Newton type methods for computing the smallest eigenvalue of a symmetric Toeplitz matrix

Wolfgang Mackens and Heinrich Voss

Technische Universität Hamburg–Harburg, Arbeitsbereich Mathematik,
Kasernenstraße 12, D—21073 Hamburg
e-mail: mackens @ tu-harburg.de, voss @ tu-harburg.de

Abstract

Several methods for computing the smallest eigenvalue of a symmetric positive definite Toeplitz matrix are presented. They converge from the left to the minimum eigenvalue, and they rely on Newton's method and interpolation of the characteristic polynomial with no need for introductory bisection steps. The methods are conceptually much simpler than the ones introduced by the same authors based on rational interpolation of the secular equation.

Keywords Toeplitz matrix, eigenvalue problem

1 Introduction

In this report we present several methods for computing the smallest eigenvalue λ_1 of a symmetric and positive definite Toeplitz matrix T which are based on Newton's method for the characteristic polynomial and on interpolation. This problem is of considerable interest in signal processing. Given the covariance sequence of the observed data, Pisarenko [8] suggested a method which determines the sinusoidal frequencies from the eigenvector of the covariance matrix associated with the minimum eigenvalue of T.

Several approaches have been reported in the literature for computing the minimal eigenvalue of a real symmetric Toeplitz matrix. Cybenko and Van Loan [1] introduced an algorithm which is a combination of a bisection method and Newton's method for the secular equation, and which was generalized to the computation of the complete spectrum by Trench [10] and by Noor and Morgera [6]. In recent papers [3], [12] the authors presented a generalization of this approach where the Newton method is replaced by a root finding method based on rational Hermitian interpolation of the secular equation, and a further generalization in [4] where the

method in [3] is shown to be equivalent to a projection method. Finally the second author in [11] took advantage of symmetry properties of the eigenvectors to improve the methods from [3] and [4].

Mastronardi and Boley [5] considered the Newton method for the characteristic equation. The advantage over the methods above is that it does not need a bisection phase to determine a suitable initial approximation for Newton's method but with the initial guess $\mu_0 = 0$ it converges from the left to the smallest eigenvalue. The disadvantageous, however, is that the initial steps of Newtons's method usually yield only little progress. Moreover the cost of the modified Durban algorithm involved in every step of Mastonardi and Boley's method is about $3n^2$ flops whereas the cost of one step of the methods mentioned in the last paragraph is only about $2n^2$ flops.

In this report we discuss several improvements of the method of Mastronardi and Boley. It is organized as follows. In Section 2 we briefly sketch a modified version of Mastronardi and Boley's approach as well as the secant method for the characteristic polynomial. Both methods are accelerated substantially by a double step strategy based on a theorem due to Stoer. In Section 3 we take advantage of previous Newton or secant steps, and we use Hermitian interpolation to enhance these methods. The paper closes with concluding remarks and an Appendix containing listings of all MATLAB functions used in the methods of this paper.

2 Newton type methods for the characteristic polynomial

Let $T_n \in \mathbb{R}^{(n,n)}$ be a symmetric positive definite Toeplitz matrix. We assume that the diagonal is normalized to 1, and we denote the principal submatrices of T_n by

$$T_j = \begin{pmatrix} 1 & t_1 & t_2 & \dots & t_{j-2} & t_{j-1} \\ t_1 & 1 & t_1 & \dots & t_{j-3} & t_{j-2} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ t_{j-1} & t_{j-2} & t_{j-3} & \dots & t_1 & 1 \end{pmatrix} \in \mathbb{R}^{(j,j)}.$$

We also use

$$t^{(j)} := (t_1\,,\, t_2\,,\, \dots\,,\, t_{j-1}\,,\, t_j)^T$$

to denote the entries of T_{j+1} omitting the diagonal entry 1.

Let

$$\lambda_1(T_j) \le \lambda_2(T_j) \le \dots \le \lambda_j(T_j)$$

be the eigenvalues of T_j ordered by magnitude. Then the following interlacing property holds

$$\lambda_i(T_{j+1}) \le \lambda_i(T_j) \le \lambda_{i+1}(T_{j+1}), \quad 1 \le i \le j \le n.$$

If λ is not an eigenvalue of T_j then

$$T_{j+1} - \lambda I_{j+1} = \begin{pmatrix} 1-\lambda & , & (t^{(j)})^T \\ t^{(j)} & , & T_j - \lambda I_j \end{pmatrix}$$

$$= \begin{pmatrix} 1-\lambda-(t^{(j)})^T(T_j-\lambda I_j)^{-1}t^{(j)} & , & (t^{(j)})^T(T_j-\lambda I_j)^{-1} \\ 0 & , & I_j \end{pmatrix} \begin{pmatrix} 1 & , & 0^T \\ t^{(j)} & , & T_j-\lambda I_j \end{pmatrix}.$$

Hence, if λ is not in the spectrum of any of the principal submatrices T_j, $j = 1, \ldots, n-1$, then the characteristic polynomials

$$\chi_j(\lambda) := \det(T_j - \lambda I_j)$$

of T_j satisfy the recurrence relation

$$\chi_{j+1}(\lambda) = \chi_j(\lambda)(1-\lambda-(t^{(j)})^T(T_j-\lambda I_j)^{-1}t^{(j)}), \quad 1 \le j \le n-1, \tag{1}$$

and for the derivatives it holds that

$$\chi'_{j+1}(\lambda) = \chi'_j(\lambda)(1-\lambda-(t^{(j)})^T(T_j-\lambda I_j)^{-1}t^{(j)}) - \chi_j(\lambda)(1+\|(T_j-\lambda I_j)^{-1}t^{(j)}\|_2^2). \tag{2}$$

Durbin's algorithm for the Yule–Walker system (cf.[2], p. 194 ff)

$$(T_{n-1}-\lambda I_{n-1})y^{(n-1)} = -t^{(n-1)}$$

yields a lower triangular matrix

$$L_n = \begin{pmatrix} 1 & 0 & \ldots & 0 \\ \ell_{21} & 1 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ \ell_{n1} & \ell_{n2} & \ldots & 1 \end{pmatrix}$$

such that

$$L_n(T_n-\lambda I_n)L_n^T = \mathrm{diag}\{1, \beta_1, \ldots, \beta_{n-1}\}. \tag{3}$$

Hence, the characteristic polynomial $\chi_n(\lambda)$ can be evaluated using Durbin's algorithm. The cost of one evaluation is $2n^2$ flops.

2.1 Newton's method

For the characteristic polynomial

$$\chi_i(\lambda) := \prod_{j=1}^{i}(\lambda_j(T_i) - \lambda)$$

of T_i it obviously holds that

$$\chi_i^{(2k)}(\lambda) > 0, \quad \text{for every } \lambda < \lambda_1(T_i) \text{ and } 2k \le i \tag{4}$$

and

$$\chi_i^{(2k+1)}(\lambda) < 0 \quad \text{for every } \lambda < \lambda_1(T_i) \text{ and } 2k+1 \le i. \tag{5}$$

In particular χ_n is convex and monotonely decreasing for $\lambda < \lambda_1(T_n)$, and therefore Newton's method converges from the left to the smallest eigenvalue of T_n for every initial value $\mu_0 < \lambda_1(T_n)$.

Along with the computation of the matrix L_n Durbin's algorithm yields the solutions $y^{(j)}$ of the the linear systems

$$(T_j - \lambda I_j)y^{(j)}(\lambda) = -t^{(j)}, \quad j = 1, \ldots, n-1.$$

Hence, evaluating $\|y^{(j)}(\lambda)\|_2^2$, $j = 1, \ldots, n-1$, (i.e. with n^2 additional flops) Durbin's algorithm can be modified such that $\chi_n'(\lambda)$ is obtained as well, and a Newton step for the characteristic polynomial can be performed requiring $3n^2$ flops.

Mastronardi and Boley [5] observed that in the modified Durbin algorithm $\chi_n(\lambda)$ and $\chi_n'(\lambda)$ need not be computed explicitely but that a recurrence relation holds for the rational function $\phi_n(\lambda) := \chi_n(\lambda)/\chi_n'(\lambda)$. From (1) and (2) it follows that

$$\frac{1}{\phi_{j+1}} = \frac{1}{\phi_j} - \frac{1 + \|y^{(j)}(\lambda)\|_2^2}{1 - \lambda + (t^{(j)})^T y^{(j)}(\lambda)}. \tag{6}$$

Mastronardi and Boley claim that it is preferable to use this recurrence relation to avoid cancelation. However, in our numerical experiments we did not observe unstable behaviour for the separate calculation of $\chi_n(\lambda)$ and $\chi_n'(\lambda)$.

An error bound for the Newton iterates can be obtained in the following way. Let

$$w := \begin{pmatrix} 1 \\ y^{(n-1)}(\mu) \end{pmatrix}, \quad y^{(n-1)}(\mu) := -(T_{n-1} - \mu I_{n-1})^{-1} t^{(n-1)}.$$

Then the Rayleigh quotient is given by

$$\begin{aligned} R(w) &= \frac{w^T T_n w}{\|w\|_2^2} = \frac{1 + (t^{(n-1)})^T y^{(n-1)}(\mu) + \mu \|y^{(n-1)}(\mu)\|_2^2}{1 + \|y^{(n-1)}(\mu)\|_2^2} \\ &= \mu + \frac{1 - \mu + (t^{(n-1)})^T y^{(n-1)}(\mu)}{1 + \|y^{(n-1)}(\mu)\|_2^2}. \end{aligned}$$

Notice that

$$\|y^{(n-1)}(\mu)\| \quad \text{and} \quad \beta(\mu) := 1 - \mu + (t^{(n-1)})^T y^{(n-1)}(\mu)$$

were already determined in the generalized Durbin algorithm. Hence the error bounds

$$\mu_{k+1} = \mu_k - \phi_n(\mu_k) \le \lambda_1(T_n) \le \mu_k + \frac{\beta(\mu_k)}{1 + \|y^{(n-1)}(\mu_k)\|_2^2} \tag{7}$$

are free. Mastronardi and Boley proved the error estimation

$$\lambda_1(T_n) - \mu_{k+1} \le \beta(\mu_k)$$

and adopted it in their stopping criterion. Obviously this is less accurate than

$$\lambda_1(T_n) - \mu_{k+1} \le \frac{\beta(\mu_k)}{1 + \|y^{(n-1)}(\mu_k)\|_2^2} + \phi_n$$

which is obtained from the inclusion (7).

The following function is a MATLAB realization of Newton's method for the characteristic equation:

```
function lambda=nm(t,n,tol);
% Newton's method for computing the minimum eigenvalue of a symmetric
% positive definite Toeplitz matrix which is normalized by t0=1.
% Iteration is terminated if the relative error is less than tol
% Uses function N_STEP to determine a Newton increment.
%
lambda=0;
err=tol+1;
while err > tol
  [phi,normy,beta]=n_step(lambda,t,n);
  lambda=lambda-phi;
  err=(beta/(1+normy)+phi)/lambda;
  end
```

The Newton increment is obtained from the function.

```
function [phi,normy,beta]=n_step(lambda,t,n);
% n_step determines the Newton increment for the characteristic
% polynomial using Durbin's algorithm
%
beta=1-lambda;
phi=-1/beta;
alpha = -t(1)/beta;
for i=1:n-1
  if i > 1
    alpha = -(t(i) + flipud(y(1:i-1))'*t(1:i-1))/beta;
    y(1:i-1)=y(1:i-1)+alpha*flipud(y(1:i-1));
    end
  y(i)=alpha;
  beta = beta*(1-alpha^2);
  normy=y(1:i)'*y(1:i);
  phi = phi-(1+normy)/beta;
  end
phi=1/phi;
```

The quadratic convergence of Newton's method does not mean that the convergence is fast. It may converge slowly in the beginning if the initial value μ_0 is far from the smallest root of χ_n or if the slope of χ_n is very large. There are Toeplitz matrices of the type that we consider in our numerical examples of dimension 64, that needed 28 Newton steps to obtain an approximation of the smallest eigenvalue with relative error less than 10^{-6}. There were examples of dimension 512 that even took 45 Newton steps to get an approximation of the same accuracy.

The global behaviour of Newton's method can be improved considerably by the following theorem due to Stoer:

Theorem 1 (Stoer and Bulirsch [9], p. 274)
Let $\chi(\lambda)$ be a polynomial of degree $n > 2$, all roots of which are real, $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$. Let ξ_1 be the smallest root of $\chi'(\lambda)$. Then for every $\mu < \lambda_1$ the numbers

$$\mu' := \mu - \frac{\chi(\mu)}{\chi'(\mu)},\ \nu := \mu - 2\frac{\chi(\mu)}{\chi'(\mu)},\ \nu' := \nu - \frac{\chi(\nu)}{\chi'(\nu)}$$

are well defined and satisfy

$$\nu < \xi_1, \tag{8}$$

$$\mu' \leq \nu' \leq \lambda_1. \tag{9}$$

Theorem 1 suggests the following acceleration of Newton's method by a double step strategy: Newton double steps

$$\mu_{k+1} := \mu_k - 2\frac{\chi_n(\mu_k)}{\chi_n'(\mu_k)}$$

are performed until $\mu_{k+1} > \lambda_1(T_n)$, which can be detected by the sign of the Newton increment. Then the method switches to the original Newton method.

```
function lambda=dnm(t,n,tol);
% Double step Newton method for computing the smallest eigenvalue
% of a symmetric and positive definite Toeplitz matrix which is
% normalized by t0=1. Simple Newton steps are used after an iterate
% is larger than the smallest eigenvalue.
% The iteration is terminated if the relative error is less than tol.
% Uses function N_STEP that evaluates a Newton increment
%
lambda=0;
err=tol+1;
fac = 2;
while err > tol
  [phi,normy,beta]=n_step(lambda,t,n);
  if ((fac==2) & (phi > 0))
     fac=1; end
  err=(beta/(1+normy)+phi)/(lambda-phi);
  lambda=lambda-fac*phi;
  end;
```

Example:
To test the improvement we considered Toeplitz matrices

$$T = m\sum_{k=1}^{n} \eta_k T_{2\pi\theta_k} \tag{10}$$

where m is chosen such that the diagonal of T is normalized to 1,

$$T_\theta = (t_{ij}) = (\cos(\theta(i-j)))$$

and η_k and θ_k are uniformly distributed random numbers taken from $[0,1]$ (cf. Cybenko, Van Loan [1]).

Table 1 contains the average number of flops and the average number of modified Durbin steps needed to determine the smallest eigenvalue in 100 test problems with each of the dimensions $n = 32$, 64, 128, 256 and 512 for Newton's method and the double step Newton method. The iteration was terminated if the relative error was less than 10^{-6}.

dimension	Newton's method		double step	
	flops	steps	flops	steps
32	2.496 *E*04	7.66	1.875 *E*04	5.75
64	1.091 *E*05	8.61	7.981 *E*04	6.30
128	4.238 *E*05	8.49	2.985 *E*05	5.98
256	1.892 *E*06	9.55	1.304 *E*06	6.58
512	9.048 *E*06	11.46	5.550 *E*06	7.03

Tab. 1. Newton's method and double step Newton method

Mastronardi and Boley [5] reported a slower growth of the average number of Newton steps. Notice however, that they adopted the stopping criterion that the absolute error is less than a given tolerance. Since the average of the smallest eigenvalues decreases with the dimension for the test matrices under consideration this means that the relative errors of the eigenvalue approximations increase with the dimensions in their tests.

2.2 Secant method

We already mentioned that one evaluation of the characteristic polynomial essentially costs $2n^2$ flops and that the evaluation of its derivative needs additional n^2 flops. Two steps of Newton's method therefore require the same cost as three steps of the secant method.

Since the order of convergence of the secant method is $0.5(1+\sqrt{5})$ three steps of the secant method yield the improvement

$$|\tilde{\mu}_{k+3} - \lambda_1| \leq C_1 \left(|\tilde{\mu}_k - \lambda_1|^{0.5(1+\sqrt{5})} \right)^3 = C_1 |\tilde{\mu}_k - \lambda_1|^{2+\sqrt{5}}$$

for some constant C_1 whereas for two steps of Newton's method we obtain

$$|\mu_{k+2} - \lambda_1| \leq C_2 |\mu_k - \lambda_1|^4.$$

Hence, asymptotically it should pay to replace Newton's method by the secant method.

If the initial values $\tilde{\mu}_0$ and $\tilde{\mu}_1$ of the secant method satisfy $\tilde{\mu}_0 < \tilde{\mu}_1 < \lambda_1$ then by the monotonicity and convexity of χ_n the sequence

$$\tilde{\mu}_{k+1} = \tilde{\mu}_k - \frac{\tilde{\mu}_k - \tilde{\mu}_{k-1}}{\chi(\tilde{\mu}_k) - \chi(\tilde{\mu}_{k-1})}\chi(\tilde{\mu}_k)$$

converges monotonely increasing to λ_1. Suitable initial values can be obtained by one step of Newton's method as $\tilde{\mu}_0 = 0$ and $\tilde{\mu}_1 = -\chi_n(0)/\chi_n'(0)$.

The following MATLAB function uses the functions CHARAC and CHARACD which are nearly identical to N_STEP. Instead of updating $\phi := \chi/\chi'$ they evaluate the characteristic polynomial and its derivative recurrently.

```
function lambda=sm(t,n,tol);
% Secant method for computing the minimum eigenvalue of a symmetric
% positive definite Toeplitz matrix which is normalized by t0=1.
% Iteration is terminated if the relative error is less than tol.
% Uses function CHARAC which evaluates the characteristic polynomial
% and CHARACD which additionally determines its derivative.
%
la0=0;
err=tol+1;
[f0,df,normy,beta]=characd(0,t,n);
lambda=-f0/df;
while (err > tol)
  [f,normy,beta]=charac(lambda,t,n);
  h=(lambda-la0)*f/(f-f0);
  f0=f; la0=lambda; lambda=lambda-h;
  err=(beta/(1+normy)+h)/lambda;
  end
```

Obviously for $\tilde{\mu}_{k-1} < \tilde{\mu}_k < \lambda_1(T_n)$ it holds that

$$\tilde{\mu}_{k+1} < \tilde{\mu}_k - \frac{\chi_n(\tilde{\mu}_k)}{\chi_n'(\tilde{\mu}_k)}.$$

Therefore the following corollary analogous to Theorem 1 follows immediately from Theorem 1.

Corollary 2
Let $\chi(\lambda)$ be a polynomial of degree $n > 2$, all roots of which are real, $\lambda_1 \le \lambda_2 \le \ldots \le \lambda_n$. Let ξ_1 be the smallest root of $\chi'(\lambda)$. Then for every $\mu_{k-1} < \mu_k < \lambda_1$ the numbers

$$\mu' := \mu_k - \frac{\mu_k - \mu_{k-1}}{\chi(\mu_k) - \chi(\mu_{k-1})}\chi(\mu_k),$$

$$\nu := \mu_k - 2\frac{\mu_k - \mu_{k-1}}{\chi(\mu_k) - \chi(\mu_{k-1})}\chi(\mu_k), \quad \nu' := \nu - \frac{\nu - \mu_k}{\chi(\nu) - \chi(\mu_k)}\chi(\nu)$$

are well defined and ν *satisfies*

$$\nu < \xi_1. \tag{11}$$

If $\nu < \lambda_1$ *then* $\nu < \nu' < \lambda_1$, *else* $\lambda_1 = \nu' = \nu$ *or* $\lambda_1 < \nu' < \nu$.

Corollary 2 suggests a similar acceleration of the secant method as before for Newton's method. Secant double steps are executed as long as the increments of the secant method are positive. After an approximation is reached which is larger than the smallest eigenvalue $\lambda_1(T_n)$ the method switches to the usual secant steps.

```
function lambda=dsm(t,n,tol);
% Double step secant method for computing the minimum eigenvalue
% of a symmetric positive definite Toeplitz matrix which is
% normalized by t0=1.
% Iteration is terminated if the relative error is less than tol.
% Uses function CHARAC that evaluates the characteristic polynomial
% of T and CHARACD which additionally determines its derivative
%
la0=0;
err=tol+1;
fac=2;
[f0,df,normy,beta]=characd(0,t,n);
lambda=-2*f0/df;
while (err > tol)
  [f,normy,beta]=charac(lambda,t,n);
  h=(lambda-la0)*f/(f-f0);
  if ((fac==2) & (h>0))
    fac=1; end;
  if h < 0
    if lambda < la0
      err=-h/lambda;
    else
      err=(beta/(1+normy)+h)/(lambda-h); end
  else
    if lambda > la0
      err=(lambda-h)/max(la0,1e-20)-1;
    else
      err=h/(lambda-h); end;
    end;
  f0=f;  la0=lambda; lambda=lambda-fac*h;
  end;
```

Example:
For the same test problems as in the last section the secant method needs the average numbers of flops and of iteration steps in Table 2.

dimension	secant method		double step	
	flops	steps	flops	steps
32	2.446 *E*04	10.32	2.149 *E*04	9.01
64	1.055 *E*05	11.74	8.793 *E*04	9.70
128	4.009 *E*05	11.42	3.302 *E*05	9.32
256	1.792 *E*06	12.99	1.410 *E*06	10.12
512	8.561 *E*06	15.72	6.338 *E*06	11.51

Tab. 2. Secant method and double step secant method

3 Methods based on interpolation

3.1 Hermitian interpolation of the characteristic polynomial

One disadvantage of Newton's method and of the secant method is that they only use the last or the last two iterates but they do not take advantage of previous steps.

One way of exploiting the information about the characteristic polynomial that were gathered in the preceding iterations is to use Hermitian interpolation. This suggests the following method.

1. Let $\mu_1 := 0$; $\xi_1 := \chi_n(0)$; $\xi_1' := \chi_n'(0)$.
2. for $k = 1, 2, 3, \ldots$ until convergence do
   (i) let $p_k \in \Pi_{2k-1}$ be the polynomial satisfying the Hermitian interpolation conditions
   $$p_k(\mu_j) = \xi_j;\ p_k'(\mu_j) = \xi_j',\ j = 1, \ldots, k. \tag{12}$$
   (ii) let μ_{k+1} be the smallest root of p_k in $(\mu_k, \lambda_1(T_n))$,
   $$\xi_{k+1} = \chi_n(\mu_{k+1}),\ \xi_{k+1}' := \chi_n(\mu_{k+1}).$$

In exact arithmetic the method is well defined. This follows from

Theorem 3
Let $0 = \mu_1 < \mu_2 < \ldots < \mu_k < \lambda_1(T_n)$, *and let* $p_k \in \Pi_{2k-1}$ *be the polynomial of degree* $2k-1$ *satisfying the Hermitian interpolation conditions (12). Then* p_k *has a root in* $(\mu_k, \lambda_1(T_n))$.

Proof: By the error representation of Hermitian interpolation polynomials for every $\lambda \in [\mu_k, \lambda_1(T_n)]$ there exists $\xi \in [0, \lambda_1(T_n)]$ such that

$$\chi_n(\lambda) - p_k(\lambda) = \frac{\chi_n^{(2k)}(\xi)}{(2k)!} \prod_{j=1}^{k} (\lambda - \mu_k)^2.$$

Hence, from equation (4) we obtain

$$\chi_n(\lambda) - p_k(\lambda) \geq 0 \quad \text{for every } \lambda \in [\mu_k, \lambda_1(T_n)],$$

which implies

$$p_k(\mu_k) = \chi_n(\mu_k) > 0 = \chi_n(\lambda_1(T_n)) \geq p_k(\lambda_1(T_n)). \qquad \blacksquare$$

In most of our examples the method works quite well, and it is much faster than Newton's method. However, in finite precision arithmetic it may show unstable behaviour if polynomials of high degrees are involved. For instance in the example of dimension 512 which needed 45 Newton steps the Hermitian interpolation polynomial p_{17} of degree 33 after 17 iteration steps has no root which is greater than μ_{17}. We modified the algorithm in the following way: For fixed $m \geq 1$ and $k > m$ we replaced p_k by a polynomial of degree $2m-1$ satisfying the interpolation conditions

$$p(\mu_j) = \chi_n(\mu_j),\ p'(\mu_j) = \chi_n'(\mu_j),\ j = k-m+1, \ldots, k,$$

and determined μ_{k+1} as the smallest root of p in the interval $(\mu_k, \lambda_1(T_n))$.

```
function [la,nn]=herm(t,n,m,tol);
% Hermitian interpolation of the characteristic polynomial
% for computing the smallest eigenvalue of a symmetric positive
% definite Toeplitz matrix which is normalized by t0=1.
% Maximum degree of interpolation polynomials is m.
% Iteration is terminated if the relative error is less than tol.
% Uses function CHARACD that evaluates the characteristic polynomial
% and NAHERM for evaluating the Hermitian interpolation polynomial.
%
x(1)=0;
err=tol+1;
nn=0;
while err > tol
  nn=nn+1;
  [f(nn),df(nn),normy,beta]=characd(x(nn),t,n);
  h=-f(nn)/df(nn);
  la=x(nn)+h;
  err=(beta/(1+normy)-h)/(la+h);
  if err > tol
    if nn > 1
      la0=x(nn);
      f0=f(nn);
      mm=min(nn-1,m-1);
      while abs(h)/la>1e-8
        f1=naherm(la,x(nn-mm:nn),f(nn-mm:nn),df(nn-mm:nn),mm+1);
        h=(la-la0)*f1/(f1-f0);
```

```
        f0=f1; la0=la; la=la-h;
        end;
      end;
    x(nn+1)=la;
    end
  end
```

Example:
For the set of test examples we obtained the following average numbers of flops and of evaluations of χ_n. For $n = 512$ in one example the full interpolation scheme failed. In parentheses we added the average numbers of flops and of evaluations of χ_n for the remaining 99 test examples.

dimension	m=5		m=10		m=n	
	flops	steps	flops	steps	flops	steps
32	1.892 *E*04	4.74	1.920 *E*04	4.70	1.920 *E*04	4.70
64	7.295 *E*04	5.35	7.450 *E*04	5.22	7.560 *E*04	5.20
128	2.551 *E*05	5.01	2.523 *E*05	4.94	2.523 *E*05	4.94
256	1.077 *E*06	5.40	1.063 *E*06	5.32	1.063 *E*06	5.32
512	5.003 *E*06	6.32	4.860 *E*06	6.13	failed	
(512	4.806 *E*06	6.12	4.683 *E*06	5.96	4.492 *E*06	5.70)

Tab. 3. Hermitian interpolation

3.2 Interpolation bounds

A different way of taking advantage of approximations μ_j of λ_1 from previous iteration steps is to apply the double step Newton method until an upper bound $\mu_k > \lambda_1(T_n)$ is obtained. Then the interpolation polynomial p of degree $2k-1$ satisfying the conditions

$$p(\mu_j) = \chi_n(\mu_j),\ p'(\mu_j) = \chi_n(\mu_j), \quad j = 1, \ldots, k,$$

has a root $\mu_{k+1} \in (\mu_{k-1}, \mu_k)$. For every $\lambda \in (\mu_1, \mu_k)$ the error representation of p yields the existence of some $\xi \in (\mu_1, \mu_k)$ such that

$$\chi_n(\mu) - p(\mu) = \frac{\chi_n^{(2k)}(\xi)}{(2k)!} \prod_{j=1}^{k} (\lambda - \mu_j)^2.$$

By Theorem 1 μ_k is less than the smallest root of χ_n'. Hence, from inequality (4) one gets

$$p(\mu) \le \chi_n(\mu) \quad \text{for every } \mu \in [\mu_{k-1}, \mu_k]$$

and therefore $\mu_{k+1} \le \lambda_1(T_n)$.

Let $\tilde{p}$ be the interpolation polynomial of degree $2k-2$ that satisfies the conditions

$$\tilde{p}(\mu_1) = \chi_n(\mu_1),\ \tilde{p}(\mu_j) = \chi_n(\mu_j),\ \tilde{p}'(\mu_j) = \chi_n'(\mu_j), \quad j = 2, \ldots, k.$$

Then for every $\lambda \in (\mu_{k-1}, \mu_k)$ there exists $\tilde{\xi} \in [\mu_1, \mu_k]$ such that

$$\chi_n(\mu) - \tilde{p}(\mu) = \frac{\chi_n^{(2k-1)}(\xi)}{(2k-1)!}(\lambda - \mu_1)\prod_{j=2}^{k}(\lambda - \mu_j)^2.$$

Inequality (5) implies

$$\chi_n(\mu) \leq \tilde{p}(\mu) \quad \text{for every } \mu \in [\mu_{k-1}, \mu_k].$$

Thus, every root $\tilde{\mu}_{k+1} \in (\mu_{k-1}, \mu_k)$ of $\tilde{p}$ is an upper bound of $\lambda_1(T_n)$.

Since the degrees $2k-1$ and $2k-2$ are relatively small compared to the dimension n we obtain an error bound at negligible cost. If this bound does not satisfy our accuracy requirements we evaluate $\chi_n(\mu_{k+1})$ and $\chi_n'(\mu_{k+1})$ and repeat the interpolations with the additional knot μ_{k+1}.

The following MATLAB function uses the pegasus method to determine the root of the polynomials p and $\tilde{p}$, respectively. $p(\mu) \leq \chi_n(\mu)$ implies that for p the maximum knot which is smaller than $\lambda_1(T_n)$ and any upper bound of $\lambda_1(T_n)$ are suitable initial values of the pegasus method. Similarly $\tilde{p}$ attains different signs at any lower bound of $\lambda_1(T_n)$ and at $\mu_k > \lambda_1(T_n)$.

```
function [lambda,nn]=dnm_h(t,n,tol);
% dnm_h determines the smallest eigenvalue of a symmetric positive
% definite Toeplitz matrix using double step Newton's method until
% a parameter > lambda is reached. After that the approximations are
% improved by Hermitian interpolation of the characteristic
% polynomial. Iteration is terminated if the relative error is
% smaller than tol.
% Uses function CHARACD to evaluate the characteristic polynomial
% and its derivative, and functions NAHERM and NAHERMM to evaluate
% Hermitian interpolation polynomials p and p~ by Neville-Aitken's
% method.
%
x(1)=0;
[f(1),df(1),normy,beta]=characd(0,t,n);
x(2)=-2*f(1)/df(1);
nn=1;
while f(nn) > 0
  nn=nn+1;
  [f(nn),df(nn),normy,beta]=characd(x(nn),t,n);
  if f(nn) > 0
    x(nn+1)=x(nn)-2*f(nn)/df(nn); end;
```

```
  end;
phi=f(nn)/df(nn);
err=(beta/(1+normy)+phi)/(x(nn)-phi);
la_l=x(nn-1);
la_r=min(x(nn),x(nn)+beta/(1+normy));
la_rr=x(nn);
while err > tol
  la1=la_l;   f1=nah(la1,x,f,df,nn);
  la2=la_r;   f2=nah(la2,x,f,df,nn);
  while abs(la2-la1)/max(la1,1e-20) > 1e-8
    la3=la2-(la2-la1)*f2/(f2-f1);
    f3=nah(la3,x,f,df,nn);
    if sign(f2)*sign(f3) < 0
      la1=la2;la2=la3;f1=f2;f2=f3;
    else
      la2=la3;tau=min(0.5,f2/(f2+f3));f1=tau*f1;f2=f3; end;
    end;
  la_l=min(la1,la2);
  la1=la_l;   f1=nahm(la1,x,f,df,nn);
  la2=la_rr;  f2=nahm(la2,x,f,df,nn);
  while abs(la2-la1)/la1 > 1e-8
    la3=la2-(la2-la1)*f2/(f2-f1);
    f3=nahm(la3,x,f,df,nn);
    if sign(f2)*sign(f3) < 0
      la1=la2;la2=la3;f1=f2;f2=f3;
    else
      la2=la3;tau=min(f2/(f2+f3),0.5);f1=tau*f1;f2=f3; end;
    end;
  la_r=max(la1,la2);
  err=(la_r-la_l)/la_l;
  if err > tol
    nn=nn+1;
    x(nn)=la_l;
    [f(nn),df(nn),normy,beta]=characd(x(nn),t,n);
    end;
  end;
lambda=la_l;
```

Assume again that

$$\mu_1 < \mu_2 < \ldots < \mu_{k-1} < \lambda_1(T_n) < \mu_k$$

are determined by the double step secant method of subsection 2.2. Let p be the polynomial of degree $k-1$ which satisfies the Lagrangian interpolation conditions

$$p(\mu_j) = \chi_n(\mu_j), \quad j = 1, \ldots, k.$$

If $\mu \in [\mu_1, \mu_k]$ then for some $\xi \in [\mu_1, \mu_k]$ the error representation

$$\chi_n(\mu) - p(\mu) = \frac{\chi_n^{(k)}(\xi)}{k!} \prod_{j=1}^{k} (\mu - \mu_j) =: \frac{\chi_n^{(k)}(\xi)}{k!} w(\mu)$$

holds.

Since $w(\mu) \leq 0$ for $\mu \in [\mu_{k-1}, \mu_k]$ we obtain

$$\chi_n(\mu) \begin{cases} \leq p(\mu) & \text{if } k \text{ is even} \\ \geq p(\mu) & \text{if } k \text{ is odd} \end{cases} .$$

If $\tilde{p}$ denotes the polynomial of degree k that satisfies the conditions

$$\tilde{p}'(\mu_1) = \chi_n'(\mu_1) \quad \text{and} \quad \tilde{p}(\mu_j) = \chi_n(\mu_j), \quad j = 1, \ldots, k,$$

(notice that the initial step is a double Newton step and therefore $\chi_n'(\mu_1)$ is available) then we obtain from the error representation

$$\chi_n(\mu) - \tilde{p}(\mu) = \frac{\chi_n^{(k+1)}(\tilde{\xi})}{k!} (\mu - \mu_1)^2 \prod_{j=2}^{k} (\mu - \mu_j) =: \frac{\chi_n^{(k+1)}(\tilde{\xi})}{k!} \tilde{w}(\mu)$$

in a similar way as above

$$\chi_n(\mu) \begin{cases} \leq \tilde{p}(\mu) & \text{if } k \text{ is odd} \\ \geq \tilde{p}(\mu) & \text{if } k \text{ is even} \end{cases} .$$

Hence in any case the roots of the polynomials p and $\tilde{p}$ yield an inclusion of the smallest eigenvalue $\lambda_1(T_n)$. On this observation we base the following procedure.

```
function [lambda,nn]=dsm_l(t,n,tol);
% dsm_l determines the smallest eigenvalue of a symmetric positive
% definite Toeplitz matrix. Using a double Newton step and subsequent
% double secant steps parameters x(1)<x(2)<...<x(nn)<lambda<x(nn+1)
% are determined. The bounds are improved by Lagrangian interpolation
% polynomials. The iteration is terminate if the relative error is
% less than tol.
% Uses functions CHARAC and CHARACD to evaluate the characteristic
% polynomial and additionally its derivative. Uses PEGASUS1 and
% PEGASUS2 to determine a root of interpolation polynomials by the
% pegasus method.
%
  x(1)=0;
  nn=1;
% double Newton step
  [f(1),df,normy,beta]=characd(0,t,n);
  h=f(1)/df;
```

```
  x(2)=-2*h;
% double step secant steps
  while h < 0
    nn=nn+1;
    [f(nn),normy,beta]=charac(x(nn),t,n);
    h=(x(nn)-x(nn-1))*f(nn)/(f(nn)-f(nn-1));
    x(nn+1)=x(nn)-2*h;
    end;
  la_rr=x(nn);
  la_l=x(nn-1);
  la_ll=la_l;
  la_r=min(0.5*(x(nn)+x(nn+1)),x(nn)+beta/(1+normy));
  err=(la_r-la_l)/max(la_l,1e-20);
% Lagrangian interpolation steps
  while err > tol
    [la1,la2]=pegasus1(la_ll,la_rr,x,f,nn);
    if nn == 2*round(0.5*nn)
      la_r=la2;
      [la1,la2]=pegasus2(la_ll,la_r,x,f,df,nn);
      la_l=la1;
    else
      la_l=la1;
      [la1,la2]=pegasus2(la_l,la_rr,x,f,df,nn);
      la_r=la2;
      end;
    x(nn+1)=la_l;
    err=(la_r-la_l)/la_l;
    if err > tol
      nn=nn+1;
      [f(nn),normy,beta]=charac(x(nn),t,n);
      la_ll=x(nn);
      end;
    end;
  lambda=la_l;
```

Example:

dimension	double step Newton method		double step secant method	
	flops	steps	flops	steps
32	1.751 $E04$	3.99	1.745 $E04$	5.91
64	6.108 $E04$	4.35	6.506 $E04$	6.63
128	2.008 $E05$	4.06	2.295 $E05$	6.24
256	8.790 $E05$	4.40	9.927 $E05$	6.95
512	3.953 $E06$	4.99	4.618 $E06$	8.24

Tab. 4. Double step methods improved by interpolation

4 Concluding remarks

We have presented several generalizations of Newton's method for the characteristic polynomial to determine the minimum eigenvalue of a symmetric and positive definite Toeplitz matrix. They are conceptually much simpler than the approaches considered in [1], [3], [4] and [11] since they do not need a bisection phase to determine a suitable initial value in the interval $(\lambda_1(T_n), \lambda_1(T_{n-1}))$ for a root finding method for the secular equation. On the other hand the methods in [3], [4] and [11] are faster than the methods considered here. The following table shows the average number of flops and of Yule Walker systems that had to be solved in the symmetric projection method of [11] and for comparison of the method DNM_H which is the fastest method introduced in this report.

dimension	double Newton method		symmetric projection method	
	flops	steps	flops	steps
32	1.751 $E04$	3.99	1.117 $E04$	3.60
64	6.108 $E04$	4.35	3.574 $E04$	3.72
128	2.008 $E05$	4.06	1.330 $E05$	3.81
256	8.790 $E05$	4.40	5.425 $E05$	4.03
512	3.953 $E06$	4.99	2.202 $E06$	5.15

Tab. 5. DNM_H method and symmetric projection method

5 Appendix : MATLAB functions

In this Appendix we collect some m-files that are needed in the methods of the previous sections.

The function that evaluates the characteristic polynomial of a normalized Toeplitz matrix is needed in SM, DSM, and DSM_L:

```
function [f,normy,beta]=charac(la,t,n);
% charac evaluates the characteristic polynomial of a
% normalized Toeplitz matrix at la.
%
y=zeros(n-1,1);
beta=1-la;
f=beta;
alpha=-t(1)/beta;
for i=1:n-1
  if i > 1
    alpha=-(t(i)+flipud(y(1:i-1))'*t(1:i-1))/beta;
    y(1:i-1)=y(1:i-1)+alpha*flipud(y(1:i-1));
    end;
```

```
  y(i)=alpha;
  beta=beta*(1-alpha^2);
  f=f*beta;
  end;
normy=y'*y;
```

The function CHARACD evaluates both, the function value and the derivative of the characteristic polynomial. It is used in SM, DSM, HERM, DNM_H, and DSM_L.

```
function [f,df,normy,beta]=characd(la,t,n);
% charpol evaluates the characteristic polynomial of
% a normalized Toeplitz matrix and its derivative at la.
%
y=zeros(n-1,1);
beta=1-la;
f=beta;
df=-1;
alpha=-t(1)/beta;
for i=1:n-1
  if i > 1
    alpha=-(t(i)+flipud(y(1:i-1))'*t(1:i-1))/beta;
    y(1:i-1)=y(1:i-1)+alpha*flipud(y(1:i-1));
  end;
  y(i)=alpha;
  normy=y(1:i)'*y(1:i);
  beta=beta*(1-alpha^2);
  df=beta*df-f*(1+normy);
  f=f*beta;
  end;
```

NAHERM evaluates the Hermitian interpolation polynomial which satisfies the interpolation conditions $p(x_j) = f_j$, $p'(x_j) = df_j$, $j = 1, \ldots, m$, by the Neville Aitken scheme. It is used in HERM and DNM_H

```
function pp=naherm(z,x,f,df,m);
% Neville Aitken scheme evaluating the Hermitian
% interpolation polynomial of degree 2*m-1 satisfying
% p(x_j)=f_j, j=1,...,m, and p'(x_j)=df_j, j=1,...,m.
%
for j=1:m
  xx(2*j-1)=x(j);
  xx(2*j)=x(j);
  end
for j=1:m-1
```

```
  p(2*j-1)=f(j)+df(j)*(z-x(j));
  p(2*j)=f(j)+(f(j+1)-f(j))*(z-x(j))/(x(j+1)-x(j));
  end;
p(2*m-1)=f(m)+df(m)*(z-x(m));
for k=2:2*m-1
  for j=1:2*m-k
    p(j)=(p(j)*(z-xx(j+k))-p(j+1)*(z-xx(j)))/(xx(j)-xx(j+k));
    end;
  end;
pp=p(1);
```

Similarly NAHERMM evaluates the modified Hermitian interpolation polynomial where at x_1 only a Lagrangian condition $p(x_1) = f_1$ is satisfied. It is used in DNM_H.

```
function pp=nahermm(z,x,f,df,m);
% Neville Aitken scheme for evaluating the modified Hermitian
% interpolation polynomial of degree 2*m-2 satisfying
% p(x_j)=f_j, j=1,...,m, and p'(x_j)=df_j, j=2,...,m.
%
xx(1)=x(1);
for j=2:m
  xx(2*j-2)=x(j);
  xx(2*j-1)=x(j);
  end
for j=1:m-1
  p(2*j)=f(j+1)+df(j+1)*(z-x(j+1));
  p(2*j-1)=f(j)+(f(j+1)-f(j))*(z-x(j))/(x(j+1)-x(j));
  end;
for k=2:2*m-2
  for j=1:2*m-1-k
    p(j)=(p(j)*(z-xx(j+k))-p(j+1)*(z-xx(j)))/(xx(j)-xx(j+k));
    end;
  end;
pp=p(1);
```

The following two functions NALAG and NALAGH evaluate the Lagrangian and the modified Lagrangian interpolation polynomial by the Neville Aitken scheme.

```
function pp=nalag(z,x,f,m);
% NALAG evaluates the Lagrangian interpolation polynomial that
% satisfies the conditions p(x_j)=f_j, j=1,...,m.
%
for j=1:m
  p(j)=f(j);
```

```
  zz=z-x(j);
  for i=j-1:-1:1
    p(i)=p(i+1)+(p(i)-p(i+1))*zz/(x(i)-x(j));
    end;
  end;
pp=p(1);


function pp=nalagh(z,x,f,df,m);
% NALAGH evaluates the Lagrangian interpolation polynomial
% satisfying p(x_j)=f_j, j=1,...,m, und and additionally
% p'(0)=df by the Neville - Aitken scheme
%
xx(1)=0;
for j=2:m+1
  xx(j)=x(j-1);
  end;
p(1)=f(1)+df*z;
for j=2:m
  p(j)=f(j)+(f(j)-f(j-1))*(z-x(j))/(x(j)-x(j-1));
  end;
for k=2:m
  for j=1:m+1-k
    p(j)=(p(j)*(z-xx(j+k))-p(j+1)*(z-xx(j)))/(xx(j)-xx(j+k));
    end;
  end;
pp=p(1);
```

PEGASUS1 determines a root of the Lagrangian interpolation polynomial that satisfies the conditions $p(x_j) = f_j$, $j = 1, \ldots, k$, by the pegasus method. It is used in DSM_L.

```
function [la1,la2]=pegasus1(lal,lar,x,f,m);
% pegasus1 determines lower and upper bounds la1 und la2 in the
% interval [lal,lar] of a root of the Lagrangian interpolation
% polynomial of degree m-1 satisfying p(x_j)=f_j, j=1,\dots,m.
% Uses the function NALAG to evaluate this polynomial.
%
la1=lau;la2=lao;
f1=nalag(la1,x,f,m);
f2=nalag(la2,x,f,m);
while abs(la2-la1)/max(1e-20,la1) > 1e-8
  la3=la2-(la2-la1)*f2/(f2-f1);
  f3=nalag(la3,x,f,m);
  if sign(f2)*sign(f3) == 0
```

```
    if la2 == 0
      la1=la2
    else
      la1=la3;la2=la3;
      end;
  elseif sign(f2)*sign(f3) < 0
    la1=la2;la2=la3;f1=f2;f2=f3;
  else
    la2=la3;tau=min(0.5,f2/(f2+f3));f1=tau*f1;f2=f3;
    end;
  end;
la3=min(la1,la2);
la2=max(la1,la2);
la1=la3;
```

Similarly PEGASUS2 determines a root of the Lagrangian interpolation polynomial which additionally satisfies the Hermitian condition $p'(x_1) = df$.

```
function [la1,la2]=pegasus2(lau,lao,x,f,df,m);
% pegasus2 determines lower and upper bounds la1 und la2 in the
% interval [lal,lar] of a root of the Lagrangian interpolation
% polynomial of degree m-1 satisfying p(x_j)=f_j, j=1,\dots,m, and
% additionally p'(x_1)=df.
% Uses the function NALAGH to evaluate this polynomial.
%
la1=lau;la2=lao;
f1=nalagh(la1,x,f,df,m);
f2=nalagh(la2,x,f,df,m);
while abs(la2-la1)/max(1e-20,la1) > 1e-8
  la3=la2-(la2-la1)*f2/(f2-f1);
  f3=nalagh(la3,x,f,df,m);
  if sign(f2)*sign(f3) == 0
    if f2 == 0
      la1=la2;
    else
      la1=la3;la2=la3;
      end;
  elseif sign(f2)*sign(f3) < 0
    la1=la2;la2=la3;f1=f2;f2=f3;
  else
    la2=la3;tau=min(0.5,f2/(f2+f3));f1=tau*f1;f2=f3; end;
  end;
la3=min(la1,la2);
la2=max(la1,la2);
la1=la3;
```

References

[1] G. Cybenko and C. Van Loan, Computing the minimum eigenvalue of a symmetric positive definite Toeplitz matrix. SIAM J. Sci. Stat. Comput. 7 (1986), pp. 123 — 131

[2] G.H. Golub and C.F. Van Loan, Matrix Computations. 3nd edition. The John Hopkins University Press, Baltimore and London, 1996.

[3] W. Mackens and H. Voss, The minimum eigenvalue of a symmetric positive definite Toeplitz matrix and rational Hermitian interpolation. SIAM J. Matr. Anal. Appl. 18 (1997), pp. 521 — 534

[4] W. Mackens and H. Voss, A projection method for computing the minimum eigenvalue of a symmetric positive definite Toeplitz matrix. To appear in Lin. Alg. Appl.

[5] N. Mastronardi and D. Boley, Computing the smallest eigenpair of a symmetric positive definite Toeplitz matrix. To appear in SIAM J. Sci. Comput.

[6] F. Noor and S.D. Morgera, Recursive and iterative algorithms for computing eigenvalues of Hermitian Toeplitz matrices. IEEE Trans. Signal Processing 41 (1993), pp. 1272 — 1280

[7] B.N. Parlett, The Symmetric Eigenvalue Problem. Prentice—Hall, Englewood Cliffs 1980

[8] V.F. Pisarenko, The retrieval of harmonics from a covariance function. Geophys. J. R. astr. Soc. 33 (1973), pp. 347 — 366

[9] J. Stoer and R. Bulirsch, Introduction to Numerical Analysis, Springer Verlag, New York 1980

[10] W.F. Trench, Numerical solution of the eigenvalue problem for Hermitian Toeplitz matrices. SIAM J. Matr. Nal. Appl. 10 (1989), pp. 135 — 146

[11] H. Voss, Symmetric schemes for computing the minimum eigenvalue of a symmetric Toeplitz matrix. Submitted to Lin. Alg. Appl.

[12] H. Voss and W. Mackens, Computing the minimal eigenvalue of a symmetric Toeplitz matrix. ZAMM 76, Proceedings of ICIAM/GAMM95, Issue 2: Applied Analysis, O. Mahrenholtz, R. Mennicken (eds.), pp. 701 — 702 (1996)