Bollettino dell'Unione Matematica Italiana (2023) 16:81–102
https://doi.org/10.1007/s40574-022-00328-6

A Lanczos-like method for non-autonomous linear ordinary differential equations

Pierre-Louis Giscard[1] · **Stefano Pozza**[2]

Received: 7 January 2022 / Accepted: 28 May 2022 / Published online: 25 June 2022

Abstract
The time-ordered exponential is defined as the function that solves a system of coupled first-order linear differential equations with generally non-constant coefficients. In spite of being at the heart of much system dynamics, control theory, and model reduction problems, the time-ordered exponential function remains elusively difficult to evaluate. The $*$-Lanczos algorithm is a (symbolic) algorithm capable of evaluating it by producing a tridiagonalization of the original differential system. In this paper, we explain how the $*$-Lanczos algorithm is built from a generalization of Krylov subspaces, and we prove crucial properties, such as the *matching moment property*. A strategy for its numerical implementation is also outlined and will be subject of future investigation.

Keywords Lanczos algorithm · Matrix differential equations · Time-ordered exponential · Matching moments · Tridiagonal matrices · Ordinary differential equations

1 Introduction

Let $t' \geq t \in I \subseteq \mathbb{R}$ be variables—called times for convenience—in an interval I, and $\mathsf{A}(t')$ be an $N \times N$ time-dependent matrix. For a fixed t (usually $t = 0$), the time-ordered exponential of $\mathsf{A}(t')$ is defined as the unique solution $\mathsf{U}(t', t)$ of the non autonomous system of linear ordinary differential equations

$$\mathsf{A}(t')\mathsf{U}(t', t) = \frac{d}{dt'}\mathsf{U}(t', t), \quad \mathsf{U}(t, t) = \mathsf{Id}, \quad t' \geq t, \tag{1.1}$$

These authors contributed equally to this work.

✉ Stefano Pozza
pozza@karlin.mff.cuni.cz

Pierre-Louis Giscard
giscard@univ-littoral.fr

[1] Univ. Littoral Côte d'Opale, UR 2597, LMPA, Laboratoire de Mathématiques Pures et Appliquées Joseph Liouville, F-62100 Calais, France

[2] Faculty of Mathematics and Physics, Charles University, Prague 8, Czech Republic

with Id the identity matrix. Note that t represents the time on which the initial condition is given, and that the (unusual) notation $\mathsf{U}(t', t)$ will be useful later. If the matrix A commutes with itself at all times, i.e., $\mathsf{A}(\tau_1)\mathsf{A}(\tau_2) - \mathsf{A}(\tau_2)\mathsf{A}(\tau_1) = 0$ for all $\tau_1, \tau_2 \in I$, then the time-ordered exponential is given by the matrix exponential $\mathsf{U}(t', t) = \exp\left(\int_t^{t'} \mathsf{A}(\tau)\,\mathrm{d}\tau\right)$. However, when A does not commute with itself at all times, the time-ordered exponential has no known explicit form in terms of A and is rather denoted

$$\mathsf{U}(t', t) = \mathcal{T} \exp\left(\int_t^{t'} \mathsf{A}(\tau)\,\mathrm{d}\tau\right),$$

with $\mathcal{T}$ the time-ordering operator [1]. This expression, introduced by Dyson in 1952, is more a notation than an explicit form as the action of the time-ordering operator is very difficult to evaluate. In particular, $\mathsf{U}(t', t)$ does not have a Cauchy integral representation, and it cannot be evaluated via ordinary diagonalization. It is unlikely that a closed form expression for $\mathsf{U}(t', t)$ in terms of A exists at all since even when A is 2×2, U can involve very complicated special functions [2, 3].

Evaluating time-ordered exponentials is a central question in the field of system dynamics, in particular in quantum physics where A is the Hamiltonian operator. Situations where this operator does not commute with itself are routinely encountered [4], and the departure of the time-ordered exponential from a straightforward matrix exponential is responsible for many peculiar physical effects [5–7]. Further applications are found via differential Lyapunov and Riccati matrix equations, which frequently appear in control theory, filter design, and model reduction problems [8–12]. Indeed, the solutions of such differential equations involve time-ordered exponentials [13–16].

In [17], we introduced a tridiagonal form for the matrix $\mathsf{A}(t')$ from which it is possible to express a time-ordered exponential via path-sum continued fractions of finite depth. More precisely, the described procedure formulates each element of a time-ordered exponential in terms of a finite and treatable number of scalar integro-differential equations. Such a tridiagonal form is obtained by using the so-called $*$- *Lanczos algorithm*. The algorithm also appeared in [18] as the paper's motivation. Despite being at the core of results in [17] and being the motivation of [18], the $*$-Lanczos algorithm construction and the proofs of related main properties have yet not appeared in a scientific journal. The present paper aims to fill a gap in the literature as it constructs the $*$-Lanczos algorithm from a (generalized) Krylov subspace perspective, proves the related *Matching Moment Property*, introduces a bound for the approximation error, and adds further results on the breakdown issue that may affect the method.

1.1 Existing analytical approaches: Pitfalls and drawbacks

In spite of the paramount importance of the time-ordered exponential, it is usually omitted from the literature on matrix functions. Until 2015, only two families of analytical approaches existed (numerical methods will be discussed in Sect. 5). The first one to have been devised relies on Floquet theory and necessitates $\mathsf{A}(t')$ to be periodic (see, e.g., [4]). This method transforms Eq. (1.1) into an *infinite* system of coupled linear differential equations with *constant* coefficients. This system is then solved by a perturbative expansion at very low order relying on the presence of a small parameter. Orders higher than 2 or 3 are typically too involved to be treated. The second method was developed in 1954 by Wilhelm Magnus [19]. It produces an infinite series of nested commutators of A with itself at different times, the

ordinary matrix exponential of which provides the desired solution $\mathsf{U}(t', t)$. Magnus series are very much in use nowadays [4], especially because they guarantee that the approximation to $\mathsf{U}(t', t)$ is unitary in quantum mechanical calculations [4]. Nevertheless, the Magnus series for $\mathsf{U}(t', t)$ has a small convergence domain; see [20] and also [21–25].

In 2015, in the joint article [26], one of the authors presented a third method to obtain time-ordered exponentials using graph theory and necessitating only the entries $\mathsf{A}(t')_{k\ell}$ to be bounded functions of time [26]. The method formulates any desired entry or group of entries of $\mathsf{U}(t', t)$ as a branched continued fraction of *finite* depth and breadth. It has been succesfully used to solve challenging quantum dynamic problems, see e.g. [27, 28]. This approach is unconditionally convergent and it provides exact expressions in terms of a finite number of integrals and Volterra equations. However, it suffers from a complexity drawback. Indeed, it requires one to find all the simple cycles and simple paths of a certain graph G. These are the walks on G which are not self-intersecting. Unfortunately, the problem of enumerating such walks is #P-complete[1] [29], hindering the determination of exact solutions in large systems that must be treated using a further property of analytical path-sums called scale-invariance [28]. The present work with the results in [17, 18] solves this issue by transforming the original matrix into a structurally simpler one on which the path-sum solution takes the form of an ordinary, finite, continued fraction.

1.2 The non-Hermitian Lanczos algorithm: background

Consider the simpler case in which A is not time-dependent. The solution of (1.1) is given by the matrix function $\exp(\mathsf{A}(t' - t))$ which can be numerically approximated in several different ways (see, e.g., [30–32]). One possible method is the (non-Hermitian) Lanczos algorithm. Computing the (k, ℓ) element of $\exp(\mathsf{A})$ is equivalent to computing the bilinear form $\boldsymbol{e}_k^H \exp(\mathsf{A})\, \boldsymbol{e}_\ell$, with $\boldsymbol{e}_k$, $\boldsymbol{e}_\ell$ vectors from the canonical Euclidean basis, and $\boldsymbol{e}_k^H$ the usual Hermitian transpose (here it coincides with the transpose since the vector is real). The non-Hermitian Lanczos algorithm (e.g., [33–36]) gives, when no breakdown occurs, the matrices

$$\mathsf{V}_n = [\boldsymbol{v}_0, \ldots, \boldsymbol{v}_{n-1}], \quad \mathsf{W}_n = [\boldsymbol{w}_0, \ldots, \boldsymbol{w}_{n-1}],$$

whose columns are biorthonormal bases respectively for the Krylov subspaces

$$\mathrm{span}\{\boldsymbol{e}_\ell, \mathsf{A}\,\boldsymbol{e}_\ell, \ldots, \mathsf{A}^{n-1}\,\boldsymbol{e}_\ell\}, \quad \mathrm{span}\{\boldsymbol{e}_k, \mathsf{A}^H\boldsymbol{e}_k, \ldots, (\mathsf{A}^H)^{n-1}\,\boldsymbol{e}_k\}.$$

Note that for A Hermitian and $k = \ell$ we can equivalently use the Hermitian Lanczos algorithm (getting $\mathsf{V}_n = \mathsf{W}_n$). The so-called (complex) *Jacobi matrix* J_n is the tridiagonal symmetric matrix with generally complex elements obtained by

$$\mathsf{J}_n = \mathsf{W}_n^H \mathsf{A}\, \mathsf{V}_n.$$

As described in [35], we can use the approximation

$$\boldsymbol{e}_k^H \exp(\mathsf{A})\boldsymbol{e}_\ell \approx \boldsymbol{e}_1^H \exp(\mathsf{J}_n)\boldsymbol{e}_1, \tag{1.2}$$

which relies on the so-called *matching moment property*, i.e.,

$$\boldsymbol{e}_k^H (\mathsf{A})^j \boldsymbol{e}_\ell = \boldsymbol{e}_1^H (\mathsf{J}_n)^j \boldsymbol{e}_1, \quad j = 0, 1, \ldots, 2n-1; \tag{1.3}$$

[1] The #P class is the class of problems equivalent to counting the number of accepting paths of a polynomial-time non-deterministic Turing machine. Problems in #P-complete are at least as hard as NP-complete problems.

see, e.g., [35, 36] for the Hermitian case, and [37, 38] for the non-Hermitian one. The approximation (1.2) is a model reduction in two ways. First, the size of J_n is much smaller than that of A. Second, the structure of the matrix J_n is much simpler since it is tridiagonal.

Given a matrix A with size N, the Lanczos algorithm can be used as a method for its tridiagonalization (see, e.g., [39]). Assuming no breakdown, the Nth iteration of the non-Hermitian Lanczos with input the matrix A and a couple of vectors $\boldsymbol{v}$, $\boldsymbol{w}$ produces the tridiagonal matrix J_N, and the biorthogonal square matrices $\mathsf{V}_N, \mathsf{W}_N$ so that

$$\mathsf{A}^j = \mathsf{V}_N \, (\mathsf{J}_N)^j \, \mathsf{W}_N^H, \quad j = 0, 1, \ldots, \tag{1.4}$$

giving the exact expression

$$\exp(\mathsf{A}) = \mathsf{V}_N \exp(\mathsf{J}_N) \, \mathsf{W}_N^H. \tag{1.5}$$

Theorem 2.2 which we prove in this work extends this result to time-ordered exponentials.

The Lanczos approximation (1.2) is connected with several further topics, such as (formal) orthogonal polynomials, Gauss quadrature, continued fractions, the moment problem, and many others. Information about these connections and references to the related rich and vast literature can be found, e.g., in the monographs [35, 36, 40] and the surveys [37, 38].

Inspired by approximation (1.2), the $*$-Lanczos algorithm produces a model reduction of a time-ordered exponential by providing a time-dependent tridiagonal matrix T_n satisfying properties analogous to the ones described above [17]. Differently from the classical case, the $*$-Lanczos algorithm works on vector distribution subspaces and it has to deal with a non-commutative product.

The time-dependent framework in which the proposed method works is much more complicated than the time-independent Krylov subspace approximation given by the Lanczos algorithm. In this paper, we will not deal with the behavior of the $*$-Lanczos algorithm when taking into account approximations and finite-precision arithmetic problems.

1.3 Outline

The work is organized as follows: In Sect. 2, we build the $*$-Lanczos algorithm. The algorithm relies on a non-commutative $*$-product between generalized functions of two-time variables, which we describe in Sect. 2.1. Then, in Sect. 2.2, we state the main result, Theorem 2.1, which underpins the Lanczos-like procedure. Theorem 2.2 establishes that the first $2n$ $*$-moments of a certain tridiagonal matrix T_n match the corresponding $*$-moments of the original matrix A. Theorem 2.1 is proved with the tools developed in the subsequent Sect. 2.3. Section 3 is devoted to the convergence and breakdown properties of the algorithm, while examples of its use are given in Sect. 4. In Sect. 5 we outline a way to implement the Lanczos-like procedure numerically and we evaluate its computational cost. Section 6 concludes the paper.

2 The $*$-Lanczos algorithm

2.1 The $*$-product and $*$-moments

In this section, we recall the definition and the main properties of the product introduced in [18, Section 1.2].

Let t and t' be two real variables in a real time interval I. We consider the class $\mathrm{D}(I)$ of all distributions which are linear superpositions of Heaviside theta functions and Dirac delta derivatives with smooth coefficients over I^2. That is, a distribution d is in $\mathrm{D}(I)$ if and only if it can be written as

$$d(t',t) = \widetilde{d}(t',t)\Theta(t'-t) + \sum_{i=0}^{N} \widetilde{d}_i(t',t)\delta^{(i)}(t'-t), \tag{2.1}$$

where $N \in \mathbb{N}$ is finite, $\Theta(\cdot)$ stands for the Heaviside theta function (with the convention $\Theta(0) = 1$) and $\delta^{(i)}(\cdot)$ is the ith derivative of the Dirac delta distribution $\delta = \delta^{(0)}$. Here and from now on, a tilde over a function (e.g., $\widetilde{d}(t',t)$) indicates that it is an ordinary function *smooth* in both $t' \in I$ and $t \in I$. Note that we consider distributions as defined by Schwartz ([41]). Hence a distribution $f \in \mathrm{D}(I)$ should be interpreted as a linear functional applied to test functions.

We can endow the class $\mathrm{D}(I)$ with a non-commutative algebraic structure upon defining a product between its elements. For $f_1, f_2 \in \mathrm{D}(I)$ we define the convolution-like $*$ product between $f_1(t',t)$ and $f_2(t',t)$ as

$$\big(f_2 * f_1\big)(t',t) := \int_{-\infty}^{\infty} f_2(t',\tau) f_1(\tau,t)\,\mathrm{d}\tau, \tag{2.2}$$

that has as identity element the Dirac delta distribution, $1_* := \delta(t'-t)$. When $f(t',t) = f(t'-t)$ has bounded supporting set, the $*$-product $f * g$ (and $g * f$) is equivalent to the convolution product for distributions defined by Schwartz ([42, § 11] and [41, Chapter VI]). Since $\delta^{(i)}(t'-t)$ has bounded supporting set, given $f \in \mathrm{D}(I)$, the $*$-product $\delta^{(i)}(t'-t) * f$ and $f * \delta^{(i)}(t'-t)$ are well-defined and are both elements of $\mathrm{D}(I)$; see [18] for further details. Moreover, it holds

$$\begin{aligned} \delta^{(i)}(t'-t) * \delta^{(j)}(t'-t) &= \delta^{(j)}(t'-t) * \delta^{(i)}(t'-t) = \delta^{(i+j)}(t'-t); \\ \Theta(t'-t) * \delta'(t'-t) &= \delta'(t'-t) * \Theta(t'-t) = \delta(t'-t). \end{aligned}$$

Consider the subclass $\mathrm{Sm}_\Theta(I)$ of $\mathrm{D}(I)$ comprising those distributions of the form

$$f(t',t) = \tilde{f}(t',t)\Theta(t'-t). \tag{2.3}$$

For $f_1,\ f_2 \in \mathrm{Sm}_\Theta(I)$, the $*$-product between $f_1,\ f_2$ simplifies to

$$\begin{aligned} \big(f_2 * f_1\big)(t',t) &= \int_{-\infty}^{\infty} \tilde{f}_2(t',\tau)\tilde{f}_1(\tau,t)\Theta(t'-\tau)\Theta(\tau-t)\,\mathrm{d}\tau, \\ &= \Theta(t'-t)\int_{t}^{t'} \tilde{f}_2(t',\tau)\tilde{f}_1(\tau,t)\,\mathrm{d}\tau, \end{aligned}$$

which makes calculations involving such functions easier to carry out and shows that $\mathrm{Sm}_\Theta(I)$ is closed under $*$-multiplication. Together with the arguments above, this proves that $\mathrm{D}(I)$ is closed under $*$-multiplication. Hence, for $f \in \mathrm{D}(I)$, we can define its kth $*$-*power* f^{*k} as the k $*$-products $f * f * \cdots * f$, with the convention $f^{*0} = \delta(t'-t)$. First examples of $*$-powers are

$$\Theta^{*k}(t'-t) = \frac{(t'-t)^{k-1}}{(k-1)!}\Theta(t'-t); \tag{2.4}$$

$$\Big(\delta^{(j)}(t'-t)\Big)^{*k} = \delta^{(kj)}(t'-t). \tag{2.5}$$

Note that, for members of $\mathrm{Sm}_\Theta(I)$, the $*$-product reduces exactly to the Volterra composition, a product between smooth functions of two-variables developed by Volterra and Pérès [43].

We will not discuss technicality associated with the $*$-product by and between distributions such as Dirac delta derivatives since it would bring us too far from the paper's goals. More details on the distributions themselves can be found in [41] while illustrative examples of $*$-products involving such distributions can be found in [18].

The $*$-product extends directly to distributions of $\mathrm{D}(I)$ whose smooth coefficients depend on less than two variables. Indeed, consider a generalized function $f_3(t',t) = \tilde{f}_3(t')\delta^{(i)}(t'-t)$ with $i \geq -1$ and $\delta^{(-1)} = \Theta$. Then

$$\big(f_3 * f_1\big)(t',t) = \tilde{f}_3(t') \int_{-\infty}^{+\infty} \delta^{(i)}(t'-\tau) f_1(\tau,t)\,\mathrm{d}\tau,$$
$$\big(f_1 * f_3\big)(t',t) = \int_{-\infty}^{+\infty} f_1(t',\tau)\tilde{f}_3(\tau)\delta^{(i)}(\tau-t)\,\mathrm{d}\tau,$$

where $f_1(t',t) \in \mathrm{Sm}_\Theta(I)$ is as defined on p. 6. Hence the variable of $\tilde{f}_3(t')$ is treated as the left variable of a smooth function of two variables. This observation extends straightforwardly should $\tilde{f}_3$ be constant and, by linearity, to any distribution of $\mathrm{D}(I)$.

The $*$-product also naturally extends to matrices whose entries are distributions of $\mathrm{D}(I)$. Consider two of such matrices $\mathsf{A}_1(t',t)$ and $\mathsf{A}_2(t',t) \in \mathrm{D}(I)^{N\times N}$ then

$$\big(\mathsf{A}_2 * \mathsf{A}_1\big)(t',t) := \int_{-\infty}^{+\infty} \mathsf{A}_2(t',\tau)\mathsf{A}_1(\tau,t)\,\mathrm{d}\tau,$$

where the sizes of $\mathsf{A}_1, \mathsf{A}_2$ are compatible for the usual matrix product (here and in the following, we omit the dependency on t' and t when it is clear from the context) and the integral is treated component-wise. As earlier, the $*$-product is associative and distributive with respect to the addition, but it is non-commutative. The identity element with respect to this product is now $\mathsf{Id}_* := \mathsf{Id}\,1_*$, with Id the identity matrix of appropriate size.

Given a square matrix $\mathsf{A}(t',t)$ composed of elements from $\mathrm{D}(I)$, we define the k-th *matrix $*$-power* A^{*k} as the k $*$-products $\mathsf{A} * \mathsf{A} * \cdots * \mathsf{A}$. In particular, by (2.4) we get the bound

$$\|\mathsf{A}^{*k}(t',t)\|_\star \leq \left(\sup_{\substack{\tau\geq\rho \\ \tau,\rho\in I}} \|\mathsf{A}(\tau,\rho)\|_\star \right)^k \frac{(t'-t)^{k-1}}{(k-1)!}\Theta(t'-t); \quad t',t \in I,$$

with $\|\cdot\|_\star$ any induced matrix norm. As a consequence, the $*$-resolvent of any matrix depending on at most two variables is well defined, as $\mathsf{R}_*(\mathsf{A}) := (\mathsf{Id}_* - \mathsf{A})^{*-1} = \mathsf{Id}_* + \sum_{k\geq 1} \mathsf{A}^{*k}$ exists provided every entry of A is bounded for all $t', t \in I$ (see [26]). Then

$$\mathsf{U}(t',t) = \Theta(t'-t) * \mathsf{R}_*(\mathsf{A})(t',t) \tag{2.6}$$

is the time-ordered exponential of $\mathsf{A}(t',t)$; see [26]. We recall that $\Theta(\cdot)$ here stands for the Heaviside function under the convention that $\Theta(0) = 1$. Note that time-ordered exponentials are usually presented with only one-time variable, corresponding to $\mathsf{U}(t) = \mathsf{U}(t,0)$. Yet, in general $\mathsf{U}(t',t) \neq \mathsf{U}(t'-t,0)$.

In the spirit of the Lanczos algorithm, given a time-dependent matrix $\mathsf{A}(t',t)$, we will construct a matrix $\mathsf{T}_n(t',t)$ of size $n \leq N$ with a simpler (tridiagonal) structure and so that, fixing the indexes k, ℓ, it holds

$$\big(\mathsf{A}^{*j}(t',t)\big)_{k,\ell} = \big(\mathsf{T}_n^{*j}(t',t)\big)_{1,1}, \quad \text{for} \quad j = 0,\ldots,2n-1, \quad t',t \in I; \tag{2.7}$$

compare it with (1.3). In particular, when $n = N$ property (2.7) stands for every $j \geq 0$, giving

$$\mathsf{R}_*(\mathsf{A})_{k,\ell} = \mathsf{R}_*(\mathsf{T}_N)_{1,1}.$$

Hence the solution is given by the path-sum techniques exploiting the fact that the graph having T_N as its adjacency matrix admits self-loops. More in general, given time-independent vectors $\boldsymbol{v}, \boldsymbol{w}$ we call the jth $*$-*moment* of $\mathsf{A}, \boldsymbol{v}, \boldsymbol{w}$ the scalar function $\boldsymbol{w}^H(\mathsf{A}^{*j}(t', t))\,\boldsymbol{v}$, for $j \geq 0$ (note that when the * product symbol is omitted, it stands for the usual matrix-vector product). Then property (2.7) is an instance of the more general case

$$\boldsymbol{w}^H(\mathsf{A}^{*j}(t', t))\,\boldsymbol{v} = \boldsymbol{e}_1^H(\mathsf{T}_n^{*j}(t', t))\,\boldsymbol{e}_1, \quad \text{for} \quad j = 0, \ldots, 2n-1, \quad t', t \in I.$$

2.2 Building up the $*$-Lanczos process

Given a doubly time-dependent matrix $\mathsf{A}(t', t) = \widetilde{\mathsf{A}}(t')\Theta(t' - t)$ and $k+1$ scalar generalized functions $\alpha_0(t', t), \alpha_1(t', t), \ldots, \alpha_k(t', t) \in \mathrm{D}(I)$ which play the role of the *coefficients*, we define the *matrix* $*$-*polynomial* $p(\mathsf{A})(t', t)$ of degree k as

$$p(\mathsf{A})(t', t) := \sum_{j=0}^{k} \left(\mathsf{A}^{*j} * \alpha_j\right)(t', t);$$

moreover, we define the corresponding *dual* matrix $*$-polynomial as

$$p^D(\mathsf{A})(t', t) := \sum_{j=0}^{k} \left(\bar{\alpha}_j * (\mathsf{A}^{*j})\right)(t', t),$$

where, in general, $\bar{d}$ is the conjugated value of $d \in \mathrm{D}(I)$ and it is defined by conjugating the functions $\widetilde{d}$ and $\widetilde{d}_i$ in (2.1). Let $\boldsymbol{v}$ be a time independent vector, we can define the set of time-dependent vectors $p(\mathsf{A})\boldsymbol{v}$, with p a matrix $*$-polynomial. Such a set is a vector space with respect to the product $*$ and with scalars $\alpha_j(t', t)$ (the addition is the usual addition between vectors). Similarly, given a vector $\boldsymbol{w}^H$ not depending on time, we can define the vector space given by the dual vectors $\boldsymbol{w}^H p^D(\mathsf{A})$. In particular, we can define the $*$-Krylov subspaces

$$\mathcal{K}_n(\mathsf{A}, \boldsymbol{v})(t', t) := \left\{ (\,p(\mathsf{A})\boldsymbol{v})\,(t', t) \mid p \text{ of degree at most } n-1 \right\},$$
$$\mathcal{K}_n^D(\mathsf{A}, \boldsymbol{w})(t', t) := \left\{ \left(\boldsymbol{w}^H p^D(\mathsf{A})\right)(t', t) \mid p \text{ of degree at most } n-1 \right\}.$$

The vectors $\boldsymbol{v}, \mathsf{A}\boldsymbol{v}, \ldots, \mathsf{A}^{*(n-1)}\boldsymbol{v}$ and $\boldsymbol{w}^H, \boldsymbol{w}^H\mathsf{A}, \ldots, \boldsymbol{w}^H\mathsf{A}^{*(n-1)}$ are bases respectively for $\mathcal{K}_n(\mathsf{A}, \boldsymbol{v})$ and $\mathcal{K}_n^D(\mathsf{A}, \boldsymbol{w})$. We aim to derive $*$-*biorthonormal* bases $\boldsymbol{v}_0, \ldots, \boldsymbol{v}_{n-1}$ and $\boldsymbol{w}_0^H, \ldots, \boldsymbol{w}_{n-1}^H$ for the $*$-Krylov subspaces, i.e., so that

$$\boldsymbol{w}_i^H * \boldsymbol{v}_j = \delta_{ij}\, 1_*, \tag{2.8}$$

with δ_{ij} the Kronecker delta.

Assume that $\boldsymbol{w}^H\boldsymbol{v} = 1$, we can use a non-Hermitian Lanczos like biorthogonalization process for the triplet $(\boldsymbol{w}, \mathsf{A}(t', t), \boldsymbol{v})$. We shall call this method the $*$-*Lanczos process*. The first vectors of the biorthogonal bases are

$$\boldsymbol{v}_0 = \boldsymbol{v}\, 1_*, \quad \boldsymbol{w}_0^H = \boldsymbol{w}^H 1_*,$$

so that $\boldsymbol{w}_0^H * \boldsymbol{v}_0 = 1_*$. Consider now a vector $\widehat{\boldsymbol{v}}_1 \in \mathcal{K}_2(\mathsf{A}, \boldsymbol{v})$ given by

$$\widehat{\boldsymbol{v}}_1 = \mathsf{A} * \boldsymbol{v}_0 - \boldsymbol{v}_0 * \alpha_0 = \mathsf{A}\boldsymbol{v} - \boldsymbol{v}\alpha_0.$$

If the vector satisfies the $*$-biorthogonal condition $\boldsymbol{w}_0^H * \widehat{\boldsymbol{v}}_1 = 0$, then

$$\alpha_0 = \boldsymbol{w}_0^H * \mathsf{A} * \boldsymbol{v}_0 = \boldsymbol{w}^H \mathsf{A}\, \boldsymbol{v}. \tag{2.9}$$

Similarly, we get the expression

$$\widehat{\boldsymbol{w}}_1^H = \boldsymbol{w}_0^H * \mathsf{A} - \alpha_0 * \boldsymbol{w}_0^H = \boldsymbol{w}^H \mathsf{A} - \alpha_0 \boldsymbol{w}^H,$$

with α_0 given by (2.9). Hence the $*$-biorthonormal vectors are defined as

$$\boldsymbol{v}_1 = \widehat{\boldsymbol{v}}_1 * \beta_1^{*-1}, \quad \boldsymbol{w}_1 = \widehat{\boldsymbol{w}}_1,$$

with $\beta_1 = \widehat{\boldsymbol{w}}_1^H * \widehat{\boldsymbol{v}}_1$ and β_1^{*-1} its $*$-*inverse*, i.e., $\beta_1^{*-1} * \beta_1 = \beta_1 * \beta_1^{*-1} = 1_*$. We give sufficient conditions for the existence of such $*$-inverses below. Assume now that we have the $*$-biorthonormal bases $\boldsymbol{v}_0, \ldots, \boldsymbol{v}_{n-1}$ and $\boldsymbol{w}_0^H, \ldots, \boldsymbol{w}_{n-1}^H$. Then we can build the vector

$$\widehat{\boldsymbol{v}}_n = \mathsf{A} * \boldsymbol{v}_{n-1} - \sum_{i=0}^{n-1} \boldsymbol{v}_i * \gamma_i,$$

where the γ_i are determined by the condition $\boldsymbol{w}_j^H * \widehat{\boldsymbol{v}}_n = \delta_{jn} 1_*$, for $j = 0, \ldots, n-1$, giving

$$\gamma_j = \boldsymbol{w}_j^H * \mathsf{A} * \boldsymbol{v}_{n-1}, \quad j = 0, \ldots, n-1.$$

In particular, since $\boldsymbol{w}_j^H * \mathsf{A} \in \mathcal{K}_{j+1}^D(\mathsf{A}, \boldsymbol{w})$ we get $\gamma_j = 0$ for $j = 0, \ldots, n-3$. This leads to the following three-term recurrences for $n = 1, 2, \ldots$ using the convention $\boldsymbol{v}_{-1} = \boldsymbol{w}_{-1} = 0$,

$$\boldsymbol{w}_n^H = \boldsymbol{w}_{n-1}^H * \mathsf{A} - \alpha_{n-1} * \boldsymbol{w}_{n-1}^H - \beta_{n-1} * \boldsymbol{w}_{n-2}^H, \tag{2.10a}$$

$$\boldsymbol{v}_n * \beta_n = \mathsf{A} * \boldsymbol{v}_{n-1} - \boldsymbol{v}_{n-1} * \alpha_{n-1} - \boldsymbol{v}_{n-2}, \tag{2.10b}$$

with the coefficients given by

$$\alpha_{n-1} = \boldsymbol{w}_{n-1}^H * \mathsf{A} * \boldsymbol{v}_{n-1}, \quad \beta_n = \boldsymbol{w}_n^H * \mathsf{A} * \boldsymbol{v}_{n-1}. \tag{2.11}$$

Should β_n not be $*$-invertible, we would get a *breakdown* in the algorithm, since it would be impossible to compute $\boldsymbol{v}_n$. We developed a range of general methods to determine the $*$-inverse of functions of two-time variables which are gathered in [18]. These methods *constructively* show the existence of β_n^{*-1} almost everywhere on I under the following conditions:

- β_n is not identically zero in I^2;
- $\beta_n \in \mathrm{Sm}_\Theta(I)$.

The question of whether all $\alpha_n, \beta_n \in \mathrm{Sm}_\Theta(I)$ was settled affirmatively in [17].

Since the issue of breakdowns of the $*$-Lanczos algorithm is connected with the behavior of (usual) Lanczos techniques, we proceed as it is common when working with the non-Hermitian Lanczos algorithm. Thus, from now on, we assume all β_n to be $*$-invertible, while we come back to the issue of breakdowns in Sect. 3.2.

The $*$-orthogonalization process described above defines the $*$-Lanczos algorithm (Algorithm 1).

Input: A complex time-dependent matrix $\mathsf{A} = \widetilde{\mathsf{A}}(t')\Theta(t'-t)$, and time-independent complex vectors $\boldsymbol{v}$, $\boldsymbol{w}$ such that $\boldsymbol{w}^H\boldsymbol{v} = 1$.
Output: Vectors $\boldsymbol{v}_0, \ldots, \boldsymbol{v}_{n-1}$ and vectors $\boldsymbol{w}_0, \ldots, \boldsymbol{w}_{n-1}$ spanning respectively $\mathcal{K}_n(\mathsf{A}, \boldsymbol{v})$, $\overline{\mathcal{K}_n(\mathsf{A}, \boldsymbol{w})}$ and satisfying the $*$-biorthogonality conditions (2.8). The coefficients $\alpha_0, \ldots, \alpha_{n-1}$ and $\beta_1, \ldots, \beta_n$ from the recurrences (2.10).

Initialize: $\boldsymbol{v}_{-1} = \boldsymbol{w}_{-1} = 0$, $\boldsymbol{v}_0 = \boldsymbol{v}\,1_*$, $\boldsymbol{w}_0^H = \boldsymbol{w}^H 1_*$

$\alpha_0 = \boldsymbol{w}^H \mathsf{A}\,\boldsymbol{v}$

$\boldsymbol{w}_1^H = \boldsymbol{w}^H\mathsf{A} - \alpha_0\,\boldsymbol{w}^H$

$\widehat{\boldsymbol{v}}_1 = \mathsf{A}\,\boldsymbol{v} - \boldsymbol{v}\,\alpha_0$

$\beta_1 = \boldsymbol{w}^H\mathsf{A}^{*2}\,\boldsymbol{v} - \alpha_0^{*2}$

If β_1 is not $*$-invertible, then stop, otherwise

$\boldsymbol{v}_1 = \widehat{\boldsymbol{v}}_1 * \beta_1^{*-1}$

For $n = 2, \ldots$

$\alpha_{n-1} = \boldsymbol{w}_{n-1}^H * \mathsf{A} * \boldsymbol{v}_{n-1}$

$\boldsymbol{w}_n^H = \boldsymbol{w}_{n-1}^H * \mathsf{A} - \alpha_{n-1} * \boldsymbol{w}_{n-1}^H - \beta_{n-1} * \boldsymbol{w}_{n-2}^H$

$\widehat{\boldsymbol{v}}_n = \mathsf{A} * \boldsymbol{v}_{n-1} - \boldsymbol{v}_{n-1} * \alpha_{n-1} - \boldsymbol{v}_{n-2}$

$\beta_n = \boldsymbol{w}_n^H * \mathsf{A} * \boldsymbol{v}_{n-1}$

If β_n is not $*$-invertible, then stop, otherwise

$\boldsymbol{v}_n = \widehat{\boldsymbol{v}}_n * \beta_n^{*-1}$

end.

Algorithm 1 : $*$-Lanczos algorithm.

Let us define the tridiagonal matrix

$$\mathsf{T}_n := \begin{bmatrix} \alpha_0 & 1_* & & \\ \beta_1 & \alpha_1 & \ddots & \\ & \ddots & \ddots & 1_* \\ & & \beta_{n-1} & \alpha_{n-1} \end{bmatrix}, \tag{2.12}$$

and the matrices $\mathsf{V}_n := [\boldsymbol{v}_0, \ldots, \boldsymbol{v}_{n-1}]$ and $\mathsf{W}_n := [\boldsymbol{w}_0, \ldots, \boldsymbol{w}_{n-1}]$. Then the three-term recurrences Eqs. (2.10) read, in matrix form,

$$\mathsf{A} * \mathsf{V}_n = \mathsf{V}_n * \mathsf{T}_n + (\boldsymbol{v}_n * \beta_n)\boldsymbol{e}_n^H, \tag{2.13}$$

$$\mathsf{W}_n^H * \mathsf{A} = \mathsf{T}_n * \mathsf{W}_n^H + \boldsymbol{e}_n\,\boldsymbol{w}_n^H. \tag{2.14}$$

Hence the tridiagonal matrix (2.12) can be expressed as

$$\mathsf{T}_n = \mathsf{W}_n^H * \mathsf{A} * \mathsf{V}_n.$$

The following property of T_n is fundamental for the time-ordered exponential approximation; its proof is given in Sect. 2.3.

Theorem 2.1 (Matching Moment Property) *Let A, $\boldsymbol{w}$, $\boldsymbol{v}$ and T_n be as described above, then*

$$\boldsymbol{w}^H (A^{*j})\, \boldsymbol{v} = \boldsymbol{e}_1^H (T_n^{*j})\, \boldsymbol{e}_1, \quad \textit{for} \quad j = 0, \ldots, 2n-1. \tag{2.15}$$

Consider the time-ordered exponential U_n given by the differential equation

$$\mathsf{T}_n(t', t)\mathsf{U}_n(t', t) = \frac{d}{dt'}\mathsf{U}_n(t', t), \quad \mathsf{U}_n(t, t) = \mathsf{Id}. \tag{2.16}$$

Theorem 2.1 and Eq. (2.6) justify the use of the approximation $\boldsymbol{w}^H \mathsf{U}(t', t)\, \boldsymbol{v} \approx \boldsymbol{e}_1^H \mathsf{U}_n(t', t)\, \boldsymbol{e}_1$ [17]. The system (2.16) can be seen as a reduced order model of the initial differential Eq. (1.1) from two points of view. First, n may be much smaller than the size of A; in this sense, in Sect. 3, we will discuss the convergence behavior of the approximation using Theorem 2.1. Secondly, looking at A and T_n as adjacency matrices, A may correspond to a graph with a complex structure, while T_n corresponds to a very simple graph composed of one path with possible self-loops. Then the path-sum method gives

$$\mathsf{R}_*(\mathsf{T}_n)_{1,1}(t', t) = \left(1_* - \alpha_0 - \left(1_* - \alpha_1 - (1_* - \ldots)^{*-1} * \beta_2\right)^{*-1} * \beta_1\right)^{*-1}, \tag{2.17}$$

see [26, 44]. This expression is analogous to the one for the first diagonal entry of the inverse of an ordinary tridiagonal matrix [45], except here all operations are taken with respect to the $*$-product.

For $n = N$, we get

$$\mathsf{V}_N * \mathsf{W}_N^H = \mathsf{W}_N^H * \mathsf{V}_N = \mathsf{Id}\, 1_*. \tag{2.18}$$

Theorem 2.2 *Let A, V_N, W_N and T_N be as described above, then*

$$A^{*j} = V_N * T_N^{*j} * W_N^H, \quad j = 0, 1, \ldots,$$

and thus

$$R_*(A) = V_N * R_*(T_N) * W_N^H.$$

The theorem follows by using (2.18). Here, any entry of $R_*(T_N)$ is computable using a path-sum continued fraction of depth at most N.

Remark 2.3 The Lanczos-like method presented here for the time-ordered exponential is immediately valid for the ordinary matrix exponential function, since the latter is obtained from the former in the situation where A commutes with itself at all times,

$$\mathcal{T} e^{\int \mathsf{A}(\tau)\, \mathrm{d}\tau} = e^{\int_t^{t'} \mathsf{A}(\tau)\, \mathrm{d}\tau}.$$

This situation includes the case where A is time-independent, in which case setting $t = 0$ and $t' = 1$ above yields the matrix exponential of A. However, the $*$-Lanczos algorithm cannot be considered a generalization of the Lanczos algorithm since its outputs on constant matrices are made of distributions and time dependent functions.

2.3 Matching $*$-moments through $*$-biorthonormal polynomials

In order to prove Theorem 2.1, we will exploit the connection between the $*$-Lanczos algorithm and families of $*$-biorthonormal polynomials. Let us define the set of $*$-polynomials

$$\mathcal{P}_* := \left\{ p(\lambda) = \sum_{j=0}^{k} \lambda^{*j} * \gamma_j(t', t) \right\},$$

with $\gamma_j(t', t) \in \mathrm{D}(I)$. Consider a $*$-sesquilinear form $[\cdot, \cdot] : \mathcal{P}_* \times \mathcal{P}_* \to \mathrm{D}(I)$, i.e., so that given $p_1, p_2, q_1, q_2 \in \mathcal{P}_*$ and $\alpha, \beta \in \mathrm{D}(I)$, it satisfies

$$[q_1 * \alpha, p_1 * \beta] = \bar{\alpha} * [q_1, p_1] * \beta,$$
$$[q_1 + q_2, p_1 + p_2] = [q_1, p_1] + [q_2, p_1] + [q_1, p_2] + [q_2, p_2].$$

From now on we assume that every considered $*$-sesquilinear form $[\cdot, \cdot]$ also satisfies

$$[\lambda * q, p] = [q, \lambda * p]. \tag{2.19}$$

The $*$-sesquilinear form $[\cdot, \cdot]$ is determined by its $*$-*moments* defined as

$$m_j(t, t') := [\lambda^{*j}, 1] = [1, \lambda^{*j}], \quad j = 0, 1, \ldots .$$

We aim to build sequences of $*$-polynomials $p_0, p_1, \ldots$ and $q_0, q_1, \ldots$ so that they are $*$-biorthonormal with respect to $[\cdot, \cdot]$, i.e.,

$$[q_i, p_j] = \delta_{ij} 1_*, \tag{2.20}$$

where the subindex j in p_j and q_j corresponds to the degree of the $*$-polynomial. Here and in the following we assume $m_0 = 1_*$, getting $p_0 = q_0 = 1_*$. Consider the $*$-polynomial

$$q_1(\lambda) = \lambda * q_0(\lambda) - q_0(\lambda) * \bar{\alpha}_0.$$

The orthogonality conditions (2.20) give $\alpha_0 = [\lambda * q_0, p_0]$. Similarly, we get the $*$-polynomial

$$p_1(\lambda) * \beta_1 = \lambda * p_0(\lambda) - p_0(\lambda) * \alpha_0,$$

with $\alpha_0 = [q_0, \lambda * p_0]$, $\beta_1 = [q_1, \lambda * p_0]$. Repeating the $*$-orthogonalization process, we obtain the three-term recurrences for $n = 1, 2, \ldots$

$$q_n(\lambda) = \lambda * q_{n-1}(\lambda) - q_{n-1}(\lambda) * \bar{\alpha}_{n-1} - q_{n-2}(\lambda) * \bar{\beta}_{n-1} \tag{2.21a}$$
$$p_n(\lambda) * \beta_n = \lambda * p_{n-1}(\lambda) - p_{n-1}(\lambda) * \alpha_{n-1} - p_{n-2}(\lambda), \tag{2.21b}$$

with $p_{-1} = q_{-1} = 0$ and

$$\alpha_{n-1} = [q_{n-1}, \lambda * p_{n-1}], \quad \beta_n = [q_n, \lambda * p_{n-1}]. \tag{2.22}$$

Note that deriving the recurrences needs property (2.19). The $*$-biorthonormal polynomials $p_0, \ldots, p_n$ and $q_0, \ldots, q_n$ exist under the assumption that $\beta_1, \ldots, \beta_n$ are $*$-invertible.

Let A be a time-dependent matrix, and $\boldsymbol{w}$, $\boldsymbol{v}$ time-independent vectors such that $\boldsymbol{w}^H \boldsymbol{v} = 1$. Consider the $*$-sesquilinear form $[\cdot, \cdot]$ defined by

$$[q, p] = \boldsymbol{w}^H q^D(\mathsf{A}) * p(\mathsf{A})\, \boldsymbol{v}.$$

Assume that there exist $*$-polynomials $p_0, \ldots, p_n$ and $q_0, \ldots, q_n$ $*$-biorthonormal with respect to $[\cdot, \cdot]$. Defining the vectors

$$\boldsymbol{v}_j = p_j(\mathsf{A})\, \boldsymbol{v}, \qquad \boldsymbol{w}_j^H = \boldsymbol{w}^H q_j^D(\mathsf{A}),$$

and using the recurrences (2.21) gives the $*$-Lanczos recurrences (2.10). Moreover, the coefficients in (2.22) are the $*$-Lanczos coefficients in (2.11).

Let T_n be a tridiagonal matrix as in (2.12) composed of the coefficients (2.22) associated with the $*$-sesquilinear form $[\cdot, \cdot]$. Then we can define the $*$-sesquilinear form

$$[q, p]_n = \boldsymbol{e}_1^H q^D(\mathsf{T}_n) * p(\mathsf{T}_n)\, \boldsymbol{e}_1.$$

The following lemmas show that

$$m_j = [\lambda^{*j}, 1_*] = [\lambda^{*j}, 1_*]_n, \quad j = 0, \dots, 2n-1,$$

proving Theorem 2.1.

Lemma 2.4 *Let $p_0, \dots, p_n$ and $q_0, \dots, q_n$ be $*$-biorthonormal polynomials with respect to the $*$-sesquilinear form $[\cdot,\cdot]$. Assume that the coefficients $\beta_1, \dots, \beta_n$ in the related recurrences (2.21) are $*$-invertible. Then the $*$-polynomials are also $*$-biorthonormal with respect to the form $[\cdot,\cdot]_n$ defined above.*

Proof Consider the vectors $\boldsymbol{y}_j^H = \boldsymbol{e}_1^H \mathsf{T}_n^{*j}$ and $\boldsymbol{x}_j = \mathsf{T}_n^{*j} \boldsymbol{e}_1$. Since the matrix T_n is tridiagonal, for $j = 1, \dots, n-1$, we have

$$\begin{aligned} \boldsymbol{e}_i^H \boldsymbol{x}_j = 0, \ \text{for } i \geq j+2, \quad \text{and} \quad \boldsymbol{e}_{j+1}^H \boldsymbol{x}_j = \beta_j * \cdots * \beta_1, \\ \boldsymbol{y}_j^H \boldsymbol{e}_i = 0, \ \text{for } i \geq j+2, \quad \text{and} \quad \boldsymbol{y}_j^H \boldsymbol{e}_{j+1} = 1_* . \end{aligned}$$

By assumption, the product $\beta_j * \cdots * \beta_1$ is $*$-invertible. Therefore there exist $*$-polynomials $\widehat{p}_0, \dots, \widehat{p}_{n-1}$ and $\widehat{q}_0, \dots, \widehat{q}_{n-1}$ so that, for $i = 0, \dots, n-1$, we get

$$1_* \boldsymbol{e}_{i+1}^H = \boldsymbol{e}_1^H \, \widehat{q}_i^D(\mathsf{T}_n), \qquad 1_* \boldsymbol{e}_{i+1} = \widehat{p}_i(\mathsf{T}_n) \, \boldsymbol{e}_1.$$

Such $*$-polynomials are $*$-biorthonormal with respect to $[\cdot,\cdot]_n$ since they satisfy

$$[\widehat{q}_i, \widehat{p}_j]_n = 1_* \boldsymbol{e}_{i+1}^H * 1_* \boldsymbol{e}_{j+1} = \delta_{ij} 1_*.$$

Moreover, for $i = 0, \dots, n-1$, the corresponding recurrence coefficients (2.22) are the same as the ones of the $*$-polynomials $p_0, \dots, p_{n-1}$ and $q_0, \dots, q_{n-1}$. Indeed,

$$\begin{aligned} \widehat{\alpha}_{i-1} = [\widehat{q}_{i-1}, \lambda * \widehat{p}_{i-1}]_n = \boldsymbol{e}_{i-1}^H \mathsf{T}_n \, \boldsymbol{e}_{i-1} = \alpha_{i-1}, \\ \widehat{\beta}_i = [\widehat{q}_i, \lambda * \widehat{p}_{i-1}]_n = \boldsymbol{e}_i^H \, \mathsf{T}_n \boldsymbol{e}_{i-1} = \beta_i. \end{aligned}$$

Since $\widehat{p}_0 = p_0 = \widehat{q}_0 = q_0 = 1_*$, we get $\widehat{p}_i = p_i$ and $\widehat{q}_i = q_i$ for $i = 0, \dots, n-1$. □

Lemma 2.5 *Let $p_0, \dots, p_{n-1}$ and $q_0, \dots, q_{n-1}$ be $*$-biorthonormal polynomials with respect to a $*$-sesquilinear form $[\cdot,\cdot]_A$ and to a $*$-sesquilinear form $[\cdot,\cdot]_B$. If $[1_*, 1_*]_A = [1_*, 1_*]_B = 1_*$, then $[\lambda^{*j}, 1_*]_A = [\lambda^{*j}, 1_*]_B$ for $j = 0, \dots, 2n-1$.*

Proof We prove it by induction. Let $m_j = [\lambda^{*j}, 1_*]_A$ and $\widehat{m}_j = [\lambda^{*j}, 1_*]_B$ for $j = 0, 1, \dots, 2n-1$. By the expression for the coefficients in (2.22) we get

$$[q_0, \lambda * p_0]_A = \alpha_0 = [q_0, \lambda * p_0]_B.$$

Hence $m_1 = \alpha_0 = \widehat{m}_1$. Assuming $m_j = \widehat{m}_j$ for $j = 0, \dots, 2k-3$ we will prove that $m_{2k-2} = \widehat{m}_{2k-2}$ and $m_{2k-1} = \widehat{m}_{2k-1}$, for $k = 2, \dots, n$. The coefficient expressions in (2.22) gives

$$[q_{k-1}, \lambda * p_{k-2}]_A = \beta_{k-1} = [q_{k-1}, \lambda * p_{k-2}]_B,$$

which can be rewritten as

$$\sum_{i=0}^{k-1} \sum_{j=0}^{k-2} \bar{a}_i * m_{i+j+1} * b_j = \sum_{i=0}^{k-1} \sum_{j=0}^{k-2} \bar{a}_i * \widehat{m}_{i+j+1} * b_j,$$

with a_i, b_j the coefficients respectively of q_{k-1} and p_{k-2}. The inductive assumption implies

$$\bar{a}_{k-1} * m_{2k-2} * b_{k-2} = \bar{a}_{k-1} * \widehat{m}_{2k-2} * b_{k-2}.$$

The leading coefficients of the $*$-polynomials q_{2k-2} and p_{2k-2} are respectively $a_{k-1} = 1_*$ and $b_{k-2} = (\beta_{k-2} * \cdots * \beta_1)^{*-1}$. Hence $m_{2k-2} = \widehat{m}_{2k-2}$. Repeating the same argument with the coefficient α_{k-1} (2.22) concludes the proof. □

3 Convergence, breakdown, and related properties

3.1 The convergence behavior of intermediate approximations

Assuming no breakdown, the $*$-Lanczos algorithm in conjunction with the path-sum method converges to the solution $\boldsymbol{w}^H \mathsf{U}(t', t)\boldsymbol{v}$ in N iterations, with N the size of A; see Theorem 2.2. Most importantly, intermediate $*$-Lanczos iterations provide a sequence of approximations $\int_t^{t'} \mathsf{R}_*(\mathsf{T}_n)_{1,1}(\tau, t)\,\mathrm{d}\tau$, $n = 1, \ldots, N$, whose convergence behavior we analyze hereafter.

As we have already discussed before, under the assumption that all entries of A are smooth over I and that all the $\beta_j^{(1,0)}(t, t) \neq 0$, for every $t \in I$, all the α_j and β_j distributions are elements of $\mathrm{Sm}_\Theta(I)$. The proof of this statement is very long and technical and serves only to establish the theoretical *feasibility* of tridigonalization for systems of coupled linear differential equations with variable coefficients using smooth functions. It was therefore presented in a separate work. We refer the reader to [17] for a full exposition and proof, while here we only state some of the main results:

Theorem 3.1 [17] *Let $A(t', t) = \widetilde{A}(t')\Theta(t' - t)$ be an $N \times N$ matrix composed of elements from $Sm_\Theta(I)$. Let α_{n-1} and β_n be the coefficients generated by Algorithm 1 running on A and the time-independent vectors $\boldsymbol{w}$, $\boldsymbol{v}$ ($\boldsymbol{w}^H\boldsymbol{v} = 1$). Let*

$$\beta_j^{(1,0)}(t', t) := \frac{\partial}{\partial t}\beta_j(t', t), \quad t', t \in I.$$

For any $1 \leq n \leq N$, assuming that $\beta_j^{(1,0)}(t, t) \neq 0$ for every $t \in I$, $j = 1, \ldots, n-1$, we get $\beta_n(t', t), \alpha_{n-1}(t', t) \in Sm_\Theta(I)$. In addition, all required $*$-inverses β_j^{*-1} exist and are of the form

$$\beta_j^{*-1} = b_j^L(t', t) * \delta^{(3)}(t' - t),$$

with $b_j^L \in Sm_\Theta(I)$.

All b_j^L have explicit expansions in terms of β_j which are given in [17] but not reproduced here owing to length concerns. When $\beta_n \not\equiv 0$, but $\beta_n^{(1,0)}(\rho, \rho) = 0$ for some $\rho \in I$, the results of Theorem 3.1 hold if we restrict the intial interval I to a subinteval $J \subset I$ so that $\rho \notin J$.

Thanks to these regularity results, we can establish the following bound for the approximation error:

Proposition 3.2 *Let us consider the setting and assumptions of* Theorem 3.1. *Moreover, let U designate the time-ordered exponential of A and let T_n be the tridiagonal matrix (2.12) such that*

$$\boldsymbol{w}^H A^{*j}\boldsymbol{v} = (T_n^{*j})_{1,1}, \quad \text{for} \quad j = 0, \ldots, 2n-1.$$

Then, for $t' \geq t$, with $t', t \in I$,

$$\left| \boldsymbol{w}^H U(t', t)\boldsymbol{v} - \int_t^{t'} R_*(T_n)_{1,1}(\tau, t)\,d\tau \right| \leq \frac{C^{2n} + D_n^{2n}}{(2n)!}(t' - t)^{2n} e^{(C+D_n)(t'-t)}.$$

Here

$$C := \sup_{t' \in I} \|A(t')\|_\infty, \quad D_n := 3 \sup_{t',t \in I^2} \max_{0 \le j \le n-1} \left\{ |\alpha_j(t',t)|, |\beta_j(t',t)| \right\}$$

are both finite, with $\|\cdot\|_\infty$ *the matrix norm induced by the uniform norm.*

Proof Assume $t' \ge t$. Observe that

$$\boldsymbol{w}^H \mathsf{U}(t',t)\boldsymbol{v} - \int_t^{t'} \mathsf{R}_*(\mathsf{T}_n)(\tau,t)_{11}\,\mathrm{d}\tau = \int_t^{t'} \sum_{j=2n}^{\infty} \boldsymbol{w}^H \mathsf{A}^{*j}(\tau,t)\boldsymbol{v} - (\mathsf{T}_n^{*j})_{1,1}(\tau,t)\,\mathrm{d}\tau,$$

so that

$$\left| \boldsymbol{w}^H \mathsf{U}(t',t)\boldsymbol{v} - \int_t^{t'} \mathsf{R}_*(\mathsf{T}_n)(\tau,t)_{11}\,\mathrm{d}\tau \right| \le \int_t^{t'} \sum_{j=2n}^{\infty} \left| \boldsymbol{w}^H \mathsf{A}^{*j}(\tau,t)\boldsymbol{v} \right| + \left| (\mathsf{T}_n^{*j})_{1,1}(\tau,t) \right| \mathrm{d}\tau.$$

Now $\sup_{t' \in I} |\boldsymbol{w}^H \mathsf{A}(t')\boldsymbol{v}| \le C$ and

$$\left| \int_t^{t'} \boldsymbol{w}^H \mathsf{A}^{*j}(\tau,t)\boldsymbol{v}\,\mathrm{d}\tau \right| \le \Theta(t'-t) * \left(C\Theta(t'-t) \right)^{*j} = C^j \frac{(t'-t)^j}{j!}.$$

We proceed similarly for the terms involving T_n. Theorem 3.1 implies the existence of $\widehat{D}_n := \sup_{t',t \in I^2} \max_{0 \le j \le n-1} \left\{ |\alpha_j(t',t)|, |\beta_j(t',t)| \right\} < +\infty$. The matrix element $(\mathsf{T}_n^{*j})_{1,1}$ is given by the sum of $*$-products of coefficients α_i, β_i and 1_*. Replacing all the factors in those $*$-products with $\widehat{D}_n \Theta(t'-t)$ gives an upper bound for $(\mathsf{T}_n^{*j})_{1,1}$. Hence we get

$$\left| \left(\mathsf{T}_n^{*j} \right)_{1,1} \right| \le \left(\left(\widehat{D}_n \mathsf{P}_n \Theta(t'-t) \right)^{*j} \right)_{1,1} \le \widehat{D}_n^j \|\mathsf{P}_n\|_\infty^j \Theta(t'-t)^{*j},$$

where P_n is the $n \times n$ tridiagonal matrix whose nonzero entries are equal to 1. Note that $\|\mathsf{P}_n\|_\infty = 3$. Hence the error can be bounded by

$$\begin{aligned}
\sum_{j=2n}^{\infty} \left(C^j + D_n^j \right) \frac{(t'-t)^j}{j!} &\le \frac{(C^{2n} + D_n^{2n})}{2n!} (t'-t)^{2n} \sum_{j=0}^{\infty} \frac{2n!}{(2n+j)!} \left(C^j + D_n^j \right) (t'-t)^j, \\
&\le \frac{(C^{2n} + D_n^{2n})}{2n!} (t'-t)^{2n} \sum_{j=0}^{\infty} \frac{(C+D_n)^j (t'-t)^j}{j!}, \\
&\le \frac{(C^{2n} + D_n^{2n})}{2n!} (t'-t)^{2n} e^{(C+D_n)(t'-t)},
\end{aligned}$$

concluding the proof. □

Analogously to the classical non-Hermitian Lanczos algorithm, we need to further assume D_n to be not too large for $n \ge 1$ in order to get a meaningful bound. Such an assumption can be verified a-posteriori. The bound of Proposition 3.2 demonstrates that under reasonable assumptions, the approximation error has a super-linear decay. Assuming no breakdown and exact precision arithmetic, we also recall that the algorithm does necessarily converge in at most N steps, independently of the error bound. The computational cost of the algorithm is discussed separately in Sect. 5.

3.2 Breakdown

In the classical non-Hermitian Lanczos algorithm, a breakdown appears either when an invariant Krylov subspace is produced (*lucky breakdown*) or when the last vectors of the biorthogonal bases $\boldsymbol{v}_n, \boldsymbol{w}_n$ are nonzero, but $\boldsymbol{w}_n^H \boldsymbol{v}_n = 0$ (*serious breakdown*); for further details refer, e.g., to [33, 34, 39, 46–48]. Analogously, in the $*$-Lanczos algorithm 1, a *lucky breakdown* arises when either $\boldsymbol{w}_n \equiv 0$ or $\widehat{\boldsymbol{v}}_n \equiv 0$. In such a case, the algorithm has converged to the solution, as the following proposition shows.

Proposition 3.3 *Assume that the $*$-Lanczos algorithm 1 does not breakdown until the nth step when a lucky breakdown arises, i.e., $\widehat{\boldsymbol{v}}_n \equiv 0$ (or $\boldsymbol{w}_n \equiv 0$). Then*

$$\boldsymbol{w}^H (A^{*j})\, \boldsymbol{v} = \boldsymbol{e}_1^H (T_n^{*j})\, \boldsymbol{e}_1, \quad \text{for } j \geq 0,$$
$$\boldsymbol{w}^H U(t', t) \boldsymbol{v} = \int_0^t R_*(T_n)_{1,1}(\tau, t)\, d\tau.$$

Proof Assuming $\widehat{\boldsymbol{v}}_n \equiv 0$, eq. (2.13) becomes

$$\mathsf{A} * \mathsf{V}_n = \mathsf{V}_n * \mathsf{T}_n. \tag{3.1}$$

By $*$-multiplying (3.1) from the left by A, we get

$$\mathsf{A} * \mathsf{A} * \mathsf{V}_n = \mathsf{A} * \mathsf{V}_n * \mathsf{T}_n,$$

which becomes

$$\mathsf{A}^{*2} * \mathsf{V}_n = \mathsf{V}_n * \mathsf{T}_n^{*2},$$

using (3.1). Repeating the argument gives

$$\mathsf{A}^{*j} * \mathsf{V}_n = \mathsf{V}_n * \mathsf{T}_n^{*j}, \quad j \geq 0,$$

from which we get that $\mathsf{W}_n^H * \mathsf{A}^{*j} * \mathsf{V}_n = \mathsf{T}_n^{*j}$, $j \geq 0$. The proposition easily follows from here. An analogous proof holds for the case $\widehat{\boldsymbol{w}}_n \equiv 0$.

□

The $*$-Lanczos algorithm construction and its polynomial interpretation in Sect. 2.3 suggest that it may be possible to deal with the serious breakdown issue by a look-ahead strategy analogous to the one for the non-Hermitian Lanczos algorithm; see, e.g., [33, 34]. If $i \neq j$, then $\boldsymbol{w}^H \boldsymbol{v} = 0$, which does not satisfy the $*$-Lanczos assumption. Moreover, if $i = j$ and A is a sparse non-Hermitian matrix, then it may be possible that $\mathsf{A}_{ii} \equiv 0$ and $\mathsf{A}^{*2_{ii}} \equiv 0$. As a consequence, we get $\beta_1 \equiv 0$. We can try to fix these problems rewriting the approximation of the time-ordered exponential U as

$$\boldsymbol{e}_i^H \mathsf{U} \boldsymbol{e}_j = (\boldsymbol{e} + \boldsymbol{e}_i)^H \mathsf{U} \boldsymbol{e}_j - \boldsymbol{e}^H \mathsf{U} \boldsymbol{e}_j,$$

with $\boldsymbol{e} = (1, \ldots, 1)^H$. Then one can approximate $(\boldsymbol{e} + \boldsymbol{e}_i)^H \mathsf{U}\, \boldsymbol{e}_j$ and $\boldsymbol{e}^H \mathsf{U}\, \boldsymbol{e}_j$ separately, which are less likely going to have a breakdown, thanks to the fact that $\boldsymbol{e}$ is a full vector; see, e.g., [35, Section 7.3]. Note that while this strategy can prevent a breakdown in the first iterations, a serious breakdown might still happen in later ones.

4 Examples

In this section, we use the $*$-Lanczos algorithm 1 on examples in ascending order of difficulty. All the computations have been performed using MATHEMATICA 11.

Example 4.1 (Ordinary matrix exponential) Let us first consider a constant matrix

$$\mathsf{A} = \begin{pmatrix} -1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & -1 \end{pmatrix}.$$

Because A commutes with itself at all times, its time-ordered exponential coincides with its ordinary exponential, $\mathcal{T} e^{\int \mathsf{A}(\tau)\,\mathrm{d}\tau} \equiv e^{\mathsf{A}(t')}$ (we set $t = 0$). Note that the matrix chosen here is symmetric only to lead to concise expressions suitable for presentation in an article, e.g.,

$$\left(e^{\mathsf{A}t'}\right)_{11} = -\frac{1}{2}\sinh(2t') + \frac{1}{2}\cosh(2t') + \frac{1}{2}\cosh\left(\sqrt{2}t'\right), \tag{4.1}$$

and that such symmetries are not a requirement of the $*$-Lanczos approach.

Now let us find the result of Eq. (4.1) with Algorithm 1. We define $\boldsymbol{w} := \boldsymbol{v}^H := (1, 0, 0)$, $\boldsymbol{w}_0 = \boldsymbol{w}1_*$, $\boldsymbol{v}_0 = \boldsymbol{v}1_*$, from which it follows that $\alpha_0(t', t) = -1 \times \Theta(t' - t)$ and $\boldsymbol{w}_1 = \widehat{\boldsymbol{v}}_1^H = (0, 1, 1)\Theta(t' - t)$. Furthermore, since A is a constant matrix times $\Theta(t' - t)$, we have $\mathsf{A}^{*n} = \tilde{\mathsf{A}}^n \times \Theta(t' - t)^{*n} = \tilde{\mathsf{A}} \times (t' - t)^{n-1}/(n-1)! \times \Theta(t' - t)$ and similarly $\alpha_0^{*2}(t', t) = \tilde{\alpha}_0^2 \times (t' - t)\Theta(t' - t)$. Thus

$$\beta_1 = \boldsymbol{w}^H \mathsf{A}^2 \boldsymbol{v} \times (t' - t)\Theta(t' - t) - \tilde{\alpha}_0^2 (t' - t)\Theta(t' - t) = 2(t' - t)\Theta(t' - t).$$

The $*$-inverse follows as $\beta^{*-1} = \frac{1}{2}\delta''$ [18], from which we get

$$\boldsymbol{v}_1 = \widehat{\boldsymbol{v}}_1 * \beta_1^{*-1} = (0, 1, 1)^H \frac{1}{2}\delta'(t' - t),$$

Now it follows that

$$\begin{aligned}
\alpha_1(t', t) &= \boldsymbol{w}_1 * \mathsf{A} * \boldsymbol{v}_1 = \frac{1}{2}\Theta(t - t'), \\
\boldsymbol{w}_2(t', t) &= \boldsymbol{w}_1 * \mathsf{A} - \alpha_1 * \boldsymbol{w}_1 - \beta_1 * \boldsymbol{w}_0 = (0, 1, -1)\frac{1}{2}(t' - t)\Theta(t' - t), \\
\widehat{\boldsymbol{v}}_2(t', t) &= \mathsf{A} * \boldsymbol{v}_1 - \boldsymbol{v}_1 * \alpha_1 - \boldsymbol{v}_0 = (0, 1, -1)^H \frac{1}{4}\delta(t' - t), \\
\beta_2 &= \boldsymbol{w}_2 * \mathsf{A} * \boldsymbol{v}_1 = \frac{1}{4}(t' - t)\Theta(t' - t).
\end{aligned}$$

Then $\beta_2^{*-1} = 4\delta''$ and so

$$\boldsymbol{v}_2 = \widehat{\boldsymbol{v}}_2 * \beta_2^{*-1} = (0, 1, -1)^H \delta''(t' - t)\alpha_2 \qquad = \boldsymbol{w}_2 * \mathsf{A} * \boldsymbol{v}_2 = -\frac{3}{2}\Theta(t' - t).$$

At this point we have determined the $*$-Lanczos matrices T, V and W entirely

$$\mathsf{T} = \begin{pmatrix} -\Theta & \delta & 0 \\ 2\Theta^{*2} & \frac{1}{2}\Theta & \delta \\ 0 & \frac{1}{4}\Theta^{*2} & -\frac{3}{2}\Theta \end{pmatrix}, \quad \mathsf{V} = \begin{pmatrix} \delta & 0 & 0 \\ 0 & \frac{1}{2}\delta' & \delta'' \\ 0 & \frac{1}{2}\delta' & -\delta'' \end{pmatrix}, \quad \mathsf{W}^H = \begin{pmatrix} \delta & 0 & 0 \\ 0 & \Theta & \Theta \\ 0 & \frac{1}{2}\Theta^{*2} & -\frac{1}{2}\Theta^{*2} \end{pmatrix}.$$

In all of these expressions, Θ is a short-hand notation for $\Theta(t' - t)$ and δ, δ' and δ'' are to be evaluated in $t' - t$. It is now straightforward to verify the matching moment property

$\left(\mathsf{T}^{*j}\right)_{11} = \left(\mathsf{A}^{*j}\right)_{11}$ for all $j \in \mathbb{N}$. We can also check directly that the time-ordered exponential of A is correctly determined from T using either the general formula of Eq. (2.17) or, because the situation is so simple that all entries depend only on $t' - t$, we may use a Laplace transform with respect to $t' - t$. This gives $\mathsf{T}(s)$, and the inverse Laplace-transform of the resolvent $\left(\mathsf{I} - \mathsf{T}(s)\right)^{-1}_{11}$ is the desired quantity. Both procedures give the same result, namely the derivative of $e^{\mathsf{A}t}$ as it should [26], i.e.,

$$(\mathsf{Id}_* - \mathsf{T})^{*-1}_{11}(t', 0) = \left(\sinh(2t') + \frac{1}{\sqrt{2}} \sinh\left(\sqrt{2}t'\right) - \cosh(2t')\right) \Theta(t'),$$

which is indeed the derivative of Eq. (4.1).

Example 4.2 (Time-ordered exponential of a time-dependent matrix) In this example, we consider the 5×5 time-dependent matrix $\mathsf{A}(t', t) = \tilde{\mathsf{A}}(t')\Theta(t' - t)$ with

$$\tilde{\mathsf{A}}(t') = \begin{pmatrix} \cos(t') & 0 & 1 & 2 & 1 \\ 0 & \cos(t') - t' & 1 - 3t' & t' & 0 \\ 0 & t' & 2t' + \cos(t') & 0 & 0 \\ 0 & 1 & 2t' + 1 & t' + \cos(t') & t' \\ t' & -t' - 1 & -6t' - 1 & 1 - 2t' & \cos(t') - 2t' \end{pmatrix}.$$

The matrix $\tilde{\mathsf{A}}$ does not commute with itself at different times $\tilde{\mathsf{A}}(t')\tilde{\mathsf{A}}(t) - \tilde{\mathsf{A}}(t)\tilde{\mathsf{A}}(t') \neq 0$, and the corresponding differential system Eq. (1.1) has no known analytical solution. We use Algorithm 1 to determine the tridiagonal matching moment matrix T such that $\left(\mathsf{A}^{*j}\right)_{11} = \left(\mathsf{T}^{*j}\right)_{11}$ for $j \in \mathbb{N}$. We define $\boldsymbol{w} := \boldsymbol{v}^H := (1, 0, 0, 0, 0)$, $\boldsymbol{w}_0 = \boldsymbol{w}1_*$, $\boldsymbol{v}_0 = \boldsymbol{v}1_*$, from which it follows that

$$\begin{aligned} \alpha_0(t', t) &= \cos(t')\Theta(t' - t), \\ \boldsymbol{w}_1 &= \big(0, 0, 1, 2, 1\big)\Theta(t' - t), \\ \widehat{\boldsymbol{v}}_1 &= \big(0, 0, 0, 0, t'\big)^H \Theta(t' - t), \\ \beta_1(t', t) &= \frac{1}{2}\left(t'^2 - t^2\right)\Theta(t' - t). \end{aligned}$$

Observing that $\beta_1 = \Theta(t' - t) * t'\Theta(t' - t)$, we get $\beta_1^{*-1} = \frac{1}{t}\delta'(t' - t) * \delta'(t' - t) = -\frac{1}{t^2}\delta'(t' - t) + \frac{1}{t}\delta''(t' - t)$, so that

$$\boldsymbol{v}_1 = \widehat{\boldsymbol{v}}_1 * \beta_1^{*-1} = \big(0,\ 0,\ 0,\ 0,\ 1\big)^H \delta'(t' - t),$$

which terminates the initialization phase of the Algorithm. We proceed with

$$\begin{aligned} \alpha_1(t', t) &= \boldsymbol{w}_1 * \mathsf{A} * \boldsymbol{v}_1 = \cos(t)\Theta(t' - t), \\ \boldsymbol{w}_2 &= \boldsymbol{w}_1 * \mathsf{A} - \alpha_1 * \boldsymbol{w}_1 - \beta_1 * \boldsymbol{w}_0, \\ &= \big(0,\ t' - t,\ t' - t,\ t' - t,\ 0\big)\Theta(t' - t), \\ \widehat{\boldsymbol{v}}_2 &= \mathsf{A} * \boldsymbol{v}_1 - \boldsymbol{v}_1 * \alpha_1 - \boldsymbol{v}_0 = \big(0,\ 0,\ 0,\ t,\ -2t\big)^H \delta(t' - t), \\ \beta_2 &= \boldsymbol{w}_2 * \mathsf{A} * \boldsymbol{v}_1 = t(t' - t)\Theta(t' - t). \end{aligned}$$

As we did for β_1, we factorize $\beta_2 = \Theta(t' - t) * t\,\Theta(t' - t)$ so that its $*$-inverse is $\beta_2^{*-1} = \frac{1}{t'}\delta'(t' - t) * \delta'(t' - t) = \frac{1}{t'}\delta''$. Then

$$\boldsymbol{v}_2 = \big(0,\ 0,\ 0,\ 1,\ -2\big)^H \delta''(t' - t).$$

Continuing in this fashion yields the tridiagonal output matrix $\mathsf{T}_5 \equiv \mathsf{T}$,

$$\mathsf{T}=\begin{pmatrix} \cos(t')\Theta & \delta & 0 & 0 & 0 \\ \frac{1}{2}(t'^2-t^2)\Theta & \cos(t)\Theta & \delta & 0 & 0 \\ 0 & t(t'-t)\Theta & \widetilde{\alpha}_2(t',t)\Theta & \delta & 0 \\ 0 & 0 & -\frac{1}{2}(3t^2-4tt'+t'^2)\Theta & \widetilde{\alpha}_3(t',t)\Theta & \delta \\ 0 & 0 & 0 & (-2t^2+3tt'-t'^2)\Theta & \widetilde{\alpha}_4(t',t)\Theta \end{pmatrix},$$

with

$$\begin{aligned} \widetilde{\alpha}_2(t',t) &= (t'-t)\sin(t)+\cos(t), \\ \widetilde{\alpha}_3(t',t) &= \frac{1}{2}\left(4(t'-t)\sin(t)-\left((t-t')^2-2\right)\cos(t)\right), \\ \widetilde{\alpha}_4(t',t) &= \frac{1}{6}\left(\left((t-t')^2-18\right)(t-t')\sin(t)+\left(6-9(t-t')^2\right)\cos(t)\right), \end{aligned}$$

and the bases matrices

$$\mathsf{V}_5 = \begin{pmatrix} \delta & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \delta^{(3)} & -2\delta^{(4)} \\ 0 & 0 & 0 & 0 & \delta^{(4)} \\ 0 & 0 & \delta'' & -\delta^{(3)} & \delta^{(4)} \\ 0 & \delta' & -2\delta'' & 2\delta^{(3)} & -3\delta^{(4)} \end{pmatrix}, \quad \mathsf{W}_5^H = \begin{pmatrix} \delta & 0 & 0 & 0 & 0 \\ 0 & 0 & \Theta & 2\Theta & \Theta \\ 0 & \Theta^{*2} & \Theta^{*2} & \Theta^{*2} & 0 \\ 0 & \Theta^{*3} & 2\Theta^{*3} & 0 & 0 \\ 0 & 0 & \Theta^{*4} & 0 & 0 \end{pmatrix}.$$

In all of these expressions, Θ and $\delta^{(n)}$ are short-hand notations respectively for $\Theta(t'-t)$ and $\delta^{(n)}(t'-t)$. All the required β_j^{*-1} were calculated using the strategies described in [18], getting the factorized $*$-inverses

$$\beta_3^{*-1} = \frac{1}{t}\Theta(t'-t) * \delta^{(3)}(t'-t), \quad \beta_4^{*-1} = \frac{t'}{t^2}\Theta(t'-t) * \delta^{(3)}(t'-t).$$

We have also verified that $\left(\mathsf{A}^{*j}\right)_{11} = \left(\mathsf{T}^{*j}\right)_{11}$ holds for j up to 9. The $*$-resolvent of T has no closed-form expression, its Neumann series likely converging to a hitherto undefined special function. Ultimately, such difficulties are connected with the propensity of systems of coupled linear ordinary differential equations with non-constant coefficients to produce transcendent solutions.

5 Outlook: numerical implementation

We do not expect closed-forms to exist in most cases for the entries of time-ordered matrix exponentials as these can involve complicated special functions [2]. Also, very large matrices $\mathsf{A}(t')$ are to be treatable by the algorithm for it to be relevant to most applications. For these reasons, it is fundamental to implement the $*$-Lanczos algorithm numerically, e.g., using time discretization approximations.

As shown in [26], there exists an isometry Φ between the algebra of distributions of $\mathrm{D}(I)$ equipped with the $*$-product and the algebra of *time-continuous* operators (for which the time variables t' and t serve as line and row indices). Consider, for simplicity, a discretization of the interval I with constant time step Δt; then these operators become ordinary matrices. Specifically, given $f, g \in \mathrm{Sm}_\Theta(I)$, their discretization counterparts are the *lower triangular* matrices F, G. Moreover, the function $f * g$ corresponds to the matrix $\mathsf{F}\,\mathsf{G}\,\Delta t$, with the usual matrix product. In other terms, the isometry Φ followed by a time discretization sends the

$*$-product to the ordinary matrix product times Δt. Similarly, the Dirac delta distribution is sent to the identity matrix times $1/(\Delta t)$, the kth Dirac delta derivative $\delta^{(k)}$ is sent to the finite difference matrix

$$\left(\mathsf{M}_{\delta^{(k)}}\right)_{ij} = \frac{1}{(\Delta t)^{k+1}} \begin{cases} (-1)^{i-j}\binom{k}{i-j}, & \text{if } i \geq j \\ 0, & \text{else} \end{cases},$$

and Θ is sent to the matrix $(\mathsf{M}_\Theta)_{ij} = 1$ if $i \geq j$ and 0 otherwise. Most importantly, in this picture, the $*$-inverse of a function $f(t', t) \in D(I)$ is given as $\mathsf{F}^{-1}/(\Delta t)^2$, with F the lower triangular matrix corresponding to f. Moreover, the time-discretized version of the path-sum formulation Eq. (2.17) only involves ordinary matrix resolvents. At the same time, the final integration of $\mathsf{R}_*(\mathsf{T}_n)_{11}$ yielding $\boldsymbol{w}^H \mathsf{U}\, \boldsymbol{v}$ becomes a left multiplication by M_Θ. Therefore, a numerical implementation of the time-discretized $*$-Lanczos algorithm only requires *ordinary operations on triangular matrices*.

We can now meaningfully evaluate the numerical cost of the time-discretized version of the algorithm. Let N_t be the number of time subintervals in the discretization of I for both the t and t' time variables. Then time-discretized $*$-multiplications or $*$-inversions cost $O(N_t^3)$ operations. Considering a sparse time-dependent matrix $\mathsf{A}(t)$ with N_{nz} nonzero elements, the $*$-Lanczos algorithm therefore necessitates $O(N_i \times N_t^3 \times N_{nz})$ operations to obtain the desired $\boldsymbol{w}^H \mathsf{U}\, \boldsymbol{v}$. Here N_i is the number of iterations needed to get an error lower than a given tolerance. Unfortunately, as well-explained in [36], the presence of computational errors can slow down the (usual) Lanczos algorithm convergence. Hence, in general, we cannot assume $N_i \approx N$ since the $*$-Lanczos algorithm could analogously require more iterations. However, in many cases, the (usual) Lanczos algorithm demands *few* iterations to reach the tolerance also in finite precision arithmetic. We expect the $*$-Lanczos algorithm to behave analogously, giving $N_i \ll N$ in many cases. Concerning N_t, there is no reason to expect that is would depend on N since N_t controls the quality of individual generalized functions. We also remark that the $*$-Lanczos algorithm can exploit the sparsity structure of the matrix A, making it inherently competitive when dealing with large sparse matrices that are typical of applications.

The classical numerical methods (e.g., Runge–Kutta methods) for the approximation of the system of ODEs (1.1) are known to perform poorly in certain cases. These include for example, very large system sizes, or in the presence of highly oscillatory coefficients. Consequently, in the last decades, novel techniques have been sought and proposed, many of which are based on the Magnus series; see, for instance, [4, 23, 49–56]. However, for large matrices, these methods are known to be highly consuming in resources. This motivates the research of novel approaches in particular for large-scale problems. Here the guaranteed convergence of the $*$-Lanczos algorithm in a finite number of iterations, the sequence of approximations it produces, its ability to exploit matrix sparsity and its relations with numerically well studied Lanczos procedures are all promising elements which justify further works on concrete numerical implementations. More precise theoretical results about the overall approximation quality and further issues on numerical applications of the present algorithm are beyond the scope of this work. They will appear together with a practical implementation of the algorithm in a future contribution.

6 Conclusion

In this work, we constructed the $*$-Lanczos algorithm as a biorthogonalization process of Krylov subspaces composed of distributions with respect to the $*$-product, a convolution-like product. The algorithm relies on a non-commutative operation and is analogous in spirit to the non-Hermitian Lanczos algorithm. The $*$-Lanczos algorithm can express the element of a time-ordered exponential of size $N \times N$ by the path-sum continued fraction (2.17). To our knowledge, such an expression is the only one composed of $\mathcal{O}(N)$ scalar integro-differential equations. Such a time-ordered exponential approximation relies on the matching moment property proved in this paper.

The overall approach generates a controllable sequence of time-ordered exponential approximations, offers an innovative perspective of the connection between numerical linear algebra and differential calculus, and opens the door to efficient numerical algorithms for large-scale computations.

Acknowledgements We thank Francesca Arrigo, Des Higham, Jennifer Pestana, and Francesco Tudisco for their invitation to the University of Strathclyde, without which this work would not have come to fruition. The first author was supported in part by 2019 Alcohol project ANR-19-CE40-0006 and 2020 Magica project ANR-20-CE29-0007. The second author was supported by Charles University Research programs No. PRIMUS/21/SCI/009 and UNCE/SCI/023.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Dyson, F.J.: Divergence of perturbation theory in quantum electrodynamics. Phys. Rev. **85**(4), 631–632 (1952). https://doi.org/10.1103/PhysRev.85.631
2. Xie, Q., Hai, W.: Analytical results for a monochromatically driven two-level system. Phys. Rev. A **82**, 032117 (2010)
3. Hortaçsu, M.: Heun functions and some of their applications in physics. Adv. High Energy Phys. **2018**, 8621573 (2018)
4. Blanes, S., Casas, F., Oteo, J.A., Ros, J.: The Magnus expansion and some of its applications. Phys. Rep. **470**(5), 151–238 (2009)
5. Autler, S.H., Townes, C.H.: Stark effect in rapidly varying fields. Phys. Rev. **100**, 703–722 (1955)
6. Shirley, J.H.: Solution of the Schrödinger equation with a Hamiltonian periodic in time. Phys. Rev. **138**, 979–987 (1965)
7. Lauder, M.A., Knight, P.L., Greenland, P.T.: Pulse-shape effects in intense-field laser excitation of atoms. Opt. Acta **33**(10), 1231–1252 (1986)
8. Reid, W.T.: Riccati matrix differential equations and non-oscillation criteria for associated linear differential systems. Pac. J. Math. **13**(2), 665–685 (1963)
9. Kwakernaak, H., Sivan, R.: Linear Optimal Control Systems, vol. 1. Wiley-interscience, New York (1972)
10. Corless, M., Frazho, A.: Linear Systems and Control: An Operator Perspective. Pure and Applied Mathematics. Marcel Dekker, New York (2003)
11. Blanes, S.: High order structure preserving explicit methods for solving linear-quadratic optimal control problems. Numer. Algor. **69**(2), 271–290 (2015)
12. Benner, P., Cohen, A., Ohlberger, M., Willcox, K.: Model Reduction and Approximation: Theory and Algorithms. Computational Science and Engineering. SIAM, Philadelphia (2017)
13. Kučera, V.: A review of the matrix Riccati equation. Kybernetika **9**(1), 42–61 (1973)
14. Abou-Kandil, H., Freiling, G., Ionescu, V., Jank, G.: Matrix Riccati Equations in Control and Systems. Theory Systems & Control: Foundations & Applications. Birkhäuser, Basel (2003)
15. Hached, M., Jbilou, K.: Numerical solutions to large-scale differential Lyapunov matrix equations. Numer. Algor. **79**(3), 741–757 (2018)

16. Kirsten, G., Simoncini, V.: Order reduction methods for solving large-scale differential matrix Riccati equations. SIAM J. Sci. Comput. **42**(4), 2182–2205 (2020). https://doi.org/10.1137/19m1264217
17. Giscard, P.-L., Pozza, S.: Tridiagonalization of systems of coupled linear differential equations with variable coefficients by a Lanczos-like method. Linear Algebra Appl. **624**, 153–173 (2021). https://doi.org/10.1016/j.laa.2021.04.011
18. Giscard, P.-L., Pozza, S.: Lanczos-like algorithm for the time-ordered exponential: the $*$-inverse problem. Appl. Math. **65**(6), 807–827 (2020). https://doi.org/10.21136/am.2020.0342-19
19. Magnus, W.: On the exponential solution of differential equations for a linear operator. Comm. Pure Appl. Math. **7**(4), 649–673 (1954)
20. Casas, F.: Sufficient conditions for the convergence of the Magnus expansion. J. Phys. A **40**(50), 15001–15017 (2007). https://doi.org/10.1088/1751-8113/40/50/006
21. Fel'dman, E.B.: On the convergence of the Magnus expansion for spin systems in periodic magnetic fields. Phys. Lett. **104A**(9), 479–481 (1984)
22. Maricq, M.M.: Convergence of the Magnus expansion for time dependent two level systems. J. Chem. Phys. **86**(10), 5647–5651 (1987)
23. Iserles, A., Munthe-Kaas, H.Z., Nørsett, S.P., Zanna, A.: Lie-group methods. Acta Numer. **9**, 215–365 (2000). https://doi.org/10.1017/S0962492900002154
24. Moan, P.C., Niesen, J.: Convergence of the Magnus series. Found. Comput. Math. **8**(3), 291–301 (2007). https://doi.org/10.1007/s10208-007-9010-0
25. Sánchez, S., Casas, F., Fernández, A.: New analytic approximations based on the Magnus expansion. J. Math. Chem. **49**(8), 1741–1758 (2011)
26. Giscard, P.-L., Lui, K., Thwaite, S.J., Jaksch, D.: An exact formulation of the time-ordered exponential using path-sums. J. Math. Phys. **56**(5), 053503 (2015)
27. Balasubramanian, V., DeCross, M., Kar, A., Parrikar, O.: Quantum complexity of time evolution with chaotic Hamiltonians. J. High Energ. Phys. **134** (2020)
28. Giscard, P.-L., Bonhomme, C.: Dynamics of quantum systems driven by time-varying Hamiltonians: solution for the Bloch–Siegert Hamiltonian and applications to NMR. Phys. Rev. Res. (2020). https://doi.org/10.1103/PhysRevResearch.2.023081
29. Flum, J., Grohe, M.: The parameterized complexity of counting problems. SIAM J. Comput. **33**, 892–922 (2004)
30. Moler, C., Van Loan, C.: Nineteen dubious ways to compute the exponential of a matrix. SIAM Rev. **20**(4), 801–836 (1978)
31. Moler, C., Van Loan, C.: Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. SIAM Rev. **45**(1), 3–49 (2003)
32. Higham, N.J.: Functions of Matrices. Theory and Computation, p. 425. SIAM, Philadelphia (2008)
33. Gutknecht, M.H.: A completed theory of the unsymmetric Lanczos process and related algorithms. I. SIAM J. Matrix Anal. Appl. **13**(2), 594–639 (1992)
34. Gutknecht, M.H.: A completed theory of the unsymmetric Lanczos process and related algorithms. II. SIAM J. Matrix Anal. Appl. **15**(1), 15–58 (1994)
35. Golub, G.H., Meurant, G.: Matrices, Moments and Quadrature with Applications. Princeton Ser. Appl. Math. Princeton University Press, Princeton, p. 363 (2010)
36. Liesen, J., Strakoš, Z.: Krylov Subspace Methods: Principles and Analysis. Numer. Math. Sci. Comput. Oxford University Press, Oxford (2013)
37. Pozza, S., Pranić, M.S., Strakoš, Z.: Gauss quadrature for quasi-definite linear functionals. IMA J. Numer. Anal. **37**(3), 1468–1495 (2017)
38. Pozza, S., Pranić, M.S., Strakoš, Z.: The Lanczos algorithm and complex Gauss quadrature. Electron. Trans. Numer. Anal. **50**, 1–19 (2018)
39. Parlett, B.N.: Reduction to tridiagonal form and minimal realizations. SIAM J. Matrix Anal. Appl. **13**(2), 567–593 (1992)
40. Draux, A.: Polynômes Orthogonaux Formels. Lecture Notes in Math., vol. 974. Springer, Berlin, p. 625 (1983)
41. Schwartz, L.: Théorie Des Distributions, Nouvelle édition, entièrement corrigée, refondue et augmentée. Hermann, Paris (1978)
42. Halperin, I., Schwartz, L.: *Introduction to the Theory of Distributions*. University of Toronto Press, Toronto (2019). https://doi.org/10.3138/9781442615151. https://toronto.degruyter.com/view/title/550976
43. Volterra, V., Pérès, J.: Leçons sur la Composition et les Fonctions Permutables. Éditions Jacques Gabay, Paris (1928)
44. Giscard, P.-L., Thwaite, S.J., Jaksch, D.: *Walk-sums, continued fractions and unique factorisation on digraphs* (2012). arXiv:1202.5523 [cs.DM]

45. Kılıç, E.: Explicit formula for the inverse of a tridiagonal matrix by backward continued fractions. Appl. Math. Comput. **197**(1), 345–357 (2008)
46. Taylor, D.R.: Analysis of the look ahead Lanczos algorithm. PhD thesis, University of California, Berkeley (1982)
47. Parlett, B.N., Taylor, D.R., Liu, Z.A.: A look-ahead Lanczos algorithm for unsymmetric matrices. Math. Comp. **44**(169), 105–124 (1985)
48. Freund, R.W., Gutknecht, M.H., Nachtigal, N.M.: An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices. SIAM J. Sci. Comput. **14**, 137–158 (1993)
49. Hochbruck, M., Lubich, C.: Exponential integrators for quantum-classical molecular dynamics. BIT Numer. Math. **39**(4), 620–645 (1999). https://doi.org/10.1023/A:1022335122807
50. Budd, C.J., Iserles, A., Iserles, A., Nørsett, S.P.: On the solution of linear differential equations in lie groups. Philos. Trans. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci. **357**(1754), 983–1019 (1999). https://doi.org/10.1098/rsta.1999.0362
51. Iserles, A.: On the global error of discretization methods for highly-oscillatory ordinary differential equations. BIT Numer. Math. **42**(3), 561–599 (2002). https://doi.org/10.1023/A:1022049814688
52. Iserles, A.: On the method of Neumann series for highly oscillatory equations. BIT Numer. Math. **44**(3), 473–488 (2004). https://doi.org/10.1023/B:BITN.0000046810.25353.95
53. Degani, I., Schiff, J.: RCMS: Right correction Magnus series approach for oscillatory ODEs. J. Comput. Appl. Math. **193**(2), 413–436 (2006). https://doi.org/10.1016/j.cam.2005.07.001
54. Cohen, D., Jahnke, T., Lorenz, K., Lubich, C.: Numerical integrators for highly oscillatory Hamiltonian systems: A review. In: Mielke, A. (ed.) Analysis, Modeling and Simulation of Multiscale Problems, pp. 553–576. Springer, Berlin (2006)
55. Bader, P., Iserles, A., Kropielnicka, K., Singh, P.: Efficient methods for linear Schrödinger equation in the semiclassical regime with time-dependent potential. Proc. R. Soc. A Math. Phys. Eng. Sci. **472**, 20150733 (2016)
56. Blanes, S., Casas, F.: A Concise Introduction to Geometric Numerical Integration. CRC Press, Bocan Raton (2017)