Embedded Singly Diagonally Implicit Runge-Kutta-Nyström General Method (3,4) in (4,5) for Solving Second Order IVPs

Fudziah Ismail, Raed Ali Al-Khasawneh, and Mohamed Suleiman

***Abstract*— Singly diagonally implicit Runge-Kutta-Nyström general (SDIRKNG) method of third-order embedded in fourth-order for the integration second-order IVPs is presented. A set of test problems are tested upon and the numerical comparisons with the existing embedded Runge-Kutta methods show the advantage of the new method.**

Keywords — Runge-Kutta-Nystóm, Second-order IVPs.

I. INTRODUCTION

Systems of second order ordinary differential equations (ODEs) arise naturally in many physical simulation problems, and the general form of the second order ODEs can be written as the following

$$y'' = f(x, y, y'), \quad x_0 \le x \le x_n \tag{1.1}$$

with the given initial conditions

$$y(x_0) = y_0, \; y'(x_0) = y'_0,$$

where $y \in \Re^n$, and $f : \Re \times \Re^n \times \Re^n \to \Re^n$. The function f is assumed to have derivative of arbitrary order everywhere in $\Re$. Equation (1.1) can be solved numerically by reducing it to system of first-order equations and then use embedded Runge-Kutta (RK) method such method can be seen in [1] and [2]. Or it can be solved directly using Runge-Kutta-Nyström (RKN) pairs as can be seen in [3]-[6]. These pairs generates approximations y_{n+1}, $\bar{y}_{n+1}$, y'_{n+1}, $\bar{y}'_{n+1}$ to

$y(x_{n+1})$ and $y'(x_{n+1})$ respectively, for n=0,1,…, according to

$$\left.\begin{aligned} y_{n+1} &= y_n + hy'_n + h^2 \sum_{i=1}^{q} b_i k_i, \quad y'_{n+1} = y'_n + h \sum_{i=1}^{q} b'_i k_i, \\ \bar{y}_{n+1} &= y_n + hy'_n + h^2 \sum_{i=1}^{q} \bar{b}_i k_i, \quad \bar{y}'_{n+1} = y'_n + h \sum_{i=1}^{q} \bar{b}'_i k_i, \end{aligned}\right\} \tag{1.2}$$

Manuscript received March 9, 2007.

Fudziah Ismail , Department of Mathematics, University Putra Malaysia, 43400, Serdang, Selangor, Malaysia. (e-mail: fudziah_i@ yahoo.com.my).

Raed Ali Al-Khasswneh, Amman, Jordan (e-mail: rkhassawneh@yahoo.com).

Mohamed Suleiman, Department of Mathematics, University Putra Malaysia, 43400, Serdang, Selangor, Malaysia. (e-mail: msuleiman@ putra.edu.my).

where the first two formulae are of order p, while the second two are of order p-1, q is the number of stages and

$$k_i = f(x_n + c_i h, y_n + c_i h y'_n + h^2 \sum_{j=1}^{i} a_{ij} k_j, y'_n + \sum_{j=1}^{i} a'_{ij} k_j), \quad i = 1, \ldots, q. \tag{1.3}$$

We refer to (1.2) as the generalized Runge-Kutta-Nyström (RKNG) pair.

Where the coefficients $a_{ij}, a'_{ij}, b_i, b'_i$ determine the method and they satisfy the following equations

$$\frac{1}{2} c_i^2 = \sum_{j=1}^{i} a_{ij}, \; (i = 1, \ldots, q), \tag{1.4}$$

$$\text{and } c_i = \sum_{j=1}^{i} a'_{ij}, \; (i = 1, \ldots, q). \tag{1.5}$$

The local truncation error (LTE) at the point x_n is given by

$$LTE = \left| y_n - \bar{y}_n \right|$$

and it is the basis for choosing the stepsize for the integration.

II. DERIVATION OF THE METHOD

Fine [5] listed the order conditions of RKNG method up to order 6 where the order conditions of y' up to order p-1 can be generated from the order conditions of y up to order p as follows:

If the y error coefficient of order i is

$$\tau_j^{(i)} = \psi(a, a', b', c) - \frac{1}{\varsigma},$$

then the corresponding y' coefficient of order $i-1$ can be expressed as

$$\tau'^{(i-1)}_j = \psi(a, a', b, c) - \frac{i}{\varsigma}.$$

Using this technique and Fine [4] work, we listed all the order conditions related to the method up to order four in Table 2.1 where the first four equations are related to y , the next equation (equation (2.5)) is related to both y and y' and the following eight equations are related to y'. For the (3,4), that is

third order four stage method, the first two equations of y and the first four equations of y' should be satisfied while for the (4,5), fourth order five stage method, all the 13 equations in Tables 2.1 should be satisfied.

Table 2.1: Equations of Conditions up to order 4.

Equations of Conditions for y	**Equations of Conditions for y'**
$\sum_i b_i = \frac{1}{2}$ (2.1)	$\sum_i b'_i = 1$ (2.6)
$\sum_i b_i c_i = \frac{1}{6}$ (2.2)	$\sum_i b'_i c_i = \frac{1}{2}$ (2.7)
$\sum_i b_i c_i^2 = \frac{1}{12}$ (2.3)	$\sum_i b'_i c_i^2 = \frac{1}{3}$ (2.8)
$\sum_{ij} b_i a_{ij} c_j = \frac{1}{24}$ (2.4)	$\sum_{ij} b'_i a'_{ij} c_j = \frac{1}{6}$ (2.9)
	$\sum_i b'_i c_i^3 = \frac{1}{4}$ (2.10)
Equations of Conditions for y and y'	$\sum_{ij} b'_i c_i a'_{ij} c_j = \frac{1}{8}$ (2.11)
$\sum_{ij} b'_i a_{ij} c_j^2 = \frac{1}{24}$ (2.5)	$\sum b'_i a'_{ij} c_j^2 = \frac{1}{12}$ (2.12)
	$\sum b'_i a'_{ij} a'_{jk} c_k = \frac{1}{24}$ (2.13)

There are 19 order conditions to be satisfied for (3,4) embedded in (4,5) method. To simplify the derivation, we started solving the order conditions which depend on a' using the simplifying assumptions and once we solved the order conditions for y', the order conditions for y can be solved together with the order condition that depend on both a and a' using the following transformation

$$b_i = (1-c_i)b'_i, \quad \bar{b}_i = (1-c_i)\bar{b'}_i, \quad i = 1,...,q. \qquad (2.14)$$

We now give the details on how to solve the equation for y' and y using the simplifying assumption. Equation (2.9) minus $\frac{1}{2}$ of (2.8) yields $\sum_i b'_i(\sum_{ij} a'_{ij} c_j - \frac{c_i^2}{2}) = 0$,thus

$$\sum_j^i a'_{ij} c_j = \frac{c_i^2}{2} \quad (i=2,3,4). \qquad (2.15)$$

Equation (2.15) is called the simplifying assumption. It does not hold for i=1, because we do not want $c_1 = a_{11} = 0$.

Therefore $b'_1 = 0$. Thus equation (2.9) can be removed, equation (2.10) minus $\frac{1}{2}$ of (2.11) gives

$$\sum_i b'_i c_i (\sum_{ij} a'_{ij} c_j - \frac{c_i^2}{2}) = 0,$$

Hence, equation (2.11) can also be removed.

Finally, equation (2.13) minus $\frac{1}{2}$ equation (2.12) gives

$$\sum_{ij} b'_i a'_{ij} (\sum_{jk} a'_{jk} c_k - \frac{c_j^2}{2}) = 0,$$

equation (2.13) is equivalent to (2.12) if

$$\sum_i b'_i a'_{i1} = 0. \qquad (2.16)$$

and (2.15) hold.

For the lower order method equation (2.15) cannot be satisfied for i=1, therefore we need to have

$$\bar{b'}_1 = 0.$$

From equation (1.5) and for i=1, we have $a'_{11} = c_1$, and for i=2, we get

$$c_2 - \gamma = a'_{21}. \qquad (2.17)$$

From equation (2.15) and for i=2, we obtain

$$a'_{21} c_1 + a'_{22} c_2 = \frac{c_2^2}{2}, \qquad (2.18)$$

substituting the value of a'_{21} from equation (2.17) into equation (2.18) yields

$$c_2 = \gamma(2 \pm \sqrt{2}). \qquad (2.19)$$

Now, we have 12 equations to be solved with 17 unknowns. Therefore, we have five free parameters which are chosen to be γ, c_3, c_4, c_5 and a'_{52}.

Taking $\gamma = 0.25, c_3 = 0.5, \quad c_4 = 0.75, \quad c_5 = 0.9$ and $a'_{52} = 0.25$ together with the values of $c_1 = \gamma$, $c_2 = \gamma(2-\sqrt{2})$, $b'_1 = 0$, $\bar{b}'_1 = 0$, we solved the system using Maple.

Then solve the order conditions for y as follows:

By using (2.14) , the vector weights b of y can be calculated using the vector weights b' of y'. After we found the weights of y, we are left with equation (2.4) and (2.5). There are six unknowns in the coefficient matrix of y, so the system of two equations can be solved with four free parameters which are chosen to be a_{42}, a_{43}, a_{52} and a_{53}. In Table 2.2, we present the coefficients for the SDIRKNG (3,4) embedded in (4,5) method using the value of $\gamma = 0.25$.

Table 2.2: SDIRK Method (3,4) Embedded in (4,5) with γ=0.25

γ	$\frac{\gamma^2}{2}$				
$(2-\sqrt{2})\gamma$	a_{21}	$\frac{\gamma^2}{2}$			
0.500000	a_{31}	−1.7720525	$\frac{\gamma^2}{2}$		
0.750000	a_{41}	−0.9500000	0.9500000	$\frac{\gamma^2}{2}$	
0.900000	a_{51}	0.7500000	−0.7000000	−0.7577296	$\frac{\gamma^2}{2}$
	0	0.2875266	0.1506343	0.0427529	
0.0190861					
	0	0.3333333	0.0285954	0.1380711	

γ	γ				
$(2-\sqrt{2})\gamma$	a'_{21}	γ			
0.500000	a'_{31}	0.6035533	γ		
0.750000	a'_{41}	−0.2004826	−0.2080426	γ	
0.900000	a'_{51}	0.2500000	0.7728468	−0.2996467	γ
	0	0.3368584	0.3012686	0.1710116	0.1908612
	0	0.3905249	0.0571909	0.5522847	

where the values of a_{i1} and a'_{i1} for (i=1(1)5) are given by $a_{i1} = \frac{1}{2}c_i^2 - \sum_{j=2}^{i} a_{ij}$, and $a'_{i1} = c_i - \sum_{j=2}^{i} a_{ij}$ respectively..

III. IMPLEMENTATION AND NUMERICAL RESULTS

The method derived in the previous section is used to solve the second order Initial value problems (IVPs). At the beginning of the program, the problem is considered as non stiff and therefore we do simple iterations, when there is a pointer of stiffness ($h_{acc} > h_{iter}$), then the whole system is automatically changed to stiff and solve using Newton iteration. Where h_{acc} is the largest stepsize that could achieved the desired local accuracy.

and h_{iter} is the largest stepsize for the iterations of the solutions to converge. The simple iteration on k_i is given by

$$
\begin{aligned}
{}^{(m)}k_i &= f(t_n + c_i h, y_n + hc_i y'_n + h^2 \sum_{j=1}^{i-1} a_{ij}\,{}^{(m)}k_j + h^2\gamma\,{}^{(m-1)}k_i, \\
&\quad y'_n + h\sum_{j=1}^{i-1} a'_{ij}\,{}^{(m)}k_j + h\gamma\,{}^{(m-1)}k_i)
\end{aligned}
$$

And the Newton iteration on k_i is given by ${}^{(m+1)}k_i = {}^{(m)}k_i + \Delta^{(m)}k_i$

$$
\begin{aligned}
&= f(\{y'_n + h\sum_{j=1}^{i-1} a'_{ij}\,{}^{(m+1)}k_j + ha'_{ii}({}^{(m)}k_i + \Delta^{(m)}k_i)\}, \\
&\qquad \{y_n + c_i h y'_n + h^2 \sum_{j=1}^{i-1} a_{ij}\,{}^{(m+1)}k_j \\
&\qquad + h^2 a_{ii}({}^{(m)}k_i + \Delta^{(m)}k_i)\})
\end{aligned}
\qquad (3.1)
$$

Expanding using Taylor series with two variables yields

$$
\begin{aligned}
{}^{(m)}k_i + \Delta^{(m)}k_i &= f(y'_n + h\sum_{j=1}^{i-1} a'_{ij}\,{}^{(m+1)}k_j + ha'_{ii}\,{}^{(m)}k_i, \\
&\quad y_n + c_i h y'_n + h^2 \sum a_{ij}\,{}^{(m+1)}k_j + h^2 a_{ii}\,{}^{(m)}k_i) + \\
&\quad ha'_{ii}\Delta^{(m)}k_i \frac{\partial f}{\partial y'} + h^2 a_{ii}\Delta^{(m)}k_i \frac{\partial f}{\partial y},
\end{aligned}
$$

collect all the terms involving $\Delta^{(m)}k_i$ we have

$$
\begin{aligned}
(1 - ha'_{ii}\frac{\partial f}{\partial y'} - h^2 a_{ii}\frac{\partial f}{\partial y})\Delta^{(m)}k_i &= f(y'_n + h\sum_{j=1}^{i-1} a'_{ij}\,{}^{(m+1)}k_i + ha'_{ii}\,{}^{(m)}k_i \\
&, y_n + c_i h y'_n + h^2 \sum a_{ij}\,{}^{(m+1)}k_j + h^2 a_{ii}\,{}^{(m)}k_i) - {}^{(m)}k_i .
\end{aligned}
$$

Thus since all the diagonal elements of A are equal $\left(\frac{\gamma^2}{2}\right)$ and diagonal elements of A' are also equal (γ), we obtain

$$
\begin{aligned}
\left(I - (h\gamma J' + h^2\frac{\gamma^2}{2}J)\right)\Delta^{(m)}k_i &= f(y'_n + h\sum_{j=1}^{i-1} a'_{ij}\,{}^{(m+1)}k_i + ha'_{ii}\,{}^{(m)}k_i, \\
y_n + c_i h y'_n + h^2 \sum a_{ij}\,{}^{(m+1)}k_j + h^2 a_{ii}\,{}^{(m)}k_i) &- {}^{(m)}k_i . \qquad (3.2)
\end{aligned}
$$

In this paper we test a few problems consisting of stiff and nonstiff problems and equation (3.2) is the coefficient matrix for the Newton iterations.

The following are some of the problems tested. Note that the third and fourth problems are stiff IVPs of second order while the first two problems are non-stiff second order IVPs.

Problem 3.1 $y''_1 = -y'_2$, $y_1(0) = 0$, $y'_1(0) = \frac{1}{1-e^{-1}}$,

$y''_2 = -y'_1$, $y_2(0) = 1$, $y'_2(0) = \frac{1}{1-e^{-1}}$,

$0 \le x \le 10$,

Solution: $y_1(x) = \frac{1-e^{-x}}{1-e^{-1}}$, $y_2(x) = \frac{2-e^{-1}-e^{-x}}{1-e^{-1}}$

Source: Edwards Jr and Penny [7].

Problem 3.2 : $y_1'' = -4x^2 y_1 + \dfrac{2y_1'}{r_1 r_2}, \quad x \geq x_0,$

$$y_2'' = -4x^2 y_2 + \frac{2y_2'}{r_1 r_2},\ x \geq x_0,$$

$$\sqrt{0.5\pi} \leq x \leq 10 \quad y_1(x_0) = 0,\ y_2(x_0) = 1,$$

$$y_1'(x_0) = -(2\pi)^{\frac{1}{2}}, \quad y_2'(x_0) = 0,$$

$$r_1 = \sqrt{y_1'^2 + y_2'^2} \text{ and } r_2 = \sqrt{y_1^2 + y_2^2}$$

Solution: $y_1(x) = \cos(x^2)$, $y_2(x) = \sin(x^2)$.

Source: Sharp and Fine [8]

Problem 3.3: $y'' + 8y' + ky = 0$, $0 \leq x \leq 10$,

$$y(0) = 1, \quad y'(0) = -12, \quad k = 16.$$

Solution: $y(x) = (1 - 8x)e^{-4x}$.

Problem 3.4.

$$y'' + 6y' + 9y = 0, \quad y(0) = 1, \ y'(0) = -3, \quad 0 \leq x \leq 20$$

Solution: $y(x) = e^{-3x}$.

The results obtained from the new method which was derived in section 2 are compared with the results when the same problems are solved using singly diagonally implicit Runge-Kutta (SDIRK) method (3,4) embedded in (4,5) which was derived by Hairer and Wanner [9] and SDIRK method (3,5) embedded in (4,6) which was derived by Butcher and Chen [10]. In the SDIRK method the problems are reduced to first-order system of differential equation twice the dimension by considering the vector (y, y') as the new variables. All the methods are of the same order .

The numerical results are given in Tables 3.1-3.4. The following notations are used as follows:

TOL ~ the chosen tolerance,
MTD ~ method used,
FCN ~ the number of functions evaluations,
STEP ~ the number of successful steps,
FSTP ~ the number of failed steps,
JAC ~ the number of Jacobian evaluations.
MAX ERR~ max $|y(t_i) - y_{t_i}|$, (absolute value of the true solution minus the computed solution at the mesh point *i*).

where 1.234567(-6) means 1.23456 x 10^{-6}.

Methods used are:

A1: SDIRKN method (3,4) embedded in (4,5) which was derived in this paper where the local truncation error is $\left\| y_n - \bar{y}_n \right\|$.

A2: SDIRK method (3,4) embedded in (4,5) by Hairer and Wanner .

A3: SDIRK method (3,5) embedded in (4,6) by Butcher and Chen.

Table 3.1: Numerical Results for Problem 3.1.

TOL	MTD	FCN	STEP	FSTP	MAXERR
10^{-2}	A1	358	32	1	9.095466(-3)
	A2	288	52	0	1.015307(-3)
	A3	332	49	0	1.161654(-3)
10^{-4}	A1	424	30	1	3.081559(-4)
	A2	464	84	0	3.521043(-5)
	A3	574	144	0	2.584102(-4)
10^{-6}	A1	1062	96	1	1.395924(-9)
	A2	1267	230	0	6.260004(-9)
	A3	938	144	0	1.183762(-9)
10^{-8}	A1	3288	298	2	4.366451(-5)
	A2	3808	692	0	4.495573(-8)
	A3	3308	505	3	1.583404(-6)
10^{-10}	A1	9221	837	3	4.318437(-6)
	A2	16183	2942	0	4.130849(-9)
	A3	9208	1411	6	9.936897(-8)

Table 3.2: Numerical Results for Problem 3.2

TOL	MTD	FCN	STEP	FSTP	MAXERR
10^{-2}	A1	2275	198	1	5.707671(-2)
	A2	2817	506	6	1.739446(-1)
	A3	3048	468	1	2.115526(-1)
10^{-4}	A1	7079	643	1	5.560518(-3)
	A2	8342	1516	1	7.176503(-3)
	A3	7780	1196	1	1.400480(-2)
10^{-6}	A1	22654	2058	2	4.281767(-4)
	A2	26426	1516	1	2.316212(-4)
	A3	19596	3014	1	8.788254(-4)
10^{-8}	A1	71310	6481	2	8.019036(-5)
	A2	85122	15476	1	4.845471(-5)
	A3	49289	7582	1	7.205344(-5)
10^{-10}	A1	223397	20307	3	1.309299(-6)
	A2	1844388	335342	1	4.846149(-5)
	A3	123896	19060	1	4.844024(-5)

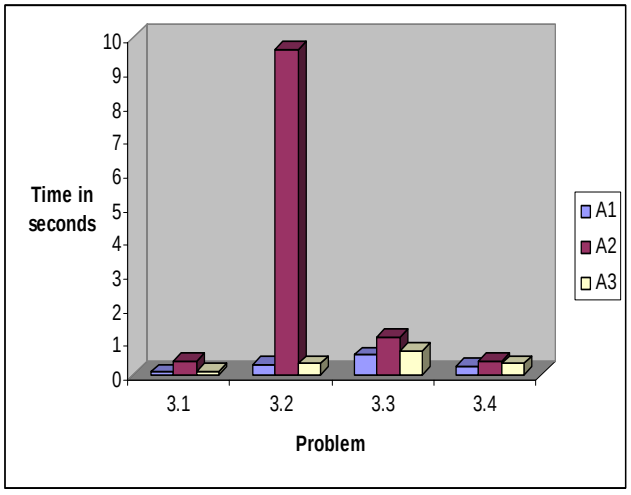
Table 3.3: Numerical Results for Problem 3.3.

TOL	MTD	FCN	STEP	FSTP	JAC	MAXERR
10^{-2}	A1	556	98	1	Not	5.675743(-4)
	A2	634	112	1	stiff	3.447152(-4)
	A3	490	64	1		3.390885(-4)
10^{-4}	A1	1094	124	2	1	4.045746(-5)
	A2	1142	204	2	1	1.526613(-5)
	A3	1317	188	3	1	2.735042(-5)
10^{-6}	A1	2770	366	2	1	3.523178(-6)
	A2	2935	530	2	1	1.668552(-7)
	A3	3618	539	3	1	1.703345(-6)
10^{-8}	A1	7943	911	2	1	4.678987(-8)
	A2	11207	1434	2	1	1.594997(-8)
	A3	8220	1240	3	1	1.061384(-7)
10^{-10}	A1	17023	2089	4	1	2.345620(-9)
	A2	40191	7302	3	1	4.495724(-9)
	A3	20435	3069	5	1	6.663856(-9)

Table 3.4: Numerical Results for Problem 3.4

TOL	MTD	FCN	STEP	FSTP	JAC	MAXERR
10^{-2}	A1	997	87	2	1	3.314479(-2)
	A2	447	77	1	1	6.743232(-4)
	A3	239	27	0	1	2.494116(-3)
10^{-4}	A1	1404	124	2	1	4.042037(-4)
	A2	777	137	5	1	3.837458(-5)
	A3	1066	153	2	1	3.184944(-5)
10^{-6}	A1	2970	266	2	1	3.508986(-5)
	A2	1877	337	1	1	3.837458(-5)
	A3	2682	395	3	1	2.367550(-6)
10^{-8}	A1	7865	711	5	1	3.427374(-6)
	A2	5445	913	1	1	5.824995(-8)
	A3	7439	1078	3	1	3.133465(-8)
10^{-10}	A1	23023	2089	5	1	6.934120(-7)
	A2	149596	27195	1	1	7.538816(-10)
	A3	27877	4224	7	4	8.910569(-10)

Figure 3.1: Time taken by methods A1, A2 and A3 to solve the problems over all the tolerances



IV. CONCLUSION

From the numerical results, we noticed that for all the problems except problem 3.2, method RKNG (3,4) embedded in (4,5) (method A1) gives better results in terms of function evaluations, number of steps and total time taken to solve the problems compared to Hairer's (method A2) and Butcher's method (method A3).

Comparing A2 and A3 we observed that for most of the problems A3 performed better than A2 and in terms of number of steps, functions evaluations and total time taken over all the tolerances. In terms of absolute error method A3 produced the smallest error compared to A1 and A2.

Here we can conclude that the new method can be used to solve both stiff and non-stiff general second order IVPs directly without having to reduce the problems to first order system hence less time is needed to solve the problems.

REFERENCES

[1] J. R. Dormand and P. J. Prince, A family of embedded Runge-Kutta formulae. *J. Comput. Appl. Maths.*, (1980), 6(1): 19-26.

[2] J. H. Verner, Explicit Runge-Kutta methods with estimates of the local truncation error. *SIAM J. Numer. Anal.* (1978) 15(4), 67-75.

[3] E.Fehlberg, Classical seventh-, sixth-, and fifth-order Runge-Kutta-Nyström formulas with stepsize control for general second-order differential equations. *NASA Technical Report R*-432. (1974).

[4] J. M. Fine,. *Runge-Kutta-Nyström methods for the general second order ordinary differential equations*, Ph.D. Dissertation, Univ. Texas, Austin (1984).

[5] J. M. Fine, *Low order Runge-Kutta-Nyström methods with interpolants*. Department of Computer Science Technical report No. 183/85, University of Toronto (1985).

[6] J. M. Fine, Low Order Practical Runge-Kutta-Nyström Methods. *Computing by Springer-Verlag* (1987), 38: 281-297.

[7] C. H.Edwards Jr, and D,. E. Penny, *Elementary Differential Equations with Boundary Value Problem.* Prentice-Hall, Englewood Cliffs, New Jersey.(1993).

[8] P. W. Sharp, and J. M. Fine, . Some Nyström pairs for the general second-order initial-value problem. *Journal of Comut. And Appl. Math.*,(1992) 42: 279-291.

[9] E. Hairer, and G. Wanner, *Solving Ordinary Differential Equations II, stiff and Differential Algebraic Problems*, Berlin: Springer-Verlag (1991).

[10] J. C. Butcher, and D. J. L. Chen. A new type of singly-implicit Runge-Kutta method, *Applied Numerical Mathematics*, (2000), 34: 179–188.