AIMS Mathematics, 11(1): 2935–2953.
DOI: 10.3934/math.2026117
Received: 28 September 2025
Revised: 17 January 2026
Accepted: 20 January 2026
Published: 29 January 2026

https://www.aimspress.com/journal/Math

Research article

Runge-Kutta pairs for scalar autonomous Neural ODEs

Ibraheem Alolyan[1], **Theodore E. Simos**[2,3,*] **and Charalampos Tsitouras**[4]

[1] Mathematics Dept., College of Science, King Saud University, P.O. Box 2455, Riyadh, 11451, Saudi Arabia

[2] Center for Applied Mathematics and Bioinformatics, Gulf University for Science and Technology, West Mishref, 32093, Kuwait; ORCID: https://orcid.org/0000-0002-9220-6924

[3] Section of Mathematics, Dept. of Civil Engineering, Democritus Univ. of Thrace, Xanthi 67100, Greece

[4] General Department, National & Kapodistrian University of Athens, 34400 Euripus Campus, Greece; ORCID: https://orcid.org/0000-0001-6801-8117

* **Correspondence:** Email: tsimos.conf@gmail.com.

Abstract: Runge–Kutta (RK) pairs remain among the most effective tools for the numerical integration of ordinary differential equations; however in the scalar autonomous setting, their structure allows further efficiency. In this case, the system of order conditions simplifies, with fewer equations needing to be satisfied, which in turn enables the construction of embedded pairs of higher accuracy than in the general case. Notably, one may design pairs of order 7(5) requiring only eight stages per step, whereas conventional RK pairs with the same number of stages are limited to order 6(5). In the present work, we derived the modified set of equations of condition up to seventh order, and by means of differential evolution techniques, constructed a new embedded pair of order 7(5) specifically adapted to scalar autonomous problems. The performance of the method was assessed in the context of system identification through neural ODEs, where it was used to approximate governing dynamics from data. The logistic growth and saturating cubic models were employed as a representative test cases, illustrating both the efficiency advantages of the proposed scheme. Numerical experiments confirmed that the new pair provides a valuable bridge between high–order RK methodology and modern machine learning approaches to dynamical systems.

Keywords: order conditions; integer partitions; evolutionary optimization; neural ODEs; system identification
Mathematics Subject Classification: 65L05, 65L06, 05C05, 90C26, 68T07

1. Introduction

The first-order initial value problem (IVP) can be stated as

$$y' = f(x, y), \quad y(x_0) = y_0 \in \mathbb{R}^m, \quad x \in [x_0, x_e], \tag{1.1}$$

where $f : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}^m$. Formulation (1.1) serves as a mathematical model for a wide range of applications in science and engineering. Recently, it has also become the foundation of an emerging research area known as neural ordinary differential equations (neural ODEs). In this framework, the vector field f is represented or approximated by a neural network, thereby combining numerical analysis with modern machine learning techniques. Neural ODEs provide a continuous-time counterpart to deep residual networks, allow for adaptive computation of hidden states, and offer a natural way to model dynamical systems from data [6, 8, 16, 20]. This synthesis has opened new directions in system identification, time-series prediction, and scientific machine learning, where classical ODE solvers, such as DP5(4) [7] and Tsit5 [26], play a central role in the training and evaluation of these models.

Among the widely used numerical techniques for solving (1.1) are Runge–Kutta (RK) pairs, which can be represented by an extended Butcher tableau [2, 3]:

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

where b^T, $\widehat{b}^T$, and c belong to $\mathbb{R}^s$ and $A \in \mathbb{R}^{s\times s}$ is strictly lower triangular. To advance the numerical solution from (x_n, y_n) to $x_{n+1} = x_n + h_n$, the method computes two approximations, y_{n+1} and $\widehat{y}_{n+1}$, of the exact solution $x(t_{n+1})$ at each step. These approximations correspond to orders p and q (with $q < p$), and are defined as

$$y_{n+1} = y_n + h_n \sum_{i=1}^{s} b_i f_{ni}, \tag{1.2}$$

and

$$\widehat{y}_{n+1} = y_n + h_n \sum_{i=1}^{s} \widehat{b}_i f_{ni},$$

where

$$f_{ni} = f\left(x_n + c_i h_n,\, y_n + h_n \sum_{j=1}^{i-1} a_{ij} f_{nj}\right), \tag{1.3}$$

for $i = 1, 2, \ldots, s \geq p$. The simplifying relation

$$A \cdot e = c, \quad e = [1, 1, \ldots, 1]^T \in \mathbb{R}^s, \tag{1.4}$$

is satisfied by all methods considered in this work.

The local error estimate is given by

$$\epsilon_{n+1} = \left\| y_{n+1} - \widehat{y}_{n+1} \right\|,$$

which serves as the basis for adaptive step-size control:

$$h_{n+1} = 0.9 \cdot h_n \cdot \left(\frac{\varepsilon}{\epsilon_{n+1}}\right)^{1/(q+1)}$$

where ε denotes the user-specified tolerance. This update is applied even when $\varepsilon < \epsilon_{n+1}$, in which case h_{n+1} becomes a reduced step size relative to h_n. For further discussion on this adaptive strategy, see [1, 21].

In what follows, the general RK formulation is retained, but the subsequent theoretical analysis and derivation of order conditions will focus on scalar autonomous initial value problems of the form

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

with $f : \mathbb{R} \to \mathbb{R}$. Throughout the paper, x denotes the independent variable of the IVP and is retained in the RK stage formulation for notational consistency. In the scalar autonomous case considered here, f does not depend explicitly on x, so that $f(x, y) \equiv f(y)$, and the appearance of x in the stage arguments plays no analytical role.

In recent years, neural ordinary differential equations have continued to attract sustained interest, both from a methodological perspective and in applications to data-driven modeling. Several recent contributions emphasize the central role of numerical integration schemes in neural ODE frameworks, including issues of stability, accuracy, and solver efficiency during training and inference [5, 17]. These developments further motivate the study of high-order RK pairs and their performance in contemporary neural ODE settings.

2. RK methods and rooted trees theory

2.1. Expansions of Taylor series

We may, without restriction, assume that $x' = 1$, so that (1.1) reduces to the autonomous form $y' = f(y)$, which is more convenient to handle. When the p-order RK method (1.2-1.3) is applied to this formulation, the objective is essentially to approximate the associated Taylor-type system

$$y(x_{n+1}) \approx y(x_n) + hy'(x_n) + \frac{1}{2!}y''(x_n) + \cdots + \frac{1}{p!}y^{(p)}(x_n). \tag{2.1}$$

Conversely, by expanding f_{ni} about the point (x_n, y_n), relation (1.2) leads to

$$y_{n+1} = y_n + hq_{11}x'_n + h^2q_{21}y''_n + h^3\left(q_{31}f'f + q_{32}f''f^2\right) + \cdots, \tag{2.2}$$

where the terms f_{ij} are determined solely by the coefficients b, c, and A.

We obtain the following identities:

$$\begin{aligned} y'' &= \frac{\partial f(y(x))}{\partial x} = \frac{\partial f}{\partial y} f = f'f, \\ y''' &= \frac{\partial^2 f}{\partial y^2} \cdot (f, f) + \frac{\partial f}{\partial y} \cdot \frac{\partial f}{\partial y} \cdot f = f''(f, f) + f'f'f, \end{aligned}$$

$$\begin{aligned} y'''' &= \frac{\partial^3 f}{\partial y^3}\cdot(f,f,f)+\frac{\partial f}{\partial y}\cdot\frac{\partial f}{\partial y}\cdot\frac{\partial f}{\partial y}\cdot f \\ &\quad +3\cdot\frac{\partial^2 f}{\partial y^2}\cdot(\tfrac{\partial f}{\partial y}\cdot f,\ f)+\frac{\partial f}{\partial y}\cdot\frac{\partial^2 f}{\partial y^2}\cdot(f,f) \\ &= f'''(f,f,f)+f'f'f'f+3f''(f'f,\ f)+f'f''(f,f), \\ &\quad \cdots \end{aligned}$$

In this context, the expressions $f'''(f,f,f)$, $f''(f,f)$, $f''(f'f,f)$, and $f'f''(f,f)$ are interpreted as Fréchet derivatives, following the notation and definitions in [15, p. 158].

By comparing (2.1) with (2.2), we obtain

$$\begin{aligned} &y(x_{n+1})-y_{n+1}=h(\phi_{11}-1)f+h^2\Big(\phi_{21}-\tfrac{1}{2}\Big)\tfrac{\partial f}{\partial y}f+ \\ &h^3\Big[\Big(\phi_{31}-\tfrac{1}{6}\Big)f^T\cdot\tfrac{\partial^2 f}{\partial y^2}\cdot f+\Big(\phi_{32}-\tfrac{1}{6}\Big)\tfrac{\partial f}{\partial y}\cdot\tfrac{\partial f}{\partial y}\cdot f\Big]+\cdots \end{aligned} \tag{2.3}$$

Imposing the conditions $q_{11}=\phi_{11}-1=0$, $q_{21}=\phi_{21}-\frac{1}{2}=0$, $q_{31}=\phi_{31}-\frac{1}{6}=0$, and $q_{32}=\phi_{32}-\frac{1}{6}=0$, we establish the necessary conditions that must be satisfied for the development of a third-order method. The algebraic requirements up to order five are presented in the first column of Table 1. Within this table, the notation c^2 refers to the component-wise (Hadamard) product of the vector c, that is, $c^2 = c\odot c$.

Table 1. The equations of condition of RK methods, for orders 1–5.

Order conditions	Elementary differential	Order of appeared derivatives
$q_{11}=b\cdot e-1$	f	
$q_{21}=b\cdot c-\frac{1}{2}$	$f'f$	1
$q_{31}=\frac{1}{2}b\cdot c^2-\frac{1}{6}$	$f''\cdot(f,f)$	2
$q_{32}=b\cdot A\cdot c-\frac{1}{6}$	$f'\cdot f'\cdot f$	1 – 1
$q_{41}=\frac{1}{6}b\cdot c^3-\frac{1}{24}$	$f'''\cdot(f,f,f)$	3
$q_{42}=\frac{1}{2}b\cdot A\cdot c^2-\frac{1}{24}$	$f'\cdot f''\cdot(f,f)$	1 – 2
$q_{43}=b\cdot(c\odot(A\cdot c))-\frac{1}{8}$	$f''\cdot(f'\cdot f,f)$	2 – 1
$q_{44}=b\cdot A^2\cdot c-\frac{1}{24}$	$f'\cdot f'\cdot f'\cdot f$	1 – 1 – 1
$q_{51}=\frac{1}{24}b\cdot c^4-\frac{1}{120}$	$f''''\cdot(f,f,f,f)$	4
$q_{52}=b\cdot\Big(c^2\odot(A\cdot c)\Big)-\frac{1}{20}$	$f'''\cdot(f'\cdot f,f,f)$	3 – 1
$q_{53}=\frac{1}{2}b\cdot\Big(c\odot(A\cdot c^2)\Big)-\frac{1}{30}$	$f''\cdot(f''\cdot(f,f),f)$	2 – 2
$q_{54}=b\cdot\Big(c\odot(A^2\cdot c)\Big)-\frac{1}{30}$	$f''\cdot(f'\cdot f'\cdot f,f)$	2 – 1 – 1
$q_{55}=\frac{1}{2}b\cdot(A\cdot c)^2-\frac{1}{40}$	$f''\cdot(f'\cdot f,f'\cdot f)$	2 – 1 – 1
$q_{56}=\frac{1}{6}b\cdot A\cdot c^3-\frac{1}{120}$	$f'\cdot f'''\cdot(f,f,f)$	1 – 3
$q_{57}=b\cdot A\cdot(c\odot(A\cdot c))-\frac{1}{40}$	$f'\cdot f''\cdot(f'\cdot f,f)$	1 – 2 – 1
$q_{58}=\frac{1}{2}b\cdot A^2\cdot c^2-\frac{1}{120}$	$f'\cdot f'\cdot f''(f,f)$	1 – 1 – 2
$q_{59}=b\cdot A^3\cdot c-\frac{1}{120}$	$f'\cdot f'\cdot f'\cdot f'\cdot f$	1 – 1 – 1 – 1

In a similar manner, $(A \cdot c)^2$ denotes the element-wise product of the components of the vector $A \cdot c$, that is, $(A \cdot c)^2 = (A \cdot c) \odot (A \cdot c)$. More generally, c^i represents the vector obtained by raising each entry of c to the power i. These operations are always carried out first. In the following tables, component-wise products of vectors are indicated using the Hadamard product symbol $\odot$. For instance, $c^3 = (c^2) \odot c$. Whenever $\odot$ occurs, it is considered of lower precedence compared with other operations. Furthermore, we set $c^0 = e$ and introduce the notation $C = \mathrm{diag}(c)$.

The third column of Table 1 provides a numbering of the occurrences of derivatives within the corresponding elementary differentials. For example, in the case of q_{32} we find in two instances f', which are recorded in the notation $1 - 1$.

2.2. Trees and rooted trees

Equation (2.3) may alternatively be expressed as

$$y(x_{n+1}) - y_{n+1} = \sum_{i=1}^{\infty} \sum_{\tau \in T_i} h^i \frac{1}{\sigma(\tau)} \left(\Phi(\tau) - \frac{1}{\gamma(\tau)} \right) F(\tau),$$

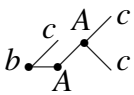
where T_i denotes the set of rooted trees of order i, and the functions σ and γ assign integer values to each tree τ. The quantity Φ is a polynomial involving the coefficients A, b, and c, while $F(\tau)$ refers to the corresponding elementary differential as defined in [4].

A RK method is said to be of order p if and only if the following condition holds:

$$X(\tau) = \frac{1}{\sigma(\tau)} \left(\Phi(\tau) - \frac{1}{\gamma(\tau)} \right) = 0, \quad \text{for all } \tau \in T_i, \text{ for } i = 1, 2, \ldots, p.$$

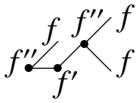
This relation above forms a set of equations of condition, which are linear in the elements of b and nonlinear in the elements of A, c (see, for example, Butcher [4] or Hairer, Nørsett and Wanner [12]). The notation $T^{(i)}$ refers to the vector composed of all elements belonging to the set $X(T_i)$, arranged in a specified, albeit arbitrary, order.

Each order condition corresponds uniquely to a particular rooted tree τ. This correspondence becomes evident when one assigns b to the root, places A at the internal nodes, and assigns c to the leaves, subsequently applying a prefix-style multiplication. Using this systematic representation, one can construct each polynomial $\Phi(\tau)$ directly from the associated order condition. For instance, consider the labeling of the nodes in the tree τ



that corresponds to $\Phi(\tau) = b \cdot \left(c \odot \left(A^2 \cdot c^2 \right) \right)$.

In an analogous manner, the corresponding elementary differential is derived. Each node is assigned to $f^{(\mu)}$, where μ denotes the number of its successors. For terminal nodes, $f^{(0)}$ is interpreted simply as f. Applying this rule to the same tree as considered previously, we obtain



and conclude to the elementary differential

$$F(\tau) = f''(f, f' f''(f, f)).$$

In the second column of Table 1, we interpret this relation for order conditions up to five.

The number of rooted trees (i.e., the number of equations of condition) are given in the second row of Table 2 for orders up to ten.

Table 2. The numbers for order conditions for systems of equations and for scalar autonomous problems.

Order	1	2	3	4	5	6	7	8	9	10
System	1	1	2	4	9	20	48	115	286	719
Scalar aut.	1	1	2	3	5	7	11	15	22	30

For $m > 1$ in (1.1) we observe

$$F(q_{43}) = f''(f'f, f) \neq f'f''(f, f) = F(q_{42}).$$

Then, two different equations of condition are required (i.e., q_{42} and q_{43}). In the scalar autonomous case, we have

$$f' \cdot f''(f, f) = f'' \cdot f' \cdot f^2 = f''(f' \cdot f, f).$$

Then, we only need one order condition $q^*_{42} = q_{42} + q_{43}$.

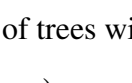
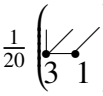
When we record the derivative degrees at the corresponding nodes for each tree τ of order i, we find that we come to a relation of trees with the integer partitions of number $i - 1$ [22]. So q_{42}  is associated to $1 - 2$ while q_{43}  is associated to $2 - 1$, [19]. These two trees resulted in q^*_{42} indeed. Similarly, $q_{52} = \frac{1}{2}b \cdot \left(c^2 * (A \cdot c)\right) - \frac{1}{20}$  corresponding to $3 - 1$ and $t_{53} = \frac{1}{6}b \cdot A \cdot c^3 - \frac{1}{120}$  , corresponding to $1 - 3$ combine to $q^*_{52} = \frac{1}{2}b \cdot \left(c^2 \odot (A \cdot c)\right) - \frac{1}{20} + \frac{1}{6}b \cdot A \cdot c^3 - \frac{1}{120}$, [19]. This relation is explained in the third column of Table 1. The remaining equations of the condition for orders $1 - 6$ are listed in Table 3. In the latter table, we observe for example that $q^*_{54} = q_{54} + q_{55} + q_{57} + q_{58}$ since q_{54}, q_{55}, q_{57}, and q_{58} are associated with the same partition of number 4. The equations for order seven can be found in Table 4.

Table 3. The equations of condition of orders one through six for scalar autonomous RK methods.

$$q^*_{11} = b \cdot e - 1$$

$$q^*_{21} = b \cdot c - \tfrac{1}{2}$$

$$q^*_{31} = \tfrac{1}{2} b \cdot c^2 - \tfrac{1}{6}$$

$$q^*_{32} = b \cdot A \cdot c - \tfrac{1}{6}$$

$$q^*_{41} = \tfrac{1}{6} b \cdot c^3 - \tfrac{1}{24}$$

$$q^*_{42} = \tfrac{1}{2} b \cdot A \cdot c^2 - \tfrac{1}{24} + b \cdot (c \odot (A \cdot c)) - \tfrac{1}{8}$$

$$q^*_{43} = b \cdot A^2 \cdot c - \tfrac{1}{24}$$

$$q^*_{51} = \tfrac{1}{24} b \cdot c^4 - \tfrac{1}{120}$$

$$q^*_{52} = \tfrac{1}{2} b \cdot \left(c^2 \odot (A \cdot c)\right) - \tfrac{1}{20} + \tfrac{1}{6} b \cdot A \cdot c^3 - \tfrac{1}{120}$$

$$q^*_{53} = \tfrac{1}{2} b \cdot \left(c \odot \left(A \cdot c^2\right)\right) - \tfrac{1}{30}$$

$$q^*_{54} = b \cdot \left(c \odot \left(A^2 \cdot c\right)\right) - \tfrac{1}{30} + b \cdot A \cdot (c \odot (A \cdot c)) - \tfrac{1}{40} + \tfrac{1}{2} b \cdot A^2 \cdot c^2 - \tfrac{1}{120} + \tfrac{1}{2} b \cdot (A \cdot c)^2 - \tfrac{1}{40}$$

$$q^*_{55} = b \cdot A^3 \cdot c - \tfrac{1}{120}$$

$$q^*_{61} = \tfrac{1}{120} b \cdot c^5 - \tfrac{1}{720}$$

$$q^*_{62} = \tfrac{1}{6} b \cdot \left(c^3 \odot (A \cdot c)\right) - \tfrac{1}{72} + \tfrac{1}{24} b \cdot A \cdot c^4 - \tfrac{1}{720}$$

$$q^*_{63} = \tfrac{1}{4} b \cdot \left(c^2 \odot \left(A \cdot c^2\right)\right) - \tfrac{1}{72} + \tfrac{1}{6} b \cdot \left(c \odot \left(A \cdot c^3\right)\right) - \tfrac{1}{144}$$

$$q^*_{64} = \tfrac{1}{2} b \cdot \left(c \odot (A \cdot c)^2\right) - \tfrac{1}{48} + \tfrac{1}{2} b \cdot \left(c^2 \odot \left(A^2 \cdot c\right)\right) - \tfrac{1}{72} + \tfrac{1}{6} b \cdot A^2 \cdot c^3 - \tfrac{1}{720}$$
$$+ \tfrac{1}{2} b \cdot A \cdot \left(c^2 \odot (A \cdot c)\right) - \tfrac{1}{120}$$

$$q^*_{65} = b \cdot (c \odot (A \cdot (c \odot (A \cdot c)))) - \tfrac{1}{48} + \tfrac{1}{2} b \cdot \left(\left(A \cdot c^2\right) \odot (A \cdot c)\right)$$
$$- \tfrac{1}{72} + \tfrac{1}{2} b \cdot \left(c \odot \left(A^2 \cdot c^2\right)\right) - \tfrac{1}{144} + \tfrac{1}{2} b \cdot A \cdot \left(c \odot \left(A \cdot c^2\right)\right) - \tfrac{1}{180}$$

$$q^*_{66} = \tfrac{1}{2} b \cdot A^3 \cdot c^2 - \tfrac{1}{720} + b \cdot A^2 \cdot (c \odot (A \cdot c)) - \tfrac{1}{240} + b \cdot A \cdot \left(c \odot \left(A^2 \cdot c\right)\right) - \tfrac{1}{180}$$
$$+ b \cdot \left(c \odot \left(A^3 \cdot c\right)\right) - \tfrac{1}{144} + \tfrac{1}{2} b \cdot A \cdot (A \cdot c)^2 - \tfrac{1}{240} + b \cdot \left((A \cdot c) \odot (A^2 \cdot c)\right) - \tfrac{1}{72}$$

$$q^*_{67} = b \cdot A^4 \cdot c - \tfrac{1}{720}$$

In consequence, the theory of unrestricted partitions of an integer is associated to the enumeration of order conditions [22, p. 122]. In the last row of Table 2, we list the order conditions up to tenth

order.

Table 4. Truncation error coefficients of seventh order.

$$
\begin{aligned}
q^*_{71} &= \tfrac{1}{720} b \cdot c^6 - \tfrac{1}{5040} \\
q^*_{72} &= \tfrac{1}{12}(-\tfrac{1}{28} + b(c^2 \odot Ac^3)) \\
q^*_{73} &= \tfrac{1}{12}(-\tfrac{1}{21} + b(c^3 \odot Ac^2)) + \tfrac{1}{24}(-\tfrac{1}{35} + b(c \odot Ac^4)) \\
q^*_{74} &= \tfrac{1}{8}(-\tfrac{1}{63} + b(Ac^2)^2) + \tfrac{1}{2}(-\tfrac{1}{105} + b(c \odot A(c \odot Ac^2))) \\
q^*_{75} &= \tfrac{1}{24}(-\tfrac{1}{14} + b(c^4 \odot Ac)) + \tfrac{1}{120}(-\tfrac{1}{42} + bAc^5) \\
q^*_{76} &= \begin{pmatrix} \tfrac{1}{2}(-\tfrac{1}{42} + b(c \odot Ac \odot Ac^2)) + \tfrac{1}{6}(-\tfrac{1}{56} + b(Ac \odot Ac^3)) \\ +\tfrac{1}{2}(-\tfrac{1}{56} + b(c^2 \odot A(c \odot Ac))) \\ +\tfrac{1}{2}(-\tfrac{1}{70} + b(c \odot A(c^2 \odot Ac))) + \tfrac{1}{4}(-\tfrac{1}{84} + b(c^2 \odot A^2c^2)) \\ +\tfrac{1}{6}(-\tfrac{1}{140} + b(c \odot A^2c^3)) + \tfrac{1}{4}(-\tfrac{1}{126} + bA(c^2 \odot Ac^2)) + \tfrac{1}{6}(-\tfrac{1}{168} + bA(c \odot Ac^3)) \end{pmatrix} \\
q^*_{77} &= \begin{matrix} \tfrac{1}{4}(-\tfrac{1}{28} + b(c^2 \odot (Ac)^2)) + \tfrac{1}{26}(-\tfrac{1}{42} + b(c^3 \odot A^2c)) \\ +\tfrac{1}{2}(-\tfrac{1}{84} + bA(c^3 \odot Ac)) + \tfrac{1}{24}(-\tfrac{1}{210} + bA^2c^4) \end{matrix} \\
q^*_{78} &= \begin{pmatrix} -\tfrac{17}{840} + b(Ac \odot A(c \odot Ac)) + \tfrac{1}{2}(-\tfrac{1}{140} + b(c \odot A(Ac)^2)) + b(c \odot A(c \odot A^2c)) \\ +\tfrac{1}{2}(-\tfrac{1}{126} + b(Ac^2 \odot A^2c)) + \tfrac{1}{2}(-\tfrac{1}{168} + b(Ac \odot A^2c^2)) \\ +b(c \odot A^2(c \odot Ac)) + \tfrac{1}{2}(-\tfrac{1}{420} + b(c \odot A^3c^2)) + \tfrac{1}{2}(-\tfrac{1}{252} + bA(Ac \odot Ac^2)) \\ +bA(c \odot A(c \odot Ac)) + \tfrac{1}{2}(-(1/504) + bA(c \odot A^2c^2)) \\ +\tfrac{1}{2}(-\tfrac{1}{630} + bA^2(c \odot Ac^2)) \end{pmatrix} \\
q^*_{79} &= \begin{pmatrix} \tfrac{1}{6}(-\tfrac{1}{56} + b(Ac)^3) - \tfrac{1}{84} + b(c \odot Ac \odot A^2c) + \tfrac{1}{2}(-\tfrac{1}{168} + b(c^2 \odot A^3c)) \\ +\tfrac{1}{2}(-\tfrac{1}{168} + bA(c \odot (Ac)^2)) + \tfrac{1}{2}(-\tfrac{1}{252} + bA(c^2 \odot A^2c)) \\ +\tfrac{1}{2}(-\tfrac{1}{420} + bA^2(c^2 \odot Ac)) + \tfrac{1}{6}(-\tfrac{1}{840} + bA^3c^3) \end{pmatrix} \\
q^*_{710} &= \begin{pmatrix} -\tfrac{43}{5040} + \tfrac{1}{2}(-\tfrac{1}{252} + b(A^2c)^2) + b(Ac \odot A^3c) + b(c \odot A^4c) + bA(Ac \odot A^2c) \\ +bA(c \odot A^3c) + \tfrac{1}{2}(-\tfrac{1}{840} + bA^2(Ac)^2) + bA^2(c \odot A^2c) \\ +bA^3(c \odot Ac) + \tfrac{1}{2}(-\tfrac{1}{2520} + bA^4c^2) \end{pmatrix} \\
q^*_{711} &= -\tfrac{1}{5040} + bA^5c
\end{aligned}
$$

The procedure is somehow automated by the Mathematica [29] listing given in Table 5. This approach to addressing RK and related issues has proven fruitful for our research group over many years [10]. There, we may insert some $\Phi(\tau)$ and get the associated integer partition. In consequence we may group and add the various original (i.e. for conventional RK) order conditions. For example,

as a paradigm, we observe that for $\Phi(\tau) = b \cdot \left(c \odot \left(A^2 \cdot c^2\right)\right)$ we worked with above, we get

```
In[1]:= RKS[b . (c*a . a . c^2)]
Out[1]= {1, 2, 2}
```

as expected by two observations of f'' and one for f' in $F(\tau)$. Notice that Mathematica handles Hadamard product by using the symbol " $*$ ". More indicative runs are shown below.

```
In[2]:= RKS[b . (a . c^3)^2]
RKS[b . a . (c*a . c*a . c^2)]
RKS[b . (a . c^2)^2]
RKS[b . a . (c*(a . c)^2)]
Out[3]= {2, 3, 3}
Out[4]= {1, 1, 2, 3}
Out[5]= {2, 2, 2}
Out[6]= {1, 1, 1, 3}
```

Table 5. Mathematica package relating any $\Phi(\tau)$ with an integer partition.

```
(*-------------------------------------------------------------------------*)
(*-- RKS associates some order condition of n-th order with an
     integer partition of number (n-1).                              ---*)

ClearAll[RKS, parseTree];

parseTree[expr_] :=
  Module[{count, degrees, args, i, childCount, childDegrees, k, base},
    Which[
      (* Leaf node: c or c^k *)
      expr === c, {1, {}},
      Head[expr] === Power && expr[[1]] === c, {expr[[2]], {}},

      (* Product of terms *)
      Head[expr] === Times,
        count = 0; degrees = {};
        args = List @@ expr;
        Do[
          {childCount, childDegrees} = parseTree[args[[i]]];
          count += childCount;
          degrees = Join[degrees, childDegrees],
          {i, Length[args]}
        ];
        {count, degrees},

      (* General power: expr^k (NOT only c^k).
                                        Treat as k-fold product of expr. *)
```

```
      Head[expr] === Power,
        base = expr[[1]]; k = expr[[2]];
        {childCount, childDegrees} = parseTree[base];
        {k*childCount, Flatten@Table[childDegrees, {k}]},

      (* Dot product: a . (subtree) *)
      Head[expr] === Dot && expr[[1]] === a,
        args = List @@ expr;
        If[Length[args] == 2,
          {childCount, childDegrees} = parseTree[args[[2]]],
          {childCount, childDegrees} = parseTree[Dot @@ Rest[args]]
        ];
        {1, Join[{childCount}, childDegrees]},

      (* Default *)
      True, {0, {}}
    ]
  ];

RKS[expr_] :=
  Module[{restExpr, rootCount, degrees},
    If[Head[expr] === Dot && expr[[1]] === b,
      restExpr = If[Length[expr] == 2, expr[[2]], Dot @@ Rest[expr]];
      {rootCount, degrees} = parseTree[restExpr];
      Sort[Join[{rootCount}, degrees]],
      {}
    ]
  ];
(*-------------------------------------------------------------------------*)
```

3. Derivation of a new pair of orders 7(5)

Here we are interested for pairs using effectively eight stages per step. A technique called first stage as last (FSAL) [11] is in common use with these methods. This means that $s = 9$ but the ninth stage is reused as first stage in the next step. We have then

$$a_{9j} = b_j,\ j = 1, 2, \cdots, 8.$$

Thus, we arrive at a cost of eight stages per step. The parameters involved after (1.4) are 28. Namely,

$$c_4, c_3, c_2, c_5, c_5, c_7, c_6, c_8, a_{32}, a_{42}, a_{53}, a_{43}, a_{54}, a_{52}, a_{62}, a_{63},$$
$$a_{65}, a_{64}, a_{63}, a_{72}, a_{74}, a_{73}, a_{75}, a_{76}, a_{82}, a_{84}, a_{83}, a_{87}, a_{86}, a_{85}$$

and

$$b_6, b_7, b_8, b_3, b_4, b_5, b_2, b_1, \widehat{b_4}, \widehat{b_5}, \widehat{b_6}, \widehat{b_2}, \widehat{b_1}, \widehat{b_3}.$$

For the remaining coefficients, we impose the assumptions $c_9 = 1$, $A \cdot e = c$, and $a_{j1} = c_j - \sum_{k=1}^{j-1} a_{jk}$ for $j = 2, 3, \ldots, 8$, while we also fix $\widehat{b_9} = \frac{1}{20} \neq b_9 = 0$.

These settings enable the construction of an embedded pair of orders 7(5), relying on 46 free parameters. This is feasible because the seventh-order method requires 30 order conditions, and the embedded fifth-order method introduces an additional 12 conditions, totaling 42 constraints. In contrast, the derivation of a conventional RK 7(5) pair entails solving 85 + 17 = 102 conditions and always demands more than eight stages.

In our case, we need to satisfy 42 equations in 46 unknowns. Following common RK methodology, this can be approached by introducing simplifying assumptions. For instance, one might consider setting $A \cdot c = \frac{1}{2}c^2$ and $b_2 = 0$, which would reduce the fifth-order condition $q_{43} = b \cdot (c \odot (A \cdot c)) - \frac{1}{8} = \frac{1}{2} b \cdot c^3 - \frac{1}{8}$ to $q_{41} = \frac{1}{6} b \cdot c^3 - \frac{1}{24}$. However, such simplifications are ineffective in the scalar autonomous case, as q_{43} and q_{42} are already combined into a single condition.

To handle the system numerically, we define a fitness function of the form

$$f = \sqrt{q_{11}^{*2} + q_{21}^{*2} + \cdots + q_{7,10}^{*2} + q_{7,11}^{*2} + \widetilde{q}_{11}^{*2} + \widetilde{q}_{21}^{*2} + \cdots + \widetilde{q}_{55}^{*2}},$$

where, for example, $\widetilde{q}_{11}^{*} = \widetilde{b} \cdot e - 1$ and $\widetilde{q}_{21}^{*} = \widetilde{b} \cdot c - \frac{1}{2}$.

In recent work, we have observed promising performance using evolutionary optimization techniques for such tasks [14, 25], with differential evolution in particular proving effective [24]. Given the stochastic nature of these methods, success is not guaranteed from a single execution. Therefore, we conducted several hundred independent runs, periodically adjusting hyperparameters such as population size or crossover rate. Among the candidate methods generated, we selected and report here the one referred to as RKT7(5)sa, whose coefficients—accurate to 20 decimal digits—are given in Table 6.

4. System identification of the logistic growth model via neural ordinary differential equations

The newly developed 7(5) pair is compared with conventional RK pairs that also employ eight stages per step. In particular, we consider the Verner 6(5) pair, denoted V6(5) [28], which is implemented in the Mathematica routine `NDSolve` [18]. We also include the well-known Prince–Dormand pair of the same orders [21].

To carry out this comparison, the parameter estimation problem for the logistic growth model is formulated within the neural ordinary differential equation framework. In this setting, the parameters of the ODE are treated as learnable weights in a neural network and are identified through gradient-based optimization.

4.1. Problem formulation

The logistic growth model is governed by the differential equation

$$\frac{dy}{dx} = ry\left(1 - \frac{y}{K}\right), \quad y(0) = y_0$$

where r (growth rate) and K (carrying capacity) are unknown parameters that must be estimated from observed data $\{(x_i, y_i)\}_{i=0}^{N}$. In this study we select $K = 20$, $r = \frac{2}{3}$, $y_0 = 4$, and the sampling points $x_0 = 0, x_1 = 0.5, x_2 = 1, \cdots, x_{20} = 10$.

4.2. Neural ODE framework

The estimation procedure is expressed as the following optimization problem:

$$\min_{r,K} \mathcal{L}(r, K) = \sum_{i=1}^{N} \left(y(x_i; r, K) - y_i^{\text{obs}} \right)^2$$

where $y(x; r, K)$ denotes the numerical solution of the ODE computed at high accuracy.

Table 6. Coefficients of the new RK pair of orders 7(5), truncated to 20 digits of accuracy.

c	$a_{i,j}$							
0								
$\frac{3126950044}{6340454809}$	$\frac{3126950044}{6340454809}$							
$\frac{2925260923}{7384058871}$	$\frac{1409308203}{5844379817}$	$\frac{1241711759}{8010015880}$						
$\frac{12903252065}{17942357093}$	$-\frac{1020731809}{11540432617}$	$-\frac{4714153547}{9701249380}$	$\frac{5226224729}{4040277075}$					
$\frac{217505207}{9788803286}$	$\frac{123622898}{10118617655}$	$\frac{50100379}{10510834428}$	$-\frac{61933580}{13280927793}$	$\frac{56692585}{5726971281}$				
$\frac{10349224237}{13351775809}$	$-\frac{1851880001}{10493576314}$	$\frac{2267005388}{7926949457}$	$\frac{94195913}{25353633348}$	$\frac{1038599794}{5134665429}$	$\frac{473890712}{1031043033}$			
$\frac{1}{2}$	$\frac{2501503843}{1281925320}$	$\frac{2165041787}{3817512528}$	$-\frac{1250479490}{2575286531}$	$\frac{1092854340}{8011015349}$	$-\frac{12634115317}{7812182078}$	$-\frac{540245062}{10366164209}$		
1	$\frac{7298749599}{7525085150}$	$-\frac{1823729089}{6340700813}$	$\frac{21031845089}{11264853748}$	$-\frac{2076731971}{4608168119}$	$-\frac{9407575974}{8606408537}$	$\frac{6275127720}{6788927527}$	$-\frac{10172635702}{10939526927}$	
1	$-\frac{4809048936}{16296720419}$	$\frac{526440466}{12577696081}$	$\frac{4573852771}{9115376312}$	$\frac{814665160}{16177999481}$	$\frac{11660260934}{24533487373}$	$\frac{1929604159}{6071758170}$	$-\frac{2084680067}{13121129790}$	$\frac{537337537}{8030919015}$ 0
7^{th}-order b_i	$-\frac{4809048936}{16296720419}$	$\frac{526440466}{12577696081}$	$\frac{4573852771}{9115376312}$	$\frac{814665160}{16177999481}$	$\frac{11660260934}{24533487373}$	$\frac{1929604159}{6071758170}$	$-\frac{2084680067}{13121129790}$	$\frac{537337537}{8030919015}$ 0
5^{th}-order $\hat{b}_i$	$-\frac{1331169250}{3777195453}$	$\frac{2015341798}{43365771787}$	$\frac{4379057318}{9619773033}$	$\frac{732162403}{8658787400}$	$\frac{4733837548}{8735354689}$	$\frac{2161647320}{7812116743}$	$-\frac{796511107}{6438314893}$	$\frac{99292633}{4667992240}$ $\frac{1}{20}$

4.3. Numerical integration schemes

Numerical integration of the ODE was performed with the following pairs that effectively use eight-stages per step:

- PD6(5), the non-FSAL Runge–Kutta–Prince–Dormand pair of orders 6(5) [21].
- V6(5), the Runge–Kutta–Verner, FSAL pair of orders 6(5) [28].
- RKT7(5)sa, the pair proposed here.

All pairs were executed with tolerances $\varepsilon = 10^{-7}$ & 10^{-9}. The observed computation times and the corresponding accuracies in parameter estimation were recorded.

4.4. Gradient computation

Gradients of the loss function with respect to the parameters were approximated using central finite differences:

$$\frac{\partial \mathcal{L}}{\partial r} \approx \frac{\mathcal{L}(r+\delta, K) - \mathcal{L}(r-\delta, K)}{2\delta}$$
$$\frac{\partial \mathcal{L}}{\partial K} \approx \frac{\mathcal{L}(r, K+\delta) - \mathcal{L}(r, K-\delta)}{2\delta}$$

with $\delta = 0.000005$, chosen to balance numerical stability and precision.

4.5. Optimization strategy

Parameter updates followed a momentum-based gradient descent scheme:

$$v_r^{(k+1)} = \gamma v_r^{(k)} + \eta \frac{\partial \mathcal{L}}{\partial r}$$
$$v_K^{(k+1)} = \gamma v_K^{(k)} + \eta \frac{\partial \mathcal{L}}{\partial K}$$
$$r^{(k+1)} = r^{(k)} - v_r^{(k+1)}$$
$$K^{(k+1)} = K^{(k)} - v_K^{(k+1)}$$

where $\gamma = 0.9$ is the momentum coefficient and $\eta = 0.0001$ is the learning rate.

4.6. Implementation details

- *Initialization*: Parameters initialized away from the true values ($r^{0)} = 0.5$, $K^{(0)} = 30$).
- *Constraints*: Parameters restricted to physically meaningful ranges ($0.1 \le r \le 2$, $10 \le K \le 100$).
- *Termination*: Optimization was carried out for 1400 iterations.
- *Monitoring*: Loss values and parameter trajectories were recorded every 200 iterations.

4.7. Convergence metrics and results

The accuracy of the estimation was assessed using the relative errors

$$\epsilon_r = \left| \frac{r_{\text{est}} - r}{r} \right|, \quad \epsilon_K = \left| \frac{K_{\text{est}} - K}{K} \right|$$

where r_{est} and K_{est} denote the estimates obtained via the neural ODE procedure. The method was successful in recovering the model parameters, while retaining interpretability and leveraging efficient gradient-based optimization. Since computational times and numerical accuracies may give rise to ambiguity regarding which method is genuinely superior, we introduce an efficiency metric, defined analogously to the discussion in [13, 23]

$$\text{eff}_r = \text{time} \times \epsilon_r^{1/6}, \qquad \text{eff}_K = \text{time} \times \epsilon_K^{1/6}.$$

Accordingly, lower values of eff_r and eff_K indicate higher computational efficiency. We remark that the exponent $1/6$ is determined by the order of the embedded lower-order formula and, consequently, by the order of the local error estimator employed in the adaptive scheme.

All computations were performed using Mathematica version 13.3 on a system equipped with an AMD Ryzen 3900X processor operating at 3.79 GHz. The results of all pairs are presented in Tables 7–9.

Table 7. Results obtained with PD6(5).

ϵ	Time	ϵ_r	ϵ_K	Efficiencies
10^{-7}	5.03sec	6.0×10^{-6}	1.0×10^{-6}	0.68, 0.50
10^{-9}	5.79sec	9.9×10^{-7}	1.7×10^{-7}	0.58, 0.43

Table 8. Results obtained with V6(5).

ϵ	Time	ϵ_r	ϵ_K	Efficiencies
10^{-7}	5.12sec	3.6×10^{-6}	7.4×10^{-7}	0.63, 0.49
10^{-9}	5.94sec	1.0×10^{-7}	3.6×10^{-8}	0.40, 0.34

Table 9. Results obtained with RKT7(5)sa.

ϵ	Time	ϵ_r	ϵ_K	Efficiencies
10^{-7}	5.19sec	4.5×10^{-7}	3.2×10^{-7}	0.45, 0.43
10^{-9}	6.08sec	4.5×10^{-8}	1.0×10^{-8}	0.36, 0.28

A direct comparison of the four schemes shows that the RKT7(5)sa method consistently outperforms all other pairs in efficiency. A Mathematica script with the coefficients of RKT7(5)sa that reproduces the last line of Table 9 can be found in

`http://users.uoa.gr/~tsitourasc/rkt75sa.m`

with the understanding that times may differ due to different machines used. That same script can be used for checking the other pairs. Except for the coefficients the only other change is setting `"DifferenceOrder"->6` for V6(5) and PD6(5). In any case the times ought to differ analogously.

5. Parameter estimation for a saturating cubic model exhibiting pitchfork bifurcation dynamics

We consider the scalar autonomous differential equation

$$\frac{dy}{dx} = -\alpha y^3 + \beta y$$

with initial condition $y(0) = \frac{1}{2}$, where $\alpha = \frac{1}{10}$ and $\beta = \frac{1}{2}$. This equation serves as a prototypical model for systems exhibiting pitchfork bifurcation behavior. It arises in various contexts, including symmetry-breaking phenomena in physics, density-dependent population growth in biology, and nonlinear damping in engineering. The linear term βy promotes initial growth, while the cubic term $-\alpha y^3$ introduces saturation, stabilizing the system at finite amplitude. We generate synthetic data using

the exact solution and apply the pairs under consideration to assess their effectiveness in recovering the parameters α and β through gradient-based optimization.

We worked analogously with the previous example using the same sampling points in the interval $[0, 10]$. The whole setting remains the same. We choose as initial guesses $\alpha^{(0)} = \frac{1}{8}$, $\beta^{(0)} = \frac{2}{3}$. The process is terminated after 400 iterations. The parameters restricted to ranges ($0.01 \leq \alpha \leq 1$, $\frac{1}{10} \leq \beta \leq 2$). Finally we recorded the cost (in seconds) and the analogous ϵ_α, ϵ_β for this problem in Tables 10–12.

Table 10. Results obtained with PD6(5).

ϵ	Time	ϵ_α	ϵ_β	Efficiencies
10^{-7}	1.53sec	7.9×10^{-6}	6.7×10^{-6}	0.22, 0.21
10^{-9}	1.75sec	6.8×10^{-7}	5.8×10^{-7}	0.16, 0.16

Table 11. Results obtained with V6(5).

ϵ	Time	ϵ_α	ϵ_β	Efficiencies
10^{-7}	1.51sec	1.3×10^{-5}	1.0×10^{-5}	0.23, 0.22
10^{-9}	1.80sec	2.1×10^{-7}	1.4×10^{-7}	0.14, 0.13

Table 12. Results obtained with RKT7(5)sa.

ϵ	Time	ϵ_α	ϵ_β	Efficiencies
10^{-7}	1.58sec	3.5×10^{-6}	2.9×10^{-6}	0.19, 0.19
10^{-9}	1.85sec	5.9×10^{-8}	5.8×10^{-8}	0.11, 0.11

The RKT7(5)sa scheme also demonstrated superior performance in the cubic nonlinear model, achieving higher efficiency compared to conventional Runge–Kutta pairs.

6. Conventional numerical tests

We also made some tests over a couple of IVPs under the traditional numerical analysis framework. The problems chosen are the following:

	Problem	Solution
1	$y' = -\frac{y^3}{2}$, $y(0) = 1$	$y(x) = \frac{1}{\sqrt{1+x}}$
2	$y' = e^{-y}$, $y(0) = 1$	$y(x) = \log(x + e)$

Both problems were run in the interval $t \in [0, 20]$ for all three pairs and for tolerances $\varepsilon = 10^{-7}, 10^{-8}, 10^{-9}, 10^{-10}, 10^{-11}$. For each run, we recorded the stages spend versus the global error

achieved over the mesh points. The resulted graphs are shown as efficiency curves in Figures 1 and 2. Gaining about half a digit when applying methods that use the same amount of stages per step is a rather interesting outcome.

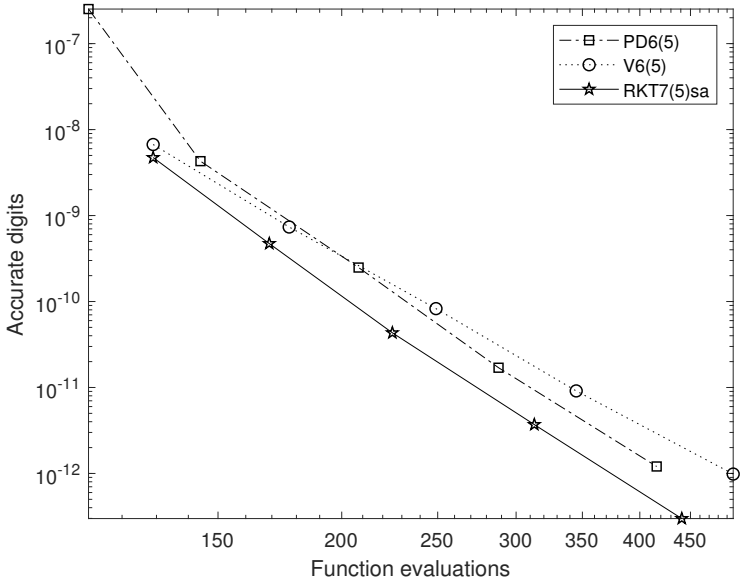


Figure 1. Efficiency curves on first problem.

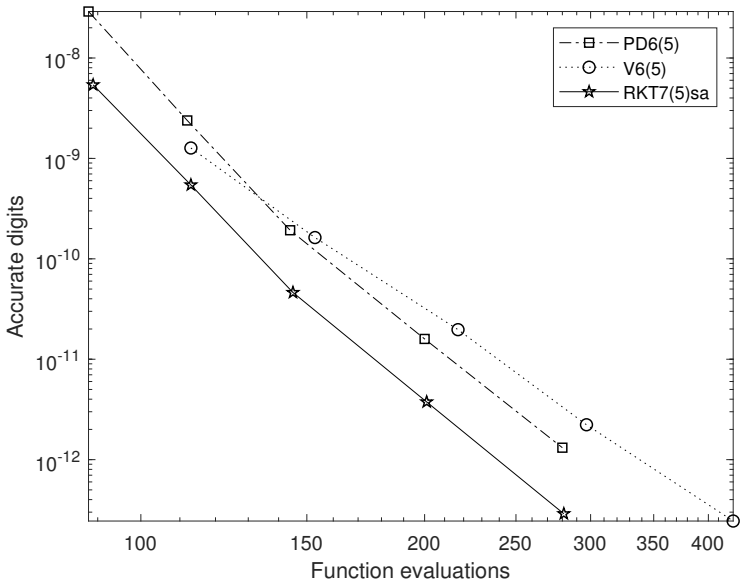


Figure 2. Efficiency curves on second problem.

7. Conclusions

In this work, we have presented and analyzed a novel embedded Runge–Kutta (RK) pair of orders 7(5), specifically tailored for scalar autonomous ordinary differential equations. By leveraging the structural simplifications inherent to the scalar autonomous case, we significantly reduced the number of required order conditions compared to the general case. This allowed us to construct an efficient pair requiring only eight function evaluations per step and attaining higher algebraic order.

To derive the method, we organized the order conditions using integer partition techniques and employed high-precision differential evolution to optimize the coefficients. The resulting method, denoted as RKT7(5)sa, achieves a favorable balance between accuracy and computational efficiency, outperforming other classical methods in our experiments.

The practical effectiveness of the proposed method was demonstrated through system identification tasks based on both the logistic growth model and a cubic nonlinear model within the neural ODE framework. In both cases, the results confirmed improved parameter estimation accuracy and reduced computational cost across a range of tolerance settings. Conventional numerical tests also demonstrated the efficiency gains for the new pair.

Author contributions

Ibraheem Alolyan: Supervision, validation, formal analysis, Rreview, funding acquisition; Theodore E. Simos: Project administration, methodology, formal analysis, visualization, writing–original draft, writing–review & editing; Charalampos Tsitouras: Conceptualization, investigation, methodology, software, formal analysis, visualization, writing–original draft, writing & editing. All authors have read and approved the final version of the manuscript for publication.

Use of Generative-AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This work was supported and funded by the Ongoing Research Funding Program (ORF-2025-1246) King Saud University, Riyadh, Saudi Arabia.

Conflict of interest

Prof. Charalampos Tsitouras is an editorial board member for AIMS Mathematics and was not involved in the editorial review and/or the decision to publish this article.

This work does not have any conflicts of interest.

Data availability statement

A Mathematica script with the coefficients of RKT7(5)sa and an indicative run for the Neural ODEs paradigm from section 4, can be found in `http://users.uoa.gr/~tsitourasc/rkt75sa.m`.

References

1. R. W. Brankin, I. Gladwell, J. R. Dormand, P. J. Prince, W. L. Seward, Algorithm 670: A Runge-Kutta-Nyström code, *ACM T. Math. Software*, **15** (1989), 31–40. https://doi.org/10.1145/62038.69650
2. J. C. Butcher, Implicit Runge-Kutta processes, *Math. Comput.*, **18** (1964), 50–64. https://doi.org/10.1090/S0025-5718-1964-0159424-9
3. J. C. Butcher, On Runge-Kutta Processes of high order, *J. Aust. Math. Soc.*, **4** (1964), 179–194. https://doi.org/10.1017/S1446788700023387
4. J. C. Butcher, *The numerical analysis of ordinary differential equations: Runge-Kutta and general linear methods*, Wiley: Chichester, UK, 1987.
5. M. Caldana, J. S. Hesthaven, Neural ordinary differential equations for model order reduction for stiff systems, *Int. J. Numer. Meth. Eng.*, **126** (2025), e70060. https://doi.org/10.1002/nme.70060
6. T. Q. Chen, Y. Rubanova, J. Bettencourt, D. Duvenaud, Neural ordinary differential equations, *Adv. Neural Inform. Process. Syst.*, 2018.
7. J. R. Dormand, P. J. Prince, A family of Embedded Runge-Kutta formulae, *J. Comput. Appl. Math.*, **6** (1980), 19–26. https://doi.org/10.1016/0771-050X(80)90013-3
8. E. Dupont, A. Doucet, Y. W. Teh, Augmented neural ODEs, *Adv. Neural Inform. Process. Syst.*, 2019.
9. W. H. Enright, J. D. Pryce, Two FORTRAN packages for assessing initial value methods, *ACM T. Math. Software*, **13** (1987), 1–27.
10. I. T. Famelis, C. Tsitouras, Symbolic derivation of order conditions for hybrid Numerov-type methods solving $y'' = f(x, y)$, *J. Comput. Appl. Math.*, **218** (2008), 543–555. https://doi.org/10.1016/j.cam.2007.09.017
11. E. Fehlberg, *Low order classical Runge-Kutta formulas with step-size control and their application to some heat transfer problems*, NASA Technical Report TR R-315, 1969.
12. E. Hairer, S. P. Nørsett, G. Wanner, *Solving ordinary differential equations I*, Springer: Berlin, Germany, 1987. https://doi.org/10.1007/978-3-662-12607-3
13. H. Jerbi, S. B. Aoun, M. Omri, T. E. Simos, C. Tsitouras, A neural network type approach for constructing Runge-Kutta pairs of orders six and five that perform best on problems with oscillatory solutions, *Mathematics*, **10** (2022), 827. https://doi.org/10.3390/math10050827
14. H. Jerbi, S. Maali, S. B. Aoun, A. N. Aledaily, V. Jeyamani, T. E. Simos, et al., On High-Order Runge-Kutta pairs for linear inhomogeneous problems, *Axioms*, **14** (2025), 245. https://doi.org/10.3390/axioms14040245
15. J. D. Lambert, *Numerical methods for ordinary differential systems*, Wiley: Chichester, UK, 1991.
16. Y. Lu, A. Zhong, Q. Li, B. Dong, *Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations*, In: Proc. 35th Int. Conf. Machine Learning, Stockholm, Sweden, PMLR, **80** (2018), 3276–3285.

17. S. Massaroli, M. Poli, D. Park, J. Yamashita, A. Asama, J. Park, Dissipative Hamiltonian neural networks, *Adv. Neural Inform. Process. Syst.*, **33** (2020), 33.

18. Wolfram Research, *NDSolve Mathematica function*, 2025. Available from: `https://reference.wolfram.com/language/tutorial/NDSolveExplicitRungeKutta.html`.

19. G. Papageorgiou, C. Tsitouras, Runge-Kutta pairs for scalar autonomous initial value problems, *Int. J. Comput. Math.*, **80** (2003), 201–209. https://doi.org/10.1080/00207160304669

20. M. Poli, S. Massaroli, A. Yamashita, H. Asama, J. Park, Graph neural ordinary differential equations, *arXiv preprint*, 2019. https://doi.org/10.48550/arXiv.1911.07532

21. P. J. Prince, J. R. Dormand, High order embedded Runge-Kutta formulae, *J. Comput. Appl. Math.*, **7** (1981), 67–75. https://doi.org/10.1016/0771-050X(81)90010-3

22. J. Riordan, *An introduction to combinatorial analysis*, Wiley: New York, NY, USA, 1958.

23. L. F. Shampine, Some practical Runge-Kutta formulas, *Math. Comput.*, **48** (1985), 135–150. https://doi.org/10.1090/S0025-5718-1986-0815836-3

24. R. Storn, K. Price, Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces, *J. Global Optim.*, **11** (1997), 341–359. https://doi.org/10.1023/A:1008202821328

25. T. E. Simos, C. Tsitouras, 6th order Runge-Kutta pairs for scalar autonomous IVP, *Appl. Comput. Math.*, **19** (2020), 412–421.

26. C. Tsitouras, Runge–Kutta pairs of order 5(4) satisfying only the first column simplifying assumption, *Comput. Math. Appl.*, **62** (2011), 770–775. https://doi.org/10.1016/j.camwa.2011.06.002

27. C. Tsitouras, S. N. Papakostas, Cheap error estimation for Runge-Kutta methods, *SIAM J. Sci. Comput.*, **20** (1999), 2067–2088. https://doi.org/10.1137/S1064827596302230

28. J. H. Verner, Numerically optimal Runge-Kutta pairs with interpolants, *Numer. Algorithms*, **53** (2010), 383–396. https://doi.org/10.1007/s11075-009-9290-3

29. Wolfram Research, Inc., *Mathematica, Version 13.3*, Champaign, IL, 2023.

AIMS Press

© 2026 the Author(s), licensee AIMS Press. This is an open access article distributed under the terms of the Creative Commons Attribution License (https://creativecommons.org/licenses/by/4.0)