words: Parallelism, Fifth order Runge-Kutta methods, Ordinary differential equations

1 Introduction

The primary objective in applying parallelism in numerical computation is the significant reduction in time which appears to be one of the contributing costs in computers execution. Iserles and Nørsett's (see [1]) 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. Furthermore the structure is meant for diagonally implicit Runge-Kutta (DIRK) methods where a higher accuracy could be achieved as well as it can be applied to solve stiff ordinary differential equations (ODEs).

Previous methods using the sparsity structure are fourth order DIRK methods suitable to be implemented on two processors (Nørsett and Simonsen [2], Jackson [3], Van Der Houwen et. al [4], Jackson and Nørsett [5] and Burrage [6]). In this paper, we present fifth order Runge-Kutta methods, where a pattern of DIRK is developed to permit the implementation of parallel computation using three processors.

2 The Runge-Kutta Method

The initial value problem (IVP) for a system of first order ODEs is defined by

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

The general s-stage Runge-Kutta method for problem (1) is defined by

$$y_{n+1} = y_n + h \sum_{i=1}^{s} b_i k_i \qquad (2)$$

where

$$k_i = f\left(x_n + c_i h, y_n + h \sum_{j=1}^{s} a_{ij} k_j\right), \quad i = 1, 2, \ldots, s.$$

assuming the following holds:

$$c_i = \sum_{j=1}^{s} a_{ij},\ \ i = 1, 2, \ldots, s. \qquad (3)$$

In *Butcher's array* (see [7]) the coefficients in (2) are written as

$$\begin{array}{c|cccccc} c_1 & a_{11} & a_{12} & a_{13} & \ldots & a_{1s} \\ c_2 & a_{21} & a_{12} & a_{13} & \ldots & a_{1s} \\ \vdots & & \vdots & & & \vdots \\ c_s & a_{s1} & a_{s2} & a_{s3} & \ldots & a_{ss} \\ \hline & b_1 & b_2 & b_3 & \ldots & b_s \end{array}$$

or simply as

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

with the s-dimensional vectors c and b and the $s \times s$ matrix A, denoted by $c = [c_1, c_2, c_3, \ldots, c_s]^T, b = [b_1, b_2, b_3, \ldots, b_s]^T$ and $A = [a_{ij}]$ respectively. The method is said to be explicit if $a_{ij} = 0$ for $i \leq j$, semi-implicit if $a_{ij} = 0$ for $i < j$ and fully implicit otherwise.

2.1 The Workload and the Needs for Parallel Computing

Shampine [8] has noted that the cost of an integration is conventionally measured by the number of function evaluations required and in Runge-Kutta methods this is done at every stage which is represented by k_i in (2). This means that the cost would increase if a higher number of k involved and would continue to increase if we deal with a large problem. An example of problem that requires a huge number of calculations is predicting the motion of the astronomical bodies in space or better known as the N-body problem. The N-body problem also appears in modeling chemical and biological systems at the molecular level and takes enormous computational power. Basically there are two reasons for using parallel computing, to save time and to solve large problem. Having a few processors or sometimes referred to as *workers* or *laborers* or *slaves* to do calculations concurrently, large problem could be solved in shorter time.

2.2 Parallelism of the Method

The theory expressed in Iserles and Nørsett depicted through directed digraphs explicitly showed groups of stages which are independence of each other. The sparsity pattern as shown in Figure 1 together with its digraph has been chosen in developing our method as we intend to implement the method on three processors.

Initial efforts in deriving the method are based wholly on the array in Figure 2 which represents the Runge-Kutta method for the sparsity structure in Figure 1.

However we found that with certain a_{ij}'s assigned to zero, the freedom in manipulating the system of equations obtained from the order conditions is limited leading to the lacks of unknowns compare to the number of equations. These would make it impossible to find the solution. To overcome this situation, we decided to add an explicit first stage to the method

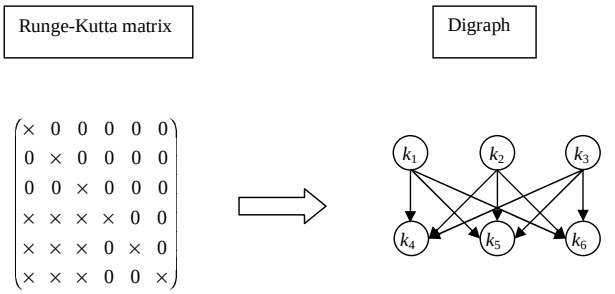

Figure 1: Sparsity structure and digraph for Runge-Kutta methods with six stages for three processors

c_1	α					
c_2	0	β				
c_3	0	0	γ			
c_4	a_{51}	a_{52}	a_{53}	α		
c_5	a_{61}	a_{62}	a_{63}	0	β	
c_6	a_{71}	a_{72}	a_{73}	0	0	γ
	b_1	b_2	b_3	b_4	b_5	b_6

Figure 2: Butcher's array for Runge-Kutta methods with six stages for three processors

without changing the original sparsity structure. The new matrix is shown in Figure 3 together with its digraph.

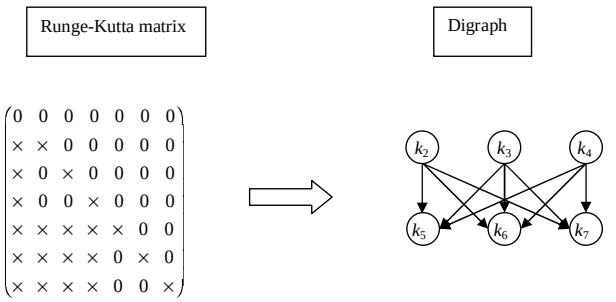

Figure 3: The modified sparsity structure and digraph for Runge-Kutta methods for three processors

3 Derivation of Fifth Order Parallel Runge-Kutta Methods for Three Processors

In order to derive a fifth order diagonally implicit Runge-Kutta method, 17 equations have to be satis-

fied. The equations associated with the order of the method are given in Table 1.

We used four assumptions (Butcher [7, 9]) to reduce and simplify the equations so that it would be easier to solve. The assumptions are

$$\text{(i)} \sum_{i=1}^{7} b_i c_i^k = \frac{1}{k+1}, \quad \text{for } k = 0, 1, 2, 3, 4. \tag{4}$$

$$\text{(ii)} \sum_{j=1}^{7} a_{ij} c_j = \frac{1}{2} c_i^2, \quad \text{for } i = 1, 2, \ldots, 7. \tag{5}$$

$$\text{(iii)} \sum_{j=1}^{7} a_{ij} c_j^2 = \frac{1}{3} c_i^3, \quad \text{for } i = 1, 2, \ldots, 7. \tag{6}$$

$$\text{(iv)} \sum_{i=1}^{7} b_i a_{ij} = b_j (1 - c_j), \quad \text{for } j = 1, 2, \ldots, 7. \tag{7}$$

According to Butcher [9], assumption (4) usually denoted by $B(\eta)$ is necessary to be satisfied for a method to be of order η. Therefore equations (8), (9), (10), (12) and (16) have to be satisfied. Equation (8) is also known as the consistency condition (see [7]).

Using (5) and (6), we removed equations (11),(13) – (15), (17) – (19),(21) – (23) and (24). Generally, (5) and (6) are meant for $i = 1, 2, \ldots, s$. In fact some unknowns can be determined when further inspection is done. For example when $i = 2$, equation (5) is simplified to be

$$\begin{aligned} & \alpha c_2 = \frac{c_2^2}{2} \\ \Rightarrow \quad & c_2 = 0 \text{ or } c_2 = 2\alpha \end{aligned}$$

But further inspection on equation (6) for $i = 2$, shows that c_2 either equal to zero or 3α. Since we do not want c_2 to be zero, we assigned b_2 to zero. The same thing arises for $i = 3$ and 4, therefore we have b_3 and b_4 also equal to zero. With the same arguments we imposed

$$\begin{aligned} \sum b_i a_{i2} = 0, \quad \sum b_i c_i a_{i2} = 0, \\ \sum b_i a_{i3} = 0, \quad \sum b_i c_i a_{i3} = 0, \\ \sum b_i a_{i4} = 0, \quad \sum b_i c_i a_{i4} = 0, \end{aligned}$$

As for (20), the last assumption (7) will be applied where it will give values for c_5, c_6 and c_7 which are $1 - \alpha$, $1 - \beta$ and $1 - \gamma$, respectively. The simplifying processes leave us with a total of 17 equations and 18 unknowns to be solved.

We solved the equations using Mathematica and have came out with two sets of solution. The first solution is obtained when we set the value of $\alpha = 0.25$ and $\beta = 0.5$ which will be denoted as P3DIRK5(i)

Table 1: Equations of order conditions for Runge-Kutta methods of order 5

Order of the tree	Tree	Elementary weights
1		$\sum b_i = 1$ (8)
2		$\sum b_i c_i = \frac{1}{2}$ (9)
3		$\sum b_i c_i^2 = \frac{1}{3}$ (10)
3		$\sum b_i a_{ij} c_j = \frac{1}{6}$ (11)
4		$\sum b_i c_i^3 = \frac{1}{4}$ (12)
4		$\sum b_i c_i a_{ij} c_j = \frac{1}{8}$ (13)
4		$\sum b_i a_{ij} c_j^2 = \frac{1}{12}$ (14)
4		$\sum b_i a_{ij} a_{jk} c_k = \frac{1}{24}$ (15)
5		$\sum b_i c_i^4 = \frac{1}{5}$ (16)
5		$\sum b_i c_i^2 a_{ij} c_j = \frac{1}{10}$ (17)
5		$\sum b_i a_{ij} c_j a_{ik} c_k = \frac{1}{20}$ (18)
5		$\sum b_i c_i a_{ij} c_j^2 = \frac{1}{15}$ (19)
5		$\sum b_i a_{ij} c_j^3 = \frac{1}{20}$ (20)
5		$\sum b_i c_i a_{ij} a_{jk} c_k = \frac{1}{30}$ (21)
5		$\sum b_i a_{ij} c_j a_{jk} c_k = \frac{1}{40}$ (22)
5		$\sum b_i a_{ij} a_{jk} c_k^2 = \frac{1}{60}$ (23)
5		$\sum b_i a_{ij} a_{jk} a_{kl} c_l = \frac{1}{120}$ (24)

Table 2: The solution for P3DIRK5(i)

0	**0**						
0.33333	**0.08333**	**0.25**					
0.4	**-0.1**	**0**	**0.5**				
0	**-0.4**	**0**	**0**	**0.4**			
0.75	**-0.796878**	**1.68743**	**-1.1718**	**0.78125**	**0.25**		
0.5	**-0.631947**	**1.49994**	**-1.56244**	**0.694444**	**0**	**0.5**	
0.6	**-1.01**	**2.15991**	**-1.94991**	**1**	**0**	**0**	**0.4**
	0.12963	**0**	**0**	**0**	**1.18519**	**2**	**-2.31481**

Table 3: The solution for P3DIRK5(ii)

0	0						
0.5	0	0.5					
0.4	-0.35	0	0.75				
0	-0.13333	0	0	0.13333			
0.5	2.74199	-0.666667	0.520833	-2.59615	0.5		
0.25	-1.6278	0.416667	-0.911458	1.6226	0	0.75	
0.866667	-0.852469	0.25679	0.329012	1	0	0	0.13333
	0.0897436	0	0	0	0.30303	0.288288	0.318938

and for the second solution α is 0.5 and β is 0.75, denoted as P3DIRK5(ii). Both solutions are given in Table 2 and 3, respectively.

4 Numerical Experiments

All problems that were tested are non-stiff problems. We compare the accuracy of the methods to fifth order DIRK methods by Al-Rabeh [10].

Problem 1:
$y' = -y$
$y(0) = 1, \ 0 \le x \le 20.$
Exact solution: $y(x) = e^{-x}$.
Source: Artificial problem.

Problem 2:
$y_1' = -y_1 - \sqrt{3}y_2$
$y_2' = \sqrt{3}y_1 - y_2$
$y_1(0) = 1, \ y_2(0) = 0, \ 0 \le x \le 20.$
Exact solution:
$y_1(x) = e^{-x}\cos\sqrt{3}x$
$y_2(x) = e^{-x}\sin\sqrt{3}x$
Source: Tam [11].

Problem 3:
$y_1' = y_2 - y_1$
$y_2' = y_1 - 2y_2 + y_3$
$y_3' = y_2 - y_3$
$y_1(0) = 2, \ y_2(0) = 0, \ y_3(0) = 1, \ 0 \le x \le 20.$
Exact solution:
$y_1(x) = \frac{1}{2}e^{-3x} + 1 + \frac{1}{2}e^{-x}$
$y_2(x) = 1 - e^{-3x}$
$y_3(x) = 1 + \frac{1}{2}e^{-3x} - \frac{1}{2}e^{-x}$
Source: Shampine [12].

Problem 4:
$y_1' = y_2$
$y_2' = -y_3$
$y_3' = y_4$
$y_4' = y_2 + 2e^x$
$y_1(0) = 0, \ y_2(0) = -2,$
$y_3(0) = 0, \ y_4(0) = 2, \ 0 \le x \le 10.$
Exact solution:
$y_1(x) = -e^x + e^{-x}$
$y_2(x) = -e^x - e^{-x}$
$y_3(x) = e^x - e^{-x}$
$y_3(x) = e^x + e^{-x}$
Source: Bronson [13].

Problem 5:
$y_1' = y_3, \ y_2' = y_4, \ y_3' = -y_1/r^3, \ y_4' = -y_2/r^3$
$r = \sqrt{y_1^2 + y_2^2}$
$y_1(0) = 1, \ y_2(0) = 0,$
$y_3(0) = 0, \ y_4(0) = 1, \ 0 \le x \le 75.$
Exact solution:
$y_1(x) = \cos x, \ y_2(x) = \sin x,$
$y_3(x) = -\sin x, \ y_4(x) = \cos x.$
Source: Dormand [14].

The code for the algorithm to run the method is done in C language. Tables 4 and 5 show the performance comparison between P3DIRK5(i), P3DIRK5(ii) and Al-Rabeh in term of maximum error. The step sizes used are 10^{-2}, 10^{-3}, 10^{-4} and 10^{-5}. The maximum error is defined as

$$\max_{1 \le i \le \text{steps}} (|y_i - y(x_i)|)$$

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

Table 4: Numerical results for test problems 1–5 using P3DIRK5(i),P3DIRK5(ii) and Al-Rabeh with step-sizes 10^{-2} and 10^{-3}

Problem	Method	h=10^{-2}	h=10^{-3}
1	P3DIRK5(i)	1.2898313E-03	1.2956515E-04
	P3DIRK5(ii)	8.9293887E-04	8.9698708E-05
	Al-Rabeh	7.0377347E-04	7.0695342E-05
2	P3DIRK5(i)	1.8299346E-03	1.8475820E-04
	P3DIRK5(ii)	1.2667456E-03	1.2777535E-04
	Al-Rabeh	9.9836453E-04	1.0070508E-04
3	P3DIRK5(i)	1.4575903E-07	1.3666056E-07
	P3DIRK5(ii)	1.3094262E-07	1.2992110E-11
	Al-Rabeh	9.2675797E-10	3.6946779E-11
4	P3DIRK5(i)	2.6018265E-01	3.9015584E-02
	P3DIRK5(ii)	1.7170707E-01	1.7145870E-02
	Al-Rabeh	1.3527754E-01	1.3510061E-02
5	P3DIRK5(i)	5.8036620E-01	5.8702730E-02
	P3DIRK5(ii)	4.0384028E-01	4.0648689E-02
	Al-Rabeh	3.1876621E-01	3.2034277E-02

5 Conclusion

From Tables 4 and 5, it is observed that P3DIRK5(ii) gives better performance in term of accuracy compared to P3DIRK5(i) and as accurate as the method

Table 5: Numerical results for test problems 1–5 using P3DIRK5(i),P3DIRK5(ii) and Al-Rabeh with step-sizes 10^{-4} and 10^{-5}

Problem	Method	h=10^{-4}	h=10^{-5}
1	P3DIRK5(i)	1.2962347E-05	1.3004077E-06
	P3DIRK5(ii)	8.9739102E-06	8.9743141E-07
	Al-Rabeh	7.0727168E-06	7.0730351E-07
2	P3DIRK5(i)	1.8667436E-05	2.0585134E-06
	P3DIRK5(ii)	1.2788567E-05	1.2789670E-06
	Al-Rabeh	1.0079157E-05	1.0079559E-06
3	P3DIRK5(i)	1.3629250E-07	1.3625571E-07
	P3DIRK5(ii)	2.0317081E-13	1.7443824E-12
	Al-Rabeh	3.6806225E-11	3.6786352E-11
4	P3DIRK5(i)	2.2239429E-02	2.0562710E-02
	P3DIRK5(ii)	1.7143998E-03	1.7143773E-04
	Al-Rabeh	1.3478727E-03	1.3179881E-04
5	P3DIRK5(i)	5.8451758E-03	5.5989628E-04
	P3DIRK5(ii)	4.0650716E-03	4.0650674E-04
	Al-Rabeh	3.2038271E-03	3.2039211E-04

by Al-Rabeh in most of the problems tested except for Problem 3 where as the step size gets smaller P3DIRK5(ii) performed better. As a whole the two new methods are comparable to Al-Rabeh's method. In addition, the methods have the advantage in reducing the cost of computation since it could be implemented in parallel which is very significant when large problems come in hands.

Acknowledgements: The first author acknowledge Universiti Kebangsaan Malaysia and Ministry of Higher Education, Malaysia, for the financial support received throughout the course of this study.

References:

[1] A. Iserles and S.–P. Nørsett, On the theory of parallel Runge-Kutta methods, *IMA Journal of Numerical Analysis* 10, 1990, pp. 463–488.

[2] S.–P. Nørsett and H.–H. Simonsen, *Aspects of parallel Runge-Kutta methods,* in Numerical methods for ordinary differential equations, A. Bellen, C.–W. Gear and E. Russo, eds., Lecture Notes in Mathematics: pp. 103–107, Springer-Verlag, Berlin 1987.

[3] K.–R. Jackson, A survey of parallel numerical methods for initial value problems for ordinary differential equations, *IEEE Transactions on Magnetic* 27(5), 1991, pp. 3792–3797.

[4] P.–J. Van Der Houwen, B.–P. Sommeijer and W. Couzy, 1992. Embedded diagonally implicit Runge-Kutta algorithms on parallel computer, *Maths. Of Comp.* 58, 1992, pp. 135–59.

[5] K.–R. Jackson and S.–P. Nørsett, The potential for parallelism in Runge-Kutta methods. Part 1: RK formulas in standard form, *Siam J. Numer. Anal.* 32(1), 1995, pp. 49–82.

[6] K. Burrage, *Parallel and sequential methods for ordinary differential equations,* Oxford University Press Inc., New York 1995.

[7] J.–C. Butcher, *The numerical analysis of ordinary differential equations: Runge-Kutta and General Linear methods,* John Wiley & Sons, 1987.

[8] L.–F. Shampine, *Numerical solution of ordinary differential equations,* Chapman & Hall, New York 1994.

[9] J.–C. Butcher, *Numerical methods for ordinary differential equations,* John Wiley & Sons, 2003.

[10] A.–H. Al-Rabeh, Optimal order diagonally implicit Runge-Kutta methods. *BIT* 33, 1993, pp. 620–633.

[11] H.–W. Tam, Two-stage parallel methods for the numerical solution of ordinary differential equations, *Siam J. Sci. Stat. Comput.* 13(5), 1992, pp. 1062–1084.

[12] L.–F. Shampine, *What everyone solving differential equations numerically should know,* in Computational Techniques for Ordinary Differential Equations, I. Gladwell and D.–K. Sayers, eds., pp. 1–17. Academic Press, 1980.

[13] R. Bronson, *Schaum's outline of modern introductory differential equations,* McGraw-Hill, 1973.

[14] J.–R. Dormand, *Numerical methods for differential equations: A computational approach,* CRC Press, Inc. 1996.