ADV MATH SCI JOURNAL

Advances in Mathematics: Scientific Journal **10** (2021), no.11, 3449–3460
ISSN: 1857-8365 (printed); 1857-8438 (electronic)
https://doi.org/10.37418/amsj.10.11.6

THREE POINTS BLOCK EMBEDDED DIAGONALLY IMPLICIT RUNGE-KUTTA METHOD FOR SOLVING ODES

Yong Faezah Rahim[1] and Mohd Ezad Hafidz Hafidzuddin

ABSTRACT. Block Embedded Diagonally Implicit Runge-Kutta (BEDIRK4(3)) method derived using Butcher analysis and equi-distribution of error approach is outperformed standard Runge-Kutta (RK) formulae. BEDIRK4(3) method produces approximation to the solution of initial value problem (IVP) at a block of three points simultaneously. The standard one step RK3(2) method is used to approximate the solution at the first point of the block. At the second points the solution is approximated using RK4(2) method which is generated by the previous research. The same approach is used to obtain the solution at the third point. The code for this method was built and the algorithm developed is suitable for solving stiff system. The efficiency of the method is supported by some numerical results.

1. INTRODUCTION

Diagonally Implicit Runge-Kutta (DIRK) methods are amongst the most popular method currently use for solving stiff systems of ordinary differential equations. DIRK method is attractive because of its low implementation cost, which mean that the computational effort involved is generally less than what is required by fully implicit RK method. Many Runge-Kutta codes are based on embedded pairs

[1]*corresponding author*

2020 *Mathematics Subject Classification.* 65L06.

Key words and phrases. Block method, stiff equations, embedded method, Runge-Kutta.

Submitted: 20.10.2021; *Accepted:* 12.11.2021; *Published:* 23.11.2021.

of explicit RK formula. For example, the code based on Dormand and Prince [1] with explicit embedded formula 6(5) method. Then the idea was extended to implicit method by Norsett and Thomson [2], Ismail and Suleiman [3], Butcher and Chen [4]. Ismail et al [5] has improved the embedded diagonally implicit Runge-Kutta method for solving delay differential equation.

A general $s-$stage Runge-Kutta method for the problem

$$y' = f(x, y),\ y(a) = \eta,\ f : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}^m \tag{1.1}$$

is defined by

$$Y_{n+1} = y_n + h \sum_{i=1}^{s} b_i k_i, \tag{1.2}$$

where

$$k_i = f\left(x_n + c_i h,\ y_n + h \sum_{i=1}^{s} b_i k_i\right),\ i = 1, 2, \ldots, s. \tag{1.3}$$

In order to present the embedded methods, we shall modify the Butcher array to the following form:

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

The method defined by c, A and b^T has order p and that defined by c, A and b^T has order $p + 1$. Define embedding method of (1.2) as

$$\bar{Y}_{n+1} = y_n + h \sum_{i=1}^{s} \hat{b}_i k_i. \tag{1.4}$$

Then the local truncation error (LTE) could be estimated using (1.2) and (1.4) and is given by vector E^T:

$$\text{LTE} = Y_{n+1} - \bar{Y}_{n+1} = h \sum_{i=1}^{s} \left(b_i - \hat{b}_i\right) k_i. \tag{1.5}$$

Traditionally, RK integration proceeds one step at a time, with several function evaluation in each step. There are many researchers have improved the existing methods or produced new methods that efficiently reducing the calculation cost.

An example of the well-known method that proves successfully to reduce the calculation cost is block method. This method capable of evaluating functions in set of several steps at once instead of one step at a time. By proceeding a block at a time, the number of function evaluation is reduced well below than attained by conventional RK method. The notion of using a block method has appeared explicitly in the literature. Cash [6] has derived a block diagonally implicit Runge-Kutta (BDIRK) method that can solve stiff and non-stiff problems of IVPs in ODEs. The derivation of BEDIRK4(2) can be obtained in Rahim [7]. A new block Runge-Kutta (NBRK) method with various weights for solving stiff ordinary differential equations (ODEs) were discussed by Aksah et al. [8]. Block backward differentiation formula (BBDF) of variable step were proposed by Ibrahim et al. [9] and Zawawi et al. [10] for solving stiff ODEs had proved to have better accuracy and smaller computational time.

2. Derivation of Embedded Block Method

The method that will be derived is produced approximation to the solution of the IVP at a block of three points simultaneously. The values of y_{n+1} and y_{n+2} are obtained using the two-point block method described by Rahim et al. [11]. Then this method is used to generate the new formulae for approximating solution at x_{n+3}. The solution at x_{n+1} and x_{n+2} is given as follows:

0.2928932	0.2928932			
1.0918831	0.7989899	0.2928932		
1.2928932	0.7407892	0.2592108	0.2928932	
	0.384776	1	0.615224	
2	0.903156	0	0.8039508	0.2928932
	0.716302	0.1624244	0.890524	0.2307496

By taking the same $[a_{ij}]$ and $[c_i]$ and normalizing the matrix by dividing all the coefficients by three since moving three steps forward, the following table is obtained:

0.0976311	0.0976311				
0.363961	0.26633	0.0976311			
0.430964	0.24693	0.0864036	0.0976311		
0.666667	0.301052	0	0.267984	0.0976311	
	$\hat{b}_1$	$\hat{b}_2$	$\hat{b}_3$	$1-\hat{b}_1-\hat{b}_2-\hat{b}_3$	
c_5	a_{51}	a_{52}	a_{53}	$c_5-a_{51}-a_{52}$ $-a_{53}-0.0976311$	0.0976311
	b_1	b_2	b_3	b_4	$1-b_1-b_2-b_3-b_4$

The method that is constructed is of order four with embedded method of order three. For the embedding method to have order three, the conditions to be followed are:

$$\sum \hat{b}_i c_i = \frac{1}{2} \tag{2.1}$$

$$\sum \hat{b}_i c_i^2 = \frac{1}{3} \tag{2.2}$$

$$\sum \hat{b}_i a_{ij} c_j = \frac{1}{6}. \tag{2.3}$$

Solving the above equations, then,

0.0976311	0.0976311			
0.363961	0.26633	0.0976311		
0.430964	0.24693	0.0864036	0.0976311	
0.666667	0.301052	0	0.267984	0.0976311
	0.463762	−0.795688	0.609366	0.72256

For fourth order accuracy, according to Butcher [12] the coefficients c_5, a_{51}, a_{52}, a_{53}, b_1, b_2, b_3, b_5 must satisfy the following eight order equations:

t	$\Phi(t) = \dfrac{1}{\gamma(t)}$ where $\gamma(t)$ = density of the tree
τ	$\sum b_i = 1$
$[\tau]$	$\sum b_i c_i = \frac{1}{2}$
$[\tau^2]$	$\sum b_i {c_i}^2 = \frac{1}{3}$
$[_2\tau]_2$	$\sum b_i a_{ij} c_j = \frac{1}{6}$
$[\tau^3]$	$\sum b_i {c_i}^3 = \frac{1}{4}$
$[\tau[\tau]]$	$\sum b_i c_i a_{ij} c_j = \frac{1}{8}$
$[_2\tau^2]_2$	$\sum b_i a_{ij} c_j^2 = \frac{1}{12}$
$[_3\tau]_3$	$\sum b_i a_{ij} a_{ik} c_k = \frac{1}{24}$

The trees associated with order four method can be divided into three categories:

a. τ, $[\tau]$, $[\tau^2]$, $[\tau^3]$
b. $[_2\tau]_2$, $[_2\tau^2]_2$, $[_3\tau]_3$
c. $[\tau[\tau]]$.

For t in category a), $\Phi(t) = \dfrac{1}{\gamma(t)}$ takes the form

$$\sum b_i c_i^{k-1} = \frac{1}{k}, \;\; k = 1, 2, 3, 4. \tag{2.4}$$

The three trees in category b): given

$$\sum_i b_i a_{ij} = b_j \left(1 - c_j\right), \;\; j = 2, 3, 4. \tag{2.5}$$

Then it follows that,

$$\sum_i b_i a_{ij} c_j = \sum b_j \left(1 - c_j\right) c_j = \frac{1}{2} - \frac{1}{3} = \frac{1}{6}, \tag{2.6}$$

$$\sum_i b_i a_{ij} c_j^2 = \sum b_j \left(1 - c_j\right) c_j^2 = \frac{1}{3} - \frac{1}{4} = \frac{1}{12}, \tag{2.7}$$

$$\sum_i b_i a_{ij} a_{jk} c_k = \sum b_j \left(1 - c_j\right) a_{jk} c_k = \frac{1}{6} - \frac{1}{8} = \frac{1}{24}. \tag{2.8}$$

It can be seen that the trees in category b) can be ignored if the remaining order conditions are satisfied.

A single tree in category c) yields the equation

$$\sum_{ij} b_i c_i a_{ij} c_j = \frac{1}{8}. \tag{2.9}$$

Therefore, for order four method with five stages, there are eight parameters with four constraints to be solved.

(1) $\sum b_i c_i = \frac{1}{2}$
(2) $\sum b_i c_i^2 = \frac{1}{3}$
(3) $\sum b_i c_i^3 = \frac{1}{4}$
(4) $\sum b_i c_i a_{ij} c_j = \frac{1}{8}$

The following choice of parameters from Billington [13] could lead to major reductions in computational effort. The choice is:

(1) $c_5 = c_1 + 1$
(2) $a_{51} = b_1$
(3) $a_{52} = b_2$
(4) $a_{53} = b_3$
(5) $a_{54} = b_4$
(6) $a_{55} = a_{11}$

Solving all the above equations, thus giving the following matrix.

$$\begin{array}{c|ccccc}
0.0976311 & 0.0976311 & & & & \\
0.363961 & 0.26633 & 0.0976311 & & & \\
0.430964 & 0.24693 & 0.0864036 & 0.0976311 & & \\
0.666667 & 0.301052 & 0 & 0.267984 & 0.0976311 & \\
1.0976311 & 0.463762 & -0.795688 & 0.609366 & 0.72256 & 0.0976311 \\
\hline
 & 0.307553 & -0.488833 & 0.741355 & 0.35846 & 0.081465
\end{array}$$

Finally, multiplying back all the coefficients by three then the formula for approximating the third point of the block is found.

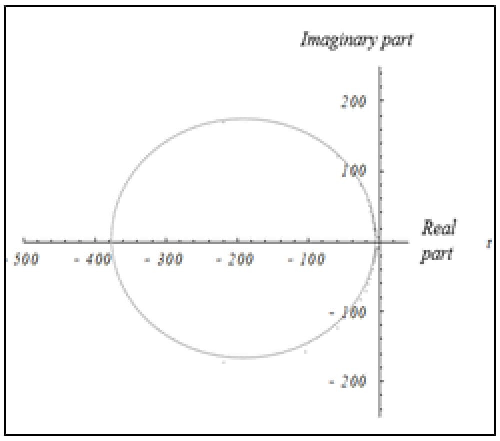
0.2928932	0.2928932				
1.0918831	0.7989899	0.2928932			
1.2928932	0.7407892	0.2592108	0.2928932		
2	0.903156	0	0.8039508	0.2928932	
	1.391286	-2.387064	1.828098	2.16768	
3.2928932	1.391286	-2.387064	1.828098	2.16768	0.2928932
	0.922659	-1.466499	2.224065	1.07538	0.244395

3. Stability of the method

The stability region of the method can be examined by applying the test equation $y' = \lambda y$ to the block formula. After substituting $y' = \lambda y$, such that $\hat{y} = h\lambda$ the stability polynomial is given as

$$\begin{aligned}\frac{y_{n+3}}{y_n} &= \frac{\det(I - \hat{h}A + \hat{h}ub^T)}{\det(I - \hat{h}A)} = R(\hat{h}) \\ &= \frac{-1.13892(23.6617 + \hat{h})(7.41279 + 2.045\hat{h} + \hat{h}^2)(2.32237 + 2.82722\hat{h} + \hat{h}^2)}{(-3.4105 + \hat{h})(11.6774 - 6.83444\hat{h} + \hat{h}^2)(11.649 - 6.82613\hat{h} + \hat{h}^2)}.\end{aligned}$$

The stability region can be defined by letting $|R(\hat{h})| < 1$, then plotting the stability polynomial and the result is shown below in figure below with the interval of absolute stability is approximately $(-350, 0)$. One of the criteria for a good method to be useful is that it must have a region of absolute stability.



4. Problem Tested

In this section, some of the problems obtained from Enright et al. [14] and Burden et al. [15] are tested upon. The problems are given as follows:

Problem 1:

$$\begin{aligned} &y' = -9y \\ &y(0) = 1 \\ &0 \leq x \leq 20 \end{aligned}$$

Problem 2:

$$\begin{aligned} &y' = \frac{50}{y} - 50y \\ &y(0) = \sqrt{2} \\ &0 \leq x \leq 20 \end{aligned}$$

Problem 3:

$$\begin{aligned} &{y_1}' = -y_1 + {y_2}^2 + {y_3}^2 + {y_4}^2 \\ &{y_2}' = -10y_2 + 10({y_3}^2 + {y_4}^2) \\ &{y_3}' = -40y_3 + 40{y_4}^2 \\ &{y_4}' = -100y_4 + 2 \\ &y_1(0) = y_2(0) = y_3(0) = y_4(0) = 1 \\ &0 \leq x \leq 20 \end{aligned}$$

Problem 4:

$$\begin{aligned} &{y_1}' = -0.5y_1 \\ &{y_2}' = -y_2 \\ &{y_3}' = -100y_3 \\ &{y_4}' = -90y_4 \\ &y_1(0) = y_2(0) = y_3(0) = y_4(0) = 1 \\ &0 \leq x \leq 20 \end{aligned}$$

5. Numerical Result and Conclusion

The table below show the performance of the BEDIRK 4(3) block method by solving the problems in the previous section using 10^{-2}, 10^{-4}, 10^{-6}. The results are compared to the results when the problems are solved using standard DIRK 3(2) non-block method. Since the method is an implicit method, thus iterations are needed to obtain the numerical solutions. Initially the system is considered as non-stiff and simple iterations are used, once there is an indication of stiffness, the whole system is considered as stiff and Newton iterations are used.

The following notation are used in the table:

- TOL: The chosen tolerance
- FCN: The number of function evaluations
- FSTEP: The number of failed step
- STEP: The number of successful steps
- JACO: The number of Jacobian evaluations
- EMAX: The maximum global error

1	Non-block method			Block method		
TOL	10^{-2}	10^{-4}	10^{-6}	10^{-2}	10^{-4}	10^{-6}
FCN	305	634	2083	154	274	754
STEPS	31	63	207	12	22	61
FSTEP	1	2	3	2	2	3
JACO	1	1	1	1	1	1
EMAX	$2.7819(-3)$	$4.2486(-5)$	$4.8411(-7)$	$5.2222(-3)$	$4.0603(-4)$	$1.4582(-6)$

2	Non-block method			Block method		
TOL	10^{-2}	10^{-4}	10^{-6}	10^{-2}	10^{-4}	10^{-6}
FCN	259	446	1210	145	269	474
STEPS	27	55	120	12	21	39
FSTEP	1	2	4	1	3	2
JACO	1	1	1	1	1	1
EMAX	$5.5878(-4)$	$7.7664(-5)$	$4.8828(-7)$	$2.9329(-3)$	$9.6266(-5)$	$1.2375(-6)$

3	Non-block method			Block method		
TOL	10^{-2}	10^{-4}	10^{-6}	10^{-2}	10^{-4}	10^{-6}
FCN	874	1140	3675	414	585	1339
STEPS	87	112	364	364	47	110
FSTEP	1	3	5	2	5	3
JACO	1	1	1	1	1	1
EMAX	$1.2225(-3)$	$4.2027(-5)$	$4.8234(-7)$	$8.0833(-3)$	$1.8789(-4)$	$2.1989(-6)$

4	Non-block method			Block method		
TOL	10^{-2}	10^{-4}	10^{-6}	10^{-2}	10^{-4}	10^{-6}
FCN	354	775	2915	217	566	1520
STEPS	38	86	311	18	46	123
FSTEP	1	3	6	1	4	7
JACO	1	1	1	1	1	1
EMAX	$2.1371(-3)$	$4.2509(-5)$	$7.8218(-7)$	$6.6224(-2)$	$1.6548(-4)$	$1.5090(-6)$

From the results, it is shown that block method is about $20\%-40\%$ more efficient than non-block method and perform better in terms of function evaluations and number of steps taken. It is also be seen that the method is more reliable at higher tolerance than that at lower tolerance because as the tolerance increases the differences of number of steps taken and function evaluation become more obvious between block method and non-block method. By using block method, the result requested at off-step points can be computed by interpolation. This gives a large saving in computational effort when many output points are specified. It is understood that the non-block method produced solution with better accuracy since it is approximating one point for each step, but block method can still has the accuracy within the given tolerance. These results are expected since block method approximate the numerical solution at three points simultaneously. Since the solution carried forward at the end of the block is order four, it is expected that the global error associated with block method to be rather small than standard method. The method is absolute stable with large stability region $(-350, 0)$ hence it is suitable for solving stiff problems. In general, the block method is performed better than conventional method because of less computational effort needed to complete the given task and the result are still within the required accuracy.

ACKNOWLEDGMENT

This work was supported by research grant (GP-IPM/2018/9619000) from Universiti Putra Malaysia, Malaysia.

REFERENCES

[1] J. DORMAND, P. PRINCE: *High order embedded runge-kutta formula*, J. Comput. Appl. Math **7** (1981), 67–75.

[2] S. NORSETT, P. THOMPSEN: *Embedded sdirk methods of basic order three*, BIT **24** (1984), 634–464.

[3] F. ISMAIL, M. SULEIMAN: *Embedded singly diagonally implicit runge-kutta method (4,5) in (5,6) for the integration of stiff systems of odes*, International Journal of Computer Math **66** (1998), 325–341.

[4] J. BUTCHER, D. CHEN: *A new type of singly-implicit runge-kutta method*, Applied Numerical Mathematics **34** (2000), 179–188.

[5] F. ISMAIL, A. RAED, M. SULEIMAN, M. HASSAN: *Embedded pair of diagonally implicit runge-kutta method for solving ordinary differential equations*, Sains Malaysiana **39**(6) (2010), 1049–1054.

[6] J. CASH: *Block embedded explicit runge-kutta methods*, Computers and Mathematics with Applications **11**(4) (1985), 395–409.

[7] Y. RAHIM: *Block diagonally implicit runge-kutta method for solving ordinary differential equations*, MSc thesis, Universiti Putra Malaysia, 2004.

[8] S. AKSAH, Z. IBRAHIM, Y. RAHIM, S. IBRAHIM: *Weighted block runge-kutta method for solving stiff ordinary differential equations*, Malaysian Journal of Mathematical Sciences **10**(3) (2016), 345–360.

[9] Z. IBRAHIM, K. OTHMAN, M. SULEIMAN: *Variable step size block backward differentiation formula for solving stiff odes*, in: Proceedings of World Academy of Science, Engineering and Technology, **2** (2007), 785–789.

[10] S. ZAWAWI, Z. IBRAHIM, F. ISMAIL, Z. MAJID: *Diagonally implicit block backward differentiation formulas for solving ordinary differential equations*, International Journal of Mathematics and Mathematical Sciences **2012** (2012), 1–8.

[11] Y. RAHIM, F. ISMAIL, M. SULEIMAN: *Block embedded diagonally implicit runge-kutta method of order four containing a formula of order two (4(2)) for solving ordinary differential equations*, Bulletin Science Putra **13** (2012), 1.

[12] J. BUTCHER: *Numerical Methods for Ordinary Differential Equations*, England: John Wiley & Sons, 2003.

[13] S. BILLINGTON: *Type insensitive codes for the solution of stiff and non-stiff systems of odes*, Master's thesis, University of Manchester, 1983.

[14] W. ENRIGHT, T. HULL, B. LINDBERG: *Comparing numerical methods for stiff systems of odes*, techreport 69, Department of Computer Science, University of Toronto, Canada, 1974.

[15] R. BURDEN, J. FAIRES, A. BURDEN: *Numerical Analysis (10th ed.)*, Brooks: Cole-Thomson Learning, 2015.

CENTRE OF FOUNDATION STUDIES FOR AGRICULTURAL SCIENCE,
UNIVERSITI PUTRA MALAYSIA,
43400 UPM SERDANG, SELANGOR,
MALAYSIA.
Email address: `yfaezah@upm.edu.my`

CENTRE OF FOUNDATION STUDIES FOR AGRICULTURAL SCIENCE,
UNIVERSITI PUTRA MALAYSIA,
43400 UPM SERDANG, SELANGOR,
MALAYSIA.
Email address: `ezadhafidz@upm.edu.my`