Numer Algor (2019) 80:819–848
https://doi.org/10.1007/s11075-018-0508-0

ORIGINAL PAPER

A matrix-less and parallel interpolation–extrapolation algorithm for computing the eigenvalues of preconditioned banded symmetric Toeplitz matrices

Sven-Erik Ekström[1] · Carlo Garoni[2,3]

Received: 22 August 2017 / Accepted: 4 March 2018 / Published online: 24 March 2018

Abstract In the past few years, Bogoya, Böttcher, Grudsky, and Maximenko obtained the precise asymptotic expansion for the eigenvalues of a Toeplitz matrix $T_n(f)$, under suitable assumptions on the generating function f, as the matrix size n goes to infinity. On the basis of several numerical experiments, it was conjectured by Serra-Capizzano that a completely analogous expansion also holds for the eigenvalues of the preconditioned Toeplitz matrix $T_n(u)^{-1}T_n(v)$, provided $f = v/u$ is monotone and further conditions on u and v are satisfied. Based on this expansion, we here propose and analyze an interpolation–extrapolation algorithm for computing the eigenvalues of $T_n(u)^{-1}T_n(v)$. The algorithm is suited for parallel implementation and it may be called "matrix-less" as it does not need to store the entries of the matrix. We illustrate the performance of the algorithm through numerical experiments and we also present its generalization to the case where $f = v/u$ is non-monotone.

Keywords Preconditioned Toeplitz matrices · Eigenvalues · Asymptotic eigenvalue expansion · Polynomial interpolation · Extrapolation

Mathematics Subject Classification (2010) 15B05 · 65F15 · 65D05 · 65B05

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

Carlo Garoni
carlo.garoni@usi.ch; carlo.garoni@uninsubria.it

[1] Department of Information Technology, Division of Scientific Computing, Uppsala University, ITC, Lägerhyddsv. 2, Hus 2, P.O. Box 337, SE-751 05 Uppsala, Sweden

[2] Institute of Computational Science, University of Italian Switzerland (USI), Via Giuseppe Buffi 13, 6900 Lugano, Switzerland

[3] Department of Science and High Technology, University of Insubria, Via Valleggio 11, 22100 Como, Italy

1 Introduction

A matrix of the form

$$\left[a_{i-j}\right]_{i,j=1}^{n}=\begin{bmatrix} a_0 & a_{-1} & \cdots & \cdots & a_{-(n-1)} \\ a_1 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & a_{-1} \\ a_{n-1} & \cdots & \cdots & a_1 & a_0 \end{bmatrix},$$

whose entries are constant along each diagonal, is called a Toeplitz matrix. Given a function $g:[-\pi,\pi]\to\mathbb{C}$ belonging to $L^1([-\pi,\pi])$, the nth Toeplitz matrix associated with g is defined as

$$T_n(g)=\left[\hat{g}_{i-j}\right]_{i,j=1}^{n},$$

where the numbers $\hat{g}_k$ are the Fourier coefficients of g,

$$\hat{g}_k=\frac{1}{2\pi}\int_{-\pi}^{\pi}g(\theta)\mathrm{e}^{-\mathrm{i}k\theta}\mathrm{d}\theta, \qquad k\in\mathbb{Z}.$$

We refer to $\{T_n(g)\}_n$ as the Toeplitz sequence generated by g, which in turn is called the generating function of $\{T_n(g)\}_n$. It is not difficult to see that, whenever g is real, $T_n(g)$ is Hermitian for all n. Moreover, if g is real non-negative and not almost everywhere equal to zero in $[-\pi,\pi]$, then $T_n(g)$ is Hermitian positive definite for all n; see [9, 14]. In the case where g is a real cosine trigonometric polynomial (RCTP), that is, a function of the form

$$g(\theta)=\hat{g}_0+2\sum_{k=1}^{m}\hat{g}_k\cos(k\theta), \qquad \hat{g}_0,\hat{g}_1,\ldots,\hat{g}_m\in\mathbb{R}, \qquad m\in\mathbb{N},$$

the nth Toeplitz matrix generated by g is the real symmetric banded matrix given by

$$T_n(g)=\begin{bmatrix}
\hat{g}_0 & \hat{g}_1 & \cdots & \hat{g}_m & & & & & & \\
\hat{g}_1 & \ddots & \ddots & & \ddots & & & & & \\
\vdots & \ddots & \ddots & \ddots & & \ddots & & & & \\
\hat{g}_m & & \ddots & \ddots & \ddots & & \ddots & & & \\
 & \ddots & & \ddots & \ddots & \ddots & & \ddots & & \\
 & & \boxed{\hat{g}_m \ \cdots \ \hat{g}_1 \ \hat{g}_0 \ \hat{g}_1 \ \cdots \ \hat{g}_m} & & & & & & & \\
 & & & \ddots & & \ddots & \ddots & \ddots & & \ddots \\
 & & & & \ddots & & \ddots & \ddots & \ddots & \hat{g}_m \\
 & & & & & \ddots & & \ddots & \ddots & \vdots \\
 & & & & & & \ddots & & \ddots & \hat{g}_1 \\
 & & & & & & \hat{g}_m & \cdots & \hat{g}_1 & \hat{g}_0
\end{bmatrix}.$$

The numerical approximation of the eigenvalues of real symmetric banded Toeplitz matrices is a problem that has been faced by several authors; see, e.g., Arbenz [2], Badía and Vidal [3], Bini and Pan [5], the authors and Serra-Capizzano [13], and Trench [16–20]. Less attention has been devoted to the numerical approximation of the eigenvalues of preconditioned banded symmetric Toeplitz matrices of the form $T_n(u)^{-1}T_n(v)$, with u, v being RCTPs. Yet, this problem is worthy of consideration as noted in [4, Section 1]. Some algorithms to solve it have been proposed in [1, 4]. For general discussions on the various algorithmic proposals for solving eigenvalue problems related to banded Toeplitz matrices, we refer the reader [2, Section 1] and [4, Section 1].

In this paper, we propose a new algorithm for the numerical approximation of the eigenvalues of preconditioned banded symmetric Toeplitz matrices. The algorithm relies on the following conjecture, which has been formulated by Serra-Capizzano in [1], on the basis of several numerical experiments.

Conjecture 1 Let u, v be RCTPs, with $u > 0$ on $(0, \pi)$, and suppose that $f = v/u$ is monotone increasing over $(0, \pi)$. Set $X_n = T_n(u)^{-1}T_n(v)$ for all n. Then, for every integer $\alpha \geq 0$, every n and every $j = 1, \ldots, n$, the following asymptotic expansion holds:

$$\lambda_j(X_n) = f(\theta_{j,n}) + \sum_{k=1}^{\alpha} c_k(\theta_{j,n})h^k + E_{j,n,\alpha}, \tag{1}$$

where:

- The eigenvalues of X_n are arranged in non-decreasing order, $\lambda_1(X_n) \leq \ldots \leq \lambda_n(X_n)$.[1]
- $\{c_k\}_{k=1,2,\ldots}$ is a sequence of functions from $(0, \pi)$ to $\mathbb{R}$ which depends only on u, v.
- $h = \frac{1}{n+1}$ and $\theta_{j,n} = \frac{j\pi}{n+1} = j\pi h$.
- $E_{j,n,\alpha} = O(h^{\alpha+1})$ is the remainder (the error), which satisfies the inequality $|E_{j,n,\alpha}| \leq C_\alpha h^{\alpha+1}$ for some constant C_α depending only on α, u, v.

In the case where $u = 1$ identically, Conjecture 1 was originally formulated and supported through numerical experiments in [13]. In the case where $u = 1$ identically and v satisfies some additional assumptions, Conjecture 1 was formally proved by Bogoya, Böttcher, Grudsky, and Maximenko in a sequence of recent papers [6, 8, 10].

Assuming Conjecture 1, in Section 2 of this paper, we describe and analyze a new algorithm for computing the eigenvalues of $X_n = T_n(u)^{-1}T_n(v)$; and in Section 3, we illustrate its performance through numerical experiments. The algorithm, which is suited for *parallel implementation* and may be called *matrix-less* as it does not need to store the entries of X_n, combines the extrapolation procedure proposed in [1, 13]—which allows the computation of *some* of the eigenvalues of X_n—with an appropriate interpolation process, thus allowing the simultaneous computation of *all* the eigenvalues of X_n. In Section 4, we provide a generalization of the

[1]Note that the eigenvalues of X_n are real, because X_n is similar to the symmetric matrix $T_n(u)^{-1/2}T_n(v)T_n(u)^{-1/2}$.

algorithm to the case where $f = v/u$ is non-monotone; this generalization is based on another conjecture which is analogous to Conjecture 1 and which will be discussed later on. In Section 5, we draw conclusions and suggest possible future lines of research.

2 The algorithm

Throughout this paper, we associate with each positive integer $n \in \mathbb{N} = \{1, 2, 3, \ldots\}$ the stepsize $h = \frac{1}{n+1}$ and the grid points $\theta_{j,n} = j\pi h$, $j = 1, \ldots, n$. For notational convenience, we will always denote a positive integer and the associated stepsize in a similar 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; etc. Throughout this section, we make the following assumptions:

- u, v, f are as in Conjecture 1.
- $n, n_1, \alpha \in \mathbb{N}$ are fixed parameters and $X_n = T_n(u)^{-1}T_n(v)$.
- $n_k = 2^{k-1}(n_1 + 1) - 1$ for $k = 2, \ldots, \alpha$.
- $j_k = 2^{k-1}j_1$ for $j_1 = 1, \ldots, n_1$ and $k = 2, \ldots, \alpha$. Note that $j_k = j_k(j_1)$ depends not only on k but also on j_1, though we hide the dependence on j_1 for notational simplicity. Note also that j_k is the index in $\{1, \ldots, n_k\}$ such that $\theta_{j_k,n_k} = \theta_{j_1,n_1}$. Hence, the grid $\{\theta_{j_k,n_k} : j_1 = 1, \ldots, n_1\}$ is the same as the grid $\{\theta_{j_1,n_1} : j_1 = 1, \ldots, n_1\}$ for all $k = 2, \ldots, \alpha$.

A graphical representation of the grids $\{\theta_{1,n_k}, \ldots, \theta_{n_k,n_k}\}$, $k = 1, \ldots, \alpha$, is reported in Fig. 1 for $n_1 = 5$ and $\alpha = 4$. For each "level" $k = 2, \ldots, \alpha$, the corresponding red circles highlight the subgrid $\{\theta_{j_k,n_k} : j_1 = 1, \ldots, n_1\}$ which coincides with the coarsest grid $\{\theta_{j_1,n_1} : j_1 = 1, \ldots, n_1\}$.

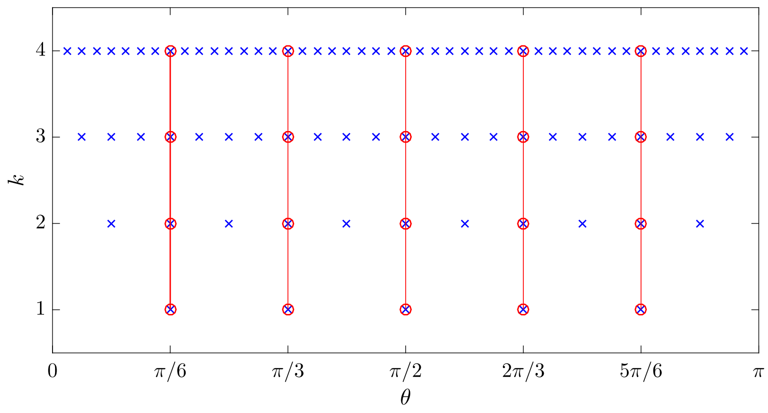

Fig. 1 Representation of the grids $\{\theta_{1,n_k}, \ldots, \theta_{n_k,n_k}\}$, $k = 1, \ldots, \alpha$, for $n_1 = 5$ and $\alpha = 4$

2.1 Description and formulation of the algorithm

The algorithm we are going to describe is designed for computing the eigenvalues of X_n in the case where n is large with respect to $n_1, \ldots, n_\alpha$, so that the computation of the eigenvalues of X_n is hard from a computational viewpoint but the computation of the eigenvalues of $X_{n_1}, \ldots, X_{n_\alpha}$—which is required in the algorithm—can be efficiently performed by any standard eigensolver (e.g., MATLAB's `eig` function); see also Remark 1 below. The algorithm is composed of two phases: a first phase where we invoke extrapolation procedures from [1, 13] and a second phase where local interpolation techniques are employed.

Extrapolation For each fixed $j_1 = 1, \ldots, n_1$, we apply α times the expansion (1) with $n = n_1, n_2, \ldots, n_\alpha$ and $j = j_1, j_2, \ldots, j_\alpha$. Since $\theta_{j_1,n_1} = \theta_{j_2,n_2} = \ldots = \theta_{j_\alpha,n_\alpha}$ (by definition of $j_2, \ldots, j_\alpha$), we obtain

$$\begin{cases} E_{j_1,n_1,0} &= c_1(\theta_{j_1,n_1})h_1 + c_2(\theta_{j_1,n_1})h_1^2 + \ldots + c_\alpha(\theta_{j_1,n_1})h_1^\alpha + E_{j_1,n_1,\alpha} \\ E_{j_2,n_2,0} &= c_1(\theta_{j_1,n_1})h_2 + c_2(\theta_{j_1,n_1})h_2^2 + \ldots + c_\alpha(\theta_{j_1,n_1})h_2^\alpha + E_{j_2,n_2,\alpha} \\ \vdots \\ E_{j_\alpha,n_\alpha,0} &= c_1(\theta_{j_1,n_1})h_\alpha + c_2(\theta_{j_1,n_1})h_\alpha^2 + \ldots + c_\alpha(\theta_{j_1,n_1})h_\alpha^\alpha + E_{j_\alpha,n_\alpha,\alpha} \end{cases} \tag{2}$$

where

$$E_{j_k,n_k,0} = \lambda_{j_k}(X_{n_k}) - f(\theta_{j_1,n_1}), \qquad k = 1, \ldots, \alpha,$$

and

$$|E_{j_k,n_k,\alpha}| \leq C_\alpha h_k^{\alpha+1}, \qquad k = 1, \ldots, \alpha. \tag{3}$$

Let $\tilde{c}_1(\theta_{j_1,n_1}), \ldots, \tilde{c}_\alpha(\theta_{j_1,n_1})$ be the approximations of $c_1(\theta_{j_1,n_1}), \ldots, c_\alpha(\theta_{j_1,n_1})$ obtained by removing all the errors $E_{j_1,n_1,\alpha}, \ldots, E_{j_\alpha,n_\alpha,\alpha}$ in (2) and by solving the resulting linear system:

$$\begin{cases} E_{j_1,n_1,0} &= \tilde{c}_1(\theta_{j_1,n_1})h_1 + \tilde{c}_2(\theta_{j_1,n_1})h_1^2 + \ldots + \tilde{c}_\alpha(\theta_{j_1,n_1})h_1^\alpha \\ E_{j_2,n_2,0} &= \tilde{c}_1(\theta_{j_1,n_1})h_2 + \tilde{c}_2(\theta_{j_1,n_1})h_2^2 + \ldots + \tilde{c}_\alpha(\theta_{j_1,n_1})h_2^\alpha \\ \vdots \\ E_{j_\alpha,n_\alpha,0} &= \tilde{c}_1(\theta_{j_1,n_1})h_\alpha + \tilde{c}_2(\theta_{j_1,n_1})h_\alpha^2 + \ldots + \tilde{c}_\alpha(\theta_{j_1,n_1})h_\alpha^\alpha \end{cases} \tag{4}$$

Note that this way of computing approximations for $c_1(\theta_{j_1,n_1}), \ldots, c_\alpha(\theta_{j_1,n_1})$ was already proposed in [1, 13], and it 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 [15, Section 3.4]. In this regard, the asymptotic expansion (1) plays here the same role as the Euler–Maclaurin summation formula [15, Section 3.3]. For more advanced studies on extrapolation methods, we refer the reader to [11]. The next theorem shows that the approximation error $|c_k(\theta_{j_1,n_1}) - \tilde{c}_k(\theta_{j_1,n_1})|$ is $O(h_1^{\alpha-k+1})$.

Theorem 1 There exists a constant A_α depending only on α, u, v such that, for $j_1 = 1, \dots, n_1$ and $k = 1, \dots, \alpha$,

$$|c_k(\theta_{j_1,n_1}) - \tilde{c}_k(\theta_{j_1,n_1})| \leq A_\alpha h_1^{\alpha-k+1}. \tag{5}$$

Proof See Appendix A. □

Interpolation Fix an index $j \in \{1, \dots, n\}$. To compute an approximation of $\lambda_j(X_n)$ through the expansion (1), we would need the value $c_k(\theta_{j,n})$ for each $k = 1, \dots, \alpha$. Of course, $c_k(\theta_{j,n})$ is not available in practice, but we can approximate it by interpolating in some way the values $\tilde{c}_k(\theta_{j_1,n_1})$, $j_1 = 1, \dots, n_1$. For example, we may define $\tilde{c}_k(\theta)$ as the interpolation polynomial of the data $(\theta_{1,n_1}, \tilde{c}_k(\theta_{1,n_1})), \dots, (\theta_{n_1,n_1}, \tilde{c}_k(\theta_{n_1,n_1}))$—so that $\tilde{c}_k(\theta)$ is expected to be an approximation of $c_k(\theta)$ over the whole interval $(0, \pi)$—and take $\tilde{c}_k(\theta_{j,n})$ as an approximation to $c_k(\theta_{j,n})$. It is known, however, that interpolation over a large number of uniform nodes is not advisable as it may give rise to spurious oscillations (Runge's phenomenon [12, p. 78]). It is therefore better to adopt another kind of approximation. An alternative could be the following: we approximate $c_k(\theta)$ by the spline function $\tilde{c}_k(\theta)$ which is linear on each interval $[\theta_{j_1,n_1}, \theta_{j_1+1,n_1}]$ and takes the value $\tilde{c}_k(\theta_{j_1,n_1})$ at θ_{j_1,n_1} for all $j_1 = 1, \dots, n_1$. This strategy removes for sure any spurious oscillation, yet it is not accurate. In particular, it does not preserve the accuracy of approximation at the nodes θ_{j_1,n_1} established in Theorem 1, i.e., there is no guarantee that $|c_k(\theta) - \tilde{c}_k(\theta)| \leq B_\alpha h_1^{\alpha-k+1}$ for $\theta \in (0, \pi)$ or $|c_k(\theta_{j,n}) - \tilde{c}_k(\theta_{j,n})| \leq B_\alpha h_1^{\alpha-k+1}$ for $j = 1, \dots, n$, with B_α being a constant depending only on α, u, v. As proved in Theorem 2, a local approximation strategy that preserves the accuracy (5), at least if $c_k(\theta)$ is sufficiently smooth, is the following: let $\theta^{(1)}, \dots, \theta^{(\alpha-k+1)}$ be $\alpha - k + 1$ points of the grid $\{\theta_{1,n_1}, \dots, \theta_{n_1,n_1}\}$ which are closest to the point $\theta_{j,n}$,[2] and let $\tilde{c}_{k,j}(\theta)$ be the interpolation polynomial of the data $(\theta^{(1)}, \tilde{c}_k(\theta^{(1)})), \dots, (\theta^{(\alpha-k+1)}, \tilde{c}_k(\theta^{(\alpha-k+1)}))$; then, we approximate $c_k(\theta_{j,n})$ by $\tilde{c}_{k,j}(\theta_{j,n})$. Note that, by selecting $\alpha - k + 1$ points from $\{\theta_{1,n_1}, \dots, \theta_{n_1,n_1}\}$, we are implicitly assuming that $n_1 \geq \alpha - k + 1$.

Theorem 2 Let $1 \leq k \leq \alpha$, and suppose $n_1 \geq \alpha - k + 1$ and $c_k \in C^{\alpha-k+1}([0, \pi])$. For $j = 1, \dots, n$, if $\theta^{(1)}, \dots, \theta^{(\alpha-k+1)}$ are $\alpha - k + 1$ points of $\{\theta_{1,n_1}, \dots, \theta_{n_1,n_1}\}$ which are closest to $\theta_{j,n}$, and if $\tilde{c}_{k,j}(\theta)$ is the interpolation polynomial of the data $(\theta^{(1)}, \tilde{c}_k(\theta^{(1)})), \dots, (\theta^{(\alpha-k+1)}, \tilde{c}_k(\theta^{(\alpha-k+1)}))$, then

$$|c_k(\theta_{j,n}) - \tilde{c}_{k,j}(\theta_{j,n})| \leq B_\alpha h_1^{\alpha-k+1} \tag{6}$$

for some constant B_α depending only on α, u, v.

Proof See Appendix A. □

[2]These $\alpha - k + 1$ points are uniquely determined by $\theta_{j,n}$ except in the case where $\theta_{j,n}$ coincides with either a grid point θ_{j_1,n_1} or the midpoint between two consecutive grid points θ_{j_1,n_1} and θ_{j_1+1,n_1}.

Formulation of the algorithm We are now ready to formulate our algorithm for computing the eigenvalues of X_n. As we shall see in Remark 4, the algorithm is suited for *parallel implementation*. Since it does not even need to store the entries of X_n, it may be called *matrix-less*. It can be used for computing either a specific eigenvalue $\lambda_j(X_n)$, a subset of the eigenvalues of X_n, or the whole spectrum of X_n. A plain (non-parallel) MATLAB implementation of this algorithm is reported in Appendix B.

Algorithm 1 Given two RCTPs u, v (with $u > 0$ on $(0, \pi)$ and $f = v/u$ monotone increasing over $(0, \pi)$ as in Conjecture 1), three integers $n, n_1, \alpha \in \mathbb{N}$ with $n_1 \geq \alpha$, and $S \subseteq \{1, \dots, n\}$, we compute an approximation of the eigenvalues $\{\lambda_j(X_n) : j \in S\}$ as follows:

```
1. For $j_1 = 1, \dots, n_1$ compute $\tilde{c}_1(\theta_{j_1,n_1}), \dots, \tilde{c}_\alpha(\theta_{j_1,n_1})$ by solving (4)
2. For $j \in S$
     • For $k = 1, \dots, \alpha$
         – Determine $\alpha - k + 1$ points $\theta^{(1)}, \dots, \theta^{(\alpha-k+1)} \in \{\theta_{1,n_1}, \dots, \theta_{n_1,n_1}\}$
           which are closest to $\theta_{j,n}$
         – Compute $\tilde{c}_{k,j}(\theta_{j,n})$, where $\tilde{c}_{k,j}(\theta)$ is the interpolation polynomial of
           $(\theta^{(1)}, \tilde{c}_k(\theta^{(1)})), \dots, (\theta^{(\alpha-k+1)}, \tilde{c}_k(\theta^{(\alpha-k+1)}))$
     • Compute $\tilde{\lambda}_j(X_n) = f(\theta_{j,n}) + \sum_{k=1}^{\alpha} \tilde{c}_{k,j}(\theta_{j,n}) h^k$
3. Return $\{\tilde{\lambda}_j(X_n) : j \in S\}$ as an approximation to $\{\lambda_j(X_n) : j \in S\}$
```

Remark 1 Algorithm 1 is specifically designed for computing the eigenvalues of X_n in the case where the matrix size n is quite large. When applying this algorithm, it is implicitly assumed that n_1 and α are small (much smaller than n), so that each $n_k = 2^{k-1}(n_1 + 1) - 1$ 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 (e.g., MATLAB's `eig` function).

Remark 2 A careful evaluation shows that the computational cost of Algorithm 1 is bounded by

$$C(\alpha^2 n_1 + \alpha^3 |S|) + \sum_{k=1}^{\alpha} C_{\text{eig}}(n_k),$$

where $|S|$ is the cardinality of S, C is a constant depending only on f, and $C_{\text{eig}}(n_k)$ is the cost for computing the eigenvalues of X_{n_k}.

Remark 3 Algorithm 1 can be optimized in several ways. For example, if $S = \{j\}$, so that only the jth eigenvalue $\lambda_j(X_n)$ must be computed, then in the first step one can just compute the values $\tilde{c}_1(\theta_{j_1,n_1}), \dots, \tilde{c}_\alpha(\theta_{j_1,n_1})$ for $\theta_{j_1,n_1} \in \{\theta^{(1)}, \dots, \theta^{(\alpha)}\}$, where $\theta^{(1)}, \dots, \theta^{(\alpha)}$ are α points in $\{\theta_{1,n_1}, \dots, \theta_{n_1,n_1}\}$ which are closest to $\theta_{j,n}$. Indeed, only these values are needed in the second step. A similar consideration applies in the case where only the extremal eigenvalues of X_n must be computed, and also in the case where S is a small subset of $\{1, \dots, n\}$ of the form $\{j, \dots, j + r\}$, with $r \ll n$.

Remark 4 Suppose $|S| = n$ and consider the ideal situation where we have n processors. Then, the jth processor can compute the jth eigenvalue $\lambda_j(X_n)$ independently of the others. In view of Remark 3, the jth processor can act as follows:

- In the first step of the algorithm, it computes only the values $\tilde{c}_1(\theta_{j_1,n_1}), \dots, \tilde{c}_\alpha(\theta_{j_1,n_1})$ for $\theta_{j_1,n_1} \in \{\theta^{(1)}, \dots, \theta^{(\alpha)}\}$, where $\theta^{(1)}, \dots, \theta^{(\alpha)}$ are α points in $\{\theta_{1,n_1}, \dots, \theta_{n_1,n_1}\}$ which are closest to $\theta_{j,n}$.
- It performs the second step of the algorithm for the index j only.

It is clear that such a parallel implementation is very fast as the computation of all the eigenvalues of X_n takes the same time as the computation of one eigenvalue only. A similar consideration also applies in the case where $|S| < n$ and we have $|S|$ processors, each of which has to compute only one of the requested $|S|$ eigenvalues. In a more realistic situation, we will not have a number of processors equal to $|S|$ if $|S|$ is large. Instead, we will have p processors with $p \ll |S|$. In this case, we can divide S into p different subsets $S_1, \dots, S_p$ of approximately the same cardinality and assign to the ith processor the computation of the eigenvalues corresponding to S_i, $i = 1, \dots, p$. When doing so, it is advisable that each S_i is constructed so that the "positions" $\theta_{j,n}$ of the related eigenvalues $\lambda_j(X_n)$ are close to each other, because in this way each processor will have the possibility to perform a reduced form of the first step of the algorithm, in analogy with what has been explained above for the case $p = |S|$. For example, if $|S| = n$ and n is a multiple of p, then we can assign to the ith processor the computation of the eigenvalues $\lambda_j(X_n)$ for $j = (i-1)(n/p)+1, \dots, i(n/p)$, so that in the first step of the algorithm the ith processor will only have to compute $\tilde{c}_1(\theta_{j_1,n_1}), \dots, \tilde{c}_\alpha(\theta_{j_1,n_1})$ for θ_{j_1,n_1} in a neighborhood of the interval $[\theta_{(i-1)(n/p)+1,n}, \theta_{i(n/p),n}]$.

2.2 Error estimate

Theorem 3 Assume that Conjecture 1 holds. Suppose $n \geq n_1 \geq \alpha$ and $c_k \in C^{\alpha-k+1}([0,\pi])$ for $k = 1, \dots, \alpha$. Let $(\tilde{\lambda}_1(X_n), \dots, \tilde{\lambda}_n(X_n))$ be the approximation of $(\lambda_1(X_n), \dots, \lambda_n(X_n))$ computed by Algorithm 1. Then, there exists a constant D_α depending only on α, u, v such that, for $j = 1, \dots, n$,

$$|\lambda_j(X_n) - \tilde{\lambda}_j(X_n)| \leq D_\alpha h_1^\alpha h.$$

Proof By (1) and Theorem 2,

$$\begin{aligned}
|\lambda_j(X_n) - \tilde{\lambda}_j(X_n)| &= \left| f(\theta_{j,n}) + \sum_{k=1}^{\alpha} c_k(\theta_{j,n}) h^k + E_{j,n,\alpha} - f(\theta_{j,n}) - \sum_{k=1}^{\alpha} \tilde{c}_{k,j}(\theta_{j,n}) h^k \right| \\
&= \left| \sum_{k=1}^{\alpha} (c_k(\theta_{j,n}) - \tilde{c}_{k,j}(\theta_{j,n})) h^k + E_{j,n,\alpha} \right| \\
&\leq B_\alpha \sum_{k=1}^{\alpha} h_1^{\alpha-k+1} h^k + C_\alpha h^{\alpha+1} \leq D_\alpha h_1^\alpha h,
\end{aligned}$$

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

Remark 5 The error estimate provided in Theorem 3 suggests that the eigenvalue approximations provided by Algorithm 1 improve as n increases, i.e., as h decreases. Numerical experiments reveal that this is in fact the case (see Example 2 below).

Remark 6 Theorem 3 shows that, for any fixed $\alpha \geq 1$, the numerical eigenvalues computed by Algorithm 1 converge like h_1^α to the exact eigenvalues as n_1 grows. In practice, it is advisable to fix α and increase n_1 until a proper stopping criterion is reached. The other way (fix n_1 and increase α) is not advisable as the constant D_α in Theorem 3 apparently grows very quickly with α (see Example 1 below) and, consequently, there is no guarantee on the convergence of the algorithm as α grows (see Example 5 below).

3 Numerical experiments

In this section, we illustrate through numerical examples the performance of Algorithm 1. Numerical experiments have been performed with MATLAB R2015b (64 bit) on a platform with 4GB RAM, using an Intel® Celeron® Processor N2820 (up to 2.39 GHz, 1 MB L2 cache). The CPU times for Algorithm 1 refer to the plain MATLAB implementation reported in Appendix B. In what follows, the symbol $\varepsilon_{j,n}$ denotes the error $|\lambda_j(X_n) - \tilde{\lambda}_j(X_n)|$, which occurs when approximating the exact eigenvalue $\lambda_j(X_n)$ with the corresponding numerical eigenvalue $\tilde{\lambda}_j(X_n)$ computed by Algorithm 1. The inputs u, v, n, n_1, α with which Algorithm 1 is applied are specified in each example.

Example 1 Let

$$\begin{aligned} u(\theta) &= 1, \\ v(\theta) &= 6 - 8\cos(\theta) + 2\cos(2\theta). \end{aligned}$$

Note that $f(\theta) = v(\theta)/u(\theta) = v(\theta)$ is monotone increasing on $(0, \pi)$. Suppose we want to approximate the eigenvalues of $X_n = T_n(u)^{-1}T_n(v) = T_n(f)$ for $n = 5000$. Let $\tilde{\lambda}_j(X_n)$ be the approximation of $\lambda_j(X_n)$ obtained by applying Algorithm 1 with $n_1 = 10$ and $\alpha = 7$. In Fig. 2, we plot the errors $\varepsilon_{j,n}$ versus $\theta_{j,n}$ for $j = 1, \ldots, n$. We note that the largest errors are attained when either $\theta_{j,n} \approx 0$ or $\theta_{j,n} \approx \pi$. As highlighted also in Example 3 below, this is probably due to two concomitant factors:

- The errors $\varepsilon_{j,n}$ are supposed to be smaller for $\theta_{j,n} \in [\theta_{1,n_1}, \theta_{n_1,n_1}] = [\pi/11, 10\pi/11]$, because in this case the approximations $\tilde{c}_{k,j}(\theta_{j,n})$ computed by Algorithm 1 for the values $c_k(\theta_{j,n})$ are expected to be more accurate as the interpolation polynomial $\tilde{c}_{k,j}(\theta)$ is evaluated inside the convex hull of the interpolation nodes.
- $\theta = 0$ and $\theta = \pi$ are the two points on $[0, \pi]$ where f' vanishes, which means that the monotonicity of f is "weak" around these points (recall that Algorithm 1 works under the assumption that f is monotone as in Conjecture 1).

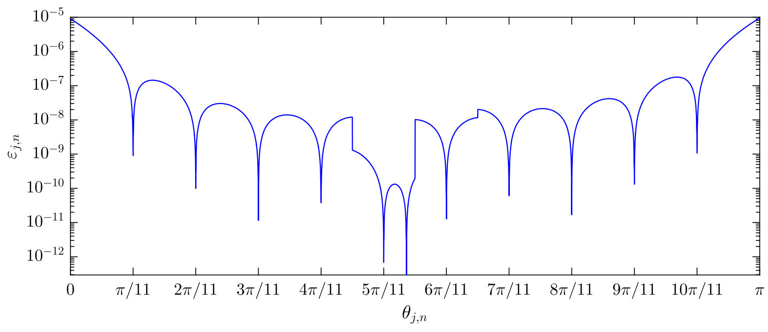

Fig. 2 Example 1: errors $\varepsilon_{j,n}$ versus $\theta_{j,n}$ for $j = 1, \ldots, n$ in the case where $u(\theta) = 1$, $v(\theta) = 6 - 8\cos(\theta) + 2\cos(2\theta)$, $n = 5000$, $n_1 = 10$, and $\alpha = 7$

In reference to the previous discussion, we note that the maximum error for $\theta_{j,n} \in [\theta_{1,n_1}, \theta_{n_1,n_1}]$ is given by

$$\max\{\varepsilon_{j,n} : \theta_{j,n} \in [\theta_{1,n_1}, \theta_{n_1,n_1}]\} \approx 1.7803 \cdot 10^{-7},$$

which is about two order of magnitude less than

$$\max_{j=1,\ldots,n} \varepsilon_{j,n} \approx 9.5167 \cdot 10^{-6}.$$

A careful look at Fig. 2 shows that, aside from the exceptional minimum attained inside the interval $(5\pi/11, 6\pi/11)$, the local minima of $\varepsilon_{j,n}$ are attained when $\theta_{j,n}$ is approximately equal to some of the grid points θ_{j_1,n_1}, $j_1 = 1, \ldots, n_1$. This is no surprise, because for $\theta_{j,n} = \theta_{j_1,n_1}$ we have $\tilde{c}_{k,j}(\theta_{j,n}) = \tilde{c}_k(\theta_{j_1,n_1})$ and $c_k(\theta_{j,n}) = c_k(\theta_{j_1,n_1})$, which means that the error of the approximation $\tilde{c}_{k,j}(\theta_{j,n}) \approx c_k(\theta_{j,n})$ reduces to the error of the approximation $\tilde{c}_k(\theta_{j_1,n_1}) \approx c_k(\theta_{j_1,n_1})$; that is, we are not introducing further error due to the interpolation process. To conclude, we make the following observation: for α, u, v as in this example, Theorem 3 yields

$$D_\alpha \geq \frac{\max_{j=1,\ldots,n} \varepsilon_{j,n}}{h_1^\alpha h} \approx 9.2745 \cdot 10^5 > \alpha^\alpha = 8.23543 \cdot 10^5.$$

This suggests that, unfortunately, the best constant D_α for which the error estimate of Theorem 3 is satisfied grows very quickly with α.

Example 2 Let u, v, f be as in Example 1. Suppose we want to approximate the eigenvalues of $X_n = T_n(u)^{-1}T_n(v) = T_n(f)$ for $n = 10000$. Let $\tilde{\lambda}_j(X_n)$ be the approximation of $\lambda_j(X_n)$ obtained by applying Algorithm 1 with $n_1 = 10$ and $\alpha = 7$ as in Example 1. In Fig. 3, we plot the errors $\varepsilon_{j,n}$ versus $\theta_{j,n}$ for $j = 1, \ldots, n$. We note that the errors in Fig. 3 are smaller than in Fig. 2. This shows that the eigenvalue approximations provided by Algorithm 1 improve as n increases (see also Remark 5).

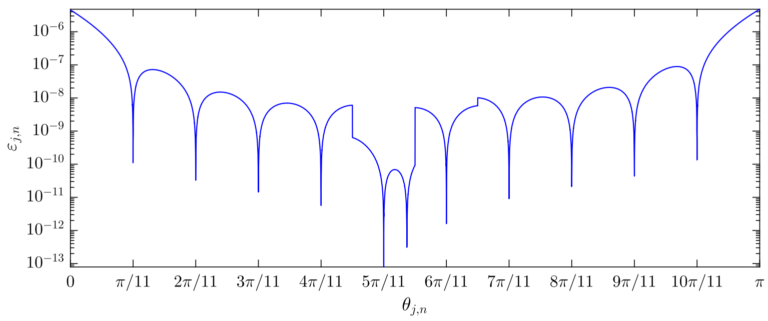

Fig. 3 Example 2: errors $\varepsilon_{j,n}$ versus $\theta_{j,n}$ for $j = 1, \ldots, n$ in the case where $u(\theta) = 1$, $v(\theta) = 6 - 8\cos(\theta) + 2\cos(2\theta)$, $n = 10000$, $n_1 = 10$, and $\alpha = 7$

Example 3 Let

$$\begin{aligned} u(\theta) &= 1, \\ v(\theta) &= -\frac{1}{4} - \frac{1}{2}\cos(\theta) + \frac{1}{4}\cos(2\theta) - \frac{1}{12}\cos(3\theta). \end{aligned}$$

Note that $f(\theta) = v(\theta)/u(\theta) = v(\theta)$ is monotone increasing on $(0, \pi)$. Suppose we want to approximate the eigenvalues of $X_n = T_n(u)^{-1}T_n(v) = T_n(f)$ for $n = 10000$. Let $\tilde{\lambda}_j^{(m)}(X_n)$ be the approximation of $\lambda_j(X_n)$ obtained by applying Algorithm 1 with $n_1 = 10 \cdot 2^{m-1}$ and $\alpha = 5$. In Fig. 4, we plot the errors $\varepsilon_{j,n}^{(m)} = |\lambda_j(X_n) - \tilde{\lambda}_j^{(m)}(X_n)|$ versus $\theta_{j,n}$ for $j = 1, \ldots, n$ and $m = 1, 2, 3, 4$. We see from the figure that, as m increases, the error decreases rather quickly everywhere except in a neighborhood of the point $\theta = \pi/3$ where f' vanishes. Actually, the three points of $[0, \pi]$ where f' vanishes are 0, $\pi/3$, π, and these are precisely the points around which the error is higher than elsewhere. We remark that, as in Examples 1 and 2, the error $\varepsilon_{j,n}^{(m)}$ attains its local minima when $\theta_{j,n}$ is approximately equal to some of the nodes $\theta_{1,n_1}, \ldots, \theta_{n_1,n_1}$.

Example 4 Let

$$\begin{aligned} u(\theta) &= 1, \\ v(\theta) &= \frac{301}{400} - \cos(\theta) + \frac{1}{5}\cos(2\theta) + \frac{1}{10}\cos(3\theta) - \frac{1}{20}\cos(4\theta) + \frac{1}{400}\cos(6\theta). \end{aligned}$$

Note that $f(\theta) = v(\theta)/u(\theta) = v(\theta)$ is monotone increasing on $(0, \pi)$ and $f'(\theta) = 0$ only for $\theta = 0, \pi$.[3] Suppose we want to approximate the eigenvalues of $X_n = T_n(u)^{-1}T_n(v) = T_n(f)$ for $n = 10000$. Let $\tilde{\lambda}_j^{(m)}(X_n)$ be the approximation of $\lambda_j(X_n)$ obtained by applying Algorithm 1 with $n_1 = 25 \cdot 2^{m-1}$ and $\alpha = 5$. In Fig. 5, we plot the errors $\varepsilon_{j,n}^{(m)}$ versus $\theta_{j,n}$ for $j = 1, \ldots, n$ and $m = 1, 2, 3, 4$. Considerations analogous to those of Example 3 apply also in this case.

[3]Note that we always have $g'(0) = g'(\pi) = 0$ whenever $g(\theta)$ is an RCTP.

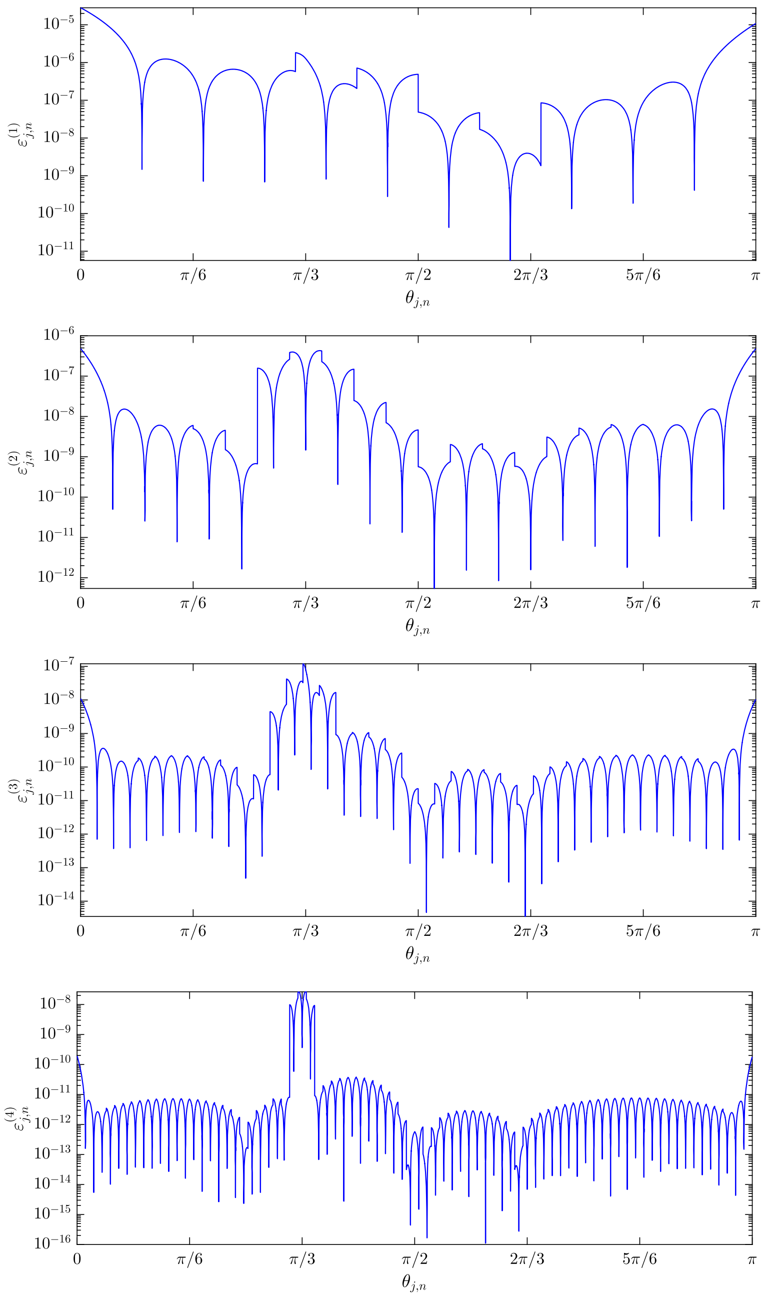

Fig. 4 Example 3: errors $\varepsilon^{(m)}_{j,n}$ versus $\theta_{j,n}$ for $j = 1, \ldots, n$, in the case where $u(\theta) = 1$, $v(\theta) = -\frac{1}{4} - \frac{1}{2}\cos(\theta) + \frac{1}{4}\cos(2\theta) - \frac{1}{12}\cos(3\theta)$, $n = 10000$, $n_1 = 10 \cdot 2^{m-1}$, and $\alpha = 5$

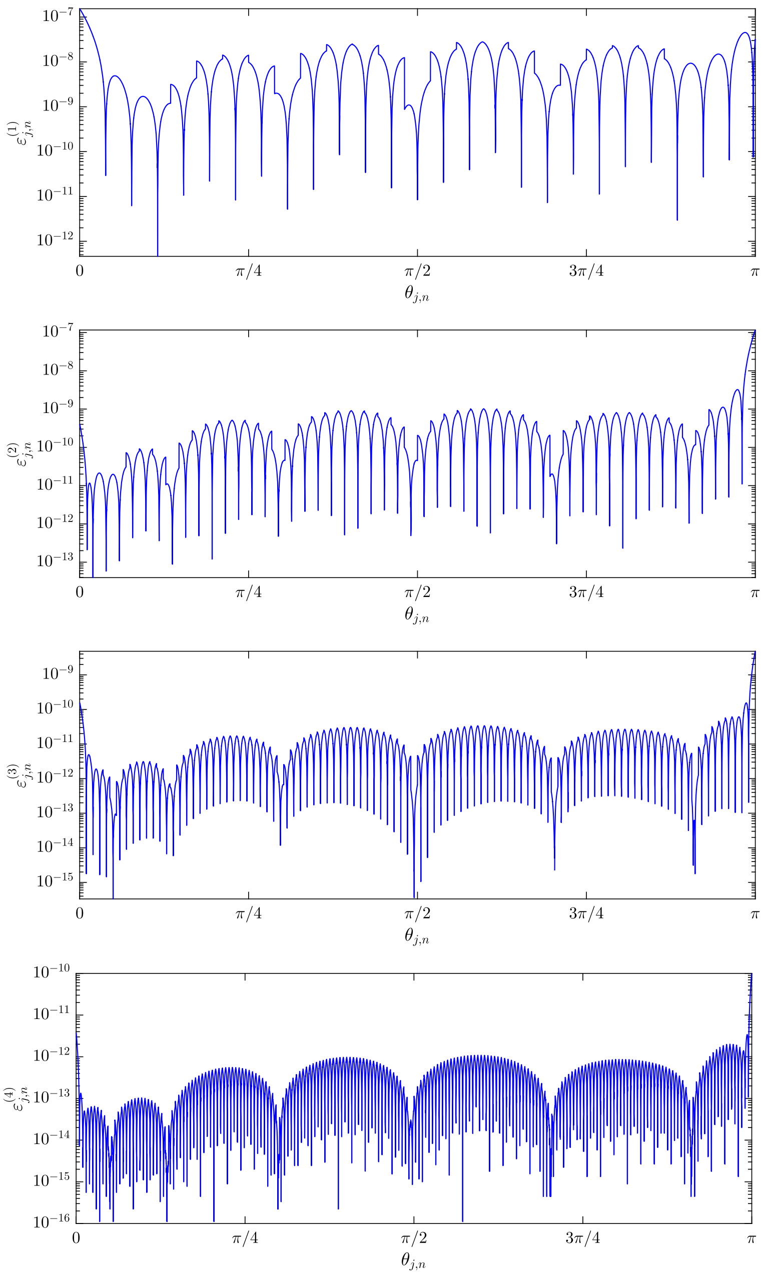

Fig. 5 Example 4: errors $\varepsilon^{(m)}_{j,n}$ versus $\theta_{j,n}$ for $j = 1, \dots, n$, in the case where $u(\theta) = 1$, $v(\theta) = \frac{301}{400} - \cos(\theta) + \frac{1}{5}\cos(2\theta) + \frac{1}{10}\cos(3\theta) - \frac{1}{20}\cos(4\theta) + \frac{1}{400}\cos(6\theta)$, $n = 10000$, $n_1 = 25 \cdot 2^{m-1}$, and $\alpha = 5$

Example 5 Let u, v, f as in Example 4. Suppose we want to approximate the eigenvalues of $X_n = T_n(u)^{-1}T_n(v) = T_n(f)$ for $n = 10000$. Let $\tilde{\lambda}_j^{(m)}(X_n)$ be the approximation of $\lambda_j(X_n)$ obtained by applying Algorithm 1 with $n_1 = 25$ and $\alpha = 4 + m$. In Fig. 6, we plot the errors $\varepsilon_{j,n}^{(m)}$ versus $\theta_{j,n}$ for $j = 1, \dots, n$ and $m = 1, 2, 3, 4$. By comparing Fig. 5 with Fig. 6, we see that the strategy of keeping n_1 fixed and increasing α is much less efficient than the strategy of keeping α fixed and increasing n_1. Indeed, while in Fig. 5 the error $\varepsilon_{j,n}^{(m)}$ decreases approximately in a uniform way by one order of magnitude as m increases, this is not observed in Fig. 6. Note also that the computational cost of Algorithm 1 for $n_1 = 25 \cdot 2^{m-1}$ and $\alpha = 5$ (as in Fig. 5) is essentially the same as the cost of Algorithm 1 for $n_1 = 25$ and $\alpha = 4 + m$ (as in Fig. 6), because the main task of the algorithm in both cases is the computation of the eigenvalues of X_{n_α}, and in both cases n_α is approximately equal to $25 \cdot 2^{m+3}$. The bad behavior of Algorithm 1 when increasing α finds an explanation in the fact that, as observed in Example 1, the constant D_α appearing in the error estimate of Theorem 3 apparently grows very quickly with α.

Example 6 Let

$$\begin{aligned} u(\theta) &= 3 + 2\cos(\theta), \\ v(\theta) &= 2 - \cos(\theta) - \cos(2\theta). \end{aligned}$$

Note that $f(\theta) = v(\theta)/u(\theta) = 1 - \cos(\theta)$ is monotone increasing on $(0, \pi)$ and $f'(\theta) = 0$ only for $\theta = 0, \pi$. Suppose we want to approximate the eigenvalues of $X_n = T_n(u)^{-1}T_n(v)$ for $n = 5000$. Let $\tilde{\lambda}_j^{(m)}(X_n)$ be the approximation of $\lambda_j(X_n)$ obtained by applying Algorithm 1 with $n_1 = 50 \cdot 2^{m-1}$ and $\alpha = 4$. The graph of the errors $\varepsilon_{j,n}^{(m)}$ versus $\theta_{j,n}$ is shown in Fig. 7 for $j = 1, \dots, n$ and $m = 1, 2, 3, 4$. Table 1 compares the CPU times for computing the eigenvalues of X_n by using MATLAB's `eig` function and Algorithm 1.

Example 7 This example is suggested by the cubic B-spline isogeometric analysis discretization of second-order eigenvalue problems [14, Section 10.7.3]. Let

$$\begin{aligned} u(\theta) &= 1208 + 1191\cos(\theta) + 120\cos(2\theta) + \cos(3\theta), \\ v(\theta) &= 40 - 15\cos(\theta) - 24\cos(2\theta) - \cos(3\theta). \end{aligned}$$

It can be shown that $u(\theta) > 0$ on $(0, \pi)$,

$$f(\theta) = \frac{v(\theta)}{u(\theta)} = \frac{40 - 15\cos(\theta) - 24\cos(2\theta) - \cos(3\theta)}{1208 + 1191\cos(\theta) + 120\cos(2\theta) + \cos(3\theta)}$$

is monotone increasing on $(0, \pi)$, and $f'(\theta) = 0$ only for $\theta = 0, \pi$. Suppose we want to approximate the eigenvalues of $X_n = T_n(u)^{-1}T_n(v)$ for $n = 5000$. Let $\tilde{\lambda}_j^{(m)}(X_n)$ be the approximation of $\lambda_j(X_n)$ obtained by applying Algorithm 1 with $n_1 = 50 \cdot 2^{m-1}$ and $\alpha = 4$. The graph of the errors $\varepsilon_{j,n}^{(m)}$ versus $\theta_{j,n}$ is shown in Fig. 8 for $j = 1, \dots, n$ and $m = 1, 2, 3, 4$. The CPU times are reported in Table 2.

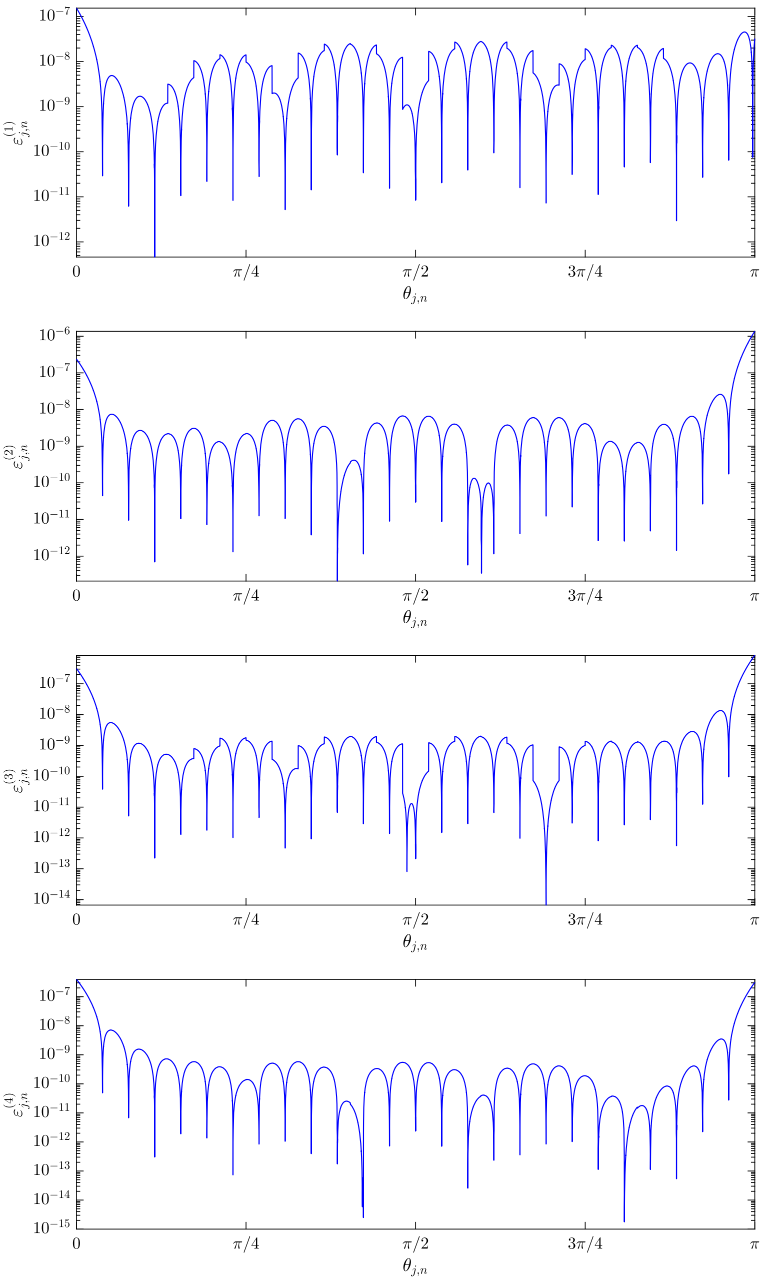

Fig. 6 Example 5: errors $\varepsilon^{(m)}_{j,n}$ versus $\theta_{j,n}$ for $j = 1, \ldots, n$, in the case where $u(\theta) = 1$, $v(\theta) = \frac{301}{400} - \cos(\theta) + \frac{1}{5}\cos(2\theta) + \frac{1}{10}\cos(3\theta) - \frac{1}{20}\cos(4\theta) + \frac{1}{400}\cos(6\theta)$, $n = 10000$, $n_1 = 25$, and $\alpha = 4 + m$

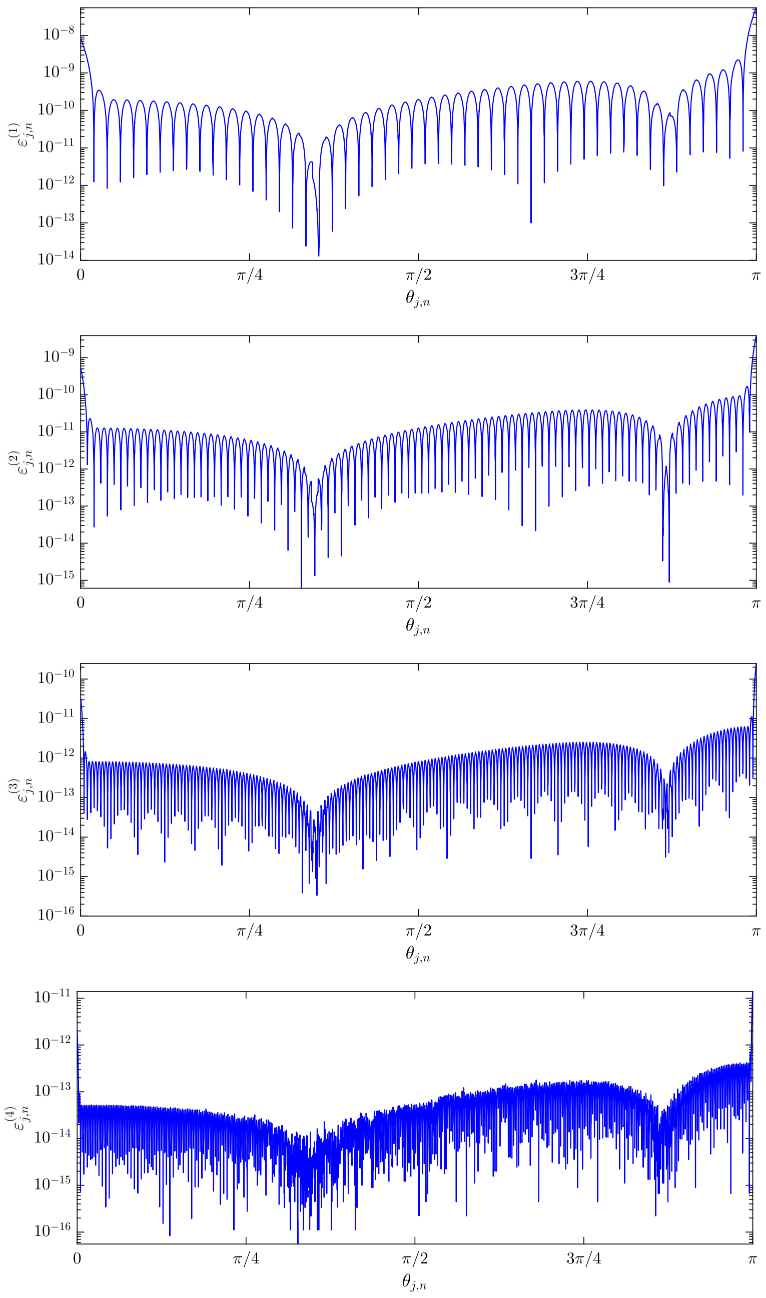

Fig. 7 Example 6: errors $\varepsilon^{(m)}_{j,n}$ versus $\theta_{j,n}$ for $j = 1, \ldots, n$, in the case where $u(\theta) = 3 + 2\cos(\theta)$, $v(\theta) = 2 - \cos(\theta) - \cos(2\theta)$, $n = 5000$, $n_1 = 50 \cdot 2^{m-1}$, and $\alpha = 4$

Table 1 Example 6 (Fig. 7): CPU times for computing the eigenvalues of X_n in the case where $u(\theta) = 3 + 2\cos(\theta)$, $v(\theta) = 2 - \cos(\theta) - \cos(2\theta)$, and $n = 5000$

Method	CPU time
Algorithm 1 with $n_1 = 50$ and $\alpha = 4$	1.81 s
Algorithm 1 with $n_1 = 100$ and $\alpha = 4$	7.14 s
Algorithm 1 with $n_1 = 200$ and $\alpha = 4$	32.45 s
Algorithm 1 with $n_1 = 400$ and $\alpha = 4$	144.08 s
MATLAB's `eig` function	694.76 s

Example 8 Let

$$u(\theta) = 8 - 3\cos(\theta) - 4\cos(2\theta) - \cos(3\theta),$$
$$v(\theta) = \frac{35}{2} - 12\cos(\theta) - 6\cos(2\theta) + \frac{1}{2}\cos(4\theta).$$

It can be shown that $u(\theta) > 0$ on $(0, \pi)$,

$$f(\theta) = \frac{v(\theta)}{u(\theta)} = 2 - \cos(\theta)$$

is monotone increasing on $(0, \pi)$, and $f'(\theta) = 0$ only for $\theta = 0, \pi$. Suppose we want to approximate the smallest five eigenvalues of $X_n = T_n(u)^{-1}T_n(v)$ for $n = 5000$. Let $\tilde{\lambda}_j(X_n)$ be the approximations of $\lambda_j(X_n)$ obtained by applying Algorithm 1 with $n_1 = 100$ and $\alpha = 4$. Table 3 shows the errors $\varepsilon_{j,n}$ for $j = 1, \ldots, 5$, whereas Table 4 compares the CPU times for computing the eigenvalues of X_n by using Algorithm 1, MATLAB's `eig` function, and MATLAB's `eigs` function (applied to the generalized eigenvalue problem $T_n(v)\mathbf{x} = \lambda T_n(u)\mathbf{x}$ with $T_n(v)$ and $T_n(u)$ allocated as sparse matrices through MATLAB's `sparse` command).

4 Generalization to the non-monotone case

With reference to Conjecture 1, suppose that the function $f = v/u$ is monotone decreasing on $(0, \pi)$. Then, $-f = -v/u$ is monotone increasing on $(0, \pi)$ and, moreover, $T_n(u)^{-1}T_n(v) = -T_n(u)^{-1}T_n(-v)$. This immediately implies that Algorithm 1 allows one to compute the eigenvalues of $T_n(u)^{-1}T_n(v)$ even in the case where $f = v/u$ is monotone decreasing on $(0, \pi)$: it suffices to apply the algorithm with $X_n = T_n(u)^{-1}T_n(-v)$. Some limitations on the applicability of Algorithm 1 arise when f is non-monotone on $(0, \pi)$. This is precisely the case we are going to investigate in this section. We begin by formulating the following conjecture.

Conjecture 2 Let u, v be RCTPs, with $u > 0$ on $(0, \pi)$, and suppose that $f = v/u$ restricted to the interval $I \subseteq (0, \pi)$ is monotone and $f^{-1}(f(I)) = I$. Set $X_n = T_n(u)^{-1}T_n(v)$ for all n. Then, for every integer $\alpha \geq 0$, every n and every $j = 1, \ldots, n$ such that $\theta_{j,n} \in I$, the following asymptotic expansion holds:

$$\lambda_{\rho_n(j)}(X_n) = f(\theta_{j,n}) + \sum_{k=1}^{\alpha} c_k(\theta_{j,n})h^k + E_{j,n,\alpha}, \tag{7}$$

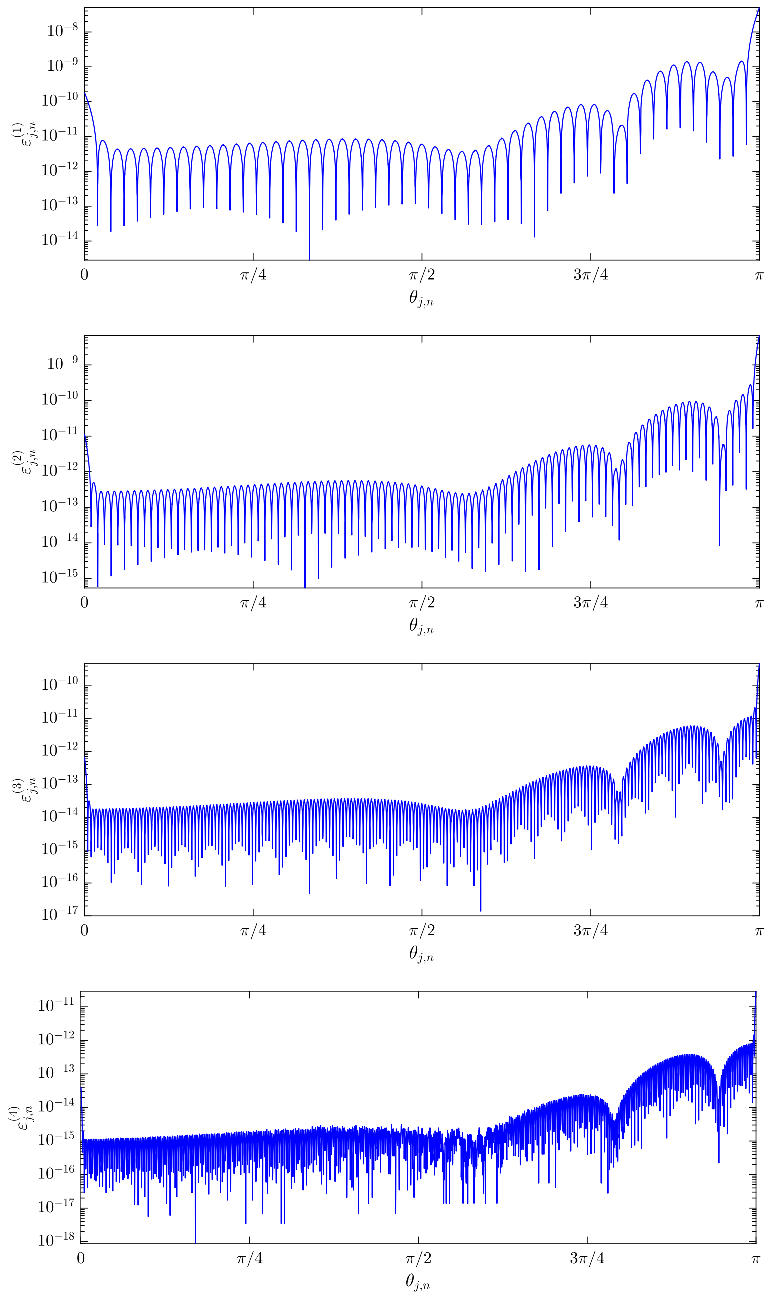

Fig. 8 Example 7: errors $\varepsilon^{(m)}_{j,n}$ versus $\theta_{j,n}$ for $j = 1, \dots, n$, in the case where $u(\theta) = 1208+1191\cos(\theta)+120\cos(2\theta) + \cos(3\theta)$, $v(\theta) = 40 - 15\cos(\theta) - 24\cos(2\theta) - \cos(3\theta)$, $n = 5000$, $n_1 = 50 \cdot 2^{m-1}$, and $\alpha = 4$

Table 2 Example 7 (Fig. 8): CPU times for computing the eigenvalues of X_n in the case where $u(\theta) = 1208 + 1191\cos(\theta) + 120\cos(2\theta) + \cos(3\theta)$, $v(\theta) = 40 - 15\cos(\theta) - 24\cos(2\theta) - \cos(3\theta)$, and $n = 5000$

Method	CPU time
Algorithm 1 with $n_1 = 50$ and $\alpha = 4$	1.69 s
Algorithm 1 with $n_1 = 100$ and $\alpha = 4$	2.77 s
Algorithm 1 with $n_1 = 200$ and $\alpha = 4$	18.30 s
Algorithm 1 with $n_1 = 400$ and $\alpha = 4$	280.27 s
MATLAB's `eig` function	1265.55 s

where:

- The eigenvalues of X_n are arranged in non-decreasing order, $\lambda_1(X_n) \le \ldots \le \lambda_n(X_n)$.
- $\rho_n = \sigma_n^{-1}$ is the inverse of σ_n, where σ_n is a permutation of $\{1, \ldots, n\}$ such that $f(\theta_{\sigma_n(1),n}) \le \ldots \le f(\theta_{\sigma_n(n),n})$.
- $\{c_k\}_{k=1,2,\ldots}$ is a sequence of functions from I to $\mathbb{R}$ which depends only on u, v.
- $h = \frac{1}{n+1}$ and $\theta_{j,n} = \frac{j\pi}{n+1} = j\pi h$.
- $E_{j,n,\alpha} = O(h^{\alpha+1})$ is the error, which satisfies the inequality $|E_{j,n,\alpha}| \le C_\alpha h^{\alpha+1}$ for some constant C_α depending only on α, u, v.

Conjecture 2 is clearly an extension of Conjecture 1. Indeed, in the case where f is monotone increasing on $(0, \pi)$, if we take $I = (0, \pi)$ and we note that both σ_n and ρ_n reduce to the identity on $\{1, \ldots, n\}$, we see that Conjecture 2 reduces to Conjecture 1. Conjecture 2 is based on the numerical experiments carried out in [1, 13]. In the case where $u = 1$ identically, it was already formulated in [13]. In the case where $u = 1$ identically and $\alpha = 0$, it can be formally proved by adapting the argument used by Bogoya, Böttcher, Grudsky, and Maximenko in the proof of [7, Theorem 1.6].

In the situation described in Conjecture 2, we propose the following natural modification of Algorithm 1 for computing the eigenvalues of X_n corresponding to the the interval I (that is, the eigenvalues $\lambda_{\rho_n(j)}(X_n)$ corresponding to points $\theta_{j,n} \in I$). In what follows, for any integer n_1, we denote by $n_1(I)$ the cardinality of $\{\theta_{1,n_1}, \ldots, \theta_{n_1,n_1}\} \cap I$.

Algorithm 2 With the notation introduced in Conjecture 2, given two RCTPs u, v (with $u > 0$ on $(0, \pi)$ and $f = v/u$ such that f restricted to the interval $I \subseteq (0, \pi)$ is monotone and $f^{-1}(f(I)) = I$), three integers $n, n_1, \alpha \in \mathbb{N}$ with $n_1(I) \ge \alpha$ and

Table 3 Example 8: errors $\varepsilon_{j,n}$ for $j = 1, \ldots, 5$, in the case where $u(\theta) = 8 - 3\cos(\theta) - 4\cos(2\theta) - \cos(3\theta)$, $v(\theta) = \frac{35}{2} - 12\cos(\theta) - 6\cos(2\theta) + \frac{1}{2}\cos(4\theta)$, $n = 5000$, $n_1 = 100$, and $\alpha = 4$

j	1	2	3	4	5
$\varepsilon_{j,n}$	$1.56 \cdot 10^{-6}$	$1.42 \cdot 10^{-6}$	$1.47 \cdot 10^{-6}$	$1.34 \cdot 10^{-6}$	$1.39 \cdot 10^{-6}$

Table 4 Example 8: CPU times for computing the smallest five eigenvalues of X_n in the case where $u(\theta) = 8 - 3\cos(\theta) - 4\cos(2\theta) - \cos(3\theta)$, $v(\theta) = \frac{35}{2} - 12\cos(\theta) - 6\cos(2\theta) + \frac{1}{2}\cos(4\theta)$, and $n = 5000$

Method	CPU time
Algorithm 1 with $n_1 = 100$ and $\alpha = 4$	1.13 s
MATLAB's `eig` function	346.21 s
MATLAB's `eigs` function	Does not converge

$S \subseteq I$, we compute approximations of the eigenvalues $\{\lambda_{\rho_n(j)}(X_n) : \theta_{j,n} \in S\}$ as follows:

1. For $j_1 = 1, \dots, n_1$ such that $\theta_{j_1,n_1} \in I$ compute $\tilde{c}_1(\theta_{j_1,n_1}), \dots, \tilde{c}_\alpha(\theta_{j_1,n_1})$ by solving the linear system

$$\begin{cases} E_{j_1,n_1,0} = \tilde{c}_1(\theta_{j_1,n_1})h_1 + \tilde{c}_2(\theta_{j_1,n_1})h_1^2 + \ldots + \tilde{c}_\alpha(\theta_{j_1,n_1})h_1^\alpha \\ E_{j_2,n_2,0} = \tilde{c}_1(\theta_{j_1,n_1})h_2 + \tilde{c}_2(\theta_{j_1,n_1})h_2^2 + \ldots + \tilde{c}_\alpha(\theta_{j_1,n_1})h_2^\alpha \\ \vdots \\ E_{j_\alpha,n_\alpha,0} = \tilde{c}_1(\theta_{j_1,n_1})h_\alpha + \tilde{c}_2(\theta_{j_1,n_1})h_\alpha^2 + \ldots + \tilde{c}_\alpha(\theta_{j_1,n_1})h_\alpha^\alpha \end{cases} \tag{8}$$

where $n_k = 2^{k-1}(n_1 + 1) - 1$, $j_k = 2^{k-1}j_1$, and

$$E_{j_k,n_k,0} = \lambda_{\rho_{n_k}(j_k)}(X_{n_k}) - f(\theta_{j_1,n_1}), \qquad k = 1, \dots, \alpha.$$

2. For $j = 1, \dots, n$ such that $\theta_{j,n} \in S$
   - For $k = 1, \dots, \alpha$
     - Determine $\alpha - k + 1$ points $\theta^{(1)}, \dots, \theta^{(\alpha-k+1)} \in \{\theta_{1,n_1}, \dots, \theta_{n_1,n_1}\} \cap I$ which are closest to $\theta_{j,n}$
     - Compute $\tilde{c}_{k,j}(\theta_{j,n})$, where $\tilde{c}_{k,j}(\theta)$ is the interpolation polynomial of $(\theta^{(1)}, \tilde{c}_k(\theta^{(1)})), \dots, (\theta^{(\alpha-k+1)}, \tilde{c}_k(\theta^{(\alpha-k+1)}))$
   - Compute $\tilde{\lambda}_{\rho_n(j)}(X_n) = f(\theta_{j,n}) + \sum_{k=1}^{\alpha} \tilde{c}_{k,j}(\theta_{j,n})h^k$
3. Return $\{\tilde{\lambda}_{\rho_n(j)}(X_n) : \theta_{j,n} \in S\}$ as an approximation to $\{\lambda_{\rho_n(j)}(X_n) : \theta_{j,n} \in S\}$

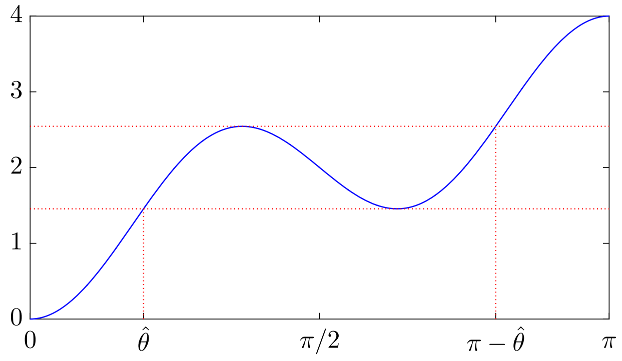

Fig. 9 Example 9: graph of $f(\theta) = v(\theta)/u(\theta) = 2 - \cos(\theta) - \cos(3\theta)$ over $(0, \pi)$

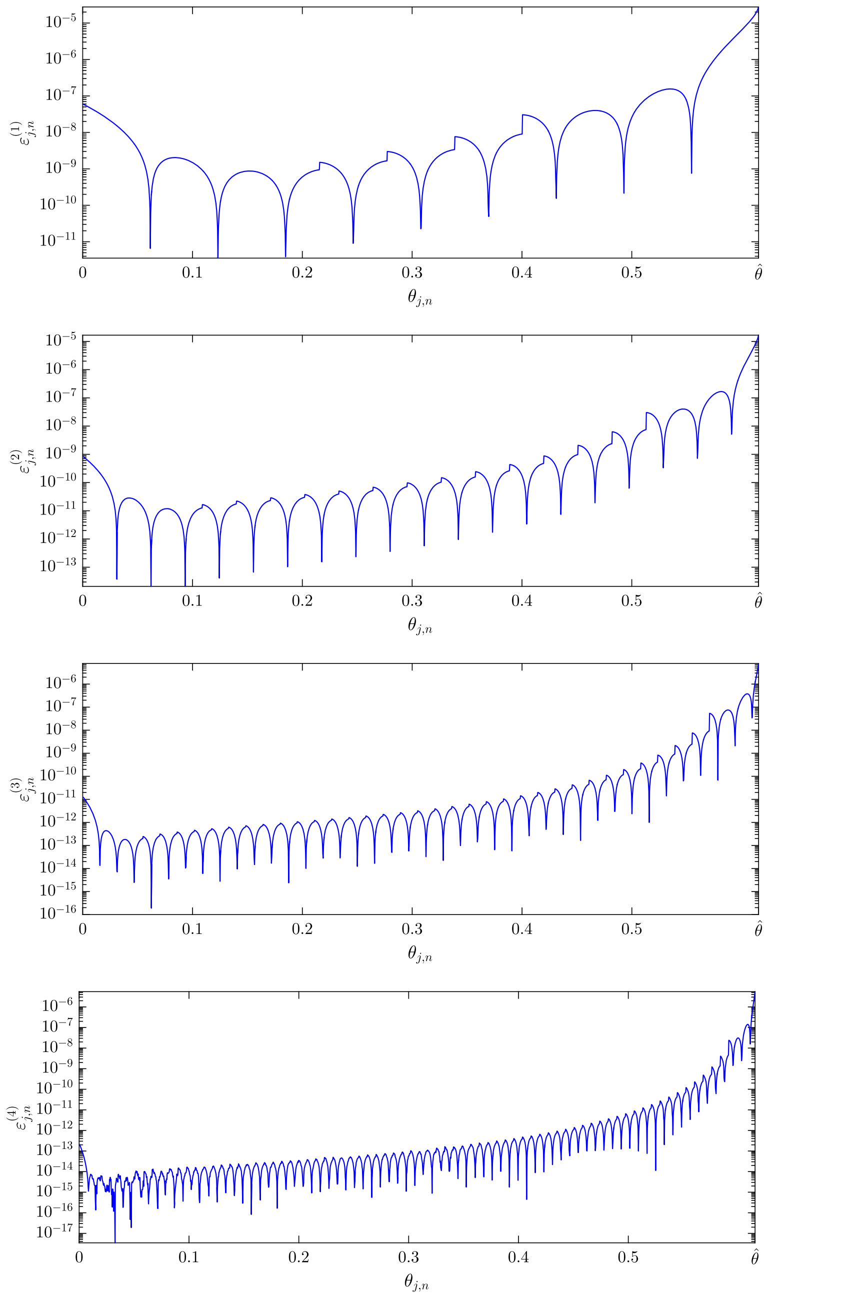

Fig. 10 Example 9: errors $\varepsilon^{(m)}_{j,n}$ versus $\theta_{j,n}$ for $\theta_{j,n} \in I = (0, \hat{\theta})$, in the case where $u(\theta) = 1$, $v(\theta) = 2 - \cos(\theta) - \cos(3\theta)$, $n = 10000$, $n_1 = 50 \cdot 2^{m-1}$, and $\alpha = 5$

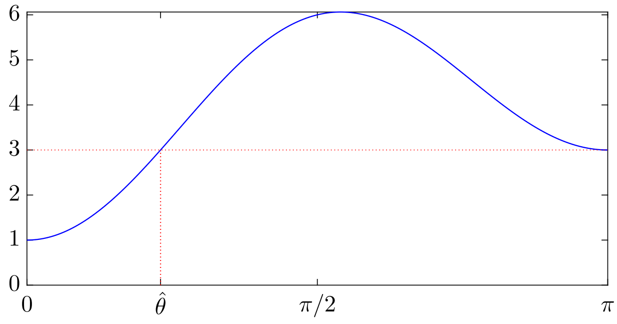

Fig. 11 Example 10: graph of $f(\theta) = v(\theta)/u(\theta) = 4 - \cos(\theta) - 2\cos(2\theta)$ over $(0, \pi)$

Example 9 Let

$$\begin{aligned} u(\theta) &= 1, \\ v(\theta) &= 2 - \cos(\theta) - \cos(3\theta). \end{aligned}$$

The graph of $f(\theta) = v(\theta)/u(\theta) = v(\theta)$ is depicted in Fig. 9. The hypotheses of Conjecture 2 are satisfied with either $I = (0, \hat\theta)$ or $I = (\pi - \hat\theta, \pi)$, where $\hat\theta = 0.61547970867038...$ To fix the ideas, let $I = (0, \hat\theta)$. Note that any permutation σ_n which sorts the samples $f(\theta_{1,n}), \ldots, f(\theta_{n,n})$ in non-decreasing order is such that $\sigma_n(j) = j$ whenever $\theta_{j,n} \in I$. As a consequence, $\rho_n(j) = j$ whenever $\theta_{j,n} \in I$. Set $X_n = T_n(u)^{-1}T_n(v) = T_n(f)$ and let $\{\tilde\lambda_j^{(m)}(X_n) : \theta_{j,n} \in I\}$ be the approximation of $\{\lambda_j(X_n) : \theta_{j,n} \in I\}$ obtained for $n = 10000$ by applying Algorithm 2 with $n_1 = 50 \cdot 2^{m-1}$, $\alpha = 5$, and $S = I$. The graph of the errors $\varepsilon_{j,n}^{(m)} = |\lambda_j(X_n) - \tilde\lambda_j^{(m)}(X_n)|$ versus $\theta_{j,n}$ is shown in Fig. 10 for $\theta_{j,n} \in I$ and $m = 1, 2, 3, 4$. We note that the error $\varepsilon_{j,n}^{(m)}$ tends to increase as $\theta_{j,n}$ moves toward $\hat\theta$, that is, as $\theta_{j,n}$ approaches to exit the interval I over which f satisfies the assumptions of Conjecture 2. Moreover, in a neighborhood of $\hat\theta$, the error decreases very slowly. This phenomenon is related to the fact that the expansion (7) does not hold in $[\hat\theta, \pi - \hat\theta]$ and, in fact, the errors $E_{j,n,0} = \lambda_{\rho_n(j)}(X_n) - f(\theta_{j,n})$ have a wild behavior inside this interval; see [13, Fig. 7].

Example 10 Let

$$\begin{aligned} u(\theta) &= 2 + \cos(3\theta), \\ v(\theta) &= 8 - 3\cos(\theta) - \frac{9}{2}\cos(2\theta) + 4\cos(3\theta) - \frac{1}{2}\cos(4\theta) - \cos(5\theta). \end{aligned}$$

The graph of $f(\theta) = v(\theta)/u(\theta) = 4 - \cos(\theta) - 2\cos(2\theta)$ is depicted in Fig. 11. The hypotheses of Conjecture 2 are satisfied with $I = (0, \hat\theta)$, where $\hat\theta = 0.72273424781341...$ Any permutation σ_n which sorts the samples $f(\theta_{1,n}), \ldots, f(\theta_{n,n})$ in non-decreasing order is such that $\sigma_n(j) = j$ whenever $\theta_{j,n} \in I$. As a consequence, $\rho_n(j) = j$ whenever $\theta_{j,n} \in I$. Set $X_n = T_n(u)^{-1}T_n(v)$ and let $\{\tilde\lambda_j^{(m)}(X_n) : \theta_{j,n} \in I\}$ be the approximation of $\{\lambda_j(X_n) : \theta_{j,n} \in I\}$ obtained for $n = 5000$ by applying Algorithm 2 with $n_1 = 25 \cdot 2^{m-1}$, $\alpha = 5$, and $S = I$. The graph of the errors $\varepsilon_{j,n}^{(m)} = |\lambda_j(X_n) - \tilde\lambda_j^{(m)}(X_n)|$ versus $\theta_{j,n}$ is shown in Fig. 12 for

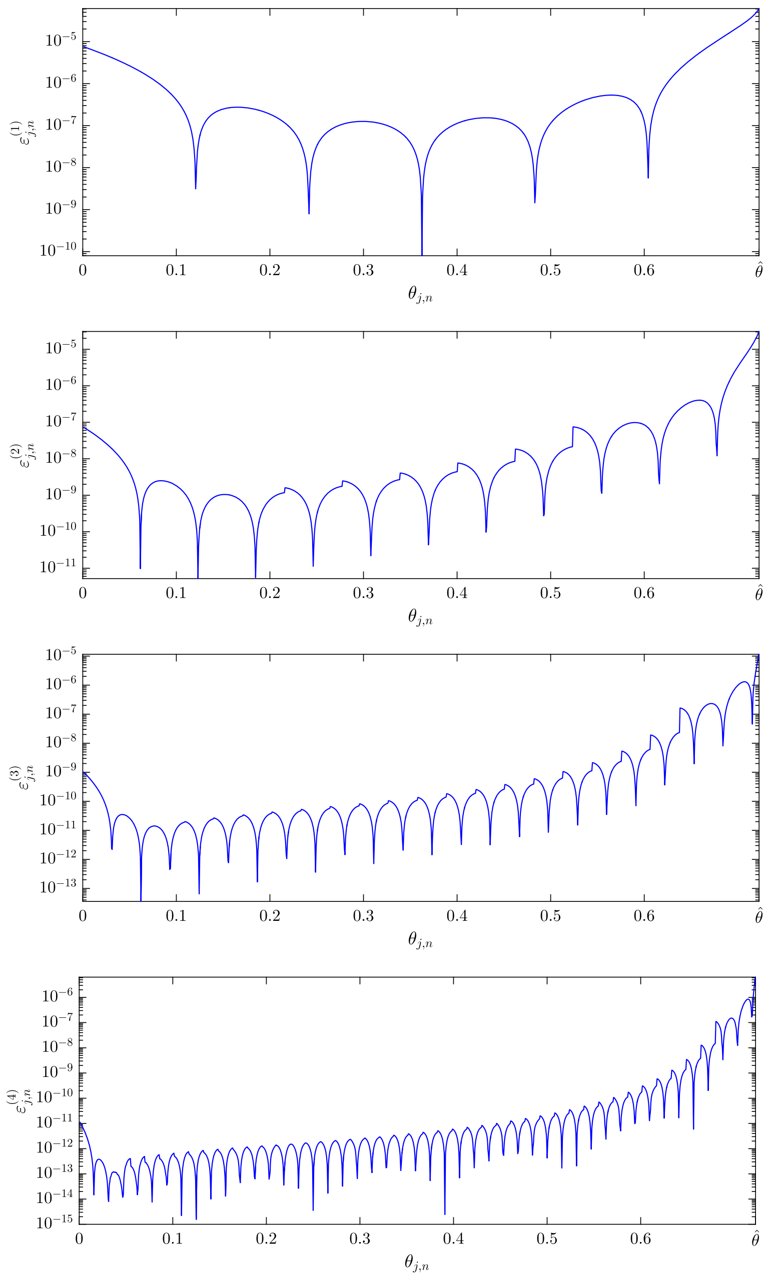

Fig. 12 Example 10: errors $\varepsilon_{j,n}^{(m)}$ versus $\theta_{j,n}$ for $\theta_{j,n} \in I = (0, \hat{\theta})$, in the case where $u(\theta) = 2 + \cos(3\theta)$, $v(\theta) = 8 - 3\cos(\theta) - \frac{9}{2}\cos(2\theta) + 4\cos(3\theta) - \frac{1}{2}\cos(4\theta) - \cos(5\theta)$, $n = 5000$, $n_1 = 25 \cdot 2^{m-1}$, and $\alpha = 5$

$\theta_{j,n} \in I$ and $m = 1, 2, 3, 4$. Considerations analogous to those in Example 10 apply also in this case.

5 Conclusions and perspectives

We have proposed and analyzed a matrix-less parallel interpolation–extrapolation algorithm for computing the eigenvalues of preconditioned banded symmetric Toeplitz matrices of the form $T_n(u)^{-1}T_n(v)$, where u, v are RCTPs, $u > 0$ on $(0, \pi)$, and $f = v/u$ is monotone on $(0, \pi)$. We have illustrated the performance of the algorithm through numerical experiments, and we have presented its generalization to the case where $f = v/u$ is non-monotone. We conclude by suggesting two possible future lines of research:

- Algorithm 1, as well as its generalized version for the non-monotone case (Algorithm 2), is based on a local interpolation strategy, as described in Section 2.1. An interesting topic for future research could be the following: try another kind of approximation (for example, an higher-order spline approximation) to see whether this reduces the errors and accelerates the convergence of both these algorithms.
- Understand whether an asymptotic eigenvalue expansion analogous to (7) holds without the hypothesis that f restricted to some interval $I \subseteq (0, \pi)$ is monotone and satisfies $f^{-1}(f(I)) = I$. Such a result would eliminate any limitation in the applicability of Algorithm 2 (provided that the latter is properly modified according to the new expansion).

Funding Information The research of Sven-Erik Ekström is cofinanced by the Graduate School in Mathematics and Computing (FMB) and Uppsala University. Carlo Garoni is a Marie-Curie fellow of the Italian INdAM (Istituto Nazionale di Alta Matematica) under grant agreement PCOFUND-GA-2012-600198.

Appendix A

This appendix collects the proofs of Theorems 1 and 2.

Proof of Theorem 1 We follow the argument in [1, Section 2]. Equations (2) and (4) can be rewritten as

$$A(h_1, \ldots, h_1)\mathbf{c}(j_1) = \mathbf{E}_0(j_1) - \mathbf{E}_\alpha(j_1) \tag{9}$$

$$A(h_1, \ldots, h_1)\tilde{\mathbf{c}}(j_1) = \mathbf{E}_0(j_1), \tag{10}$$

where

$$\mathbf{c}(j_1) = \begin{bmatrix} c_1(\theta_{j_1,n_1}) \\ \vdots \\ c_\alpha(\theta_{j_1,n_1}) \end{bmatrix}, \qquad \tilde{\mathbf{c}}(j_1) = \begin{bmatrix} \tilde{c}_1(\theta_{j_1,n_1}) \\ \vdots \\ \tilde{c}_\alpha(\theta_{j_1,n_1}) \end{bmatrix},$$

$$\mathbf{E}_0(j_1) = \begin{bmatrix} E_{j_1,n_1,0} \\ \vdots \\ E_{j_\alpha,n_\alpha,0} \end{bmatrix}, \qquad \mathbf{E}_\alpha(j_1) = \begin{bmatrix} E_{j_1,n_1,\alpha} \\ \vdots \\ E_{j_\alpha,n_\alpha,\alpha} \end{bmatrix}, \tag{11}$$

and

$$A(h_1, \ldots, h_\alpha) = \operatorname{diag}(h_1, \ldots, h_\alpha)\, V(h_1, \ldots, h_\alpha), \tag{12}$$

with $V(h_1, \ldots, h_\alpha)$ being the Vandermonde matrix associated with the nodes $h_1, \ldots, h_\alpha$,

$$V(h_1, \ldots, h_\alpha) = \begin{bmatrix} 1 & h_1 & h_1^2 & \cdots & h_1^{\alpha-1} \\ 1 & h_2 & h_2^2 & \cdots & h_2^{\alpha-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & h_\alpha & h_\alpha^2 & \cdots & h_\alpha^{\alpha-1} \end{bmatrix}.$$

By (9), (10), and (12), we have

$$\tilde{\mathbf{c}}(j_1) - \mathbf{c}(j_1) = A(h_1, \ldots, h_\alpha)^{-1}\mathbf{E}_\alpha(j_1) = V(h_1, \ldots, h_\alpha)^{-1}\mathbf{F}_\alpha(j_1),$$

where

$$\mathbf{F}_\alpha(j_1) = \operatorname{diag}(h_1, \ldots, h_\alpha)^{-1}\mathbf{E}_\alpha(j_1) = \begin{bmatrix} E_{j_1,n_1,\alpha}/h_1 \\ \vdots \\ E_{j_\alpha,n_\alpha,\alpha}/h_\alpha \end{bmatrix}.$$

Note that, by (3),

$$|(\mathbf{F}_\alpha(j_1))_k| = |E_{j_k,n_k,\alpha}/h_k| \le C_\alpha h_k^\alpha, \qquad k = 1, \ldots, \alpha. \tag{13}$$

The inverse of $V(h_1, \ldots, h_\alpha)$ is explicitly given by

$$(V(h_1, \ldots, h_\alpha)^{-1})_{ij} = \begin{cases} (-1)^{\alpha-i} \dfrac{\displaystyle\sum_{\substack{1 \le k_1 < \ldots < k_{\alpha-i} \le \alpha \\ k_1, \ldots, k_{\alpha-i} \ne j}} h_{k_1} \cdots h_{k_{\alpha-i}}}{\displaystyle\prod_{\substack{1 \le k \le \alpha \\ k \ne j}} (h_j - h_k)}, & 1 \le i < \alpha, \\[2em] \dfrac{1}{\displaystyle\prod_{\substack{1 \le k \le \alpha \\ k \ne j}} (h_j - h_k)}, & i = \alpha. \end{cases} \tag{14}$$

Taking into account (13) and the equation $h_k = 2^{1-k}h_1$ for $k = 1, \ldots, \alpha$, we obtain the following:

- For $i = \alpha$,

$$\begin{aligned}
&|\tilde{c}_\alpha(\theta_{j_1,n_1}) - c_\alpha(\theta_{j_1,n_1})| = |(\tilde{\mathbf{c}}(j_1) - \mathbf{c}(j_1))_\alpha| \\
&= \left|\sum_{j=1}^{\alpha}(V(h_1, \ldots, h_\alpha)^{-1})_{\alpha j}(\mathbf{F}_\alpha(j_1))_j\right| \\
&\le \sum_{j=1}^{\alpha} \frac{|(\mathbf{F}_\alpha(j_1))_j|}{\prod\limits_{\substack{1\le k\le\alpha \\ k\ne j}} |h_j - h_k|} \le \sum_{j=1}^{\alpha} \frac{C_\alpha h_j^\alpha}{h_j^{\alpha-1} \prod\limits_{\substack{1\le k\le\alpha \\ k\ne j}} |1 - h_k/h_j|} \\
&= C_\alpha h_1 \sum_{j=1}^{\alpha} \frac{2^{1-j}}{\prod\limits_{\substack{1\le k\le\alpha \\ k\ne j}} |1 - 2^{j-k}|} = A(\alpha)h_1,
\end{aligned}$$

  with $A(\alpha)$ depending only on α, u, v just like C_α.
- For $1 \le i < \alpha$,

$$\begin{aligned}
&|\tilde{c}_i(\theta_{j_1,n_1}) - c_i(\theta_{j_1,n_1})| = |(\tilde{\mathbf{c}}(j_1) - \mathbf{c}(j_1))_i| \\
&= \left|\sum_{j=1}^{\alpha}(V(h_1, \ldots, h_\alpha)^{-1})_{ij}(\mathbf{F}_\alpha(j_1))_j\right| \\
&\le \sum_{j=1}^{\alpha} \frac{|(\mathbf{F}_\alpha(j_1))_j| \sum\limits_{\substack{1\le k_1<\ldots<k_{\alpha-i}\le\alpha \\ k_1,\ldots,k_{\alpha-i}\ne j}} h_{k_1}\cdots h_{k_{\alpha-i}}}{\prod\limits_{\substack{1\le k\le\alpha \\ k\ne j}} |h_j - h_k|} \\
&\le \sum_{j=1}^{\alpha} \frac{C_\alpha h_j^\alpha \sum\limits_{\substack{1\le k_1<\ldots<k_{\alpha-i}\le\alpha \\ k_1,\ldots,k_{\alpha-i}\ne j}} h_{k_1}\cdots h_{k_{\alpha-i}}}{h_j^{\alpha-1}\prod\limits_{\substack{1\le k\le\alpha \\ k\ne j}} |1 - h_k/h_j|} \\
&= C_\alpha h_1^{\alpha-i+1} \sum_{j=1}^{\alpha} \frac{2^{1-j} \sum\limits_{\substack{1\le k_1<\ldots<k_{\alpha-i}\le\alpha \\ k_1,\ldots,k_{\alpha-i}\ne j}} 2^{1-k_1}\cdots 2^{1-k_{\alpha-i}}}{\prod\limits_{\substack{1\le k\le\alpha \\ k\ne j}} |1 - 2^{j-k}|} \\
&= A(\alpha, i)h_1^{\alpha-i+1},
\end{aligned}$$

  with $A(\alpha, i)$ depending only on α, i, u, v.

In conclusion, Theorem 1 is proved with $A_\alpha = \max_{i=1,\dots,\alpha} A(\alpha, i)$, where $A(\alpha, \alpha) = A(\alpha)$. □

Proof of Theorem 2 Let $L_1, \dots, L_{\alpha-k+1}$ be the Lagrange polynomials associated with the nodes $\theta^{(1)}, \dots, \theta^{(\alpha-k+1)}$,

$$L_r(\theta) = \prod_{\substack{s=1\\ s\neq r}}^{\alpha-k+1} \frac{\theta - \theta^{(s)}}{\theta^{(r)} - \theta^{(s)}}, \qquad r = 1, \dots, \alpha - k + 1.$$

The interpolation polynomial of the data $(\theta^{(1)}, \tilde{c}_k(\theta^{(1)})), \dots, (\theta^{(\alpha-k+1)}, \tilde{c}_k(\theta^{(\alpha-k+1)}))$ is

$$\tilde{c}_{k,j}(\theta) = \sum_{r=1}^{\alpha-k+1} \tilde{c}_k(\theta^{(r)}) L_r(\theta)$$

and the interpolation polynomial of the data $(\theta^{(1)}, c_k(\theta^{(1)})), \dots, (\theta^{(\alpha-k+1)}, c_k(\theta^{(\alpha-k+1)}))$ is

$$p(\theta) = \sum_{r=1}^{\alpha-k+1} c_k(\theta^{(r)}) L_r(\theta).$$

Considering that $\theta^{(1)}, \dots, \theta^{(\alpha-k+1)}$ are $\alpha - k + 1$ points from $\{\theta_{1,n_1}, \dots, \theta_{n_1,n_1}\}$ which are closest to $\theta_{j,n}$, the length of the smallest interval I containing the nodes $\theta^{(1)}, \dots, \theta^{(\alpha-k+1)}$ and the point $\theta_{j,n}$ is bounded by $(\alpha - k + 1)\pi h_1$. Hence, by Theorem 1, for all $\theta \in I$ we have

$$\begin{aligned} |\tilde{c}_{k,j}(\theta) - p(\theta)| &\le \sum_{r=1}^{\alpha-k+1} |\tilde{c}_{k,j}(\theta^{(r)}) - c_k(\theta^{(r)})| \prod_{\substack{s=1\\ s\neq r}}^{\alpha-k+1} \frac{|\theta - \theta^{(s)}|}{|\theta^{(r)} - \theta^{(s)}|} \\ &\le \sum_{r=1}^{\alpha-k+1} A_\alpha h_1^{\alpha-k+1} \prod_{\substack{s=1\\ s\neq r}}^{\alpha-k+1} \frac{(\alpha - k + 1)\pi h_1}{\pi h_1} \\ &= A_\alpha h_1^{\alpha-k+1} (\alpha - k + 1)^{\alpha-k+1}. \end{aligned} \tag{15}$$

Since $c_k \in C^{\alpha-k+1}([0, \pi])$ by assumption, from interpolation theory we know that for every $\theta \in I$ there exists $\xi(\theta) \in I$ such that

$$c_k(\theta) - p(\theta) = \frac{c_k^{(\alpha-k+1)}(\xi(\theta))}{(\alpha - k + 1)!} \prod_{r=1}^{\alpha-k+1} (\theta - \theta^{(r)});$$

see, e.g., [12, Theorem 3.1.1]. Thus, for all $\theta \in I$, we have

$$\begin{aligned}|c_k(\theta) - p(\theta)| &\le \frac{|c_k^{(\alpha-k+1)}(\xi(\theta))|}{(\alpha-k+1)!} \prod_{r=1}^{\alpha-k+1} |\theta - \theta^{(r)}| \\ &\le \frac{\|c_k^{(\alpha-k+1)}\|_\infty}{(\alpha-k+1)!} \prod_{r=1}^{\alpha-k+1} (\alpha-k+1)\pi h_1 \\ &= \frac{(\alpha-k+1)^{\alpha-k+1}\pi^{\alpha-k+1}\|c_k^{(\alpha-k+1)}\|_\infty}{(\alpha-k+1)!} h_1^{\alpha-k+1}. \end{aligned} \tag{16}$$

From (15) and (16) we obtain

$$|c_k(\theta) - \tilde{c}_{k,j}(\theta)| \le B(k,\alpha)h_1^{\alpha-k+1} \le B_\alpha h_1^{\alpha-k+1}, \qquad \theta \in I, \tag{17}$$

where

$$B(k,\alpha) = \frac{(\alpha-k+1)^{\alpha-k+1}\pi^{\alpha-k+1}\|c_k^{(\alpha-k+1)}\|_\infty}{(\alpha-k+1)!} + A_\alpha(\alpha-k+1)^{\alpha-k+1}$$

and $B_\alpha = \max_{i=1,\ldots,\alpha} B(i,\alpha)$. Since $\theta_{j,n} \in I$, it is clear that (6) follows from (17). □

Appendix B

This appendix provides a plain MATLAB implementation of Algorithm 1.

```
function lambdaS = eigs_preconditioned_toeplitz(n,cu,cv,n1,alpha,S)
% INPUT
%        n: positive integer (size of X_n = T_n(u)^(-1) * T_n(v))
%       cu: row vector of the coefficients of the trigonometric polynomial
%           u(t) = cu(1)+2*cu(2)*cos(t)+...+2*cu(end)*cos((end-1)*t)
%       cv: row vector of the coefficients of the trigonometric polynomial
%           v(t) = cv(1)+2*cv(2)*cos(t)+...+2*cv(end)*cos((end-1)*t)
%       n1: positive integer (number of points of the coarsest grid
%           theta_{j1,n1} = j1*pi/(n1+1), j1=1,...,n1)
%    alpha: positive integer (number of coefficients c_k(theta)
%           to be approximated on the coarsest grid by the tilde c_k(theta))
%        S: row vector containing the indices corresponding to the
%           eigenvalues of X_n to be computed; the indices should be sorted
%           in increasing order, and it is understood that the eigenvalues
%           of X_n are sorted in increasing order as well
% OUTPUT
%  lambdaS: row vector of length length(S) containing the approximations
%           of the eigenvalues of X_n corresponding to the indices S
%           computed by using Algorithm 1 with n1 and alpha as inputs
% FURTHER SPECIFICATIONS
%   This Matlab function works under the same assumptions as in this paper,
%   i.e., u(t), v(t), f(t)=v(t)/u(t) should be as in Conjecture 1 and n1
%   should be greater or equal to alpha
% EXAMPLE (CORRESPONDING TO EXAMPLE 8 OF THIS PAPER)
%   n = 5000; cu = [8, -1.5, -2, -0.5]; cv = [17.5, -6, -3, 0, 0.25];
%   n1 = 100; alpha = 4; S = 1:5;
%   lambdaS = eigs_preconditioned_toeplitz(n,cu,cv,n1,alpha,S)

lu = length(cu); lv = length(cv);
```

```
u = @(t)cu(1)+sum(2*cu(2:lu).*cos((1:lu-1)*t));
v = @(t)cv(1)+sum(2*cv(2:lv).*cos((1:lv-1)*t));
f = @(t) arrayfun(@(t)v(t)./u(t),t);

nn = zeros(1,alpha); hh = zeros(1,alpha);
for k = 1:alpha
    nn(k) = 2^(k-1)*(n1+1)-1;
    hh(k) = 1/(nn(k)+1);
end

A = zeros(alpha);
for i = 1:alpha
    for j = 1:alpha
        A(i,j) = hh(i)^j;
    end
end

E = zeros(alpha,n1);
j1 = 1:n1;
theta = j1*pi*hh(1);
TTu = toeplitz( [cu, sparse(1, nn(alpha) - lu)] );
TTv = toeplitz( [cv, sparse(1, nn(alpha) - lv)] );
for k = 1:alpha
    eigX = sort(eig(full(TTv(1:nn(k),1:nn(k))),full(TTu(1:nn(k),1:nn(k)))));
    jk = 2^(k-1)*j1;
    E(k,:) = eigX(jk)' - f(theta);
end

c_tilde = A\E;

lS = length(S);
lambdaS = zeros(1,lS);
h = 1/(n+1);
t = S*pi*h;
for j = 1:lS
    ell = t(j)*(n1+1)/pi;
    poly_evals = zeros(1,alpha);
    for k = 1:alpha
        indices = localization(ell,alpha-k+1);
        if indices(1)<1
           indices = indices - indices(1) + 1;
        end
        if indices(end)>n1
            indices = indices - indices(end) + n1;
        end
        tt = indices*pi*hh(1);
        poly_evals(k) = polyval(polyfit(tt,c_tilde(k,indices),alpha-k),t(j));
    end
    lambdaS(j) = polyval([poly_evals(end:-1:1) f(t(j))],h);
end

end

function u = localization(x,m)

% INPUT
%        x: real number
%        m: natural number >= 1
% OUTPUT
%        u: row vector of length m such that u(1),...,u(m) are m integers
%           that are closest to x (which are not uniquely determined
%           in some cases)
```

```
b = mod(m,2);
v = (m + b)/2;
fx = floor(x);
cx = ceil(x);

if x - fx <= cx - x
   u = (fx - v + 1):(fx + v - b);
else
   u = (cx - v + b):(cx + v - 1);
end

end
```

References

1. 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. Alg. (in press). https://doi.org/10.1007/s11075-017-0404-z
2. Arbenz, P.: Computing the eigenvalues of banded symmetric Toeplitz matrices. SIAM J. Sci. Stat. Comput. **12**, 743–754 (1991)
3. Badía, J.M., Vidal, A.M.: Parallel algorithms to compute the eigenvalues and eigenvectors of symmetric Toeplitz matrices. Parallel Algorithms Appl. **13**, 75–93 (2000)
4. Bini, D., Di Benedetto, F.: Solving the generalized eigenvalue problem for rational Toeplitz matrices. SIAM J. Matrix Anal. Appl. **11**, 537–552 (1990)
5. Bini, D., Pan, V.: Efficient algorithms for the evaluation of the eigenvalues of (block) banded Toeplitz matrices. Math. Comput. **50**, 431–448 (1988)
6. 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)
7. Bogoya, J.M., Böttcher, A., Grudsky, S.M., Maximenko, E.A.: Maximum norm versions of the Szegő and Avram–Parter theorems for Toeplitz matrices. J. Approx. Theory **196**, 79–100 (2015)
8. 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)
9. Böttcher, A., Silbermann, B.: Introduction to Large Truncated Toeplitz Matrices. Springer, New York (1999)
10. 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)
11. Brezinski, C., Redivo Zaglia, M.: Extrapolation Methods: Theory and Practice. North-Holland, Elsevier Science Publishers B.V., Amsterdam (1991)
12. Davis, P.J.: Interpolation and Approximation. Dover, New York (1975)
13. Ekström, S.-E., Garoni, C., Serra-Capizzano, S.: Are the eigenvalues of banded symmetric Toeplitz matrices known in almost closed form? Exper. Math. (in press). https://doi.org/10.1080/10586458.2017.1320241
14. Garoni, C., Serra-Capizzano, S.: Generalized Locally Toeplitz Sequences: Theory and Applications, vol. I. Springer, Cham (2017)
15. Stoer, J., Bulirsch, R.: Introduction to Numerical Analysis, 3rd edn. Springer, New York (2010)
16. Trench, W.F.: On the eigenvalue problem for Toeplitz band matrices. Linear Algebra Appl. **64**, 199–214 (1985)
17. Trench, W.F.: Characteristic polynomials of symmetric rationally generated Toeplitz matrices. Linear Multilinear Algebra **21**, 289–296 (1987)
18. Trench, W.F.: Numerical solution of the eigenvalue problem for symmetric rationally generated Toeplitz matrices. SIAM J. Matrix Anal. Appl. **9**, 291–303 (1988)
19. Trench, W.F.: Numerical solution of the eigenvalue problem for Hermitian Toeplitz matrices. SIAM J. Matrix Anal. Appl. **10**, 135–146 (1989)
20. Trench, W.F.: Numerical solution of the eigenvalue problem for efficiently structured Hermitian matrices. Linear Algebra Appl. **154–156**, 415–432 (1991)