axioms

Article

Development of an Efficient Diagonally Implicit Runge–Kutta–Nyström 5(4) Pair for Special Second Order IVPs

Musa Ahmed Demba [1,2,3,†], Norazak Senu [4,†], Higinio Ramos [5,6,*,†] and Wiboonsak Watthayu [7,†]

1 KMUTTFixed Point Research Laboratory, KMUTT-Fixed Point Theory and Applications Research Group, Department of Mathematics, Faculty of Science, King Mongkut's University of Technology Thonburi (KMUTT), 126 Pracha-Uthit Road, Bang Mod, Thung Khru, Bangkok 10140, Thailand
2 Center of Excellence in Theoretical and Computational Science (TaCS-CoE), Science Laboratory Building, King Mongkut's University of Technology Thonburi (KMUTT), 126 Pracha-Uthit Road, Bang Mod, Thung Khru, Bangkok 10140, Thailand
3 Department of Mathematics, Faculty of Computing and Mathematical Sciences, Kano University of Science and Technology, Wudil P.M.B 3244, Nigeria
4 Department of Mathematics & Statistics and Institute for Mathematical Research, Universiti Putra Malaysia, Serdang 43400, Malaysia
5 Department of Applied Mathematics, Faculty of Sciences, University of Salamanca, 37008 Salamanca, Spain
6 Escuela Politécnica Superior, Avda. de Requejo, 33, 49022 Zamora, Spain
7 Department of Mathematics, Faculty of Science, King Mongkut's University of Technology Thonburi (KMUTT), 126 Pracha-Uthit Road, Bang Mod, Thung Khru, Bangkok 10140, Thailand
* Correspondence: higra@usal.es
† These authors contributed equally to this work.

check for updates

Citation: Demba, M.A.; Senu, N.; Ramos, H.; Watthayu, W. Development of an Efficient Diagonally Implicit Runge–Kutta–Nyström 5(4) Pair for Special Second Order IVPs. *Axioms* **2022**, *11*, 565. https://doi.org/10.3390/axioms11100565

Academic Editors: Zacharias A. Anastassi and Patricia J. Y. Wong

Received: 27 August 2022
Accepted: 13 October 2022
Published: 18 October 2022

Abstract: In this work, a new pair of diagonally implicit Runge–Kutta–Nyström methods with four stages is constructed. The proposed method has been derived to solve initial value problems of special second-order ordinary-differential equations. The principal local truncation error of the new method is obtained, and the main characteristics of the new method are analyzed. Some numerical experiments are performed, which demonstrate the robustness and efficiency of the new embedded pair.

Keywords: diagonally implicit methods; embedded pairs; Runge–Kutta–Nyström; oscillatory problems; initial value problems

MSC: 65L05; 65L06

1. Introduction

A lot of methods have been constructed to solve numerically the initial value problem (IVP) of the special second order ordinary differential equation of the form

$$y'' = f(x, y), \qquad y(x_0) = y_0, \qquad y'(x_0) = y'_0, \tag{1}$$

where $y \in \Re^d$ and $f : \Re \times \Re^d \to \Re^d$ are assumed to be sufficiently differentiable. Problem (1) is usually found in different areas such as fluid mechanics, quantum and physical chemistry, astronomy, and many others. The class of Runge–Kutta–Nyström (RKN) codes has been usually considered to obtain approximate solutions to problem (1). Regarding such usage, different methods of the class of diagonally implicit RKN methods have been studied by Sommeijer in [1], Van der Houwen and Sommeijer in [2], Imoni et al. in [3], Sharp et al. in [4], Senu et al. in [5–8], Papageorgiou et al. in [9], Al-khasawneh et al. in [10], and Ismail et al. in [11]. In this study, we develop a new efficient 5(4) diagonally implicit RKN method (DIRKN) in variable step-size to solve the problem in (1). To the best of our knowledge, the derived method in this paper is the only DIRKN5(4) embedded pair that is correctly constructed in the literature. The only one currently available in the literature is the one developed by Imoni et al. in [12], but it was wrongly constructed. This is because, substituting the coefficients of the method in the order conditions for a RKN method up to order five, some of the order conditions fail to be satisfied. The proposed method can solve accurately the usual test equation: $y'' = -w^2 y$. The numerical experiments bring out the performance of the developed method compared to some diagonally implicit embedded RKN codes in the literature.

The rest of the paper is organized in this way: Section 2 gives a detailed explanation of the diagonally implicit RKN pairs. Section 3 focuses on the development of the new method and provides details on the stability properties of the developed pair. Some numerical experiments are presented in Section 4. A detailed explanation on the results obtained is given in Section 5, and finally, in Section 6 we give a conclusion.

2. Basic Concepts

A RKN method with r-stages for solving the problem in (1) is generally expressed by the formulas:

$$y_{n+1} = y_n + hy'_n + h^2 \sum_{l=1}^{r} b_l f(x_n + c_l h, Y_l), \tag{2}$$

$$y'_{n+1} = y'_n + h \sum_{l=1}^{r} d_l f(x_n + c_l h, Y_l), \tag{3}$$

$$Y_l = y_n + c_l h y'_n + h^2 \sum_{j=1}^{r} a_{lj} f(x_n + c_j h, Y_j), \tag{4}$$

where, as usual, y_{n+1} and y'_{n+1} represent approximate values of $y(x_{n+1})$ and $y'(x_{n+1})$ respectively, and $x_{n+1} = x_n + h$, $n = 0, 1, \ldots, h$ being the stepsize.

The above method may be formulated using the Butcher array, in the form

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

being $A = (a_{ij})_{r\times r}$ a matrix of coefficients, $c = (c_1, c_2, \ldots, c_r)^T$ is the vector of stages, and $b = (b_1, b_2, \ldots, b_r)$ and $d = (d_1, d_2, \ldots, d_r)$ contain the remaining coefficients of the formulas in (2) and (3). A brief notation for this method is (c, A, b, d).

A RKN method can either be explicit or implicit. It is said to be explicit if A is a strictly lower triangular matrix; otherwise, it is called implicit. An implicit RKN method is said to be diagonally implicit if the matrix A is lower triangular, and the diagonal entries are equal, i.e., $a_{ij} = 0$, for $i < j$, and $a_{ii} = \delta$.

A $m(n)$ pair of embedded RKN methods is formed by a method (c, A, b, d) with order m and another one $(c, A, \hat{b}, \hat{d})$ with order $n < m$, where both methods share the coefficients in c and A. The method of higher order provides approximate values y_{n+1}, y'_{n+1}, and the method of lower order provides approximate values $\hat{y}_{n+1}, \hat{y}'_{n+1}$. The second approximation is used to obtain an estimate of the local truncation error.

An embedded-type pair of RKN methods may be given by means of the Butcher tableau:

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

In this paper, we consider a variable step-size approach based on the local error estimate obtained through the embedding procedure. The local error estimate at the point $x_{n+1} = x_n + h$ is provided through the differences $\eta_{n+1} = \hat{y}_{n+1} - y_{n+1}$ and $\eta'_{n+1} = \hat{y}'_{n+1} - y'_{n+1}$.

Let $\mathrm{Est}_{n+1} = \max(\|\eta_{n+1}\|_\infty, \|\eta'_{n+1}\|_\infty)$ be the local error approximation to manage the step-length on each step. To advance the solution, we consider the step-length control strategy given in [13]:

$$h_{n+1} = \frac{9}{10}\left(\frac{Tol}{Est_{n+1}}\right)^{\frac{1}{m+1}} \tag{5}$$

Tol being the tolerance selected by the user, and $\frac{9}{10}$ a safety factor. If $Est_{n+1} < Tol$, then the step is accepted, and we continue with the procedure by performing local extrapolation, meaning that the more accurate approximation will be used to advance the integration. If $Est_{n+1} \geq Tol$, then the computations at the current step are rejected, and the step size will be updated using the formula in (5).

3. Development of the New Pair

In this section, we will derive the DIRKN5(4)4D, which is a new diagonally implicit 5(4) embedded pair of constant coefficients with four stages.

To achieve this, the order conditions for a RKN method in Equations (2)–(4) up to order five, as derived in [14], together with some simplifying assumptions as given below must be considered (see also [15,16]). Although, to derive our method, we will only consider conditions up to order 5 for the solution and the derivative, we give below the order conditions up to order six.

Order conditions for y:

$$Order2: \quad \sum b_i = \frac{1}{2}, \tag{6}$$

$$Order3: \quad \sum b_i c_i = \frac{1}{6}, \tag{7}$$

$$Order4: \quad \frac{1}{2}\sum b_i c_i^2 = \frac{1}{24}, \tag{8}$$

$$\begin{aligned} Order5: \quad & \frac{1}{6}\sum b_i c_i^3 = \frac{1}{120}, \\ & \sum b_i a_{ij} c_j = \frac{1}{120}, \end{aligned} \tag{9}$$

$$\begin{aligned} Order6: \quad & \frac{1}{24}\sum b_i c_i^4 = \frac{1}{720}, \\ & \frac{1}{4}\sum b_i c_i a_{ij} c_j = \frac{1}{720}, \\ & \frac{1}{2}\sum b_i a_{ij} c_j^2 = \frac{1}{720}. \end{aligned} \tag{10}$$

Order conditions for y':

$$\begin{aligned}
&Order1: && \sum d_i = 1, && (11)\\
&Order2: && \sum d_i c_i = \frac{1}{2}, && (12)\\
&Order3: && \frac{1}{2}\sum d_i c_i^2 = \frac{1}{6}, && (13)\\
&Order4: && \frac{1}{6}\sum d_i c_i^3 = \frac{1}{24}, &&\\
& && \sum d_i a_{ij} c_j = \frac{1}{24}, && (14)\\
&Order5: && \frac{1}{24}\sum d_i c_i^4 = \frac{1}{120}, &&\\
& && \frac{1}{4}\sum d_i c_i a_{ij} c_j = \frac{1}{120}, &&\\
& && \frac{1}{2}\sum d_i a_{ij} c_j^2 = \frac{1}{120}, && (15)\\
&Order6: && \frac{1}{120}\sum d_i c_i^5 = \frac{1}{720}, &&\\
& && \frac{1}{20}\sum d_i c_i^2 a_{ij} c_j = \frac{1}{720}, &&\\
& && \frac{1}{10}\sum d_i c_i a_{ij} c_j^2 = \frac{1}{720}, &&\\
& && \frac{1}{6}\sum d_i a_{ij} c_j^3 = \frac{1}{720}, &&\\
& && \sum d_i a_{ij} a_{jk} c_k = \frac{1}{720}. && (16)
\end{aligned}$$

All subscripts i, j, k vary from 1 to r. Most DIRKN methods require the c_i to satisfy the following condition (see [8]):

$$\frac{1}{2}c_i^2 = \sum_{j=1}^{r} a_{ij},\ (i = 1, 2, \ldots, r). \quad (17)$$

For a higher order RKN method, a simplifying assumption given in [17] is usually used to reduce the number of order conditions as given by the following equation:

$$b_i = d_i(1 - c_i),\ (i = 1, 2, \ldots, r). \quad (18)$$

To obtain the fifth-order method of the embedded pair, we consider the system of equations formed by the order conditions up to order 5, together with the simplifying assumptions given in Equations (17) and (18). This gives a system of 21 equations with 18 unknowns. If we fixed $c_1 = \frac{1}{10}$, and take c_3 and c_4 as free parameters, we obtain a two-parameter family of methods.

As simple values, we chose $c_3 = \frac{7}{10}$, and $c_4 = 1$, and, therefore, we obtain the coefficients of the fifth-order four-stage DIRKN method of the embedded DIRKN5(4)D pair, as given below:

$$a_{11} = a_{22} = a_{33} = a_{44} = \frac{1}{200},\ a_{21} = \frac{91}{1800},\ a_{31} = \frac{4143}{35000},\ a_{32} = \frac{4257}{35000},\ a_{41} = \frac{11061}{43400},$$

$$b_1 = \frac{25}{126},\ b_2 = \frac{27}{154},\ b_3 = \frac{25}{198},\ b_4 = 0,\ c_2 = \frac{1}{3},\ c_3 = \frac{7}{10},\ c_4 = 1,\ d_1 = \frac{125}{567},$$

$$d_2 = \frac{81}{308}, d_3 = \frac{125}{297},\ d_4 = \frac{31}{324},\ a_{42} = \frac{4644}{59675},\ a_{43} = \frac{1107}{6820}.$$

The coefficients of the principal terms of the local truncation errors (PLTE) of the main formulas of the above method to approximate the solution and the derivative are $||\tau^{(6)}|| = 9.56 \times 10^{-4}$ and $||\tau'^{(6)}|| = 1.09 \times 10^{-3}$, respectively.

To derive the fourth-order method to form the embedded pair, we utilized the coefficients of the lower triangular matrix A and the vector c of the fifth-order method derived above. Considering the system of equations formed by the order conditions up to order 4, together with the simplifying assumption as given in Equation (17) for $r = 4$; this gives a system of 12 equations with 8 unknowns. Taking b_4 as a free parameter and solving this system, we obtain a one-parameter family whose coefficients are

$$b_1 = \frac{25}{126} - \frac{10b_4}{7},\ b_2 = \frac{27}{1544} + \frac{243b_4}{77},\ b_3 = \frac{25}{198} - \frac{30b_4}{11},$$
$$d_1 = \frac{125}{567},\ d_2 = \frac{81}{308},\ d_3 = \frac{125}{297},\ d_4 = \frac{31}{324}.$$

Taking $b_4 = \frac{1}{2}$, we obtain the coefficients of the fourth-order four-stage RKN method of the embedded pair 5(4)D as given below, with the coefficients of A and c shared by both methods

$$b_1 = -\frac{65}{126},\ b_2 = \frac{135}{77},\ b_3 = -\frac{245}{198},\ d_1 = \frac{125}{567},\ d_2 = \frac{81}{308},\ d_3 = \frac{125}{297},\ d_4 = \frac{31}{324}.$$

The coefficients of the principal terms of the local truncation errors (PLTE) of the main formulas of the above method to approximate the solution and the derivative are $||\tau^{(5)}|| = 2.43 \times 10^{-2}$ and $||\tau'^{(5)}|| = 8.33 \times 10^{-3}$, respectively.

The coefficients of the newly developed DIRKN5(4)4D are collected in Table 1.

Table 1. Coefficients of the DIRKN5(4)4D method.

$\frac{1}{10}$	$\frac{1}{200}$			
$\frac{1}{3}$	$\frac{91}{1800}$	$\frac{1}{200}$		
$\frac{7}{10}$	$\frac{4143}{35000}$	$\frac{4257}{35000}$	$\frac{1}{200}$	
1	$\frac{11061}{43400}$	$\frac{4644}{59675}$	$\frac{1107}{6820}$	$\frac{1}{200}$
	$\frac{25}{126}$	$\frac{27}{154}$	$\frac{25}{198}$	0
	$\frac{125}{567}$	$\frac{81}{308}$	$\frac{125}{297}$	$\frac{31}{324}$
	$-\frac{65}{126}$	$\frac{135}{77}$	$-\frac{245}{198}$	$\frac{1}{2}$
	$\frac{125}{567}$	$\frac{81}{308}$	$\frac{125}{297}$	$\frac{31}{324}$

Stability Analysis

Applying the newly developed DIRKN5(4)4D method to the test equation $y'' = -w^2 y$, the linear stability is derived, and letting $\tilde{h} = v^2 = w^2h^2$, the approximate solution verifies the recurrence equation

$$L_{n+1} = E(\tilde{h})L_n,$$

where

$$L_{n+1} = \begin{bmatrix} y_{n+1} \\ hy'_{n+1} \end{bmatrix},\ L_n = \begin{bmatrix} y_n \\ hy'_n \end{bmatrix},\ E(\tilde{h}) = \begin{bmatrix} 1 - \tilde{h}b^T N^{-1} e & 1 - \tilde{h}b^T N^{-1} c \\ -\tilde{h}d^T N^{-1} e & 1 - \tilde{h}d^T N^{-1} c \end{bmatrix},$$

being $N = I + \tilde{h}A$ with $A = \left(a_{ij}\right)_{4\times 4}$ the matrix of coefficients, I the identity matrix of dimension four, and

$$b = (b_1, b_2, b_3, b_4)^T,\ d = (d_1, d_2, d_3, d_4)^T,\ e = (1, 1, 1, 1)^T,\ c = (c_1, c_2, c_3, c_4)^T,$$

vectors of coefficients. It is assumed that, for sufficiently small values of $v = wh$, the eigenvalues of $E(\tilde{h})$ are complex conjugates [2]. Under this assumption, an oscillatory numerical solution should be obtained. The oscillatory character depends on the eigenvalues of the stability matrix $E(\tilde{h})$. The characteristic equation of this matrix can be expressed as:

$$\lambda^2 - tr(E(\tilde{h}))\lambda + det(E(\tilde{h})) = 0. \tag{19}$$

Definition 1. *Given the method in (2)–(4), an interval $I = (-\tilde{h}_s, 0)$ is said to be an interval of absolute stability if for all $\tilde{h} \in I$, it is $|\lambda_{1,2}| < 1$, where $\lambda_{1,2}$ are the solutions of the equation in (19).*

Definition 2. *An interval $(-\tilde{h}_p, 0)$ corresponding to the RKN method in Equations (2)–(4) is said to be an interval of periodicity if for every $\tilde{h} \in (-\tilde{h}_p, 0)$, $|\lambda_{1,2}| = 1$, with $\lambda_1 \neq \lambda_2$, where $\lambda_{1,2}$ are the roots of the equation in (19).*

The following result can be readily obtained using the above definitions and any computer algebra system, such as the Maple package.

Proposition 1. *The higher-order method of the new embedded pair DIRKN5(4)4D has an interval of absolute stability $(-9.42, 0)$, while the lower-order method of the new embedded pair DIRKN5(4)4D has an interval of periodicity $(-2.40, 0)$.*

4. Some Examples

To assess the performance of the proposed method, we will consider some well known pairs of DIRKN methods appeared in the literature for numerical comparisons:

- DIRKN5(4)4D: The constructed DIRKN embedded pair in this paper;
- DIRKN4(3)R: The embedded DIRKN 4(3) pair derived in [10];
- DIRKN4(3)I: The 4(3) embedded DIRKN pair derived in [3];
- DIRKN4(3)N: The 4(3) pair of embedded DIRKN methods derived by Senu et al. in [8].

The above methods will be used to solve some well-known oscillatory IVPs. They have been implemented in the C programing environment using a PC with 2.30 GHz processor, Intel(R) core(TM) i3-7020U CPU, and 12.0 GB of RAM:

Example 1 (*The Model Problem in* [18])**.** *The first example is the test equation problem*

$$y'' \;=\; -25y,\; y(0) = 0,\; y'(0) = 5, x \in [0, 10],$$

whose exact solution is given by

$$y(x) = \sin(5x).$$

Example 2 (*The Orbital Problem in* [19])**.**

$$\begin{aligned} y_1'' &= -y_1 + \frac{1}{1000}\cos(x),\; y_1(0) = 1,\; y_1'(0) = 0, \\ y_2'' &= -y_2 + \frac{1}{1000}\sin(x),\; y_2(0) = 0,\; y_2'(0) = \frac{9995}{10000}, x \in [0, 10]. \end{aligned}$$

The exact solution is

$$y_1(x) = \cos(x) + \tfrac{1}{2000}\, x\sin(x),$$

$$y_2(x) = \sin(x) - \tfrac{1}{2000}\, x\cos(x).$$

Example 3 (*A Nonlinear System in* [20]).

$$\begin{aligned} y_1'' + w^2 y_1 &= \frac{2y_1y_2 - \sin(2wx)}{({y_1}^2 + {y_2}^2)^{\frac{3}{2}}}, y_1(0) = 1, y_1'(0) = 0, \\ y_2'' + w^2 y_2 &= \frac{{y_1}^2 - {y_2}^2 - \cos(2wx)}{({y_1}^2 + {y_2}^2)^{\frac{3}{2}}}, y_2(0) = 0, y_2'(0) = w, \quad x \in [0, 10], \end{aligned}$$

with a known solution given by

$$y_1(x) = \cos(wx),$$

$$y_2(x) = \sin(wx).$$

Example 4 (*An Almost Periodic Problem in* [20]).

$$\begin{aligned} y_1'' &= -y_1 + \epsilon \cos(\Psi x), y_1(0) = 1, y_1'(0) = 0, \\ y_2'' &= -y_2 + \epsilon \sin(\Psi x), y_2(0) = 0, y_2'(0) = 1,\ x \in [0, 10]. \end{aligned}$$

The exact solution is

$$y_1(x) = \tfrac{(1-\epsilon-\Psi^2)}{(1-\Psi^2)} \cos(x) + \tfrac{\epsilon}{(1-\Psi^2)} \cos(\Psi x),$$

$$y_2(x) = \tfrac{(1-\epsilon\Psi-\Psi^2)}{(1-\Psi^2)} \sin(x) + \tfrac{\epsilon}{(1-\Psi^2)} \sin(\Psi x),$$

where $\epsilon = 0.001$ *and* $\Psi = 0.1$.

Example 5 (*The Two-Body Gravitational Problem in* [21]).

$$\begin{aligned} y_1'' &= -\frac{y_1}{({y_1}^2 + {y_2}^2)^{\frac{3}{2}}},\ y_1(0) = 1,\ y_1'(0) = 0, \\ y_2'' &= -\frac{y_2}{({y_1}^2 + {y_2}^2)^{\frac{3}{2}}},\ y_2(0) = 0,\ y_2'(0) = 1,\ x \in [0, 10]. \end{aligned}$$

The exact solution is

$$y_1(x) = \cos(x),$$

$$y_2(x) = \sin(x).$$

Example 6 (*The Linear Strehmel-Weiner Problem in* [22]).

$$\begin{aligned} y_1'' &= -20.2y_1 - 9.6y_3 + 150\cos(10x),\ y_1(0) = 1,\ y_1'(0) = 0, \\ y_2'' &= 7989.6y_1 - 10000y_2 - 6004.2y_3 + 75\cos(10x),\ y_2(0) = 2,\ y_2'(0) = 0, \\ y_3'' &= -9.6y_1 - 5.8y_3 + 75\cos(10x),\ y_3(0) = -2,\ y_3'(0) = 0,\ x \in [0, 10]. \end{aligned}$$

The exact solution is

$$\begin{aligned} y_1(x) &= \cos(x) + 2\cos(5x) - 2\cos(10x), \\ y_2(x) &= 2\cos(x) + \cos(5x) - \cos(10x), \\ y_3(x) &= -2\cos(x) + \cos(5x) - \cos(10x). \end{aligned}$$

After solving the above problems, the obtained data were collected in Tables 2–7, where we have considered different tolerances, Tol. The tables present the usual values as

- NFE: number of function evaluations;
- NSTEP: number of steps;
- FSTEP: number of failed steps;

- MAXER: maximum absolute errors;
- CPU: computational time in seconds.

We can see that the proposed method presents very good results concerning the errors, number of steps, and computation time.

Table 2. Data for Example 1.

TOL	METHOD	NSTEP	NFE	FSTEP	MAXER	CPU(s)
10^{-2}	DIRKN5(4)4D	62	775	17	1.166687(−3)	0.114
	DIRKN4(3)R	88	1062	20	1.300211(−1)	0.136
	DIRKN4(3)I	106	1235	77	1.098554(−1)	0.129
	DIRKN4(3)N	51	611	11	8.478071(−3)	0.109
10^{-4}	DIRKN5(4)4D	150	1700	22	2.221516(−5)	0.130
	DIRKN4(3)R	271	2874	18	1.473537(−3)	0.279
	DIRKN4(3)I	209	2446	156	5.054511(−3)	0.188
	DIRKN4(3)N	159	1781	21	2.422741(−4)	0.172
10^{-6}	DIRKN5(4)4D	369	3881	21	3.512952(−7)	0.172
	DIRKN4(3)R	847	8697	25	1.717557(−5)	0.194
	DIRKN4(3)I	430	3750	62	3.540320(−3)	0.177
	DIRKN4(3)N	492	5084	18	4.551367(−6)	0.187
10^{-8}	DIRKN5(4)4D	919	9399	23	4.796842(−9)	0.167
	DIRKN4(3)R	2670	26945	27	2.051531(−7)	0.515
	DIRKN4(3)I	1344	10837	17	1.169709(−3)	0.193
	DIRKN4(3)N	1547	15490	2	5.675619(−8)	0.263

Table 3. Data for Example 2.

TOL	METHOD	NSTEP	NFE	FSTEP	MAXER	CPU(s)
10^{-6}	DIRKN5(4)4D	82	822	0	1.410894(−8)	0.117
	DIRKN4(3)R	129	1292	0	8.387015(−6)	0.136
	DIRKN4(3)I	89	714	0	6.452908(−4)	0.107
	DIRKN4(3)N	75	752	0	4.823713(−7)	0.112
10^{-8}	DIRKN5(4)4D	203	2032	0	1.429289(−10)	0.149
	DIRKN4(3)R	405	4052	0	8.329823(−8)	0.390
	DIRKN4(3)I	305	2442	0	1.916590(−4)	0.352
	DIRKN4(3)N	234	2342	0	4.929034(−9)	0.189
10^{-10}	DIRKN5(4)4D	510	5102	0	1.434075(−12)	0.211
	DIRKN4(3)R	1281	12812	0	2.091332(−9)	0.466
	DIRKN4(3)I	1301	10410	0	4.497424(−5)	0.621
	DIRKN4(3)N	738	7382	0	4.832812(−11)	0.232
10^{-12}	DIRKN5(4)4D	1280	12811	1	2.153833(−14)	0.599
	DIRKN4(3)R	4143	41441	1	5.985942(−10)	1.562
	DIRKN4(3)I	6001	48024	2	9.749332(−6)	1.693
	DIRKN4(3)N	2332	23322	0	2.124634(−12)	0.860

Table 4. Data for Example 3.

TOL	METHOD	NSTEP	NFE	FSTEP	MAXER	CPU(s)
10^{-4}	DIRKN5(4)4D	65	661	1	3.504117(−6)	0.114
	DIRKN4(3)R	97	981	1	8.702183(−4)	0.174
	DIRKN4(3)I	74	739	29	3.866006(−3)	0.125
	DIRKN4(3)N	56	571	1	8.689475(−5)	0.109

Table 4. *Cont.*

TOL	METHOD	NSTEP	NFE	FSTEP	MAXER	CPU(s)
10^{-6}	DIRKN5(4)4D	162	1631	1	3.524892(−8)	0.125
	DIRKN4(3)R	306	3071	2	7.979708(−6)	0.266
	DIRKN4(3)I	175	1402	0	1.463717(−3)	0.141
	DIRKN4(3)N	177	1772	0	1.305833(−6)	0.145
10^{-8}	DIRKN5(4)4D	406	4071	1	3.508535(−10)	0.141
	DIRKN4(3)R	969	9701	1	7.855478(−8)	0.432
	DIRKN4(3)I	609	4881	1	4.335748(−4)	0.156
	DIRKN4(3)N	560	5611	1	1.444078(−8)	0.266
10^{-10}	DIRKN5(4)4D	1019	10201	1	3.492207(−12)	0.328
	DIRKN4(3)R	3072	30740	2	3.577114(−9)	1.491
	DIRKN4(3)I	2599	20808	2	1.015274(−4)	0.656
	DIRKN4(3)N	1770	17711	1	1.460893(−10)	0.574

Table 5. Data for Example 4.

TOL	METHOD	NSTEP	NFE	FSTEP	MAXER	CPU(s)
10^{-4}	DIRKN5(4)4D	33	332	0	1.349489(−6)	0.102
	DIRKN4(3)R	41	412	0	8.438180(−4)	0.139
	DIRKN4(3)I	29	234	0	3.503317(−3)	0.095
	DIRKN4(3)N	25	252	0	3.999204(−5)	0.094
10^{-6}	DIRKN5(4)4D	82	822	0	1.408053(−8)	0.135
	DIRKN4(3)R	129	1292	0	8.388568(−6)	0.188
	DIRKN4(3)I	89	714	0	6.451886(−4)	0.125
	DIRKN4(3)N	75	752	0	4.826930(−7)	0.124
10^{-8}	DIRKN5(4)4D	203	2032	0	1.426580(−10)	0.141
	DIRKN4(3)R	405	4052	0	8.331265(−8)	0.503
	DIRKN4(3)I	305	2442	0	1.916412(−4)	0.453
	DIRKN4(3)N	234	2342	0	4.933000(−9)	0.407
10^{-10}	DIRKN5(4)4D	510	5102	0	1.429967(−12)	0.203
	DIRKN4(3)R	1281	12812	0	2.091295(−9)	0.528
	DIRKN4(3)I	1302	10418	0	4.497475(−5)	0.318
	DIRKN4(3)N	738	7382	0	4.836931(−11)	0.219

Table 6. Data for Example 5.

TOL	METHOD	NSTEP	NFE	FSTEP	MAXER	CPU(s)
10^{-6}	DIRKN5(4)4D	82	822	0	3.175219(−7)	0.121
	DIRKN4(3)R	129	1292	0	1.101892(−4)	0.184
	DIRKN4(3)I	89	714	0	1.735505(−2)	0.154
	DIRKN4(3)N	75	752	0	2.976045(−5)	0.119
10^{-8}	DIRKN5(4)4D	204	2042	0	3.324550(−9)	0.183
	DIRKN4(3)R	405	4052	0	8.676796(−7)	0.337
	DIRKN4(3)I	306	2450	0	5.063575(−3)	0.233
	DIRKN4(3)N	234	2342	0	3.248236(−7)	0.227
10^{-10}	DIRKN5(4)4D	510	5102	0	3.387382(−11)	0.191
	DIRKN4(3)R	1281	12812	0	4.142077(−8)	0.598
	DIRKN4(3)I	1303	10426	0	1.196753(−3)	0.533
	DIRKN4(3)N	739	7392	0	3.163731(−9)	0.266
10^{-12}	DIRKN5(4)4D	1280	12811	1	3.440165(−13)	0.624
	DIRKN4(3)R	4144	41451	1	1.565965(−8)	2.143
	DIRKN4(3)I	6004	48048	2	2.598544(−4)	5.137
	DIRKN4(3)N	2333	23332	0	2.316779(−11)	1.571

Table 7. Data for Example 6.

TOL	METHOD	NSTEP	NFE	FSTEP	MAXER	CPU(s)
10^{-4}	DIRKN5(4)4D	332	3659	36	1.929085(−6)	0.259
	DIRKN4(3)R	1437	17207	315	5.815997(−4)	1.440
	DIRKN4(3)I	878	8782	278	3.948050(−3)	1.112
	DIRKN4(3)N	368	3773	9	3.073405(−5)	1.396
10^{-6}	DIRKN5(4)4D	819	8552	40	1.951671(−8)	0.592
	DIRKN4(3)R	1765	18062	46	1.677046(−6)	1.647
	DIRKN4(3)I	1323	13478	578	2.769727(−3)	1.212
	DIRKN4(3)N	988	9892	0	1.088239(−6)	1.186
10^{-8}	DIRKN5(4)4D	2041	20772	40	1.912657(−10)	1.421
	DIRKN4(3)R	5569	56160	52	2.077480(−8)	3.575
	DIRKN4(3)I	3196	25976	66	9.770284(−4)	3.171
	DIRKN4(3)N	3127	31272	0	1.078572(−8)	2.772
10^{-10}	DIRKN5(4)4D	5112	51573	51	3.427481(−12)	4.024
	DIRKN4(3)R	17870	179125	47	6.237488(−9)	15.023
	DIRKN4(3)I	13589	109504	112	1.844331(−4)	7.265
	DIRKN4(3)N	9888	98898	2	1.099867(−10)	5.355

To further show the robustness and performance of the proposed method, we present the efficiency curves of DIRKN5(4)4D compared to other existing DIRKN methods. Figures 1–6 show the efficiency curves for the examples considered, where one can observe the best performance of the proposed method. We utilized the following tolerances: Tol $= 10^{-2k}$, $k = 1,2,3,4$ for problem 1, and $k = 3,4,5,6$ for problems 2 and 5, and $k = 2,3,4,5$ for problems 3, 4, and 6.

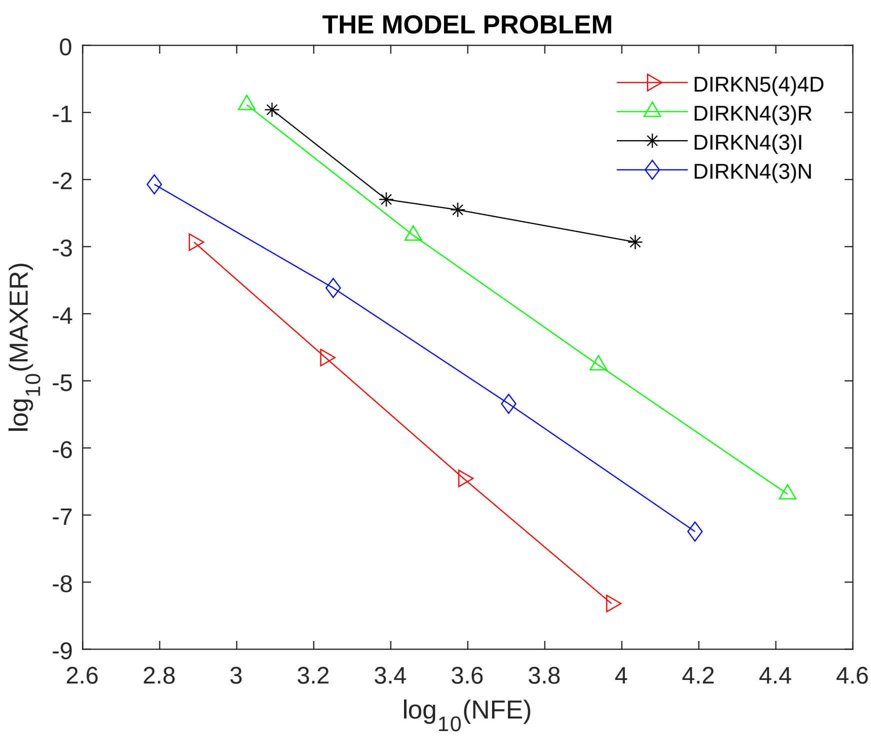

Figure 1. Efficiency curves for Example 1.

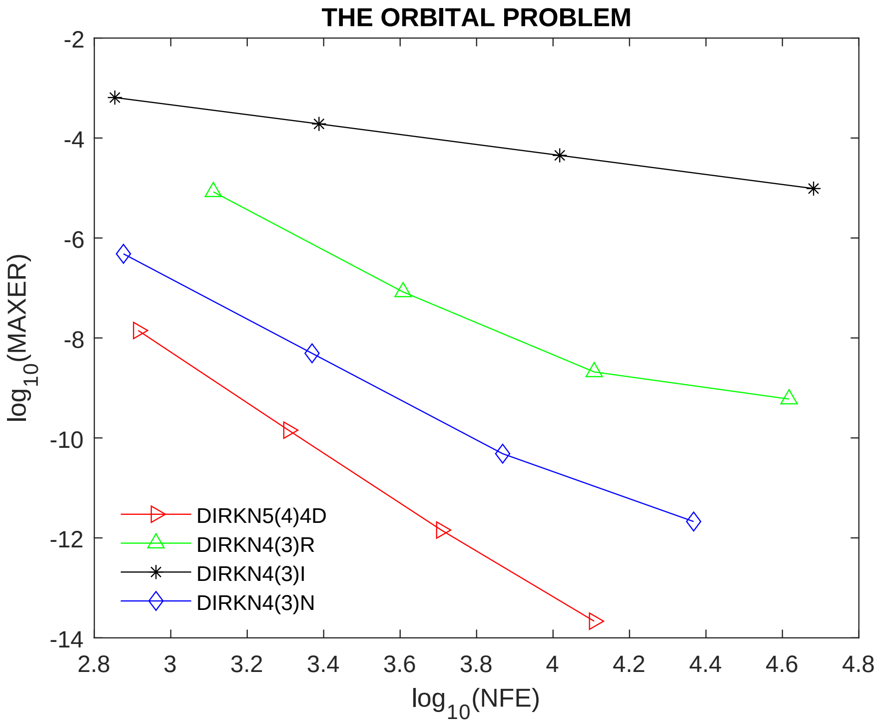

Figure 2. Efficiency curves for Example 2.

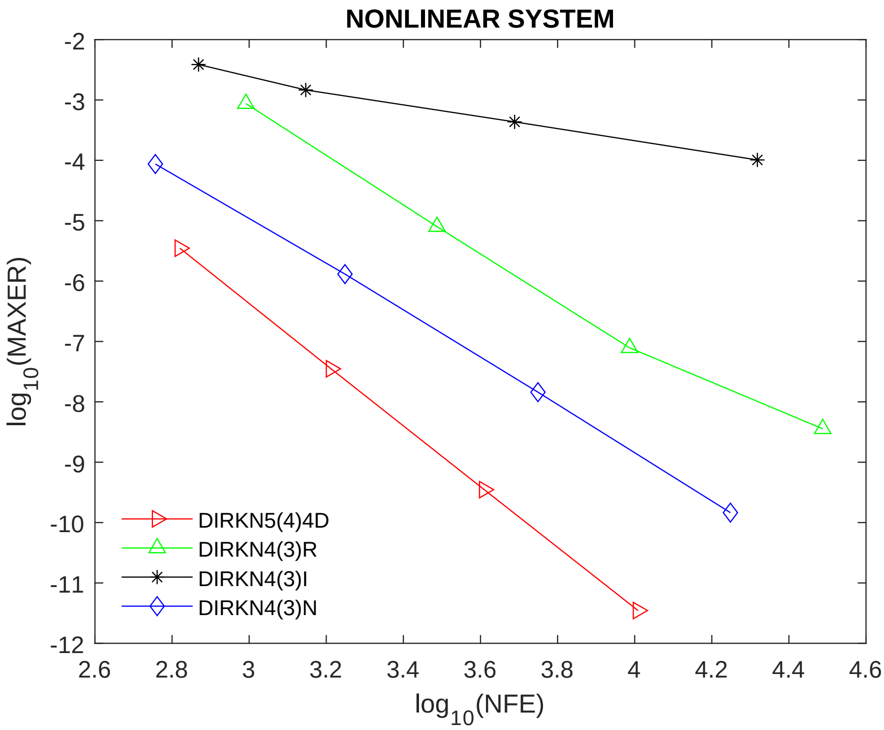

Figure 3. Efficiency curves for Example 3.

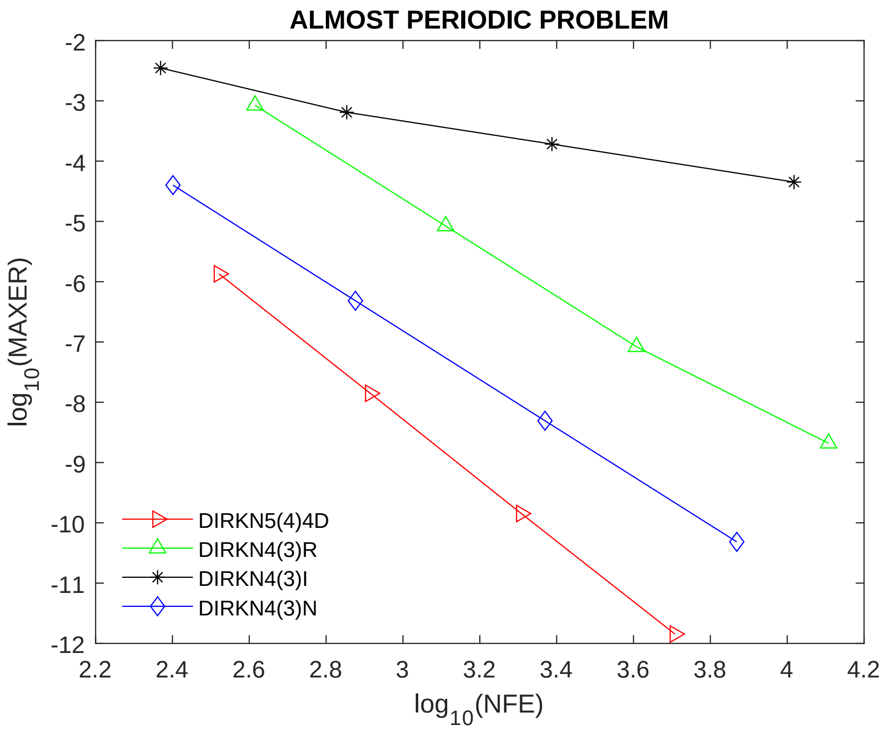

Figure 4. Efficiency curves for Example 4.

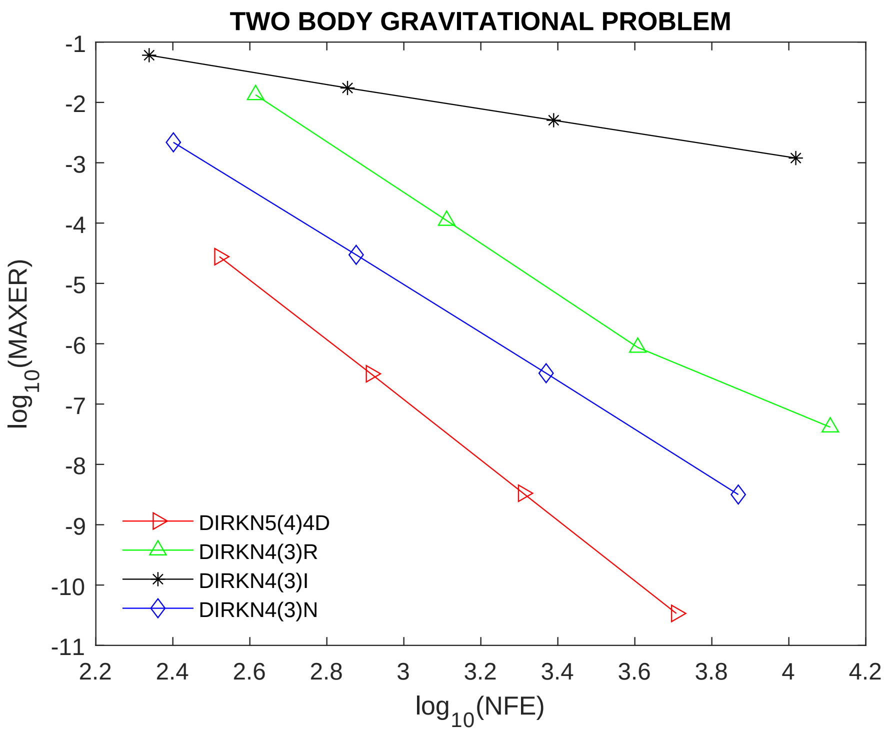

Figure 5. Efficiency curves for Example 5.

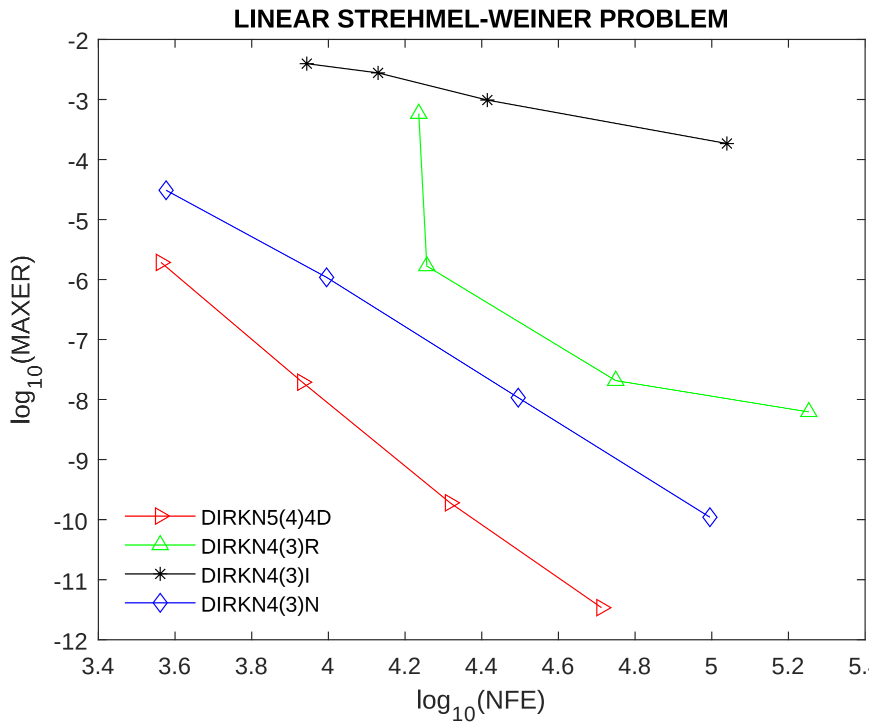

Figure 6. Efficiency curves for Example 6.

5. Discussion of Results

The newly developed method DIRKN5(4)4D has the lowest error norm, the lowest number of function evaluations per step, and the lowest CPU time, meaning that it has high efficiency and accuracy when solving all the given modeled problems as shown in Tables 2–7 and in Figures 1–6. Therefore, the DIRKN5(4)4D is suitable for the numerical solution of the problem in (1), showing a better performance than other embedded DIRKN methods in the literature.

6. Conclusions

In this paper, we have obtained an efficient diagonally implicit 5(4) embedded RKN pair. The developed method is of constant coefficients. In addition, we computed the principal local truncation errors for the higher and lower order methods in the new DIRKN5(4)4D pair. Furthermore, the stability intervals have been obtained. The numerical experiments show clearly that DIRKN5(4)4D is more efficient than other DIRKN methods used for comparisons.

Author Contributions: Conceptualization, M.A.D., N.S. and W.W.; Data curation, H.R.; Formal analysis, M.A.D., N.S. and H.R.; Investigation, M.A.D., N.S., H.R. and W.W.; Methodology, N.S., H.R. and W.W.; Supervision, N.S., H.R. and W.W.; Validation, W.W.; Visualization, M.A.D.; Writing—original draft, M.A.D.; Writing—review and editing, H.R. All authors have read and agreed to the published version of the manuscript.

Funding: The Center of Excellence in Theoretical and Computational Science (TaCS-CoE), KMUTT, the Thailand Science Research and Innovation (TSRI) Basic Research Fund, for the fiscal year 2022 with project No. FRB650048/0164. The first author appreciates the support of the Petchra Pra Jom Klao PhD Research Scholarship from KMUTT with Grant No. 15/2562.

Data Availability Statement: Not applicable.

Acknowledgments: The authors appreciate the support rendered by Poom Kumam, through the Center of Excellence in Theoretical and Computational Science (TaCS-CoE), for providing a conducive environment to conduct this research, and the King Mongkut's University of Technology, Thonburi (KMUTT), for the financial support.

Conflicts of Interest: The authors have no conflict of interest to declare.

References

1. Sommeijer, B.P. A note on a diagonally implicit Runge–Kutta–Nyström method. *J. Comput. Appl. Math.* **1987**, *19*, 395–399. [CrossRef]
2. der Houwen, P.J.V.; Sommeijer, B.P. Diagonally implicit Runge–Kutta–Nyström methods for oscillatory problems. *SIAM J. Numer. Anal.* **1989**, *26*, 414–429. [CrossRef]
3. Imoni, S.; Otunta, F.; Ramamohan, T. Embedded implicit Runge–Kutta–Nyström method for solving second-order differential equations. *Int. J. Comput. Math.* **2006**, *83*, 777–784. [CrossRef]
4. Sharp, P.; Fine, J.; Burrage, K. Two-stage and three-stage diagonally implicit Runge–Kutta–Nyström methods methods of orders three and four. *IMA J. Numer. Anal.* **1990**, *10*, 489–504. [CrossRef]
5. Senu, N.; Suleiman, M.; Ismail, F.; Othman, M. A singly diagonally implicit Runge–Kutta–Nyström method for solving oscillatory problems. *IAENG Int. J. Appl. Math.* **2011**, *41*, 155–161.
6. Senu, N.; Suleiman, M.; Ismail, F.; Othman, M. A new diagonally implicit Runge–Kutta–Nyström method for periodic ivps. *WSEAS Trans. Math.* **2010**, *9*, 679–688. [CrossRef]
7. Senu, N.; Suleiman, M.; Ismail, M.; Othman, M. A fourth-order diagonally implicit Runge–Kutta–Nyström method with dispersion of high order. In Proceedings of the 4th International Conference on Applied Mathematics Simulation, Modelling (ASM'10), Corfu Island, Greece, 22–25 July 2010; pp. 78–82.
8. Senu, N.; Suleiman, M.; Ismail, F.; Arifin, N.M. New 4 (3) pairs diagonally implicit Runge–Kutta–Nyström method for periodic ivps. *Dyn. Nat. Soc.* **2012**, *2012*, 324989. [CrossRef]
9. Papageorgiou, G.; Famelis, I.T.; Tsitouras, C. A p-stable singly diagonally implicit Runge–Kutta–Nyström method. *Numer. Algorithms* **1998**, *17*, 345–353. [CrossRef]
10. Al-Khasawneh, R.A.; Ismail, F.; Suleiman, M. Embedded diagonally implicit Runge–Kutta–Nyström 4(3) pair for solving special second-order ivps. *Appl. Math. Comput.* **2007**, *190*, 1803–1814. [CrossRef]
11. Ismail, F.; Al-Khasawneh, R.A.; Suleiman, M. Embedded singly diagonally implicit Runge–Kutta–Nyström general method (3, 4) in (4, 5) for solving second order ivps. *Int. J. Appl. Math.* **2007**, *37*, 2.
12. Imoni, S.; Ikhile, M. Zero dissipative DIRKN pairs of order 5(4) for solving special second order ivps. *Acta Univ. Palacki. Olomuc. Fac. Rerum Nat. Math.* **2014**, *52*, 53–69.
13. Simos, T.E. Embedded Runge–Kutta methods for periodic initial-value problems. *Math. Comput. Simul.* **1993**, *35*, 387–395. [CrossRef]
14. Senu, N. Runge–Kutta–Nyström Methods for Solving Oscillatory Problems. Ph.D. Thesis, Universiti Putra Malaysia, Serdang, Malaysia, 2010.
15. Hairer, E.; Nørsett, S.P.; Wanner, G. *Solving Ordinary Differential Equations I*; Springer: New York, NY, USA, 1993.
16. Hairer, E.; Wanner, G. A theory for Nyström methods. *Numer. Math.* **1976**, *25*, 383–400. [CrossRef]
17. Butcher, J.C. *Numerical Methods for Ordinary Differential Equations*; John Wiley and Sons, Ltd.: Hoboken, NJ, USA, 2008; Volume 2.
18. Medvedev, M.; Simos, T.E.; Tsitouras, C. Explicit, two-stage, sixth-order, hybrid four-step methods for solving $y'' = f(x, y)$. *Math. Methods Appl. Sci.* **2018**, *41*, 6997–7006. [CrossRef]
19. Kalogiratou, Z.; Monovasilis, T.; Simos, T.E. Two-derivative Runge–Kutta methods with optimal phase properties. *Math. Meth. Appl. Sci.* **2020**, *43*, 1267–1277. [CrossRef]
20. de Vyver, H.V. A Runge–Kutta–Nyström pair for the numerical integration of perturbed oscillators. *Comput. Phys. Commun.* **2005**, *167*, 129–142. [CrossRef]
21. Moo, K.; Senu, N.; Ismail, F.; Suleiman, M. A zero-dissipative phase-fitted fourth order diagonally implicit Runge–Kutta–Nyström method for solving oscillatory problems. *Math. Probl. Eng.* **2014**, *2014*, 985120. [CrossRef]
22. Cong, N.H. A-stable diagonally implicit Runge–Kutta–Nyström methods for parallel computers. *Numer. Algorithms* **1993**, *4*, 263–281. [CrossRef]