BIT Numerical Mathematics (2025) 65:2
https://doi.org/10.1007/s10543-024-01041-w

BIT

Matrix-less spectral approximation for large structured matrices

Giovanni Barbarino[1] · Melker Claesson[2] · Sven-Erik Ekström[2] · Carlo Garoni[3] · David Meadon[2] · Hendrik Speleers[3]

Received: 6 July 2022 / Accepted: 27 September 2024

Abstract
Sequences of structured matrices of increasing size, such as generalized locally Toeplitz sequences, arise in many scientific applications and especially in the numerical discretization of linear differential problems. We assume that the eigenvalues of a matrix X_n, belonging to a sequence of such kind, are given by a regular expansion. Under this working hypothesis, we propose a method for computing approximations of the eigenvalues of X_n for large n and we provide a theoretical analysis of its convergence. The method is called matrix-less because it does not operate on the matrix X_n but on a few similar matrices of smaller size combined with an interpolation-extrapolation strategy. The working hypothesis and the performance of the proposed eigenvalue approximation method are benchmarked on several numerical examples, with a special attention to matrices arising from the discretization of variable-coefficient differential problems.

Keywords Spectral approximation · Structured matrices · Discretization matrices · Generalized locally Toeplitz sequences · Eigenvalue expansion · Interpolation and extrapolation

Communicated by Elias Jarlebring.

G. Barbarino, C. Garoni, and H. Speleers are members of the Research Group GNCS (Gruppo Nazionale per il Calcolo Scientifico) of INdAM (Istituto Nazionale di Alta Matematica). G. Barbarino was supported by an Academy of Finland Grant (Suomen Akatemian Päätös 331240), by the Alfred Kordelinin Säätiö Grant 210122, and by the ERC Consolidator Grant 101085607 through the Project eLinoR. S.-E. Ekström was supported by the Swedish Research Council through the International Postdoc Grant (Registration Number 2019-00495). C. Garoni and H. Speleers were supported by the MUR Excellence Department Project MatMod@TOV (CUP E83C23000330006) awarded to the Department of Mathematics of the University of Rome Tor Vergata and by an INdAM-GNCS Project (CUP E53C22001930001). C. Garoni was also supported by the Department of Mathematics of the University of Rome Tor Vergata through the Project RICH_GLT (CUP E83C22001650005). D. Meadon was supported by the Centre for Interdisciplinary Mathematics (CIM) at Uppsala University. H. Speleers acknowledges the Italian Research Center in High Performance Computing, Big Data, and Quantum Computing (CUP E83C22003230001).

Extended author information available on the last page of the article

Published online: 16 December 2024

Mathematics Subject Classification Primary 65F15 · 15A18 · 65L10 · 65L15 · 15B05; Secondary 65B05 · 65D05

1 Introduction

Consider the discretization of a linear differential problem using a structured mesh characterized by a fineness parameter n. In this case, the computation of the numerical solution reduces to solving a linear discrete problem identified by a matrix X_n. The size of X_n grows as n increases, i.e., as the mesh is progressively refined, and ultimately we are left with a sequence of matrices X_n such that $\text{size}(X_n) \to \infty$ as $n \to \infty$. What is often observed in practice is the following:

- The sequence $\{X_n\}_n$ possesses a sort of (possibly hidden) structure and, in particular, it falls in the class of generalized locally Toeplitz (GLT) sequences; see Section 2.2.
- The eigenvalues of X_n, up to a small number of outliers, are asymptotically distributed as equispaced samples of a function f, the so-called spectral symbol; see Section 2.1.

The spectral symbol f allows us to extract accurate information about the spectrum of both X_n and the operator $\mathscr{X}$ underlying the considered differential problem; different ways have been proposed in recent works [1, 16, 17, 38]. A special role in all these papers is played by the monotone rearrangement of f, which is denoted by $f^\dagger$ and is used in [38] to formulate analytical predictions for the eigenvalues of both X_n and $\mathscr{X}$. We refer the reader to Section 2.1 for the precise definition of $f^\dagger$.

Here, we assume as a working hypothesis that the eigenvalues of X_n are not only distributed as a spectral symbol f but also given by the regular expansion (3.1). The first expansion function c_0 necessarily coincides with $f^\dagger$ (see Theorem 3.1), while the other functions $c_1, \ldots, c_\alpha$ can be interpreted as "higher-order symbols". Under this working hypothesis, we propose a method for computing approximations of the eigenvalues of X_n when n is large and we provide a theoretical analysis of its convergence. The method is called matrix-less because it does not operate on the matrix X_n but on a few similar matrices of smaller size combined with an interpolation-extrapolation strategy. It is a flexible version of an algorithm originally appeared in [34] for Hermitian Toeplitz matrices and tested since then on several other structured matrices, including block Toeplitz matrices [32], preconditioned Toeplitz matrices [2, 33], and matrices arising from the isogeometric analysis discretization of constant-coefficient differential equations [31]. The main novelties of this paper compared to [2, 31–34] are the following:

- The function $c_0 = f^\dagger$ is numerically computed by our eigenvalue approximation method along with $c_1, \ldots, c_\alpha$. This makes the method flexible as it becomes applicable even in the case where $f^\dagger$ is not known, which is usually the case in real-world applications.
- The theoretical convergence analysis of our eigenvalue approximation method is an extension of a similar analysis found in previous works to the case where $c_0 = f^\dagger$ is computed along with $c_1, \ldots, c_\alpha$. Moreover, the proof of Theorem 4.1

is considerably simplified compared to its analogue in [33], and Theorem 3.1 is completely new.
- An extensive numerical experimentation is presented, with the purpose of highlighting strengths and weaknesses of the proposed method. Its performance is benchmarked on several sequences of matrices X_n belonging to the class of GLT sequences, with a special attention to matrices arising from the discretization of variable-coefficient differential problems. To our knowledge, the latter matrices have not been considered in the matrix-less literature heretofore. It turns out that the proposed method yields better approximations of the eigenvalues of X_n compared to the analytical predictions considered in [38], which is no surprise as the latter only rely on the first expansion function $c_0 = f^\dagger$, while the former also exploit the "higher-order symbols" $c_1, \ldots, c_\alpha$.
- The overall presentation of the matrix-less paradigm is carried out for the first time in a systematic way and in full generality for arbitrary sequences of structured matrices, including (but not limited to) GLT sequences.

The present work can hence be considered as a review and a generalization of [2, 31–34].

It is worth noting that the working hypothesis is numerically illustrated to be often plausible but is not true in general. The efficiency of the proposed method in the numerical experimentation confirms that reasoning as if the working hypothesis were true leads to accurate eigenvalue approximations. In a sense, this is a testimony of the fact that the matrices X_n forming a GLT sequence—such as those arising from the discretization of differential problems—are much more structured than one might expect and in particular their spectra follow a "regular pattern".

Finally, we remark that neither the mainstream numerical methods for the approximation of the eigenvalues of large structured matrices [3, 41] nor the special eigensolvers for Toeplitz and Toeplitz-like matrices proposed by several authors—see, e.g., Arbenz [4], Badía and Vidal [6, 7], Bini and Di Benedetto [18], Bini and Pan [19], Di Benedetto [28, 29], Handy and Barlow [39], Ng and Trench [40], and Trench [43–47]—follow a matrix-less approach like the one considered herein. Even the matrix-free methods such as the power method [41, Chapter 4] or the Lanczos algorithm [41, Chapter 6] need to perform matrix-vector multiplications, which are not necessary in our case.

The paper is organized as follows. In Section 2, we collect some background material. In Section 3, we formulate the working hypothesis. In Section 4, we describe our eigenvalue approximation method and provide a theoretical convergence analysis. In Section 5, we present a number of numerical experiments both to support the plausibility of the working hypothesis and to show the performance of the method for various GLT sequences $\{X_n\}_n$, with a special attention to those arising from the discretization of variable-coefficient differential problems. In Section 6, we collect some concluding remarks and suggest possible future extensions. For a smoother reading of the paper, the proofs of the theorems are postponed to Appendix A.

2 Background

A matrix-sequence is a sequence of the form $\{X_n\}_n$, where X_n is an $n \times n$ matrix. Let μ_k be the Lebesgue measure in $\mathbb{R}^k$. Throughout this paper, all terminology from measure theory (such as "measurable function", "a.e.", etc.) always refers to the Lebesgue measure. We denote by $C_c(\mathbb{C})$ the space of continuous functions with bounded support defined on $\mathbb{C}$. If X is an $n \times n$ matrix, we denote by $\lambda_1(X), \ldots, \lambda_n(X)$ its eigenvalues, by $\|X\|_2$ its induced 2-norm, and by $X^\dagger$ its Moore–Penrose pseudoinverse.

2.1 Spectral distribution of a matrix-sequence

Definition 2.1 Let $\{X_n\}_n$ be a matrix-sequence and let $f : D \subset \mathbb{R}^k \to \mathbb{C}$ be measurable with $0 < \mu_k(D) < \infty$. We say that $\{X_n\}_n$ has a spectral distribution described by f, or equivalently that f is the spectral symbol of $\{X_n\}_n$, if

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} F(\lambda_i(X_n)) = \frac{1}{\mu_k(D)} \int_D F(f(\boldsymbol{x}))\mathrm{d}\boldsymbol{x}, \qquad \forall\, F \in C_c(\mathbb{C}). \tag{2.1}$$

In this case, we write $\{X_n\}_n \sim_\lambda f$.

The informal meaning behind the spectral distribution (2.1) is the following [36, p. 46]: Assuming that f is continuous a.e., the eigenvalues of X_n, except possibly for $o(n)$ outliers, are approximately equal to the samples of f over a uniform grid in the domain D (for n large enough).

The spectral symbol of a matrix-sequence is not unique. In particular, if we have $\{X_n\}_n \sim_\lambda f$ for some real measurable function $f : D \subset \mathbb{R}^k \to \mathbb{R}$ with $0 < \mu_k(D) < \infty$, then we also have $\{X_n\}_n \sim_\lambda f^\dagger$, where

$$f^\dagger : (0,1) \to \mathbb{R}, \qquad f^\dagger(t) = \inf\left\{ u \in \mathbb{R} : \frac{\mu_k\{\boldsymbol{x} \in D : f(\boldsymbol{x}) \le u\}}{\mu_k(D)} \ge t \right\}.$$

This result follows from the fact that

$$\int_0^1 F(f^\dagger(t))\mathrm{d}t = \frac{1}{\mu_k(D)} \int_D F(f(\boldsymbol{x}))\mathrm{d}\boldsymbol{x}, \qquad \forall\, F \in C_c(\mathbb{C}); \tag{2.2}$$

see [8, Section 2]. The function $f^\dagger$ is monotone non-decreasing on $(0,1)$ and is referred to as the monotone rearrangement of f. In the next lemma, we recall from [8] a simple procedure for constructing the monotone rearrangement of an a.e. continuous function.

Lemma 2.1 *Let $f : D \subset \mathbb{R}^k \to \mathbb{R}$ be continuous a.e. on D, where D is a bounded set with positive measure contained in $[a_1, b_1] \times \cdots \times [a_k, b_k]$ whose boundary ∂D satisfies $\mu_k(\partial D) = 0$. For each $r \in \mathbb{N} = \{1, 2, 3, \ldots\}$, consider the uniform samples*

$$f(\boldsymbol{x}^{(r)}_{i_1,\ldots,i_k}), \qquad \boldsymbol{x}^{(r)}_{i_1,\ldots,i_k} = \left(a_1 + i_1 \frac{b_1 - a_1}{r}, \ldots, a_k + i_k \frac{b_k - a_k}{r}\right),$$

with

$$(i_1,\dots,i_k)\in\mathcal{I}_r(D)=\left\{(i_1,\dots,i_k)\in\mathbb{N}^k : 1\le i_1,\dots,i_k\le r,\ \boldsymbol{x}^{(r)}_{i_1,\dots,i_k}\in D\right\},$$

sort them in ascending order, and put them into a vector $(s_0,s_1,\dots,s_{\omega(r)})$*, where* $\omega(r)=\#\mathcal{I}_r(D)-1$*. Let* $f_r^\dagger:[0,1]\to\mathbb{R}$ *be the piecewise linear function that interpolates the samples* $(s_0,s_1,\dots,s_{\omega(r)})$ *over the equally spaced nodes* $(0,\frac{1}{\omega(r)},\frac{2}{\omega(r)},\dots,1)$ *in* $[0,1]$*. Then* $f_r^\dagger\to f^\dagger$ *a.e. on* $(0,1)$ *as* $r\to\infty$*.*

2.2 GLT sequences

In this section, we collect some basics on the theory of GLT sequences. For a comprehensive treatment of the topic, we refer the reader to [11, 12, 36, 37]. For a more concise introduction to the subject, we recommend [24].

A GLT sequence $\{X_n\}_n$ is a special matrix-sequence equipped with a measurable function $f:[0,1]\times[-\pi,\pi]\to\mathbb{C}$, the so-called GLT symbol. We use the notation $\{X_n\}_n\sim_{\text{GLT}} f$ to indicate that $\{X_n\}_n$ is a GLT sequence with GLT symbol f. The three fundamental examples of GLT sequences are listed below.

- A matrix-sequence $\{Z_n\}_n$ such that $Z_n=R_n+N_n$ with $n^{-1}\text{rank}(R_n)\to 0$ and $\|N_n\|_2\to 0$ is referred to as a zero-distributed sequence. A matrix-sequence $\{Z_n\}_n$ is zero-distributed if and only if $\{Z_n\}_n\sim_{\text{GLT}} f(x,\theta)=0$.
- Let $n\in\mathbb{N}$ and $a:[0,1]\to\mathbb{C}$. The nth diagonal sampling matrix generated by a is the $n\times n$ diagonal matrix given by $D_n(a)=\text{diag}_{i=1,\dots,n}a(\frac{i}{n})$. Whenever a is continuous a.e. on $[0,1]$, we have $\{D_n(a)\}_n\sim_{\text{GLT}} f(x,\theta)=a(x)$.
- Let $n\in\mathbb{N}$ and $g\in L^1([-\pi,\pi])$. The nth Toeplitz matrix generated by g is the $n\times n$ matrix $T_n(g)=[g_{i-j}]_{i,j=1}^n$, where for $k\in\mathbb{Z}$ the number $g_k=\frac{1}{2\pi}\int_{-\pi}^{\pi}g(\theta)\mathrm{e}^{-\mathrm{i}k\theta}\mathrm{d}\theta$ is the kth Fourier coefficient of g. For all $g\in L^1([-\pi,\pi])$ we have $\{T_n(g)\}_n\sim_{\text{GLT}} f(x,\theta)=g(\theta)$.

The set of GLT sequences is a *-algebra closed under pseudoinversion. In practice, if $\{X_n\}_n\sim_{\text{GLT}} f$ and $\{Y_n\}_n\sim_{\text{GLT}} g$, then we have $\{X_n^*\}_n\sim_{\text{GLT}}\overline{f}$, $\{\alpha X_n+\beta Y_n\}_n\sim_{\text{GLT}}\alpha f+\beta g$ for all $\alpha,\beta\in\mathbb{C}$, $\{X_nY_n\}_n\sim_{\text{GLT}} fg$, and $\{X_n^\dagger\}_n\sim_{\text{GLT}} f^{-1}$ whenever $f\neq 0$ a.e. on $[0,1]\times[-\pi,\pi]$. Any GLT sequence $\{X_n\}_n\sim_{\text{GLT}} f$ usually enjoys a spectral distribution. This happens, for instance, when the matrices X_n are Hermitian or "almost" Hermitian, in which case the spectral symbol coincides with the GLT symbol f. GLT sequences can be formally defined as follows.

Definition 2.2 Let $\{X_n\}_n$ be a matrix-sequence and let $f:[0,1]\times[-\pi,\pi]\to\mathbb{C}$ be measurable. We say that $\{X_n\}_n$ is a GLT sequence with GLT symbol f, and we write $\{X_n\}_n\sim_{\text{GLT}} f$, if there exist functions $a_{i,m},g_{i,m}$, $i=1,\dots,N_m$, such that:

- $a_{i,m}:[0,1]\to\mathbb{C}$ is continuous a.e. on $[0,1]$ and $g_{i,m}\in L^1([-\pi,\pi])$.
- $f_m(x,\theta)=\sum_{i=1}^{N_m}a_{i,m}(x)g_{i,m}(\theta)$ tends to $f(x,\theta)$ a.e. on $[0,1]\times[-\pi,\pi]$ as $m\to\infty$.

– $\{X_{n,m}\}_n = \{\sum_{i=1}^{N_m} D_n(a_{i,m})T_n(g_{i,m})\}_n$ tends to $\{X_n\}_n$ as $m \to \infty$, in the following sense: For every m there exists n_m such that, for $n \geq n_m$,

$$X_n = X_{n,m} + R_{n,m} + N_{n,m}, \qquad \operatorname{rank}(R_{n,m}) \leq c(m)n, \qquad \|N_{n,m}\|_2 \leq \omega(m),$$

where $n_m, c(m), \omega(m)$ depend only on m, and $\lim_{m\to\infty} c(m) = \lim_{m\to\infty} \omega(m) = 0$.

We remark that many matrix-sequences $\{X_n\}_n$ arising from the discretization of differential problems fall in the class of GLT sequences. This is precisely the reason behind the interest in GLT sequences.

3 Working hypothesis

A matrix-sequence $\{X_n\}_n$ such that X_n has only real eigenvalues for all n is referred to as a spectrally real matrix-sequence. In this paper, we assume as a working hypothesis that the eigenvalues of such X_n are not only distributed as a spectral symbol f according to Definition 2.1, but are also given by the regular expansion (3.1). The first expansion function c_0 must necessarily coincide with $f^\dagger$ (see Theorem 3.1), while the other functions $c_1, \ldots, c_\alpha$ can be interpreted as "higher-order symbols". The formal statement of our working hypothesis is the following.

Working hypothesis Let $\{X_n\}_n$ be a spectrally real matrix-sequence. We say that $\{X_n\}_n$ satisfies the working hypothesis if there exists a sequence of functions $c_k : (0,1) \to \mathbb{R}$, $k = 0, 1, \ldots$, with c_0 continuous a.e. on $(0,1)$, such that, for every integer $\alpha \geq 0$, every n and every $j = 1, \ldots, n$, the following asymptotic expansion holds:

$$\lambda_j(X_n) = \sum_{k=0}^{\alpha} c_k(t_{j,n})h^k + E_{j,n,\alpha}, \tag{3.1}$$

where

– the eigenvalues of X_n are arranged in ascending order, $\lambda_1(X_n) \leq \cdots \leq \lambda_n(X_n)$;
– $h = \frac{1}{n+1}$ and $t_{j,n} = \frac{j}{n+1} = jh$;
– $|E_{j,n,\alpha}| \leq C_\alpha h^{\alpha+1}$ for some constant C_α depending only on α.

At least in the case where the strong condition "for every integer $\alpha \geq 0$" is replaced by "for some integer $\alpha \geq 0$" (say, $0 \leq \alpha \leq 3$), the working hypothesis is satisfied if $X_n = T_n(f)$ and f is a real function with certain properties. This happens, e.g., for $\alpha = 0, 1, 2$ if f enjoys the so-called simple-loop property, i.e., f is a real function in $L^1([-\pi,\pi])$ such that: (a) $f(-\pi) = f(\pi)$; (b) $\sum_{k\in\mathbb{Z}} k^4|f_k| < \infty$, where the numbers $f_k = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(\theta)\mathrm{e}^{-\mathrm{i}k\theta}\mathrm{d}\theta$ are the Fourier coefficients of f; (c) $f([-\pi,\pi]) = [0, M]$ with $M > 0$; (d) $f(0) = 0$ and $f''(0) > 0$; (e) if, with abuse of notation, we denote again by f the 2π-periodic extension of f to $\mathbb{R}$, then there exists $\theta_0 \in (0, 2\pi)$ such that $f(\theta_0) = M$, $f'(\theta) > 0$ for $0 < \theta < \theta_0$, $f'(\theta) < 0$ for $\theta_0 < \theta < 2\pi$, and

$f''(\theta_0) < 0$; see [21, Theorem 2.3]. We refer the reader to [13, 14, 21, 23, 25] for further details on Toeplitz matrices generated by simple-loop symbols.

In general, we may expect that the working hypothesis is satisfied (for some integer $\alpha \geq 0$) if $\{X_n\}_n$ is spectrally real and $\{X_n\}_n \sim_\lambda f$ with f being a real smooth function. On the other hand, if f is not smooth, it is unlikely that the working hypothesis is met. Nevertheless, as we are going to see in Section 5, reasoning as if the working hypothesis were true allows us to formulate an eigenvalue approximation method that is efficient for large matrices belonging to structured (spectrally real) matrix-sequences such as GLT sequences.

Theorem 3.1 *Let $\{X_n\}_n$ be a spectrally real matrix-sequence satisfying the working hypothesis. Then, the following properties hold:*

1. $\{X_n\}_n \sim_\lambda c_0$.
2. $c_0 : (0,1) \to \mathbb{R}$ *coincides a.e. with a non-decreasing function on* $(0,1)$.
3. *If* $\{X_n\}_n \sim_\lambda f$ *then* $c_0 = f^\dagger$ *a.e. on* $(0,1)$.

Proof See Appendix A. □

4 Eigenvalue approximation method

Following the same notation as [33], we associate with each positive integer n the stepsize $h = \frac{1}{n+1}$ and the grid $\{t_{1,n}, \ldots, t_{n,n}\}$ with $t_{j,n} = jh$ for $j = 1, \ldots, n$. We will always denote a positive integer and the associated stepsize in the same way, in the sense that if the positive integer is denoted by n, the associated stepsize is denoted by h; if the positive integer is denoted by n_j, the associated stepsize is denoted by h_j; and so on. Throughout this section, we make the following assumptions:

- $\{X_n\}_n$ is a spectrally real matrix-sequence satisfying the working hypothesis.
- $\alpha \geq 0$ and $n, n_0 \geq 1$ are fixed integers.
- $n_k = 2^k(n_0+1) - 1$ for $k = 0, \ldots, \alpha$. Note that $n_k = n_k(n_0)$ depends not only on k but also on n_0; we hide the dependence on n_0 for notational simplicity.
- $j_k = 2^k j_0$ for $j_0 = 1, \ldots, n_0$ and $k = 0, \ldots, \alpha$. Note that $j_k = j_k(j_0)$ depends not only on k but also on j_0; we hide the dependence on j_0 for notational simplicity. Note also that j_k is the index in $\{1, \ldots, n_k\}$ such that $t_{j_k,n_k} = t_{j_0,n_0}$.

A graphical representation of the grids $\{t_{1,n_k}, \ldots, t_{n_k,n_k}\}$, $k = 0, \ldots, \alpha$, is depicted in Fig. 1 for $n_0 = 5$ and $\alpha = 3$. For each "level" k, the corresponding red circles highlight the subgrid $\{t_{j_k,n_k} : j_0 = 1, \ldots, n_0\}$ which coincides with the coarsest grid $\{t_{j_0,n_0} : j_0 = 1, \ldots, n_0\}$.

4.1 Description of the eigenvalue approximation method

The method we are going to describe is designed for computing approximations of the eigenvalues of X_n in the case where n is large compared to $n_0, \ldots, n_\alpha$, so that the computation of the eigenvalues of X_n is expensive from a computational point of view (if performed by a standard eigensolver), but the computation of the eigenvalues

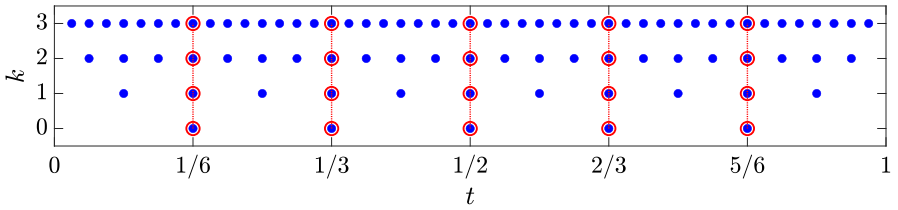

Fig. 1 Representation of the grids $\{t_{1,n_k}, \ldots, t_{n_k,n_k}\}$, $k = 0, \ldots, \alpha$, for $n_0 = 5$ and $\alpha = 3$

of $X_{n_0}, \ldots, X_{n_\alpha}$—which is required by our method—can be efficiently performed by a standard eigensolver; see also Remark 4.1 below. Our method is composed of two phases: a first phase where we invoke extrapolation procedures, and a second phase where local interpolation techniques are employed.

4.1.1 Extrapolation

For each fixed $j_0 = 1, \ldots, n_0$, we apply $\alpha + 1$ times the expansion (3.1) with $n = n_0, n_1, \ldots, n_\alpha$ and $j = j_0, j_1, \ldots, j_\alpha$. Since $t_{j_0,n_0} = t_{j_1,n_1} = \ldots = t_{j_\alpha,n_\alpha}$ (by definition of $j_1, \ldots, j_\alpha$), we obtain

$$\begin{cases} \lambda_{j_0,n_0}(X_{n_0}) = \sum_{k=0}^{\alpha} c_k(t_{j_0,n_0}) h_0^k + E_{j_0,n_0,\alpha} \\ \lambda_{j_1,n_1}(X_{n_1}) = \sum_{k=0}^{\alpha} c_k(t_{j_0,n_0}) h_1^k + E_{j_1,n_1,\alpha} \\ \qquad\qquad \vdots \\ \lambda_{j_\alpha,n_\alpha}(X_{n_\alpha}) = \sum_{k=0}^{\alpha} c_k(t_{j_0,n_0}) h_\alpha^k + E_{j_\alpha,n_\alpha,\alpha} \end{cases}$$

i.e.,

$$\begin{bmatrix} 1 & h_0 & h_0^2 & \cdots & h_0^\alpha \\ 1 & h_1 & h_1^2 & \cdots & h_1^\alpha \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & h_\alpha & h_\alpha^2 & \cdots & h_\alpha^\alpha \end{bmatrix} \begin{bmatrix} c_0(t_{j_0,n_0}) \\ c_1(t_{j_0,n_0}) \\ \vdots \\ c_\alpha(t_{j_0,n_0}) \end{bmatrix} = \begin{bmatrix} \lambda_{j_0,n_0}(X_{n_0}) \\ \lambda_{j_1,n_1}(X_{n_1}) \\ \vdots \\ \lambda_{j_\alpha,n_\alpha}(X_{n_\alpha}) \end{bmatrix} - \begin{bmatrix} E_{j_0,n_0,\alpha} \\ E_{j_1,n_1,\alpha} \\ \vdots \\ E_{j_\alpha,n_\alpha,\alpha} \end{bmatrix}, \tag{4.1}$$

where

$$|E_{j_k,n_k,\alpha}| \le C_\alpha h_k^{\alpha+1}, \qquad k = 0, \ldots, \alpha. \tag{4.2}$$

Let $\tilde{c}_0(t_{j_0,n_0}), \ldots, \tilde{c}_\alpha(t_{j_0,n_0})$ be the approximations of $c_0(t_{j_0,n_0}), \ldots, c_\alpha(t_{j_0,n_0})$ computed by removing the errors $E_{j_0,n_0,\alpha}, \ldots, E_{j_\alpha,n_\alpha,\alpha}$ in (4.1) and by solving the resulting linear system:

$$\begin{bmatrix} 1 & h_0 & h_0^2 & \cdots & h_0^\alpha \\ 1 & h_1 & h_1^2 & \cdots & h_1^\alpha \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & h_\alpha & h_\alpha^2 & \cdots & h_\alpha^\alpha \end{bmatrix} \begin{bmatrix} \tilde{c}_0(t_{j_0,n_0}) \\ \tilde{c}_1(t_{j_0,n_0}) \\ \vdots \\ \tilde{c}_\alpha(t_{j_0,n_0}) \end{bmatrix} = \begin{bmatrix} \lambda_{j_0,n_0}(X_{n_0}) \\ \lambda_{j_1,n_1}(X_{n_1}) \\ \vdots \\ \lambda_{j_\alpha,n_\alpha}(X_{n_\alpha}) \end{bmatrix}. \tag{4.3}$$

Note that this way of computing approximations for $c_0(t_{j_0,n_0}), \ldots, c_\alpha(t_{j_0,n_0})$ is completely analogous to the Richardson extrapolation procedure that is employed in the context of Romberg integration to accelerate the convergence of the trapezoidal rule [42, Section 3.4]. In this regard, the asymptotic expansion (3.1) plays here the same role as the Euler–Maclaurin summation formula [42, Section 3.3]. For more advanced studies on extrapolation methods, we refer the reader to [26]. The next theorem gives a bound for the approximation error $|c_k(t_{j_0,n_0}) - \tilde{c}_k(t_{j_0,n_0})|$.

Theorem 4.1 *There exists a constant A_α depending only on α such that, for $j_0 = 1, \ldots, n_0$ and $k = 0, \ldots, \alpha$,*

$$|c_k(t_{j_0,n_0}) - \tilde{c}_k(t_{j_0,n_0})| \le A_\alpha h_0^{\alpha-k+1}. \tag{4.4}$$

Proof See Appendix A. □

4.1.2 Interpolation

Fix an index $j \in \{1, \ldots, n\}$. To compute an approximation of $\lambda_j(X_n)$ through the expansion (3.1) we would need the value $c_k(t_{j,n})$ for each $k = 0, \ldots, \alpha$. Of course, $c_k(t_{j,n})$ is not available in practice, but we can approximate it by interpolating in some way the values $\tilde{c}_k(t_{j_0,n_0})$, $j_0 = 1, \ldots, n_0$. As shown in Theorem 4.2, a local approximation strategy that preserves the accuracy (4.4), at least if $c_k(t)$ is sufficiently smooth, is the following: Let $t^{(1)}, \ldots, t^{(\beta_k)}$, $\beta_k \ge \alpha - k + 1$, be distinct points from the grid $\{t_{1,n_0}, \ldots, t_{n_0,n_0}\}$ which are closest to the point $t_{j,n}$,[1] and let $\tilde{c}_{k,j}(t)$ be the interpolation polynomial of the data $(t^{(1)}, \tilde{c}_k(t^{(1)})), \ldots, (t^{(\beta_k)}, \tilde{c}_k(t^{(\beta_k)}))$; then, we approximate $c_k(t_{j,n})$ by $\tilde{c}_{k,j}(t_{j,n})$. Note that, by selecting β_k points from $\{t_{1,n_0}, \ldots, t_{n_0,n_0}\}$, we are implicitly assuming that $n_0 \ge \beta_k$.

Theorem 4.2 *Let $0 \le k \le \alpha$, let $\beta_k \ge \alpha - k + 1$ be a positive integer depending only on k, α, suppose that $c_k \in C^{\beta_k}([0,1])$, and fix $1 \le j \le n$. If $t^{(1)}, \ldots, t^{(\beta_k)}$ are β_k distinct points from $\{t_{1,n_0}, \ldots, t_{n_0,n_0}\}$ which are closest to $t_{j,n}$, and if $\tilde{c}_{k,j}(t)$ is the interpolation polynomial of the data $(t^{(1)}, \tilde{c}_k(t^{(1)})), \ldots, (t^{(\beta_k)}, \tilde{c}_k(t^{(\beta_k)}))$, then*

$$|c_k(t_{j,n}) - \tilde{c}_{k,j}(t_{j,n})| \le B_\alpha h_0^{\alpha-k+1} \tag{4.5}$$

for some constant B_α depending only on α.

Proof See Appendix A. □

[1] These β_k points are uniquely determined by $t_{j,n}$ except possibly in the case where $t_{j,n}$ coincides with either a grid point t_{j_0,n_0} or the midpoint between two consecutive grid points t_{j_0,n_0} and t_{j_0+1,n_0}.

4.2 Formulation of the algorithm

We are now ready to translate our method into an algorithm for computing approximations of the eigenvalues of X_n. A plain MATLAB implementation of this algorithm is reported in [9, Appendix C].

Algorithm 4.1 The inputs are the following:

- Three integers α, n, n_0 with $\alpha \geq 0$ and $n, n_0 \geq 1$.
- The matrices $X_{n_0}, \ldots, X_{n_\alpha}$, where $n_k = 2^k(n_0+1)-1$ for all $k = 0, \ldots, \alpha$ and $\{X_n\}_n$ is a spectrally real matrix-sequence which is assumed to satisfy the working hypothesis.
- A vector $(\beta_0, \ldots, \beta_\alpha)$ consisting of $\alpha+1$ positive integers $\beta_k \in \{1, \ldots, n_0\}$.
- A set $S \subseteq \{1, \ldots, n\}$.

The algorithm computes an approximation of the eigenvalues $\{\lambda_j(X_n) : j \in S\}$ as follows:

1. Compute the eigenvalues of the small matrices $X_{n_0}, \ldots, X_{n_\alpha}$.
2. For $j_0 = 1, \ldots, n_0$ compute the vector $[\tilde{c}_0(t_{j_0,n_0}), \tilde{c}_1(t_{j_0,n_0}), \ldots, \tilde{c}_\alpha(t_{j_0,n_0})]^T$ by solving (4.3).
3. For $j \in S$
   - For $k = 0, \ldots, \alpha$
     - Find β_k points $t^{(1)}, \ldots, t^{(\beta_k)} \in \{t_{1,n_0}, \ldots, t_{n_0,n_0}\}$ which are closest to $t_{j,n}$.
     - Compute $\tilde{c}_{k,j}(t_{j,n})$, where $\tilde{c}_{k,j}(t)$ is the interpolation polynomial of the data $(t^{(1)}, \tilde{c}_k(t^{(1)})), \ldots, (t^{(\beta_k)}, \tilde{c}_k(t^{(\beta_k)}))$.
   - Compute $\tilde{\lambda}_j(X_n) = \sum_{k=0}^{\alpha} \tilde{c}_{k,j}(t_{j,n})h^k$.
4. Return $\{\tilde{\lambda}_j(X_n) : j \in S\}$ as an approximation of $\{\lambda_j(X_n) : j \in S\}$.

Remark 4.1 Algorithm 4.1 is specifically designed for computing approximations of the eigenvalues of X_n in the case where the matrix size n is large. When applying this algorithm, it is implicitly assumed that n_0 and α are small (much smaller than n), so that each $n_k = 2^k(n_0+1)-1$ is small as well and the computation of the eigenvalues of X_{n_k}—which is required in the first step—can be efficiently performed by any standard eigensolver.

Remark 4.2 An elegant choice for the number β_k of interpolation points is $\beta_k = \alpha - k + 1$, which is the minimal value satisfying the assumptions of Theorem 4.2. Another valid choice is $\beta_k = \beta(\alpha)$ for all $k = 0, \ldots, \alpha$ with $\beta(\alpha) \geq \alpha + 1$.

Remark 4.3 In practice, the dominant term in the computational cost of Algorithm 4.1 is $C_{\text{eig}}(n_\alpha)$, i.e., the cost for computing the eigenvalues of X_{n_α}. As we shall see in the numerical experiments, this cost, as well as the overall cost of Algorithm 4.1, is usually much less than the cost $C_{\text{eig}}(n)$ for computing the eigenvalues of X_n.

Remark 4.4 The $(\alpha+1)\times n_0$ matrix

$$\tilde{C}_{\alpha,n_0}=\begin{bmatrix}\tilde{c}_0(t_{1,n_0}) & \tilde{c}_0(t_{2,n_0}) & \cdots & \tilde{c}_0(t_{n_0,n_0})\\ \tilde{c}_1(t_{1,n_0}) & \tilde{c}_1(t_{2,n_0}) & \cdots & \tilde{c}_1(t_{n_0,n_0})\\ \vdots & \vdots & & \vdots\\ \tilde{c}_\alpha(t_{1,n_0}) & \tilde{c}_\alpha(t_{2,n_0}) & \cdots & \tilde{c}_\alpha(t_{n_0,n_0})\end{bmatrix}$$

is implicitly computed in the second step of Algorithm 4.1. If n_0 is large enough then, by Theorem 4.1,

$$\tilde{C}_{\alpha,n_0}\approx\begin{bmatrix}c_0(t_{1,n_0}) & c_0(t_{2,n_0}) & \cdots & c_0(t_{n_0,n_0})\\ c_1(t_{1,n_0}) & c_1(t_{2,n_0}) & \cdots & c_1(t_{n_0,n_0})\\ \vdots & \vdots & & \vdots\\ c_\alpha(t_{1,n_0}) & c_\alpha(t_{2,n_0}) & \cdots & c_\alpha(t_{n_0,n_0})\end{bmatrix},$$

and so a plot of the kth row of $\tilde{C}_{\alpha,n_0}$ versus the grid points $(t_{1,n_0},t_{2,n_0},\ldots,t_{n_0,n_0})$ produces the (approximate) graph of the kth function c_k in the expansion (3.1).

Remark 4.5 The matrix-less approach has already been proposed in the recent literature for the computation of the eigenvalues of matrices belonging to special GLT sequences. However, contrary to its earlier versions appeared in [2, 31–34], Algorithm 4.1 computes not only the rows $1,\ldots,\alpha$ of the matrix $\tilde{C}_{\alpha,n_0}$ in Remark 4.4, but also the 0th row, i.e., an approximation of the function c_0 in (3.1). This strategy was suggested by the method proposed in [22, 35] to compute the so-called "spectral function" and makes the matrix-less paradigm flexible, because Algorithm 4.1 is applicable even in the case where c_0 is not known. In the case where c_0 is known, the eigenvalue approximations $\tilde{\lambda}_j(X_n)=\tilde{c}_0(t_{j,n})+\sum_{k=1}^{\alpha}\tilde{c}_k(t_{j,n})h^k$ returned by Algorithm 4.1 can be updated by considering the generally more accurate eigenvalue approximations $\tilde{\lambda}_j(X_n)=c_0(t_{j,n})+\sum_{k=1}^{\alpha}\tilde{c}_k(t_{j,n})h^k$ obtained from the expansion (3.1) by using c_0 instead of $\tilde{c}_0$. Alternatively, in the case where c_0 is known, we could reformulate Algorithm 4.1 in analogy with [33, Algorithm 1] to avoid the computation of $\tilde{c}_0$. We point out, however, that c_0 is usually not known in real-world applications such as, e.g., the case of GLT sequences arising from the discretization of variable-coefficient differential problems, which is the case of interest in this paper.

4.3 Error estimate

Theorem 4.3 *Suppose that $c_k\in C^{\beta_k}([0,1])$ and $\beta_k\geq\alpha-k+1$ for $k=0,\ldots,\alpha$, with β_k depending only on k and α. Let $n\geq n_0$ and let $(\tilde{\lambda}_1(X_n),\ldots,\tilde{\lambda}_n(X_n))$ be the approximation of $(\lambda_1(X_n),\ldots,\lambda_n(X_n))$ computed by Algorithm 4.1. Then, there exists a constant D_α depending only on α such that*

$$\max_{j=1,\ldots,n}|\lambda_j(X_n)-\tilde{\lambda}_j(X_n)|\leq D_\alpha h_0^{\alpha+1}.$$

Proof See Appendix A. □

Remark 4.6 Theorem 4.3 shows that, for any fixed $\alpha \geq 0$, the numerical eigenvalues computed by Algorithm 4.1 are affected by an error of the order of $h_0^{\alpha+1}$. In practice, to improve the eigenvalue approximations, it is advisable to fix α and increase n_0 up to a maximum allowed value such that $n_\alpha = 2^\alpha(n_0+1)-1 < n$. The other way (fix n_0 and increase α) is less advisable; see Example 5.1 below.

5 Numerical experiments

In this section, we illustrate through numerical examples the plausibility of the working hypothesis and the performance of our eigenvalue approximation method for various spectrally real GLT sequences $\{X_n\}_n \sim_\lambda f$ with f being a real function. A special attention is devoted to GLT sequences arising from the discretization of variable-coefficient differential problems. In each example, we proceed as follows.

(a) We support the plausibility of the working hypothesis for the considered matrix-sequence $\{X_n\}_n$ by exploiting Theorem 3.1 and Remark 4.4. If we fix $\alpha \geq 0$ and $0 \leq k \leq \alpha$, and we plot the kth row $(\tilde{c}_k(t_{1,n_0}), \ldots, \tilde{c}_k(t_{n_0,n_0}))$ of $\tilde{C}_{\alpha,n_0}$ versus the grid points $(t_{1,n_0}, \ldots, t_{n_0,n_0})$ for different (large) values of n_0, then

- the resulting plots should overlap and reproduce the graph of a function c_k;
- in the case $k = 0$, the resulting plots should also overlap with the graph of $f^\dagger$, the monotone rearrangement of the spectral symbol f of $\{X_n\}_n$.

We will see that both the above expectations are usually met, at least for small values of α.

(b) We illustrate the performance of our method for computing approximations of all the eigenvalues of X_n for a large value of n. This is done by fixing α and n_0, and by

- comparing the CPU time taken by Algorithm 4.1 for computing approximations of the eigenvalues of X_n with the CPU time taken by MATLAB's `eig` function for computing the eigenvalues of X_n;
- showing, for $j = 1, \ldots, n$, both the absolute and the relative eigenvalue errors

$$\varepsilon_{A,j,n}^{(\alpha,n_0)} = \left|\tilde{\lambda}_j^{(\alpha,n_0)}(X_n) - \lambda_j(X_n)\right|, \qquad \varepsilon_{R,j,n}^{(\alpha,n_0)} = \left|\frac{\tilde{\lambda}_j^{(\alpha,n_0)}(X_n) - \lambda_j(X_n)}{\lambda_j(X_n)}\right|,$$

with $(\lambda_1(X_n), \ldots, \lambda_n(X_n))$ and $(\tilde{\lambda}_1^{(\alpha,n_0)}(X_n), \ldots, \tilde{\lambda}_n^{(\alpha,n_0)}(X_n))$ being, respectively, the vector of the eigenvalues of X_n (sorted as usual in ascending order) and its approximation obtained by Algorithm 4.1.

In some of the examples, besides the programs described in (a) and (b), we also consider the following program.

(c) We illustrate the performance of our method for computing approximations of a specific subset of the eigenvalues of X_n for a large value of n. This is done by

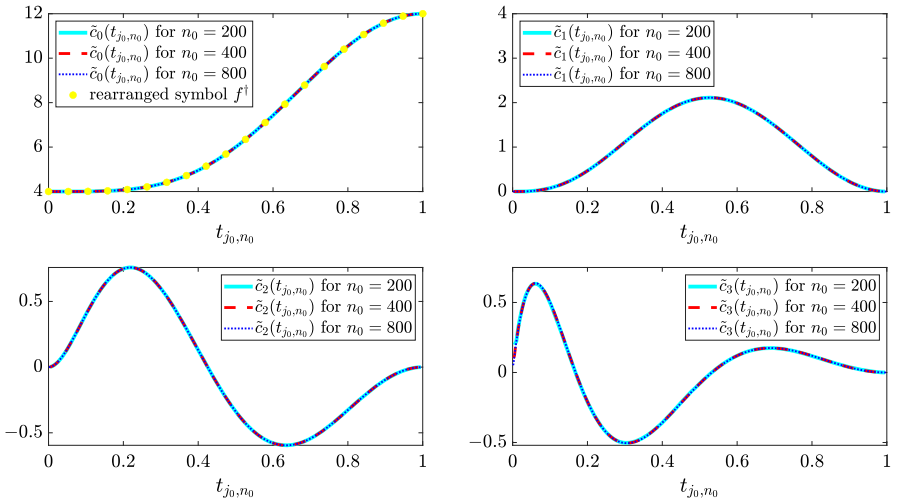

Fig. 2 Example 5.1 — $(\tilde{c}_k(t_{1,n_0}), \ldots, \tilde{c}_k(t_{n_0,n_0}))$ versus $(t_{1,n_0}, \ldots, t_{n_0,n_0})$ for $k = 0, \ldots, \alpha$ and different values of n_0 in the case $\alpha = 3$

fixing $S \subseteq \{1, \ldots, n\}$, α and n_0, and by comparing the CPU time taken by Algorithm 4.1 for computing approximations of the eigenvalues of X_n corresponding to the indices in S with the CPU time taken by MATLAB's `eigs` function for computing the eigenvalues of X_n corresponding to the indices in S.

Algorithm 4.1 is always applied with $\beta_k = \alpha + 2$ for $k = 0, \ldots, \alpha$. The CPU times are always measured in seconds. The CPU times for Algorithm 4.1 refer to the MATLAB implementation reported in [9, Appendix C] and do not account for the time spent in the construction of the matrices $X_{n_0}, \ldots, X_{n_\alpha}$. Similarly, the CPU times for MATLAB's `eig` and `eigs` functions do not account for the time spent in the construction of the matrix X_n. Note that if the times spent in the construction of the matrices were taken into consideration, the matrix-less Algorithm 4.1 would gain over MATLAB's `eig` and `eigs` functions, because the construction of the large matrix X_n is more time-consuming than the construction of the small matrices $X_{n_0}, \ldots, X_{n_\alpha}$. The numerical experiments have been performed with MATLAB R2021a (64 bit) on a platform with 16 GB RAM and an Intel® Core™ i5-9400H Processor (Quad-Core, 8 MB Intel® Smart Cache, 2.50 GHz, 4.30 GHz Turbo).

Example 5.1 (banded Toeplitz matrix) Let $X_n = T_n(f)$ with $f(\theta) = 7 - 4\cos\theta + \cos(2\theta)$. The matrix-sequence $\{X_n\}_n$ is spectrally real and $\{X_n\}_n \sim_{\text{GLT},\lambda} f$.

- In Fig. 2, we collect the results related to item (a) for $\alpha = 3$. It is clear that, for each fixed $k = 0, \ldots, \alpha$, the plots obtained for different values of n_0 overlap and reproduce the graph of a function c_k. Moreover, for $k = 0$, the plots also overlap with the graph of $f^\dagger$, the monotone rearrangement of the spectral symbol f obtained through the procedure described in Lemma 2.1.
- In Table 1 and Fig. 3, we collect the results related to item (b) for $n = 20000$. We consider two strategies for improving the eigenvalue approximations returned by

Table 1 Example 5.1 — CPU times for computing all the eigenvalues of X_n in the case $n = 20000$. The corresponding eigenvalue errors are shown in Fig. 3

Method	CPU time
Algor. 4.1 ($\alpha = 3,\ n_0 = 100$)	1.15
Algor. 4.1 ($\alpha = 3,\ n_0 = 200$)	1.18
Algor. 4.1 ($\alpha = 3,\ n_0 = 400$)	1.34
Algor. 4.1 ($\alpha = 4,\ n_0 = 100$)	1.53
Algor. 4.1 ($\alpha = 5,\ n_0 = 100$)	1.97
MATLAB's `eig` function	5.54

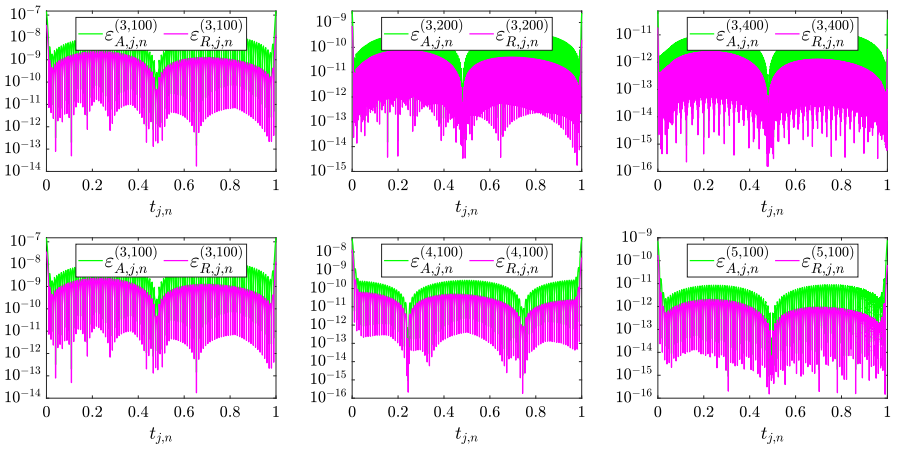

Fig. 3 Example 5.1 — $(\varepsilon_{A,1,n}^{(\alpha,n_0)}, \ldots, \varepsilon_{A,n,n}^{(\alpha,n_0)})$ and $(\varepsilon_{R,1,n}^{(\alpha,n_0)}, \ldots, \varepsilon_{R,n,n}^{(\alpha,n_0)})$ versus $(t_{1,n}, \ldots, t_{n,n})$ in the case $n = 20000$. First row: $\alpha = 3$ and $n_0 = 100, 200, 400$. Second row: $\alpha = 3, 4, 5$ and $n_0 = 100$

Algorithm 4.1: First, we fix $\alpha = 3$ and increase n_0 from 100 to 400; second, we fix $n_0 = 100$ and increase α from 3 to 5. Table 1 shows that the first strategy is slightly faster, and Fig. 3 that it yields overall better eigenvalue errors. We remark that, for $n = 20000$ and $\alpha = 3$, we have to keep $n_0 < 2500$, because otherwise $n_\alpha = 2^\alpha(n_0 + 1) - 1$ would be larger than n. We also remark that in this case the CPU times of Algorithm 4.1 are comparable to the CPU time of MATLAB's `eig` function because X_n is a real sparse symmetric matrix, a kind of matrix for which MATLAB's `eig` function is extremely efficient. However, in general, the CPU times of Algorithm 4.1 are lower than that of MATLAB's `eig` (see the examples below).

Remark 5.1 Example 5.1 shows that, in order to achieve higher accuracy in the eigenvalue approximations returned by Algorithm 4.1, it is better to fix α and increase n_0 rather than fix n_0 and increase α. This was already observed in [33, Section 3] and can be explained in two ways: First, assuming the working hypothesis is satisfied, the constant D_α in Theorem 4.3 seems to grow very quickly with α [33, Example 1]; second, the working hypothesis is usually not satisfied and in particular the expansion (3.1) is hardly met for large values of α. The latter consideration also hints that choosing small values of α is preferable. In practice, we suggest taking $\alpha = 2, 3, 4$ but not more. In all the other examples of this paper, we will focus on the "winning strategy" of keeping α fixed (to a small value) and increasing n_0.

Remark 5.2 Example 5.1 deals with a banded Toeplitz matrix. For an example with a full Toeplitz matrix, we refer the reader to [9, Example 5.2], which has not been included here for conciseness purposes.

Example 5.2 (preconditioned Toeplitz matrix) Let $X_n = T_n(u)^{-1}T_n(v)$ with $u(\theta) = 2+\cos(3\theta)$ and $v(\theta) = 8-3\cos\theta-\frac{9}{2}\cos(2\theta)+4\cos(3\theta)-\frac{1}{2}\cos(4\theta)-\cos(5\theta)$. The matrix-sequence $\{X_n\}_n$ is spectrally real and $\{X_n\}_n \sim_{\mathrm{GLT},\lambda} f(\theta) = v(\theta)/u(\theta) = 4-\cos\theta-2\cos(2\theta)$; see [36, Exercise 8.4].

- In Fig. 4, we collect the results related to item (a) for $\alpha = 2$ and we also show the graph of the spectral symbol f over $[0,\pi]$. For each fixed $k = 0,\dots,\alpha$, the plots obtained for different values of n_0 seem to overlap, but for $k = 1, 2$ they are not reproducing the graph of a function c_k outside the interval $(0,\hat{t})$, with $\hat{t} = \hat{\theta}/\pi$ and $\hat{\theta} = 0.722734\dots$ We remark that the spectral symbol f is not monotone on $[0,\pi]$ and in particular it does not satisfy the simple-loop property. Nevertheless, f is monotone on the interval $(0,\hat{\theta})$ and $f^{-1}(f((0,\hat{\theta}))) = (0,\hat{\theta})$. This suggests that we may still say that the working hypothesis is "satisfied on $(0,\hat{t})$", in the sense that the expansion (3.1) holds for the indices $j = 1,\dots,n$ such that $t_{j,n} \in (0,\hat{t})$; see also [33, Examples 9 and 10].
- In Table 2 and Fig. 5, we collect the results related to item (b) for $n = 10000$. We see from Table 2 that the CPU times of Algorithm 4.1 are lower than that of MATLAB's `eig` function. Concerning the eigenvalue errors shown in Fig. 5, as expected, we observe a difficulty in the approximation of the eigenvalues $\lambda_j(X_n)$ corresponding to points $t_{j,n} \notin (0,\hat{t})$ for which the expansion (3.1) fails, but for the others the algorithm works well.
- In Table 3, we collect the results related to item (c) for $n = 10000$ and $S = \{1,\dots,2300\}$. The choice of S is inspired by the fact that, as illustrated in Fig. 5, Algorithm 4.1 works well for the smallest $\hat{t}n \approx 2300.53$ eigenvalues. To improve its efficiency, MATLAB's `eigs` function has been applied to the generalized eigenvalue problem $T_n(v)\boldsymbol{x} = \lambda\, T_n(u)\boldsymbol{x}$ with $T_n(v)$ and $T_n(u)$ allocated as sparse matrices through MATLAB's `sparse` command. We see from Table 3 that the CPU times of Algorithm 4.1 are lower than that of MATLAB's `eigs` function.

Example 5.3 (finite difference matrices) Consider the following second-order differential problem:

$$\begin{cases} -(a(x)u'(x))' = f(x), & x \in (0,1), \\ u(0) = 0, \quad u(1) = 0. \end{cases}$$

In the classical finite difference method based on second-order central differences over the uniform grid $x_i = \frac{i}{n+1}$, $i = 0,\dots,n+1$, the computation of the numerical solution reduces to solving a linear system whose coefficient matrix is the symmetric tridiagonal matrix given by

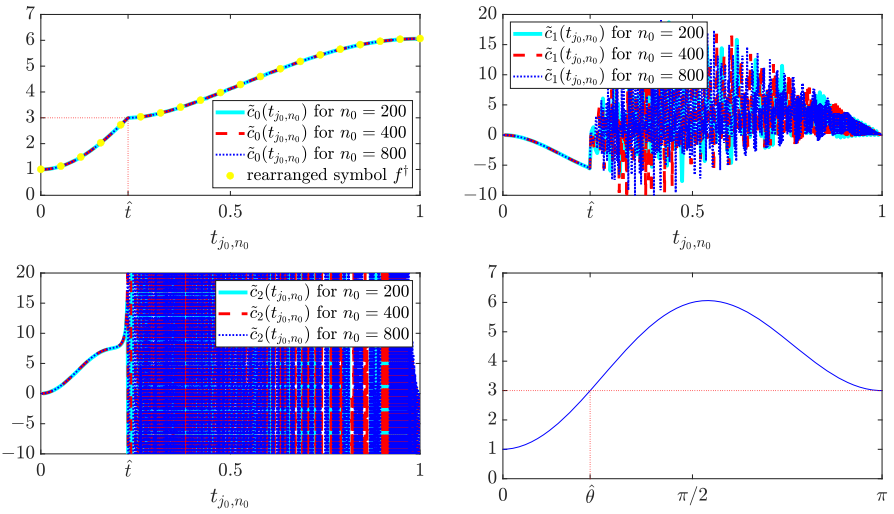

Fig. 4 Example 5.2 — Top left–right and bottom left: $(\tilde{c}_k(t_{1,n_0}), \ldots, \tilde{c}_k(t_{n_0,n_0}))$ versus $(t_{1,n_0}, \ldots, t_{n_0,n_0})$ for $k = 0, \ldots, \alpha$ and different values of n_0 in the case $\alpha = 2$. Bottom right: graph of f over $[0, \pi]$

Table 2 Example 5.2 — CPU times for computing all the eigenvalues of X_n in the case $n = 10000$. The corresponding eigenvalue errors are shown in Fig. 5

Method	CPU time
Algor. 4.1 ($\alpha = 1,\ n_0 = 400$)	0.56
Algor. 4.1 ($\alpha = 1,\ n_0 = 800$)	2.22
Algor. 4.1 ($\alpha = 1,\ n_0 = 1600$)	17.42
Algor. 4.1 ($\alpha = 2,\ n_0 = 200$)	0.71
Algor. 4.1 ($\alpha = 2,\ n_0 = 400$)	2.39
Algor. 4.1 ($\alpha = 2,\ n_0 = 800$)	17.74
MATLAB's `eig` function	462.16

Table 3 Example 5.2 — CPU times for computing the smallest eigenvalues of X_n corresponding to the indices $\{1, \ldots, 2300\}$ in the case $n = 10000$

Method	CPU time
Algor. 4.1 ($\alpha = 1,\ n_0 = 400$)	0.34
Algor. 4.1 ($\alpha = 1,\ n_0 = 800$)	2.06
Algor. 4.1 ($\alpha = 1,\ n_0 = 1600$)	17.10
Algor. 4.1 ($\alpha = 2,\ n_0 = 200$)	0.41
Algor. 4.1 ($\alpha = 2,\ n_0 = 400$)	2.15
Algor. 4.1 ($\alpha = 2,\ n_0 = 800$)	17.37
MATLAB's `eigs` function	104.09

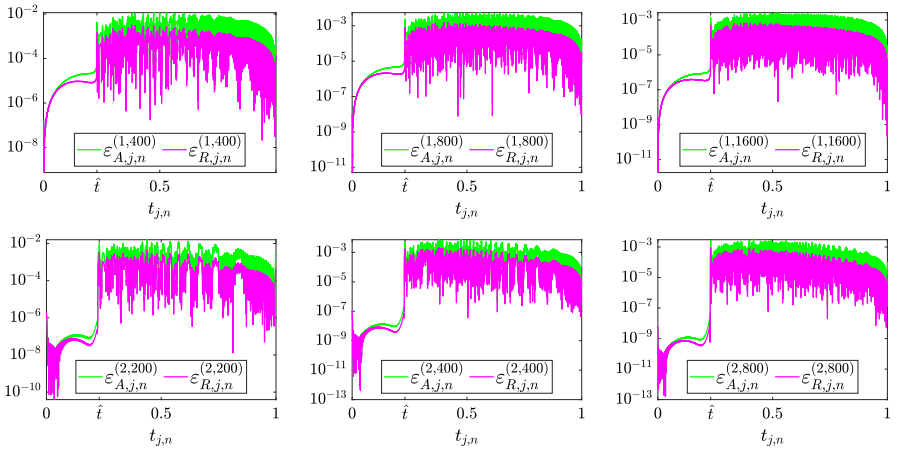

Fig. 5 Example 5.2 — $(\varepsilon_{A,1,n}^{(\alpha,n_0)}, \ldots, \varepsilon_{A,n,n}^{(\alpha,n_0)})$ and $(\varepsilon_{R,1,n}^{(\alpha,n_0)}, \ldots, \varepsilon_{R,n,n}^{(\alpha,n_0)})$ versus $(t_{1,n}, \ldots, t_{n,n})$ in the case $n = 10000$. First row: $\alpha = 1$ and $n_0 = 400, 800, 1600$. Second row: $\alpha = 2$ and $n_0 = 200, 400, 800$

$$X_n = \begin{bmatrix} a_{\frac{1}{2}} + a_{\frac{3}{2}} & -a_{\frac{3}{2}} & & & & \\ -a_{\frac{3}{2}} & a_{\frac{3}{2}} + a_{\frac{5}{2}} & -a_{\frac{5}{2}} & & & \\ & -a_{\frac{5}{2}} & \ddots & \ddots & & \\ & & \ddots & \ddots & -a_{n-\frac{1}{2}} & \\ & & & -a_{n-\frac{1}{2}} & a_{n-\frac{1}{2}} + a_{n+\frac{1}{2}} & \end{bmatrix},$$

where $a_i = a(x_i)$ for all i in the real interval $[0, n+1]$; see [36, Section 10.5.1] for more details. The matrix-sequence $\{X_n\}_n$ is spectrally real and we have $\{X_n\}_n \sim_{\text{GLT},\lambda} f(x,\theta) = a(x)(2 - 2\cos\theta)$ whenever a is continuous a.e. on $(0, 1)$; see [10].

- In Fig. 6, we collect the results related to item (a) for $\alpha = 3$ and for two choices of the coefficient $a(x)$, namely $a(x) = x+1$ (left column) and $a(x) = \mathrm{e}^{-x}\sin(\frac{\pi}{2}x) + \mathrm{e}^{x}\cos(\frac{\pi}{2}x)$ (right column). In both cases, for each fixed $k = 0, \ldots, \alpha$, the plots obtained for different values of n_0 seem to overlap, but the reproduction of the graph of a function c_k is not clearly visible for $k \geq 2$. In particular, there are "problems" around $t = 0.64,\ 1$ for $a(x) = x + 1$, and around $t = 0.38,\ 0.72$ for $a(x) = \mathrm{e}^{-x}\sin(\frac{\pi}{2}x) + \mathrm{e}^{x}\cos(\frac{\pi}{2}x)$. Therefore, we expect that our method will encounter difficulties in the approximation of the eigenvalues $\lambda_j(X_n)$ corresponding to the "critical" points $t_{j,n}$, i.e., the points $t_{j,n} \approx 0.64,\ 1$ for $a(x) = x+1$ and the points $t_{j,n} \approx 0.38,\ 0.72$ for $a(x) = \mathrm{e}^{-x}\sin(\frac{\pi}{2}x) + \mathrm{e}^{x}\cos(\frac{\pi}{2}x)$.
- In Table 4 and Fig. 7, we collect the results related to item (b) for $n = 20000$. We see from Table 4 that the CPU times of Algorithm 4.1 are comparable to the CPU time of MATLAB's `eig` function, the latter being extremely efficient for real sparse symmetric matrices. Concerning the eigenvalue errors shown in Fig. 7, we may say that Algorithm 4.1 works well for all eigenvalues, except for

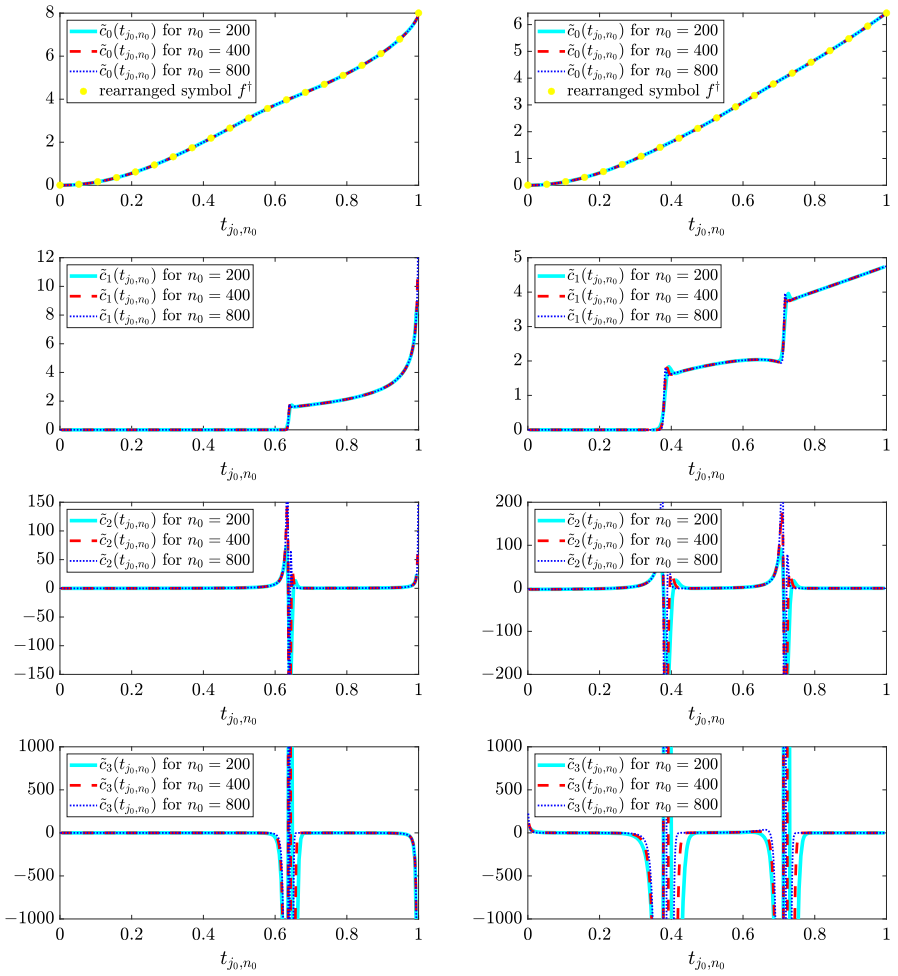

Fig. 6 Example 5.3 — $(\tilde{c}_k(t_{1,n_0}), \ldots, \tilde{c}_k(t_{n_0,n_0}))$ versus $(t_{1,n_0}, \ldots, t_{n_0,n_0})$ for $k = 0, \ldots, \alpha$ and different values of n_0 in the case $\alpha = 3$. Left column: $a(x) = x+1$. Right column: $a(x) = \mathrm{e}^{-x}\sin(\frac{\pi}{2}x)+\mathrm{e}^{x}\cos(\frac{\pi}{2}x)$

  - the smallest eigenvalues (this is due to the fact that the minimal eigenvalue of X_n is very small and converges to 0 as $n \to \infty$);
  - the eigenvalues $\lambda_j(X_n)$ corresponding to the critical points $t_{j,n}$.

- In Table 5, we collect the results related to item (c) for $n = 20000$ and $S = \{18000, \ldots, 20000\}$. We see from Table 5 that the CPU times of Algorithm 4.1 are lower than that of MATLAB's `eigs` function.

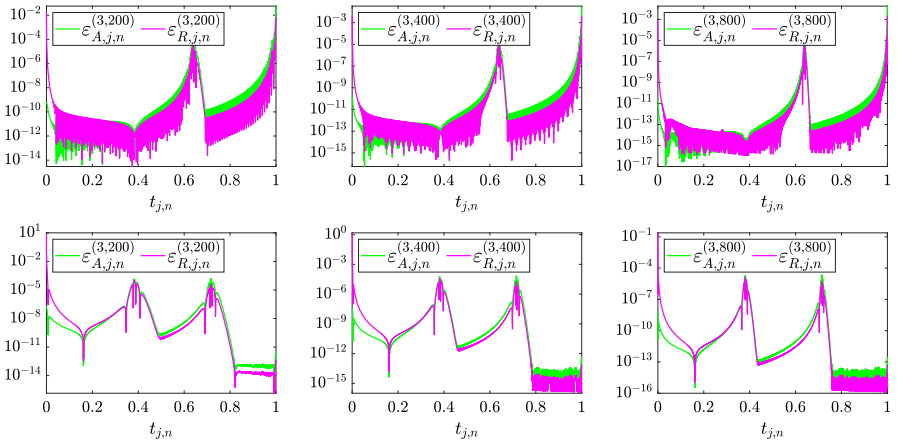

Fig. 7 Example 5.3 — $(\varepsilon^{(\alpha,n_0)}_{A,1,n}, \ldots, \varepsilon^{(\alpha,n_0)}_{A,n,n})$ and $(\varepsilon^{(\alpha,n_0)}_{R,1,n}, \ldots, \varepsilon^{(\alpha,n_0)}_{R,n,n})$ versus $(t_{1,n}, \ldots, t_{n,n})$ in the case $n = 20000$. First row: $a(x) = x + 1$, $\alpha = 3$ and $n_0 = 200, 400, 800$. Second row: $a(x) = \mathrm{e}^{-x}\sin(\frac{\pi}{2}x) + \mathrm{e}^{x}\cos(\frac{\pi}{2}x)$, $\alpha = 3$ and $n_0 = 200, 400, 800$

Table 4 Example 5.3 — CPU times for computing all the eigenvalues of X_n in the case $n = 20000$. The corresponding eigenvalue errors are shown in Fig. 7

Method	CPU time for $a(x) = x + 1$	CPU time for $a(x) = \mathrm{e}^{-x}\sin(\frac{\pi}{2}x) + \mathrm{e}^{x}\cos(\frac{\pi}{2}x)$
Algor. 4.1 ($\alpha = 3,\ n_0 = 200$)	1.22	1.23
Algor. 4.1 ($\alpha = 3,\ n_0 = 400$)	1.30	1.34
Algor. 4.1 ($\alpha = 3,\ n_0 = 800$)	1.77	1.69
MATLAB's `eig` function	4.19	3.92

Table 5 Example 5.3 — CPU times for computing the largest eigenvalues of X_n corresponding to the indices $\{18000, \ldots, 20000\}$ in the case $n = 20000$

Method	CPU time for $a(x) = x + 1$	CPU time for $a(x) = \mathrm{e}^{-x}\sin(\frac{\pi}{2}x) + \mathrm{e}^{x}\cos(\frac{\pi}{2}x)$
Algor. 4.1 ($\alpha = 3,\ n_0 = 200$)	0.16	0.16
Algor. 4.1 ($\alpha = 3,\ n_0 = 400$)	0.28	0.27
Algor. 4.1 ($\alpha = 3,\ n_0 = 800$)	0.75	0.71
MATLAB's `eigs` function	567.48	675.51

Example 5.4 (finite element matrices) Consider the following system of differential equations:

$$\begin{cases} -(a(x)u'(x))' + v'(x) = f(x), & x \in (0,1), \\ -u'(x) - \rho v(x) = g(x), & x \in (0,1), \\ u(0) = 0, \quad u(1) = 0, \\ v(0) = 0, \quad v(1) = 0, \end{cases}$$

where $\rho \in \mathbb{R}$ is a constant. In the classical finite element method based on the piecewise linear hat functions $\varphi_1, \ldots, \varphi_n$ defined on the uniform grid $x_i = \frac{i}{n+1}$, $i = 0, \ldots, n+1$, the computation of the numerical solution reduces to solving a linear system whose coefficient matrix is given by

$$A_{2n} = \begin{bmatrix} K_n & H_n \\ H_n^T & -\rho M_n \end{bmatrix},$$

where

$$\begin{aligned} K_n &= \left[\int_0^1 a(x)\varphi_j'(x)\varphi_i'(x)\mathrm{d}x \right]_{i,j=1}^n, \\ H_n &= \left[\int_0^1 \varphi_j'(x)\varphi_i(x)\mathrm{d}x \right]_{i,j=1}^n = -\mathrm{i}\, T_n(\sin\theta), \\ M_n &= \left[\int_0^1 \varphi_j(x)\varphi_i(x)\mathrm{d}x \right]_{i,j=1}^n = \frac{1}{3(n+1)} T_n(2+\cos\theta); \end{aligned}$$

see [36, Section 10.6.2] for more details. Since A_{2n} enjoys a so-called saddle-point structure [15, p. 3], a key tool for the numerical solution of a linear system with matrix A_{2n} is the Schur complement of A_{2n}, i.e.,

$$S_n = \rho M_n + H_n^T K_n^{-1} H_n = \frac{\rho}{3(n+1)} T_n(2+\cos\theta) + T_n(\sin\theta) K_n^{-1} T_n(\sin\theta);$$

see [15, Section 5]. Let $X_n = (n+1)S_n$ be the normalized Schur complement. The matrix-sequence $\{X_n\}_n$ is spectrally real and we have

$$\{X_n\}_n \sim_{\mathrm{GLT},\lambda} f(x,\theta) = \frac{\rho}{3}(2+\cos\theta) + \frac{1+\cos\theta}{2\,a(x)}$$

whenever $a \in L^1((0,1))$ and $a \neq 0$ a.e. on $(0,1)$; see [36, Theorem 10.13].

- In Fig. 8, we collect the results related to item (a) for $\alpha = 3$ and for two choices of $a(x)$, ρ, namely $a(x) = 1+\sqrt{x}$, $\rho = 3.7$ (left column) and $a(x) = 2\,\chi_{(0,1/2]}(x) + \chi_{(1/2,1)}(x)$, $\rho = 1$ (right column), where χ_E denotes the characteristic (indicator) function of the set E. In both cases, for each fixed $k = 0, \ldots, \alpha$, the plots obtained for different values of n_0 seem to overlap, but it is clear that we cannot assume the validity of the working hypothesis, especially for the second choice of $a(x)$, which is a discontinuous function.
- In Table 6 and Fig. 9, we collect the results related to item (b) for $n = 10000$. We see from Table 6 that the CPU times of Algorithm 4.1 are lower than that of MATLAB's `eig` function. Concerning the eigenvalue errors shown in Fig. 9, we can say that Algorithm 4.1 works reasonably well for most eigenvalues and for both choices of $a(x)$, ρ. Once again, we observe that the difficulties are in the

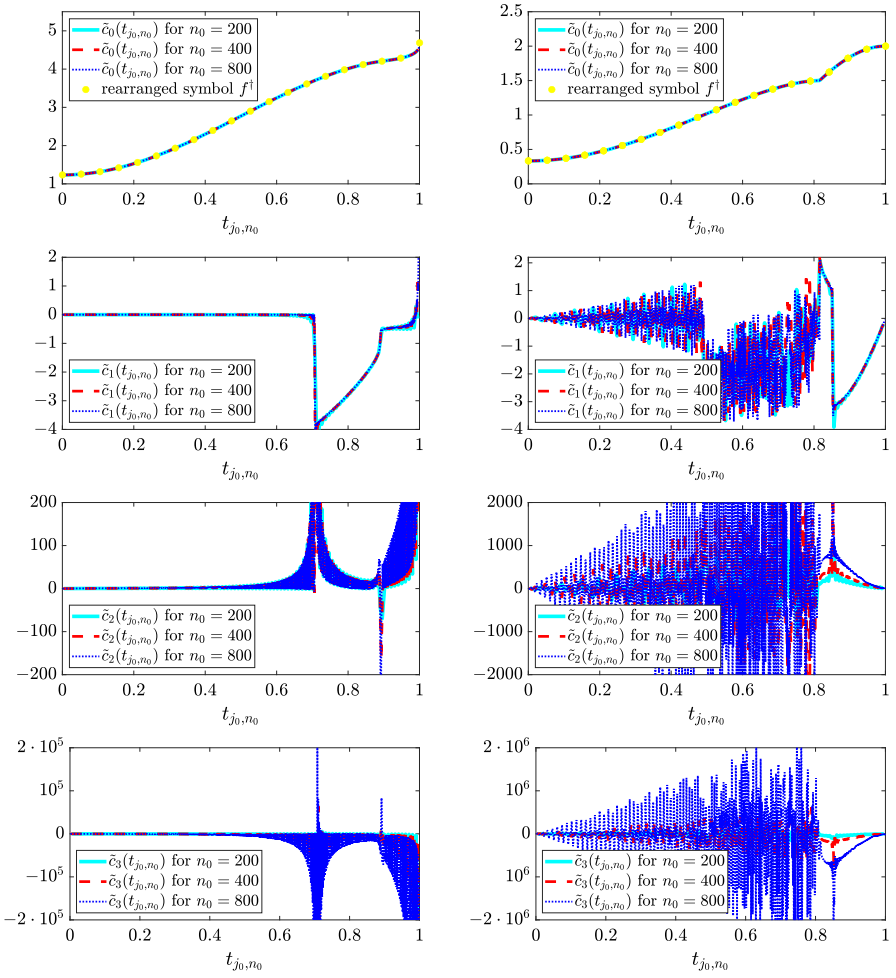

Fig. 8 Example 5.4 — $(\tilde{c}_k(t_{1,n_0}), \ldots, \tilde{c}_k(t_{n_0,n_0}))$ versus $(t_{1,n_0}, \ldots, t_{n_0,n_0})$ for $k = 0, \ldots, \alpha$ and different values of n_0 in the case $\alpha = 3$. Left column: $a(x) = 1+\sqrt{x}$, $\rho = 3.7$. Right column: $a(x) = 2\,\chi_{(0,1/2]}(x)+\chi_{(1/2,1)}(x)$, $\rho = 1$

approximation of the eigenvalues $\lambda_j(X_n)$ corresponding to the critical points $t_{j,n}$ where $\tilde{c}_1, \tilde{c}_2, \tilde{c}_3$ have problems.

In the last two examples (Examples 5.5–5.6), we focus on the numerical solution of variable-coefficient eigenvalue problems. Note that eigenvalue problems can be considered as an important application where all the eigenvalues of the resulting discretization matrix have to be computed as they represent the numerical solution of the problem, i.e., the computed approximations for the exact eigenvalues; see [20, 38].

Table 6 Example 5.4 — CPU times for computing all the eigenvalues of X_n in the case $n = 10000$. The corresponding eigenvalue errors are shown in Fig. 9

Method	CPU time for $\begin{cases} a(x) = 1 + \sqrt{x} \\ \rho = 3.7 \end{cases}$	CPU time for $\begin{cases} a(x) = 2\,\chi_{(0,1/2]}(x) + \chi_{(1/2,1)}(x) \\ \rho = 1 \end{cases}$
Algor. 4.1 ($\alpha = 3,\ n_0 = 100$)	0.90	0.86
Algor. 4.1 ($\alpha = 3,\ n_0 = 200$)	2.22	2.14
Algor. 4.1 ($\alpha = 3,\ n_0 = 400$)	13.53	13.22
MATLAB's `eig` function	307.34	302.79

Example 5.5 (B-spline Galerkin matrices) Consider the following eigenvalue problem:

$$\begin{cases} -(a(x)u_j'(x))' = \lambda_j b(x) u_j(x), & x \in (0,1), \\ u_j(0) = 0, \quad u_j(1) = 0. \end{cases}$$

If we apply the Galerkin method based on the B-splines $B_{1,p}, \ldots, B_{n,p}$ of degree p defined over the uniform grid $x_i = \frac{i}{n-p+2}$, $i = 0, \ldots, n-p+2$, and vanishing on the boundary of $[0, 1]$, then the computation of the numerical solution reduces to solving a classical discrete eigenvalue problem whose matrix is given by $L_n = M_n^{-1} K_n$, where

$$K_n = \left[\int_0^1 a(x) B_{j,p}'(x) B_{i,p}'(x) \mathrm{d}x \right]_{i,j=1}^{n}, \qquad M_n = \left[\int_0^1 b(x) B_{j,p}(x) B_{i,p}(x) \mathrm{d}x \right]_{i,j=1}^{n};$$

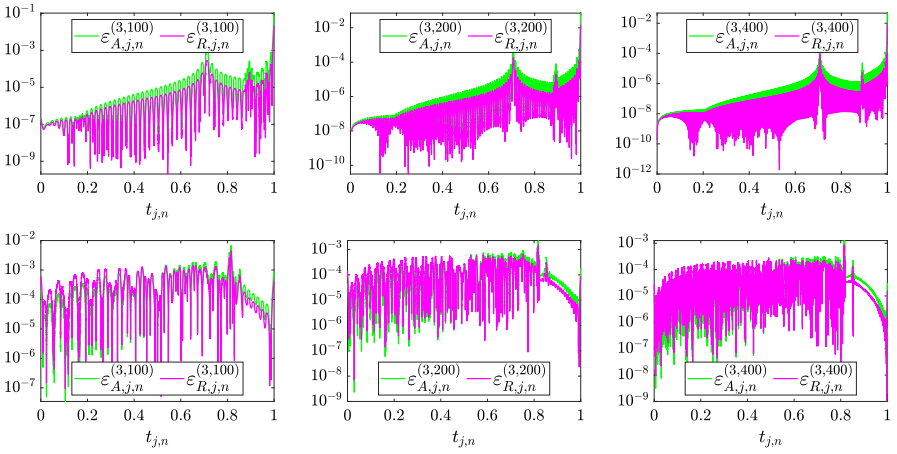

Fig. 9 Example 5.4 — $(\varepsilon_{A,1,n}^{(\alpha,n_0)}, \ldots, \varepsilon_{A,n,n}^{(\alpha,n_0)})$ and $(\varepsilon_{R,1,n}^{(\alpha,n_0)}, \ldots, \varepsilon_{R,n,n}^{(\alpha,n_0)})$ versus $(t_{1,n}, \ldots, t_{n,n})$ in the case $n = 10000$. First row: $a(x) = 1 + \sqrt{x}$, $\rho = 3.7$, $\alpha = 3$ and $n_0 = 100, 200, 400$. Second row: $a(x) = 2\,\chi_{(0,1/2]}(x) + \chi_{(1/2,1)}(x)$, $\rho = 1$, $\alpha = 3$ and $n_0 = 100, 200, 400$

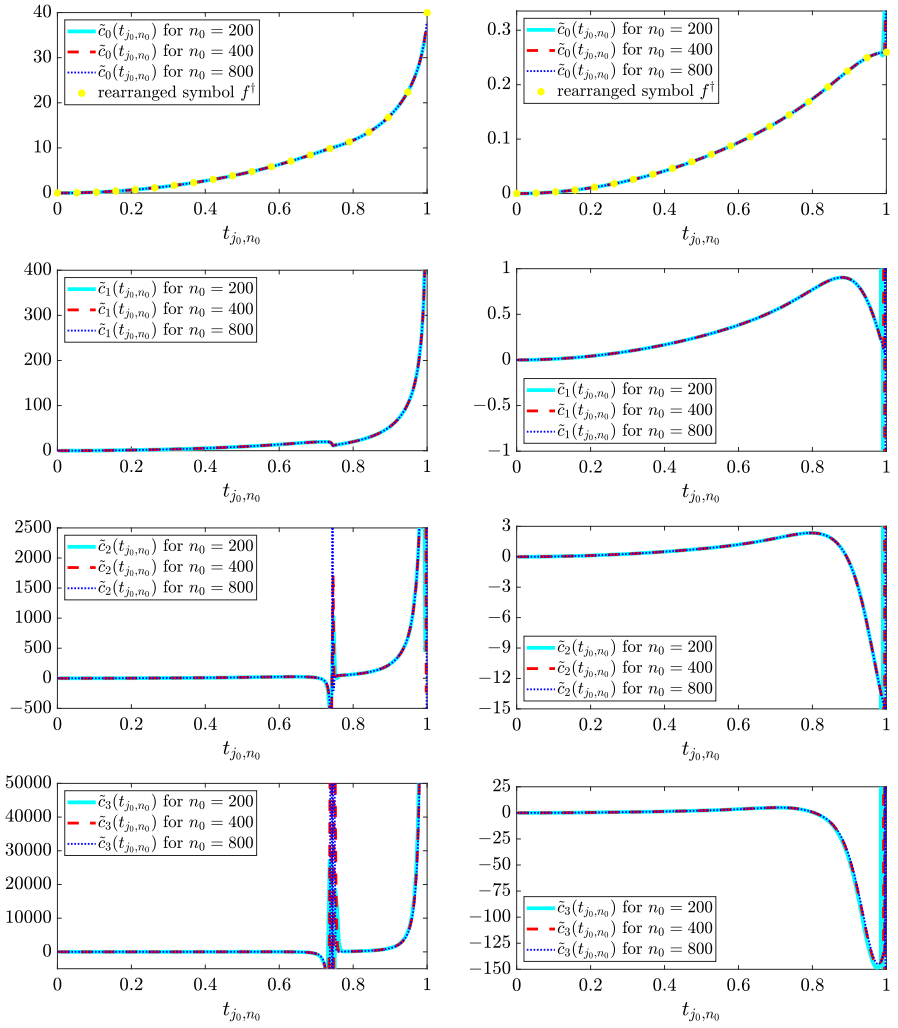

Fig. 10 Example 5.5 — $(\tilde{c}_k(t_{1,n_0}), \ldots, \tilde{c}_k(t_{n_0,n_0}))$ versus $(t_{1,n_0}, \ldots, t_{n_0,n_0})$ for $k = 0, \ldots, \alpha$ and different values of n_0 in the case $\alpha = 3$. Left column: $p = 2$, $a(x) = 1 + x^2$, $b(x) = 1 - 0.5\,x$. Right column: $p = 3$, $a(x) = 2.1 + 1.05\,x$, $b(x) = 80 + 40\,x$

see [36, Sections 10.7.2–10.7.3] for more details. Let $X_n = (n-p+2)^{-2} L_n$. Assuming that $a, b \in L^1((0,1))$ and $a, b > 0$ a.e. on $(0,1)$, the matrix-sequence $\{X_n\}_n$ is spectrally real and we have

$$\{X_n\}_n \sim_{\mathrm{GLT},\,\lambda} f(x,\theta) = \frac{a(x)}{b(x)} \frac{f_p(\theta)}{h_p(\theta)},$$

where f_p and h_p are suitable functions; see [36, Theorem 10.16].

Table 7 Example 5.5 — CPU times for computing all the eigenvalues of X_n in the case $n = 10000$. The corresponding eigenvalue errors are shown in Fig. 11

Method	CPU time for $\begin{cases} p=2 \\ a(x)=1+x^2 \\ b(x)=1-0.5\,x \end{cases}$	CPU time for $\begin{cases} p=3 \\ a(x)=2.1+1.05\,x \\ b(x)=80+40\,x \end{cases}$
Algor. 4.1 ($\alpha = 3,\ n_0 = 100$)	0.78	0.81
Algor. 4.1 ($\alpha = 3,\ n_0 = 200$)	2.16	2.17
Algor. 4.1 ($\alpha = 3,\ n_0 = 400$)	13.48	14.17
MATLAB's `eig` function	303.29	307.51

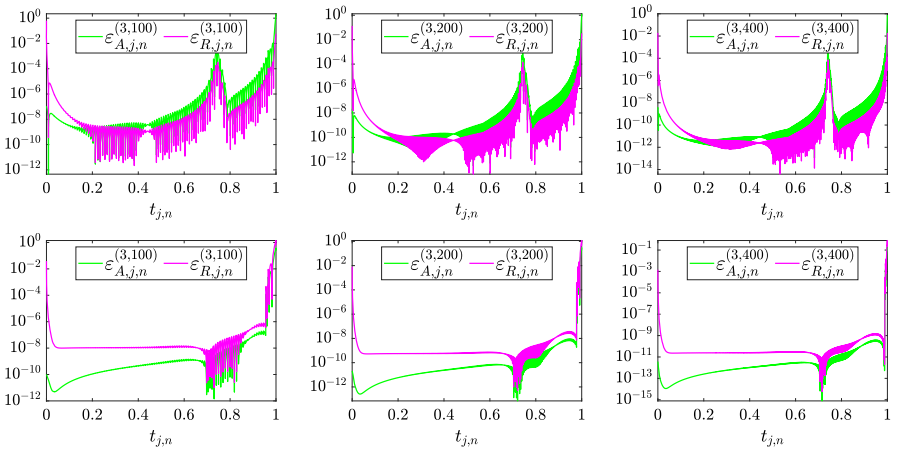

Fig. 11 Example 5.5 — $(\varepsilon_{A,1,n}^{(\alpha,n_0)}, \ldots, \varepsilon_{A,n,n}^{(\alpha,n_0)})$ and $(\varepsilon_{R,1,n}^{(\alpha,n_0)}, \ldots, \varepsilon_{R,n,n}^{(\alpha,n_0)})$ versus $(t_{1,n}, \ldots, t_{n,n})$ in the case $n = 10000$. First row: $p = 2$, $a(x) = 1 + x^2$, $b(x) = 1 - 0.5\,x$, $\alpha = 3$ and $n_0 = 100, 200, 400$. Second row: $p = 3$, $a(x) = 2.1 + 1.05\,x$, $b(x) = 80 + 40\,x$, $\alpha = 3$ and $n_0 = 100, 200, 400$

– In Fig. 10, we collect the results related to item (a) for $\alpha = 3$ and for two choices of p, $a(x)$, $b(x)$ inspired by [38, Section 3.3.1], namely $p = 2$, $a(x) = 1 + x^2$, $b(x) = 1 - 0.5\,x$ (left column) and $p = 3$, $a(x) = 2.1 + 1.05\,x$, $b(x) = 80 + 40\,x$ (right column). In both cases, for each fixed $k = 0, \ldots, \alpha$, the plots obtained for different values of n_0 overlap. The second choice highlights an aspect that did not appear before: the presence of a few large outlier eigenvalues, which are responsible for the slight mismatch between $\tilde{c}_0$ and $f^\dagger$ at the end of the interval $(0, 1)$.
– In Table 7 and Fig. 11, we collect the results related to item (b) for $n = 10000$. We see from Table 7 that the CPU times of Algorithm 4.1 are lower than that of MATLAB's `eig` function. Concerning the eigenvalue errors shown in Fig. 11, we may say that Algorithm 4.1 works well for most eigenvalues and for both choices of p, $a(x)$, $b(x)$. However, serious difficulties are observed in the approximation of the eigenvalues $\lambda_j(X_n)$ corresponding to the critical points $t_{j,n}$ where $\tilde{c}_1, \tilde{c}_2, \tilde{c}_3$

have problems. These critical points include in particular those corresponding to the outliers for the second choice of p, $a(x)$, $b(x)$, i.e., the points $t_{j,n} \approx 1$, for which the approximation is not satisfactory.

Remark 5.3 Example 5.5 highlights an aspect that is worth to be emphasized. Suppose that we interpret Algorithm 4.1 as an eigenpredictor, i.e., as an algorithm whose purpose is to give a (fast) prediction of the eigenvalues of X_n. Then, already for $\alpha = 3$ and $n_0 = 200$, such eigenpredictor involving not only c_0 but also the "higher-order symbols" c_1, c_2, c_3 is more accurate (especially for small eigenvalues) than the analytical prediction in [38] based on the sole rearranged symbol $c_0 = f^\dagger$. In this regard, the present paper provides a positive answer to the question raised in [38, Section 4.2].

Example 5.6 (B-spline collocation matrices) Consider again the eigenvalue problem of Example 5.5. If we apply the collocation method based on the B-splines $B_{1,p}, \dots, B_{n,p}$ of degree p defined on the uniform grid $x_i = \frac{i}{n-p+2}$, $i = 0, \dots, n-p+2$, and vanishing on the boundary of $[0, 1]$, and if we use as collocation points the related Greville abscissae $\xi_{1,p}, \dots, \xi_{n,p}$, a common choice in the literature [5], then the computation of the numerical solution reduces to solving a classical discrete eigenvalue problem whose matrix is given by $L_n = M_n^{-1} K_n$, where

$$K_n = \Big[- a(\xi_{i,p}) B''_{j,p}(\xi_{i,p}) \Big]_{i,j=1}^{n}, \qquad M_n = \Big[b(\xi_{i,p}) B_{j,p}(\xi_{i,p}) \Big]_{i,j=1}^{n};$$

see [36, Sections 10.7.1 and 10.7.3] for more details. Let $X_n = (n-p+2)^{-2} L_n$. Based on the theory of GLT sequences and [36, Section 10.7.1], we may expect that, under suitable assumptions on the coefficients a, b—e.g., $a, b \in C([0, 1])$ and $a, b > 0$ a.e. on $[0, 1]$—the matrix-sequence $\{X_n\}_n$ is spectrally real and

$$\{X_n\}_n \sim_{\mathrm{GLT},\,\lambda} f(x, \theta) = \frac{a(x)}{b(x)} \frac{f_p(\theta)}{h_p(\theta)},$$

where f_p and h_p are the functions in [36, eqs. (10.147) and (10.149)]. This time, however, contrary to all the previous examples, a formal proof of the fact that $\{X_n\}_n$ is spectrally real and $\{X_n\}_n \sim_{\mathrm{GLT},\,\lambda} f(x, \theta)$ is not available. A numerical evidence of the fact that $\{X_n\}_n$ is spectrally real will be obtained by computing the eigenvalues of X_n for different small values of n and observing that they are real. Note that this computation is anyhow necessary for applying Algorithm 4.1. Moreover, when plotting $(\tilde{c}_0(t_{1,n_0}), \dots, \tilde{c}_0(t_{n_0,n_0}))$ versus $(t_{1,n_0}, \dots, t_{n_0,n_0})$ as in all the previous examples, the resulting graph (up to a few outliers as in Example 5.5) will provide us with the monotone rearrangement of the spectral symbol of $\{X_n\}_n$, and we will check that it coincides with the monotone rearrangement of $f(x, \theta)$, thus confirming numerically that $f(x, \theta)$ is indeed the spectral symbol of $\{X_n\}_n$ as expected. This shows in particular that the numerical computation of the monotone rearrangement of the spectral symbol can be achieved through the matrix-less approach described in this paper even when the spectral symbol is not known; see also [35, Section 4].

– In Fig. 12, we collect the results related to item (a) for $\alpha = 3$ and for two choices of $p, a(x), b(x)$, namely $p = 4$, $a(x) = 1 - x + x^2$, $b(x) = \mathrm{e}^x$ (left column) and

Table 8 Example 5.6 — CPU times for computing all the eigenvalues of X_n in the case $n = 10000$. The corresponding eigenvalue errors are shown in Fig. 13

Method	CPU time for $\begin{cases} p = 4 \\ a(x) = 1 - x + x^2 \\ b(x) = \mathrm{e}^x \end{cases}$	CPU time for $\begin{cases} p = 5 \\ a(x) = 1 + \cos(\frac{\pi}{2}x) \\ b(x) = 3 + \sin(\frac{\pi}{2}x) \end{cases}$
Algor. 4.1 ($\alpha = 3,\ n_0 = 100$)	0.84	0.80
Algor. 4.1 ($\alpha = 3,\ n_0 = 200$)	2.35	2.48
Algor. 4.1 ($\alpha = 3,\ n_0 = 400$)	14.30	19.84
MATLAB's `eig` function	306.75	573.91

$p = 5$, $a(x) = 1 + \cos(\frac{\pi}{2}x)$, $b(x) = 3 + \sin(\frac{\pi}{2}x)$ (right column). In both cases, for each fixed $k = 0, \ldots, \alpha$, the plots obtained for different values of n_0 overlap. For the second choice, we also note the presence of a few large outlier eigenvalues, which are responsible for the slight mismatch between $\tilde{c}_0$ and $f^\dagger$ at the end of the interval $(0, 1)$.

– In Table 8 and Fig. 13, we collect the results related to item (b) for $n = 10000$. We see from Table 8 that the CPU times of Algorithm 4.1 are lower than that of MATLAB's `eig` function. Concerning the eigenvalue errors shown in Fig. 13, we can say that Algorithm 4.1 works well for most eigenvalues and for both choices of $p, a(x), b(x)$. However, serious difficulties are observed in the approximation of the eigenvalues $\lambda_j(X_n)$ corresponding to the critical points $t_{j,n}$ where $\tilde{c}_1, \tilde{c}_2, \tilde{c}_3$ have problems. These critical points include in particular those corresponding to the outliers for the second choice of $p, a(x), b(x)$, i.e., the points $t_{j,n} \approx 1$, for which the approximation is bad.

6 Conclusions and perspectives

We have assumed as a working hypothesis that the eigenvalues of the matrices X_n, forming a spectrally real GLT sequence $\{X_n\}_n$, are given by a regular expansion. Based on the working hypothesis, we have proposed a method for computing approximations of the eigenvalues of X_n for large n and we have provided a theoretical analysis of its convergence. Our method does not require the knowledge of the spectral symbol, in contrast to earlier works on the matrix-less approach. The plausibility of the working hypothesis as well as the performance of the method have been illustrated through numerical experiments. The experiments testify that reasoning as if the working hypothesis were true leads to quite satisfactory results in terms of CPU times and eigenvalue approximations, thus showing numerically that the spectra of the matrices X_n forming a GLT sequence are more "regular" than one might expect. Some concluding remarks on the proposed approach are in order.

– Based on the numerical experiments, we can say that the eigenvalue approximations produced by our method are accurate for the eigenvalues $\lambda_j(X_n)$

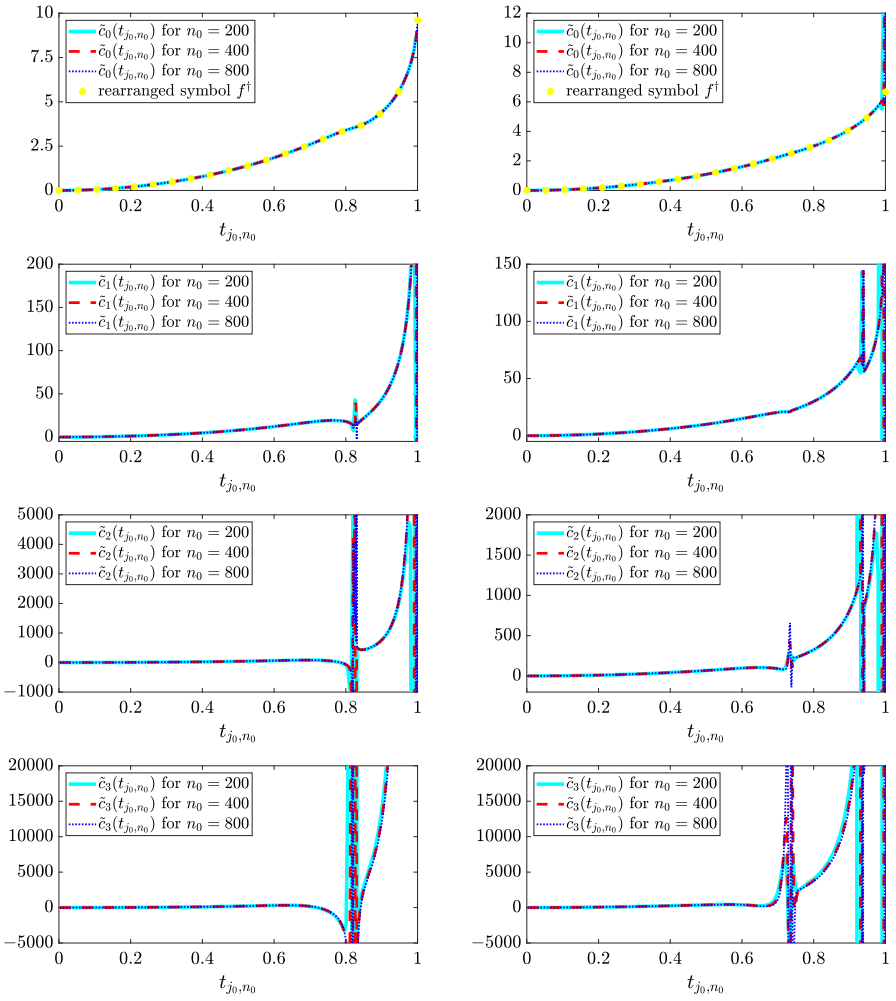

Fig. 12 Example 5.6 — $(\tilde{c}_k(t_{1,n_0}), \ldots, \tilde{c}_k(t_{n_0,n_0}))$ versus $(t_{1,n_0}, \ldots, t_{n_0,n_0})$ for $k = 0, \ldots, \alpha$ and different values of n_0 in the case $\alpha = 3$. Left column: $p = 4$, $a(x) = 1 - x + x^2$, $b(x) = \mathrm{e}^x$. Right column: $p = 5$, $a(x) = 1 + \cos(\frac{\pi}{2}x)$, $b(x) = 3 + \sin(\frac{\pi}{2}x)$

corresponding to the "nice" points $t_{j,n}$ where the expansion "works", i.e., the functions $\tilde{c}_k$ exhibit a "smooth" behavior. On the contrary, the approximations might not be accurate for the eigenvalues $\lambda_j(X_n)$ corresponding to the "critical" points $t_{j,n}$ where the expansion "fails", i.e., the functions $\tilde{c}_k$ exhibit a "wild" behavior. On the basis of this observation, one can first analyze the expansion functions $\tilde{c}_k$, and then predict a better approximation of the eigenvalues corresponding to the "nice" points compared to the eigenvalues corresponding to the "critical" points.

– The proposed eigenvalue approximation method has the following weaknesses:

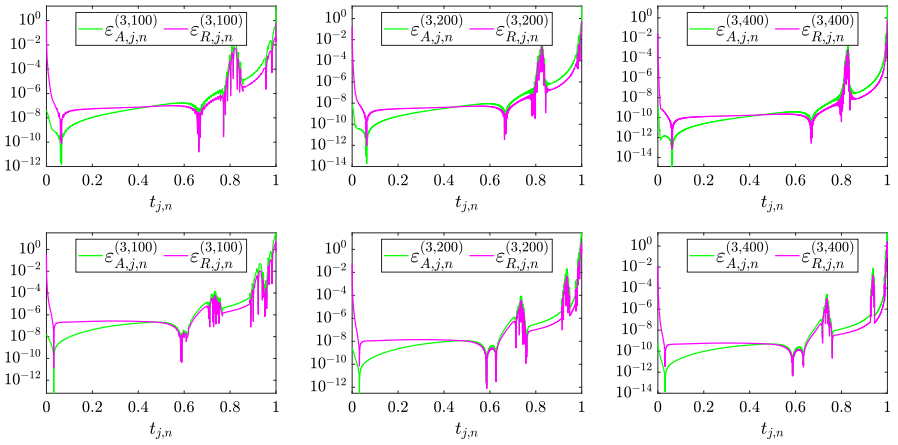

Fig. 13 Example 5.6 — $(\varepsilon_{A,1,n}^{(\alpha,n_0)}, \ldots, \varepsilon_{A,n,n}^{(\alpha,n_0)})$ and $(\varepsilon_{R,1,n}^{(\alpha,n_0)}, \ldots, \varepsilon_{R,n,n}^{(\alpha,n_0)})$ versus $(t_{1,n}, \ldots, t_{n,n})$ in the case $n = 10000$. First row: $p = 4$, $a(x) = 1 - x + x^2$, $b(x) = \mathrm{e}^x$, $\alpha = 3$ and $n_0 = 100, 200, 400$. Second row: $p = 5$, $a(x) = 1 + \cos(\frac{\pi}{2}x)$, $b(x) = 3 + \sin(\frac{\pi}{2}x)$, $\alpha = 3$ and $n_0 = 100, 200, 400$

  - As remarked in Example 5.1, since we cannot increase n_0 until some "stopping criterion" is satisfied, we cannot obtain an approximation of the eigenvalues within an a priori fixed tolerance.
  - We do not have a priori estimates, with explicit constants, on the approximation accuracy that we can achieve with this matrix-less approach.

- The proposed eigenvalue approximation method has the following strengths:
  - It is fast and does not need to construct the matrix X_n. Moreover, if we have to compute approximations of the eigenvalues of X_n and $X_{n'}$ for two large values n, n', we can compute the values $\tilde{c}_k(t_{1,n_0}), \ldots, \tilde{c}_k(t_{n_0,n_0})$, $k = 0, \ldots, \alpha$, once (first two steps of Algorithm 4.1), and then use them twice for computing the approximations of the eigenvalues of X_n and $X_{n'}$, without constructing neither X_n nor $X_{n'}$. On the contrary, a standard eigensolver should first construct both X_n and $X_{n'}$, which could be time-consuming.
  - As observed in Remark 5.3, our method yields a more accurate analytical prediction of the eigenvalues of X_n than the approach described in [38]. This is especially true for the small eigenvalues, which are often the object of interest in engineering applications.
- The eigenvalue approximations produced by our method can be used as initial guess for iterative refinement algorithms [30]. In this way, it is possible to get high precision eigenvalue approximations starting from a reliable initial guess obtained through our fast matrix-less approach.

We end with a few suggestions for possible future lines of research.

- The rule $n_k = 2^k(n_0+1)-1, k = 0, \ldots, \alpha$, is somehow a limitation of the proposed approach. Moreover, it has no intrinsic value: it is just a way to simplify the presentation. For instance, there would be no significant difference if the previous

rule were replaced with $n_k = 2^k n_0$, $k = 0, \dots, \alpha$, or other similar variants. A future research could take care of removing this limitation and cast the proposed eigenvalue approximation method into a more general framework.
– Extend the proposed method to multilevel block GLT sequences so as to include sequences of matrices arising from the discretization of multidimensional partial differential equations (PDEs) and systems of PDEs.

A Proofs of Theorems 3.1, 4.1, 4.2, 4.3

This appendix contains the proofs of Theorems 3.1, 4.1, 4.2, 4.3. While Theorem 3.1 is completely new, Theorems 4.1–4.3 are generalizations of [33, Theorems 1–3] as they address the more general case where $c_0 = f^\dagger$ is computed along with $c_1, \dots, c_\alpha$. Moreover, the proof of Theorem 4.1 is considerably simplified compared to its analogue in [33]. For the proof of Theorem 3.1, we need Lemmas A.1 and A.2; see [8, Lemma 3.1] for Lemma A.1 and [36, pp. 275–276] for Lemma A.2. In what follows, the characteristic function of the set E is denoted by χ_E and the Vandermonde matrix by $V(x_1, x_2, \dots, x_m) = \left[x_i^{j-1}\right]_{i,j=1}^m$.

Lemma A.1 *Let $f : (0,1) \to \mathbb{R}$ be measurable and let $\mathcal{G}_n = \{x_{i,n}\}_{i=1,\dots,n} \subset (0,1)$ be an asymptotically uniform grid in $(0,1)$, i.e., $\lim_{n\to\infty} \left(\max_{i=1,\dots,n} \left|x_{i,n} - \frac{i}{n}\right|\right) = 0$. Then, $\lim_{n\to\infty} \sum_{i=1}^n f(x_{i,n}) \chi_{[(i-1)/n,\, i/n)}(x) = f(x)$ for every continuity point x of f in $(0,1)$. In particular, if f is continuous a.e. on $(0,1)$ then the previous limit relation holds for almost every $x \in (0,1)$. Finally, if f is continuous a.e. and bounded on $(0,1)$ then $\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^n f(x_{i,n}) = \int_0^1 f(x)\mathrm{d}x$.*

Lemma A.2 *Let $u, v : (0,1) \to \mathbb{R}$ be monotone non-decreasing with $\int_0^1 F(u(t))\mathrm{d}t = \int_0^1 F(v(t))\mathrm{d}t$ for all $F \in C_c(\mathbb{C})$. Then, $u = v$ a.e. on $(0,1)$.*

Proof of Theorem 3.1 We prove the three properties of the theorem separately.

1. We prove the spectral distribution $\{X_n\}_n \sim_\lambda c_0$. By the working hypothesis,

$$\lambda_j(X_n) = c_0(t_{j,n}) + E_{j,n,0}, \qquad |E_{j,n,0}| \le C_0 h, \qquad j = 1, \dots, n. \tag{A.1}$$

For every $F \in C_c(\mathbb{C})$ and every n,

$$\left|\frac{1}{n}\sum_{j=1}^n F(\lambda_j(X_n)) - \int_0^1 F(c_0(t))\mathrm{d}t\right| \le \frac{1}{n}\sum_{j=1}^n |F(\lambda_j(X_n)) - F(c_0(t_{j,n}))| + \left|\frac{1}{n}\sum_{j=1}^n F(c_0(t_{j,n})) - \int_0^1 F(c_0(t))\mathrm{d}t\right|.$$

The first term in the right-hand side is bounded by $\omega_F(C_0 h)$, with ω_F being the modulus of continuity of F, and tends to 0 as $n \to \infty$. The second term in the right-hand side tends to 0 by Lemma A.1 since $F(c_0(t))$ is an a.e. continuous bounded

function on $(0,1)$ and the grid $\{t_{j,n} : j = 1,\dots,n\}$ is asymptotically uniform in $(0,1)$. We conclude that $\{X_n\}_n \sim_\lambda c_0$.

2. We prove that c_0 coincides a.e. with a non-decreasing function on $(0,1)$. Let $I_{j,n} = \left[\frac{j-1}{n}, \frac{j}{n}\right)$ for $j = 1,\dots,n$. In view of (A.1), we define $\lambda_n(t) = \sum_{j=1}^n \lambda_j(X_n)\chi_{I_{j,n}}(t) = c_{0,n}(t) + E_{0,n}(t)$, where $c_{0,n}(t) = \sum_{j=1}^n c_0(t_{j,n})\chi_{I_{j,n}}(t)$ and $E_{0,n}(t) = \sum_{j=1}^n E_{j,n,0}\chi_{I_{j,n}}(t)$. The function $\lambda_n : (0,1) \to \mathbb{R}$ is non-decreasing on $(0,1)$ as $\lambda_1(X_n) \le \cdots \le \lambda_n(X_n)$. Moreover, $\lambda_n \to c_0$ a.e. on $(0,1)$ because $c_{0,n} \to c_0$ a.e. on $(0,1)$ by Lemma A.1 and $|E_{n,0}(t)| \le C_0 h$ for all $t \in (0,1)$. As a consequence, c_0 is non-decreasing on a set $E \subseteq (0,1)$ of measure 1, since it is the limit a.e. on $(0,1)$ of a sequence of non-decreasing functions. Now, for every $t \in (0,1)$, both $E_t^- = (0,t] \cap E$ and $E_t^+ = [t,1) \cap E$ are non-empty since E is dense in $(0,1)$. Thus, by the monotonicity of c_0 on E, we have $-\infty < \sup_{u \in E_t^-} c_0(u) \le \inf_{u \in E_t^+} c_0(u) < \infty$, and the function $\hat{c}_0(t) = \inf_{u \in E_t^+} c_0(u)$ is well-defined and non-decreasing on $(0,1)$. Moreover, if $t \in E$ then $\hat{c}_0(t) = c_0(t)$. We conclude that c_0 coincides a.e. with a non-decreasing function on $(0,1)$.

3. Suppose that $\{X_n\}_n \sim_\lambda f$. By the relation $\{X_n\}_n \sim_\lambda c_0$ and (2.2), for every $F \in C_c(\mathbb{C})$ we have

$$\begin{aligned}\int_0^1 F(f^\dagger(t))\mathrm{d}t &= \frac{1}{\mu_k(D)}\int_D F(f(\boldsymbol{x}))\mathrm{d}\boldsymbol{x} = \lim_{n\to\infty}\frac{1}{n}\sum_{j=1}^n F(\lambda_j(X_n)) \\ &= \int_0^1 F(c_0(t))\mathrm{d}t = \int_0^1 F(\hat{c}_0(t))\mathrm{d}t,\end{aligned}$$

where $D \subset \mathbb{R}^k$ is the domain of f and $\hat{c}_0$ is a monotone non-decreasing function on $(0,1)$ that coincides a.e. with c_0 (such a function $\hat{c}_0$ exists by the previous statement 2). We conclude that $f^\dagger = \hat{c}_0 = c_0$ a.e. on $(0,1)$ by Lemma A.2. □

Proof of Theorem 4.1 Equations (4.1) and (4.3) can be rewritten as

$$\begin{aligned}V(h_0,h_1,\dots,h_\alpha)\boldsymbol{c}(j_0) &= \boldsymbol{\Lambda}(j_0) - \boldsymbol{E}_\alpha(j_0), \\ V(h_0,h_1,\dots,h_\alpha)\tilde{\boldsymbol{c}}(j_0) &= \boldsymbol{\Lambda}(j_0),\end{aligned} \tag{A.2}$$

where $\boldsymbol{c}(j_0) = [c_0(t_{j_0,n_0}),\dots,c_\alpha(t_{j_0,n_0})]^T$, $\tilde{\boldsymbol{c}}(j_0) = [\tilde{c}_0(t_{j_0,n_0}),\dots,\tilde{c}_\alpha(t_{j_0,n_0})]^T$, $\boldsymbol{\Lambda}(j_0) = [\lambda_{j_0,n_0}(X_{n_0}),\dots,\lambda_{j_\alpha,n_\alpha}(X_{n_\alpha})]^T$, and $\boldsymbol{E}_\alpha(j_0) = [E_{j_0,n_0,\alpha},\dots,E_{j_\alpha,n_\alpha,\alpha}]^T$. Taking into account that $h_k = 2^{-k}h_0$ for $k = 0,\dots,\alpha$, we have $V(h_0,h_1,\dots,h_\alpha) = V_\alpha \mathrm{diag}(1,h_0,h_0^2,\dots,h_0^\alpha)$, where $V_\alpha = V(1,2^{-1},\dots,2^{-\alpha})$. By (A.2),

$$\begin{aligned}\tilde{\boldsymbol{c}}(j_0) - \boldsymbol{c}(j_0) &= V(h_0,h_1,\dots,h_\alpha)^{-1}\boldsymbol{E}_\alpha(j_0) \\ &= \mathrm{diag}(1,h_0^{-1},h_0^{-2},\dots,h_0^{-\alpha})V_\alpha^{-1}\boldsymbol{E}_\alpha(j_0).\end{aligned}$$

By (4.2), $|(\boldsymbol{E}_\alpha(j_0))_k| = |E_{j_k,n_k,\alpha}| \le C_\alpha h_k^{\alpha+1} \le C_\alpha h_0^{\alpha+1}$ for $k = 0,\dots,\alpha$. Thus, for every $k = 0,\dots,\alpha$,

$$|\tilde{c}_k(t_{j_0,n_0}) - c_k(t_{j_0,n_0})| = |(\tilde{\boldsymbol{c}}(j_0) - \boldsymbol{c}(j_0))_k| \\ \le h_0^{-k} \|V_\alpha^{-1} \boldsymbol{E}_\alpha(j_0)\|_\infty \le h_0^{-k} \|V_\alpha^{-1}\|_\infty C_\alpha h_0^{\alpha+1},$$

and Theorem 4.1 is proved with $A_\alpha = C_\alpha \|V_\alpha^{-1}\|_\infty$. □

Remark A.1 The constant $A_\alpha = C_\alpha \|V_\alpha^{-1}\|_\infty$ in Theorem 4.1 can be replaced with the constant $A_\alpha = C_\alpha \|V_\alpha^{-1} \mathrm{diag}(1, 2^{-(\alpha+1)}, 2^{-2(\alpha+1)}, \ldots, 2^{-\alpha(\alpha+1)})\|_\infty$, which is smaller because $\|V_\alpha^{-1}\|_\infty \ge 2^\alpha \to \infty$ and $\sup_{\alpha\in\mathbb{N}} \|V_\alpha^{-1} \mathrm{diag}(1, 2^{-(\alpha+1)}, 2^{-2(\alpha+1)}, \ldots, 2^{-\alpha(\alpha+1)})\|_\infty \in (6, 7)$; see [9, Remark A.3 and Appendix B].

Proof of Theorem 4.2 Let $L_1, \ldots, L_{\beta_k}$ be the Lagrange polynomials associated with $t^{(1)}, \ldots, t^{(\beta_k)}$, i.e.,

$$L_r(t) = \prod_{\substack{s=1 \\ s\neq r}}^{\beta_k} \frac{t - t^{(s)}}{t^{(r)} - t^{(s)}}, \qquad r = 1, \ldots, \beta_k.$$

The interpolation polynomials of the two data sets $(t^{(1)}, \tilde{c}_k(t^{(1)})), \ldots, (t^{(\beta_k)}, \tilde{c}_k(t^{(\beta_k)}))$ and $(t^{(1)}, c_k(t^{(1)})), \ldots, (t^{(\beta_k)}, c_k(t^{(\beta_k)}))$ are given by, respectively, $\tilde{c}_{k,j}(t) = \sum_{r=1}^{\beta_k} \tilde{c}_k(t^{(r)}) L_r(t)$ and $c_{k,j}(t) = \sum_{r=1}^{\beta_k} c_k(t^{(r)}) L_r(t)$. Since $t^{(1)}, \ldots, t^{(\beta_k)}$ are β_k distinct points from $\{t_{1,n_0}, \ldots, t_{n_0,n_0}\}$ which are closest to $t_{j,n}$, the length of the smallest interval I containing the nodes $t^{(1)}, \ldots, t^{(\beta_k)}$ and the point $t_{j,n}$ is bounded by $\beta_k h_0$. Hence, by Theorem 4.1, for all $t \in I$ we have

$$|\tilde{c}_{k,j}(t) - c_{k,j}(t)| \le \sum_{r=1}^{\beta_k} |\tilde{c}_k(t^{(r)}) - c_k(t^{(r)})| \, |L_r(t)| \\ \le \sum_{r=1}^{\beta_k} A_\alpha h_0^{\alpha-k+1} \prod_{\substack{s=1 \\ s\neq r}}^{\beta_k} \frac{\beta_k h_0}{h_0} = A_\alpha h_0^{\alpha-k+1} \beta_k^{\beta_k}. \tag{A.3}$$

Since $c_k \in C^{\beta_k}([0, 1])$ by assumption, from standard interpolation theory we know that for every $t \in I$ there exist $\xi(t) \in I$ such that

$$c_k(t) - c_{k,j}(t) = \frac{c_k^{(\beta_k)}(\xi(t))}{\beta_k!} \prod_{r=1}^{\beta_k} (t - t^{(r)});$$

see, e.g., [27, Theorem 3.1.1]. Thus, for all $t \in I$ we have

$$|c_k(t) - c_{k,j}(t)| \le \frac{\|c_k^{(\beta_k)}\|_\infty}{\beta_k!} \prod_{r=1}^{\beta_k} |t - t^{(r)}| \\ \le \frac{\|c_k^{(\beta_k)}\|_\infty}{\beta_k!} \prod_{r=1}^{\beta_k} \beta_k h_0 \le \frac{\beta_k^{\beta_k} \|c_k^{(\beta_k)}\|_\infty}{\beta_k!} h_0^{\alpha-k+1}. \tag{A.4}$$

From (A.3)–(A.4) we obtain

$$\begin{aligned} |c_k(t) - \tilde{c}_{k,j}(t)| &\le |c_k(t) - c_{k,j}(t)| + |c_{k,j}(t) - \tilde{c}_{k,j}(t)| \\ &\le B(k,\alpha) h_0^{\alpha-k+1} \le B_\alpha h_0^{\alpha-k+1} \end{aligned} \tag{A.5}$$

for all $t \in I$, where

$$B(k,\alpha) = \frac{\beta_k^{\beta_k} \|c_k^{(\beta_k)}\|_\infty}{\beta_k!} + A_\alpha \beta_k^{\beta_k}$$

and $B_\alpha = \max_{i=0,\dots,\alpha} B(i,\alpha)$ depends only on α (recall that β_k depends only on k, α). Since $t_{j,n} \in I$, it is clear that (4.5) follows from (A.5). □

Proof of Theorem 4.3 By (3.1) and Theorem 4.2, for every $j = 1, \dots, n$,

$$\begin{aligned} |\lambda_j(X_n) - \tilde{\lambda}_j(X_n)| &= \left| \sum_{k=0}^{\alpha} c_k(t_{j,n}) h^k + E_{j,n,\alpha} - \sum_{k=0}^{\alpha} \tilde{c}_{k,j}(t_{j,n}) h^k \right| \\ &= \left| \sum_{k=0}^{\alpha} (c_k(t_{j,n}) - \tilde{c}_{k,j}(t_{j,n})) h^k + E_{j,n,\alpha} \right| \\ &\le B_\alpha \sum_{k=0}^{\alpha} h_0^{\alpha-k+1} h^k + C_\alpha h^{\alpha+1} \le D_\alpha h_0^{\alpha+1}, \end{aligned}$$

where $D_\alpha = (\alpha+1) B_\alpha + C_\alpha$. □

Funding Open access funding provided by Università degli Studi di Roma Tor Vergata within the CRUI-CARE Agreement.

Declarations

Conflicts of interest The authors declare that they have no conflict of interest.

References

1. Adriani, A., Bianchi, D., Serra-Capizzano, S.: Asymptotic spectra of large (grid) graphs with a uniform local structure (part I): theory. Milan J. Math. **88**, 409–454 (2020)

2. Ahmad, F., Al-Aidarous, E.S., Alrehaili, D.A., Ekström, S.-E., Furci, I., Serra-Capizzano, S.: Are the eigenvalues of preconditioned banded symmetric Toeplitz matrices known in almost closed form? Numer. Algor. **78**, 867–893 (2018)
3. Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., Sorensen, D.: LAPACK Users' Guide, 3rd edn. SIAM, Philadelphia (1999)
4. Arbenz, P.: Computing the eigenvalues of banded symmetric Toeplitz matrices. SIAM J. Sci. Stat. Comput. **12**, 743–754 (1991)
5. Auricchio, F., Beirão da Veiga, L., Hughes, T.J.R., Reali, A., Sangalli, G.: Isogeometric collocation methods. Math. Models Methods Appl. Sci. **20**, 2075–2107 (2010)
6. Badía, J.M., Vidal, A.M.: Parallel computation of the eigenstructure of Toeplitz-plus-Hankel matrices on multicomputers. In: Dongarra, J., Waśniewski, J. (eds.) Parallel Scientific Computing, pp. 33–40. Springer, Berlin (1994)
7. Badía, J.M., Vidal, A.M.: Parallel algorithms to compute the eigenvalues and eigenvectors of symmetric Toeplitz matrices. Parallel Algor. Appl. **13**, 75–93 (1998)
8. Barbarino, G., Bianchi, D., Garoni, C.: Constructive approach to the monotone rearrangement of functions. Expo. Math. **40**, 155–175 (2022)
9. Barbarino, G., Claesson, M., Ekström, S.-E., Garoni, C., Meadon, D., Speleers, H.: Matrix-less eigensolver for large structured matrices. Technical Report 2021-007, Department of Information Technology, Uppsala University (2021)
10. Barbarino, G., Garoni, C.: An extension of the theory of GLT sequences: sampling on asymptotically uniform grids. Linear Multilinear Algebra **71**, 2008–2025 (2023)
11. Barbarino, G., Garoni, C., Serra-Capizzano, S.: Block generalized locally Toeplitz sequences: theory and applications in the unidimensional case. Electron. Trans. Numer. Anal. **53**, 28–112 (2020)
12. Barbarino, G., Garoni, C., Serra-Capizzano, S.: Block generalized locally Toeplitz sequences: theory and applications in the multidimensional case. Electron. Trans. Numer. Anal. **53**, 113–216 (2020)
13. Barrera, M., Böttcher, A., Grudsky, S.M., Maximenko, E.A.: Eigenvalues of even very nice Toeplitz matrices can be unexpectedly erratic. Oper. Theory Adv. Appl. **268**, 51–77 (2018)
14. Batalshchikov, A.A., Grudsky, S.M., Malisheva, I.S., Mihalkovich, S.S., Ramírez de Arellano, E., Stukopin, V.A.: Asymptotics of eigenvalues of large symmetric Toeplitz matrices with smooth simple-loop symbols. Linear Algebra Appl. **580**, 292–335 (2019)
15. Benzi, M., Golub, G.H., Liesen, J.: Numerical solution of saddle point problems. Acta Numer. **14**, 1–137 (2005)
16. Bianchi, D.: Analysis of the spectral symbol associated to discretization schemes of linear self-adjoint differential operators. Calcolo **58**, 38 (2021)
17. Bianchi, D., Serra-Capizzano, S.: Spectral analysis of finite-dimensional approximations of $1d$ waves in non-uniform grids. Calcolo **55**, 47 (2018)
18. Bini, D., Di Benedetto, F.: Solving the generalized eigenvalue problem for rational Toeplitz matrices. SIAM J. Matrix Anal. Appl. **11**, 537–552 (1990)
19. Bini, D., Pan, V.: Efficient algorithms for the evaluation of the eigenvalues of (block) banded Toeplitz matrices. Math. Comput. **50**, 431–448 (1988)
20. Boffi, D.: Finite element approximation of eigenvalue problems. Acta Numer. **19**, 1–120 (2010)
21. Bogoya, J.M., Böttcher, A., Grudsky, S.M., Maximenko, E.A.: Eigenvalues of Hermitian Toeplitz matrices with smooth simple-loop symbols. J. Math. Anal. Appl. **422**, 1308–1334 (2015)
22. Bogoya, J.M., Ekström, S.-E., Serra-Capizzano, S., Vassalos, P.: Matrix-less methods for the spectral approximation of large non-Hermitian Toeplitz matrices: a concise theoretical analysis and a numerical study. Numer. Linear Algebra Appl. **31**, e2545 (2024)
23. Bogoya, J.M., Grudsky, S.M., Maximenko, E.A.: Eigenvalues of Hermitian Toeplitz matrices generated by simple-loop symbols with relaxed smoothness. Oper. Theory Adv. Appl. **259**, 179–212 (2017)
24. Böttcher, A., Garoni, C., Serra-Capizzano, S.: Exploration of Toeplitz-like matrices with unbounded symbols is not a purely academic journey. Sb. Math. **208**, 1602–1627 (2017)
25. Böttcher, A., Grudsky, S.M., Maximenko, E.A.: Inside the eigenvalues of certain Hermitian Toeplitz band matrices. J. Comput. Appl. Math. **233**, 2245–2264 (2010)
26. Brezinski, C., Redivo Zaglia, M.: Extrapolation Methods: Theory and Practice. North-Holland, Elsevier Science Publishers B.V., Amsterdam (1991)
27. Davis, P.J.: Interpolation and Approximation. Dover Publications, New York (1975)

28. Di Benedetto, F.: Computing eigenvalues and singular values of Toeplitz matrices. Calcolo **33**, 237–248 (1996)
29. Di Benedetto, F.: Generalized updating problems and computation of the eigenvalues of rational Toeplitz matrices. Linear Algebra Appl. **267**, 187–219 (1997)
30. Dongarra, J.J., Moler, C.B., Wilkinson, J.H.: Improving the accuracy of computed eigenvalues and eigenvectors. SIAM J. Numer. Anal. **20**, 23–45 (1983)
31. Ekström, S.-E., Furci, I., Garoni, C., Manni, C., Serra-Capizzano, S., Speleers, H.: Are the eigenvalues of the B-spline isogeometric analysis approximation of $-\Delta u = \lambda u$ known in almost closed form? Numer. Linear Algebra Appl. **25**, e2198 (2018)
32. Ekström, S.-E., Furci, I., Serra-Capizzano, S.: Exact formulae and matrix-less eigensolvers for block banded symmetric Toeplitz matrices. BIT Numer. Math. **58**, 937–968 (2018)
33. Ekström, S.-E., Garoni, C.: A matrix-less and parallel interpolation-extrapolation algorithm for computing the eigenvalues of preconditioned banded symmetric Toeplitz matrices. Numer. Algor. **80**, 819–848 (2019)
34. Ekström, S.-E., Garoni, C., Serra-Capizzano, S.: Are the eigenvalues of banded symmetric Toeplitz matrices known in almost closed form? Exper. Math. **27**, 478–487 (2018)
35. Ekström, S.-E., Vassalos, P.: A matrix-less method to approximate the spectrum and the spectral function of Toeplitz matrices with real eigenvalues. Numer. Algor. **89**, 701–720 (2022)
36. Garoni, C., Serra-Capizzano, S.: Generalized Locally Toeplitz Sequences: Theory and Applications, vol. I. Springer, Cham (2017)
37. Garoni, C., Serra-Capizzano, S.: Generalized Locally Toeplitz Sequences: Theory and Applications, vol. II. Springer, Cham (2018)
38. Garoni, C., Speleers, H., Ekström, S.-E., Reali, A., Serra-Capizzano, S., Hughes, T.J.R.: Symbol-based analysis of finite element and isogeometric B-spline discretizations of eigenvalue problems: exposition and review. Arch. Comput. Methods Engrg. **26**, 1639–1690 (2019)
39. Handy, S.L., Barlow, J.L.: Numerical solution of the eigenproblem for banded symmetric Toeplitz matrices. SIAM J. Matrix Anal. Appl. **15**, 205–214 (1994)
40. Ng, M.K., Trench, W.F.: Numerical solution of the eigenvalue problem for Hermitian Toeplitz-like matrices. Technical Report TR-CS-97-14, Department of Computer Science and Computer Sciences Laboratory, The Australian National University (1997)
41. Saad, Y.: Numerical Methods for Large Eigenvalue Problems, Revised edn. SIAM, Philadelphia (2011)
42. Stoer, J., Bulirsch, R.: Introduction to Numerical Analysis, 3rd edn. Springer, New York (2002)
43. Trench, W.F.: On the eigenvalue problem for Toeplitz band matrices. Linear Algebra Appl. **64**, 199–214 (1985)
44. Trench, W.F.: Numerical solution of the eigenvalue problem for symmetric rationally generated Toeplitz matrices. SIAM J. Matrix Anal. Appl. **9**, 291–303 (1988)
45. Trench, W.F.: Numerical solution of the eigenvalue problem for Hermitian Toeplitz matrices. SIAM J. Matrix Anal. Appl. **10**, 135–146 (1989)
46. Trench, W.F.: Numerical solution of the eigenvalue problem for efficiently structured Hermitian matrices. Linear Algebra Appl. **154–156**, 415–432 (1991)
47. Trench, W.F.: A note on computing the eigenvalues of banded Hermitian Toeplitz matrices. SIAM J. Sci. Comput. **14**, 248–252 (1993)

Authors and Affiliations

Giovanni Barbarino[1] · Melker Claesson[2] · Sven-Erik Ekström[2] · Carlo Garoni[3] · David Meadon[2] · Hendrik Speleers[3]

✉ Carlo Garoni
garoni@mat.uniroma2.it

Giovanni Barbarino
giovanni.barbarino@umons.ac.be

Melker Claesson
melkerclaesson@gmail.com

Sven-Erik Ekström
sven-erik.ekstrom@uu.se

David Meadon
david.meadon@uu.se

Hendrik Speleers
speleers@mat.uniroma2.it

[1] Mathematics and Operational Research Unit, University of Mons, Mons, Belgium

[2] Department of Information Technology, Uppsala University, Uppsala, Sweden

[3] Department of Mathematics, University of Rome Tor Vergata, Rome, Italy