Numerical Algorithms 24 (2000) 99–116

Coupled Vandermonde matrices and the superfast computation of Toeplitz determinants *

Peter Kravanja ** and Marc Van Barel

Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200 A, B-3001 Heverlee, Belgium

E-mail: peter.kravanja@na-net.ornl.gov; marc.vanbarel@cs.kuleuven.ac.be

Let n be a positive integer, let $a_{-n+1}, \ldots, a_{-1}, a_0, a_1, \ldots, a_{n-1}$ be complex numbers and let $T := [a_{k-l}]_{k,l=0}^{n-1}$ be a nonsingular $n \times n$ complex Toeplitz matrix. We present a superfast algorithm for computing the determinant of T. Superfast means that the arithmetic complexity of our algorithm is $O(N \log^2 N)$, where N denotes the smallest power of 2 that is larger than or equal to n. We show that $\det T$ can be computed from the determinant of a certain coupled Vandermonde matrix. The latter matrix is related to a linearized rational interpolation problem at roots of unity and we show how its determinant can be calculated by multiplying the pivots that appear in the superfast interpolation algorithm that we presented in a previous publication.

Keywords: Toeplitz determinants, rational interpolation, coupled Vandermonde matrices

AMS subject classification: 65F40, 65D05

1. Introduction

Let n be a positive integer, let $a_{-n+1}, \ldots, a_{-1}, a_0, a_1, \ldots, a_{n-1}$ be complex numbers and let $T = T_n := [a_{k-l}]_{k,l=0}^{n-1}$ be a nonsingular $n \times n$ complex Toeplitz matrix. We consider the problem of computing the determinant of T.

Let N denote the smallest power of 2 that is larger than or equal to n. In [5,6] we used a formula of Heinig and Rost [4] for the generating function of T^{-1} to represent the $N \times N$ matrix

$$\left[T^{-1}\right]_N := \begin{bmatrix} T^{-1} & 0 \\ 0 & 0 \end{bmatrix}$$

in terms of discrete Fourier transform matrices and diagonal matrices. The latter matrices involve certain polynomials (known as the *canonical fundamental system* of T) evaluated at roots of unity. These polynomials can be computed by solving two linearized rational interpolation problems at the $2N$th roots of unity. In [5,6] we presented a stabilized generically superfast algorithm for solving such interpolation

* This research was partially supported by the Fund for Scientific Research-Flanders (FWO-V), project "Orthogonal Systems and Their Applications", grant #G.0278.97.

** Corresponding author.

problems. (An earlier version of this algorithm can be found in [8].) Superfast means that the arithmetic complexity of our algorithm is $O(\nu \log^2 \nu)$ for a problem that consists of ν interpolation points. Generically refers to the fact that in some exceptional cases the complexity of the algorithm is only $O(\nu^2)$. We combined our superfast interpolation algorithm and the explicit formula for $[T^{-1}]_N$ to obtain a superfast Toeplitz solver, i.e., a superfast algorithm for solving linear systems of equations that have Toeplitz structure.

The two linearized rational interpolation problems that are related to the canonical fundamental system can be expressed in terms of a certain *coupled Vandermonde* matrix V_{C}. We will show how $\det T$ can be computed from $\det V_{\mathrm{C}}$ and how $\det V_{\mathrm{C}}$ can be computed from the pivots that appear in our interpolation algorithm. This will enable us to compute $\det T$ in a generically superfast way.

In a companion paper [7] we considered the related problem of computing the determinant of a complex Hankel matrix H. By exploiting the connections that exist between Hankel, Loewner, Cauchy and coupled Vandermonde matrices, we were able to show that the determinant of H can be computed from the determinant of a certain coupled Vandermonde matrix V_{C}^h (different from V_{C}). The present paper constitutes an improvement on [7] in the following sense: in [7] the size of H had to be a power of 2 whereas in the present paper there is no restriction on the size of T. Also, the entries that appear in V_{C} are less expensive to compute than those in V_{C}^h.

Note. In the case of *linear systems* of Toeplitz equations, the idea of transforming a Toeplitz matrix into a coupled Vandermonde matrix was proposed by Heinig [2,3].

2. A coupled Vandermonde matrix based on the symbol of T

The *symbol* of T is defined as the function

$$a : \mathbb{C}_0 \to \mathbb{C} : z \mapsto a(z) := \frac{a_{-n+1}}{z^{n-1}} + \cdots + \frac{a_{-1}}{z} + a_0 + a_1 z + \cdots + a_{n-1} z^{n-1}.$$

Define $\omega_0, \ldots, \omega_{2N-1}$ as the $2N$th roots of unity,

$$\omega_k := \exp\left(\frac{2\pi \mathrm{i}}{2N} k\right), \quad k = 0, 1, \ldots, 2N-1,$$

and let V_{2N} be the corresponding *Vandermonde* matrix,

$$V_{2N} := \begin{bmatrix} 1 & \omega_0 & \ldots & \omega_0^{2N-1} \\ \vdots & \vdots & & \vdots \\ 1 & \omega_{2N-1} & \ldots & \omega_{2N-1}^{2N-1} \end{bmatrix} \in \mathbb{C}^{2N \times 2N}.$$

Note that $V_{2N}/\sqrt{2N}$ is a unitary matrix. In theorem 2 below we will need an explicit expression for the determinant of V_{2N}. The fact that $V_{2N} V_{2N}^H = 2N I_{2N}$ immediately implies that $|\det V_{2N}|^2 = (2N)^{2N}$ and, hence, $|\det V_{2N}| = (2N)^N$. In other words,

$\det V_{2N} = \alpha_N (2N)^N$, where $\alpha_N \in \mathbb{C}$ and $|\alpha_N| = 1$. In the next theorem we obtain the value of this constant α_N of modulus one in case N is a power of 2, which is the case that interests us in this paper.

Theorem 1. Suppose $N = 2^p$, where $p \in \mathbb{N} \setminus \{0, 1\}$. Then $\det V_{2N} = \mathrm{i}(2N)^N$. Also, $\det V_2 = -2$ and $\det V_4 = -16\mathrm{i}$.

Proof. The $2N$th roots of unity $\omega_0, \omega_1, \ldots, \omega_{2N-1}$ can be grouped into two groups. Define

$$\omega_k^+ := \omega_{2k} \quad \text{and} \quad \omega_k^- := \omega_{2k+1}$$

for $k = 0, 1, \ldots, N-1$. Then $\omega_0^+, \omega_1^+, \ldots, \omega_{N-1}^+$ are the Nth roots of unity and $\omega_0^-, \omega_1^-, \ldots, \omega_{N-1}^-$ are rotated Nth roots of unity,

$$\omega_k^- = \eta \omega_k^+, \quad k = 0, 1, \ldots, N-1,$$

where $\eta := \exp(\pi \mathrm{i}/N)$. Note that $\eta^N = -1$. Let $V_N \in \mathbb{C}^{N \times N}$ be the Vandermonde matrix based on the Nth roots of unity,

$$V_N := \begin{bmatrix} 1 & \omega_0^+ & \ldots & \left[\omega_0^+\right]^{N-1} \\ \vdots & \vdots & & \vdots \\ 1 & \omega_{N-1}^+ & \ldots & \left[\omega_{N-1}^+\right]^{N-1} \end{bmatrix}.$$

By rearranging the rows of V_{2N} we obtain the following:

$$P_{2N} V_{2N} = \begin{bmatrix} 1 & \omega_0^+ & \ldots & \left[\omega_0^+\right]^{N-1} & \left[\omega_0^+\right]^{N} & \ldots & \left[\omega_0^+\right]^{2N-1} \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ 1 & \omega_{N-1}^+ & \ldots & \left[\omega_{N-1}^+\right]^{N-1} & \left[\omega_{N-1}^+\right]^{N} & \ldots & \left[\omega_{N-1}^+\right]^{2N-1} \\ 1 & \omega_0^- & \ldots & \left[\omega_0^-\right]^{N-1} & \left[\omega_0^-\right]^{N} & \ldots & \left[\omega_0^-\right]^{2N-1} \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ 1 & \omega_{N-1}^- & \ldots & \left[\omega_{N-1}^-\right]^{N-1} & \left[\omega_{N-1}^-\right]^{N} & \ldots & \left[\omega_{N-1}^-\right]^{2N-1} \end{bmatrix}.$$

Here $P_{2N} \in \mathbb{R}^{2N \times 2N}$ is the permutation matrix defined by

$$P_{2N}^{\mathrm{T}} := \begin{bmatrix} e_1 & e_3 & \ldots & e_{2N-1} & e_2 & e_4 & \ldots & e_{2N} \end{bmatrix},$$

where $e_j \in \mathbb{R}^{2N}$ denotes the jth canonical basis vector for $j = 1, \ldots, 2N$. Let $D_\eta := \mathrm{diag}(1, \eta, \ldots, \eta^{N-1}) \in \mathbb{C}^{N \times N}$. Since $[\omega_k^+]^N = 1$ and $[\omega_k^-]^N = \eta^N = -1$ for $k = 0, 1, \ldots, N-1$, it follows that

$$P_{2N} V_{2N} = \begin{bmatrix} V_N & V_N \\ V_N D_\eta & -V_N D_\eta \end{bmatrix}.$$

The Schur complement formula then implies that

$$\begin{aligned} \det P_{2N} \det V_{2N} &= \det V_N \det\left(-V_N D_\eta - V_N D_\eta V_N^{-1} V_N\right) \\ &= \det V_N \det(-2 V_N D_\eta) = (-2)^N \det D_\eta [\det V_N]^2. \end{aligned}$$

Since $\det D_\eta = \exp[(\pi \mathrm{i}/N)(1/2)(N-1)N] = \mathrm{i}^{N-1}$ and $\det P_{2N} = (-1)^{(1/2)N(N-1)} = \mathrm{i}^{N(N-1)}$ we may conclude that

$$\begin{aligned}\det V_{2N} &= (-1)^N 2^N \mathrm{i}^{N-1} \mathrm{i}^{-N(N-1)} [\det V_N]^2 = 2^N \mathrm{i}^{2N-(N-1)^2} [\det V_N]^2 \\ &= 2^N \mathrm{i}^{4N-(1+N^2)} [\det V_N]^2 = 2^N \mathrm{i}^{-(1+N^2)} [\det V_N]^2.\end{aligned}$$

Here we have used the fact that $\mathrm{i}^4 = 1$. As N is even, N^2 is a multiple of 4 and, hence, $\mathrm{i}^{N^2} = 1$. Thus,

$$\det V_{2N} = 2^N \mathrm{i}^3 [\det V_N]^2.$$

The fact that

$$\det V_2 = \det \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = -2 = 2\mathrm{i}^2$$

implies that $\det V_{2N}$ consists of two factors: a power of 2 and a power of i. Let ε_{2N} denote the power of 2 that appears in $\det V_{2N}$. Then $\varepsilon_2 = 1$ and in general we have the recursion

$$\varepsilon_{2N} = N + 2\varepsilon_N.$$

The solution to this difference equation is given by

$$\varepsilon_{2N} = N + N \log_2 N.$$

Similarly, if we let ζ_{2N} denote the power of i that appears in $\det V_{2N}$, then $\zeta_2 = 2$ and

$$\zeta_{2N} = 3 + 2\zeta_N.$$

It follows that

$$\zeta_{2N} = 5N - 3.$$

Therefore,

$$\det V_{2N} = 2^{\varepsilon_{2N}} \mathrm{i}^{\zeta_{2N}} = 2^{N+N\log_2 N} \mathrm{i}^{5N-3} = \mathrm{i}^{N+1} (2N)^N.$$

For $N = 1$ and $N = 2$ we have indeed that $\det V_2 = -2$ and $\det V_4 = -16\mathrm{i}$. If N is a power of 2 that is larger than or equal to 4, then N is a multiple of 4 and, hence, $\mathrm{i}^N = 1$. This proves the theorem. □

Let us define the *coupled Vandermonde* matrix $V_{\mathrm{C}} \in \mathbb{C}^{2N \times 2N}$ as follows: the $(k+1)$th row of V_{C} is given by

$$\begin{bmatrix} \omega_k^n & \omega_k^{n+1} & \dots & \omega_k^{2N-1} & a(\omega_k) & \omega_k a(\omega_k) & \dots & \omega_k^{n-1} a(\omega_k) \end{bmatrix}$$

for $k = 0, 1, \dots, 2N-1$. The following theorem shows how $\det T$ can be computed from $\det V_{\mathrm{C}}$.

Theorem 2. $\det V_{\mathrm{C}} = (-1)^n \det V_{2N} \det T$.

Proof. We partition the Vandermonde matrix V_{2N} as

$$V_{2N} =: \begin{bmatrix} V_{2N}^{(n)} & V_{2N}^{(2N-n)} \end{bmatrix},$$

where $V_{2N}^{(n)} \in \mathbb{C}^{2N\times n}$ and $V_{2N}^{(2N-n)} \in \mathbb{C}^{2N\times(2N-n)}$. Let us consider the rectangular Toeplitz matrix in $\mathbb{C}^{2N\times n}$ whose first row is given by

$$[\,a_0 \quad a_{-1} \quad \ldots \quad a_{-n+1}\,]$$

and whose first column is given by

$$[\,a_0 \quad a_1 \quad \ldots \quad a_{n-1} \quad 0 \quad \ldots \quad 0 \quad a_{-n+1} \quad \ldots \quad a_{-2} \quad a_{-1}\,]^{\mathrm{T}}.$$

Let us partition this matrix as

$$\begin{bmatrix} T \\ \widetilde{T} \end{bmatrix},$$

where $\widetilde{T}$ belongs to $\mathbb{C}^{(2N-n)\times n}$. Define D_a as the diagonal matrix

$$D_a := \operatorname{diag}\big(a(\omega_k)\big)_{k=0}^{2N-1}.$$

Then the following holds:

$$V_{2N}\begin{bmatrix} T & 0 \\ \widetilde{T} & I_{2N-n} \end{bmatrix} = \begin{bmatrix} D_a V_{2N}^{(n)} & V_{2N}^{(2N-n)} \end{bmatrix} = V_{\mathrm{C}}\begin{bmatrix} 0 & I_{2N-n} \\ I_n & 0 \end{bmatrix}.$$

One can easily verify that

$$\det\begin{bmatrix} 0 & I_{2N-n} \\ I_n & 0 \end{bmatrix} = (-1)^{n(2N-n)} \det I_n = (-1)^{n^2} = (-1)^n.$$

It follows that indeed $\det V_{\mathrm{C}} = (-1)^n \det V_{2N} \det T$. □

Theorems 1 and 2 imply that the determinant of the Toeplitz matrix T can be computed from the determinant of the coupled Vandermonde matrix V_{C}:

$$\det T = \frac{1}{\mathrm{i}}(-1)^n (2N)^{-N} \det V_{\mathrm{C}}$$

if $N = 2^p$, where $p \in \mathbb{N} \setminus \{0, 1\}$. It remains to compute $\det V_{\mathrm{C}}$ in an efficient and accurate way. In the next section we will exploit the connection between coupled Vandermonde matrices and linearized rational interpolation problems. We will show how the determinant of a coupled Vandermonde matrix can be computed by multiplying the pivots that appear in the superfast interpolation algorithm that we presented in [5,8].

3. Superfast rational interpolation

Instead of the coupled Vandermonde matrix V_{C} that we have defined in the previous section, we will now consider a more general coupled Vandermonde matrix in $\mathbb{C}^{2N\times 2N}$, which we will also denote by V_{C}. Let $m := 2N - n$ and suppose that the

complex numbers s_k, e_k and f_k are given for $k = 1, \ldots, m+n$. We assume that the s_k's are mutually distinct. Suppose that the kth row of V_{C} is given by

$$\begin{bmatrix} e_k & e_k s_k & \ldots & e_k s_k^{m-1} & f_k & f_k s_k & \ldots & f_k s_k^{n-1} \end{bmatrix}$$

for $k = 1, \ldots, m+n$. Observe that by setting

$$s_k := \omega_{k-1}, \qquad e_k := \omega_{k-1}^n \quad \text{and} \quad f_k := a(\omega_{k-1})$$

for $k = 1, \ldots, m+n = 2N$ we obtain the coupled Vandermonde matrix defined in the previous section.

Without loss of generality we may assume that $m \geqslant n$. Indeed, if m happens to be less than n, then we can swap the left and right block of V_{C}, an operation that leads only to a possible sign change of $\det V_{\mathrm{C}}$.

Let us introduce the following notation. Let $g = [g_k]_{k=1}^{\gamma} \in \mathbb{C}^{\gamma}$ be a column vector of length γ for some positive integer γ. Then $\Lambda(\gamma)$ denotes the diagonal matrix

$$\Lambda(\gamma) := \operatorname{diag}(g_1, \ldots, g_\gamma) \in \mathbb{C}^{\gamma \times \gamma}.$$

Also, let $g_{\alpha:\beta} := [g_k]_{k=\alpha}^{\beta} \in \mathbb{C}^{\beta-\alpha+1}$ for some integers α and β, $1 \leqslant \alpha \leqslant \beta \leqslant \gamma$. Define the column vectors $s, e, f \in \mathbb{C}^{m+n}$ as

$$s := \begin{bmatrix} s_1 \\ \vdots \\ s_{m+n} \end{bmatrix}, \qquad e := \begin{bmatrix} e_1 \\ \vdots \\ e_{m+n} \end{bmatrix} \quad \text{and} \quad f := \begin{bmatrix} f_1 \\ \vdots \\ f_{m+n} \end{bmatrix}$$

and let $V_{:,1:l}(s)$ denote the first l columns of the Vandermonde matrix based on s,

$$V_{:,1:l}(s) := \begin{bmatrix} 1 & s_1 & \ldots & s_1^{l-1} \\ \vdots & \vdots & & \vdots \\ 1 & s_{m+n} & \ldots & s_{m+n}^{l-1} \end{bmatrix} \in \mathbb{C}^{(m+n)\times l}$$

for some integer $l \in \{1, \ldots, m+n\}$. Then the coupled Vandermonde matrix V_{C} can be written as

$$V_{\mathrm{C}} = \begin{bmatrix} \Lambda(e) V_{:,1:m}(s) & \Lambda(f) V_{:,1:n}(s) \end{bmatrix}.$$

The numerical stability of the algorithm for computing $\det V_{\mathrm{C}}$ that we are going to derive is enhanced via pivoting. The algorithm uses a certain criterion to change the order of the components of s, which leads to a corresponding change in e and f. This form of pivoting corresponds to a permutation of the rows of V_{C} and hence can only change the sign of the determinant of V_{C}. It holds that

$$V_{\mathrm{C}}' := PV_{\mathrm{C}} = \begin{bmatrix} \Lambda(e') V_{:,1:m}(s') & \Lambda(f') V_{:,1:n}(s') \end{bmatrix},$$

where P is the permutation matrix, $s' := Ps$, $e' := Pe$ and $f' := Pf$ are the permuted s, e and f vectors. To simplify the notation we will henceforth omit the primes. In other words, we will identify s' with s, e' with e, etc.

Let us partition the matrices $V_{:,1:m}(s)$ and $V_{:,1:n}(s)$ as follows:

$$V_{:,1:m}(s) = \begin{bmatrix} V_{1,1} & V_{1,2} \\ V_{2,1} & V_{2,2} \end{bmatrix} \quad \text{and} \quad V_{:,1:n}(s) = \begin{bmatrix} V_1 \\ V_2 \end{bmatrix},$$

where the row size of $V_{1,1}$, $V_{1,2}$ and V_1 is equal to $m-n$, the row size of $V_{2,1}$, $V_{2,2}$ and V_2 is equal to $2n$, the column size of $V_{1,1}$ and $V_{2,1}$ is equal to $m-n$ and the column size of $V_{1,2}$, $V_{2,2}$, V_1 and V_2 is equal to n. Then V_{C} can be written as

$$V_{\mathrm{C}} = \begin{bmatrix} \Lambda(e_{1:m-n})V_{1,1} & \Lambda(e_{1:m-n})V_{1,2} & \Lambda(f_{1:m-n})V_1 \\ \Lambda(e_{m-n+1:m+n})V_{2,1} & \Lambda(e_{m-n+1:m+n})V_{2,2} & \Lambda(f_{m-n+1:m+n})V_2 \end{bmatrix}.$$

Our algorithm for computing the determinant of V_{C} will emerge from the constructive proof of the following equation.

Theorem 3.

$$V_{\mathrm{C}} \begin{bmatrix} N_{1,1} & N_{1,2} \\ 0 & N_{2,2} \\ 0 & D \end{bmatrix} = \begin{bmatrix} L_1 & 0 \\ X & L_2 \end{bmatrix}, \tag{1}$$

where the row size of $N_{1,1}$, $N_{1,2}$ and L_1 is equal to $m-n$, the row size of $N_{2,2}$ and D is equal to n, the row size of X and L_2 is equal to $2n$, the column size of $N_{1,1}$, L_1 and X is equal to $m-n$ and the column size of $N_{1,2}$, $N_{2,2}$, D and L_2 is equal to $2n$. The matrix $N_{1,1}$ has to be an upper triangular matrix containing ones on the main diagonal, the matrices $N_{2,2}$ and D have to be block upper triangular matrices whose blocks have size 1×2, the matrix L_1 has to be a lower triangular matrix and the matrix L_2 has to be a block lower triangular matrix whose blocks have size 2×2.

From the proof we will obtain an algorithm for constructing these matrices. If the 1×2 diagonal blocks of $N_{2,2}$ and D are combined into square blocks, then the determinant of these blocks has to be equal to one. Hence,

$$\det V_{\mathrm{C}} = \pm \det L_1 \det L_2, \tag{2}$$

where the possible sign change is equal to the determinant of

$$\begin{bmatrix} N_{2,2} \\ D \end{bmatrix}$$

and corresponds to the permutation needed to combine the diagonal blocks of $N_{2,2}$ and D.

Proof. Let us translate (1) into interpolation terms. We consider the jth column of the matrix $N_{1,1}$ as the stacking vector of a polynomial $n_{1,1}^{(j)}(z)$ for $j = 1, \ldots, m-n$.

These polynomials have to satisfy the following properties. Let $j \in \{1, \ldots, m-n\}$. Then

$$n_{1,1}^{(j)}(z) \text{ is a monic polynomial of degree } j-1,$$
$$e_k n_{1,1}^{(j)}(s_k) = 0 \quad \text{for } k = 1, \ldots, j-1.$$

The latter implies that $n_{1,1}^{(j)}(s_k) = 0$ for each $e_k \neq 0,\ k \in \{1, \ldots, j-1\}$. Without loss of generality we may assume that $e_k \neq 0$ for $k = 1, \ldots, m-n-1$. Indeed, suppose that there exists no permutation of the e_k's such that the first $m-n-1$ ones are different from zero. Then at least $m+n-(m-n-2) = 2n+2$ e_k's are equal to zero. However, as $2n+2 > n$ it follows that in this case $\det V_{\mathrm{C}} = 0$, which we exclude. Thus, $n_{1,1}^{(j)}(s_k) = 0$ for $j = 1, \ldots, m-n$ and $k = 1, \ldots, j-1$. In other words,

$$n_{1,1}^{(j)}(z) = \prod_{k=1}^{j-1}(z - s_k)$$

for $j = 1, \ldots, m-n$. Note that the diagonal entries of the lower triangular matrix L_1 are equal to

$$e_j n_{1,1}^{(j)}(s_j) = e_j \prod_{k=1}^{j-1}(s_j - s_k)$$

for $j = 1, \ldots, m-n$. Hence,

$$\det L_1 = \prod_{j=1}^{m-n} e_j \prod_{k=1}^{j-1}(s_j - s_k) = e_1 \cdots e_{m-n} \det V(s_1, \ldots, s_{m-n}), \tag{3}$$

where $V(s_1, \ldots, s_{m-n})$ denotes the Vandermonde matrix with nodes $s_1, \ldots, s_{m-n}$.

We consider the jth block column (of size $m \times 2$) of

$$\begin{bmatrix} N_{1,2} \\ N_{2,2} \end{bmatrix}$$

as the stacking vector of a row polynomial vector $n^{(j)}(z) \in \mathbb{C}[z]^{1\times 2}$ for $j = 1, \ldots, n$. Similarly, we consider the jth block column (of size $n \times 2$) of D as the stacking vector of a row polynomial vector $d^{(j)}(z) \in \mathbb{C}[z]^{1\times 2}$ for $j = 1, \ldots, n$. Then $\deg n^{(j)}(z) < m$ and $\deg d^{(j)}(z) < n$ for $j = 1, \ldots, n$. These polynomials have to satisfy the following properties. Let $j \in \{1, \ldots, n\}$. Then

$$\deg n^{(j)}(z) \leqslant m-n+j-1, \qquad \deg d^{(j)}(z) \leqslant j-1,$$
$$\det \begin{bmatrix} \text{coefficient of } z^{m-n+j-1} \text{ in } n^{(j)}(z) \\ \text{coefficient of } z^{j-1} \text{ in } d^{(j)}(z) \end{bmatrix} = 1, \tag{4}$$
$$e_k n^{(j)}(s_k) + f_k d^{(j)}(s_k) = 0 \quad \text{for } k = 1, \ldots, m-n+2(j-1).$$

If we define the 2×2 matrix polynomial $B_j(z)$ as

$$B_j(z) := \begin{bmatrix} n^{(j)}(z) \\ d^{(j)}(z) \end{bmatrix} \in \mathbb{C}[z]^{2\times 2},$$

then we can write the last property as

$$[e_k \quad f_k] B_j(s_k) = [0 \quad 0], \quad k = 1, \ldots, m-n+2(j-1).$$

In other words, $B_j(z)$ has to satisfy certain interpolation conditions, in addition to certain degree conditions and a determinant condition. □

Let us summarize what we have obtained so far. The determinant of the coupled Vandermonde matrix V_{C} can be computed from the product of $\det L_1$ and $\det L_2$, cf. equation (2). We have already obtained an explicit expression for $\det L_1$, cf. equation (3). The matrix L_2 is a block lower triangular matrix (whose blocks have size 2×2) and, hence, its determinant is equal to the product of the determinants of the diagonal blocks. We have seen that this matrix is related to linearized rational interpolation problems. We will now show how the matrix polynomials $B_j(z)$ can be computed. The 2×2 diagonal blocks of L_2 are obtained as a by-product of our interpolation algorithm.

Let $b_1(z)$ be the interpolating polynomial of degree $\leqslant m-n-1$ that satisfies $b_1(s_k) = -f_k/e_k$ for $k = 1, \ldots, m-n$. Define $B_1(z)$ as

$$B_1(z) := \begin{bmatrix} \prod_{k=1}^{m-n} (z - s_k) & b_1(z) \\ 0 & 1 \end{bmatrix}. \tag{5}$$

Clearly $B_1(z)$ satisfies (4) for $j = 1$. We will use the following recurrence relation to compute the 2×2 matrix polynomials $B_2(z), \ldots, B_n(z)$:

$$B_{j+1}(z) = B_j(z) B_{j\to j+1}(z), \quad j = 1, \ldots, n-1.$$

We will determine the 2×2 matrix polynomials $B_{j\to j+1}(z)$ such that

$$\deg B_{j\to j+1}(z) = 1 \quad \text{and} \quad \det\big(\operatorname{hdc} B_{j\to j+1}(z)\big) = 1 \tag{6}$$

for $j = 1, \ldots, n-1$. Here "hdc" denotes the highest degree coefficient. This already guarantees that $B_2(z), \ldots, B_n(z)$ satisfy the first three conditions of (4). The degrees of freedom that remain in fixing $B_{j\to j+1}(z)$ are used to satisfy the interpolation condition of (4), i.e.,

$$[e_k \quad f_k] B_j(s_k) B_{j\to j+1}(s_k) = [0 \quad 0] \quad \text{for } k = 1, \ldots, m-n+2j.$$

Since

$$[e_k \quad f_k] B_j(s_k) = [0 \quad 0]$$

for $k = 1, \ldots, m - n + 2(j-1)$, we have to determine $B_{j\to j+1}(z)$ such that

$$\begin{bmatrix} e^\star & f^\star \end{bmatrix} B_j(s^\star) B_{j\to j+1}(s^\star) = \begin{bmatrix} 0 & 0 \end{bmatrix}$$

and

$$\begin{bmatrix} e^{\star\star} & f^{\star\star} \end{bmatrix} B_j(s^{\star\star}) B_{j\to j+1}(s^{\star\star}) = \begin{bmatrix} 0 & 0 \end{bmatrix},$$

where $s^\star := s_{m-n+2j-1}$, $s^{\star\star} := s_{m-n+2j}$ and similarly for $e^\star$, $e^{\star\star}$, $f^\star$ and $f^{\star\star}$. If we define

$$\begin{bmatrix} l^\star & r^\star \end{bmatrix} := \begin{bmatrix} e^\star & f^\star \end{bmatrix} B_j(s^\star)$$

and

$$\begin{bmatrix} l^{\star\star} & r^{\star\star} \end{bmatrix} := \begin{bmatrix} e^{\star\star} & f^{\star\star} \end{bmatrix} B_j(s^{\star\star}),$$

then $B_{j\to j+1}(z)$ has to satisfy

$$\begin{aligned} \begin{bmatrix} l^\star & r^\star \end{bmatrix} B_{j\to j+1}(s^\star) &= \begin{bmatrix} 0 & 0 \end{bmatrix}, \\ \begin{bmatrix} l^{\star\star} & r^{\star\star} \end{bmatrix} B_{j\to j+1}(s^{\star\star}) &= \begin{bmatrix} 0 & 0 \end{bmatrix}. \end{aligned} \tag{7}$$

Note that the 2×2 diagonal block of L_2 that corresponds to $B_j(z)$ is equal to

$$\begin{bmatrix} l^\star & r^\star \\ l^{\star\star} & r^{\star\star} \end{bmatrix}. \tag{8}$$

Define the matrix polynomials $B^{(L)}_{j\to j+1}(z)$ and $B^{(R)}_{j\to j+1}(z)$ as

$$B^{(L)}_{j\to j+1}(z) := \begin{bmatrix} z - s_L & \alpha_L \\ 0 & 1 \end{bmatrix} \quad \text{and} \quad B^{(R)}_{j\to j+1}(z) := \begin{bmatrix} 1 & 0 \\ \alpha_R & z - s_R \end{bmatrix},$$

where $\alpha_L, \alpha_R, s_L, s_R \in \mathbb{C}$, $s_L \neq s_R$. We may assume that the 2×2 matrix (8) is nonsingular. Indeed, otherwise (2) implies that $\det V_{\mathrm{C}} = 0$. Thus, it is impossible that both $l^\star$ and $r^\star$ are equal to zero. Suppose that $l^\star \neq 0$. Then let $s_L := s^\star$ and $\alpha_L := -r^\star / l^\star$. It follows that

$$\begin{bmatrix} l^\star & r^\star \end{bmatrix} B^{(L)}_{j\to j+1}(s^\star) = \begin{bmatrix} 0 & 0 \end{bmatrix}.$$

We call the construction of $B^{(L)}_{j\to j+1}(z)$ a *left step* towards $B_{j\to j+1}(z)$. We have that

$$\begin{bmatrix} l^{\star\star} & r^{\star\star} \end{bmatrix} B^{(L)}_{j\to j+1}(s^{\star\star}) = \begin{bmatrix} l^{\star\star}(s^{\star\star} - s^\star) & l^{\star\star}\left(-\dfrac{r^\star}{l^\star}\right) + r^{\star\star} \end{bmatrix} =: \begin{bmatrix} \tilde{l}^{\star\star} & \tilde{r}^{\star\star} \end{bmatrix}.$$

Since

$$l^\star \tilde{r}^{\star\star} = \det \begin{bmatrix} l^\star & r^\star \\ l^{\star\star} & r^{\star\star} \end{bmatrix}, \tag{9}$$

it holds that $\tilde{r}^{\star\star} \neq 0$. Now let $s_R := s^{\star\star}$ and $\alpha_R := -\tilde{l}^{\star\star} / \tilde{r}^{\star\star}$. Then

$$\begin{bmatrix} \tilde{l}^{\star\star} & \tilde{r}^{\star\star} \end{bmatrix} B^{(R)}_{j\to j+1}(s^{\star\star}) = \begin{bmatrix} 0 & 0 \end{bmatrix}.$$

The construction of $B^{(R)}_{j\to j+1}(z)$ is called a *right step* towards $B_{j\to j+1}(z)$. We combine both steps to obtain $B_{j\to j+1}(z)$:

$$B_{j\to j+1}(z) = B^{(L)}_{j\to j+1}(z)B^{(R)}_{j\to j+1}(z) = \begin{bmatrix} z+(\alpha_L\alpha_R - s_L) & \alpha_L(z-s_R) \\ \alpha_R & z-s_R \end{bmatrix}.$$

This matrix polynomial satisfies indeed the degree and determinant condition (6) and the interpolation condition (7).

Similarly, in case $r^\star \neq 0$, one can construct $B_{j\to j+1}(z)$ as a right step followed by a left step.

3.1. *Fast algorithms for computing the matrix polynomial* $B_n(z)$

We can now formulate the following algorithm for computing $B_n(z)$. One can easily check that it requires $O((m+n)^2)$ floating point operations.

Compute $B_1(z)$ via (5).
for $j = 1, \ldots, n-1$ **do**
  $s^\star \leftarrow s_{m-n+2j-1}$; $s^{\star\star} \leftarrow s_{m-n+2j}$
  $e^\star \leftarrow e_{m-n+2j-1}$; $e^{\star\star} \leftarrow e_{m-n+2j}$
  $f^\star \leftarrow f_{m-n+2j-1}$; $f^{\star\star} \leftarrow f_{m-n+2j}$
  $[l^\star \quad r^\star] \leftarrow [e^\star \quad f^\star]B_j(s^\star)$
  $[l^{\star\star} \quad r^{\star\star}] \leftarrow [e^{\star\star} \quad f^{\star\star}]B_j(s^{\star\star})$
  if left step followed by right step **then**
    $B_{j+1}(z) \leftarrow B_j(z)\cdot B^{(L)}_{j\to j+1}(z)\cdot B^{(R)}_{j\to j+1}(z)$
  else
    $B_{j+1}(z) \leftarrow B_j(z)\cdot B^{(R)}_{j\to j+1}(z)\cdot B^{(L)}_{j\to j+1}(z)$
  end if
end for

The algorithm computes the values $l^\star$, $r^\star$, $l^{\star\star}$ and $r^{\star\star}$ only at the time when they are needed to continue the computations. To compute these values, the algorithm needs the value that the matrix polynomial $B_j(z)$ takes at the points $s^\star$ and $s^{\star\star}$. Our algorithm can be categorized as a *Levinson-like algorithm*. Note that a pivoting strategy cannot be based on the values of $[e_k \quad f_k]B_j(s_k)$ in the interpolation points s_k that have not yet been considered, as these values are not available. A *Schur-like algorithm* can overcome this problem.

Compute $B_1(z)$ via (5).
for $k = 1, \ldots, 2n$ **do**
  $[l_k \quad r_k] \leftarrow [e_{m-n+k} \quad f_{m-n+k}]B_1(s_{m-n+k})$
end for
for $j = 1, \ldots, n-1$ **do**

Apply pivoting on $[\,s_{m-n+k} \quad l_k \quad r_k\,]_{k=2(j-1)+1}^{2n}$.
if left step followed by right step **then**
for $k = 2j+1, \ldots, 2n$ **do**

$$[\,l_k \quad r_k\,] \leftarrow [\,l_k \quad r_k\,] \begin{bmatrix} s_{m-n+k} - s^\star & \alpha_L \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \alpha_R & s_{m-n+k} - s^{\star\star} \end{bmatrix}$$

end for
$B_{j+1}(z) \leftarrow B_j(z) \cdot B_{j\to j+1}^{(L)}(z) \cdot B_{j\to j+1}^{(R)}(z)$
else
for $k = 2j+1, \ldots, 2n$ **do**

$$[\,l_k \quad r_k\,] \leftarrow [\,l_k \quad r_k\,] \begin{bmatrix} 1 & 0 \\ \alpha_R & s_{m-n+k} - s^\star \end{bmatrix} \begin{bmatrix} s_{m-n+k} - s^{\star\star} & \alpha_L \\ 0 & 1 \end{bmatrix}$$

end for
$B_{j+1}(z) \leftarrow B_j(z) \cdot B_{j\to j+1}^{(R)}(z) \cdot B_{j\to j+1}^{(L)}(z)$
end if
end for

Note that the explicit computation of $B_{j+1}(z)$ can, in fact, be skipped since we only need some of the values of $[\,l_k \quad r_k\,]$ to compute the determinant of L_2.

Observe that the determinant of L_2 can be computed from the products of $l^\star$ and $\tilde{r}^{\star\star}$, cf. equation (9).

3.2. Superfast algorithms for computing the matrix polynomial $B_n(z)$

Superfast algorithms for computing $B_n(z)$ are based on a divide-and-conquer strategy. Let us start by considering the case that the matrix polynomial $B_1(z)$ is given and that n is even. Then $B_n(z)$ can be computed as

$$B_n(z) \leftarrow B_1(z) B_{1\to n/2}(z) B_{n/2\to n}(z),$$

where

$$\deg B_{1\to n/2}(z) = \frac{n}{2} \quad \text{and} \quad \det\big(\text{hdc}\, B_{1\to n/2}(z)\big) = 1,$$
$$\deg B_{n/2\to n}(z) = \frac{n}{2} \quad \text{and} \quad \det\big(\text{hdc}\, B_{n/2\to n}(z)\big) = 1,$$

and the following interpolation conditions are satisfied:

$$[\,e_k \quad f_k\,] B_1(s_k) B_{1\to n/2}(s_k) = [\,0 \quad 0\,]$$

for $k = (m-n)+1, \ldots, (m-n)+n$ and

$$[\,e_k \quad f_k\,] B_1(s_k) B_{1\to n/2}(s_k) B_{n/2\to n}(s_k) = [\,0 \quad 0\,]$$

for $k = (m-n)+n+1, \ldots, (m-n)+2n$.

If n is a power of 2, then these considerations lead to the following recursive algorithm:

$B_n(z) \leftarrow$ `compute_B_power_of_2` $(B_1(z), s, e, f, m, n)$
– Let τ_{fast} be an integer $\geqslant 1$.
if $2n \leqslant 2^{\tau_{\text{fast}}}$ **then**
compute $B_n(z)$ using a fast method
end if
for $k = (m-n)+1, \ldots, (m-n)+n$ **do**
$[l^{(1)}_{k-(m-n)} \quad r^{(1)}_{k-(m-n)}] \leftarrow [e_k \quad f_k] B_1(s_k)$ ... **[A]**
end for
$s^{(1)} \leftarrow s_{(m-n)+1:(m-n)+n}$
$l^{(1)} \leftarrow l^{(1)}_{1:n}$; $r^{(1)} \leftarrow r^{(1)}_{1:n}$
$B_{1\to n/2}(z) \leftarrow$ `compute_B_power_of_2` $(I_2, s^{(1)}, l^{(1)}, r^{(1)}, n/2, n/2)$
$B_{n/2}(z) \leftarrow B_1(z) \cdot B_{1\to n/2}(z)$ ... **[B]**
for $k = (m-n)+n+1, \ldots, (m-n)+2n$ **do**
$[l^{(2)}_{k-m} \quad r^{(2)}_{k-m}] \leftarrow [e_k \quad f_k] B_{n/2}(s_k)$ ... **[A′]**
end for
$s^{(2)} \leftarrow s_{m+1:m+n}$
$l^{(2)} \leftarrow l^{(2)}_{1:n}$; $r^{(2)} \leftarrow r^{(2)}_{1:n}$
$B_{n/2\to n}(z) \leftarrow$ `compute_B_power_of_2` $(I_2, s^{(2)}, l^{(2)}, r^{(2)}, n/2, n/2)$
$B_n(z) \leftarrow B_{n/2}(z) \cdot B_{n/2\to n}(z)$ ... **[B′]**

The matrix polynomial multiplications in steps **[B]** and **[B′]** can be done via the celebrated fast Fourier transform (see, for example, [1]). In this case, these steps require $\mathrm{O}((m-n/2)\log(m-n/2))$ and $\mathrm{O}(m \log m)$ floating point operations, respectively.

If the interpolation points s_k are the $(m+n)$th roots of unity, then these points can be permuted such that steps **[A]** and **[A′]** can also be done via FFT. The algorithm needs to be adapted only slightly. We will return to this problem in a moment. Let us first concentrate on how to compute $B_1(z)$. After all, `compute_B_power_of_2` assumes that $B_1(z)$ is already available.

We are going to compute the matrix polynomial $B_1(z)$ as the last element in the sequence $B_1^{(0)}(z), B_1^{(1)}(z), \ldots, B_1^{(m-n)}(z)$, where

$$B_1^{(0)}(z) := \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

and

$$B_1^{(j)}(z) := \begin{bmatrix} \prod_{k=1}^{j}(z - s_k) & \beta_j(z) \\ 0 & 1 \end{bmatrix}, \quad j = 1, \ldots, m-n,$$

where $\beta_j(z)$ denotes the interpolating polynomial of degree $j-1$ that satisfies $\beta_j(s_k) = -f_k/e_k$ for $k = 1, \ldots, j$. Clearly, $B_1(z) = B_1^{(m-n)}(z)$, cf. equation (5). One can easily

verify that the matrix polynomials $B_1^{(j)}(z)$, $j = 1, \ldots, m-n$, can be computed in a fast way via the following Schur-type algorithm:

$B_1^{(0)}(z) \leftarrow \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
for $j = 1, \ldots, m-n$ **do**
  $\alpha_L \leftarrow -f_j/e_j$
  for $k = j+1, \ldots, m-n$ **do**
    $[e_k \quad f_k] \leftarrow [e_k(s_k - s_j) \quad e_k\alpha_L + f_k]$
  end for
  $B_1^{(j)}(z) \leftarrow B_1^{(j-1)}(z) \begin{bmatrix} z - s_j & \alpha_L \\ 0 & 1 \end{bmatrix}$
end for

Note that this algorithm corresponds to the Schur-type algorithm that we have presented earlier. However, only left steps are used. Otherwise, we will not obtain the required degree structure.

If $m-n$ is a power of 2, then a divide-and-conquer strategy similar to the one presented above can be used to obtain a superfast algorithm for computing $B_1(z)$.

Let us combine what we have obtained so far. The matrix polynomial $B_1(z)$ is computed as the last element in the sequence $B_1^{(0)}(z), B_1^{(1)}(z), \ldots, B_1^{(m-n)}(z)$. Our algorithm takes into account the first $m-n$ interpolation conditions. If $m-n$ is a power of 2, then the computations can be done in a superfast way. Given $B_1(z)$, the matrix polynomial $B_n(z)$ is computed as the last element in the sequence $B_1(z), B_2(z), \ldots, B_n(z)$. Our algorithm takes into account the remaining $2n$ interpolation conditions. If $2n$ is a power of 2, then the computations can be done in a superfast way. Of course, we do not know if $m-n$ and $2n$ are a power of 2. We only know that $m+n = 2N$ is a power of 2. However, by combining the two sequences, i.e., by computing $B_n(z)$ as the last element in the sequence

$$B_1^{(0)}(z), B_1^{(1)}(z), \ldots, B_1^{(m-n)}(z), B_2(z), \ldots, B_n(z),$$

we can obtain a superfast algorithm. Note that the degree structure of $B_1(z)$ requires that the first $m-n$ steps in this algorithm have to be left steps. The remaining steps can either be a left step followed by a right step or a right step followed by a left step.

The previous considerations lead to the following algorithm. We assume that the interpolation points s_k are the $(m+n)$th roots of a complex number γ of modulus one. In other words, let $\alpha \in [0, 2\pi)$ and $\gamma := \mathrm{e}^{\mathrm{i}\alpha}$. Then we define

$$s_k := \exp\left(\frac{\mathrm{i}\alpha}{m+n}\, k\right), \quad k = 1, \ldots, m+n.$$

In the divide-and-conquer step, we will split the vector s up into its odd and its even part: $\nu := (m+n)/2$ and

$$s_k^{(1)} := s_{2k-1} \quad \text{and} \quad s_k^{(2)} := s_{2k}$$

for $k = 1, \ldots, \nu$. Also, let $s^{(1)} := s_{1:\nu}^{(1)}$ and $s^{(2)} := s_{1:\nu}^{(2)}$.

$B_n(z) \leftarrow$ `compute_B` (s, e, f, m, n)
– We assume that $m \geqslant n$ and that $m+n$ is $\geqslant 2$ and a power of 2.
– $s = [s_k]_{k=1}^{m+n}$ where

$$s_k = \exp\left(\frac{\mathrm{i}\alpha}{m+n}\, k\right) \quad \text{for } k = 1, \ldots, m+n \text{ and } \alpha \in [0, 2\pi).$$

– Is the interpolation problem a polynomial one?
if $n = 0$ **then**
Let $b^{(m)}(z)$ be the interpolating polynomial of degree $< m$ that satisfies $b^{(m)}(s_k) = -f_k/e_k$ for $k = 1, \ldots, m$.

$$B_n(z) \leftarrow \begin{bmatrix} z^m - \mathrm{e}^{\mathrm{i}\alpha} & b^{(m)}(z) \\ 0 & 1 \end{bmatrix}$$

else
– Let τ_{fast} be an integer $\geqslant 1$.
if $m + n \leqslant 2^{\tau_{\text{fast}}}$ **then**
Use a fast method to compute $B_n(z)$.
Keep in mind that the first $m-n$ interpolation conditions can only correspond to left steps while the remaining $2n$ interpolation conditions can correspond to a left step followed by a right step or vice versa.
else
– divide-and-conquer
$\nu \leftarrow (m+n)/2$
$s^{(1)} \leftarrow [s_{2k-1}]_{k=1}^{\nu}$; $s^{(2)} \leftarrow [s_{2k}]_{k=1}^{\nu}$
$e^{(1)} \leftarrow [e_{2k-1}]_{k=1}^{\nu}$; $e^{(2)} \leftarrow [e_{2k}]_{k=1}^{\nu}$
$f^{(1)} \leftarrow [f_{2k-1}]_{k=1}^{\nu}$; $f^{(2)} \leftarrow [f_{2k}]_{k=1}^{\nu}$
$n_1 \leftarrow \max\{0, n - \nu/2\}$; $m_1 \leftarrow \nu - n_1$
$B^{(1)}(z) \leftarrow$ `compute_B` $(s^{(1)}, e^{(1)}, f^{(1)}, m_1, n_1)$
for $k = 1, \ldots, \nu$ **do**
$[e_k^{(2)} \quad f_k^{(2)}] \leftarrow [e_k^{(2)} \quad f_k^{(2)}] B^{(1)}(s_k^{(2)})$
end for
$n_2 \leftarrow n - n_1$; $m_2 \leftarrow m - m_1$
$B^{(2)}(z) \leftarrow$ `compute_B` $(s^{(2)}, e^{(2)}, f^{(2)}, m_2, n_2)$
$B_n(z) \leftarrow B^{(1)}(z) \cdot B^{(2)}(z)$
end if
end if

3.3. Numerical stability

The numerical stability of our superfast algorithm can be enhanced in several ways: via pivoting of the interpolation conditions, via iterative refinement of the computed solutions to the linearized rational interpolation problems, via downdating of interpolation conditions, and by postponing what we have called "difficult interpolation points" until the very end of the algorithm. We have introduced these stabilizing techniques in the context of fast and superfast solvers for linear systems of equations that have Hankel or Toeplitz structure and we refer the reader to [5,8] for more details.

An important difference with the solution of structured linear systems, though, is the fact that we cannot use iterative refinement at the very end of the algorithm to increase the accuracy of the computed determinant. It is an open question if a procedure exists for refining an approximation for a determinant that is similar to the classical iterative refinement procedure that can be used to improve an approximation for the solution of a linear system of equations.

4. Numerical experiments

We have implemented our algorithm in Fortran 90.

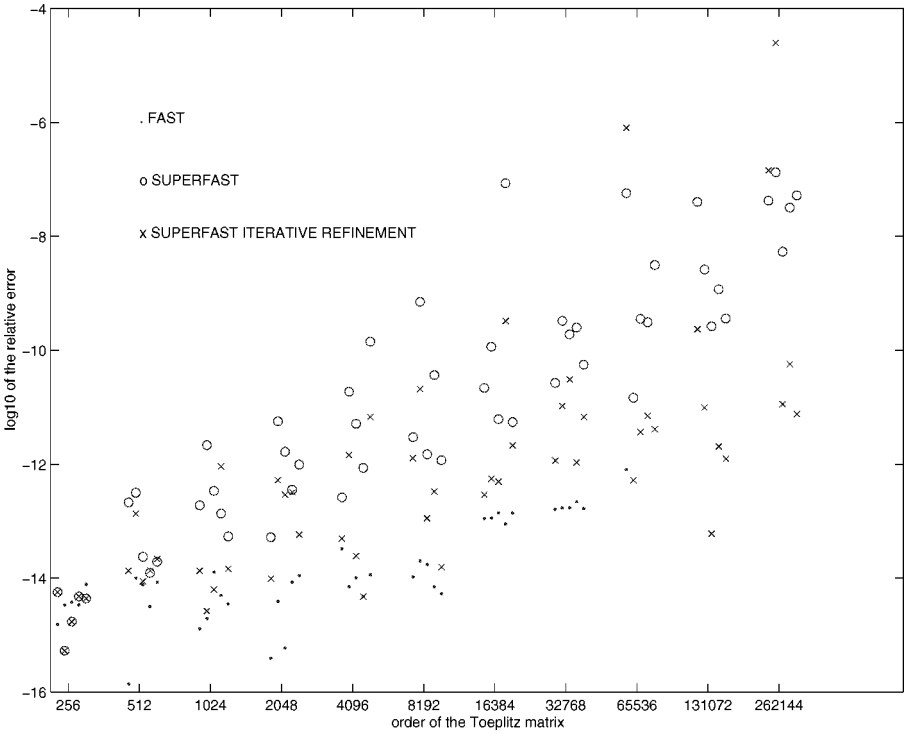

Figure 1. The accuracy of the results obtained by our approach.

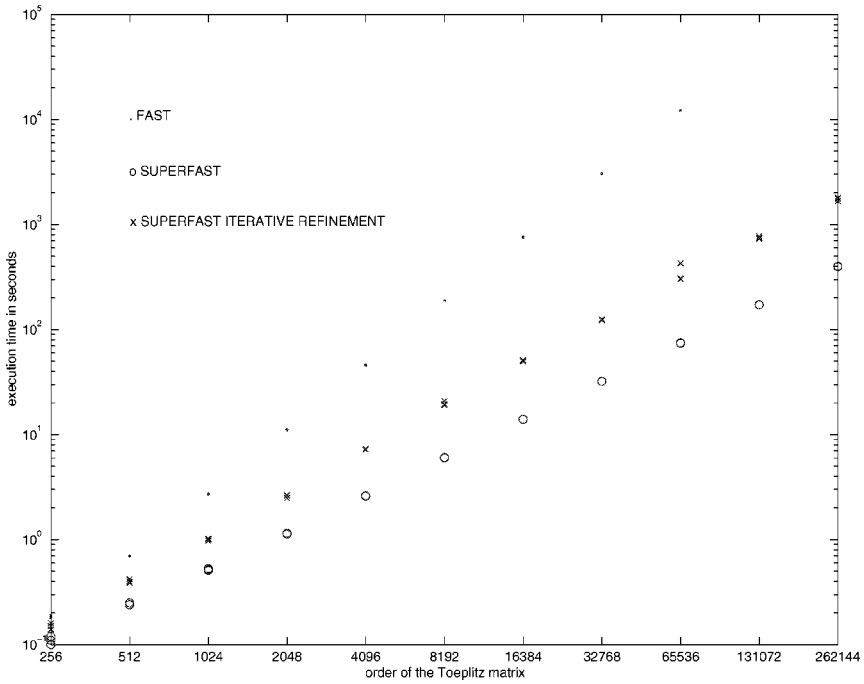

Figure 2. Execution time.

In the following numerical experiments we have considered circulant Toeplitz matrices whose entries are uniformly distributed in $[0, 1]$. It is well known that circulant matrices can be transformed into diagonal matrices via FFT [1]. We have used this to obtain the exact value of the determinant. We let $n = 2^k$, where $k = 8, \ldots, 18$. Note that $2^{18} = 262144$. For each value of n we consider five matrices.

The accuracy of the results obtained by our approach is shown in figure 1. We plot the logarithm with base 10 of the absolute value of the relative error versus the order of the Toeplitz matrix. We show the accuracy obtained by the fast algorithm, by the superfast algorithm in case no iterative refinement is applied, and by the superfast algorithm in case the computed solutions to interpolation problems involving more than 2^8 interpolation conditions are refined iteratively.

In figure 2 we plot the corresponding execution time in seconds versus the order of the Toeplitz matrix.

References

[1] D.A. Bini and V.Y. Pan, *Polynomial and Matrix Computations*, Vol. 1: *Fundamental Algorithms* (Birkhäuser, Basel, 1994).

[2] G. Heinig, Solving Toeplitz systems via extension and transformation, in: *Proc. of the Workshop Toeplitz Matrices: Structure, Algorithms and Applications*, Cortona, Italy (September 9–12, 1996), Calcolo 33 (1996) 115–129.

[3] G. Heinig, Transformation approaches for fast and stable solution of Toeplitz systems and polynomial equations, in: *Proc. of the Internat. Workshop "Recent Advances in Applied Mathematics"*, Kuwait (May 4–7, 1996) pp. 223–238.
[4] G. Heinig and K. Rost, *Algebraic Methods for Toeplitz-like Matrices and Operators*, Operator Theory: Advances and Applications, Vol. 13 (Birkhäuser, Basel, 1984).
[5] P. Kravanja, On computing zeros of analytic functions and related problems in structured numerical linear algebra, Ph.D. thesis, Department of Computer Science, Katholieke Universiteit Leuven (1999).
[6] M. Van Barel, G. Heinig and P. Kravanja, A stabilized superfast solver for nonsymmetric Toeplitz systems, Report TW 293, Department of Computer Science, Katholieke Universiteit Leuven (October 1999).
[7] M. Van Barel and P. Kravanja, On the generically superfast computation of Hankel determinants, in: *Large-Scale Scientific Computations of Engineering and Environmental Problems II*, eds. M. Griebel, S. Margenov and P. Yalamov, Notes on Numerical Fluid Mechanics, Vol. 73 (Vieweg, 2000) pp. 57–64; *Proc. of the 2nd Workshop on Large-Scale Scientific Computations*, Sozopol, Bulgaria (June 2–6, 1999).
[8] M. Van Barel and P. Kravanja, A stabilized superfast solver for indefinite Hankel systems, in: *Internat. Linear Algebra Society Symposium "Linear Algebra in Control Theory, Signals and Image Processing*", University of Manitoba, Canada (6–8 June 1997), special issue of Linear Algebra Appl. 284(1–3) (1998) 335–355.