mathematics

MDPI

Article

Nine-Stage Runge–Kutta–Nyström Pairs Sharing Orders Eight and Six

Hadeel Alharbi [1], Kusum Yadav [1], Rabie A. Ramadan [1,2,3], Houssem Jerbi [4], Theodore E. Simos [5,6,7,8,9,*] and Charalampos Tsitouras [10]

1 College of Computer Science and Engineering, University of Hail, Ha'il 81481, Saudi Arabia; h.alharbe@uoh.edu.sa (H.A.); y.kusum@uoh.edu.sa (K.Y.); rabie@rabieramadan.org (R.A.R.)
2 Information Systems Department, College of Economics, Management & Information Systems, Nizwa University, Nizwa 616, Sultanate of Oman
3 Computer Engineering Department, Faculty of Engineering, Cairo University, Giza 12613, Egypt
4 Department of Industrial Engineering, College of Engineering, University of Hail, Ha'il 81481, Saudi Arabia; h.jerbi@uoh.edu.sa
5 Center for Applied Mathematics and Bioinformatics, Gulf University for Science and Technology, Mubarak Al-Abdullah 32093, Kuwait
6 Laboratory of Inter-Disciplinary Problems of Energy Production, Ulyanovsk State Technical University, 32 Severny Venetz Street, 432027 Ulyanovsk, Russia
7 Department of Medical Research, China Medical University Hospital, China Medical University, Taichung City 40402, Taiwan
8 Data Recovery Key Laboratory of Sichun Province, Neijing Normal University, Neijiang 641100, China
9 Section of Mathematics, Department of Civil Engineering, Democritus University of Thrace, 67100 Xanthi, Greece
10 General Department, National & Kapodistrian University of Athens, Euripus Campus, 34400 Psachna, Greece; tsitourasc@uoa.gr
* Correspondence: simos@ulstu.ru

Abstract: We explore second-order systems of non-stiff initial-value problems (IVPs), particularly those cases where the first derivatives are absent. These types of problems are of significant interest and have applications in various domains, such as astronomy and physics. Runge–Kutta–Nyström (RKN) pairs stand out as highly effective methods of addressing these IVPs. In order to create a pair with eighth and sixth orders, we need to address a certain known set of equations concerning the coefficients. When constructing such pairs for use in double-precision arithmetic, we often need to meet various conditions. Primarily, we aim to maintain small coefficient magnitudes to prevent a loss of accuracy. Nevertheless, in the context of quadruple precision, we can tolerate larger coefficients. This flexibility enables us to establish pairs with eighth and sixth orders that exhibit significantly reduced truncation errors. Traditionally, these pairs are constructed to go through eight stages per step. Here, we propose using nine stages per step. Then we have available more coefficients in order to further reduce truncation errors. As a result, we construct a novel pair that, as anticipated, achieves superior performance compared to equivalent-order pairs in various significant problem scenarios.

Keywords: initial value problem; Runge–Kutta–Nyström; quadruple precision

MSC: 65L05; 65L06

Citation: Alharbi, H.; Yadav, K.; Ramadan, R.A.; Jerbi, H.; Simos, T.E.; Tsitouras, C. Nine-Stage Runge–Kutta–Nyström Pairs Sharing Orders Eight and Six. *Mathematics* **2024**, *12*, 316. https://doi.org/10.3390/math12020316

Academic Editors: Valery Karachik and Juan Ramón Torregrosa Sánchez

Received: 27 November 2023
Revised: 4 January 2024
Accepted: 12 January 2024
Published: 18 January 2024

1. Introduction

We center our attention on a particular class of second-order initial-value problems (IVPs), which are defined as shown below:

$$\xi'' = \phi(x, \xi), \xi(x_0) = \xi_0, \ \xi'(x_0) = \xi_0'. \tag{1}$$

Here, $\phi : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}^m$ is assumed to be sufficiently continuously differentiable, and the initial conditions are given by $(\xi_0, \xi_0') \in \mathbb{R}^{2m}$.

We compute some approximation of the solution to Equation (1) in a number of points that are separate from one other (x_n, ξ_n, ξ'_n), applying an explicit Runge–Kutta–Nyström method that is of algebraic order p. The structure of the method is as follows (for further details on these methods, refer to [1] and ([2] p. 283)):

$$\phi_i = \phi(x_n + c_i\tau_n, \xi_n + c_i\tau_n\xi'_n + \tau_n^2 \sum_{j=1}^{i-1} d_{ij}\phi_j), \quad i = 1, 2, 3, \cdots, s-1, s$$

$$\xi_{n+1} = \xi_n + \tau_n\xi'_n + \tau_n^2 \textstyle\sum_{i=1}^{s} w_i\phi_i,$$
$$\xi'_{n+1} = \xi'_n + \tau_n \textstyle\sum_{i=1}^{s} w'_i\phi_i.$$

Here, $\tau_n = x_{n+1} - x_n$, representing the step size. Over the last five decades, there has been a consistent interest in these methods. Noteworthy works have been made by various researchers. E. Fehlberg [3] presented a detailed list of order conditions up to the eighth order. Dormand and colleagues [4,5] gave a series of State-of-the-Art pairs of orders, 4(3), 6(4), and 8(6). El-Mikkawy et al. [6] introduced the idea of adding a stage more than the minimal requirement, in order to increase the efficiency. This technique inspired our approach here. Moreover, novel RKN approaches with distinct features have been introduced. For example, RKN methods were also explored by Houven et al., aiming to minimize errors in phase, while Calvo et al. [7] and Yoshida [8] devised RKN algorithms that embed symplecticity.

In what follows, we choose $p = 8$ and merge the previously mentioned method with a companion sixth-order formula. Consequently, we also compute a sixth-order estimate, using the same ϕ_i values:

$$\hat{\xi}_{n+1} = \xi_n + \tau_n\xi'_n + \tau_n^2 \textstyle\sum_{i=1}^{s} \hat{w}_i\phi_i,$$
$$\hat{\xi}'_{n+1} = \xi'_n + \tau_n \textstyle\sum_{i=1}^{s} \hat{w}'_i\phi_i.$$

The approximations of ξ_n and ξ'_n in higher orders are utilized in all situations to advance the solutions over time.

Consequently, we possess an error estimation:

$$\epsilon = \max\left(\|\xi_{n+1} - \hat{\xi}_{n+1}\|, \|\xi'_{n+1} - \hat{\xi}'_{n+1}\|\right) = O\left(\tau^7\right).$$

Next, we contrast ϵ with TOL, a user-defined small positive value referred to as the tolerance, to gauge the size of the subsequent step as

$$\tau_{n+1} = 0.9\tau_n \left(\frac{TOL}{\epsilon}\right)^{1/7}, \tag{2}$$

commonly employed with RKN8(6) pairs [4,9]; if TOL is less than ϵ, we halt the advancement of the solution. In this scenario, we reiterate the current step, but now we employ τ_{n+1} as the updated, shorter version, instead of τ_n.

The coefficients are often represented using the Butcher tableau, as described in [10]. Consequently, the method can be expressed using the following form:

c	D
	$w, \hat{w}$
	$w', \hat{w}'$

where $D \in \mathbb{R}^{s\times s}$, and $c, \hat{w}^{\mathrm{T}}, w^{\mathrm{T}}, \hat{w}^{\mathrm{T}}, w'^{\mathrm{T}} \in \mathbb{R}^s$, meaning the weights are expressed as row vectors.

In this context, we explore a nine-stage pair ($s = 9$). The coefficients for this pair are provided in the tableau displayed in Table 1, which is commonly named the "Butcher tableau". This type of tableau is in common use when tabulating the coefficients of the type of methods we propose here ([2], p. 134).

Table 1. The Butcher tableau for RKN pairs of orders 8(6).

0									
c_2	d_{21}								
c_3	d_{31}	d_{32}							
c_4	d_{41}	d_{42}	d_{43}						
c_5	d_{51}	d_{52}	d_{53}	d_{54}					
c_6	d_{61}	d_{62}	d_{63}	d_{64}	d_{65}				
c_7	d_{71}	d_{72}	d_{73}	d_{74}	d_{75}	d_{76}			
1	d_{81}	d_{82}	d_{83}	d_{84}	d_{85}	d_{86}	d_{87}		
1	w_1	0	w_3	w_4	w_5	w_6	w_7	0	
8th-order w	w_1	0	w_3	w_4	w_5	w_6	w_7	0	0
6th-order $\hat{w}$	$\hat{w}_1$	0	$\hat{w}_3$	$\hat{w}_4$	$\hat{w}_5$	$\hat{w}_6$	$\hat{w}_7$	0	0
8th-order w'	w'_1	0	w'_3	w'_4	w'_5	w'_6	w'_7	w'_8	w'_9
6th-order $\hat{w}'$	$\hat{w}'_1$	0	$\hat{w}'_3$	$\hat{w}'_4$	$\hat{w}'_5$	$\hat{w}'_6$	$\hat{w}'_7$	$\hat{w}'_8$	$\hat{w}'_9$

Usually, these pairs make use of merely eight stages per iteration, as the ultimate stage is re-used as the initial one of the subsequent step. This leads to the values in the ninth stage matching to the vector w. In simpler terms, $d_{9j} = w_j$ when $j = 1, 2, 3, \cdots, 7, 8$. This device is frequently denoted as FSAL, which stands for first stage as last.

Eighth-order RKN pairs, which efficiently employ eight stages per step, were analyzed in previous works [5,11]. It is worth noting that eighth-order RKN techniques, utilizing seven stages per step, have been developed specifically for linear inhomogeneous problems.

Thus, to form a pair with eighth and sixth orders, it becomes necessary to tackle a well-known set of equations related to the coefficients. When creating such pairs for implementation in double-precision arithmetic, we encounter specific conditions that must be satisfied. Our primary objective is to keep the coefficient magnitudes small, to prevent any loss of accuracy. However, when dealing with quadruple precision, we can accept larger coefficients. This increased flexibility allows us to design pairs with eighth and sixth orders, resulting in significantly reduced truncation errors. Conventionally, these pairs are designed to undergo eight stages per step. In this context, we suggest employing nine stages per step, providing us with additional coefficients to further minimize truncation errors. Consequently, we have devised an innovative pair that, as expected, demonstrates superior performance compared to pairs of equivalent orders across various significant problem scenarios.

2. Eighth-Order Runge–Kutta–Nyström Methods

We utilize an RKN method for (1) and employ Taylor-series expansions for $\xi(x_n + \tau) - \xi_{n+1}$ and $\xi'(x_n + \tau) - \xi'_{n+1}$. By equating the expressions up to τ^8 for an eighth-order method, the outcomes found below are achieved:

$$\xi(x_n + \tau) - \xi_{n+1} = \tau^2 e_{2,1} Q_{2,1} + \tau^3 e_{3,1} Q_{3,1} + \cdots + \tau^8 (e_{8,1} Q_{8,1} + ... + e_{8,20} Q_{8,20}) + O\left(\tau^9\right) \tag{3}$$

$$\xi'(x_n + \tau) - \xi'_{n+1} = \tau \tilde{e}_{1,1} Q_{1,1} + \tau^2 \tilde{e}_{2,1} Q_{2,1} + \cdots + \tau^8 (\tilde{e}_{8,1} Q_{8,1} + ... + \tilde{e}_{8,36} Q_{8,36}) + O\left(\tau^9\right). \tag{4}$$

Expression e_{ij} depends on the variables w, D, and c, while $\tilde{e}_{ij}$ depends on w', D, and c. An algorithm deriving them symbolically can be found in [12]. The expression Q_{ij} involves elementary differentials related to ξ', ϕ, and their partial derivatives, which are inherent to the problem and beyond the method's control. However, in the case of an eighth-order RKN method, it is imperative to eliminate the coefficients e_{ij} and $\tilde{e_{ij}}$ in expressions (3) and (4) up

to the specified order. Table 2 presents the count of the order requirements (i.e., e_{ij} and $\bar{e}_{ij}$ for each order. For instance, a third algebraic order method entails satisfying two equations for ξ and addressing four order conditions for ξ'.

Table 2. Number of order conditions of an RKN method.

		Order									
method	number of - order→	1	2	3	4	5	6	7	8	9	10
RKN	order conditions for ξ	0	1	1	2	3	6	10	20	36	72
	order conditions for ξ'	1	1	2	3	6	10	20	36	72	137

Upon scrutiny of the Butcher tableau above [10] (specifically, Table 1) and aligning the count of coefficients at hand for a nine-stage procedure with the order prerequisites documented up to the eighth order in Table 2, a conspicuous incongruity emerges. To tackle this issue, we adopt various simplification hypotheses that notably curtail the quantity of order stipulations.

First, we assume

$$w = w' \cdot (I_s - C). \tag{5}$$

Here, $I_s \in \mathbb{R}^{s \times s}$ denotes the identity matrix, and $C = \mathrm{diag}(c)$. With this assumption, we naturally satisfy the order prerequisites for ξ after eliminating equations of equivalent rank for ξ'. Our primary goal is to remove exclusively $\bar{e}_{ij}$ concerning w', D, c.

Once again, upon aggregating the figures in the ultimate row of Table 2, it becomes evident that there are still an excessive number of conditions for the existing coefficients. To tackle this issue, we proceed by introducing the following assumptions:

$$D \cdot \mathbb{I} = \frac{1}{2}c^2,\ D \cdot c = \frac{1}{6}c^3,\ D \cdot c^2 = \frac{1}{12}c^4. \tag{6}$$

Here, we define c^i as an element-wise product of matrices, denoted as $\circ$ (i.e., Hadamard multiplication):

$$c^i = \underbrace{c \circ \cdots \circ c \circ c}_{i-\text{ times.}},$$

It is important to highlight that this multiplication operation is of lesser precedence than the dot product.

We additionally consider the row simplification requirement for RKN methods, expressed as

$$w' \cdot (D + C - \frac{1}{2}(C \circ C) - \frac{1}{2}I_s).$$

Additionally, we consider the following subsidiary simplifying assumptions:

$$(w \cdot D)_2 = 0,\ (w' \cdot D)_2 = 0,\ (w' \cdot (C \circ C) \cdot D)_2 = 0,\ (\hat{w} \cdot D)_2 = 0.$$

This substantial decrease in the count of the order prerequisites enables us to advance in the computation of the coefficients for an eighth-order method (w', w, D, and c), utilizing the subsequent procedure. In the following, we give the Mathematica [13] version of the algorithm, for the interested reader:

```
RKNT86[c4_, c5_, c6_, c7_, d85_, d86_, d87_, d92_, dw9_] :=
 Module[{e, dw, dw1, dw3, dw4, dw5, dw6, dw7, dw8, dww1, dww3, dww4,
   dww5, dww6, dww7, dww8, dww, c, c2, c3, cc, ii, w, ww, d, d21, d31,
    d32, d41, d42, d43, , d51, d52, d53, d54, d61, d62, d63, d64, d65,
    d71, d72, d73, d74, d75, d76, d81, d82, d83, d84, d91, d93, d94,
   d95, d96, d97, mond, vanderl, simp1, equs, equ11, so, de, dc, dc2,
   wdc}, e = {1, 1, 1, 1, 1, 1, 1, 1, 1};
```

```
dw = {dw1, 0, dw3, dw4, dw5, dw6, dw7, dw8, dw9};
dww = {dww1, 0, dww3, dww4, dww5, dww6, dww7, dww8, dw9 - 3/20};
c = {0, c2, c3, c4, c5, c6, c7, 1, 1};
cc = DiagonalMatrix[c]; ii = IdentityMatrix[9];
w = dw.(ii - cc);
ww = dww.(ii - cc);
d = {{0, 0, 0, 0, 0, 0, 0, 0, 0}, {d21, 0, 0, 0, 0, 0, 0, 0,
   0}, {d31, d32, 0, 0, 0, 0, 0, 0, 0}, {d41, d42, d43, 0, 0, 0, 0,
   0, 0}, {d51, d52, d53, d54, 0, 0, 0, 0, 0}, {d61, d62, d63, d64,
   d65, 0, 0, 0, 0}, {d71, d72, d73, d74, d75, d76, 0, 0, 0}, {d81,
   d82, d83, d84, d85, d86, d87, 0, 0}, {d91, d92, d93, d94, d95,
   d96, d97, 0, 0}};
mond = {dw.e == 1, dw.c == 1/2, dw.c^2 == 1/3, dw.c^3 == 1/4,
  dw.c^4 == 1/5, dw.c^5 == 1/6, dw.c^6 == 1/7};
vanderl = {dww.e - 1, dww.c - 1/2, dww.(c c) - 1/3,
  dww.(c c c) - 1/4, dww.(c c c c) - 1/5, dww.(c c c c c) - 1/6};
simp1 = {(w.d)[[2]] == 0, (dw.d)[[2]] == 0, (dw.(c^2*d))[[2]] ==
   0, (dww.d)[[2]] == 0};
equs = {dw.(cc - ii).(cc - c7*ii).d.(cc - c3*ii).(cc - c4*ii).c ==
   Integrate[(x - 1)*(x - c7)*
     Integrate[
      Integrate[(x - c3)*(x - c4)*x, {x, 0, x}], {x, 0, x}], {x, 0,
     1}],
  dw.(cc - ii).d.(cc - c3*ii).(cc - c4*ii).(cc - c5*ii).c ==
   Integrate[(x - 1)*
     Integrate[
      Integrate[(x - c3)*(x - c4)*(x - c5)*x, {x, 0, x}], {x, 0,
       x}], {x, 0, 1}],
  dw.(cc - ii).d.(cc - c3*ii).(cc - c4*ii).(cc - c6*ii).c ==
   Integrate[(x - 1)*
     Integrate[
      Integrate[(x - c3)*(x - c4)*(x - c6)*x, {x, 0, x}], {x, 0,
       x}], {x, 0, 1}]};
equ11 =
 dww.d.(cc - c3 ii).(cc - c4 ii).c -
  c2*(c2 - c3)*(c2 - c4)*(dww.d)[[2]] -
  Integrate[
   Integrate[
    Integrate[(x - c3)*(x - c4)*x, {x, 0, x}], {x, 0, x}], {x, 0,
    1}];
c3 = (15 - 20*c4 - 20*c5 + 28*c4*c5 - 20*c6 + 28*c4*c6 + 28*c5*c6 -
    42*c4*c5*c6 - 20*c7 + 28*c4*c7 + 28*c5*c7 - 42*c4*c5*c7 +
    28*c6*c7 - 42*c4*c6*c7 - 42*c5*c6*c7 +
    70*c4*c5*c6*
     c7)/(2*(10 - 14*c4 - 14*c5 + 21*c4*c5 - 14*c6 + 21*c4*c6 +
      21*c5*c6 - 35*c4*c5*c6 - 14*c7 + 21*c4*c7 + 21*c5*c7 -
      35*c4*c5*c7 + 21*c6*c7 - 35*c4*c6*c7 - 35*c5*c6*c7 +
      70*c4*c5*c6*c7));
c2 = c3/2;
so = Solve[mond, {dw1, dw3, dw4, dw5, dw6, dw7, dw8}];
{dw1, dw3, dw4, dw5, dw6, dw7, dw8} = Simplify[so[[1, All, 2]]];
de = d.e - c^2/2; dc = d.c - c^3/6; dc2 = d.c^2 - c^4/12;
d32 = c3^3/6/c2;
so = Solve[{dc[[4]] == 0, dc2[[4]] == 0}, {d42, d43}]; {d42, d43} =
```

```
 Simplify[so[[1, All, 2]]];
so = Solve[simp1, {d82, d72, d62, d52}]; {d82, d72, d62, d52} =
 Simplify[so[[1, All, 2]]];
so = Solve[equs, {d65, d76, d75}]; {d65, d76, d75} =
 Simplify[so[[1, All, 2]]];
so = Solve[
  Join[dc[[5 ;; 8]], dc2[[5 ;; 8]]] == Array[0 &, 8], {d53, d54,
   d63, d64, d73, d74, d83, d84}];
{d53, d54, d63, d64, d73, d74, d83, d84} = Simplify[so[[1, All, 2]]];
wdc = dw.(d - cc^2/2 + cc - ii/2);
so = Solve[
  wdc[[3 ;; 7]] == Array[0 &, 5], {d93, d94, d95, d96, d97}];
{d93, d94, d95, d96, d97} = Simplify[so[[1, All, 2]]];
so = Solve[
  de[[2 ;; 9]] == Array[0 &, 8], {d21, d31, d41, d51, d61, d71, d81,
    d91}];
{d21, d31, d41, d51, d61, d71, d81, d91} = so[[1, All, 2]];
so = Solve[{Join[vander1, {equ11}] == {0, 0, 0, 0, 0, 0, 0}}, {dww1,
    dww3, dww4, dww5, dww6, dww7, dww8}];
{dww1, dww3, dww4, dww5, dww6, dww7, dww8} =
 Simplify[so[[1, All, 2]]]; Return[{d, c, w, ww, dw, dww}]]
```

It is worth noting that a simplified algorithm like this has never been seen before. It greatly assisted us in developing our pair.

3. Construction of RKN Pair-Sharing Orders Eight and Six

Applying the algorithm detailed in the preceding section, we have the ability to construct an eighth-order RKN technique while maintaining a nine-stage requirement per step. This procedure grants us a total of nine independent variables to exploit for optimizing the effectiveness of our innovative approach. We opt to minimize the components associated with the primary error, specifically the Euclidean magnitude of the coefficients e_{9j} and $\tilde{e}_{9j}$ from the ninth-order series expansions (3) and (4).

In the context of double-precision arithmetic, our objective is to maintain small coefficient magnitudes. Large coefficients in the order of 10^4, function values around 10^3, and tolerances like $\varepsilon = 10^{-10}$ would push the limits of the available digits. However, when employing quadruple precision, we can accommodate these larger coefficients even with much lower tolerances, approximately down to 10^{-24}. With this allowance for increased coefficients, we can now proceed to a new minimization approach.

To tackle this challenge, we opt to employ the differential evolution algorithm [14,15]. Differential evolution is an iterative process, and within each iteration, known as a generation g, we operate with a population of individuals denoted as $\left(c_4^{(g)}, c_5^{(g)}, \cdots, w_9'^{(g)}\right)_i$, where $i = 1, 2, \cdots, P$, and P signifies the population size. A population $\left(c_4^{(0)}, c_5^{(0)}, \cdots, w_9'^{(0)}\right)_i$, with $i = 1, 2, \cdots, P$, is randomly generated in the first step of the method. Additionally, we define the fitness function as follows:

$$\Phi = \sqrt{e_{9,1}^2 + e_{9,2}^2 + \cdots + e_{9,36}^2} + \sqrt{\tilde{e}_{9,1}^2 + \tilde{e}_{9,2}^2 + \cdots + \tilde{e}_{9,72}^2} = \left\| T^{(9)} \right\|_2 + \left\| T'^{(9)} \right\|_2,$$

representing the discrepancy resulting from a ninth-order method. The fitness function is then applied to each individual within the initial population, with the objective of minimizing it.

This procedure unfolds in three distinct phases: differentiation, crossover, and selection. For its execution, we utilized the DeMat software [16] within the MATLAB environment [17], where the latter stage is implemented. It is essential to note that achieving a successful outcome is not guaranteed in a single optimization attempt. Consequently, we

iterated through this procedure multiple times, often several hundred, to ultimately reach a solution. To further enhance accuracy, we meticulously refined the obtained result. This fine-tuning process involved working with multi-precision arithmetic and leveraging the `NMinimize` function available in Mathematica [13].

Table 3 provides an overview of the key attributes associated with the primary eighth-order RKN pairs examined in this study. The norms displayed in the table represent the Euclidean norm of the ninth-order coefficients (i.e., those of τ^9) within expressions (3) and (4). We anticipate superior performance from our new method, as it yields considerably diminished local truncation errors.

Table 3. Fundamental traits of the RKN pairs under examination.

Pair	Stages	FSAL	$\left\|T^{(9)}\right\|_2$	$\left\|T'^{(9)}\right\|_2$
RKNT8(6) [18]	9	YES	$1.7 \cdot 10^{-8}$	$1.6 \cdot 10^{-8}$
RKNT8(6)q9	9	NO	$1.8 \cdot 10^{-10}$	$1.8 \cdot 10^{-10}$

The parameters of the new method we have developed are available in the following lists:

$$w_1 = \frac{3191538187421696}{76607108605432915}, \quad w_3 = \frac{13815874303602012}{69579866183121917}, \quad w_4 = \frac{14604812893174087}{79378705834398872}$$
$$w_5 = \frac{12061218770183621}{166622303733231213}, \quad w_6 = \frac{15609617015400}{233291059437933767}, \quad w_7 = \frac{371765604219257}{111475530824146994}$$

$$\hat{w}_1 = \frac{4544292102832777}{109056534231464193}, \quad \hat{w}_3 = \frac{4682651711005479}{23585400043481548}, \quad \hat{w}_4 = \frac{46722285954615265}{253893219962912894}$$
$$\hat{w}_5 = \frac{4751354290135738}{65721585748949841}, \quad \hat{w}_6 = \frac{20872833551830}{134159415686285343}, \quad \hat{w}_7 = \frac{275420922524446}{83046920983443867}$$

$$w'_1 = \frac{3191538187421696}{76607108605432915}, \quad w'_3 = \frac{10308242332317290}{44357423208271919}, \quad w'_4 = \frac{7107618457535881}{21873268413857328}$$
$$w'_5 = \frac{22056521909108756}{75044404292647497}, \quad w'_6 = \frac{15596425292979}{34434009875005756}, \quad w'_7 = \frac{325257858967320448}{9895379989758637}$$
$$w'_8 = -\frac{264730262449877449}{7963593493382224}, \quad w'_9 = \frac{17208373}{35885750}$$

$$\hat{w}'_1 = \frac{4544292102832777}{109056534231464193}, \quad \hat{w}'_3 = \frac{18333976229602070}{78901367072948263}, \quad \hat{w}'_4 = \frac{146694624662575579}{451359699798674378}$$
$$\hat{w}'_5 = \frac{40221502534828457}{137021353651599420}, \quad \hat{w}'_6 = \frac{91894267481143}{87253900673082639}, \quad \hat{w}'_7 = \frac{776789986225611057}{23764274461164518}$$
$$\hat{w}'_8 = -\frac{1116801360586595899}{33934531992244452}, \quad \hat{w}'_9 = \frac{23651021}{71771500}$$

$$c_2 = \frac{2595146787461113}{35654960162808999}, \quad c_3 = \frac{23785164771277655}{163393282122478121}, \quad c_4 = \frac{14427641}{33259908}$$
$$c_5 = \frac{26914142}{35708683}, \quad c_6 = \frac{15577224}{18277247}, \quad c_7 = \frac{38090011}{38093876}$$

$$d_{21} = \frac{295132092736843}{1114198293530546 63}, \quad d_{31} = \frac{378512699615967}{107173587955359337}, \quad d_{32} = \frac{802015671331405}{113542950051902326}$$
$$d_{41} = \frac{9945580188014483}{107861941766479192}, \quad d_{42} = -\frac{21127832523454066}{115356389813386625}, \quad d_{43} = \frac{18088716445271473}{97760613913942175}$$
$$d_{51} = -\frac{184569114806220359}{112841400437628580}, \quad d_{52} = \frac{595308873796066195}{146500969503370446}, \quad d_{53} = -\frac{95938071830688501}{39190010187048758}$$

$$d_{54} = \frac{247402358899752 29}{813283159026444 10}, \quad d_{61} = \frac{828692824853675681}{365166841077510}, \quad d_{62} = -\frac{8922830626242929564}{1616145596072733}$$

$$d_{63} = \frac{5309688443105545745}{1512691814917754}, \quad d_{64} = -\frac{1024584250889564737}{3835297140363491}, \quad d_{65} = \frac{243334944688840544}{26685249097802661}$$

$$d_{71} = -\frac{198499310481410068}{14988189920044743}, \quad d_{72} = \frac{988020934248343439}{30631779844455146}, \quad d_{73} = -\frac{372584950612767755}{18396862167620476}$$

$$d_{74} = \frac{93706617067436735}{54962117018052057}, \quad d_{75} = \frac{2652169291282213}{72706638769934851}, \quad d_{76} = \frac{3326767107636}{4558341505398664 7}$$

$$d_{81} = -\frac{172476446800076249}{77764528330584470}, \quad d_{82} = \frac{1052320941122775251}{32896321613528843}, \quad d_{83} = -\frac{287682559714467205}{6581569888910478}$$

$$d_{84} = \frac{336649615658501777}{14242861273902858}, \quad d_{85} = -\frac{94884627}{9749078}, \quad d_{86} = -\frac{177655963}{35046632}$$

$$d_{87} = \frac{112476592}{20068355}, \quad d_{91} = \frac{1589642054066860483}{2111418052567415}, \quad d_{92} = \frac{206513499}{21728459}$$

$$d_{93} = -\frac{5009179395035143313}{3047562608623994}, \quad d_{94} = -\frac{2192653675860564860}{1440780190602451}, \quad d_{95} = -\frac{1099957025566422337}{1624301323788501}$$

$$d_{96} = -\frac{3640940497065881569}{10360892974776789}, \quad d_{97} = \frac{1917284830561677115}{4934686172719308}, \quad d_{98} = 0.$$

In order to investigate the linear stability, we adopt the methodologies presented in Horn [19] or Dormand et al. [4]. We, therefore, analyze the test problem $\xi'' = \mu^2\xi$ (where μ is a complex number). By considering that $\xi' = \mu\xi$, we deduce the recursive relations for ξ and ξ' as follows:

$$\begin{aligned} \xi_{n+1} &= \left\{1 + \kappa^2 w\left(I - \kappa^2 D\right)^{-1} e + \kappa\left(1 + \kappa^2 w\left(I - \kappa^2 D\right)^{-1} c\right)\right\} \cdot \xi_n = R(\kappa) \cdot \xi_n, \\ \xi'_{n+1} &= \left\{\kappa w'\left(I - \kappa^2 D\right)^{-1} e + \left(1 + \kappa^2 w'\left(I - \kappa^2 D\right)^{-1} c\right)\right\} \cdot \xi'_n = R'(\kappa) \cdot \xi'_n, \end{aligned}$$

with $\kappa = \mu\tau$. Thus, there are two absolute stability regions for RKN methods. Namely, for ζ and ζ'. We may produce them requiring $|R(\kappa)| < 1$ and $|R'(\kappa)| < 1$. Such regions are shown in Figures 1 and 2, where they are compared to the corresponding ones of the competitor method RKNT8(6) [18].

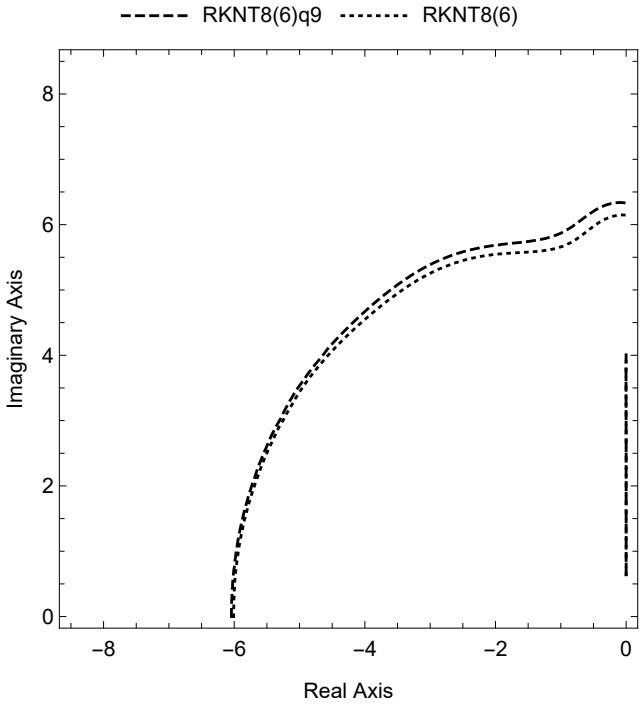

Figure 1. Absolute stability regions for $\xi(x)$.

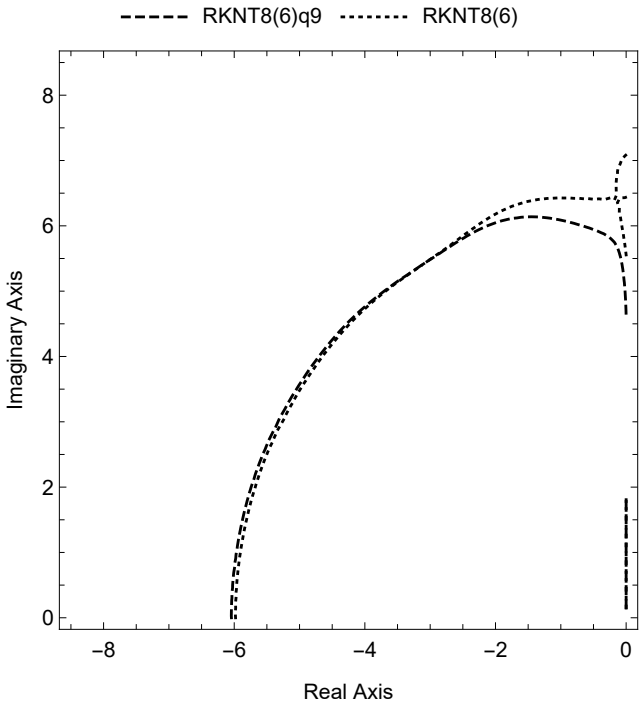

Figure 2. Absolute stability regions for $\xi'(x)$.

This stability analysis type is linked to the A-stability of Runge–Kutta methods. Alternatively, we can explore stability regarding the test problem $\xi'' = -\mu^2\xi$ [20], which leads to the identification of intervals of periodicity. In any case, here we are interested in reaching very high accuracies. In consequence, extended stability regions play a lesser role in achieving this.

4. Numerical Results

Below, we provide numerical tests to demonstrate the effectiveness of our new approach.

4.1. The Methods

We chose the following explicit eighth-order methods for our testing:

- The RKNT8(6) pair with orders $8(6)$, as described in [18].
- The RKNT8(6)q9 pair with orders $8(6)$ introduced in this work.

We conducted the tests using these pairs by assessing the error ϵ at each step. Formula (2) was employed to determine the new step size, as the error's asymptotic behavior was $O(\tau^7)$. All simulations were performed following the framework outlined in the previous section. The selection of RKNT8(6) was justified, as this pair clearly outperformed all other similar pairs (i.e., sharing orders 8(6)) in quadruple precision tests [18]. The latter pair achieved this by itself attaining a very low principal truncation norm (see Table 3).

4.2. The Problems

In our experiments, we chose several established problems from the existing literature. These problems were solved with tolerances of $10^{-20}, 10^{-21}, 10^{-22}, 10^{-23}, 10^{-24}$. During these runs, we collected data on the number of steps taken, including both accepted and rejected steps, as well as the maximum global error observed at the final point. The results are presented in various efficiency plots. All computational work was performed using the software Mathematica [13].

4.2.1. Inhomogeneous Equation

The first test problem we considered was the following inhomogeneous equation:

$$\xi'' = -100\xi(x) + 99 \cdot \sin(x),\ \xi(0) = 1,\ \xi'(0) = 11,$$

which has the theoretical solution,

$$\xi(x) = \cos(10x) + \sin(10x) + \sin(x).$$

We addressed this issue over the range $x \in [0, 10\pi]$. The associated efficiency graphs are presented in Figure 3.

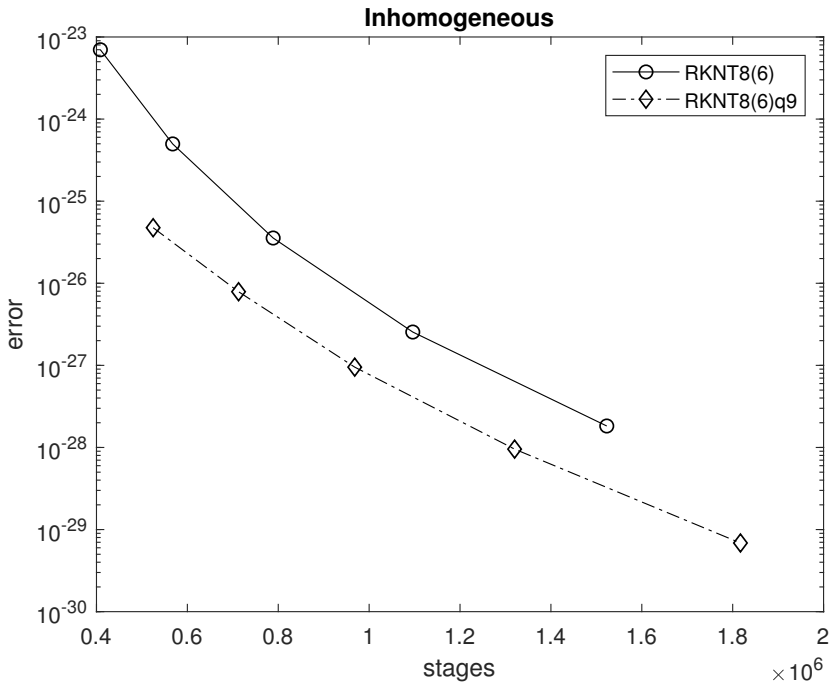

Figure 3. Efficiency plots for the inhomogeneous equation.

4.2.2. Inhomogeneous Linear System

Then, we considered the system

$$\xi'' = \begin{bmatrix} \frac{1}{100} & -\frac{1}{10} \\ -\frac{1}{10} & \frac{1}{100} \end{bmatrix} \cdot \xi + \begin{bmatrix} 0 \\ \sin x \end{bmatrix},$$

with the analytical solution

$$\xi = \begin{bmatrix} \cos 0.3x - \frac{1000}{10101} \sin x \\ \cos 0.3x - \frac{10100}{10101} \sin x \end{bmatrix}.$$

We integrated that problem in the interval $x \in [0, 10\pi]$, and the efficiency plots are shown in Figure 4.

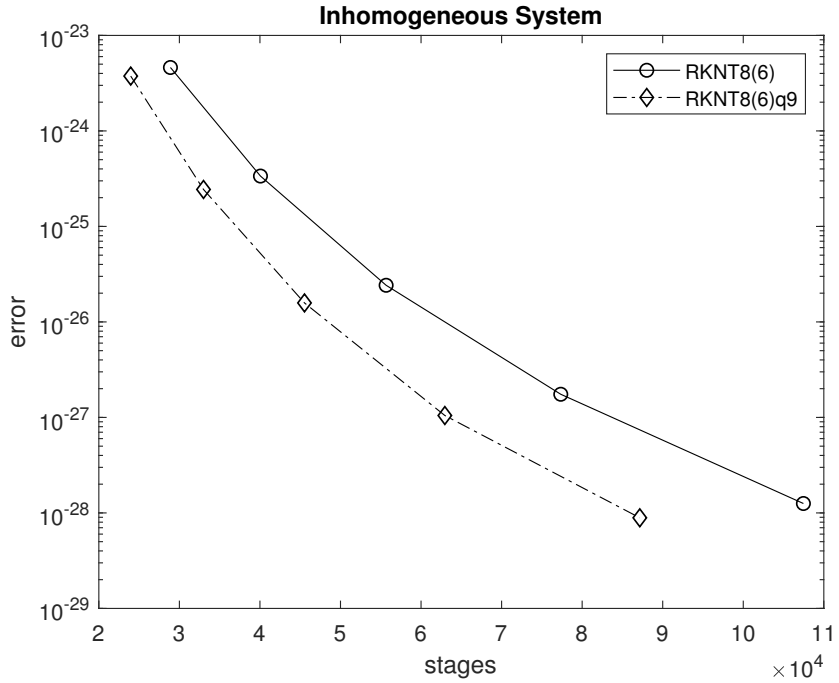

Figure 4. Efficiency plots for the linear inhomogeneous system.

4.2.3. Problem F

We continued with problem F, described in

$$^1\xi'' = -4x^2 \cdot {}^1\xi - 2\frac{^2\xi}{\sqrt{^1\xi^2 + {}^2\xi^2}},\ ^2\xi'' = -4x^2 \cdot {}^2\xi + 2\frac{^1\xi}{\sqrt{^1\xi^2 + {}^2\xi^2}},\ x \in \left[\sqrt{\frac{1}{2}\pi}, 10\right],$$

sharing, initially, the values

$$^1\xi\left(\sqrt{\frac{1}{2}\pi}\right) = 0,\ ^2\xi\left(\sqrt{\frac{1}{2}\pi}\right) = 1,\ ^1\xi'\left(\sqrt{\frac{1}{2}\pi}\right) = -\left(\sqrt{2\pi}\right),\ ^2\xi'\left(\sqrt{\frac{1}{2}\pi}\right) = 0$$

and the analytical solution

$$^1\xi(x) = \cos x^2,\ ^2\xi = \sin x^2.$$

Here, $^1\xi$ and $^2\xi$ are components and time steps.

We conducted an integration of the issue over the interval $x \in [0, 10\pi]$. The solution's theory can be found in [3]. The graphs with the performances are presented in Figure 5.

4.2.4. Coupled Nonlinear Pendulum

In conclusion, we examined a refined version of the nonlinear problem as described in [2], p. 297. The equations governing the motion are as follows:

$$\begin{aligned} ^1\xi'' &= -\sin(^1\xi) - 0.2(\sin(^1\xi) - \sin(^2\xi))\cos(^1\xi) + e^{-10x}, \\ ^2\xi'' &= -\sin(^2\xi) - 0.1(\sin(^2\xi) - \sin(^1\xi))\cos(^2\xi). \end{aligned}$$

We conducted the integration over the interval $x \in [0, 496]$ with an initial state of complete rest, wherein $^1\xi(0) =, ^2\xi(0) =, ^1\xi'(0) =, ^2\xi'(0) = 0$.

As no analytical solution was known, we approximated the solution by performing an integration with a very stringent tolerance of $TOL = 10^{-28}$. The efficiency plots can be found in Figure 6.

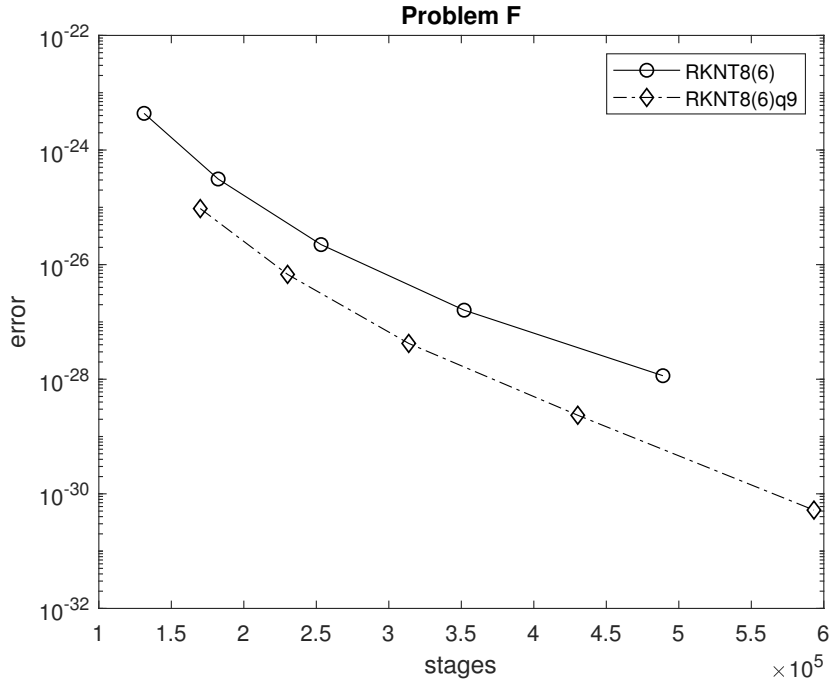

Figure 5. Efficiency plots for problem F.

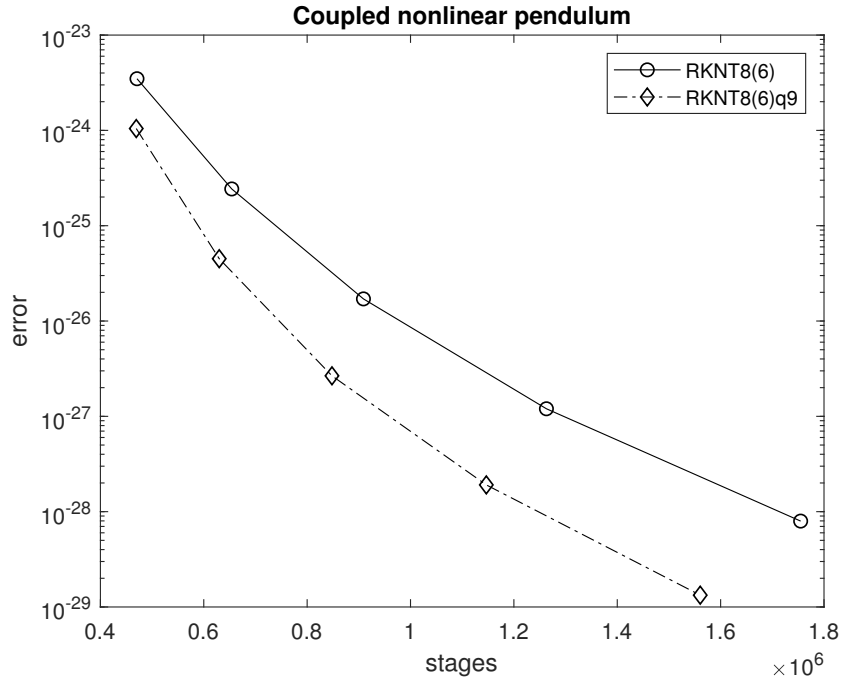

Figure 6. Efficiency plots for the coupled nonlinear pendulum.

4.3. Discussion of the Results

The function evaluation (stages) needed for concluding the integrations for various tolerances and the true global error (i.e., the difference from the theoretical solution) at the end point are plotted in Figures 3–6 for both pairs under consideration. The results indicate that the new pair significantly outperformed the other RKN8(6) pairs in the tested problems, achieving an increase of approximately more than a digit of accuracy in most cases. These findings highlight the superior performance of the new approach when high levels of accuracy are needed for solving specific second-order initial-value problems, surpassing previous methods.

5. Conclusions

In this study, we focused on Runge–Kutta–Nyström pairs tailored for solving second-order initial-value problems where the first derivative is absent. We took advantage of the capabilities of working with quadruple-precision arithmetic, allowing us to handle large coefficients. The primary innovation of our approach lies in the fact that our proposed method features significantly smaller truncation error terms when compared to other eighth-order pairs documented in the literature. Our efforts were substantiated by numerical tests conducted on relevant problems, affirming the effectiveness of our method.

Author Contributions: Conceptualization, K.Y., R.A.R., H.J., T.E.S. and C.T.; methodology, H.A., T.E.S. and C.T.; software, H.A., K.Y., R.A.R., H.J., T.E.S. and C.T.; validation, H.A., K.Y., R.A.R., H.J., T.E.S. and C.T.; formal analysis, H.J., T.E.S. and C.T.; investigation, H.A., K.Y., R.A.R., H.J., T.E.S. and C.T.; resources, T.E.S. and C.T.; data curation, H.A., K.Y., R.A.R., H.J., T.E.S. and C.T.; writing—original draft, C.T.; writing—review & editing, T.E.S.; visualization, H.A., K.Y., R.A.R., H.J., T.E.S. and C.T.; supervision, T.E.S.; project administration, T.E.S.; funding acquisition, H.J. Each author made an equal contribution. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Research Deanship of Hail University-KSA Project Number (BA-2127).

Data Availability Statement: The readers can access the method's coefficients in Mathematica format at the following web address: http://users.uoa.gr/~tsitourasc/rknt86q9.m (accessed on 26 November 2023).

Acknowledgments: The authors acknowledge the Research Deanship of Hail University-KSA for administrative, financial, and technical support.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Hairer, E. Méthodes de Nyström pour l'équation différentielle $y'' = f(x, y)$. *Numer. Math.* **1976**, *27*, 283–300. [CrossRef]
2. Hairer, E.; Nørsett, S.P.; Wanner, G. *Solving Ordinary Differential Equations I: Nonstiff Problems*, 2nd ed.; Springer: Berlin/Heidelberg, Germany, 1993.
3. Fehlberg, E. Eine Runge-Kutta-Nyström-Formel 9-ter Ordnung rnit Schrittweitenkontrolle fur Differentialgleichungen $\ddot{x} = f(t, x)$. *J. Appl. Math. Mech.* **1981**, *61*, 477–485.
4. Dormand, J.R.; El-Mikkawy, M.E.A.; Prince, P.J. Families of Runge-Kutta-Nyström formulae. *IMA J. Numer. Anal.* **1987**, *7*, 235–250. [CrossRef]
5. Dormand, J.R.; El-Mikkawy, M.E.A.; Prince, P.J. High-Order Runge-Kutta-Nyström formulae. *IMA J. Numer. Anal.* **1987**, *7*, 423–430. [CrossRef]
6. El-Mikkawy, M.E.A.; Rahmo, E. A new optimized non-FSAL embedded RungeKuttaNyström algorithm of orders 6 and 4 in six stages. *Appl. Math. Comput.* **2003**, *145*, 33–43.
7. Calvo, M.P.; Sanz-Serna, J.M. High order symplectic Runge-Kutta-Nyström methods. *SIAM J. Sci. Comput.* **1993**, *14*, 1237–1252. [CrossRef]
8. Yoshida, H. Construction of higher order symplectic integrators. *Phys. Lett. A* **1990**, *150*, 262–268. [CrossRef]
9. Brankin, R.W.; Gladwell, I.; Dormand, J.R.; Prince, P.J.; Seward, W.L. ALGORITHM 670: A Runge-Kutta-Nyström Code. *ACM Trans. Math. Softw.* **1989**, *15*, 31–40. [CrossRef]
10. Butcher, J.C. On Runge-Kutta processes of high order. *J. Aust. Math. Soc.* **1964**, *4*, 179–194. [CrossRef]
11. Papakostas, S.N.; Tsitouras, C. High phase-lag order Runge–Kutta and Nyström pairs. *SIAM J. Sci. Comput.* **1999**, *21*, 747–763. [CrossRef]
12. Tsitouras, C.; Famelis, I.T. Symbolic derivation of Runge-Kutta-Nyström order conditions. *J. Math. Chem.* **2009**, *46*, 896–912. [CrossRef]
13. Wolfram Research, Inc. *Mathematica*, Version 11.3.0; Wolfram Research, Inc.: Champaign, IL, USA, 2018.
14. Price, K.V.; Storn, R.M.; Lampinen, J.A. *Differential Evolution, a Practical Approach to Global Optimization*; Springer: Berlin/Heidelberg, Germany, 2005.
15. Storn, R.M.; Price, K.V. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
16. Storn, R.M.; Price, K.V.; Neumaier, A.; Zandt, J.V. DeMat. Available online: https://github.com/mikeagn/DeMatDEnrand (accessed on 26 November 2023).
17. Matlab. *MATLAB*, Version 7.10.0; The MathWorks Inc.: Natick, MA, USA, 2010.

18. Kovalnogov, V.N.; Matveev, A.F.; Generalov, D.A.; Karpukhina, T.V.; Simos, T.E.; Tsitouras, C. Runge-Kutta-Nystrom Pairs of Orders 8(6) for Use in Quadruple Precision Computations. *Mathematics* **2022**, *11*, 891. [CrossRef]
19. Horn, M.K. Developments in High Order Runge-Kutta-Nystrom Formulas. Ph.D. Thesis, The University of Texas, Austin, TX, USA, 1977.
20. der Houwen, P.J.V.; Sommeijer, B.P. Explicit Runge-Kutta (Nyström) methods with reduced phase errors for computing oscillating solutions. *SIAM J. Numer. Anal.* **1987**, *24*, 595–617. [CrossRef]