

A stabilized superfast solver for nonsymmetric Toeplitz systems

M. Van Barel, G. Heinig and P. Kravanja

Report TW 293, October 1999



Katholieke Universiteit Leuven
Department of Computer Science
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

A stabilized superfast solver for nonsymmetric Toeplitz systems

M. Van Barel, G. Heinig and P. Kravanja

Report TW293, October 1999

Department of Computer Science, K.U.Leuven

Abstract

We present a stabilized superfast solver for nonsymmetric Toeplitz systems $Tx = b$. An explicit formula for T^{-1} is expressed in such a way that the matrix-vector product $T^{-1}b$ can be calculated via FFTs and Hadamard products. This inversion formula involves certain polynomials that can be computed by solving two linearized rational interpolation problems on the unit circle. The heart of our Toeplitz solver is a superfast algorithm to solve these interpolation problems. This algorithm is stabilized via pivoting, iterative improvement, downdating, and by giving “difficult” interpolation points an adequate treatment. We have implemented our algorithm in Fortran 90. Numerical examples illustrate the effectiveness of our approach.

Keywords : nonsymmetric Toeplitz systems, stabilized superfast algorithm, rational interpolation, pivoting, iterative improvement, downdating.

AMS(MOS) Classification : Primary : 65F05 Secondary : 47B35, 15A09

A STABILIZED SUPERFAST SOLVER FOR NONSYMMETRIC TOEPLITZ SYSTEMS*

MARC VAN BAREL[†], GEORG HEINIG[‡], AND PETER KRAVANJA[§]

Abstract. We present a stabilized superfast solver for nonsymmetric Toeplitz systems $Tx = b$. An explicit formula for T^{-1} is expressed in such a way that the matrix-vector product $T^{-1}b$ can be calculated via FFTs and Hadamard products. This inversion formula involves certain polynomials that can be computed by solving two linearized rational interpolation problems on the unit circle. The heart of our Toeplitz solver is a superfast algorithm to solve these interpolation problems. This algorithm is stabilized via pivoting, iterative improvement, downdating, and by giving “difficult” interpolation points an adequate treatment. We have implemented our algorithm in Fortran 90. Numerical examples illustrate the effectiveness of our approach.

Key words. nonsymmetric Toeplitz systems, stabilized superfast algorithm, inversion formula, rational interpolation, pivoting, iterative improvement, downdating.

AMS subject classifications. Primary 65F05; Secondary 47B35, 15A09

1. Introduction. Subject of the paper are linear systems of equations

$$(1.1) \quad T_n x = b$$

with a nonsingular, in general nonsymmetric, $n \times n$ Toeplitz coefficient matrix $T := T_n := [a_{k-l}]_{k,l=0}^{n-1}$ and a right-hand side vector $b \in \mathbb{C}^n$. The aim of the paper is to present a new solution algorithm that has (generically) complexity $\mathcal{O}(n \log^2 n)$ and that avoids instable behaviour.

Since Toeplitz systems of equations appear in many applications, they are the subject of a huge number of publications. During the last decades many algorithms have been developed that exploit the Toeplitz structure of the coefficient matrix. There are two types of direct fast solvers that require $\mathcal{O}(n^2)$ operations: Levinson-type and Schur-type algorithms. For more references and information concerning the history of these algorithms, we refer the reader to the survey paper [35].

The flow of these classical fast methods is determined by the exact singularity of the leading principal submatrices of T . The fast methods compute the solutions corresponding to successive nonsingular leading principal submatrices (sections). However, in finite-precision arithmetic not only singular but also ill-conditioned sections should be avoided. In the case of a positive definite matrix, it can be guaranteed that the principal submatrices are well-conditioned. Moreover, it was proved in [5] that the Schur (Bareiss) algorithm is weakly stable in this case. Weak stability of the Levinson algorithm for a certain class of positive definite Toeplitz matrices was already proven in [12]. In the case of an indefinite and nonsymmetric matrix, simple examples show that both the Levinson and the Schur algorithm can be highly unstable.

*This research was partially supported by the Fund for Scientific Research–Flanders (FWO–V), project “Orthogonal systems and their applications,” grant #G.0278.97 and by Kuwait University Research Project SM–190.

[†]Katholieke Universiteit Leuven, Department of Computer Science, Celestijnenlaan 200 A, B-3001 Heverlee, Belgium (Marc.VanBarel@cs.kuleuven.ac.be)

[‡]Kuwait University, Department of Mathematics, POB 5969, Safat 13060, Kuwait (georg@math-1.sci.kuniv.edu.kw)

[§]Katholieke Universiteit Leuven, Department of Computer Science, Celestijnenlaan 200 A, B-3001 Heverlee, Belgium (Peter.Kravanja@na-net.ornl.gov)

A first idea to overcome instable behavior of the classical algorithms is to “look ahead” from one well-conditioned section to the next one and jump over the ill-conditioned sections that lie in between. For Toeplitz systems several look-ahead algorithms were designed in [10, 11, 14, 15, 17, 22, 24, 25, 26, 27, 34]. For Hankel systems we refer to [6, 9, 16]. Several high performance algorithms for Toeplitz matrices are described in [18], including two look-ahead Schur algorithms for symmetric indefinite block Toeplitz matrices. However, reliable look-ahead strategies are difficult to design and, moreover, the resulting algorithms may have complexity $O(n^3)$, i.e., they may not be fast.

Another approach to deal with nonsymmetric Toeplitz systems, which is, in principle, also the approach used in this paper, was first proposed in [28] and further developed in [18, 20, 23, 29, 31, 32, 36, 37, 46] and other papers. A survey of the matter is given in [43]. The idea is to transform the Toeplitz (or Hankel or Toeplitz-plus-Hankel) matrix with the help of the DFT or real discrete trigonometric transforms into a generalized Cauchy or a generalized Vandermonde matrix, which can be done with $O(n \log n)$ complexity in a stable way. The advantage of these classes of matrices is that permutation of rows or columns does not destroy the structure. Therefore pivoting strategies can be applied to stabilize the algorithm. Slightly different is the idea, which was proposed in [29, 30], to transform the Toeplitz (or Hankel) system directly into a rational interpolation problem at roots of unity. One advantage of this approach is that one can guarantee that the length of the transformations is a power of 2.

Algorithms with complexity less than $O(n^2)$ are called *superfast*. Superfast solvers are based on divide and conquer strategies. The idea for a superfast Toeplitz solver was first announced in the PhD Thesis of M. Morf [40]. Superfast algorithms were designed by Sugiyama et al. [45], Bitmead and Anderson [4], Brent, Gustavson and Yun [7] and Morf [41]. More recent algorithms can be found in [1, 2, 3, 8, 13, 19, 21, 38, 39, 42, 44].

The main disadvantage of these algorithms is that they cannot handle nearly singular leading principal submatrices and are therefore unstable in the nonsymmetric case. To overcome this problem, Gutknecht [25] and Gutknecht and Hochbruck [26, 27] developed an algorithm that combines the look-ahead idea with divide and conquer techniques. However, the implementation of this algorithm is connected with some practical problems.

In [49] a divide and conquer approach was used to obtain a superfast algorithm for rational interpolation at roots of unity. The algorithm consists of two stages. The first part, for a small number of nodes, consists in the fast algorithm from [37] including pivoting. The second part is a doubling procedure. Instead of pivoting, “difficult” points are sorted out and are treated at the end. Also, iterative refinement is applied to stabilize the algorithm.

In this paper we use a similar approach for Toeplitz systems. In contrast with [49], the Toeplitz is not first transformed into another matrix but an explicit formula for the inverse of a Toeplitz matrix is used. This formula involves the values of the “fundamental system” at roots of unity, i.e., the DFT of the fundamental system. It will be presented in Section 2. The fundamental system is a pair of polynomials containing all the information about the Toeplitz matrix. These polynomials are related to two linearized rational interpolation problems at roots of unity. This connection will be explained in Section 3. To solve these interpolation problems, we extend the superfast algorithm presented in [49]. We will incorporate “downdating” into this algorithm as

an additional stabilizing technique. In Section 5 we will present numerical examples and compare our results with those obtained in [49]. Note that in [49] the size of the Hankel system is limited to a power of 2, whereas we can handle Toeplitz systems of arbitrary size.

2. DFT representation of Toeplitz matrix inverses. Let us introduce the following notations. To each column vector $u = [u_k]_{k=0}^n \in \mathbb{C}^{n+1}$ we associate the polynomial $u(z) := \sum_{k=0}^n u_k z^k \in \mathbb{C}[z]$. The column vector \hat{u} is defined as $\hat{u} := [u_{n-k}]_{k=0}^n$. Thus, $\hat{u}(z) = z^n u(z^{-1})$.

Let $a_{-n} \in \mathbb{C}$ be arbitrary and let $\tilde{T} = \tilde{T}_n$ be given by the $n \times (n+1)$ matrix

$$\tilde{T} := [a_{k-l}]_{k,l=0}^{n-1,n} = \begin{bmatrix} a_0 & a_{-1} & \cdots & a_{-n+1} \\ a_1 & a_0 & \ddots & \\ \vdots & & \ddots & \\ a_{n-1} & \cdots & \cdots & a_0 \end{bmatrix} \begin{bmatrix} a_{-n} \\ a_{-n+1} \\ \vdots \\ a_{-1} \end{bmatrix} = \begin{bmatrix} & & & a_{-n} \\ & & & a_{-n+1} \\ & & & \vdots \\ & & & a_{-1} \\ T & & & \end{bmatrix}.$$

The polynomials $u(z)$ and $v(z)$ are called the *canonical fundamental system* of T if

- $\tilde{T}u = e_1$ and $u_n = 0$, where $e_1 := [1 \ 0 \ \cdots \ 0]^T$,
- $\tilde{T}v = 0$ and $v_n = 1$.

In other words, $u(z)$ is a polynomial of degree $n-1$ at most while $v(z)$ is a monic polynomial of degree n such that

$$\begin{bmatrix} a_0 & a_{-1} & \cdots & a_{-n+1} \\ a_1 & a_0 & \ddots & \\ \vdots & & \ddots & a_{-1} \\ a_{n-1} & \cdots & \cdots & a_0 \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{n-1} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

and

$$\begin{bmatrix} a_0 & a_{-1} & \cdots & a_{-n+1} \\ a_1 & a_0 & \ddots & \\ \vdots & & \ddots & a_{-1} \\ a_{n-1} & \cdots & \cdots & a_0 \end{bmatrix} \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_{n-1} \end{bmatrix} = - \begin{bmatrix} a_{-n} \\ a_{-n+1} \\ \vdots \\ a_{-1} \end{bmatrix}.$$

As T is assumed to be nonsingular, these polynomials do exist. Moreover, $u(z)$ is unique whereas $v(z)$ is uniquely determined given a particular value of a_{-n} . For our purposes the specific value of a_{-n} is immaterial and thus $v(z)$ is in fact unique up to a linear combination with $u(z)$. Note that by cancelling the last (zero) component of u , one obtains the first column of T^{-1} .

Remark. Let $w = [w_0 \ \cdots \ w_{n-1}]^T \in \mathbb{C}^n$ be the last column of T_n^{-1} . If $T_{n-1} := [a_{k-l}]_{k,l=0}^{n-2}$ is also nonsingular, then Cramer's rule implies that $w_{n-1} \neq 0$ and one may choose $v(z)$ as $v(z) = zw(z)/w_{n-1}$. This choice determines the value of a_{-n} . Also, if T_n is symmetric, then $w(z) = \hat{u}(z)$.

The *generating function* $M(t, s)$ of a matrix $M = [m_{k,l}]_{k,l=0}^{p,q}$ is defined as

$$M(t, s) := \sum_{k=0}^p \sum_{l=0}^q m_{k,l} t^k s^l.$$

THEOREM 2.1. *The generating function of T^{-1} is given by*

$$(2.1) \quad T^{-1}(t, s) = \frac{u(t)\hat{v}(s) - v(t)\hat{u}(s)}{1 - ts}.$$

Proof. See [33, p. 32]. \square

The matrix whose generating function is given by the right-hand side of (2.1) is called the *Toeplitz Bezoutian* of the polynomials $u(z)$ and $v(z)$.

Let $N \geq n$ be a power of 2. From the previous theorem we will now derive a formula for T^{-1} that will enable us to calculate the matrix-vector product $T^{-1}b$ in $\mathcal{O}(N \log N)$ floating point operations.

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

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

and let $\omega_k^+ := \omega_{2k}$ and $\omega_k^- := \omega_{2k+1}$ for $k = 0, 1, \dots, N-1$. Note that $\omega_0^+, \dots, \omega_{N-1}^+$ are the N th roots of unity whereas $\omega_0^-, \dots, \omega_{N-1}^-$ are the N th roots of -1 . Let $\eta := \exp(\pi i/N)$. Define the matrices \mathcal{F}_+ and \mathcal{F}_- as

$$\mathcal{F}_+ := [(\omega_k^+)^l]_{k,l=0}^{N-1} \quad \text{and} \quad \mathcal{F}_- := [(\omega_k^-)^l]_{k,l=0}^{N-1}.$$

Then \mathcal{F}_+/\sqrt{N} is unitary and $\mathcal{F}_- = \mathcal{F}_+ \text{diag}(1, \eta, \dots, \eta^{N-1})$. Matrix-vector products involving \mathcal{F}_+ or \mathcal{F}_- can be evaluated with arithmetic complexity $\mathcal{O}(N \log N)$ via the celebrated FFT.

Let $[T^{-1}]_N$ denote the $N \times N$ matrix

$$[T^{-1}]_N := \begin{bmatrix} T^{-1} & 0 \\ 0 & 0 \end{bmatrix} \in \mathbb{C}^{N \times N}.$$

Define $D := \text{diag}((\omega_0^+)^{-n}, \dots, (\omega_{N-1}^+)^{-n})$, $D_{\pm}(u) := \text{diag}(u(\omega_0^{\pm}), \dots, u(\omega_{N-1}^{\pm}))$ and similar for the matrices $D_{\pm}(v)$.

THEOREM 2.2. *The matrix $[T^{-1}]_N$ can be expressed as*

$$(2.2) \quad [T^{-1}]_N = \frac{1}{2} \mathcal{F}_-^{-1} [D_-(u) \mathcal{F}_- \mathcal{F}_+^{-1} D_+(v) - D_-(v) \mathcal{F}_- \mathcal{F}_+^{-1} D_+(u)] D \mathcal{F}_+.$$

Proof. As $\mathcal{F}_+^{-1} = \frac{1}{N} \mathcal{F}_+^H$, it follows that

$$\mathcal{F}_- [T^{-1}]_N \mathcal{F}_+^{-1} =: [c_{k,l}]_{k,l=0}^{N-1}$$

where

$$\begin{aligned} c_{k,l} &= \frac{1}{N} T^{-1}(\omega_k^-, 1/\omega_l^+) \\ &= \frac{1}{N} \frac{u(\omega_k^-)\hat{v}(1/\omega_l^+) - v(\omega_k^-)\hat{u}(1/\omega_l^+)}{1 - \omega_k^-/\omega_l^+} \\ &= \frac{1}{N} \frac{u(\omega_k^-)v(\omega_l^+) - v(\omega_k^-)u(\omega_l^+)}{1 - \omega_k^-/\omega_l^+} [\omega_l^+]^{-n}. \end{aligned}$$

One can easily verify that

$$\mathcal{F}_- \mathcal{F}_+^{-1} = \frac{2}{N} \left[\frac{1}{1 - \omega_k^- / \omega_l^+} \right]_{k,l=0}^{N-1}.$$

The expression for $[T^{-1}]_N$ then follows immediately. \square

The formula in the previous theorem allows us to compute the product $x = T^{-1}b$ in $\mathcal{O}(N \log N)$ flops provided that the polynomials $u(z)$ and $v(z)$ are known. Indeed,

$$\begin{bmatrix} x \\ 0 \end{bmatrix} = [T^{-1}]_N \begin{bmatrix} b \\ 0 \end{bmatrix}$$

and the multiplication by $[T^{-1}]_N$ can be done via six N -point (inverse) FFTs and $\mathcal{O}(N)$ flops. For preprocessing one has to compute the values of $u(\omega_k)$ and $v(\omega_k)$ for $k = 0, 1, \dots, 2N - 1$, which amounts to two $2N$ -points FFTs.

3. Interpolation interpretation. For a given Toeplitz matrix $T = [a_{k-l}]_{k,l=0}^{n-1}$, we introduce the function

$$a(z) := \sum_{k=1-n}^{n-1} a_k z^k,$$

where z is considered as a complex variable.

THEOREM 3.1. *Suppose $a_{-n} = 0$. Then the polynomials $u(z)$ and $v(z)$ are the canonical fundamental system of T if and only if there exist polynomials $\hat{r}_u(z)$ and $\hat{r}_v(z)$ satisfying the linearized rational interpolation conditions*

$$\omega_k^n \hat{r}_u(\omega_k) - a(\omega_k)u(\omega_k) = 0, \quad k = 0, 1, \dots, 2N - 1,$$

where $\deg \hat{r}_u(z) \leq 2N - n$, $\hat{r}_{u,2N-n} = 1$ and $\deg u(z) \leq n$, $u_n = 0$ and

$$\omega_k^n \hat{r}_v(\omega_k) - a(\omega_k)v(\omega_k) = 0, \quad k = 0, 1, \dots, 2N - 1,$$

where $\deg \hat{r}_v(z) \leq 2N - n$, $\hat{r}_{v,2N-n} = 0$ and $\deg v(z) \leq n$, $v_n = 1$.

Proof. Let $\delta \in \mathbb{C}$. Let $w(z)$ be a polynomial of degree $\leq n$ such that $\tilde{T}w = \delta e_1$. The case $\delta = 1$ corresponds to $w(z) = u(z)$ whereas $\delta = 0$ corresponds to $w(z) = v(z)$. The condition $\tilde{T}w = \delta e_1$ is equivalent to the existence of polynomials $r_-(z)$ and $r_+(z)$ of degree $\leq n - 1$ with $r_{-,0} = \delta$ such that

$$a(z)w(z) = r_-(1/z) + z^n r_+(z).$$

It follows that

$$\begin{aligned} a(\omega_k)w(\omega_k) &= r_-(\omega_k^{-1}) + \omega_k^n r_+(\omega_k) \\ &= r_-(\omega_k^{-1}) + \omega_k^{n-2N} r_+(\omega_k) \end{aligned}$$

for $k = 0, 1, \dots, 2N - 1$. Define

$$r(z) := r_-(z) + z^{2N-n} r_+(1/z).$$

Then $r(z)$ is a polynomial of degree $\leq 2N - n$ and $r_0 = \delta$. Thus

$$\begin{aligned} a(\omega_k)w(\omega_k) &= r(\omega_k^{-1}) \\ &= \omega_k^n [\omega_k^{2N-n} r(\omega_k^{-1})] \\ &= \omega_k^n \hat{r}(\omega_k) \end{aligned}$$

for $k = 0, 1, \dots, 2N - 1$. In other words,

$$\omega_k^n \hat{r}(\omega_k) - a(\omega_k)w(\omega_k) = 0, \quad k = 0, 1, \dots, 2N - 1,$$

where $\deg \hat{r}(z) \leq 2N - n$, $\hat{r}_{2N-n} = \delta$ and $\deg w(z) \leq n$. \square

These interpolation conditions can also be written as follows:

$$\begin{bmatrix} \omega_k^n & -a(\omega_k) \end{bmatrix} B^*(\omega_k) = \begin{bmatrix} 0 & 0 \end{bmatrix}, \quad k = 0, 1, \dots, 2N - 1,$$

where

$$B^*(z) := \begin{bmatrix} \hat{r}_u(z) & \hat{r}_v(z) \\ u(z) & v(z) \end{bmatrix} \in \mathbb{C}[z]^{2 \times 2}$$

is a 2×2 matrix polynomial. The degree of the first row of $B^*(z)$ is equal to $2N - n$ whereas the degree of the second row of $B^*(z)$ is equal to n . The second row of $B^*(z)$ gives us the inversion parameters that are needed in formula (2.2).

4. A stabilized divide and conquer approach. Define

$$f_k := \begin{bmatrix} \omega_k^n \\ -a(\omega_k) \end{bmatrix} \in \mathbb{C}^{2 \times 1}$$

for $k = 0, 1, \dots, 2N - 1$ and let \mathcal{S} be the set of all the column vector polynomials $w(z) \in \mathbb{C}[z]^{2 \times 1}$ that satisfy the interpolation conditions

$$(4.1) \quad f_k^T w(\omega_k) = 0, \quad k = 0, 1, \dots, 2N - 1.$$

If $w(z) \in \mathbb{C}[z]^{2 \times 1}$ is an arbitrary vector polynomial, then the left-hand side of (4.1) is called the *residual* with respect to $w(z)$ at the interpolation point ω_k .

The set \mathcal{S} forms a submodule of the $\mathbb{C}[z]$ -module $\mathbb{C}[z]^{2 \times 1}$. A basis for \mathcal{S} always consists of exactly two elements [47, Theorem 3.1]. Let $\{B_1(z), B_2(z)\}$ be a basis for \mathcal{S} . Then every element $w(z) \in \mathcal{S}$ can be written in a unique way as $w(z) = \alpha_1(z)B_1(z) + \alpha_2(z)B_2(z)$ with $\alpha_1(z), \alpha_2(z) \in \mathbb{C}[z]$. The matrix polynomial $B(z) := [B_1(z) \ B_2(z)] \in \mathbb{C}[z]^{2 \times 2}$ is called a *basis matrix*. Basis matrices can be characterized as follows.

THEOREM 4.1. *A matrix polynomial $C(z) = [C_1(z) \ C_2(z)] \in \mathbb{C}[z]^{2 \times 2}$ is a basis matrix if and only if $C_1(z), C_2(z) \in \mathcal{S}$ and $\deg \det C(z) = 2N$.*

Proof. This follows immediately from [47, Theorem 4.1]. \square

Note that $B^*(z)$ is a basis matrix.

Within the submodule \mathcal{S} we want to be able to consider solutions $w(z)$ that satisfy additional conditions concerning their degree-structure. To describe the degree-structure of a vector polynomial, we use the concept of τ -degree [47]. Let $\tau \in \mathbb{Z}$. The τ -degree of a vector polynomial $w(z) = [w_1(z) \ w_2(z)]^T \in \mathbb{C}[z]^{2 \times 1}$ is defined as a generalization of the classical degree:

$$\tau\text{-deg } w(z) := \max\{\deg w_1(z), \deg w_2(z) - \tau\}$$

with $\tau\text{-deg } 0 := -\infty$. The τ -highest degree coefficient of a vector polynomial

$$[w_1(z) \ w_2(z)]^T$$

with τ -degree δ is defined as the vector $[w_1 \ w_2]^T$ with w_1 the coefficient of z^δ in $w_1(z)$ and w_2 the coefficient of $z^{\delta+\tau}$ in $w_2(z)$. A set of vector polynomials in $\mathbb{C}[z]^{2 \times 1}$ is

called τ -reduced if the τ -highest degree coefficients are linearly independent. Every basis of \mathcal{S} can be transformed into a τ -reduced one. For details, we refer to [47]. Once we have a basis in τ -reduced form, the elements of \mathcal{S} can be parametrized as follows.

THEOREM 4.2. *Let $\{B_1(z), B_2(z)\}$ be a τ -reduced basis for \mathcal{S} . Define $\delta_1 := \tau\text{-deg } B_1(z)$ and $\delta_2 := \tau\text{-deg } B_2(z)$. Then every element $w(z) \in \mathcal{S}$ having τ -degree $\leq \delta$ can be written in a unique way as*

$$w(z) = \alpha_1(z)B_1(z) + \alpha_2(z)B_2(z)$$

with $\alpha_1(z), \alpha_2(z) \in \mathbb{C}[z]$, $\deg \alpha_1(z) \leq \delta - \delta_1$ and $\deg \alpha_2(z) \leq \delta - \delta_2$.

Proof. See Van Barel and Bultheel [47, Theorem 3.2]. \square

We can now summarize our aim as follows: we want to design an algorithm for computing the 2×2 matrix polynomial $B^*(z)$ as a τ -reduced basis matrix that corresponds to the interpolation data (ω_k, f_k) , $k = 0, 1, \dots, 2N - 1$, where we set $\tau := 2(n - N)$.

The following theorem will enable us to devise an interpolation algorithm that is based on a *divide and conquer* approach. It shows how basis matrices can be coupled in case the degree-structure is important.

THEOREM 4.3. *Suppose K is a positive integer. Let $\sigma_1, \dots, \sigma_K \in \mathbb{C}$ be mutually distinct and let $\phi_1, \dots, \phi_K \in \mathbb{C}^{2 \times 1}$. Suppose that $\phi_k \neq [0 \ 0]^T$ for $k = 1, \dots, K$. Let $1 \leq \kappa \leq K$. Let $\tau_K \in \mathbb{Z}$. Suppose that $B_\kappa(z) \in \mathbb{C}[z]^{2 \times 2}$ is a τ_K -reduced basis matrix with basis vectors having τ_K -degree δ_1 and δ_2 respectively, corresponding to the interpolation data*

$$\{(\sigma_k, \phi_k) : k = 1, \dots, \kappa\}.$$

Let $\tau_{\kappa \rightarrow K} := \delta_1 - \delta_2$. Let $B_{\kappa \rightarrow K}(z) \in \mathbb{C}[z]^{2 \times 2}$ be a $\tau_{\kappa \rightarrow K}$ -reduced basis matrix corresponding to the interpolation data

$$\{(\sigma_k, B_\kappa^T(\sigma_k)\phi_k) : k = \kappa + 1, \dots, K\}.$$

Then $B_K(z) := B_\kappa(z)B_{\kappa \rightarrow K}(z)$ is a τ_K -reduced basis matrix corresponding to the interpolation data

$$\{(\sigma_k, \phi_k) : k = 1, \dots, K\}.$$

Proof. See Van Barel and Bultheel [48, Theorem 3]. \square

The following algorithm implements this theorem. We start with the $2N$ th roots of unity as interpolation points. They are split into the N th roots of unity s_1 and the rotated N th roots of unity s_2 . The fact that we are dealing with (rotated) roots of unity enables us to do all polynomial evaluations and multiplications via FFTs (and diagonal scalings). As N is a power of 2, this process can be repeated. At the lowest level the interpolation problems are solved by the fast solver RATINT developed by Kravanja and Van Barel [37].

recursive function $[B(z), s_{\text{bad}}] \leftarrow \text{RECRATINT}(s, L_s, R_s, N_s, \tau)$

-- $\tau \in \mathbb{Z}$

-- $N_s = 2^{p+1}$ for some $p \in \mathbb{N}$: the number of interpolation conditions

-- $s \in \mathbb{C}^{N_s \times 1}$: the (mutually distinct) interpolation points

-- $L_s, R_s \in \mathbb{C}^{N_s \times 1}$: the initial left and right residual vectors

-- $B(z) \in \mathbb{C}[z]^{2 \times 2}$: a τ -reduced basis matrix corresponding to

```

the given interpolation data
--  $s_{\text{bad}}$ : a complex column vector containing the difficult
interpolation points
if  $N_s > 2^{\text{limit}}$  then
   $[s_1, L_{s_1}, R_{s_1}, s_2, L_{s_2}, R_{s_2}] \leftarrow \text{SPLIT}(s, L_s, R_s)$ 
   $[B_1(z), s_{\text{bad},1}] \leftarrow \text{RECRATINT}(s_1, L_{s_1}, R_{s_1}, N_s/2, \tau)$ 
  for  $k = 1(1)N_s/2$ 
     $[\tilde{L}_{s_2}(k) \ \tilde{R}_{s_2}(k)] \leftarrow [L_{s_2}(k) \ R_{s_2}(k)] \cdot B_1(s_2(k))$ 
  end for
   $\tilde{\tau} \leftarrow$  the difference between the left and right  $\tau$ -degrees of  $B_1(z)$ 
   $[\tilde{B}_2(z), \tilde{s}_{\text{bad},2}] \leftarrow \text{RECRATINT}(s_2, \tilde{L}_{s_2}, \tilde{R}_{s_2}, N_s/2, \tilde{\tau})$ 
   $B(z) \leftarrow B_1(z) \cdot \tilde{B}_2(z)$ 
   $s_{\text{bad}} \leftarrow s_{\text{bad},1} \oplus \tilde{s}_{\text{bad},2}$ 
else
   $[B(z), s_{\text{bad}}] \leftarrow \text{RATINT}(s, L_s, R_s, N_s, \tau)$ 
end if
if  $N_s = 2^{\text{downdating}}$  then
   $s^+ \leftarrow s \ominus s_{\text{bad}}$ 
   $[B(z), s_{\text{bad},3}] \leftarrow \text{DOWNDATING}(s^+, L_{s^+}, R_{s^+}, N_s)$ 
   $s_{\text{bad}} \leftarrow s_{\text{bad}} \oplus s_{\text{bad},3}$ 
end if
if  $N_s = 2^{\text{reflimit}}$  then
   $s^+ \leftarrow s \ominus s_{\text{bad}}$ 
   $[B(z)] \leftarrow \text{ITREF}(B(z), s^+, L_{s^+}, R_{s^+}, N_s, N_{\text{ref}})$ 
end if
return
function  $[B(z)] \leftarrow \text{RATINTALL}(s, L_s, R_s, N_s, \tau)$ 
--  $\tau \in \mathbb{Z}$ 
--  $N_s = 2^{p+1}$  for some  $p \in \mathbb{N}$ : the number of interpolation conditions
--  $s \in \mathbb{C}^{N_s \times 1}$ : the (mutually distinct) interpolation points
--  $L_s, R_s \in \mathbb{C}^{N_s \times 1}$ : the initial left and right residual vectors
--  $B(z) \in \mathbb{C}[z]^{2 \times 2}$ : a  $\tau$ -reduced basis matrix corresponding to
the given interpolation data
 $[B^+(z), s_{\text{bad}}] \leftarrow \text{RECRATINT}(s, L_s, R_s, N_s, \tau)$ 
 $N_{\text{bad}} \leftarrow \text{SIZE}(s_{\text{bad}})$ 
if  $N_{\text{bad}} > 0$  then
  calculate  $L_{\text{bad}}$  and  $R_{\text{bad}}$ 
   $\tau^- \leftarrow$  the difference between the left and right  $\tau$ -degrees of  $B^+(z)$ 
   $[B^-(z)] \leftarrow \text{RATINT}(s_{\text{bad}}, L_{\text{bad}}, R_{\text{bad}}, N_{\text{bad}}, \tau^-)$ 
   $B(z) \leftarrow B^+(z) \cdot B^-(z)$ 
end if
return

```

Superfast Hankel (Toeplitz) solvers are notoriously unstable when applied to indefinite systems. Algorithm `RECRATINT` is stabilized in three ways.

Difficult points. During the execution of `RATINT` all the residuals at interpolation points that may be chosen as pivot elements can be smaller (in modulus) than a certain threshold. By processing these interpolation points the accuracy would decrease. These points are therefore marked as “difficult.” They are handled only at the very

end, after RECRATINT has finished, via the fast-only algorithm RATINT. If at this stage the corresponding transformed residuals are still small, this indicates that the problem is ill-conditioned. The overall complexity of our algorithm will be $\mathcal{O}(n \log^2 n)$ as long as the number of difficult points is not too large.

Iterative improvement. The approximations for the coefficients of the polynomials that appear in the basis matrix $B(z)$ can be updated iteratively by using an inversion formula for coupled Vandermonde matrices. For more details, we refer to Van Barel and Kravanja [49]. Iterative refinement can be applied at one or more intermediate levels of the divide and conquer process. In algorithm RECRATINT, it is used only if the number of interpolation conditions is equal to 2^{reffimit} .

Downdating. Finite precision arithmetic can lead to a situation where

$$f_k^T B(s_k) \not\approx [0 \ 0]$$

for one or more interpolation points s_k . As the matrix $B(s_k)$ is singular, there exists a vector $v \in \mathbb{C}^2$ such that

$$B(s_k)v = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

Define

$$B(z) =: [B_L(z) \ B_R(z)], \quad v =: \begin{bmatrix} v_L \\ v_R \end{bmatrix}$$

and let $\alpha_L := \tau\text{-deg}B_L(z)$ and $\alpha_R := \tau\text{-deg}B_R(z)$. If $\alpha_L \geq \alpha_R$ and $v_L \neq 0$, then we replace $B_L(z)$ by

$$B_L(z) \leftarrow B(z)v/(z - s_k).$$

If, on the other hand, $\alpha_L < \alpha_R$ and $v_R \neq 0$, then we replace $B_R(z)$ by

$$B_R(z) \leftarrow B(z)v/(z - s_k).$$

If $v_L = 0$, then $B_R(z)$ is divisible by $z - s_k$. Similarly, if $v_R = 0$, then $B_L(z)$ is divisible by $z - s_k$. These considerations lead to the following algorithm.

function $[B(z), s_{\text{bad}}] \leftarrow \text{DOWNDATING}(B(z), s, L_s, R_s, N_s)$

-- N_s : the number of interpolation conditions

-- $s \in \mathbb{C}^{N_s \times 1}$: the (mutually distinct) interpolation points

-- $L_s, R_s \in \mathbb{C}^{N_s \times 1}$: the initial left and right residual vectors

-- $B(z) \in \mathbb{C}[z]^{2 \times 2}$

-- on input: a basis matrix corresponding to the given interpolation data

-- on output: the corresponding downdated basis matrix

-- s_{bad} : a complex column vector containing the interpolation points that have been downdated

$s_{\text{bad}} \leftarrow \emptyset$

for $k = 1(1)N_s$

if $\| [L_s(k) \ R_s(k)] \| > \eta$ **then**

Choose $v \in \mathbb{C}^2$ such that $B(s(k))v = [0 \ 0]^T$ and $\|v\| = 1$

-- Let $B(z) =: [B_L(z) \ B_R(z)]$ and $v =: [v_L \ v_R]^T$

$\alpha_L \leftarrow \tau\text{-deg}B_L(z)$

$\alpha_R \leftarrow \tau\text{-deg}B_R(z)$

```

if  $\alpha_L \geq \alpha_R$  then
  if  $v_L \neq 0$  then
     $B_L(z) \leftarrow B(z)v/(z - s(k))$ 
  else
     $B_R(z) \leftarrow B_R(z)/(z - s(k))$ 
  end if
else
  if  $v_R \neq 0$  then
     $B_R(z) \leftarrow B(z)v/(z - s(k))$ 
  else
     $B_L(z) \leftarrow B_L(z)/(z - s(k))$ 
  end if
end if
 $s_{\text{bad}} \leftarrow s_{\text{bad}} \oplus s(k)$ 
end if
end for
return

```

5. Numerical experiments. We consider double precision Toeplitz matrices T_n whose entries are real and random uniformly distributed in $[0, 1]$ with $n = 2^k$ for $k = 1, \dots, 18$. Note that $2^{18} = 262144$. The right-hand sides $b_n \in \mathbb{R}^n$ are calculated such that $x_n := T_n^{-1}b_n = [1 \ \dots \ 1]^T$. The calculations were done by an IBM SP2 with machine precision $\approx 0.22 \cdot 10^{-15}$ in double precision. Our software is available at

<http://www.cs.kuleuven.ac.be/~marc/hankel/>

Figures 5.1 and 5.2 show the results obtained by our algorithm in case no iterative refinement is applied (the symbols ‘+’) and in case up to 10 steps of iterative refinement are applied (the symbols ‘o’) to enhance the accuracy of the computed solution to the Toeplitz system. Interpolation problems of size less than or equal to 2^8 are solved by our fast-only algorithm. For each value of k we consider five (random) Toeplitz matrices.

Our next figures represent timings. As on our computer system measurements of execution times are done in units of 0.01 seconds, we limit the k -axis to that part where the results are meaningful. This is why in the following figures k does not start at one but at a larger value.

Figure 5.3 shows the execution time (in seconds) for Gaussian elimination with partial pivoting (these results were calculated via the LAPACK routines ZGETRF and ZGETRS), our fast algorithm and our superfast algorithm in case no iterative refinement is applied. The results are indicated with the symbols ‘+’, ‘o’ and ‘x’, respectively.

Figure 5.4 presents the results shown in Figure 5.3 in a different way. It gives the magnification of the execution time. For each k , it tells us by which factor the execution time is to be multiplied if we go from $k - 1$ to k .

Figures 5.5 and 5.6 are related to our superfast solver. For each value of k we consider five (random) Toeplitz matrices. No iterative refinement is applied. Figure 5.5 shows the percentage of the execution time spent to compute the input data for the interpolation problem formulated in Theorem 3.1, i.e., the time needed to evaluate the $a(\omega_k)$ ’s. Figure 5.6 shows the percentage of the execution time spent to apply the inversion formula given in Theorem 2.2 once the interpolation problem has been solved.

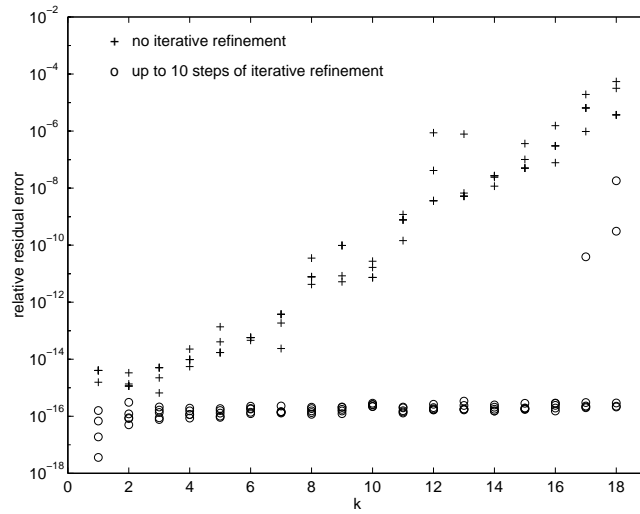


FIG. 5.1. $\frac{\|b_n - T_n \hat{x}_n\|_1}{\|b_n\|_1}$ versus $k = \log_2 n$ for $k = 1, \dots, 18$.

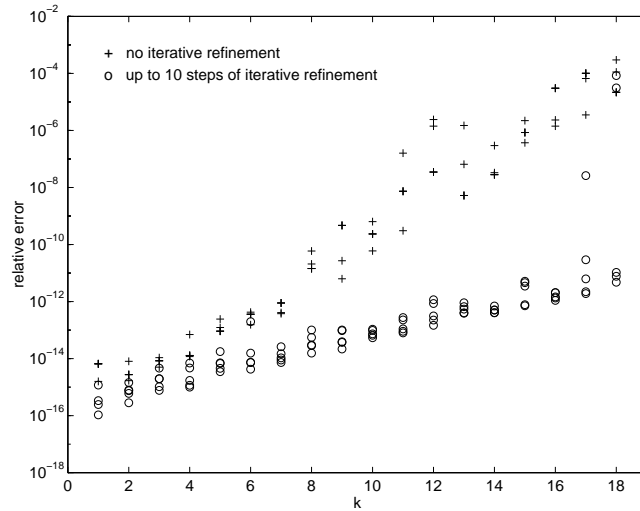
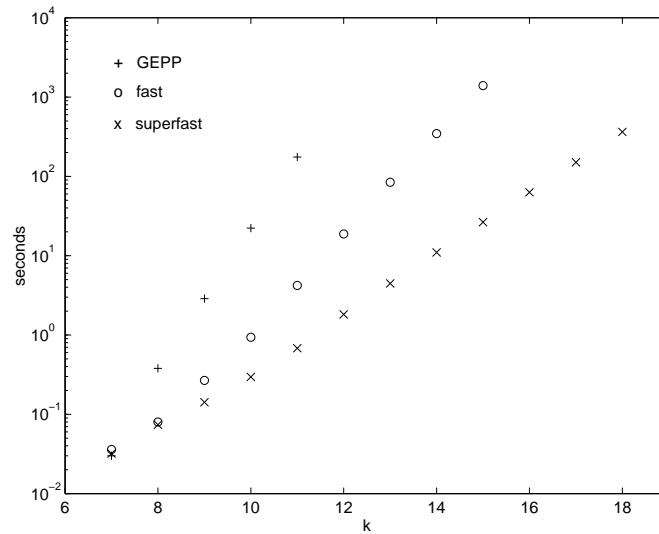
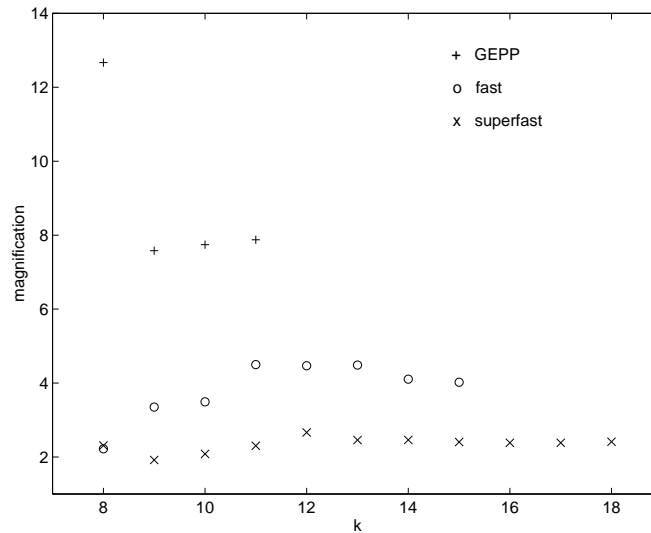


FIG. 5.2. $\frac{\|\hat{x}_n - x_n\|_1}{\|x_n\|_1}$ versus $k = \log_2 n$ for $k = 1, \dots, 18$.

We also consider matrices of size $n = 10000(5000)100000$. The entries are again real and random uniformly distributed in $[0, 1]$ and the right-hand sides are again calculated such that all the entries of the solution vector are equal to one. For each value of n we consider five matrices. Figure 5.7 shows the execution time (in seconds). The results are indicated with the symbol 'x'. The symbols 'o' correspond to the case where n is a power of two.

One expects that for n in the range $2^{k-1} < n \leq 2^k$ the execution time is more or less equal to that of the system of size 2^k . In practice, the execution time is less. This can be explained as follows. At the lowest level some of the first interpolation

FIG. 5.3. *Execution time in seconds.*FIG. 5.4. *Magnification of the execution time.*

problems can be solved via polynomial interpolation, i.e., by applying FFT.

The computed solution can be refined iteratively. Figure 5.8 shows how much execution time is spent on iterative refinement as percentage of the execution time in case no iterative refinement is applied. We consider one, two, three or four steps of iterative refinement. The results are represented by the symbols 'x', 'o', '+' and '*', respectively. For each value of k and each number of iterative refinement steps, five Toeplitz matrices are considered.

So far we have only considered iterative refinement at the Toeplitz level, i.e., we have refined the computed solution to the Toeplitz system iteratively. Iterative

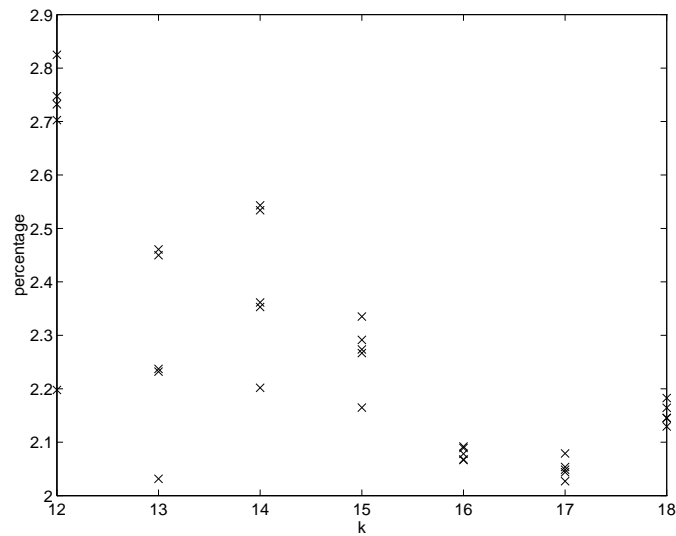


FIG. 5.5. *Percentage of the execution time spent to compute the input data for the interpolation problem.*

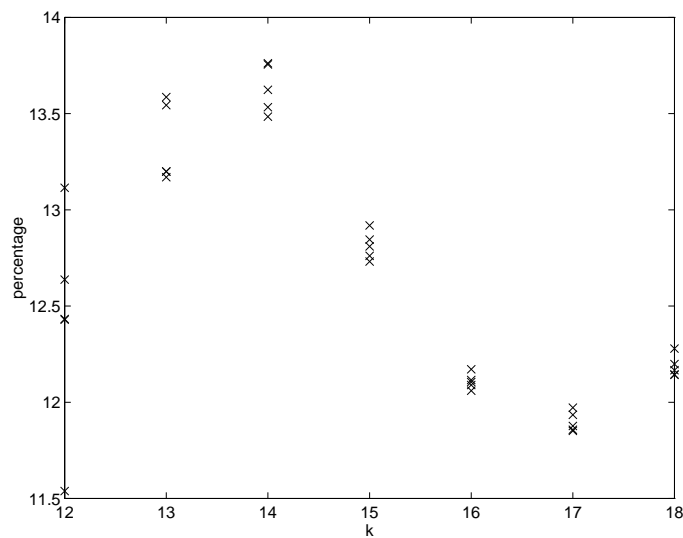


FIG. 5.6. *Percentage of the execution time spent to apply the inversion formula.*

refinement can also be applied at the interpolation level. In our next experiment we apply up to four steps of iterative refinement at the highest interpolation level. The timing results are shown in Figure 5.9. We compare the execution time spent on this kind of iterative refinement to the total execution time in case no iterative refinement whatsoever is applied. Observe that this kind of iterative refinement is much more expensive than iterative refinement applied at the Toeplitz level.

Iterative refinement at an interpolation level may be preceded by downdating. Numerical experiments indicate that the time needed to search the interpolation points that have to be downdated is approximately 45% of the time needed for one step of

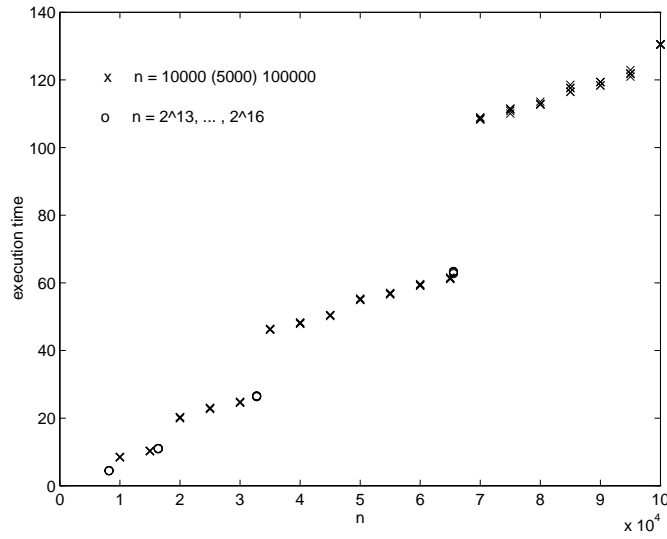


FIG. 5.7. Execution time for $n = 10000(5000)100000$ and $n = 2^{13}, \dots, 2^{16}$.

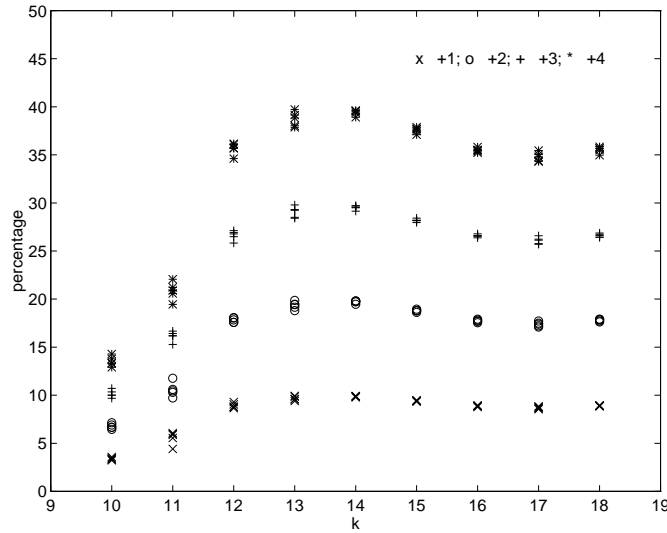


FIG. 5.8. Time spent on iterative refinement as percentage of the execution time in case no iterative refinement is applied.

iterative refinement.

The following example illustrates how important it is to find the proper combination of the stabilizing techniques that we have developed. For a certain matrix of size 2^{18} whose entries are random uniformly distributed in the interval $[0, 1]$, we have observed the following. By applying at most 10 steps of iterative refinement on the interpolation problems of size 2^{18} (this is the one but highest interpolation level; remember that a matrix of size 2^{18} corresponds to 2^{19} interpolation conditions), by considering 85 difficult points and by applying iterative refinement at the Toeplitz level, we obtain an approximation for the solution whose relative residual error is

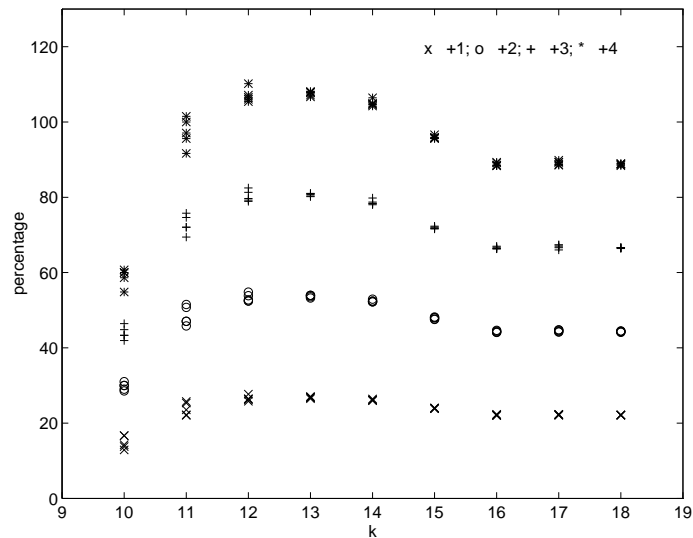


FIG. 5.9. Up to four steps of iterative refinement at the highest interpolation level. We compare the corresponding execution time to the total execution time in case no iterative refinement at all is applied.

$\mathcal{O}(10^{-15})$. If we do not consider difficult points and do not use iterative refinement at the interpolation level, then the computed approximation is so bad that iterative refinement at the Toeplitz level fails. The same holds if we only apply iterative refinement on the interpolation problems of size 2^{19} . This clearly shows the importance of combining our stabilizing tools in the correct way. One can of course apply iterative refinement on each interpolation level, but this is very costly. One has to find the correct balance between accuracy and cost. This will be the subject of future research.

REFERENCES

- [1] G. S. AMMAR AND W. B. GRAGG, *The generalized Schur algorithm for the superfast solution of Toeplitz systems*, in Rational Approximation and its Applications in Mathematics and Physics, J. Gilewicz, M. Pindor, and W. Siemaszko, eds., vol. 1237 of Lecture Notes in Mathematics, Springer, 1987, pp. 315–330.
- [2] ———, *Superfast solution of real positive definite Toeplitz systems*, SIAM J. Matrix Anal. Appl., 9 (1988), pp. 61–76.
- [3] ———, *Numerical experience with a superfast real Toeplitz solver*, Linear Algebr. Appl., 121 (1989), pp. 185–206.
- [4] R. R. BITMEAD AND B. D. O. ANDERSON, *Asymptotically fast solution of Toeplitz and related systems of linear equations*, Linear Algebr. Appl., 34 (1980), pp. 103–116.
- [5] A. W. BOJANCZYK, R. P. BRENT, F. R. D. HOOG, AND D. R. SWEET, *On the stability of the Bareiss and related Toeplitz factorization algorithms*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 40–57.
- [6] A. W. BOJANCZYK AND G. HEINIG, *A multi-step algorithm for Hankel matrices*, J. Complexity, 10 (1994), pp. 142–164.
- [7] R. BRENT, F. GUSTAVSON, AND D. YUN, *Fast solution of Toeplitz systems of equations and computation of Padé approximants*, J. Algorithms, 1 (1980), pp. 259–295.
- [8] S. CABAY AND D. CHOI, *Algebraic computations of scaled Padé fractions*, SIAM J. Comput., 15 (1986), pp. 243–270.
- [9] S. CABAY AND R. MELESHKO, *A weakly stable algorithm for Padé approximants and the inversion of Hankel matrices*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 735–765.
- [10] T. F. CHAN AND P. C. HANSEN, *A look-ahead Levinson algorithm for general Toeplitz systems*,

- IEEE Trans. Signal Process., 40 (1992), pp. 1079–1090.
- [11] ———, *A look-ahead Levinson algorithm for indefinite Toeplitz systems*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 490–506.
- [12] G. CYBENKO, *The numerical stability of the Levinson-Durbin algorithm for Toeplitz systems of equations*, SIAM J. Sci. Stat. Comput., 1 (1980), pp. 301–310.
- [13] F. DE HOOG, *A new algorithm for solving Toeplitz systems of equations*, Linear Algebr. Appl., 88/89 (1987), pp. 123–138.
- [14] R. W. FREUND, *A look-ahead Bareiss algorithm for general Toeplitz matrices*, Numer. Math., 68 (1994), pp. 35–69.
- [15] R. W. FREUND AND H. ZHA, *Formally biorthogonal polynomials and a look-ahead Levinson algorithm for general Toeplitz systems*, Linear Algebr. Appl., 188/189 (1993), pp. 255–303.
- [16] ———, *A look-ahead algorithm for the solution of general Hankel systems*, Numer. Math., 64 (1993), pp. 295–321.
- [17] K. A. GALLIVAN, S. THIRUMALAI, AND P. V. DOOREN, *A look-ahead Schur algorithm*, in Proceedings of the Fifth SIAM Conference on Applied Linear Algebra, Snowbird, Utah, June 1994, pp. 450–454.
- [18] K. A. GALLIVAN, S. THIRUMALAI, P. V. DOOREN, AND V. VERMAUT, *High performance algorithms for Toeplitz and block Toeplitz matrices*, Linear Algebr. Appl., 241–243 (1996), pp. 343–388.
- [19] L. GEMIGNANI, *Schur complements of Bezoutians and the inversion of block Hankel and block Toeplitz matrices*, Linear Algebr. Appl., 253 (1997), pp. 39–59.
- [20] I. GOHBERG, T. KAILATH, AND V. OLSHEVSKY, *Fast Gaussian elimination with partial pivoting for matrices with displacement structure*, Math. Comput., 64 (1995), pp. 1557–1576.
- [21] W. B. GRAGG, F. D. GUSTAVSON, D. D. WARNER, AND D. Y. Y. YUN, *On fast computation of superdiagonal Padé fractions*, Math. Program. Stud., 18 (1982), pp. 39–42.
- [22] W. B. GRAGG AND M. H. GUTKNECHT, *Stable look-ahead versions of the Euclidean and Chebyshev algorithms*, in Approximation and Computation: A Festschrift in Honor of Walter Gautschi, R. V. M. Zahar, ed., Birkhäuser, 1994, pp. 231–260.
- [23] M. GU, *Stable and efficient algorithms for structured systems of equations*, SIAM J. Matrix Analysis Appl., 19, 2 (1997), pp. 279–306.
- [24] M. GUTKNECHT AND M. HOCHBRUCK, *Optimized look-ahead recurrences for adjacent rows in the Padé table*, BIT, 36 (1996), pp. 264–286.
- [25] M. H. GUTKNECHT, *Stable row recurrences for the Padé table and a generically superfast look-ahead solver for non-Hermitian Toeplitz systems*, Linear Algebr. Appl., 188/189 (1993), pp. 351–421.
- [26] M. H. GUTKNECHT AND M. HOCHBRUCK, *Look-ahead Levinson- and Schur-type recurrences in the Padé table*, Electronic Transactions on Numerical Analysis, 2 (1994), pp. 104–129.
- [27] ———, *Look-ahead Levinson and Schur algorithms for non-Hermitian Toeplitz systems*, Numer. Math., 70 (1995), pp. 181–228.
- [28] G. HEINIG, *Inversion of generalized Cauchy matrices and other classes of structured matrices*, in Linear Algebra in Signal Processing, vol. 69 of IMA volumes in Mathematics and its Applications, IMA, 1994, pp. 95–114.
- [29] ———, *Solving Toeplitz systems after extension and transformation*, CALCOLO, 33 (1996), pp. 115–129.
- [30] ———, *Transformation approaches for fast and stable solution of Toeplitz systems and polynomial equations*, in Proceedings of the International Workshop “Recent Advances in Applied Mathematics”, State of Kuwait, May 4–7 1996, pp. 223–238.
- [31] G. HEINIG AND A. BOJANCZYK, *Transformation techniques for Toeplitz and Toeplitz-plus-Hankel matrices: I. Transformations*, Linear Algebr. Appl., 254 (1997), pp. 193–226.
- [32] ———, *Transformation techniques for Toeplitz and Toeplitz-plus-Hankel matrices: II. Algorithms*, Linear Algebr. Appl., 278 (1998), pp. 11–36.
- [33] G. HEINIG AND K. ROST, *Algebraic Methods for Toeplitz-like Matrices and Operators*, vol. 13 of Operator Theory: Advances and Applications, Birkhäuser, 1984.
- [34] T. HUCKLE, *A look-ahead algorithm for solving nonsymmetric linear Toeplitz equations*, in Proceedings of the Fifth SIAM Conference on Applied Linear Algebra, Snowbird, Utah, June 1994, pp. 455–459.
- [35] T. KAILATH AND A. H. SAYED, *Displacement structure: theory and applications*, SIAM Review, 37 (1995), pp. 297–386.
- [36] P. KRAVANJA AND M. VAN BAREL, *A fast block Hankel solver based on an inversion formula for block Loewner matrices*, CALCOLO, 33 (1996), pp. 147–164. Proceedings of the workshop Toeplitz Matrices: Structure, Algorithms and Applications, Cortona (Italy), September 9–12, 1996.

- [37] ———, *A fast Hankel solver based on an inversion formula for Loewner matrices*, Linear Algebr. Appl., 282 (1998), pp. 275–295.
- [38] R. KUMAR, *A fast algorithm for solving a Toeplitz system of equations*, IEEE Trans. Acoust. Speech Signal Process., 33 (1985), pp. 254–267.
- [39] G. LABAHN AND S. CABAY, *Matrix Padé fractions and their computation*, SIAM J. Comput., 18 (1989), pp. 639–657.
- [40] M. MORF, *Fast Algorithms for Multivariable Systems*, PhD Thesis, Dept. of Electrical Engineering, Stanford University, 1974.
- [41] ———, *Doubling algorithms for Toeplitz and related equations*, in Proc. IEEE Internat. Conf. on Acoustics, Speech and Signal Processing, Denver, CO, 1980, pp. 954–959.
- [42] B. R. MUSICUS, *Levinson and fast Cholesky algorithms for Toeplitz and almost Toeplitz matrices*, report, Res. Lab. of Electronics, M.I.T., 1984.
- [43] V. OLSHEVSKY, *Pivoting for structured matrices with applications*. To appear in Linear Algebra Appl.
- [44] Y. SUGIYAMA, *An algorithm for solving discrete-time Wiener-Hopf equations based on Euclid's algorithm*, IEEE Trans. Inf. Theory, 32 (1986), pp. 394–409.
- [45] Y. SUGIYAMA, M. KASAHARA, S. HIRASAWA, AND T. NAMEKAWA, *A new method for solving key equations for decoding Goppa codes*, Internat. J. Control, 27 (1975), pp. 87–99.
- [46] D. R. SWEET AND R. P. BRENT, *Error analysis of a fast partial pivoting method for structured matrices*, in Advanced Signal Processing Algorithms, Proceedings of SPIE-1995, T. Luk, ed., vol. 2563, 1995, pp. 266–280.
- [47] M. VAN BAREL AND A. BULTHEEL, *A general module theoretic framework for vector M -Padé and matrix rational interpolation*, Numer. Algorithms, 3 (1992), pp. 451–462.
- [48] ———, *The “look-ahead” philosophy applied to matrix rational interpolation problems*, in Systems and networks: Mathematical theory and applications, Volume II: Invited and contributed papers, U. Helmke, R. Mennicken, and J. Saurer, eds., vol. 79 of Mathematical Research, Akademie Verlag, 1994, pp. 891–894.
- [49] M. VAN BAREL AND P. KRAVANJA, *A stabilized superfast solver for indefinite Hankel systems*, Linear Algebr. Appl., 284 (1998), pp. 335–355. Special issue on the International Linear Algebra Society Symposium “Linear Algebra in Control Theory, Signals and Image Processing,” held at the University of Manitoba, Canada, 6–8 June 1997.