MATHEMATICS OF COMPUTATION
Volume 75, Number 254, Pages 817–832
S 0025-5718(05)01796-5
Article electronically published on December 1, 2005

COMPUTATION OF THE NEWTON STEP FOR THE EVEN AND ODD CHARACTERISTIC POLYNOMIALS OF A SYMMETRIC POSITIVE DEFINITE TOEPLITZ MATRIX

A. MELMAN

Abstract. We compute the Newton step for the characteristic polynomial and for the even and odd characteristic polynomials of a symmetric positive definite Toeplitz matrix as the reciprocal of the trace of an appropriate matrix. We show that, after the Yule–Walker equations are solved, this trace can be computed in $\mathcal{O}(n)$ additional arithmetic operations, which is in contrast to existing methods, which rely on a recursion, requiring $\mathcal{O}(n^2)$ additional arithmetic operations.

1. Introduction

In this paper we propose an efficient way to carry out Newton's method for the computation of the roots of the characteristic polynomial and the characteristic even and odd polynomials of a symmetric positive definite Toeplitz matrix. We then apply this to improve the complexity of the methods in [11] and [14], where such an approach is used to compute the smallest eigenvalue of a symmetric positive definite Toeplitz matrix.

To compute the Newton step, these methods use a recursion for the evaluation of the characteristic polynomial and its derivative which requires $\mathcal{O}(n^2)$ arithmetic operations in addition to solving the Yule–Walker equations. We show that this recursion can be replaced by a shorter computation, involving the computation of the trace of an appropriate matrix, and, after solving the Yule–Walker equations, requiring only $\mathcal{O}(n)$ additional arithmetic operations.

The advantage of the methods in [11] and [14] is their relative simplicity. Their disadvantage is that they are slower than methods based on the secular equation (see, e.g., [4], [12], and the references therein). The aforementioned reduction in complexity makes these simpler methods competitive. Our results are also applicable to methods other than Newton's method.

In Section 2 we present the basic properties of Toeplitz matrices and the notation we will use. In Section 3 we compute the Newton step for the characteristic polynomial, and in Section 4 we do the same thing for the even and odd characteristic polynomials. In Section 5 we compare the methods.

Received by the editor April 29, 2004 and, in revised form, November 11, 2004.
2000 *Mathematics Subject Classification.* Primary 65F15; Secondary 15A18.
Key words and phrases. Toeplitz matrix, even, odd, eigenvalue, characteristic polynomial, Newton's method.

2. Preliminaries

The (i,j)-th element of an $n \times n$ symmetric Toeplitz matrix T_n is given by $\rho_{|i-j|}$ for some vector $(\rho_0, \rho_1, \ldots, \rho_{n-1})^T \in \mathbb{R}^n$. The matrix T_n satisfies $JT_nJ = T_n$ and is therefore centrosymmetric. We use I for the identity matrix and J for the exchange, or "flip", matrix with ones on its southwest-northeast diagonal and zeros everywhere else. For simplicity's sake, our notation will not explicitly indicate the dimensions of the matrices I and J.

An *even vector* v is defined as a vector satisfying $Jv = v$ and an *odd vector* w as one that satisfies $Jw = -w$. If these vectors are eigenvectors, then their associated eigenvalues are called *even* and *odd*, respectively. It was shown in [3] that, given a real symmetric centrosymmetric matrix of order n, there exists an orthonormal basis for $\mathbb{R}^n$, composed of $n - \lfloor n/2 \rfloor$ even and $\lfloor n/2 \rfloor$ odd eigenvectors, where $\lfloor \alpha \rfloor$ denotes the integral part of α.

Finally, we note that for any $\lambda \in \mathbb{R}$, the matrix $(T_n - \lambda I)$ is symmetric and centrosymmetric, whenever T_n is.

For any $n \times n$ symmetric Toeplitz matrix T_n we have

$$T_n = \begin{pmatrix} A & B \\ JBJ & A \end{pmatrix} \quad \text{or} \quad T_n = \begin{pmatrix} A & s & B \\ s^T & \rho_0 & s^TJ \\ JBJ & Js & A \end{pmatrix},$$

depending on whether n is even or odd, respectively. For even n, the blocks in the matrix T_n have $\frac{n}{2}$ rows and columns. For odd n, the blocks have $\frac{n-1}{2}$ rows and columns. The column vector s has dimension $\frac{n-1}{2}$. Since T_n^{-1} is also symmetric and centrosymmetric, although not necessarily Toeplitz, we have

$$T_n^{-1} = \begin{pmatrix} H_1 & H_2 \\ JH_2J & JH_1J \end{pmatrix} \quad \text{or} \quad T_n^{-1} = \begin{pmatrix} H_1 & q & H_2 \\ q^T & \alpha & q^TJ \\ JH_2J & Jq & JH_1J \end{pmatrix},$$

depending on whether n is even or odd, respectively, with $JH_2J = H_2^T$, i.e., H_2 is persymmetric.

In what follows, an important role is played by the so-called *Yule–Walker equations*. For an $n \times n$ symmetric Toeplitz matrix T_n, defined by $(\rho_0, \rho_1, \ldots, \rho_{n-1})$, this system of linear equations is given by $T_n y^{(n)} = -t_n$, where $t_n = (\rho_1, \ldots, \rho_n)^T$. There exist several methods to solve these equations. Durbin's algorithm ([7]) solves them by recursively computing the solutions to lower-dimensional systems, provided all principal submatrices are nonsingular. This algorithm requires $2n^2 + \mathcal{O}(n)$ flops, which we define as in [9].

A more efficient method than Durbin's algorithm is what we will call the *split Durbin algorithm* from [6], where it is called the "split Levinson algorithm". We prefer this terminology because "Durbin" usually refers to the Yule–Walker equations, which have a special right-hand side, whereas "Levinson" usually refers to a system with an arbitrary right-hand side. In that we also follow [9]. To briefly explain the split Durbin algorithm, we define an *even* solution $u^{(k)}$ of the Yule–Walker equations $T_k y^{(k)} = -t_k$ as the solution of $T_k u^{(k)} = -(t_k + Jt_k)$, or $u^{(k)} = y^{(k)} + Jy^{(k)}$, and an *odd* solution as the solution of $T_k v^{(k)} = -(t_k - Jt_k)$, or $v^{(k)} = y^{(k)} - Jy^{(k)}$. This algorithm is based on the remarkable observation that the solution $y^{(k)}$ can be written either as a combination of the two successive even solutions $u^{(k)}$ and $u^{(k-1)}$ or as a combination of the two successive odd solutions $v^{(k)}$ and $v^{(k-1)}$. It is

therefore sufficient to compute either the even or the odd solutions. For full details, we refer to [6], or [13] where it is summarized in the same notation as here. The split Durbin algorithm requires $\frac{3}{2}n^2 + \mathcal{O}(n)$ flops.

Finally, we mention that there also exist so-called "superfast methods" to solve the Yule–Walker equations (see, e.g., [1], [2]). However, for matrices with dimensions up to several hundred, they are less efficient than the algorithms mentioned here, which are usually referred to as "fast methods".

Newton's method for solving $f(x) = 0$, where $f : \mathbb{R} \to \mathbb{R}$, is an iterative method, defined by

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \cdot$$

We refer to $N(\bar{x}) \stackrel{\triangle}{=} -f(\bar{x})/f'(\bar{x})$ as the Newton step for f at $x = \bar{x}$.

Throughout this paper, all matrices are assumed to be of dimension $n \times n$, with $n \geq 3$.

3. The Newton step for the characteristic polynomial

The characteristic polynomial for the symmetric matrix T_n is given by $p_n(\lambda) \stackrel{\triangle}{=} det(T_n - \lambda I)$. If Newton's method were used to compute the roots of this polynomial, then the following lemma gives a convenient (and well-known) expression for the Newton step.

Lemma 3.1. *The Newton step for the characteristic polynomial $p_n(\lambda)$ of an $n \times n$ matrix T_n at $\lambda = \bar{\lambda}$, where $\bar{\lambda}$ is not one of the eigenvalues of T_n, is given by*

$$N(\bar{\lambda}) = -\frac{p_n(\bar{\lambda})}{(p_n(\bar{\lambda}))'} = \frac{1}{tr(T_n - \bar{\lambda} I)^{-1}} \cdot \tag{1}$$

□

We now propose to use the Gohberg–Semencul formula (see [8]) for the inverse of a Toeplitz matrix to evaluate the right-hand side in (1). As in [11], we will assume that T_n is a symmetric positive definite Toeplitz matrix, which has no eigenvalues in common with its principal submatrix T_{n-1}.

With the first row of T_n given by (ρ_0, t^T), we first define

Definition 3.2.

$$\chi(\lambda) \stackrel{\triangle}{=} \rho_0 - \lambda - t^T (T_{n-1} - \lambda I)^{-1} t \, .$$

With $w = -JT_{n-1}^{-1}t$, the Gohberg–Semencul formula for the inverse of T_n is then given by

$$T_n^{-1} = \frac{1}{\chi(0)} \left(M_1 M_1^T - M_0^T M_0 \right) \, , \tag{2}$$

where

(3)

$$M_0 = \begin{pmatrix} 0 & 0 & \dots & & 0 \\ w_1 & 0 & \ddots & & 0 \\ w_2 & w_1 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 & 0 \\ w_{n-1} & w_{n-2} & \dots & w_1 & 0 \end{pmatrix}, \; M_1 = \begin{pmatrix} 1 & w_{n-1} & w_{n-2} & \dots & w_1 \\ 0 & 1 & w_{n-1} & \dots & w_2 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & & \ddots & 1 & w_{n-1} \\ 0 & 0 & \dots & 0 & 1 \end{pmatrix}.$$

In the following theorem, we compute the Newton step, using the Gohberg–Semencul formula.

Theorem 3.3. *The Newton step for the characteristic polynomial $p_n(\lambda)$ of an $n \times n$ symmetric positive definite Toeplitz matrix T_n at $\lambda = \bar{\lambda} < \lambda_{\min}(T_n)$ is given by*

$$N(\bar{\lambda}) = \frac{\chi(\bar{\lambda})}{\sum_{j=1}^{n}(2j-n)w_j^2}$$

or, equivalently, by

$$\begin{aligned} N(\bar{\lambda}) &= \frac{\chi(\bar{\lambda})}{n + \sum_{j=1}^{\frac{n}{2}-1}(n-2j)(w_{n-j}^2 - w_j^2)} && \text{(n is even)}, \\ N(\bar{\lambda}) &= \frac{\chi(\bar{\lambda})}{n + \sum_{j=1}^{\lfloor \frac{n}{2} \rfloor}(n-2j)(w_{n-j}^2 - w_j^2)} && \text{(n is odd)}, \end{aligned}$$

with χ as in Definition 3.2, and where the first row of T_n is given by (ρ_0, t^T) and $w = (w_1, \ldots, w_{n-1}) = -J(T_{n-1} - \bar{\lambda}I)^{-1}t$. For compactness of writing, we have set $w_n = 1$.

Proof. If T_n is symmetric positive definite, then for $\bar{\lambda} < \lambda_{\min}(T_n)$, so is $(T_n - \bar{\lambda}I)$. The Newton step at $\bar{\lambda}$ can therefore be determined by computing the trace of $(T_n - \bar{\lambda}I)^{-1}$ using the Gohberg–Semencul formula, which, by Lemma 3.1 and (2), yields

$$N(\bar{\lambda}) = \frac{\chi(\bar{\lambda})}{\operatorname{tr}\left(M_1 M_1^T - M_0^T M_0\right)}\,, \tag{4}$$

where M_0 and M_1 are as in (3) with $w = -J(T_{n-1} - \bar{\lambda}I)^{-1}t$.

By simple matrix multiplication, we get

$$\begin{aligned} \left(M_1 M_1^T - M_0^T M_0\right)_{j,j} &= \sum_{k=j}^{n} w_k^2 - \sum_{k=1}^{n-j} w_k^2 && (1 \le j \le n-1), \\ \left(M_1 M_1^T - M_0^T M_0\right)_{n,n} &= w_n^2 = 1\,. \end{aligned}$$

The trace of $\left(M_1 M_1^T - M_0^T M_0\right)$ is then given by

$$\begin{aligned} \operatorname{tr}\left(M_1 M_1^T - M_0^T M_0\right) &= w_n^2 + \sum_{j=1}^{n-1}\left(M_1 M_1^T - M_0^T M_0\right)_{j,j} \\ &= w_n^2 + \sum_{j=1}^{n-1}\left(\sum_{k=j}^{n} w_k^2 - \sum_{k=1}^{n-j} w_k^2\right) \\ &= w_n^2 + \sum_{j=1}^{n-1}\sum_{k=j}^{n} w_k^2 - \sum_{j=1}^{n-1}\sum_{k=1}^{n-j} w_k^2\,. \end{aligned} \tag{5}$$

The two double sums on the right-hand side of (5) are each the sum of all elements of a triangular array, summed row by row. If, instead, these elements are summed

column by column, one obtains

$$\begin{aligned}\operatorname{tr}\left(M_1M_1^T - M_0^TM_0\right) &= w_n^2 + (n-1)w_n^2 + \sum_{i=1}^{n-1} iw_i^2 - \sum_{i=1}^{n-1}(n-i)w_i^2 \\ &= \sum_{i=1}^{n}(2i-n)w_i^2 \ .\end{aligned}$$

Because $2i - n = -(2(n-i) - n)$, the trace can also be written as

$$\operatorname{tr}\left(M_1M_1^T - M_0^TM_0\right) = \begin{cases} n + \sum_{i=1}^{\frac{n}{2}-1}(n-2i)(w_{n-i}^2 - w_i^2) & (n \text{ is even}), \\ \\ n + \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor}(n-2i)(w_{n-i}^2 - w_i^2) & (n \text{ is odd}) \ . \end{cases} \tag{6}$$

Alternatively, one could use the fact that $\left(M_1M_1^T - M_0^TM_0\right)$ is centrosymmetric so that roughly only half of the diagonal elements need to be summed.

Substituting these expressions for $\operatorname{tr}\left(M_1M_1^T - M_0^TM_0\right)$ back into (4) concludes the proof. □

This means that, once the $(n-1)$-dimensional Yule–Walker system is solved (this needs to be done at every iteration), yielding $(T_{n-1} - \bar{\lambda}I)^{-1}t$, only $\mathcal{O}(n)$ flops are necessary to compute the Newton step. The total number of flops depends on the algorithm one uses for the computation of $(T_{n-1} - \bar{\lambda}I)^{-1}t$. If the split Durbin algorithm ([7]) is used, then a total of $\frac{3}{2}n^2 + \mathcal{O}(n)$ flops are needed.

This approach is in contrast to the one in [11], where the Newton step is determined by a recursion, given by

$$N_k(\bar{\lambda}) = \frac{\beta_k}{\beta_k(N_{k-1}(\bar{\lambda}))^{-1} + \left(1 + \|y^{(k-1)}\|^2\right)} \ , \tag{7}$$

where the quantities β_k are computed in the course of Durbin's algorithm, $y^{(k-1)}$ is the solution of the $(k-1)$-th Yule–Walker system, and $N_k(\bar{\lambda})$ is the Newton step for the the characteristic polynomial of the k-th principal submatrix of T_n at $\lambda = \bar{\lambda}$. If Durbin's algorithm is used, as in [11], this recursion requires $3n^2 + \mathcal{O}(n)$ flops, namely $2n^2 + \mathcal{O}(n)$ flops for Durbin's algorithm and another n^2 flops for the computation of the norms in the recursion formula. We note that using the split Durbin algorithm for this recursion instead of Durbin's algorithm would not be appropriate, as it would increase the number of flops, due to the need for the solutions of the intermediate Yule–Walker equations.

Therefore, our approach here roughly cuts the number of flops per Newton iteration in half when compared to the method in [11]. Moreover, since there is no need in our approach to compute the solutions of the intermediate Yule–Walker equations as in (7), any method which yields $(T_{n-1} - \bar{\lambda}I)^{-1}t$ can now be used. This is very important because the so-called superfast methods for this problem (see, e.g., [1] and [2]) have a complexity of only $\mathcal{O}(n \ln^2(n))$ versus $\mathcal{O}(n^2)$ flops for the fast algorithms that we mentioned before. This would provide a dramatic speedup.

Remark. It may be better numerically to compute quantities of the form $a^2 - b^2$, which appear in the previous theorem and elsewhere in this paper, as $(a-b)(a+b)$.

4. The Newton step for the even and odd characteristic polynomials

As we mentioned in the preliminaries, symmetric Toeplitz matrices have even and odd eigenvalues, which, as we will see, are the roots of the even and odd characteristic polynomials, respectively. In [14], the smallest even and odd eigenvalues are determined by applying Newton's method to the even and odd characteristic polynomials. As in the case of the characteristic polynomial (which has both the even and odd eigenvalues as its zeros), a recursion is used to achieve this and, as we will show, this recursion, too, can be replaced by a trace computation. We start with the following lemma, which is a special case of Lemma 3 in [3].

Lemma 4.1. *For a symmetric Toeplitz matrix T_n, defined by $(\rho_0, \ldots, \rho_{n-1})^T$ when n is even, the following holds:*

$$KT_nK^T = \begin{pmatrix} A-BJ & 0 \\ 0 & A+BJ \end{pmatrix}, \text{ with } K = \tfrac{1}{\sqrt{2}} \begin{pmatrix} I & -J \\ I & J \end{pmatrix}.$$

When n is odd, then

$$KT_nK^T = \begin{pmatrix} A-BJ & 0 & 0 \\ 0^T & \rho_0 & \sqrt{2}s^T \\ 0 & \sqrt{2}s & A+BJ \end{pmatrix}, \text{ with } K = \frac{1}{\sqrt{2}} \begin{pmatrix} I & 0 & -J \\ 0^T & \sqrt{2} & 0^T \\ I & 0 & J \end{pmatrix}. \qquad \square$$

The matrix K satisfies $KK^T = I = K^TK$. The matrix T_n can therefore be split into two parts. The eigenvalues associated with $A-BJ$ are odd, and the ones associated with the part containing $A+BJ$ are even. This means that the characteristic polynomial of T_n can be factored into two polynomials, one corresponding to the even and the other to the odd eigenvalues, i.e.,

$$\det(T_n - \lambda I) = \det(A - BJ - \lambda I)\det(A + BJ - \lambda I) \quad \text{(even dimension)}$$

and

$$\det(T_n - \lambda I) = \det(A - BJ - \lambda I)\det \begin{pmatrix} \rho_0 - \lambda & \sqrt{2}s^T \\ \sqrt{2}s & A + BJ - \lambda I \end{pmatrix} \quad \text{(odd dimension)}.$$

In both the even and the odd case, we can write this concisely as $p_n(\lambda) = p_n^e(\lambda)p_n^o(\lambda)$. We note that the index n refers to the matrix T_n and not necessarily to the degree of the polynomial to which it is attached. Throughout this paper, the superscripts "e" and "o" refer to even and odd, respectively. The even and odd eigenvalues of T_n interlace the even and odd eigenvalues, respectively, of its principal submatrix T_{n-2} (see, e.g., [5]). As in [14], we will assume that the smallest eigenvalue of T_n is not an eigenvalue of T_{n-2}.

We now compute the Newton step for the aforementioned even and odd characteristic polynomials in terms of the trace of an appropriate matrix.

Lemma 4.2. *The Newton step for the even and odd characteristic polynomials of an $n \times n$ symmetric Toeplitz matrix*

$$T_n = \begin{cases} \begin{pmatrix} A & B \\ JBJ & A \end{pmatrix} & (n \text{ is even}), \\ \begin{pmatrix} A & s & B \\ s^T & \rho_0 & s^TJ \\ JBJ & Js & A \end{pmatrix} & (n \text{ is odd}), \end{cases}$$

at $\lambda = \bar{\lambda}$, where $\bar{\lambda}$ is not one of the eigenvalues of T_n, is given by

$$N^e(\bar{\lambda}) = -\frac{p_n^e(\bar{\lambda})}{(p_n^e(\bar{\lambda}))'} = \begin{cases} \frac{1}{tr\left(A-\bar{\lambda}I+BJ\right)^{-1}} & \textit{(n is even)}, \\ \frac{1}{tr\begin{pmatrix} \rho_0-\bar{\lambda} & \sqrt{2}s^T \\ \sqrt{2}s & A-\bar{\lambda}I+BJ \end{pmatrix}^{-1}} & \textit{(n is odd)} \end{cases}$$

and

$$N^o(\bar{\lambda}) = -\frac{p_n^o(\bar{\lambda})}{(p_n^o(\bar{\lambda}))'} = \frac{1}{tr\left(A-\bar{\lambda}I-BJ\right)^{-1}} \, .$$

Proof. Denoting the even and odd eigenvalues of T_n by $\{\lambda_j^e\}_{j=1}^n$ and $\{\lambda_j^o\}_{j=1}^n$, respectively, and assuming that $\bar{\lambda}$ is not an eigenvalue of T_n, we can write the odd characteristic polynomial as $p_n^o(\lambda) = \prod_{j=1}^{\lfloor \frac{n}{2} \rfloor}(\lambda_j^o - \lambda)$. We then have

$$-\frac{(p_n^o(\bar{\lambda}))'}{p_n^o(\bar{\lambda})} = \frac{\sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} \prod_{j=1, j\neq i}^{\lfloor \frac{n}{2} \rfloor}(\lambda_j^o - \bar{\lambda})}{\prod_{j=1}^{\lfloor \frac{n}{2} \rfloor}(\lambda_j^o - \bar{\lambda})} = \sum_{j=1}^{\lfloor \frac{n}{2} \rfloor} \frac{1}{\lambda_j^o - \bar{\lambda}} \, . \tag{8}$$

However, the odd eigenvalues of T_n are the eigenvalues of $(A - BJ)$, and therefore, the odd eigenvalues of $\left(T_n - \bar{\lambda}I\right)$ are the eigenvalues of $\left(A - \bar{\lambda}I - BJ\right)$. Consequently,

$$\sum_{j=1}^{\lfloor \frac{n}{2} \rfloor} \frac{1}{\lambda_j^o - \bar{\lambda}} = \operatorname{tr}\left(A - \bar{\lambda}I - BJ\right)^{-1} \, .$$

This, together with (8), proves the theorem for the odd Newton step. The statement for the even Newton step follows analogously. □

Remark. In [14], it was shown that the even and odd Newton steps are at least as large in magnitude as the Newton step for the characteristic polynomial. However, the previous lemma gives us an intuitive argument to expect, in fact, a doubling of the magnitude initially, i.e., when the iterates are still relatively far from the smallest eigenvalue. Since $p_n(\bar{\lambda}) = p_n^e(\bar{\lambda})p_n^o(\bar{\lambda})$, we have that $(p_n(\bar{\lambda}))' = (p_n^e(\bar{\lambda}))'p_n^o(\bar{\lambda}) + p_n^e(\bar{\lambda})(p_n^o(\bar{\lambda}))'$ and therefore that

$$\frac{(p_n(\bar{\lambda}))'}{p_n(\bar{\lambda})} = \frac{(p_n^e(\bar{\lambda}))'}{p_n^e(\bar{\lambda})} + \frac{(p_n^o(\bar{\lambda}))'}{p_n^o(\bar{\lambda})} \, ,$$

or, when n is even,

$$\frac{1}{N(\bar{\lambda})} = \frac{1}{N^e(\bar{\lambda})} + \frac{1}{N^o(\bar{\lambda})} = \operatorname{tr}(A - \bar{\lambda}I + BJ)^{-1} + \operatorname{tr}(A - \bar{\lambda}I - BJ)^{-1} \, . \tag{9}$$

An analogous expression is obtained when n is odd. When $\bar{\lambda}$ is relatively far from the smallest eigenvalue, there is, in general, no reason to assume that the even and odd eigenvalues of $(A - \bar{\lambda}I)^{-1}$ would be distributed very differently, so that $\operatorname{tr}(A-\bar{\lambda}I+BJ)^{-1} \approx \operatorname{tr}(A-\bar{\lambda}I-BJ)^{-1}$, or $N^e(\bar{\lambda}) \approx N^o(\bar{\lambda})$. From (9), we then have that $N^e(\bar{\lambda}) \approx N^o(\bar{\lambda}) \approx 2N(\bar{\lambda})$, so that one could expect a significant reduction in the number of Newton steps. The numerical experiments in [14] show clear evidence of this.

Before we can compute the even and odd Newton steps explicitly, we need a few more results. We first have the following lemma.

Lemma 4.3. (1) *For a nonsingular symmetric Toeplitz matrix T of even dimension with*

$$T = \begin{pmatrix} A & B \\ JBJ & A \end{pmatrix} \quad \text{and} \quad T^{-1} = \begin{pmatrix} H_1 & H_2 \\ JH_2J & JH_1J \end{pmatrix},$$

the following holds:

$$(A - BJ)^{-1} = H_1 - H_2J \qquad \text{and} \qquad (A + BJ)^{-1} = H_1 + H_2J\,.$$

(2) *For a nonsingular symmetric Toeplitz matrix T of odd dimension with*

$$T = \begin{pmatrix} A & s & B \\ s^T & \rho_0 & s^TJ \\ JBJ & Js & A \end{pmatrix} \quad \text{and} \quad T^{-1} = \begin{pmatrix} H_1 & q & H_2 \\ q^T & \alpha & q^TJ \\ JH_2J & Jq & JH_1J \end{pmatrix},$$

the following holds:

$$(A-BJ)^{-1} = H_1 - H_2J \quad \text{and} \quad \begin{pmatrix} \rho_0 & \sqrt{2}s^T \\ \sqrt{2}s & A+BJ \end{pmatrix}^{-1} = \begin{pmatrix} \alpha & \sqrt{2}q^T \\ \sqrt{2}q & H_1+H_2J \end{pmatrix}.$$

Proof. (1) When the dimension of T is even and with K as in Lemma (4.1), we have

$$KTK^T = \begin{pmatrix} A - BJ & 0 \\ 0 & A + BJ \end{pmatrix}.$$

Setting $Y_1 = A - BJ$ and $Y_2 = A + BJ$, we obtain that $KT^{-1}K^T = \begin{pmatrix} Y_1^{-1} & 0 \\ 0 & Y_2^{-1} \end{pmatrix}$, from which we get

$$T^{-1} = K^T \begin{pmatrix} Y_1^{-1} & 0 \\ 0 & Y_2^{-1} \end{pmatrix} K = \frac{1}{2} \begin{pmatrix} Y_1^{-1} + Y_2^{-1} & \left(Y_2^{-1} - Y_1^{-1}\right) J \\ J\left(Y_2^{-1} - Y_1^{-1}\right) & J\left(Y_1^{-1} + Y_2^{-1}\right) J \end{pmatrix}.$$

Therefore,

$$T^{-1} = \begin{pmatrix} H_1 & H_2 \\ JH_2J & JH_1J \end{pmatrix} = \frac{1}{2} \begin{pmatrix} Y_1^{-1} + Y_2^{-1} & \left(Y_2^{-1} - Y_1^{-1}\right) J \\ J\left(Y_2^{-1} - Y_1^{-1}\right) & J\left(Y_1^{-1} + Y_2^{-1}\right) J \end{pmatrix},$$

which leads to

$$2H_1 = Y_1^{-1} + Y_2^{-1} \qquad \text{and} \qquad 2H_2 = \left(Y_2^{-1} - Y_1^{-1}\right) J\,.$$

From this we obtain

$$(A - BJ)^{-1} = Y_1^{-1} = H_1 - H_2J \qquad \text{and} \qquad (A + BJ)^{-1} = Y_2^{-1} = H_1 + H_2J\,.$$

This proves the first part of the lemma.

(2) When the dimension of T is odd and with K as in Lemma (4.1), we have

$$KTK^T = \begin{pmatrix} A - BJ & 0 & 0 \\ 0^T & \rho_0 & \sqrt{2}s^T \\ 0 & \sqrt{2}s & A + BJ \end{pmatrix}.$$

Setting $Y = A - BJ$ and $\begin{pmatrix} \rho_0 & \sqrt{2}s^T \\ \sqrt{2}s & A+BJ \end{pmatrix}^{-1} = \begin{pmatrix} \sigma & x^T \\ x & C \end{pmatrix}$, we obtain that

$$KT^{-1}K = \begin{pmatrix} Y^{-1} & 0 & 0 \\ 0^T & \sigma & x^T \\ 0 & x & C \end{pmatrix},$$

and therefore that

$$T^{-1} = K^T \begin{pmatrix} Y^{-1} & 0 & 0 \\ 0^T & \sigma & x^T \\ 0 & x & C \end{pmatrix} K.$$

This means that

$$T^{-1}=\begin{pmatrix} H_1 & q & H_2 \\ q^T & \alpha & q^TJ \\ JH_2J & Jq & JH_1J \end{pmatrix}=\frac{1}{2}\begin{pmatrix} Y^{-1}+C & \sqrt{2}x & -Y^{-1}J+CJ \\ \sqrt{2}x^T & 2\sigma & \sqrt{2}x^TJ \\ -JY^{-1}+JC & \sqrt{2}Jx & JY^{-1}J+JCJ \end{pmatrix},$$

so that

$$Y^{-1}+C=2H_1\ , \quad -Y^{-1}+C=2H_2J\ , \quad \sigma=\alpha\ , \quad \text{and} \quad x=\sqrt{2}q\ .$$

Finally, from this we obtain

$$(A-BJ)^{-1}=Y^{-1}=H_1-H_2J \qquad \text{and} \qquad C=H_1+H_2J\ .$$

This concludes the proof. □

Lemma 4.4. *With M_0, M_1 and $\{w\}_{j=1}^{n-1}$ as in (3), and setting $w_0=0$ and $w_n=1$ for compactness of writing, the antitrace of $\left(M_1M_1^T-M_0^TM_0\right)$ can be computed in $\mathcal{O}(n)$ flops and is given by*

$$\text{antitr}\left(M_1M_1^T-M_0^TM_0\right)=\begin{cases} 2(S_1-S_0) & (n \text{ is even}), \\ 2(S_1-S_0)+w_n^2+\sum_{j=1}^{\lfloor \frac{n}{2} \rfloor}\left(w_{n-j}^2-w_j^2\right) & (n \text{ is odd}), \end{cases}$$

with S_0 and S_1 defined by the following:

(1) $S_0 \triangleq \sum_{m=1}^{\lfloor \frac{n}{2} \rfloor} q_m w_{m-1}$, *where* $q_m \triangleq \sum_{j=m}^{\lfloor \frac{n}{2} \rfloor} w_{n-2j+m}$ *satisfies the recursion*

$$q_m=q_{m+2}+w_{n-m}+w_{n-2\lfloor \frac{n}{2} \rfloor+m}, \text{ with } \begin{cases} q_{\lfloor \frac{n}{2} \rfloor}=w_{n-\lfloor \frac{n}{2} \rfloor}, \\ q_{\lfloor \frac{n}{2} \rfloor-1}=w_{n-\lfloor \frac{n}{2} \rfloor-1}+w_{n-\lfloor \frac{n}{2} \rfloor+1}\ . \end{cases}$$

(2) $S_1 \triangleq \sum_{m=1}^{\lfloor \frac{n}{2} \rfloor} p_m w_{n-m+1}$, *where* $p_m \triangleq \sum_{j=m}^{\lfloor \frac{n}{2} \rfloor} w_{2j-m}$ *satisfies the recursion*

$$p_m=p_{m+2}+w_m+w_{2\lfloor \frac{n}{2} \rfloor-m}, \text{ with } \begin{cases} p_{\lfloor \frac{n}{2} \rfloor}=w_{\lfloor \frac{n}{2} \rfloor}, \\ p_{\lfloor \frac{n}{2} \rfloor-1}=w_{\lfloor \frac{n}{2} \rfloor-1}+w_{\lfloor \frac{n}{2} \rfloor+1}\ . \end{cases}$$

Furthermore, p_m and q_m satisfy

$$q_m=\begin{cases} p_m & (n \text{ is even}), \\ p_{m-1}-w_{m-1} & (n \text{ is odd}). \end{cases}$$

Proof. We start by considering $M_1M_1^T$. From simple matrix multiplication, we have

$$(M_1M_1^T)_{j,n-j+1}=\sum_{k=1}^{j} w_{j-1+k}w_{n-j+k}\ , \text{ for } 1 \le j \le \left\lfloor \frac{n}{2} \right\rfloor\ .$$

When n is odd, we obtain for $j=\lfloor \frac{n}{2} \rfloor+1$:

$$(M_1M_1^T)_{\lfloor \frac{n}{2} \rfloor+1,\lfloor \frac{n}{2} \rfloor+1}=\sum_{k=1}^{n-\lfloor \frac{n}{2} \rfloor} w_{\lfloor \frac{n}{2} \rfloor+k}^2=\sum_{k=1}^{\lfloor \frac{n}{2} \rfloor+1} w_{\lfloor \frac{n}{2} \rfloor+k}^2=w_n^2+\sum_{k=1}^{\lfloor \frac{n}{2} \rfloor} w_{n-k}^2\ .$$

We now define

$$S_1=\sum_{j=1}^{\lfloor \frac{n}{2} \rfloor}\sum_{k=1}^{j} w_{j+k-1}w_{n-j+k}\ .$$

Since $\operatorname{antitr}(M_1M_1^T) = \sum_{j=1}^{n}(M_1M_1^T)_{j,n-j+1}$, we have that $\operatorname{antitr}(M_1M_1^T) = 2S_1$ when n is even and that $\operatorname{antitr}(M_1M_1^T) = 2S_1 + (M_1M_1^T)_{\lfloor\frac{n}{2}\rfloor+1,\lfloor\frac{n}{2}\rfloor+1}$ when n is odd. Furthermore, S_1 is the sum of the elements of a triangular array, summed row by row with row index j. Instead, we compute this sum by summing the diagonals. Setting $m = j - k + 1$, we obtain

$$\begin{aligned} S_1 &= \sum_{m=1}^{\lfloor\frac{n}{2}\rfloor}\sum_{j=m}^{\lfloor\frac{n}{2}\rfloor} w_{2j-m}w_{n-m+1} \\ &= \sum_{m=1}^{\lfloor\frac{n}{2}\rfloor}\left(w_{n-m+1}\sum_{j=m}^{\lfloor\frac{n}{2}\rfloor} w_{2j-m}\right) \\ &= \sum_{m=1}^{\lfloor\frac{n}{2}\rfloor} p_m w_{n-m+1}\ , \end{aligned}$$

where $p_m = \sum_{j=m}^{\lfloor\frac{n}{2}\rfloor} w_{2j-m}$. The quantities p_m can be computed recursively: since

$$\begin{aligned} p_{m+2} = \sum_{j=m+2}^{\lfloor\frac{n}{2}\rfloor} w_{2j-m-2} = \sum_{j=m+2}^{\lfloor\frac{n}{2}\rfloor} w_{2(j-1)-m} &= \sum_{i=m+1}^{\lfloor\frac{n}{2}\rfloor-1} w_{2i-m} \\ &= \left(\sum_{i=m}^{\lfloor\frac{n}{2}\rfloor} w_{2i-m}\right) - w_m - w_{2\lfloor\frac{n}{2}\rfloor-m}\ , \end{aligned}$$

we have

$$p_m = p_{m+2} + w_m + w_{2\lfloor\frac{n}{2}\rfloor-m}\ .$$

The initial values are easily computed from the definition of p_m. They are

$$p_{\lfloor\frac{n}{2}\rfloor} = w_{\lfloor\frac{n}{2}\rfloor} \quad \text{and} \quad p_{\lfloor\frac{n}{2}\rfloor-1} = w_{\lfloor\frac{n}{2}\rfloor-1} + w_{\lfloor\frac{n}{2}\rfloor+1}\ .$$

Because of the recursion for p_m, the computation of S_1 can be carried out in $\mathcal{O}(n)$ flops.

The corresponding result for $M_0^TM_0$ with

$$S_0 = \sum_{j=1}^{\lfloor\frac{n}{2}\rfloor}\sum_{k=1}^{j} w_{n-j-k+1}w_{j-k}$$

follows from the observation that M_0 is obtained from M_1^T by replacing w_j with w_{n-j}. The computation of S_0 can therefore also be carried out in $\mathcal{O}(n)$ flops.

To establish the relationship between q_m and p_m, we reverse the summation order in the computation of q_m, obtaining

$$q_m = \sum_{j=m}^{\lfloor\frac{n}{2}\rfloor} w_{n-2j+m} = \sum_{j=m}^{\lfloor\frac{n}{2}\rfloor} w_{n-2(\lfloor\frac{n}{2}\rfloor+m-j)+m} = \sum_{j=m}^{\lfloor\frac{n}{2}\rfloor} w_{n-2\lfloor\frac{n}{2}\rfloor+2j-m}\ .$$

When n is even, it follows immediately that

$$q_m = \sum_{j=m}^{\lfloor\frac{n}{2}\rfloor} w_{2j-m} = p_m.$$

When n is odd, we have

$$\begin{aligned} q_m &= \sum_{j=m}^{\lfloor \frac{n}{2} \rfloor} w_{2j-m+1} \\ &= \sum_{j=m-1}^{\lfloor \frac{n}{2} \rfloor} w_{2j-(m-1)} - w_{2(m-1)-m+1} \\ &= p_{m-1} - w_{m-1} \ . \end{aligned}$$

This completes the proof. □

Theorem 4.5. *The Newton steps for the even and odd characteristic polynomials $p_n^e(\lambda)$ and $p_n^o(\lambda)$ at $\lambda = \bar{\lambda} < \lambda_{\min}(T_n)$ of an $n \times n$ symmetric positive definite Toeplitz matrix T_n, are given by the following expressions.*

For even n:

$$N^e(\bar{\lambda}) = \frac{\chi(\bar{\lambda})}{\frac{n}{2} + p_1 + \sum_{j=1}^{\frac{n}{2}-1} \left(\left(\frac{n}{2} - j \right) (w_j + w_{n-j}) + p_{j+1} \right) (w_{n-j} - w_j)} \ ,$$

$$N^o(\bar{\lambda}) = \frac{\chi(\bar{\lambda})}{\frac{n}{2} - p_1 + \sum_{j=1}^{\frac{n}{2}-1} \left(\left(\frac{n}{2} - j \right) (w_j + w_{n-j}) - p_{j+1} \right) (w_{n-j} - w_j)} \ .$$

For odd n:

$$N^e(\bar{\lambda}) = \frac{\chi(\bar{\lambda})}{\frac{n+1}{2} + \sum_{j=1}^{\lfloor \frac{n}{2} \rfloor} \left(\left(\frac{n+1}{2} - j \right) (w_{n-j}^2 - w_j^2) + p_j (w_{n-j+1} - w_j) + w_j^2 \right)} \ ,$$

$$N^o(\bar{\lambda}) = \frac{\chi(\bar{\lambda})}{\frac{n-1}{2} + \sum_{j=1}^{\lfloor \frac{n}{2} \rfloor} \left(\left(\frac{n-1}{2} - j \right) (w_{n-j}^2 - w_j^2) - p_j (w_{n-j+1} - w_j) - w_j^2 \right)} \ .$$

In these expressions, χ is as in Definition 3.2, the first row of T_n is given by (ρ_0, t^T) and $w = (w_1, \ldots, w_{n-1}) = -J(T_{n-1} - \bar{\lambda} I)^{-1} t$. The quantities p_j are as defined in Lemma 4.4. For compactness of writing, we have set $w_0 = 0$ and $w_n = 1$.

Proof. We start with the even Newton step in the case where n is even. Setting

$$T_n - \bar{\lambda} I = \begin{pmatrix} A - \bar{\lambda} I & B \\ JBJ & A - \bar{\lambda} I \end{pmatrix}$$

and

$$(T_n - \bar{\lambda} I)^{-1} = \begin{pmatrix} H_1 & H_2 \\ JH_2J & JH_1J \end{pmatrix},$$

we have, from Lemma 4.2, that

$$N^e(\bar{\lambda}) = \frac{1}{\operatorname{tr} \left(A - \bar{\lambda} I + BJ \right)^{-1}}.$$

From Lemma 4.3 we obtain that $\left(A-\bar{\lambda}I+BJ\right)^{-1}=H_1+H_2J$, and, therefore, that

$$
\begin{aligned}
(N^e(\bar{\lambda}))^{-1} &= \operatorname{tr}\left(A-\bar{\lambda}I+BJ\right)^{-1} \\
&= \operatorname{tr}\left(H_1+H_2J\right) \\
&= \operatorname{tr}\left(H_1\right)+\operatorname{antitr}\left(H_2\right) \\
&= \frac{1}{2}\operatorname{tr}\left(T_n-\bar{\lambda}I\right)^{-1}+\frac{1}{2}\operatorname{antitr}\left(T_n-\bar{\lambda}I\right)^{-1}.
\end{aligned} \tag{10}
$$

We have from the Gohberg–Semencul formula (2) that $\chi(\bar{\lambda})(T_n-\bar{\lambda}I)^{-1}=M_1M_1^T-M_0^TM_0$, where M_0 and M_1 are given by (3). From (10) and with Lemma 4.4 we therefore obtain

$$
\begin{aligned}
\chi(\bar{\lambda})(N^e(\bar{\lambda}))^{-1} &= \frac{1}{2}\operatorname{tr}(M_1M_1^T-M_0^TM_0)+\frac{1}{2}\operatorname{antitr}(M_1M_1^T-M_0^TM_0) \\
&= \frac{n}{2}+\sum_{j=1}^{\frac{n}{2}-1}\left(\frac{n}{2}-j\right)(w_{n-j}^2-w_j^2)+\sum_{j=1}^{\frac{n}{2}}(p_jw_{n-j+1}-q_jw_{j-1}) \\
&= \frac{n}{2}+\sum_{j=1}^{\frac{n}{2}-1}\left(\frac{n}{2}-j\right)(w_{n-j}^2-w_j^2)+\sum_{j=1}^{\frac{n}{2}}p_j(w_{n-j+1}-w_{j-1}) \\
&= \frac{n}{2}+\sum_{j=1}^{\frac{n}{2}-1}\left(\frac{n}{2}-j\right)(w_{n-j}^2-w_j^2)+\sum_{j=0}^{\frac{n}{2}-1}p_{j+1}(w_{n-j}-w_j) \\
&= \frac{n}{2}+p_1+\sum_{j=1}^{\frac{n}{2}-1}\left(\left(\frac{n}{2}-j\right)(w_{n-j}^2-w_j^2)+p_{j+1}(w_{n-j}-w_j)\right) \\
&= \frac{n}{2}+p_1+\sum_{j=1}^{\frac{n}{2}-1}\left(\left(\frac{n}{2}-j\right)(w_{n-j}+w_j)+p_{j+1}\right)(w_{n-j}-w_j)\,,
\end{aligned}
$$

which establishes the result for the even Newton step.

For the odd Newton step, we have, from Lemma 4.2, that

$$
N^o(\bar{\lambda})=\frac{1}{\operatorname{tr}\left(A-\bar{\lambda}I-BJ\right)^{-1}}.
$$

From Lemma 4.3 we obtain that

$$
\left(A-\bar{\lambda}I+BJ\right)^{-1}=H_1-H_2J.
$$

Analogously to the even case, we then obtain

$$
\chi(\bar{\lambda})(N^o(\bar{\lambda}))^{-1}=\frac{1}{2}\operatorname{tr}(M_1M_1^T-M_0^TM_0)-\frac{1}{2}\operatorname{antitr}(M_1M_1^T-M_0^TM_0)\,,
$$

and the rest of the proof proceeds exactly as in the even case.

When n is odd, we have, with (6) and with Lemma 4.4, that

$$\begin{aligned}
&\chi(\bar\lambda)(N^e)^{-1}(\bar\lambda)\\
&\quad= \frac{1}{2}\mathrm{tr}(M_1M_1^T - M_0^TM_0) + \frac{1}{2}\,\mathrm{antitr}(M_1M_1^T - M_0^TM_0)\\
&\quad= \frac{1}{2}\mathrm{tr}(M_1M_1^T - M_0^TM_0) + \frac{1}{2} + \frac{1}{2}\sum_{j=1}^{\lfloor\frac{n}{2}\rfloor}(w_{n-j}^2 - w_j^2) + S_1 - S_0\\
&\quad= \frac{n}{2} + \sum_{j=1}^{\lfloor\frac{n}{2}\rfloor}\left(\frac{n}{2} - j\right)(w_{n-j}^2 - w_j^2)\\
&\qquad\quad + \frac{1}{2} + \frac{1}{2}\sum_{j=1}^{\lfloor\frac{n}{2}\rfloor}(w_{n-j}^2 - w_j^2) + \sum_{j=1}^{\lfloor\frac{n}{2}\rfloor}(p_jw_{n-j+1} - q_jw_{j-1})\\
&\quad= \frac{n+1}{2} + \sum_{j=1}^{\lfloor\frac{n}{2}\rfloor}\left(\frac{n+1}{2} - j\right)(w_{n-j}^2 - w_j^2)\\
&\qquad\quad + \sum_{j=1}^{\lfloor\frac{n}{2}\rfloor}(p_jw_{n-j+1} - (p_{j-1} - w_{j-1})w_{j-1})\\
&\quad= \frac{n+1}{2} + \sum_{j=1}^{\lfloor\frac{n}{2}\rfloor}\left(\frac{n+1}{2} - j\right)(w_{n-j}^2 - w_j^2)\\
&\qquad\quad + \sum_{j=1}^{\lfloor\frac{n}{2}\rfloor}p_j(w_{n-j+1} - w_j) + w_{\lfloor\frac{n}{2}\rfloor}^2 + \sum_{j=1}^{\lfloor\frac{n}{2}\rfloor-1}w_j^2\\
&\quad= \frac{n+1}{2} + \sum_{j=1}^{\lfloor\frac{n}{2}\rfloor}\left(\left(\frac{n+1}{2} - j\right)(w_{n-j}^2 - w_j^2) + p_j(w_{n-j+1} - w_j) + w_j^2\right),
\end{aligned}$$

which is what we needed to prove. As in the case for even n, the odd Newton step follows analogously from

$$\begin{aligned}
&\chi(\bar\lambda)(N^o)^{-1}(\bar\lambda)\\
&\qquad= \frac{1}{2}\mathrm{tr}(M_1M_1^T - M_0^TM_0) - \frac{1}{2}\,\mathrm{antitr}(M_1M_1^T - M_0^TM_0)\\
&\qquad= \frac{1}{2}\mathrm{tr}(M_1M_1^T - M_0^TM_0) - \frac{1}{2} - \frac{1}{2}\sum_{j=1}^{\lfloor\frac{n}{2}\rfloor}(w_{n-j}^2 - w_j^2) - S_1 + S_0\,. \qquad \square
\end{aligned}$$

As before, this means that once the $(n-1)$-dimensional Yule–Walker system is solved (this needs to be done at every iteration), i.e., once $(T_{n-1} - \bar\lambda I)^{-1}t$ is computed, only $\mathcal{O}(n)$ flops are necessary to compute both the even and odd Newton steps. The total number of operations depends on the method used for the Yule–Walker system. If the split Durbin algorithm is used ([6]), then a total of $\frac{3}{2}n^2 + \mathcal{O}(n)$ flops needs to be carried out.

This approach contrasts with the one in [14], where the even and odd Newton steps are determined by the following recursions:

$$N_k^e = \frac{\rho_0 + \rho_{k-1} - \bar{\lambda} + \tilde{t}^T u^{(k-2)}}{\left(N_{k-2}^e\right)^{-1}\left(\rho_0 + \rho_{k-1} - \bar{\lambda} + \tilde{t}^T u^{(k-2)}\right) + \left(1 + \frac{1}{2}\|u^{(k-2)}\|^2\right)}, \tag{11}$$

$$N_k^o = \frac{\rho_0 - \rho_{k-1} - \bar{\lambda} + \tilde{t}^T v^{(k-2)}}{\left(N_{k-2}^o\right)^{-1}\left(\rho_0 - \rho_{k-1} - \bar{\lambda} + \tilde{t}^T v^{(k-2)}\right) + \left(1 + \frac{1}{2}\|v^{(k-2)}\|^2\right)}, \tag{12}$$

where $N_k^e(\bar{\lambda})$ and $N_k^o(\bar{\lambda})$ are the even and odd Newton steps, respectively, for the characteristic polynomial of the k-th principal submatrix of T_n at $\lambda = \bar{\lambda}$, and u^k and v^k are the k-th even and odd solutions of the Yule–Walker equations, respectively. The quantities $\tilde{t}^T u^{(k-2)}$ and $\tilde{t}^T v^{(k-2)}$ are computed in the course of Durbin's or the split Durbin algorithm. The algorithm in [14] uses bounds to predict the parity of the smallest eigenvalue, resulting in two phases: Phase I during which the parity of the smallest eigenvalue is unknown and both the even and odd Newton steps need to be computed, and phase II during which only the even or only the odd Newton step needs to be computed, depending on the predicted parity. If Durbin's algorithm is used in phase I, as in [14], this recursion requires $\frac{11}{4}n^2 + \mathcal{O}(n)$ flops, namely $2n^2 + \mathcal{O}(n)$ flops for Durbin's algorithm and another $\frac{3}{4}n^2 + \mathcal{O}(n)$ flops for the computation of u^k and v^k from y^k and for the norms in the recursion formula. As in the case of the algorithm in [11], using the split Durbin method would increase the computational cost in phase I. In phase II the split Durbin algorithm is used, resulting in $\frac{7}{4}n^2 + \mathcal{O}(n)$ flops, namely $\frac{3}{2}n^2 + \mathcal{O}(n)$ flops for the split Durbin algorithm and $\frac{1}{4}n^2 + \mathcal{O}(n)$ flops for the extra scalar products.

Clearly, our approach here significantly reduces the number of arithmetic operations when compared to the method in [14], especially in phase I of that method. Our results can also be used for other methods, such as the double Newton method or Laguerre-type methods. In addition, there is, strictly speaking, no need anymore to predict the parity of the smallest eigenvalue, as the computation of the Newton steps requires no more than $\mathcal{O}(n)$ flops, so that both the even and odd steps can be computed without significantly affecting the flop count. However, the parity predictor phase does seem to enhance numerical stability. Finally, we remark that, since there is no need in our approach to compute the solutions of the intermediate Yule–Walker equations as in (11) and (12), any method which yields $(T_{n-1} - \bar{\lambda}I)^{-1}t$ can be used. As we mentioned before, this is very important because of the availability of superfast methods for this problem.

5. Numerical results

In this section, we compare the number of flops for the various methods, using the numerical results from [14]. These provide us with the average number of iterations (for a sample size of 500) needed to compute the smallest eigenvalue of a matrix of the form $T = \mu \sum_{k=1}^{n} \xi_k T_{2\pi\theta_k}$, where n is the dimension of T, and θ_k, ξ_k are uniformly distributed random numbers in $(0, 1)$. The parameter μ is chosen such that $T_{kk} = 1$ for $k = 1, ..., n$, and $(T_\theta)_{ij} = \cos(\theta(i - j))$. These matrices are positive, semi-definite and Toeplitz, and even though they could theoretically be singular, we did not encounter such cases, nor did we encounter cases where the smallest eigenvalue of T_n was an eigenvalue of T_{n-1} or where the smallest even or odd eigenvalue of T_n was an eigenvalue of T_{n-2}.

TABLE 1. Comparison of the algorithms using 500 random matrices for each dimension.

Dimension	**Mastronardi–Boley**			**Even-Odd**		
	iterations	flops (recursion)	flops (trace)	iterations	flops (recursion)	flops (trace)
256	12.83	2.5×10^6	1.3×10^6	7.56 (2.96 + 4.60)	1.1×10^6	0.7×10^6
512	14.77	1.2×10^7	0.6×10^7	8.56 (3.29 + 5.27)	0.48×10^7	0.3×10^7
1024	14.06	4.4×10^7	2.2×10^7	7.51 (3.33 + 4.17)	1.7×10^7	1.2×10^7

In Table 1 we present the theoretical total number of flops carried out to compute the smallest eigenvalue, based on the number of iterations, the dimension of the matrix, and the complexity of the method used to compute the Newton step. In calculating the number of flops, we neglected the $\mathcal{O}(n)$ term, and we note that the resulting numbers are in complete agreement with the flop count in [14]. The stopping criterion required the absolute accuracy of the computed eigenvalues to be $\epsilon = 10^{-14}$, or for the value of the characteristic polynomial to become negative. This resulted, on average, in a relative accuracy of better than 10^{-6}.

The three columns under "Mastronardi–Boley" refer to the method in [11]. They contain, respectively, the number of iterations (i.e., the number of Newton steps), the total number of flops when the Newton step is computed using the recursion (7), and the total number of flops when the Newton step is computed using our method. Similarly, the three columns under "Even-Odd" refer to the method in [14]. The first column contains the number of iterations, i.e., the number of Newton steps, with, in parentheses, the number of iterations in phase I (the parity predictor phase) and phase II. An entry such as $5\,(2+3)$ therefore means five iterations, two in phase I and three in phase II. The second column contains the total number of flops when the Newton step is computed using the recursions (11) and (12) and the third column gives the total number of flops when the appropriate (even or odd) Newton step is computed using our method.

We stress that these numerical comparisons serve only as an illustration for our new way of computing a key quantity in the methods from [11] and [14]. Our results here only change the amount of work per iteration, not the *number* of iterations, which makes it easy to use the comparisons already reported in [11] and [14] to conclude that, at least for the type of matrices that were considered here, the results are competitive with those obtained for methods based on secular equations.

There are also methods, such as the one decribed in [10], that use polynomial interpolation rather than Newton's method for the computation of the smallest root of the characteristic polynomial. Judging from the numerical results there, which were for the same test matrices as ours, such methods appear to be better than the method in [11], although they fall short of the method we have presented here. On the other hand, our results can improve the method in [10], provided that the interpolation techniques that are used there do not break down numerically, which may happen when the matrices have an unfavorable eigenvalue distribution.

However, a large-scale program of numerical experiments is needed to thoroughly compare all existing methods with their modifications across different eigenvalue distributions. This is beyond the scope of the present work.

Summarizing, we can say that the method in [11], which was simple but relatively slow, has been greatly enhanced in two stages. In the first stage, the even-odd split was used in [14] to reduce the number of Newton steps, whereas in the second stage, the number of arithmetic operations to compute the Newton step was drastically reduced in the work presented here. This has resulted in a method which is three and a half to four times faster than the original one in [11]. Further improvements can be achieved by considering double Newton steps or methods other than Newton's method, such as Laguerre-type methods. All these methods benefit from our results here.

References

1. Ammar, G.S. and Gragg, W.B. (1987): The generalized Schur algorithm for the superfast solution of Toeplitz systems. In Rational Approximations and its Applications in Mathematics and Physics, J. Gilewicz, M. Pindor and W. Siemaszko, eds., Lecture Notes in Mathematics, **1237**, Berlin, pp. 315–330. MR0886705 (88c:65032)
2. Ammar, G.S. and Gragg, W.B. (1989): Numerical experience with a superfast Toeplitz solver. Linear Algebra Appl., **121**, pp. 185–206. MR1011737 (90g:65030)
3. Cantoni, A. and Butler, F. (1976): Eigenvalues and eigenvectors of symmetric centrosymmetric matrices. Linear Algebra Appl., **13**, pp. 275-288. MR0396614 (53:476)
4. Cybenko, G. and Van Loan, C. (1986): Computing the minimum eigenvalue of a symmetric positive definite Toeplitz matrix. SIAM J. Sci. Stat. Comput., Vol. 7, No. 1, pp. 123–131. MR0819462 (87b:65042)
5. Delsarte, P. and Genin, Y. (1983): Spectral properties of finite Toeplitz matrices. In Mathematical Theory of Networks and Systems, Proc. MTNS-83 International Symposium, Beer-Sheva, Israel, pp. 194–213. MR0792106 (86k:15003)
6. Delsarte, P. and Genin, Y. (1986): The split Levinson algorithm. IEEE Trans. Acoust. Speech, Signal Processing, **ASSP-34**, pp. 470–478. MR0844658 (87f:94007)
7. Durbin, J. (1960): The fitting of time series model. Rev. Inst. Int. Stat., **28**, pp. 233–243.
8. Gohberg, I.C. and Semencul, A.A. (1972): The inversion of finite Toeplitz matrices and their continual analogues. (Russian) Mat. Issled. 7 (1972), No. 2(24), 201–223. MR0353038 (50:5524)
9. Golub, G. and Van Loan, C. (1996): Matrix Computations. The Johns Hopkins University Press, Baltimore and London. MR1417720 (97g:65006)
10. Mackens, W. and Voss, H. (2000): Computing the minimum eigenvalue of a symmetric positive definite Toeplitz matrix by Newton-type methods. SIAM J. Sci. Comput., Vol. 21, no. 4, 1650–1656. MR1756049 (2001g:65043)
11. Mastronardi, N. and Boley, D. (1999): Computing the smallest eigenpair of a symmetric positive definite Toeplitz matrix. SIAM J. Sci. Comput., Vol. 20, No. 5, pp. 1921–1927. MR1694690 (2000b:65066)
12. Melman, A. (2000): Extreme eigenvalues of real symmetric Toeplitz matrices. Math. Comp., 70, No. 234, pp. 649-669. MR1813143 (2002b:65059)
13. Melman, A. (2001): A two-step even-odd split Levinson algorithm for Toeplitz systems. Linear Algebra Appl., **338**, pp. 219–237. MR1861123 (2002h:65034)
14. Melman, A. (2004): Computation of the smallest even and odd eigenvalues of a symmetric positive definite Toeplitz matrix. SIAM J. Matrix Anal. Appl., Vol. 25, No. 4, pp. 947–963. MR2081125

Department of Applied Mathematics, School of Engineering, Santa Clara University, Santa Clara, California 95053

E-mail address: amelman@scu.edu