mathematics

MDPI

Article

General Runge–Kutta–Nyström Methods for Linear Inhomogeneous Second-Order Initial Value Problems

Nadiyah Hussain Alharthi [1], Rubayyi T. Alqahtani [1], Theodore E. Simos [2,3,*] and Charalampos Tsitouras [4]

[1] Department of Mathematics and Statistics, College of Science, Imam Mohammad Ibn Saud Islamic University (IMSIU), P.O. Box 90950, Riyadh 11623, Saudi Arabia; nhalharthi@imamu.edu.sa (N.H.A.); rtalqahtani@imamu.edu.sa (R.T.A.)
[2] Center for Applied Mathematics and Bioinformatics, Gulf University for Science and Technology, West Mishref, Hawally 32093, Kuwait
[3] Section of Mathematics, Department of Civil Engineering, Democritus University of Thrace, GR-67100 Xanthi, Greece
[4] General Department, National and Kapodistrian University of Athens, Euripus Campus, GR-34400 Euboea, Greece; tsitourasc@uoa.gr
* Correspondence: simos@ulstu.ru

Abstract

In this paper, general Runge–Kutta–Nyström (GRKN) methods are developed and analyzed, tailored for second-order initial value problems of the form $y'' = Ly' + My + g(t)$, where $L, M \in \mathbb{R}^{n\times n}$ are constant matrices with $n \geq 1$. The construction of embedded pairs of orders $6(4)$ and $7(5)$, suitable for adaptive integration strategies, is emphasized. By utilizing rooted tree theory and recent simplifications for linear inhomogeneous systems, symbolic order conditions are derived, and efficient schemes are designed through algebraic and evolutionary techniques. Numerical tests verify the superiority of our new derived pairs. In particular, this work introduces novel embedded GRKN pairs with reduced-order conditions that exploit the linearity and structure of the underlying system, enabling the construction of low-stage, high-accuracy integrators. The methods incorporate FSAL (First Same As Last) formulations, making them computationally efficient. They are tested on representative physical systems in one, two, and three dimensions, demonstrating notable improvements in efficiency and accuracy over existing high-order RKN methods.

Keywords: initial value problem; linear inhomogeneous; Runge–Kutta–Nyström; order conditions; differential evolution

MSC: 65L05; 65L06

Academic Editors: Carlo Bianca and Alicia Cordero

Received: 3 July 2025
Revised: 5 August 2025
Accepted: 30 August 2025
Published: 2 September 2025

Citation: Alharthi, N.H.; Alqahtani, R.T.; Simos, T.E.; Tsitouras, C. General Runge–Kutta–Nyström Methods for Linear Inhomogeneous Second-Order Initial Value Problems. *Mathematics* **2025**, *13*, 2826. https://doi.org/10.3390/math13172826

1. Introduction

Initial value problems (IVPs) involving second-order differential equations arise in numerous applications, from classical mechanics to electrical circuits and structural dynamics. When the system is expressed in the general form

$$y'' = f(x, y, y'), \tag{1}$$

it is natural to employ specialized integrators that preserve the second-order structure for efficiency and accuracy. Among these, the Runge–Kutta–Nyström (RKN) family of methods, originally devised for equations of the form $y'' = f(x, y)$, has proven particularly effective.

However, many real-world problems require addressing the more general case where f also depends on y', as in (1). To tackle this, Generalized Runge–Kutta–Nyström (GRKN) methods have been proposed. Unlike classical RKN schemes, these accommodate velocity dependence, at the cost of more intricate order conditions.

Previous work includes the foundational NASA report by E. Fehlberg [1], who developed several high-order methods for such problems. Later, J.M. Fine [2] introduced embedded RKNG pairs of orders $3(4)$ and $4(5)$, emphasizing practical applicability and error control.

Kovalnogov et al. [3] derived Runge–Kutta–Nyström pairs for inhomogeneous linear problems of the $y'' = f(x, y)$, showing that simplifications in the rooted tree framework can significantly reduce the number of necessary order conditions, particularly for linear systems with constant coefficients.

In this work, we focus on the special variation of the problem (1) with the form

$$y'' = Ly' + My + g(t),$$

and constant matrices $L, M \in \mathbb{R}^{n\times n}$, $n \geq 1$. We construct GRKN methods for such linear inhomogeneous systems, targeting embedded pairs of orders $6(4)$ and $7(5)$. These methods are especially relevant in contexts where error control and adaptivity are critical.

GRKN methods target initial value problems; applying them to boundary value problems requires shooting methods, adding complexity and reducing direct applicability.

2. General Runge–Kutta–Nyström Methods and Order Conditions

We consider the general second-order IVP system:

$$y''(x) = f(x, y(x), y'(x)), \quad y(x_0) = y_0, \quad y'(x_0) = y'_0,$$

with $f : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$. In the linear inhomogeneous case,

$$y''(x) = f(x, y, y') = Ly' + My + g(t),\ y(x_0) = y_0,\ y'(x_0) = y'_0 \tag{2}$$

where $L, M \in \mathbb{R}^{n\times n}$, $g(t)$ is a known function and $y(x_0) = y_0$, $y'(x_0) = y'_0$ the initial values.

A general GRKN method with s stages is defined by:

$$\begin{aligned}
Y_i &= y_n + c_i h_n y'_n + h_n^2 \sum_{j=1}^{s} \overline{a}_{ij} f_j, \\
Y'_i &= y'_n + h_n \sum_{j=1}^{s} a_{ij} f_j, \\
f_i &= f(x_n + c_i h_n, Y_i, Y'_i), \\
y_{n+1} &= y_n + h_n y'_n + h_n^2 \sum_{i=1}^{s} d_i f_i, \\
y'_{n+1} &= y'_n + h_n \sum_{i=1}^{s} b_i f_i
\end{aligned} \tag{3}$$

and advances the solution from x_n, y_n, y'_n to $x_{n+1}, y_{n+1}, y'_{n+1}$.

The coefficients $a_{ij}, \overline{a}_{ij}, b_i, d_i, c_i$ form the method's Butcher tableau.

2.1. Butcher Tableau for GRKN Methods

A general tableau for a GRKN method has the structure:

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & \overline{a}_{11} & \overline{a}_{12} & \cdots & \overline{a}_{1s} \\ & \overline{a}_{21} & \overline{a}_{22} & \cdots & \overline{a}_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ & \overline{a}_{s1} & \overline{a}_{s2} & \cdots & \overline{a}_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \\ & d_1 & d_2 & \cdots & d_s \end{array}$$

and in compact form using matrices

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

Here, we deal with explicit methods and A, $\overline{A}$ are strictly lower triangular. Then, f_i's are evaluated explicitly i.e., sequentially. The method provides approximations sequentially in the points $x_1 = x_0 + h_1$, $x_2 = x_1 + h_2$, etc.

Relatively few Runge–Kutta–Nyström–Gear (RKNG) pairs have been developed in the literature. Fehlberg [1] proposed several classes of such methods, including 5(6), 6(7), and 7(8) pairs. In all these pairs the solution advances with the lower-order formula. In addition they do not incorporate a higher-order formula for y', which limits their ability to control the local error in the derivative. The respective methods involve nine, eleven, and fourteen stages; however, the final stage in each step is reused as the initial stage in the subsequent step, effectively reducing the number of function evaluations per step to eight, ten, and thirteen. This tecnique is FSAL (First Stage As Last).

In a separate contribution, Fine [2] introduced a class of 5(4) pairs that require six stages, as well as a class of 4(3) pairs with four stages.

2.2. Rooted Trees and Order Conditions

To derive order conditions for GRKN methods, we use a rooted tree formalism, originally developed by Butcher [4,5] and extended by Hairer et al. [6,7] for general second-order systems. Each rooted tree corresponds to an elementary differential of the solution, and its associated order condition ensures that the GRKN method reproduces the corresponding Taylor expansion term up to a desired order.

In linear inhomogeneous problems, the structure of f dramatically simplifies the set of rooted trees. Specifically, all higher-order derivatives of f vanish, except those that involve the first derivatives of y and y'. As a result, only a subset of the full-order conditions is needed to ensure high-order accuracy.

For example, Fine [2] showed that for a sixth-order method, only 34 order conditions are necessary (compared to more than 200 in the general case), and these can be grouped and handled systematically. Similarly, Simos and Tsitouras [8] derived simplified symbolic forms of order conditions using Frechet derivatives and Taylor expansion matching, facilitating the design of new RK pairs with fewer stages.

2.3. Embedded Pairs and Linear Problems

In adaptive solvers, embedded pairs of the form 6(4) are commonly used. These include:

- A primary method of order $p = 6$ for accuracy.
- A secondary method of order $q = 4$ for error estimation.

For the lower-order formula, we simply need two additional weight vectors $\hat{b} \neq b$ and $\hat{d} \neq d$. Then, using the same functions f_i, $i = 1, 2, \cdots, s$ we may have the fourth-order approximations $\hat{y}$ and $\hat{y}'$.

For the linear inhomogeneous problem (2), the algebraic structure allows for simplifications:

- Many elementary differentials vanish due to linearity.
- This reduces the number of conditions and parameters needed.
- Evolutionary optimization (e.g., Differential Evolution) can be employed to search for coefficients satisfying the order conditions while minimizing local error [8].

In the following, we aim to derive

- a six-stage GRKN pair of orders $6(4)$,
- an eight-stage GRKN pair of orders $7(5)$,

exploiting these simplifications to construct efficient schemes. At each integration step, a local error estimate, denoted as ϵ, is computed. This estimate is typically defined as the maximum absolute difference between the numerical solution and a more accurate reference solution (or an estimate thereof) for both position and velocity components, i.e., $\epsilon = \max\{\|y_{n+1} - \hat{y}_{n+1}\|_\infty, \|y'_{n+1} - \hat{y}'_{n+1}\|_\infty\}$, where y_{n+1} is the current step's numerical approximation and $\hat{y}_{n+1}$ is a higher-order or more refined approximation used for error control. Given a user-specified error tolerance, TOL, the current step is accepted, and the integration proceeds if $\epsilon < TOL$. Conversely, if $\epsilon \geq TOL$, the step is rejected, and the integration is reattempted with a reduced step size. In either scenario (acceptance or rejection), the step size for the subsequent integration, h_{n+1}, is adjusted based on the observed error and the desired tolerance using the formula:

$$h_{n+1} = 0.9 h_n \left(\frac{TOL}{\epsilon} \right)^{\frac{1}{q+1}}. \tag{4}$$

Here, h_n is the current step size, q is the magnitude of the lower order of the pair, and the factor of 0.9 is a safety margin. In the event of a rejected step, the calculated h_{n+1} value serves as the new reduced step size for recomputing the current step. The procedure in a flow chart is given as Algorithm 1.

Algorithm 1 Adaptive embedded GRKN method

Require: Initial values x_0, y_0, y'_0, step size h, tolerance TOL, coefficients $a_{ij}, \bar{a}_{ij}, b_i, \hat{b}_i, d_i, \hat{d}_i, c_i$
Ensure: Approximate values y_1, y'_1 at $x_1 = x_0 + h$

```
Set x_0 ← initial time
repeat
    for i = 1 to s do
        Evaluate g_i = g(x_0 + c_i h)
        Compute f_i = L Y'_i + M Y_i + g_i
        Compute stage values:
```

$$Y_i \ \& \ Y'_i \ \text{as shown in (3)}$$

Algorithm 1 *Cont.*

end for
Compute main solution:

$$y_1 = y_0 + hy'_0 + h^2 \sum_{i=1}^{s} d_i f_i, \quad y'_1 = y'_0 + h \sum_{i=1}^{s} b_i f_i$$

Compute embedded solution:

$$\hat{y}_1 = y_0 + hy'_0 + h^2 \sum_{i=1}^{s} \hat{d}_i f_i, \quad \hat{y}'_1 = y'_0 + h \sum_{i=1}^{s} \hat{b}_i f_i$$

Estimate error:

$$\epsilon = \max\{\|y_{n+1} - \hat{y}_{n+1}\|_\infty, \|y'_{n+1} - \hat{y}'_{n+1}\|_\infty\}$$

if $\epsilon \leq$ TOL **then**
    Accept step:

$$x_0 \leftarrow x_0 + h, \quad y_0 \leftarrow y_1, \quad y'_0 \leftarrow y'_1$$

else
    Reject step
end if
Update step size (for both cases):

$$h \leftarrow 0.9h \left(\frac{TOL}{\epsilon} \right)^{\frac{1}{q+1}}$$

until end of integration interval

A forthcoming section will outline the algorithmic generation of such schemes using symbolic computation and optimization.

3. Method Derivation and Symbolic Conditions for Linear Problems

The scope of this work is deliberately restricted to the integration of linear inhomogeneous second-order initial value problems of the form (2), which allows us to exploit the structure of the problem and derive reduced-order conditions. While the proposed methodology can, in principle, be extended to nonlinear problems or systems with time-dependent coefficients, such generalizations would require solving a fundamentally different and significantly more complex set of order conditions. Moreover, nonlinear cases typically demand a greater number of stages to achieve comparable accuracy, thereby increasing computational cost. These aspects are beyond the present study's focus and are reserved for future investigation. This will be clear below.

In [6], a general theory linking elementary differentials and rooted trees theory was introduced enhancing the relevant connection remarked by Butcher for conventional Runge–Kutta methods.

3.1. Summary: A Theory for General Nyström Methods

In [6], it was proposed that these direct methods offer a more general and flexible approach compared to the common practice of transforming second-order ODEs into a doubled-dimension system of first-order equations. This increased generality provides greater freedom for enhancing both stability and accuracy in numerical solutions, particularly for methods like the General Runge–Kutta–Nyström (GRKN) scheme.

The paper's core contribution lies in developing a theory that extends concepts from first-order equations to the more complex domain of second-order ODEs. This theory is

applicable to the study of a wide range of numerical methods for such problems. While the fundamental theory is often presented for autonomous systems $y'' = f(y, y')$, non-autonomous systems (where f depends explicitly on the independent variable x, as often seen in GRKN methods) can typically be made autonomous by extending the dimension of the system. This extension is built upon the novel concept of "Nyström-trees" (N-trees). These specialized trees, along with their "fat" and "meagre" nodes, provide a graphical representation of the terms arising from repeated differentiation of the ODE solution. Specifically, each "fat" node represents "f" (the function $f(y, y')$ or its autonomous equivalent), and each branch leaving this node represents a derivative: D_2 if the subsequent node is "fat," and D_1 if the adjacent node is "meagre." Meagre nodes have no ramifications and each adjacent node must be fat. The labels on the trees indicate the order of generation of these nodes following this procedure. The set of trees which appear in this way are the monotonically labeled Nyström-trees, denoted by LNT.

Central to the theory is the recursive definition of "Elementary Differentials" ($F(\text{tree}) : E \times E \to E$) for every N-tree 'tree' $\in NT$. These differentials correspond directly to the terms in the Taylor expansion of the solution. For instance, $F(\emptyset)(y, y') = y$, $F(\tau_1)(y, y') = y'$, and $F(\tau_2)(y, y') = f(y, y')$. For a composite N-tree 'tree' $= [\text{tree}_1, \ldots, \text{tree}_k; \text{tree}_{k+1}, \ldots, \text{tree}_m]$, the elementary differential is defined as:

$$F(\text{tree})(y, y') = D_1^k D_2^{m-k} f \cdot (F(\text{tree}_1), \ldots, F(\text{tree}_k), F(\text{tree}_{k+1}), \ldots, F(\text{tree}_m))$$

This allows for the introduction of "Nyström-series," a powerful tool akin to Butcher-series for first-order methods. A Nyström-series is defined with a mapping $a : NT \to \mathbb{R}$ as:

$$N(a, y_0, y_0') = \sum_{\text{tree} \in LNT} a(\text{tree}) F(\text{tree})(y_0, y_0') \frac{h^{e(\text{tree})}}{e(\text{tree})!}$$

Its derivative is $N'(a, y_0, y_0') = \sum_{\text{tree} \in LNT} a(\text{tree}) F(\text{tree})(y_0, y_0') \frac{h^{e(\text{tree})-1}}{(e(\text{tree})-1)!}$. The exact solution of the differential equation is an N-series $N(p, y_0, y_0')$ with $p(\text{tree}) = 1$ for all 'tree' $\in NT$. A significant aspect is that the coefficients $a(\text{tree})$ in these Nyström-series are uniquely determined.

Ultimately, the theoretical machinery developed, particularly the concept of Nyström-trees and Nyström-series, enables the systematic derivation of order conditions for the parameters of various Nyström methods. These conditions are obtained by comparing the Taylor series expansions of a numerical method's solution (e.g., y_{n+1} and y'_{n+1} from a GRKN method) with the exact Nyström-series expansions for the true solution $y(x_{n+1})$ and its derivative $y'(x_{n+1})$. For a method to be of a certain order, its coefficients must satisfy a set of algebraic equations that ensure the numerical solution matches the exact solution's Nyström-series up to that specific order. This theoretical framework provides not only the correct equations of conditions but also an insight into the structure of methods useful for choosing good formulas. By providing a unified and rigorous theoretical foundation, this work validates the benefits of direct Nyström methods and offers a powerful tool for their design, analysis, and optimization in the numerical solution of second-order differential equations.

We now derive a class of general Runge–Kutta–Nyström (GRKN) methods for second-order initial value problems of the form (2). We aim to construct embedded pairs of order $6(4)$ and $7(5)$, exploiting the linear and inhomogeneous structure of the problem to simplify the set of order conditions.

3.2. Order Conditions for GRKN Methods

The local truncation error (LTE) of a numerical method of order p behaves as

$$\text{LTE} = \Delta h^{p+1} + \mathcal{O}(h^{p+2}),$$

where Δ is a constant depending on the solution and its derivatives, and h is the step size.

i.e., for a 7th-order method, the LTE satisfies:

$$\text{LTE} = \Delta_7 h^8 + \mathcal{O}(h^9),$$

with Δ_7 involving derivatives of the true solution up to order eight.

In our implementation, the error is estimated using an embedded pair of order $7(5)$, and the error estimator is computed via

$$\epsilon = \max\Big(\|h^2(\hat{d} - d)f\|_\infty,\ \|h(\hat{b} - b)f\|_\infty \Big),$$

which effectively gives an estimate of the local error of order $\mathcal{O}(h^6)$, since it reflects the difference between the 7th- and 5th-order methods.

Therefore, the key characteristics are:

- True local error of the 7th-order method:

$$\text{LTE} \leq C_7 h^8.$$

- Estimated local error (from the embedded pair):

$$\text{Estimated Error} \approx \mathcal{O}(h^6).$$

- Global error (accumulated over $N = \mathcal{O}(1/h)$ steps):

$$\text{Global Error} \approx \mathcal{O}(h^7).$$

This analysis confirms the high accuracy of the method, particularly for smooth problems with controlled step sizes.

In conclusion, the behavior of numerical methods can be encoded using rooted trees, where each node corresponds to a derivative in the Taylor expansion. For general $f(t, y, y')$, we must consider both partial derivatives with respect to y and y', leading to a large tree set.

Fehlberg [1] presented the equations of condition in an elaborate form, employing indices and dummy inner variables. Here, in Tables 1 and 2, we consolidate the equations of condition for the weights b and d, up to the sixth algebraic order (i.e., Taylor matching). Our approach, however, adopts a more straightforward implementation utilizing only matrices and vectors. In these tables, the symbol "$\cdot$" denotes the classical dot product, whereas "$\odot$" represents the component-wise (Hadamard) multiplication of vectors. For instance, if $v, u \in \mathbb{R}^s$, then

$$v \odot u = [v_1 u_1,\ v_2 u_2, \cdots, v_s u_s].$$

This operation has the lowest precedence among operators. For example, in the expression $c \odot A \cdot c$, the dot product is evaluated first, followed subsequently by the component-wise multiplication.

Table 1. Equations of condition for y' for orders 1–6. Whenever † does not appear to the left of an equation, then it is not valid for problem (2).

$^\dagger b \cdot e = 1$	$^\dagger b \cdot c = \frac{1}{2}$	$^\dagger b \cdot c^2 = \frac{1}{3}$
$^\dagger b \cdot A \cdot c = \frac{1}{6}$	$^\dagger b \cdot c^3 = \frac{1}{4}$	$b \cdot (c \odot A \cdot c) = \frac{1}{8}$
$^\dagger b \cdot \overline{A} \cdot c = \frac{1}{24}$	$^\dagger b \cdot A \cdot c^2 = \frac{1}{12}$	$^\dagger b \cdot A \cdot A \cdot c = \frac{1}{24}$
$^\dagger b \cdot c^4 = \frac{1}{5}$	$b \cdot (c^2 \odot A \cdot c) = \frac{1}{10}$	$b \cdot (c \odot \overline{A} \cdot c) = \frac{1}{30}$
$b \cdot (c \odot A \cdot c^2) = \frac{1}{15}$	$^\dagger b \cdot \overline{A} \cdot c^2 = \frac{1}{60}$	$^\dagger b \cdot A \cdot c^3 = \frac{1}{20}$
$b \cdot (A \cdot c)^2 = \frac{1}{20}$	$b \cdot (c \odot A \cdot A \cdot c) = \frac{1}{30}$	$^\dagger b \cdot \overline{A} \cdot A \cdot c = \frac{1}{120}$
$b \cdot A \cdot (c \odot A \cdot c) = \frac{1}{40}$	$^\dagger b \cdot A \cdot \overline{A} \cdot c = \frac{1}{120}$	$^\dagger b \cdot A \cdot A \cdot c^2 = \frac{1}{60}$
$^\dagger b \cdot A \cdot A \cdot A \cdot c = \frac{1}{120}$	$^\dagger b \cdot c^5 = \frac{1}{6}$	$b \cdot (c^3 \odot A \cdot c) = \frac{1}{12}$
$b \cdot (c^2 \odot \overline{A} \cdot c) = \frac{1}{36}$	$b \cdot (c^2 \odot A \cdot c^2) = \frac{1}{18}$	$b \cdot (c \odot \overline{A} \cdot c^2) = \frac{1}{72}$
$b \cdot (c \odot A \cdot c^3) = \frac{1}{24}$	$b \cdot (c \odot (A \cdot c)^2) = \frac{1}{24}$	$^\dagger b \cdot \overline{A} \cdot c^3 = \frac{1}{120}$
$b \cdot (A \cdot c \odot \overline{A} \cdot c) = \frac{1}{72}$	$^\dagger b \cdot A \cdot c^4 = \frac{1}{30}$	$b \cdot (A \cdot c^2 \odot A \cdot c) = \frac{1}{36}$
$b \cdot (c^2 \odot A \cdot A \cdot c) = \frac{1}{36}$	$b \cdot (c \odot \overline{A} \cdot A \cdot c) = \frac{1}{144}$	$b \cdot (c \odot A \cdot (c \odot A \cdot c)) = \frac{1}{48}$
$b \cdot (c \odot A \cdot \overline{A} \cdot c) = \frac{1}{144}$	$b \cdot (c \odot A \cdot A \cdot c^2) = \frac{1}{72}$	$b \cdot \overline{A} \cdot (c \odot A \cdot c) = \frac{1}{240}$
$^\dagger b \cdot \overline{A} \cdot \overline{A} \cdot c = \frac{1}{720}$	$^\dagger b \cdot \overline{A} \cdot A \cdot c^2 = \frac{1}{360}$	$b \cdot A \cdot (c^2 \odot A \cdot c) = \frac{1}{60}$
$b \cdot A \cdot (c \odot \overline{A} \cdot c) = \frac{1}{180}$	$b \cdot A \cdot (c \odot A \cdot c^2) = \frac{1}{90}$	$^\dagger b \cdot A \cdot \overline{A} \cdot c^2 = \frac{1}{360}$
$^\dagger b \cdot A \cdot A \cdot c^3 = \frac{1}{120}$	$b \cdot A \cdot (A \cdot c)^2 = \frac{1}{120}$	$b \cdot (A \cdot c \odot A \cdot A \cdot c) = \frac{1}{72}$
$b \cdot (c \odot A \cdot A \cdot A \cdot c) = \frac{1}{144}$	$^\dagger b \cdot \overline{A} \cdot A \cdot A \cdot c = \frac{1}{720}$	$b \cdot A \cdot (c \odot A \cdot A \cdot c) = \frac{1}{180}$
$^\dagger b \cdot A \cdot \overline{A} \cdot A \cdot c = \frac{1}{720}$	$b \cdot A \cdot A \cdot (c \odot A \cdot c) = \frac{1}{240}$	$^\dagger b \cdot A \cdot A \cdot \overline{A} \cdot c = \frac{1}{720}$
$^\dagger b \cdot A \cdot A \cdot A \cdot c^2 = \frac{1}{360}$	$^\dagger b \cdot A \cdot A \cdot A \cdot A \cdot c = \frac{1}{720}$	

Table 2. Equations of condition for y for orders 1–6. For † see explanation in previous Table.

$^\dagger d \cdot e = \frac{1}{2}$	$^\dagger d \cdot c = \frac{1}{6}$	$^\dagger d \cdot c^2 = \frac{1}{12}$
$^\dagger d \cdot A \cdot c = \frac{1}{24}$	$^\dagger d \cdot c^3 = \frac{1}{20}$	$d \cdot (c \odot A \cdot c) = \frac{1}{40}$
$^\dagger d \cdot \overline{A} \cdot c = \frac{1}{120}$	$^\dagger d \cdot A \cdot c^2 = \frac{1}{60}$	$^\dagger d \cdot A \cdot A \cdot c = \frac{1}{120}$
$^\dagger d \cdot c^4 = \frac{1}{30}$	$d \cdot (c^2 \odot A \cdot c) = \frac{1}{60}$	$d \cdot (c \odot \overline{A} \cdot c) = \frac{1}{180}$
$d \cdot (c \odot A \cdot c^2) = \frac{1}{90}$	$^\dagger d \cdot \overline{A} \cdot c^2 = \frac{1}{360}$	$^\dagger d \cdot A \cdot c^3 = \frac{1}{120}$
$d \cdot (A \cdot c)^2 = \frac{1}{120}$	$d \cdot (c \odot A \cdot A \cdot c) = \frac{1}{180}$	$^\dagger d \cdot \overline{A} \cdot A \cdot c = \frac{1}{720}$
$d \cdot A \cdot (c \odot A \cdot c) = \frac{1}{240}$	$^\dagger d \cdot A \cdot \overline{A} \cdot c = \frac{1}{720}$	$^\dagger d \cdot A \cdot A \cdot c^2 = \frac{1}{360}$
$^\dagger d \cdot A \cdot A \cdot A \cdot c = \frac{1}{720}$		

Additionally, we define as $c^2 = c \odot c$, $c^3 = c \odot c^2, \cdots$. As an exception, this operation has first priority. In the first equation of these Tables, we observe that $e = [1, 1, 1, \cdots, 1] \in \mathbb{R}^s$.

3.3. Order Conditions for Linear Inhomogeneous Problems

However, for linear inhomogeneous problems, the right-hand side in (2) has the following properties:

- All higher-order derivatives $f_{yy}, f_{y'y'}, f_{yy'}, \ldots$ vanish.
- Nonlinear combinations of trees reduce to simpler linear combinations.
- Many trees collapse into equivalent forms.

The symmetry of the trees allows for reuse and simplification of these terms. For example, many trees result in identical algebraic expressions for linear systems, which reduces

redundancy. In Tables 1 and 2, we mark the equations of condition for the case of interest here by using † in the upper left of the equation.

The order conditions can now be formulated through the following couple of equations:

$$b \cdot \Psi(A, \overline{A}) \cdot c^j = \frac{j!}{p!}, \quad \text{for } p-1 \geq j \geq 1,\ p \geq 1, \tag{5}$$

$$d \cdot \overline{\Psi}(A, \overline{A}) \cdot c^j = \frac{j!}{p!}, \quad \text{for } p-1 \geq j \geq 1,\ p \geq 2, \tag{6}$$

where $\Psi(A, \overline{A})$ and $\overline{\Psi}(A, \overline{A})$ denote matrix products involving A and $\overline{A}$.

To facilitate the analysis, we define the *RKN-rank* of matrix A to be 1, and the *RKN-rank* of matrix $\overline{A}$ to be 2. The *sum-RKN-rank* r of an expression involving Ψ or $\overline{\Psi}$ is then defined as the total sum of the individual RKN-ranks of the matrices appearing in the corresponding matrix product. For example, the expression $\overline{A} \cdot A \cdot A$ has a sum-RKN-rank of $r = 4$.

To achieve a method of order p for the derivative y', the condition $r + j = p - 1$ must be satisfied. In contrast, to achieve the same order p for the function y, the condition becomes $r + j = p - 2$.

For instance, consider the condition

$$b \cdot A \cdot \overline{A} \cdot A \cdot c = \frac{1}{720},$$

where we identify $r = 4$, $j = 1$, and thus infer that $p = 6$ for y', i.e., Equations (5).

Similarly, for the condition

$$d \cdot A \cdot \overline{A} \cdot A \cdot c = \frac{1}{5040},$$

we again have $r = 4$, $j = 1$, which corresponds to $p = 7$ for y, i.e., Equations (6).

3.4. Producing the New Methods

For a conventional pair of order $6(4)$, we have 56, 22, 9, and 4 equations of condition for b, d, $\hat{b}$, and $\hat{d}$, respectively. Indeed, only the first three rows in Table 1 suffice for deducing the fourth-order conditions for $\hat{b}$, where it is enough to replace b with $\hat{b}$.

However, in the context of linear inhomogeneous problems, analysis of the tables confirms that only 27, 15, 8, and 4 equations of condition remain valid for b, d, $\hat{b}$, and $\hat{d}$, respectively. Consequently, we arrive at a total system comprising 54 equations.

We proceed by selecting $s = 7$ and employing the FSAL (First Same As Last or First Stage As Last) principle, i.e., $a_{sj} = b_j$ and $\overline{a}_{sj} = d_j$ for $1 \leq j \leq s$. This implies that the method requires only six stages per step. Furthermore, we impose the simplifications $A \cdot e = c$ and $\overline{A} \cdot e = A \cdot c$, thereby determining the first column of A and $\overline{A}$.

For the new pair of orders $7(5)$, we choose $s = 9$ and again employ the FSAL strategy, resulting in a method that utilizes eight stages per step. From Equations (5) and (6), it follows that, under the linear inhomogeneous problem framework, there now exist 47, 27, 15, and 8 conditions to be satisfied for b, d, $\hat{b}$, and $\hat{d}$, respectively.

To address the determination of coefficients for the two problems described above, we employed the Differential Evolution (DE) technique [9]. DE is an iterative procedure in which, at each iteration (referred to as generation g), we maintain a population of individuals—namely, the free parameters

$$\left(c_2^{(g)}, c_3^{(g)}, \cdots, \hat{a}_{s-1}^{(g)}, \hat{d}_s^{(g)}\right), \quad i = 1, 2, \ldots, N,$$

where N denotes the population size. An initial population

$$\left(c_2^{(0)}, c_3^{(0)}, \cdots, \hat{d}_{s-1}^{(0)}, \hat{d}_s^{(0)}\right), \quad i = 1, 2, \ldots, N,$$

is randomly generated at the outset of the algorithm. Subsequently, we define a fitness function to minimize, ideally driving it to zero. Specifically, this function is defined as the square root of the sum of squared residuals

$$T = \left((b \cdot e - 1)^2 + (b \cdot c - \tfrac{1}{2})^2 + \cdots + (d \cdot e - 1)^2 + \cdots + (\hat{b} \cdot e - 1)^2 + \cdots\right)^{0.5}.$$

This fitness function is then evaluated for each member of the initial population. The DE procedure proceeds through three sequential phases—Differentiation, Crossover, and Selection—updating all individuals in the population at each generation g.

For implementing this optimization approach, we utilized the MATLAB [10] software DeMat [11]. Recent studies, such as [12], demonstrate the broad applicability of hybrid metaheuristic strategies in computational mathematics and engineering contexts, further supporting the use of evolutionary approaches like Differential Evolution in the present work. Accordingly, Tsitouras and Famelis [13], demonstrate the effectiveness of leveraging symbolic computation—particularly rooted tree expansions—for automating and streamlining the derivation of order conditions in the design of efficient numerical integrators.

As a result, we successfully constructed two methods, whose coefficients are presented in Tables 3 and 4.

Table 3. Coefficients of the 6(4) pair.

c	A						
0	0	0	0	0	0	0	
$\frac{137878459}{588903905}$	$\frac{137878459}{588903905}$	0	0	0	0	0	
$\frac{7491243902}{8820050637}$	$-\frac{6912565003}{2383548425}$	$\frac{3549781859}{946745408}$	0	0	0	0	
$\frac{1255892843}{1981008397}$	$-\frac{332863433}{671077243}$	$\frac{624923549}{584067693}$	$\frac{19931549}{332029333}$	0	0	0	
$\frac{828617034}{877777573}$	$\frac{1447820040}{569227477}$	$-\frac{4577789031}{1550725129}$	$-\frac{612583895}{977387216}$	$\frac{1801563031}{910202194}$	0	0	
$\frac{251309330}{527492149}$	$\frac{225803829}{522734068}$	$\frac{440925161}{301931773}$	$\frac{1367472924}{733541063}$	$-\frac{1664626223}{450640808}$	$\frac{1241932216}{3001210703}$	0	
1	$\frac{92306947}{1303649843}$	$\frac{429325261}{1225378013}$	$\frac{221851245}{3287539774}$	$\frac{420269659}{1217598825}$	$\frac{740841171}{5528332403}$	$\frac{31129404}{967389425}$	
	0	0	0	0	0	0	
	0	0	0	0	0	0	
	$\frac{209798341}{238991032}$	0	0	0	0	0	
$\overline{A} \rightarrow$	$\frac{55984799}{732668533}$	$\frac{204286275}{907624337}$	0	0	0	0	
	$\frac{166571266}{1150705629}$	$-\frac{53953832}{232314825}$	$\frac{183479015}{1544223716}$	0	0	0	
	$-\frac{1255284611}{575881630}$	$\frac{2556741844}{1407985387}$	$-\frac{354912741}{737708101}$	$\frac{861639665}{1051992621}$	0	0	
	$\frac{139623777}{1998501059}$	$\frac{151075790}{551923941}$	$-\frac{5431732}{1654707983}$	$\frac{57784943}{394768749}$	$\frac{16965417}{1274071093}$	0	
$b \rightarrow$	$\frac{92306947}{1303649843}$	$\frac{429325261}{1225378013}$	$\frac{221851245}{3287539774}$	$\frac{420269659}{1217598825}$	$\frac{740841171}{5528332403}$	$\frac{31129404}{967389425}$	
$\hat{b} \rightarrow$	$\frac{36719078}{466615877}$	$\frac{208071437}{627991452}$	$\frac{7005164}{483157681}$	$\frac{1928335569}{4601595221}$	$\frac{125234026}{1338671715}$	$\frac{21346033}{1658393989}$	$\frac{1}{20}$
$d \rightarrow$	$\frac{139623777}{1998501059}$	$\frac{151075790}{551923941}$	$-\frac{5431732}{1654707983}$	$\frac{57784943}{394768749}$	$\frac{16965417}{1274071093}$		
$\hat{d} \rightarrow$	$\frac{102105254}{1366647511}$	$\frac{153643847}{584456494}$	$-\frac{3997442}{654400525}$	$\frac{207616691}{1340523277}$	$\frac{12972367}{1078624536}$	$\frac{752567}{467741276}$	

Table 4. Coefficients of the 7(5) pair.

c	A								
0	0	0	0	0	0	0	0	0	
$\frac{1}{8}$	$\frac{1}{8}$	0	0	0	0	0	0	0	
$\frac{1}{5}$	$\frac{53349690}{603180907}$	$\frac{202053157}{1811278965}$	0	0	0	0	0	0	
$\frac{2}{5}$	$-\frac{6018455}{1047543353}$	$-\frac{331258825}{1424686717}$	$\frac{1575913757}{2469082868}$	0	0	0	0	0	
$\frac{1}{2}$	$-\frac{34276137}{600042728}$	$-\frac{119693677}{470929744}$	$\frac{456514532}{673786419}$	$\frac{91782119}{686213876}$	0	0	0	0	
$\frac{3}{5}$	$\frac{104326539}{715754762}$	$-\frac{528081604}{478040477}$	$\frac{232828616}{145770633}$	$-\frac{346503067}{803949037}$	$\frac{458528309}{1167636595}$	0	0	0	
$\frac{4}{5}$	$\frac{164370736}{719166091}$	$\frac{287836583}{475340755}$	$-\frac{434202437}{787748133}$	$\frac{776183737}{2072773349}$	$-\frac{359707736}{568196253}$	$\frac{297706885}{383789789}$	0	0	
$\frac{5}{6}$	$\frac{571234282}{1292256085}$	$-\frac{163644024}{598795985}$	$\frac{434129672}{1024184363}$	$-\frac{458249359}{1591656232}$	$-\frac{226893932}{1525451613}$	$\frac{720660496}{1154927353}$	$\frac{70287235}{1317288008}$	0	
1	$\frac{84500844}{419992723}$	$-\frac{577927735}{469547576}$	$\frac{15167575312}{6875238419}$	$-\frac{1847505748}{646735723}$	$\frac{2648529023}{782098598}$	$-\frac{1167018767}{1234225733}$	$-\frac{1515995803}{1684134096}$	$\frac{924250955}{811144576}$	
	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	
	$\frac{6718965}{481850228}$	0	0	0	0	0	0	0	
	$\frac{54942235}{2006066137}$	$\frac{15472181}{217307372}$	0	0	0	0	0	0	
$\overline{A} \rightarrow$	$\frac{13577792}{495771991}$	$\frac{22050438}{495716135}$	$\frac{283627865}{3322412337}$	0	0	0	0	0	
	$-\frac{14180539}{845563863}$	$\frac{27053183}{151491698}$	$-\frac{13187092}{1461421841}$	$\frac{14285958}{271989619}$	0	0	0	0	
	$\frac{21160793}{121602384}$	$-\frac{85686929}{101458060}$	$\frac{250790469}{239063945}$	$-\frac{36154403}{86286747}$	$\frac{49695151}{163139864}$	0	0	0	
	$\frac{8462875}{72563974}$	$-\frac{79891174}{158211209}$	$\frac{36044633}{52796893}$	$-\frac{47016078}{174876551}$	$\frac{27008407}{127844653}$	$\frac{4837109}{116867752}$	0	0	
	$\frac{67824649}{2783623662}$	$\frac{48741640}{205101599}$	$-\frac{23811983}{397137837}$	$\frac{17380466}{88975787}$	$\frac{16168641}{556147049}$	$\frac{7125373}{559402722}$	$\frac{64065334}{1053745503}$	0	
$b \rightarrow$	$\frac{84500844}{419992723}$	$-\frac{577927735}{469547576}$	$\frac{15167575312}{6875238419}$	$-\frac{1847505748}{646735723}$	$\frac{2648529023}{782098598}$	$-\frac{1167018767}{1234225733}$	$-\frac{1515995803}{1684134096}$	$\frac{924250955}{811144576}$	
$\hat{b} \rightarrow$	$-\frac{320560198}{1387413679}$	$\frac{3304935953}{2148629380}$	$-\frac{934978493}{716035472}$	$\frac{119856567}{781419025}$	$\frac{3541196691}{4411805198}$	$-\frac{498658693}{2238995507}$	$-\frac{278833107}{894718040}$	$\frac{239425724}{454342689}$	$\frac{1}{20}$
$d \rightarrow$	$\frac{67824649}{2783623662}$	$\frac{48741640}{205101599}$	$-\frac{23811983}{397137837}$	$\frac{17380466}{88975787}$	$\frac{16168641}{556147049}$	$\frac{7125373}{559402722}$	$\frac{64065334}{1053745503}$		
$\hat{d} \rightarrow$	$\frac{17896581}{105671711}$	$-\frac{83237081}{155999235}$	$\frac{234468734}{296248719}$	$-\frac{29643419}{142582240}$	$\frac{7438697}{37050289}$	$\frac{13041803}{327653383}$	$-\frac{14651331}{867442538}$	$\frac{25835861}{453483297}$	

4. Numerical Tests

The methods considered for tests are three already known pairs from the literature and both pairs given above. Namely,

- GRKNF5(6), a fifth-order method, effectively using six stages pair step, appeared in [1]
- GRKNF6(7), a sixth-order method, effectively using ten stages pair step, appeared in [1].
- Fine5(4), an FSAL fifth-order method, effectively using six stages pair step, appeared in [2].
- NEW6(4), the FSAL pair constructed here, effectively using six stages pair step.
- NEW7(5), the FSAL pair constructed here, effectively using eight stages pair step.

The order of the pairs is included in their names. e.g., Fine5(4) is a pair using orders five and four. The solution advances with the fifth-order formula. GRKNF5(6) is a fifth-order formula since the solution advances with the fifth-order formula. The sixth-order formula is accompanying only solution in y and not in y', i.e., in the notation given here, there are only vectors $b, \hat{b}, \hat{d}$ and the solution propagates through $\hat{b}, \hat{d}$. Similar logic holds for GRKNF6(7).

To evaluate the performance of Runge–Kutta–Nyström (RKN) methods designed for systems of the form (2) with $L, M \in \mathbb{R}^{n\times n}$, we consider three test cases that arise from relevant physical and engineering applications. These problems are linear and inhomogeneous, providing structured challenges that are well suited for the development and testing of high-order RKN methods.

4.1. Damped Harmonic Oscillator with External Forcing ($n = 1$)

The classic damped harmonic oscillator with external forcing is given by:

$$y''(t) + \lambda y'(t) + \omega^2 y(t) = g(t).$$

Rewriting in standard form yields:

$$y''(t) = -\lambda y'(t) - \omega^2 y(t) + g(t),$$

with

$$L = -\lambda, \quad M = -\omega^2, \quad g(t) = A\sin(\nu t).$$

We consider a numerical example with parameters:

$$\lambda = 5, \quad \omega = 1, \quad A = 1, \quad \nu = \frac{1}{10},$$

and initial conditions:

$$y(0) = 0, \quad y'(0) = 0.$$

This problem is representative of mechanical systems with damping and periodic excitation, such as vibrating beams or mass–spring–damper systems under sinusoidal forcing [14]. All computations were carried out over the interval $[0, 10]$. Using the known value

$$y(10) \approx 0.50814725856006851284$$

as a reference, we measured the number of function evaluations required by each of the five pairs in relation to their corresponding endpoint errors. The resulting efficiency comparison is displayed in Figure 1. In Appendix A, we include a MATLAB script that contains the coefficients of the pair 7(5). By running it, we may reproduce the lowermost line of this figure.

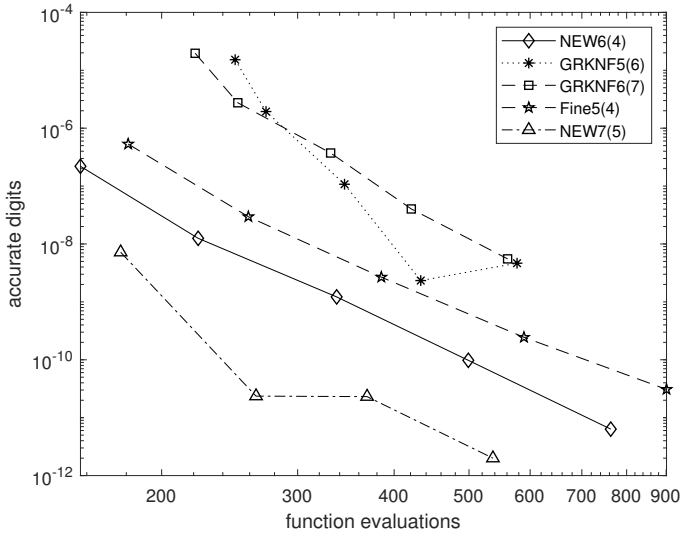

Figure 1. Efficiency curves for Problem 4.1 showing the number of function evaluations required by various GRKN pairs versus achieved digits of accuracy.

4.2. Coupled Oscillators in a 2D Framework ($n = 2$)

We next examine a two-mass system with coupling, damping, and time-varying forcing. The parameters from (2) take the form:

$$L = \begin{pmatrix} -4 & 0 \\ 0 & -0.3 \end{pmatrix}, \quad M = \begin{pmatrix} -2 & 1 \\ 1 & -3 \end{pmatrix}, \quad g(t) = \begin{pmatrix} \sin(t) \\ \cos(t) \end{pmatrix},$$

while the initial values are:

$$y(0) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad y'(0) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

This system models two coupled masses subject to damping and external excitation. Such models appear in robotic manipulators, seismology, and mechanical vibrations of plates or membranes with interacting modes [15]. All computations were performed on the interval $[0, 10]$. The known approximation

$$y(10) \approx \begin{bmatrix} 0.1566961779698483 \\ -0.4529092672497892 \end{bmatrix},$$

was employed as a reference solution. For each of the five method pairs, the number of function evaluations required to achieve their respective endpoint errors was recorded. The resulting comparison of computational efficiency is illustrated in Figure 2.

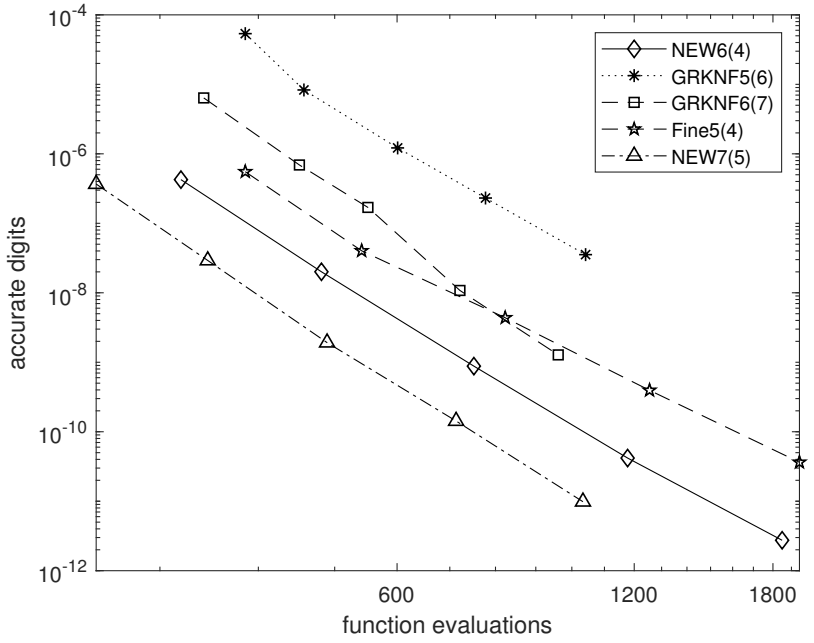

Figure 2. Efficiency curves for Problem 4.2.

4.3. Multi-Mass Coupled Oscillators with Damping and Forcing ($n = 3$)

A more complex example involves three masses connected via springs and dampers, a system common in structural and control applications. Now, the parameters from (2) take the form:

$$L = \begin{pmatrix} -6 & 0.2 & 0 \\ 0.1 & -7 & 0.1 \\ 0 & 0.3 & -5 \end{pmatrix}, \quad M = \begin{pmatrix} -5 & 2 & 0 \\ 2 & -6 & 2 \\ 0 & 2 & -5 \end{pmatrix}, \quad g(t) = \begin{pmatrix} \sin(t) \\ \cos(2t) \\ e^{-t} \end{pmatrix},$$

and the initial values are:

$$y(0) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad y'(0) = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}.$$

Such configurations occur in rail vehicle suspensions, aerospace structure control, and biomechanical systems where joint dynamics are modeled using second-order linear systems with coupling [16–18].

All computations were conducted over the interval $[0, 10]$. The following known approximation was utilized as a reference solution:

$$y(10) \approx \begin{bmatrix} 0.0622697554888544436 \\ 0.09716732533321522028 \\ 0.0103120325178873458 \end{bmatrix}.$$

For each of the five pairs of methods, the number of function evaluations necessary to achieve their respective endpoint errors was recorded. A comparative analysis of the computational efficiency is presented in Figure 3.

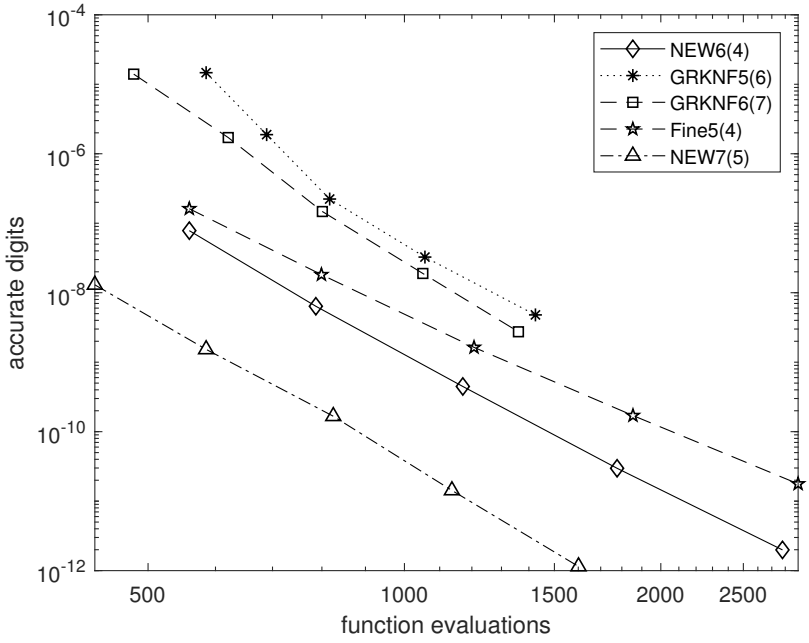

Figure 3. Efficiency curves for Problem 4.3.

Interpreting the results, it is evident that the NEW7(5) pair exhibits superior efficiency compared to all other methods. The NEW6(4) pair follows as the second most efficient. This enhanced performance can be attributed to the reduced number of stages required in their construction, as both methods are specifically tailored for problems of the form (2).

5. Conclusions

This study introduces novel embedded General Runge–Kutta–Nyström (GRKN) methods specifically designed for linear inhomogeneous second-order systems, with a focus on constructing efficient pairs of orders $6(4)$ and $7(5)$. A key contribution lies in the symbolic derivation of order conditions tailored to the linear structure, which allows for low-stage, high-accuracy schemes. The incorporation of FSAL properties and the use of evolutionary techniques for parameter optimization further enhance computational performance.

The proposed methods exhibit clear advantages over existing RKN-type integrators, especially in high-dimensional and tightly coupled systems. Future research may explore extensions to nonlinear or variable coefficient problems.

Author Contributions: Conceptualization, T.E.S. and C.T.; Methodology, T.E.S. and C.T.; Software, T.E.S. and C.T.; Validation, N.H.A., R.T.A., T.E.S. and C.T.; Formal analysis, N.H.A., R.T.A., T.E.S. and C.T.; Investigation, N.H.A., R.T.A., T.E.S. and C.T.; Resources, N.H.A., R.T.A., T.E.S. and C.T.; Data curation, N.H.A., R.T.A., T.E.S. and C.T.; Writing—original draft, C.T.; Writing—review & editing, T.E.S. and C.T.; Visualization, T.E.S. and C.T.; Supervision, T.E.S. and C.T.; Project administration, T.E.S.; Funding acquisition, N.H.A. and R.T.A. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported and funded by the Deanship of Scientific Research at Imam Mohammad Ibn Saud Islamic University (IMSIU) (grant number IMSIU-DDRSP-RP25).

Data Availability Statement: The original contributions presented in this study are included in the article. The listing in Appendix A can be also retrieved from http://users.uoa.gr/~tsitourasc/grknt75.m (accessed on 29 August 2025). Any additional data should be sought from the last author.

Conflicts of Interest: The authors affirm that they have no competing interests.

Appendix A. MATLAB Code

In the following, we present a MATLAB implementation of the NEW7(5) embedded pair. This method is specifically applied to Problem 4.1 for various tolerance levels, aiming to reproduce the lowermost curve depicted in Figure 1. The main routine, `grkn75`, accepts as input the initial value problem provided as an anonymous function of the form `@(x,y,dy)`, the initial point x_0, the final point, the initial conditions y_0, y'_0, and the desired tolerance.

The output includes the vector `tout`, containing the integration points $x_0, x_1, \ldots$, as well as the vectors `yout` and `dyout`, which store the corresponding approximations $y_0, y_1, y_2, \ldots$ and $y'_0, y'_1, y'_2, \ldots$, respectively. In addition, the scalar `fev` returns the total number of function evaluations (stages) performed during the integration.

```
% problem41
function problem41
resul=zeros(5,2);
irep=1;
for tol=[1e-6,1e-7,1e-8,1e-9],
    [tout, yout, dyout, fev] = grkn75(@(x,y,dy) -5*dy(1)-y(1)+sin(x/10), 0,10, 0, 0, tol);
     resul(irep,:)=[fev max(abs(yout(:,end)-0.50814725856006851284))];
     irep=irep+1;
end;

axis([150 900 4 12]);
loglog(resul(:,1),resul(:,2),'-.^k');
xlabel('function evaluations');
ylabel('accurate digits');
legend('NEW7(5)');
return


% grkn 7(5)
function [tout, yout, dyout, fev] = grkn75(FunFcn, t0, tfinal, y0, dy0, tol);

%------------------------------------------------------
% my new method 7(5)
c=[0,1/8,1/5,2/5,1/2,3/5,4/5,5/6,1]';
b=[0.20119597167401398108,-1.2308182696272720194,2.2061162664677621851, ...
-2.8566625938490179854,3.3864387812136187983,-0.94554726562324883883, ...
-0.90016335789451293256,1.1394404676386568117,0];
bb=[-0.23104875124991469820,1.5381600865012839022,-1.3057711936930409504, ...
```

```
0.15338322099337164246,0.80266388293964741919,-0.22271536117021783691, ...
-0.31164355085541809419,0.52697166653428861587,0.05];
a=[[0,0,0,0,0,0,0,0,0],
[0.125,0,0,0,0,0,0,0,0],
[0.088447245894008380024,0.11155275410599161998,0,0,0,0,0,0,0],
[-0.005745303984569314462,-0.23251345088521661222, ...
0.63825875486978592666,0,0,0,0,0,0],
[-0.057122827093073277786,-0.25416461483902363251,0.67753596559208772069, ...
0.13375147634000918960,0,0,0,0,0],
[0.14575738023521525598,-1.1046796859421592459,1.5972258006178789110, ...
-0.43100128372938146827,0.39269778881844654711,0,0,0,0],
[0.22855740566333236569,0.60553735393465262555,-0.55119449835624047120, ...
0.37446628565273008921,-0.63306953205127876816,0.77570298515680415892,0,0,0],
[0.44204418043038272803,-0.27328445646475068150,0.42387844189337618372, ...
-0.28790724390541638102,-0.14873885875264402674,0.62398772886280406679, ...
0.053575304513058307060,0,0],
b];
ahat=[[0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0],
[0.01394409426324895250,0,0,0,0,0,0,0,0],
[0.02738804767531949790,0.07119952193798561090,0,0,0,0,0,0,0],
[0.02738717040592154645,0.04448198564285182197,0.08536805075076989749, ...
0,0,0,0,0,0],
[-0.01677051210500938968,0.17857865056077198410,-0.00902346716741041029, ...
0.05252390900992437864,0,0,0,0,0],
[0.1740162676415949263,-0.8445551689042742859,1.049051830044885967, ...
-0.41900296693303323240,0.30461684705094525610,0,0,0,0],
[0.11662639921016454470,-0.50496532138882775870,0.68270367727888819070, ...
-0.26885295787884107250,0.21125957454004747520,0.04138959565167301271,0,0,0],
[0.02436559579726689012,0.23764631888608535300,-0.05995898849597635318, ...
0.19533927808921773220,0.02907260054525608803,0.01273746572867050314, ...
0.06079772944947978678,0,0]];
bhat=[0.024365595797266890,0.237646318886085353,-0.05995898849597635, ...
0.19533927808921773220,0.02907260054525608803,0.012737465728670503, ...
0.06079772944947978678,0,0];
bbhat=[0.169360189502373042,-0.53357364861436657,0.79145906450316170, ...
-0.20790400683843930400,0.20077298182478413180,0.03980365739120111860, ...
-0.01689026115064696005,0.05697202338193284059,0];
pow=1/6;
%-------------------------------------------------------

if nargin < 5, tol = 1.e-6; end

% Initialization
t = t0;
hmax = (tfinal - t)/5;
hmin = (tfinal - t)/2000000;
y = y0(:);
dy = dy0(:);
f = y*zeros(1,length(c));
tout = t;
yout = y;
dyout=dy;
fev = 1;

f(:,1) = feval(FunFcn,t,y,dy);
h=tol^pow/max(max(abs(f(:,1))),1);

% The main loop
   while (t < tfinal) && (h >= hmin)
      if t + h > tfinal, h = tfinal - t; end

      % Compute the slopes
```

```
   f(:,1) = feval(FunFcn,t,y,dy);
   for j = 2:length(c),
      f(:,j) = feval(FunFcn, t+c(j)*h, y+c(j)*h*dy+h^2*(ahat(j,:)*f')', ...
                                                    dy+h*(a(j,:)*f')');
   end

   % Estimate the error and the acceptable error
   delta = max(norm(h^2*(bhat-bbhat)*f','inf'), norm(h*(b-bb)*f','inf'));

   % Update the solution only if the error is acceptable
   if delta <= tol,
      t = t + h;
      y = y + h*dy + h^2*(bhat*f')';
      dy = dy + h *(b*f')';
      tout = [tout t];
      yout = [yout y];
      dyout = [dyout dy];
      fev = fev+length(c)-1;
   else
      fev = fev+length(c)-1;
   end

   % Update the step size
   if delta ~= 0.0
      h = min(hmax, 0.9*h*(tol/delta)^pow);
   end
end;

if (t < tfinal)
   disp('SINGULARITY LIKELY.')
   t
end
```

References

1. Fehlberg, E. 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. Available online: https://ntrs.nasa.gov/api/citations/19740026877/downloads/19740026877.pdf (accessed on 2 July 2025).
2. Fine, J.M. Low Order Practical Runge–Kutta–Nyström Methods. *Computing* **1987**, *38*, 281–297. [CrossRef]
3. Kovalnogov, V.N.; Fedorov, R.V.; Karpukhina, M.T.; Kornilova, M.I.; Simos, T.E.; Tsitouras, C. Runge–Kutta–Nyström Methods of Eighth Order for Addressing Linear Inhomogeneous Problems. *J. Comput. Appl. Math.* **2023**, *419*, 114778. [CrossRef]
4. Butcher, J.C. An Algebraic Theory of Integration Methods. *Math. Comput.* **1972**, *26*, 79–106. [CrossRef]
5. Butcher, J.C. *Numerical Methods for Ordinary Differential Equations*, 2nd ed.; John Wiley & Sons: Chichester, UK, 2003.
6. Hairer, E.; Wanner, G. A Theory for Nyström Methods. *Numer. Math.* **1976**, *25*, 377–395.
7. Hairer, E.; Nørsett, S.P.; Wanner, G. *Solving Ordinary Differential Equations I: Nonstiff Problems*, 2nd ed.; Springer: Berlin/Heidelberg, Germany, 1993.
8. Simos, T.E.; Tsitouras, C. Evolutionary Derivation of Runge–Kutta Pairs for Addressing Inhomogeneous Linear Problems. *Numer. Algorithms* **2021**, *87*, 511–525. [CrossRef]
9. Storn, R.; Price, K. Differential Evolution—A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
10. *MATLAB*; Version 7.10.0; The MathWorks Inc.: Natick, MA, USA, 2010.
11. Storn, R.; Price, K.; Neumaier, A.; Zandt, J.V. DeMat. Available online: https://github.com/mikeagn/DeMatDEnrand (accessed on 2 July 2025).
12. Mazraeh, H.D.; Parand, K. An innovative combination of deep Q-networks and context-free grammars for symbolic solutions to differential equations. *Eng. Appl. Artif. Intell.* **2025**, *142*, No109733. [CrossRef]
13. Tsitouras, C.; Famelis, I.T. Symbolic Derivation of Runge–Kutta–Nyström Order Conditions. *J. Math. Chem.* **2009**, *46*, 896–912. [CrossRef]
14. Nayfeh, A.H.; Mook, D.T. *Nonlinear Oscillations*; Wiley-VCH: Weinheim, Germany, 2008.
15. Inman, D.J. *Engineering Vibration*, 4th ed.; Pearson: Boston, MA, USA, 2014.

16. Bingham, C.M.; Birkett, N.M.; Sims, N.D. Control of satellite structures using passive and semi-active vibration isolation. *J. Sound Vib.* **2006**, *294*, 1–18.
17. Craig, R.R., Jr.; Kurdila, A.J. *Fundamentals of Structural Dynamics*, 2nd ed.; Wiley: Hoboken, NJ, USA, 2006.
18. Meirovitch, L. *Analytical Methods in Vibrations*; Macmillan: New York, NY, USA, 1967.