A new matrix equation expression for the solution of non-autonomous linear systems of ODEs

Stefano Pozza$^{1,*}$ and Niel Van Buggenhout$^1$

1 Faculty of Mathematics and Physics, Charles University, Sokolovská 83, Prague, 186 75, Czech Republic

The solution of systems of non-autonomous linear ordinary differential equations is crucial in a variety of applications, such as nuclear magnetic resonance spectroscopy. A new method with spectral accuracy has been recently introduced in the scalar case. The method is based on a product that generalizes the convolution. In this work, we show that it is possible to extend the method to solve systems of non-autonomous linear ordinary differential equations (ODEs). In this new approach, the ODE solution can be expressed through a linear system that can be equivalently rewritten as a matrix equation. Numerical examples illustrate the method’s efficacy and the low-rank property of the matrix equation solution.

1 Introduction

Systems of non-autonomous linear ordinary differential equations (ODEs) appear in a variety of applications, and their numerical computation is often challenging, particularly for large-to-huge size systems. For instance, in nuclear magnetic resonance spectroscopy (NMR) [1], the system solution describes the dynamics of the nuclear spins of a sample in a time-varying magnetic field. The size of such systems is $2^k \times 2^k$ for a sample with $k$ spins and is usually sparse. In [2], we proposed a new method with spectral accuracy for solving scalar non-autonomous ordinary differential equations. In the present work, we extend this method to the case of systems of non-autonomous ODEs.

Consider a matrix $\tilde{A}(t) \in \mathbb{C}^{N \times N}$ composed of elements from $C^\infty(I)$, i.e., the set of functions infinitely differentiable (smooth) over $I$, with $I$ a closed and bounded interval in $\mathbb{R}$. The system

$$\frac{d}{dt} U_s(t) = \tilde{A}(t) U_s(t), \quad U_s(s) = I_N, \quad \text{for } t \geq s, \quad t, s \in I, \quad (1)$$

has a unique solution $U_s(t) \in \mathbb{C}^N \times N$; $I_N$ stands for the $N \times N$ identity matrix. Note that the condition $U_s(s) = I_N$ is not restrictive, since, given a matrix $B \in \mathbb{C}^{N \times N}$, the matrix-valued function $V_s(t) := U_s(t)B$ solves the ODE

$$\frac{d}{dt} V_s(t) = \tilde{A}(t) V_s(t), \quad V_s(s) = B \quad \text{for } t \geq s, \quad t, s \in I.$$

At the heart of the new method for solving (1) is a non-commutative convolution-like product, denoted by $\ast$, defined between certain distributions [3]. Thanks to this product, the solution of (1) can be expressed through the $\ast$-product inverse and its formulation as a sequence of integrals and differential equations; see [4–8]. In [2], we illustrated that, by discretizing the $\ast$-product with orthogonal functions, the solution of a scalar ODE is accessible by solving a linear system. In this work, we extend the results in [2], showing that, following the same principles, we can solve (1) through a linear system. Moreover, we show that the linear system solution can be expressed as the solution of a matrix equation with a rank one right-hand side. Numerical experiments illustrate that the solution of the matrix equation can also be low-rank.

In Section 2, we recall the $\ast$-product definition and the related expression for the solution of an ODE. The expression is then discretized and approximated by the solution of a linear system. Section 3 shows how to transform the linear system into a matrix equation, and Section 4 concludes the paper.

2 Solution of an ODE by the $\ast$-product

We use the Heaviside theta function

$$\Theta(t - s) = \begin{cases} \begin{array}{ll} 1, & t \geq s \\ 0, & t < s \end{array} \end{cases},$$

to rewrite (1) in the following equivalent form

$$\frac{d}{dt} U(t, s) = \tilde{A}(t) \Theta(t - s) U(t, s), \quad U(s, s) = I_N, \quad \text{for } t, s \in I. \quad (2)$$

* Corresponding author: e-mail pozza@karlin.mff.cuni.cz,
Note that $\Theta(t-s)$ endows the condition $t \geq s$ in equation (2) and that $U(t,s)$ is the bivariate function expressing the solutions of (1) for every initial time $s \in \mathcal{I}$, with $U(t,s) = 0$ for $t < s$. From now on, we will denote with a tilde all the bivariate functions that are infinitely differentiable in both $t$ and $s$ over $\mathcal{I}$, i.e., $\tilde{f} \in C^\infty(\mathcal{I} \times \mathcal{I})$. Moreover, we define the following class of functions

$$
C^\infty_G(\mathcal{I}) := \left\{ f : f(t,s) = \tilde{f}(t,s)\Theta(t-s), \quad \tilde{f} \in C^\infty(\mathcal{I} \times \mathcal{I}) \right\}.
$$

Consider now the $N \times N$ matrices $A_1(t,s), A_2(t,s) \in (C^\infty_G(\mathcal{I}))^{N \times N}$, i.e., matrices composed of elements from $C^\infty_G(\mathcal{I})$. Then, the $\ast$-product is defined as

$$
(A_2 \ast A_1)(t,s) := \int_{\mathcal{I}} A_2(t,\tau)A_1(\tau,s) \, d\tau.
$$

(3)

The $\ast$-product can be extended to a larger class of matrices composed of elements from the class $D(\mathcal{I}) \supset C^\infty_G(\mathcal{I})$, that is, the class of the superpositions of $\Theta(t-s)$, Dirac delta distribution $\delta(t-s)$, and Dirac delta derivatives described in [6]. In such a class, $\delta(t-s)I_N$ is the $\ast$-product identity, i.e., $A(t,s) \ast \delta(t-s)I_N = \delta(t-s)I_N \ast A(t,s) = A(t,s)$. Moreover, in the larger class $D(\mathcal{I})$, the $\ast$-product admits inverses under certain conditions [6], i.e., for certain $f(t,s) \in C^\infty_G$ there exists $f(t,s)^{-\ast}$ such that $f(t,s) \ast f(t,s)^{-\ast} = f(t,s)^{-\ast} \ast f(t,s) = \delta(t-s)$.

Following [4], the solution of (2) can be expressed as

$$
U(t,s) = \Theta(t-s) \ast R_s(A)(t,s),
$$

(4)

where $A(t,s) = \tilde{A}(t) \Theta(t-s)$ and $R_s(A)$ is the $\ast$-resolvent of $A$, i.e.,

$$
R_s(A)(t,s) = \delta(t-s)I_N + \sum_{k=1}^{\infty} A(t,s)^{k\ast},
$$

with $A(t,s)^{k\ast} = A \ast \cdots \ast A$, the $k$th power of the $\ast$-product. Note that the series $\sum_{k=1}^{\infty} A(t,s)^{k\ast}$ converges for every $A \in (C^\infty_G(\mathcal{I}))^{N \times N}$. Expression (4) hides an infinite series of nested integrals. However, as shown in [2], it is possible to approximate the $\ast$-product by the usual matrix-product in the scalar case. This approximation allows us to compute (4) more simply and cheaply. We recall its basics below.

Without loss of generality, we set $\mathcal{I} = [0,1]$. Moreover, we consider the family of orthonormal shifted Legendre polynomials $\{p_k\}_k$. Then, any $f(t,s) \in C^\infty_G(\mathcal{I})$ can be expanded into the following series (e.g., [9])

$$
f(t,s) = \sum_{k=0}^{\infty} \sum_{\ell=0}^{\infty} f_{k,\ell} p_k(t)p_\ell(s), \quad t \neq s, \quad t, s \in \mathcal{I}, \quad f_{k,\ell} = \int_{\mathcal{I}} \int_{\mathcal{I}} f(\tau,\rho)p_k(\tau)p_\ell(\rho) \, d\rho \, d\tau.
$$

(5)

By defining the coefficient matrix $F_M$ and the vector $\phi_M(t)$ as

$$
F_M := \begin{bmatrix}
  f_{0,0} & f_{0,1} & \ldots & f_{0,M-1} \\
  f_{1,0} & f_{1,1} & \ldots & f_{1,M-1} \\
  \vdots & \vdots & \ddots & \vdots \\
  f_{M-1,0} & f_{M-1,1} & \ldots & f_{M-1,M-1}
\end{bmatrix}, \quad \phi_M(t) := \begin{bmatrix}
  p_0(t) \\
  p_1(t) \\
  \vdots \\
  p_{M-1}(t)
\end{bmatrix},
$$

(6)

the truncated expansion series can be written in the matrix form:

$$
f_M(t,s) := \sum_{k=0}^{M-1} \sum_{\ell=0}^{M-1} f_{k,\ell} p_k(t)p_\ell(s) = \phi_M(t)^T F_M \phi_M(s).
$$

Let us consider the functions $f, g, h \in C^\infty_G(\mathcal{I})$ so that $h = f \ast g$, and the related coefficient matrices (6), respectively, $F_M, G_M, H_M$. Following [6], $H_M$ can be approximated by the expression

$$
H_M \approx \hat{H}_m := F_M G_M.
$$

(7)

Therefore, there is a connection between the $\ast$-algebra over $D(\mathcal{I})$ and the usual matrix algebra. The elements and operations which form the $\ast$-algebra and the related elements and operations forming the usual matrix algebra are given in Table 1 (in the first two columns for the scalar case); for more details, we refer to [6].
The approximation in the scalar case can be easily extended to the matrix one. Indeed, if \( A(t, s) = [a_{ij}(t, s)]_{i,j=1}^{N} \) is an \( N \times N \) matrix with elements \( a_{ij}(t) \in C_{\infty}^{\Theta}(I) \), then for each \( a_{ij} \), we can compute the related coefficient matrices \( F_{M}^{(l,j)} \) (6) obtaining the block matrix

\[
A_{M} = \begin{bmatrix} F_{M}^{(1,1)} & \cdots & F_{M}^{(1,N)} \\ \vdots & \ddots & \vdots \\ F_{M}^{(N,1)} & \cdots & F_{M}^{(N,N)} \end{bmatrix} \in C^{MN\times MN}. \tag{8}
\]

Let us define the \( N \times N \) matrices \( A(t, s), B(t, s), C(t, s) \in (C_{\infty}^{\Theta}(I))^{N\times N} \) so that \( C(t, s) = A(t, s) * B(t, s) \) and let their coefficient matrices (8) be, respectively, \( A_{M}, B_{M}, C_{M} \). Then, analogously to the scalar case, \( C_{M} \) is approximated by

\[
C_{M} \approx \hat{C}_{M} := A_{M}B_{M}. \tag{11}
\]

As a consequence, also in the matrix case, the *-algebra can be approximated by the usual matrix algebra, as summarized in the last two columns of Table 1.

| *-operation/elements | matrix operation/elements | *-operation/elements | matrix operation/elements |
|----------------------|---------------------------|----------------------|---------------------------|
| \( q = f * g \)      | \( Q_{M} = F_{M}G_{M} \) | \( C = A * B \)     | \( C_{M} = A_{M}B_{M} \) |
| \( f + g \)          | \( F_{M} + G_{M} \)       | \( A + B \)         | \( A_{M} + B_{M} \)       |
| \( 1_{*} := \delta(t - s) \) | \( I_{M} \), identity matrix | \( 1_{*} := \delta(t - s)I_{N} \) | \( I_{MN}, \) identity matrix |
| \( R_{*}(f) := (1_{*} - f)^{-1} \) | \( R(F_{M}) := (I_{M} - F_{M})^{-1} \) | \( A^{-1} \)         | \( A_{M}^{-1} \)         |

\( R_{*}(f) := (1_{*} - A)^{-1} \)

\( R(A_{M}) := (I_{MN} - A_{M})^{-1} \)

Table 1: The *-algebra operations and the corresponding matrix algebra operation after discretization, scalar case (first two columns), matrix case (last two columns).

The matrix-valued function \( U(t, s) \) in (2) is composed of elements from \( C_{\infty}^{\Theta}(I) \). Therefore, we can define the related coefficient matrix \( U_{M} \) as in (8). Then, expression (4) can be approximated by

\[
U_{M} \approx (I_{N} \otimes T_{M})(I_{MN} - A_{M})^{-1},
\]

where \( \otimes \) is the Kronecker product, \( T_{M} \) is the coefficient matrix of \( \Theta(t - s) \), and \( A_{M} \) is the coefficient matrix of \( \tilde{A}(t)\Theta(t - s) \), with \( \tilde{A}(t) \) from (2). Moreover, we can approximate the solution of (2) for \( s = 0 \) by the formula:

\[
U(t, 0) \approx \phi_{M}(t)^{T}U_{M}\phi_{M}(0) = (I_{N} \otimes \phi_{M}(t)^{T})(I_{N} \otimes T_{M})(I_{MN} - A_{M})^{-1}(I_{N} \otimes \phi_{M}(0)) = (I_{N} \otimes \phi_{M}(t)^{T}T_{M})(I_{MN} - A_{M})^{-1}(I_{N} \otimes \phi_{M}(0)).
\]

Note that, as explained in [2], the approximation converges quickly enough to the solution only when \( s \) is the left endpoint of the interval \( I \), i.e., \( s = 0 \).

In practical situations, the initial time \( s \) of the evolution is fixed (\( s = 0 \)), and the initial condition is given as a vector \( v \in \mathbb{C}^{N} \). Then, we get the simpler problem,

\[
\frac{d}{dt}u(t) = \tilde{A}(t)\Theta(t - s)u(t), \quad u(0) = v, \quad \text{for } t, s \in I,
\]

where the solution \( u(t) \) is an \( N \)-size vector. Thus, \( u(t) \) is approximated by:

\[
u(t) \approx (I_{N} \otimes \phi_{M}(t)^{T}T_{M})(I_{MN} - A_{M})^{-1}(I_{N} \otimes \phi_{M}(0))v \\
\approx (I_{N} \otimes \phi_{M}(t)^{T}T_{M})(I_{MN} - A_{M})^{-1}(v \otimes \phi_{M}(0)).
\]

Then, solving the linear system

\[
(I_{MN} - A_{M})x = v \otimes \phi_{M}(0), \tag{10}
\]

one can approximate the solution of (9) in terms of its expansion coefficients \( u_{M} := (I_{N} \otimes T_{M})x \), that is,

\[
u(t) \approx \hat{u}(t) := (I_{N} \otimes \phi_{M}(t)^{T})u_{M}. \tag{11}
\]
2.1 Numerical examples

Given a random vector \( v \) with elements in \([0, 1]\), we aim to compute the bilinear form \( v^T u(t) \) obtained by solving the following ODE system

\[
\frac{d}{dt} u(t) = -2\sqrt{-1}\pi \hat{H}(t)u(t), \quad u(0) = v, \quad \text{for } t \in [0, T].
\]

This system of ODEs comes from Experiment 2 (Strong coupling) in [10], and \( v^T u(t) \) represents an NMR experiment with a magic angle spinning (MAS) for \( k \) spins; see, e.g., [1]. The so-called Hamiltonian \( \hat{H}(t) \) is a \( 2^k \times 2^k \) matrix-valued function and has the form

\[
\hat{H}(t) = D + B(\cos(2\pi \nu t) + \cos(4\pi \nu t)),
\]

with \( D, B \) sparse matrices described in [10]. In our experiments, we set \( T = 10^{-3}, \nu = 10^4 \), and \( k = 4, 7, 10 \), so obtaining three systems with exponentially increasing sizes.

The approximated solution \( \hat{u}(t) \) \((11)\) is computed by solving the linear system \((10)\) with \( M = 1000 \). The numerical experiments were performed using MatLab R2022a, and the linear systems were solved by the MatLab GMRES method implementation, \texttt{gmres}, with tolerance set to \( 1e-15 \). In Figure 1, we compare the approximated bilinear form \( v^T \hat{u}(t) \) with the solution obtained by the MatLab function \texttt{ode45} with relative and absolute tolerance set to \( 3e-14 \). Figure 2 reports the corresponding relative and absolute errors over the interval \([0, T]\) (the reference for the error is again the \texttt{ode45} solution). In all the experiments, GMRES stopped after a maximum of 27 iterations (for the cases \( k = 7, 10 \) due to residual stagnation). The numerical results show that the method is able to compute the solution with accuracy comparable with a well-established method.

![Fig. 1: Real and imaginary parts of \( v^T u(t) \) approximations, with \( u(t) \) the solution of (12). The red circles represent approximation \( v^T \hat{u}(t) \) from (11), while the blue line represents the \texttt{ode45} approximation. From left to right, \( k = 4, 7, 10 \).

![Fig. 2: Absolute (blue circles) and relative (red crosses) errors of approximation \( v^T \hat{u}(t) \), with \( \hat{u}(t) \) from (11). From left to right, \( k = 4, 7, 10 \).](image)

3 Matrix equation formulation

The matrix-valued function \( \hat{A}(t) \) in \((1)\) can always be written in the form

\[
\hat{A}(t) = \sum_{k=1}^{d} A_k \hat{f}_k(t),
\]

\footnote{The matrices \( F^{(i,j)}_{M} \) in the block coefficient matrix \((8)\) are numerically banded with bandwidth \( b_{i,j} \). In order to avoid error accumulation, the last \( b_{i,j} \) rows of each \( F^{(i,j)}_{M} \) have been set equal to zero; see [2].}
with \( \tilde{f}_1, \ldots, \tilde{f}_d \) distinct scalar functions and \( A_1, \ldots, A_d \) constant matrices. In many applications, \( d \) is small. For instance, in the examples from Section 2.1, we have \( d = 2 \). Then, exploiting expression (14), the (block) coefficient matrix (8) of \( A(t, s) = \tilde{A}(t)\Theta(t-s) \) becomes

\[
A_M = \sum_{k=1}^{d} A_k \otimes F_M^{(k)},
\]

with \( F_M^{(k)} \) the coefficient matrix (6) of \( \tilde{f}_k(t) \). The solution \( x \) of the linear system (10) can, hence, be rewritten in terms of the solution \( X \) of the following matrix equation

\[
X - \sum_{k=1}^{d} F_M^{(k)} X A_T^k = \phi_M(0)b^T, \quad x = vec(X),
\]

where \( vec(X) \) denotes the vectorization of \( X \), i.e., the vector obtained by stacking the columns of \( X \) into a single vector. The matrix equation (15) has a rank 1 right-hand side \( \phi_M(0)b^T \). This suggests that the solution \( X \) may have a low numerical rank. Figure 3 reports the computed singular values of \( X \), where \( x = vec(X) \) is the linear system solution of each of the experiments performed in Section 2.1. For \( k = 4 \), the solution \( X \) is full rank, while for \( k = 7, 10 \), the numerical rank of \( X \) is, respectively, 12, 72 (we consider as numerical rank the index of the last singular value before the stagnation visible in the plots). Clearly, this preliminary study shows that the numerical rank of \( X \) increases slowly with the size of \( X \).

![Fig. 3: Singular values of the matrix \( X \), with \( x = vec(X) \) the solution of (10) for the examples in Section 2.1. From left to right, \( k = 4, 7, 10 \).](image)

### 4 Discussions and conclusion

In this work, we present a new method for solving systems of non-autonomous linear ODEs. The method is based on the solution of a linear system that can be rewritten as a matrix equation. Several examples illustrate that the method is able to compute the solution with accuracy comparable to the well-established Runge-Kutta method implemented by the MatLab function \texttt{ode45}. Moreover, the experiments show that the solution of the matrix equation is a numerical low-rank matrix when the ODE system is large enough. This may be exploited using projection methods with low-rank techniques (see, e.g., [11, 12]). In [10], we also show that matrix \( A_M \) in (8) can be compressed by the Tensor Train decomposition (note that [10] uses a different family of orthogonal functions instead of the Legendre polynomials). A Tensor Train approach may further reduce the memory and computational cost of the method. Another possible approach could be extrapolation methods able to exploit the dependence of equation (10) on \( s \); see, e.g., [13, 14].

Overall, the results suggest that the presented method may be an effective solver for large-to-huge systems of ODEs once we are able to exploit the solution’s low-rank structure and the other mentioned properties. We are currently investigating these possible approaches.

**Acknowledgements** This work was supported by Charles University Research programs UNCE/SCI/023 and PRIMUS/21/SCI/009 and by the Magica project ANR-20-CE29-0007 funded by the French National Research Agency.

**References**

[1] S. Hafner and H. W. Spiess, Concepts Magn. Reson. **10**, 99–128 (1998).
[2] S. Pozza and N. Van Buggenhout, The *-product approach for linear odes: a numerical study of the scalar case, 2022.
[3] L. Schwartz, Théorie Des Distributions (Hermann, Paris, 1978).
[4] P. L. Giscard, K. Lui, S. J. Thwaite, and D. Jaksch, J. Math. Phys. 56(5), 053503 (2015).
[5] P. L. Giscard and C. Bonhomme, Phys. Rev. Research 2(Apr), 023081 (2020).
[6] P. L. Giscard and S. Pozza, Appl. Math. 65(6), 807–827 (2020).
[7] P. L. Giscard and S. Pozza, Linear Algebra Appl. 624, 153–173 (2021).
[8] P. L. Giscard and S. Pozza, Boll Unione Mat Ital (2022).
[9] R. A. Silverman et al., Special functions and their applications (Courier Corporation, 1972).
[10] S. Cipolla, S. Pozza, M. Redivo-Zaglia, and N. Van Buggenhout, Numer. Algorithms (2022).
[11] V. Simoncini, SIAM J. Sci. Comput. 29(3), 1268–1288 (2007).
[12] D. Palitta and P. Kürschner, Numer. Algorithms 88(3), 1383–1417 (2021).
[13] C. Brezinski and M. Redivo Zaglia, Extrapolation methods, Studies in Computational Mathematics, Vol. 2 (North-Holland Publishing Co., Amsterdam, 1991), Theory and practice, With 1 IBM-PC floppy disk (5.25 inch).
[14] D. Buoso, A. Karapiperi, and S. Pozza, Appl. Numer. Math. 90, 38–54 (2015).