INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

Effective Computations with Dense Structured Matrices and Applications to Polynomial Evaluation and Interpolation

by

Olen Dias

A dissertation submitted to the Graduate Faculty in Mathematics in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

The City University of New York.

1997

This manuscript has been read and accepted for the Graduate Faculty in Mathematics in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

1/24/1997 — Victor Pan

Date — Chair of Examining Committee

1/24/97 — Grace Chanel

Date — Executive Officer

Prof. Michael Anshel

Prof. Myong-hi Kim

Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

Abstract

Effective Computations with Dense Structured Matrices and Applications to Polynomial Evaluation and Interpolation by

Olen Dias

Advisor: Professor V. Y. Pan

In recent years the research has demonstrated the power of combining the technique of algebraic and numerical computing. In the past, numerical algorithms for matrix computations and algebraic algorithms for polynomial computations were developed and implemented independently of each other with very little interaction.

In this study, we attempt to take some topics from both areas and show various correlations between them, in particular, via the study of computations with dense structured matrices (e.g. Toeplitz, Hankel, Vandermonde and Cauchy matrices) and of their applications to computations with both polynomial and general matrices. The main objective is to find a numerically stable as well as arithmetically fast algorithms for some selected fundamental problems of algebraic and numerical computations.

The computational cost is very high if n is large in many applications of $n \times n$ matrices. Hence it becomes important to develope algorithms that will reduce the burden on the computational resources of time and space. Since the applied problems often impose some structure on the matrices, the structure can be used to reduce the complexity bound for some major computations with structured matrices dramatically.

This is usually achieved by means of direct methods, based on matrix factorizations, but there are also some alternative iterative methods. Here we use two different iterative methods to accomplish our goal.

Acknowledgements

I want to acknowledge and thank my committee members Prof. Michael Anshel and Prof. Myong-hi Kim for their time and support. My heart felt gratitude goes to Prof. Victor Pan , my mentor, for all the guidance and help he has extended to me. Without his encouragement my task would be insurmountable. I wish to thank all the faculty members from the Mathematices Department of graduate school.

Completion of my education would have been impossible without the support of my family, specially uncle Ajit and his family, my aunt Bela and dear friend Doris. My deepest gratitude for my husband who was a great source of encouragement and support.

Contents

1 Introduction

The research done in recent years has demonstrated the power of combining the techniques for algebraic and numerical computing. Historically, numerical algorithms for matrix computations and algebraic algorithms for polynomial computations have been developed and implemented independently of each other with very little interaction. We took some topics from both areas and showed various correlations between them, in particular, via the study of computations with dense structured matrices (e.g. Toeplitz, Hankel, Vandermonde and Cauchy matrices) and of their applications to computations with both polynomials and general matrices. Our main objective is to find a numerically stable as well as arithmetically fast algorithms for some selected fundamental problems of algebraic and numerical computations. For some computational problems , the computatitonal cost of getting their exact solution is too high, and then we shift to the approximation algorithms that at a lower computational cost compute the desired output values within a fixed bound $\epsilon > 0$ on the output errors.

Polynomial and matrix computations are classical subjects, and they play a major role in present day practical computations in sciences, engineering, statistics, and processing information. Thus, the study of algorithms and complexity of these computations is fundamental for both theory and practice of computing . Frequently, matrices encoun-

tered in practical computations have some special structure, which can be exploited to simplify the computations. In particular, computations with Toeplitz matrices have numerous applications (in particular, to algebraic coding, control, algebraic computing, and partial differential equations). Large linear systems with Hermitian positive definite Toeplitz matrices arise also in some major signal processing computations. As another example, given an $n \times n$ Vandermonde matrix V and a vector x, the computation of the product Vx is equivalent to multipoint polynomial evaluation, whereas solving the system of linear equations $Vx = b$ is the same problem as polynomial interpolation. The Cauchy matrices appear in the study of integral equations, conformal mappings and singular integrals, where, in particular, Cv and $C^{-1}v$ are sought for given C and v. Computations with structured matrices (such as Toeplitz, Cauchy, and Vandermonde matrices) can be facilitated (so that the computational time and memory space decrease dramatically) by means of representing these matrices with their low rank generators associated with operators of displacement (shift) and/or scaling [KVM], [KKM], [CKLA], [GKK], [GKKL], [HR], [P90], [GO], [BP], [H], [GKO], [GO1] , [KO], [KS]. The main idea is, for a given structured matrix A, we look for an operator F that transforms A into a low rank matrix $F(A)$ such that we could easily recover A from its image $F(A)$ and then we may take all the advantages of operating with low rank matrices, even though the input matrix may have full rank. The operators F that

shift and scale the entries of the matrices turns out to be appropriate tools for defining the Toeplitz–like, Vandermonde–like and Cauchy–like matrices, which generalize Toeplitz, Vandermonde and Cauchy matrices.

Furthermore, technically, computations with dense structured matrices, including their numerous reductions and correlations to each other, are closely related to polynomial computations and greatly exploit the power of Fast Fourier Tranform (hereafter, FFT), so as to arrive at a dramatic acceleration of the algorithms versus the case of general matrices. FFT is numerically stable and allows effective parallel implementation. Many major problems of practical computing reduce to solving a linear system of equations with a structured coefficient matrix. For a system of linear equations $Ax = b$, for any $n \times n$ matrix A, the solution requires $O(n^3)$ ops. Since in many applications n is very large, it becomes important to develop algorithms that will reduce the burden on the computational resources of time and space. Since the applied problems often impose some structure on the matrices, the structure can be used to reduce the complexity bound for some major computations with structured matrices dramatically, to $O(n)$ words of storage space and to $O(n \log n)$ or $O(n \log^2 n)$ arithematic operations, with small overhead constants in comparison with $n \times n$ general matrices, that is, from $O(n^2)$ words of storage space and $O(n^\omega)$ arithmetic operations with $2.37 < \omega < 3$ in the best algorithms. This is usually

achieved by means of direct methods, based on matrix factorizations, but there are also some alternative iterative methods.

The iterative methods generate a sequence of approximate solutions $\{x^{(k)}\}$ and essentially involve matrix A only in the context of matrix-vector multiplication. An iterative method depends on how quickly the iterates $x^{(k)}$ converage. Iterative methods are in many situations quite attractive for use on vector or parallel computers. Many, if not most, of iterative methods are based upon the following approach. The matrix A is split into an easily invertible part P and a remainder Q. With easily invertible matrices P the systems $Pz = r$ can be solved "cheaply". When A is symmetric, positive definite, then the number of iteration steps required to decrease the norm of the error matrix below $\epsilon > 0$ is roughly proportional to the square root of the condition number of the matrix A.

Again, by using the properties of structured systems, with Toeplitz–like,Vandermonde–like, and Cauchy–like matrices, we may confine the computations to operations with short generators of the involved matrices, thus decreasing their computational cost dramatically.

Stating our complexity estimates , we will assume the customary sequential and parallel random access machine (RAM) models. The RAM models allow us to read one memory location, to execute an operation from a fixed set, or to write into one memory location. To each operation its time-cost is assigned. Generally, the cost may vary

depending on the kind of operation involved, and under the logarithmic cost criterion, it may also vary depending on the precision of the operands. The overall sequential time-cost is defined as the sum of the time-cost values assigned to all the operations involved. The RAM is called arithmetic because the basic operations are arithmetic. For the arithmetic RAM's, the input consists of one or more than one sets of variables, the set of constants and the set of operations allowed in the computation.

We will measure the computational cost by the number of arithmetic operations required in order to compute or to approximate (within a prescribed error bound) the desired output values. We will assume infinite precision computations and will focus on estimating the arithmetic time-complexity (the space-complexity of $O(mn)$ words of memory suffice for the computations with $m \times n$ matrices by our algorithms).

2 Some Definitions and Basic Facts

In our exposition we will use the common definitions and some simple known facts summarized in [BP].

Let n be a positive integer, and i, j, k integer parameters, ranging from 0 to $n-1$. Matrix rows and vector components are represented by i, columns by j. W^T is the transpose of a matrix (vector) W; W^H

is the Hermitian transpose of W.

Complex vectors $\mathbf{e}$, $\mathbf{f}$, $\mathbf{g}$, $\mathbf{u}$, $\mathbf{v}$, $\mathbf{w}$, $\mathbf{x}$ and $\mathbf{y}$ are of the form

$\mathbf{h} = [h_0, \ldots, h_{n-1}]^T$ (where $\mathbf{h}$ can represent any listed vector).

$r_i = r^i$, where $\mathbf{r} = \exp(2\pi\sqrt{-1}/n)$, is a primitive n-th root of 1. Let the components u_i of $\mathbf{u}$ be pairwise distinct and not equal to integer powers of $\mathbf{r}$.

A, B, C, H, I, J, R, S, T, V, W and Z represent appropriate matrices. In particular, the matrices

$$Z = \begin{bmatrix} 0 & \cdots & \cdots & \cdots & 0 \\ 1 & 0 & \cdots & \cdots & \vdots \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & 1 & 0 \end{bmatrix}$$

and

$$J = \begin{bmatrix} 0 & \cdots & 0 & 1 \\ \vdots & & \cdot & 0 \\ 0 & \cdot & & \vdots \\ 1 & 0 & \cdots & 0 \end{bmatrix}$$

satisfy $J^2 = I$ (the identity matrix), and

$J\mathbf{u} = [u_{n-1}, \ldots, u_0]^T$,

$Z\mathbf{u} = [0, u_0, \ldots, u_{n-2}]^T$, $Z^T\mathbf{u} = [u_1, \ldots, u_{n-1}, 0]$

Definition 2.1 *A matrix $V = V(\mathbf{u}) \in F_{m+1,n+1}$ is a Vandermonde matrix if*

$$V(\mathbf{u}) = [u_i^j].$$

$V(\mathbf{r}) = [r^{ij}]_{0 \le i,j \le n-1}$ is the matrix of discrete Fourier transform (DFT) on n points, so that $V(\mathbf{r})\mathbf{u} = \left[\sum_{i=0}^{n-1} r^{ij} u_i\right]^T$ is the DFT of a vector $\mathbf{u}$.

Definition 2.2 *Given two vectors $\mathbf{u}$ and $\mathbf{v}$ such that ($u_i \neq v_j$ for all i and j) , the $m \times n$ matrix $C(\mathbf{u}, \mathbf{v}) = C$ is a Cauchy (generalized Hilbert matrix) where*

$$C(\mathbf{u}, \mathbf{v}) = \left[\frac{1}{u_i - v_j}\right]_{0 \le i,j \le n-1}$$

Definition 2.3 *. T is a Toeplitz matrix, if*

$$T = [t_{i,j}] = [t_{i+k,j+k}], \; k \geq 1,$$

that is, if all the entries of T are invariant in their shift in the diagonal direction. Thus, the matrix T is completely defined by its first row and its first column. TJ and JT are Hankel matrices for any Toeplitz matrix T; hence all the entries of H are invariant in their antidiagonal shift, and

$$H = [h_{i,j}] = [h_{i-1,j+1}].$$

Definition 2.4 *An $m \times n$ matrix*

$$Z_f(x) = [z_{ij}],$$

for a vector $\mathbf{x} = [x_0, \ldots, x_{m-1}]^T$ *and for a scalar* $f \neq 0$, *is called an* $f-$*circulant matrix if* $z_{i,j} = x_{i-j \bmod m}$ *for* $i \geq j$; $z_{i,j} = f x_{i-j \bmod m}$ *for* $i < j$.

$$Z_f(\mathbf{x}) = \begin{bmatrix} x_0 & fx_3 & fx_2 & fx_1 \\ x_1 & x_0 & fx_3 & fx_2 \\ x_2 & x_1 & x_0 & fx_3 \\ x_3 & x_2 & x_1 & x_0 \end{bmatrix}$$

is a 4×4 f-circulant matrix. In particular, $Z_f = Z_f(e^{(1)})$ and $Z = Z_0$.

Definition 2.5 *Let* $F : F_{m,n} \rightarrow F_{m,n}$ *be an operator, let* $A \in F_{m \times n}$, *and let* $G \in F_{m \times \alpha}$, $H \in F_{n \times \alpha}$ *denote two matrices such that* $F(A) = GH^T$. *Then* $\alpha = rank(F(A))$, *the rank of the matrix F(A), is called the* F-rank *of A, and the pair of the matrices G and H is called an* F-generator *of A of length* α.

A generator of A is a matrix of lower rank α , $\alpha < n$ (associated with a given structured $n \times n$ matrix A) usually written in the form

$$\Delta_{R,S}(A) = A - RAS = GH^T, \text{ for } F = \Delta \qquad (2.1)$$

or

$$\nabla_{R,S}(A) = RA - AS = GH^T, \text{ for } F = \nabla \qquad (2.2)$$

for two fixed matrices R and S, representing scaling and/or displacement, and for some $n \times \alpha$ matrices G and H. Scaling is represented by

diagonal matrices,

$$D_x = diag(x_1, x_2, \ldots, x_n) \tag{2.3}$$

for fixed $x_1, \ldots, x_n$, whereas such matrices as

$$Z_f = \begin{bmatrix} 0 & 0 & \cdots & 0 & f \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \tag{2.4}$$

$$\text{or} \quad aZ_f + bZ_\phi^T + cZ_\psi^{-1}$$

for some fixed scalars a, b, c, f, ϕ and $\psi \neq 0$, represent displacement.

Remark 2.1 *The key-idea of using operators and associated generators is that the original matrix A can be easily recovered from its generator (2.1) or (2.2) (see [BP] chapter 2) and, moreover, the basic operations (such as multiplication, addition, subtraction, and inversion) with $n \times n$ structured matrices of certain classes can be reduced to operations with their generators; represented by $O(\alpha n)$ parameters. This leads to a dramatic saving of the computational time and memory space, if α is much less than n.*

For Toeplitz, Cauchy (generalized Hilbert), and Vandermonde matrices the length α of the associated generators (2.1) and (2.2), for some

appropriate choices of the matrices R and S among the matrices of the classes (2.3) and (2.4) is as small as 1 *or* 2, *in particular,*

1) for Toeplitz matrices $R, S \in \{Z_f, Z_f^T\}$

2) for Vandermonde $R, S \in \{Z, D_x\}$ for some x.

3) for Cauchy matrices $R, S \in \{D_x, D_y\}$ for some x and y.

These 3 classes of matrices are naturally extended to Toeplitz-like, Cauchy-like, and Vandermonde-like matrices, for which α is bounded by a fixed (and not too large) constant. (These classes include or are closely related to some other well-known classes of structured matrices, such as Sylvester, Subresultant, Hankel, Hankel-like, Lowener, Bezout, and Chebyshev-Vandermonde matrices [HR], [BP],[H], [GO1] , [KO]). It was observed in [P90] that some simple correlations among the operators associated with the matrices of the 3 cited classes can be exploited in order to reduce the computations for matrices of any of the 3 classes to computations with matrices of the class for which most effective algorithms are available.

Definition 2.6 *An $m \times n$ matrix is called a* Toeplitz-like matrix *if it has F-rank bounded from above by a constant independent of m and n, where F is the operator defined in (1).*

Hereafter, let $f = 1$, $Z = Z_0$.

Definition 2.7 *For a vector* $\mathbf{v} = (v_i)$, *we write* $\|\mathbf{v}\|_\infty = max_i|v_i|$ *and* $\|\mathbf{v}\|_q = (\sum_i |v_i|^q)^{\frac{1}{q}}$, *for* $q = 1, 2$, *and the associated h-norm of a matrix* A *is defined as*

$$max_h \frac{\|A\mathbf{v}\|_h}{\|\mathbf{v}\|_h}$$

for $h = 1, 2, \infty$. *The* condition number *of a nonsingular matrix* A *associated with the h-norm of* A *is defined as*

$$cond_h(A) = \|A\|_h \|A^{-1}\|_h,$$

for $h = 1, 2, \infty$.

Definition 2.8 *A matrix* A *is called* Hermitian *if* $A^H = A$, *real symmetric if* $A^H = A^T = A$. *A matrix* A *is called* Hermitian nonnegative definite *(h.n.d.) if it can be represented as a product* $W^H W$ *for some matrix* W. *If* W *is nonsingular, then so is* A, *then* A *is a* Hermitian positive definite *(h.p.d.) matrix.*

In practical numerical computations on a computer with a fixed precision, if the output precision would not be much less than the precision of the computations and of the input data, then it is customary to call the algorithm *"numerically stable"*.

The error affecting the result of a floating point computation has two main components. The first component collects the errors generated in each arithmetic operation performed with floating arithmetic

(called the arithmetic errors) ; the second component collects the errors induced by the approximation of the input data by means of floating point numbers (called the inherent errors).

If arithmetic error is "large" with respect to the precision of computation, then algorithm is "numerically unstable" and if "small" changes in the input produce "large" changes in the output, then the computational problem is called "ill-conditioned".

3 A Fast, Preconditioned Conjugate Gradient Toeplitz and Toeplitz-like Solvers

Toeplitz Solver: Given a vector $\mathbf{v}$ and a nonsingular Toeplitz matirx T, compute the solution vector $T^{-1}\mathbf{v}$ to the Toeplitz linear system $T\mathbf{x} = \mathbf{v}$.

3.1 Background

The Gohberg-Semencul formula [GS] gives a compressed representation of the inverses of Toeplitz matrices . This formula represents the inverse of a Toeplitz matrix in the form of the sum of products of triangular Toeplitz matrices. Namely, if for the Toeplitz matrix T the equations

$$T\mathbf{x} = e_0, \quad T\mathbf{y} = e_{n-1},$$

have solutions $\mathbf{x} = (x_i)_{i=0}^{n-1}$, $\mathbf{y} = (y_i)_{i=0}^{n-1}$ and if $x_0 \neq 0$, then T is invertible.

$$x_0 T^{-1} = L(\mathbf{x})L^T(J\mathbf{y}) - L(Z\mathbf{y})L^T(ZJ\mathbf{x})$$

where, $L(\mathbf{v})$ is a lower triangular Toeplitz with first column $\mathbf{v}$, and can be computed in $O(n\log^2 n)$ ops by means of different algorithms, presented in [BGY], [BA] and [MORF] (with larger overhead constants in both cases), and in [AG], [CK], [DH] and [MUS] (with smaller overhead constants). There are also $O(n^2)$ ops algorithms, which are

superior for smaller n. Finally, there are effective iterative Toeplitz solvers that use various preconditioners, Newton's iteration and the steepest descend method (see [P92], [LV]).

Many applicatons have motivated both mathematicians and engineers to try to develope 'fast' algorithms for solving Toeplitz systems of equations. Most of the early work on fast algorithms for Toeplitz systems focused on direct methods requiring $O(n^2)$ ops. Many other fast direct algorithms have been proposed since that time, and there has been also a recent surge of interest in iterative methods, specifically conjugate gradient.

The preconditioned conjugate gradient (PCG) algorithm is a popular iterative method for solving linear systems. Although the PCG algorithm is tyes problems exist. There are essentially two computationally expensive steps of the PCG algorithm is typically used for Hermitian positive definite (h.p.d.) systems of equations, its extensions to non-Hermitian, indefinite, and overdetermined least-squares problems exist. There are essentially two computationally expensive steps of the PCG algorithm. One is a matrix-vector multiplication with the coefficient matrix T at each iteration, which, if T is a Toeplitz matrix, can be done in $O(n \log n)$ operations. The other possible expensive computation is preconditioning step. The preconditioning step should not be more expensive than the cost of all matrix-vector multiplications involved, so that on average iteration of PCG for a Toeplitz linear system

of n equations should require only $O(n \log n)$ operations.

In contrast to the direct Toeplitz solvers using order of n^2 or $n \log^2 n$ arithmetic operations [T], [CB], [CK], [AG], [BA], [BGY], [DH], the preconditioned conjugate gradient method requires $O(kn \log n)$ operations, where $k = k(T)$ is the condition number of T. Therefore, the method is particularly effective for well-conditioned Toeplitz linear systems, which motivates the search for good preconditioners that would decrease the condition number and preserve the Toeplitz structure.

In [PS] such effective preconditioning was proposed for Hermitian (or real symmetric) positive definite (hereafter h.p.d.) Toeplitz systems, based on factorization of T into the product

$$T = (T + \mu I)(I - \mu(T + \mu I)^{-1})$$

for a scalar μ. The key idea of [PS] is that an appropriate choice of the scalar μ defined by two extreme eigenvalues of T implies a substantial decrease of the condition number of both factors relatively to k and thus substantially accelerates the solution of an associated Toeplitz linear system. This algorithm, however (as well as other competitive iterative preconditioned Toeplitz solvers [C1], [C2], [CS], [S]), works neither for the unsymmetric nor for Toeplitz-like cases, which are also highly important in computational practice.

The present section gives a desired extension of the algorithm of [PS] to these cases. The extension relies on the properties of the circulant

and skew-circulant displacement operators associated with Toeplitz and Toeplitz-like matrices and, in particular, on the recent explicit formulae expressing the displacement generators of the inverses of such matrices via few vectors associated with the inverses [GO]. More specifically, we replace T by its symmetrization T^HT and respectively change the factorization. $T^HT + \mu I$ and $I - \mu(T^HT + \mu I)^{-1}$ are still Toeplitz-like matrices, which we represent by using their short displacement generators and the explicit formulae from [GO]. This still enables fast multiplication of the matrix $I - \mu(T^HT + \mu I)^{-1}$ by a vector and leads to the desired extension of the algorithm of [PS], defining fast Toeplitz-like solvers, in the case of an ill-conditioned input.

In this section, we try to follow the line of [PS]. In the next subsection, we recall some relevant results on displacement representation of Toeplitz-like matrices. In subsection 3.3 we show a general outline of the method. In subsection 3.4 we specify various policies of choosing the parameter μ and their influence on the number of arithmetic operations required for the solution of Toeplitz and Toeplitz-like linear systems. In subsection 3.5 we specify a more effective solver in the Toeplitz case.

3.2 Some Properties of Toeplitz-like Matrices

We have the following basic lemmas.

Lemma 3.1 [BP]. *Let $A \in F_{n\times n}$, $B \in F_{m\times m}$ be two Toeplitz-like matrices given with their F-generators of lengths α_A and α_B, respectively. Then AB is a Toepiltz-like matrix having an F-generator of length $\alpha_{AB} \leq \alpha_A + \alpha_B$.*

Proof: follows from the observation that

$F(AB) = F(A)B + ZAZ^TF(B)$.

Lemma 3.2 (compare [BP], [GO], [KKM]). *Let A be a nonsingular Toeplitz-like matrix with an F-generator $F(A) = G_1H_1^T$, of length l_A. Then A^{-1} is a Toeplitz-like matrix with an F-generator equal to GH^T, where $G = -A^{-1}G_1$, and $H^T = H_1^TZA^{-1}Z^T$.*

Proof: Immediate. From these results, we have the following corollary.

Corollary 3.1. *Let T be an $n \times n$ Toeplitz-like matrix with an F-generator of length α_T. Then $B = T^HT + \mu I$, $C = I - \mu B^{-1}$ are Toeplitz-like matrices with $\alpha_B \leq 2\alpha_T$ and $\alpha_C \leq 2\alpha_T$, provided that $-\mu$ is not an eigenvalue of T^HT.*

Hereafter, α will stand for α_T.

3.3 A Condition-Improving Matrix Factorization

Lemma 3.3 [PS]. *Let A be an $n \times n$ matrix, $B = A + \mu I$, $C = I - \mu B^{-1}$. Then $A = BC = CB$. If $-\mu$ is not an eigenvalue of A, then both B and C have inverses, and $A^{-1} = C^{-1}B^{-1} = B^{-1}C^{-1}$.*

Let the eigenvalues of A, B and C be given by

$$\beta_n \leq \beta_{n-1} \leq \ldots \leq \beta_1 = \lambda(A),$$

$$\gamma_n \leq \gamma_{n-1} \leq \ldots \leq \gamma_1 = \lambda(B),$$

$$\delta_n \leq \delta_{n-1} \leq \ldots \leq \delta_1 = \lambda(C).$$

By the definition of B and C, we have

$$\gamma_j = \beta_j + \mu, \quad \delta_j = 1 - \mu\gamma_j^{-1}.$$

Lemma 3.4 [PS]. *Let A, B and C be as above and let $\mu > 0$. Then the condition numbers of B and C are given by*

$$k(B) = \frac{\beta_1 + \mu}{\beta_n + \mu} \tag{3.1}$$

and

$$k(C) = \frac{\beta_1}{\beta_n}\left(\frac{\beta_n + \mu}{\beta_1 + \mu}\right), \tag{3.2}$$

so that for all $\mu > 0$, we have

$$k(A) = k(B)k(C). \tag{3.3}$$

Lemma 3.5 [PS]. *Let $\mu = \sqrt{\beta_1\beta_n}$. Then $k(B) = k(C) = \sqrt{k(A)}$.*

3.4 A Fast Toeplitz-like Solver

Consider the linear system

$$Tx = b, \tag{3.4}$$

where T is an $n \times n$ nonsingular Toeplitz -like matrix, given with its F-generator of length α. Apply the matrix factorization of the previous section to the linear system,

$$T^H T x = T^H b. \tag{3.5}$$

Let $A = T^H T$, then A is an $n \times n$ h. p. d. Toeplitz-like matrix, $\alpha_A \leq 2\alpha$. Define $B = A + \mu I$, $C = I - \mu B^{-1}$. Suppose that $-\mu$ is not an eigenvalue of A. Then, by the results of the previous section, B and C are nonsingular Toeplitz-like matrices with $\alpha_B \leq 2\alpha$ and $\alpha_C \leq 2\alpha$. By the results of [GO], B^{-1} is completely defined by its last row and its F-generator :

$$B^{-1} = Z_{lr} + \frac{1}{1-f} \sum_{m=1}^{2\alpha} Z_f(u_m) Z_1(v_m^T), \tag{3.6}$$

where f is arbitrary, $f \neq 1$, Z_{lr} is the 1-circulant matrix with the last row equal to y^T . Furthermore, u_m, v_m and y^T satisfy following equations:

$$Bu_m = g_m, \tag{3.7}$$

$$Bt_m = -Z_1^T h_m, \tag{3.8}$$

$$v_m = Z_1 t_m, \quad m = 1, 2, \ldots, 2\alpha, \tag{3.9}$$

$$By = e_{n-1}, \quad e_{n-1} = (0, 0, \ldots, 1)^T, \tag{3.10}$$

where $G = [g_1, \ldots, g_{2\alpha}]$, $H = [h_1, \ldots, h_{2\alpha}]$ of A. Therefore, we have the following algorithm:

Algorithm 1

Input: *An $n \times n$ nonsigular Toeplitz-like matrix T, a vector b, and a shift value μ.*

Output: $T^{-1}b$.

Stage 1: *solve the equations (3.7)-(3.10).*

Stage 2: *solve $Bz = T^H b$.*

Stage 3: *solve $Cx = z$; return x.*

We use conjugate gradient (CG) method [GL] to obtain the solution at stages 1 and 3 in n_B and n_C iteration steps, respectively. Stage 2 amounts to $2\alpha + 1$ multiplications of f-circulant matrices by vectors for $f = 1$ and f . Therefore, by the well-known results (see [GO]), the arithmetic cost of performing stage 1, i.e. the arithmetic cost of performing n_B steps of the CG iteration on B, equals

$$cost(B) = (4\alpha + 1)(4\alpha + 3)\phi(n)n_B,$$

and similarly at stage 3, we have

$$cost(C) = (4\alpha + 3)\phi(n)n_C,$$

for n_C iterations of CG, where $\phi(n)$ is the cost of an n-point FFT.

3.4.1 The Optimal Shift

We will next follow [PS] by choosing the optimal μ such that the total work

$$[(4\alpha+1)(4\alpha+3)n_B + (4\alpha+3)n_C]\phi(n)$$

is minimized, where n_B and n_C are the numbers of steps of the CG iteration at stages 1 and 3, respectively. Let

$$n_B = F\sqrt{k(B)}\,, \tag{3.11}$$

$$n_C = F\sqrt{k(C)}\,, \tag{3.12}$$

where F is a constant. Then by (3.3),

$$n_B\, n_C = F^2\sqrt{k(A)} = M = constant.$$

Define

$$f(n_B) = Ln_B + n_C = Ln_B + \frac{M}{n_B},$$

where $L = 4\alpha + 1$. Then $f(n_B)$ is minimized at

$$n_B = \sqrt{\frac{M}{L}}\,,\ n_C = Ln_B. \tag{3.13}$$

In view of (3.11)–(3.13), we choose μ satisfying

$$k(C) = L^2k(B). \tag{3.14}$$

Use (3.1), (3.2) and let $\mu = m\sqrt{\beta_1\beta_n}$. We have the following equation:

$$m^2(L^2 - k(A)) + m\left[2(L^2-1)\sqrt{k(A)}\right] + (L^2k(A) - 1) = 0\,,$$

so

$$m_{\pm} = \frac{-(L^2-1)\sqrt{k(A)} \pm L(k(A)-1)}{L^2 - k(A)},$$

where $k(A)=\frac{\beta_1}{\beta_n}$, $L = 4\alpha + 1$. Since $L \geq 5$, $k(A) \geq 1$,

we have $m_- > 0$ only for $k(A) > L^2$.

Lemma 3.6 [PS]. *Let $\mu = m\sqrt{\beta_1\beta_n}$, where $m = m_-$ (see above). Then*

$$k(B) = L^{-1}\sqrt{k(A)}\,, \tag{3.15}$$

$$k(C) = L\sqrt{k(A)}\,. \tag{3.16}$$

Now assume (3.13) and choose $\mu = m_-\sqrt{\beta_1\beta_n}$. Then the total cost is

$$\begin{aligned}
&(4\alpha+3)[(4\alpha+1)n_B + n_C]\phi(n) \\
&= (4\alpha+3)(Ln_B + n_C)\phi(n) \\
&= 2(4\alpha+3)F\sqrt{k(C)}\,\phi(n) \\
&= 2(4\alpha+3)\sqrt{4\alpha+1}\,k^{\frac{1}{4}}(A)F\,\phi(n).
\end{aligned} \tag{3.17}$$

For comparison, let n_{CG} be the number of iterations required by CG for A . We have

$$Cost(CG) = (4\alpha+3)n_{CG}\,\phi(n) = (4\alpha+3)k^{\frac{1}{2}}(A)F\,\phi(n). \tag{3.18}$$

Comparing with (3.17) we can see an improvement for

$$k(A) > 16(4\alpha+1)^2.$$

3.4.2 Recursive Preconditioning

We may use the factorization $A = T^H T = BC$ recursively. In particular, we may solve equations (3.7), (3.8) and (3.10) at stage 1 of algorithm 1 by choosing one optimal shift μ_1, and we may choose another optimal shift μ_2 to solve the system $Cx = z$ for x at stage 3 of algorithm 1. Since we have $\alpha_B \leq 2\alpha$, $\alpha_C \leq 2\alpha$ (where α_W denotes the length of an F-generator of W, for $W = B$, $W = C$) , it follows from (3.17), that the total computational cost of performing stages 1 and 3 is bounded by

$$2(8\alpha + 1)(8\alpha + 3)\sqrt{8\alpha + 1}\, k^{\frac{1}{4}}(B) F\, \phi(n) \tag{3.19}$$

and

$$2(8\alpha + 3)\sqrt{8\alpha + 1}\; k^{\frac{1}{4}}(C) F\phi(n), \tag{3.20}$$

respectively. Now we choose μ so as to minimize the sum of (3.19) and (3.20). Since $k(A) = k(B)\, k(C)$,

we have the solutions

$$k(B) = \frac{k^{\frac{1}{2}}(A)}{(8\alpha + 1)^2}, \quad k(C) = (8\alpha + 1)^2\, k^{\frac{1}{2}}(A),$$

and

$$\mu = \frac{\beta_n k^{\frac{1}{2}}(A)[k^{\frac{1}{2}}(A)(8\alpha + 1)^2 - 1]}{k^{\frac{1}{2}}(A) - (8\alpha + 1)^2}.$$

We have $\mu > 0$ for $k(A) > (8\alpha + 1)^4$, and the total computational cost of recursive preconditioning is

$$4(8\alpha + 1)(8\alpha + 3) F\phi(n)\, k^{\frac{1}{8}}(A). \tag{3.21}$$

This is less than the cost (3.17) of non-recursive preconditioning for

$$k(A) > \frac{2^8(8\alpha+1)^8(8\alpha+3)^8}{(4\alpha+1)^4(4\alpha+3)^8}$$

and is also less than the cost of application of the unpreconditioned (CG) method to $Ax = b$ (see (3.18)) when

$$k(A) > \left[\frac{4(8\alpha+1)(8\alpha+3)}{4\alpha+3}\right]^{\frac{8}{3}}.$$

For $\alpha = 2, 3$, we compare the estimates (3.17), (3.18) and (3.21) and show the results in the next table.

α cost	2	3
CG method	$11k^{\frac{1}{2}}(A)F\phi(n)$	$15k^{\frac{1}{2}}(A)F\phi(n)$
non-recursive	$66k^{\frac{1}{4}}(A)F\phi(n)$	$30\sqrt{13}\,k^{\frac{1}{4}}(A)F\phi(n)$
recursive	$1292k^{\frac{1}{8}}(A)F\phi(n)$	$2700k^{\frac{1}{8}}(A)F\phi(n)$

3.5 Preconditioned CG Method for a Toeplitz Matrix

In this section, we use the same notation as in the previous section, except that T now denotes a nonsingular Toeplitz matrix (so that $\alpha = 2$). Since $B = T^H T + \mu I$, multiplying the matrix B by a vector costs $8\phi(n) + O(n)$. Thus in algorithm 1 we have $cost(B) = 72\phi(n)$

at stage 1. By [GO], $cost(C) = 11\phi(n)$ at stage 3, for each iteration. Therefore, the overall work is equal to

$$(72n_B + 11n_C)\phi(n) = 11(\tilde{L}n_B + n_C)\phi(n),\ \tilde{L} = \frac{72}{11},$$

where n_B and n_C denote the number of the CG iterations at stages 1 and 3, respectively. Assume the optimal value of $\mu = m_-\sqrt{\beta_1\beta_n}$, where

$$m_\pm = \frac{-(\tilde{L}^2 - 1)\sqrt{k(A)} \pm \tilde{L}(k(A) - 1)}{\tilde{L}^2 - k(A)}.$$

Then, similarly to (3.16), we derive the following cost bound for the entire computation:

$$22\, n_C\, \phi(n) = 12\sqrt{22}\, k^{\frac{1}{4}}(A) F\, \phi(n). \tag{3.22}$$

We may compare the bound of (3.22) to the cost of the solution via the CG method (without preconditioning), which is estimated similarly to (3.18) and is bounded by

$$8k^{\frac{1}{2}}(A)F\phi(n). \tag{3.23}$$

The comparison shows that our preconditioning improves the CG method for

$$k(A) > 2450.25.$$

Now, we use the factorization $A = BC$ recursively. We choose μ_1 so as to minimize the cost of performing stage 1 of Algorithm 1, which gives us the bound

$$9 \cdot 12 \cdot \sqrt{22}\, k^{\frac{1}{4}}(B)F\phi(n) = 108\sqrt{22}\, k^{\frac{1}{4}}(B)F\phi(n), \tag{3.24}$$

where, the factor 9 comes from the equations at stage 1. At stage 3, choose μ_2 so as to decrease the cost to

$$4(8\cdot 4+1)(8\cdot 4+3)F\phi(n)k^{\frac{1}{8}}(C) = 4620F\phi(n)k^{\frac{1}{8}}(C) \qquad (3.25)$$

[compare (3.21)]. Now we choose μ so as to minimize the sum of (3.24) and (3.25). Then we obtain that

$$k(B) = \left(\frac{1155}{54}\right)^8 \cdot \frac{1}{22^{\frac{4}{3}}} \cdot k^{\frac{1}{3}}(A),$$

$$k(C) = \left(\frac{54}{1155}\right)^{\frac{8}{3}} \cdot (22)^{\frac{4}{3}} \cdot k^{\frac{2}{3}}(A),$$

and the overall cost is bounded by

$$\left[108(22)^{\frac{1}{6}}\left(\frac{1155}{54}\right)^2 + 4620\left(\frac{54}{1155}\right)^{\frac{1}{3}} 22^{\frac{1}{6}}\right] k^{\frac{1}{12}}(A)F\phi(n)$$

$$= E\, k^{\frac{1}{12}}(A)F\phi(n), \qquad (3.26)$$

where,

$$E = \left[108\left(\frac{1155}{54}\right)^2 + 4620\left(\frac{54}{1155}\right)^{\frac{1}{3}}\right] 22^{\frac{1}{6}} = 400,993.268\ldots$$

[compare(3.21)]. Therefore, the recursive method is superior to the nonrecursive method only if $k(A)$ is enourmosly large:

$$k(A) > (\frac{E}{12\sqrt{22}})^6.$$

We also compare (3.26) and (3.23) and conclude that the recursive method improves the unpreconditioned CG method only for extremely large $k(A)$, $k(A) > (\frac{E}{8})^{\frac{12}{5}}$.

4 Inversion of Cauchy–like Matrix

4.1 Background

After some permutations of the columns and the rows of a Cauchy matrix, the matrix takes the form

$$C = \left[\frac{a_i^T b_j}{t_i - s_j}\right], \qquad A^T = [a_1, \ldots, a_n], \qquad B^T = [b_1, \ldots, b_n],$$

where $a_i, b_j \in C^{\alpha \times 1}$, $D_t = \mathrm{diag}(t_1, \ldots, t_n)$, $D_s = \mathrm{diag}\ (s_1, \ldots, s_n)$, $t_i \neq s_j$, for all $i, j = 1, 2, \ldots, n$ (cf. [H], [GKO]). This allows pivoting techniques into fast algorithms for generalized Cauchy matrix. It is a well known fact that the inverse of structured matrix also posses a similar displacment structure ([KKM], [HR], [GO2]). In particular, the inverse of the above Cauchy matrix is a matrix of the form

$$C^{-1} = -\left[\frac{u_i^T w_j}{s_i - t_j}\right], \qquad where \quad u_i, w_j \in C^{\alpha \times 1},$$

$D_t = \mathrm{diag}(t_1, \ldots, t_n)$, $D_s = \mathrm{diag}\ (s_1, \ldots, s_n)$, $t_i \neq s_j$, for all $i, j = 1, 2, \ldots, n$.

A fast algorithm with partial pivoting was suggested by Heinig [H] for the inversion and solving linear system with generalized Cauchy matrices. This algorithm computes in $O(\alpha n^2)$ ops, where α is the length of the displacement operator and the vectors u_i and w_i as above, then the linear system $Cx = b$ is solved by computing matrix-vector product. In [GO3] , [GO4] a fast implementation of Gaussian elimination with partial pivoting was designed for Cauchy-like matrices. The algorithm

uses $O(\alpha n^2)$ ops for the triangular factorization of C, then the linear system can be solved in $O(n^2)$ ops via forward and back substitution [GL]. While in [GKO], the partial pivoting was incorporated into fast algorithm not only for Cauchy–like matrices but also for Toeplitz–like, Vandermonde–like and Chebyshev–Vandermonde–like matrices.

The concept of a Toeplitz–like matrix was first introduced in [KKM] using the displacement operator Z_0. Later on, different authors studied Toeplitz–like matrices, using various displacement operators of the forms (2.1) and (2.2), with different choices of matrices R and S ([HR], [AG], [GO1], [GO2]). For Toeplitz and Toeplitz–like matrices R and S are not diagonal matrices. This fact makes it difficult to directly introduce pivoting into fast Toeplitz solvers.

On the other hand, due to a simple reduction (available-like FFT) matrices, Toeplitz–like and Vandermonde–like matrices to Cauchy–like matrices ([H], [GKO]), effective algorithms for computations with Cauchy–like matrices could play most fundamental role.

In this paper, we consider the solution of a nonsingular Cauchy–like linear system, $C\mathbf{x} = \mathbf{v}$ and the inversion of a nonsingular Cauchy–like matrix C , immediately extendable to the Vandermonde–like and Toeplitz–like cases, as well as to the Chebyshev–Vandermonde cases ([GO1], [KO]).

Newton's iteration is one of the most general and very powerful

numerical algorithm. We will apply it to the matrix equation

$$I - AX^{-1} = 0$$

in which case it takes the form

$$X_{k+1} = X_k(2I - AX_k), \quad for\ k = 0, 1, \ldots.$$

Newton's iteration converges very fast and is stable algorithm, its convergence rate is quadratic [$I - AX_{k+1} = (I - AX_k)^2$]. This is a customary tool for numerous polynomial computations that can be reduced to solving the equation of the form $f(X) = 0$.

It is well known that Newton's iteration may rapidly improve a rough initial approximation to the matrix inverse (cf. e.g. [BP]), but such an iteration also rapidly destroys the structure of Cauchy–like, Toeplitz–like and Vandermonde–like matrices . In ([P92], [P93], [P93a]), it was proposed to modify Newton's iteration in order to preserve the initial displacement structure of a Toeplitz–like input matrix during the iteration. The idea was to control the growth of the length of short displacement generators by periodically chopping-off their components corresponding to the smallest singular values in the SVD of these generators.

In the present paper, we consider a similar problem of controlling the length of the associated generators in a modification of Newton's iteration, where the inverse of a fixed Cauchy–like input matrix C is

sought. The problem is substantially simplified in this case (in particular, we do not need to involve the SVD), due to the formula for the inverse matrix C^{-1} available from ([H]); this should motivate the reduction of Toeplitz–like case to the Cauchy–like case (by means of FFTs) and applying the techniques of the present paper, instead of the inversion of Toeplitz–like matrices by applying the techniques of ([P92], [P93], [P93a]).

We present our results in the following order. In the next subsection, we describe our modification of Newton's iteration for the refinement of an initial approximation to the inverse of a Cauchy–like matrix. In subsection 4.3, we estimate the computational cost of each iteration, performed by operating with short displacement generators of matrices, rather than with the matrices themselves. In subsection 4.4, we specify the assumptions about the initial approximation of the inverse that guarantes rapid convergence of the iteration, and we also specify the number of iteration steps sufficient for convergence to an approximation within a fixed output error bound. In the appendix, we briefly recall the reduction of the inversion of Toeplitz–like, Vandermonde–like, and Chebyshev–Vandermonde–like matrices to the inversion of Cauchy–like matrices.

4.2 Modified Newton's Iteration for the Inversion of Cauchy–like Matrices

Let C be an $n \times n$ nonsingular Cauchy–like matrix with the associated scaling operator

$$\nabla_{\{D_t, D_s\}}(C) = D_t C - C D_s = A B^T, \qquad (4.1)$$

$$C = \left[\frac{a_i^T b_j}{t_i - s_j}\right], \qquad A^T = [a_1, \ldots, a_n], \qquad B^T = [b_1, \ldots, b_n], \qquad (4.2)$$

where $a_i, b_j \in C^{\alpha \times 1}$, $D_t = \mathrm{diag}(t_1, \ldots, t_n)$, $D_s = \mathrm{diag}\ (s_1, \ldots, s_n)$, $t_i \neq s_j$, for all $i, j = 1, 2, \ldots, n$ (cf. [H], [GKO]). Assume that an initial approximation X_0 to C^{-1} is available with its $\nabla_{\{D_t, D_s\}}$-generator of length at most α. Then we recursively define matrices $X_1^*, X_1, X_2^*, X_2, \ldots$ as follows :

$$X_{k+1}^* = X_k(2I - CX_k), \qquad k = 0, 1, \ldots, \qquad (4.3)$$

$$X_{k+1} = -\left[\frac{(u_i^{k+1})^T w_j^{k+1}}{s_i - t_j}\right]_{i,j=1}^{n}, \qquad k = 0, 1, \ldots, \qquad (4.4)$$

where the vectors $u_i^{k+1}, w_j^{k+1} \in C^{\alpha \times 1}$ are defined by

$$U_{k+1} = \begin{bmatrix} (u_1^{k+1})^T \\ \vdots \\ (u_n^{k+1})^T \end{bmatrix} = X_{k+1}^* A, \qquad (4.5)$$

$$W_{k+1}^T = \left[w_1^{k+1}, \ldots, w_n^{k+1}\right] = B^T X_{k+1}^*, \tag{4.6}$$

Equation (4.3) represents a step of Newton's iteration for matrix inversion (cf. e.g.[BP]), and equation (4.4) "corrects" the results X_{k+1}^* of (4.3) so as to turn X_{k+1}^* into a Cauchy–like matrix, associated with the same scaling operators as C^{-1}. Namely,

$$\nabla_{\{D_s, D_t\}}(X_{k+1}) = -U_{k+1} W_{k+1}^T,$$

that is, X_{k+1} is a Cauchy–like matrix whose $\nabla_{\{D_s, D_t\}}$–generator has a length of at most α. Furthermore, we have,

Proposition 4.1 *For any $k = 0, 1, \ldots$, the matrix*

$$X_{k+1}^* = 2X_k - X_k C X_k$$

is a Cauchy -like matrix whose $\nabla_{\{D_s, D_t\}}$ - generator has a length of at most 3α.

Proof: By observing that

$\nabla_{\{D_s, D_t\}}(X_k C X_k) =$

$\nabla_{\{D_s, D_t\}}(X_k) C X_k + X_k \nabla_{\{D_t, D_s\}}(C) X_k + X_k C \nabla_{\{D_s, D_t\}}(X_k),$

we obtain that

$$\nabla_{\{D_s, D_t\}}(X_{k+1}^*) = \nabla_{\{D_s, D_t\}}(2X_k - X_k C X_k)$$

$$= 2\nabla_{\{D_s, D_t\}}(X_k) - \nabla_{\{D_s, D_t\}}(X_k C X_k)$$

$$= 2\nabla_{\{D_s, D_t\}}(X_k) - \nabla_{\{D_s, D_t\}}(X_k) C X_k - X_k \nabla_{\{D_t, D_s\}}(C) X_k - X_k C \nabla_{\{D_s, D_t\}}(X_k)$$

$$= \nabla_{\{D_s,D_t\}}(X_k)(2I - CX_k) - X_k\nabla_{\{D_t,D_s\}}(C)X_k - X_kC\nabla_{\{D_s,D_t\}}(X_k).$$

We have

$$\nabla_{\{D_s,D_t\}}(X_k) = U_kW_k^T, \quad \nabla_{\{D_t,D_s\}}(C) = AB^T,$$

where $U_k, W_k, A, B \in C^{n\times\alpha}$. *Therefore* $\nabla_{\{D_s,D_t\}}(X_{k+1}^*) = U_{k+1}^*(W_{k+1}^*)^T$, *where*

$$U_{k+1}^* = [U_k, -X_kA, -X_kCU_k] \in C^{n\times(3\alpha)},$$

$$W_{k+1}^* = [(2I - CX_k)^TW_k, X_k^TB, W_k] \in C^{n\times(3\alpha)}.$$

4.3 Computational Complexity of an Iteration Step

Next, we estimate the arithmetic cost of computing the matrices X_{k+1}, U_{k+1}, W_{k+1}, and X_{k+1}^* based on (4.3)-(4.6). Since C and X_k are Cauchy-like matrices, the computation of the $\nabla_{\{D_s,D_t\}}$ - generator of length 3α for X_{k+1}^* [according to (4.3)] uses $O(\alpha^2 n\log^2 n)$ ops (see e.g. [GO], [BP], chapter 2, section 4,11 and 12). Therefore, (4.5) and (4.6) together enable us to compute the $n \times \alpha$ matrices U_{k+1} and W_{k+1} by using $O(\alpha^2 n\log^2 n)$ ops. An additional attractive feature of this compututation is the economization of computer memory due to representation of all involved matrices by means of their short generators occupying only $O(\alpha n)$ words of memory. We also refer the reader to [BP], pages 130, 261-262, on some alternative methods for faster numerical approximation of the product of a Cauchy matrix C by a

vector, which may lead to a further decrease of the computational cost of our iteration steps.

4.4 Estimating Convergence Rate of Newton's iteration

In the following, we will estimate how fast X_k approaches C^{-1}. From [H], we have

$$C^{-1} = -\left[\frac{u_i^T w_j}{s_i - t_j}\right], \tag{4.7}$$

where

$$U = \begin{bmatrix} u_1^T \\ \vdots \\ u_n^T \end{bmatrix} = C^{-1}A, \quad W^T = [w_1, \ldots, w_n] = B^T C^{-1}, \tag{4.8}$$

$A^T = [a_1, \ldots, a_n]$, $B^T = [b_1, \ldots, b_n]$. We recall from (4.4)–(4.6) and (4.8) that

$$X_k = -\left[\frac{(u_i^k)^T (w_j^k)}{s_i - t_j}\right],$$

$$U_k = X_k^* A = (X_k^* - C^{-1})A + C^{-1}A = (X_k^* - C^{-1})A + U,$$

$$W_k^T = B^T X_k^* = B^T (X_k^* - C^{-1}) + W^T.$$

Then we obtain the following matrix equation:

$U_k W_k^T$ =

$(X_k^* - C^{-1})AB^T(X_k^* - C^{-1}) + UW^T + UB^T(X_k^* - C^{-1}) + (X_k^* - C^{-1})AW^T.$

From this equation and (4.8) and we obtain that

$$E_k = U_k W_k^T - UW^T$$

$$= (X_k^*-C^{-1})AB^T(X_k^*-C^{-1})+C^{-1}AB^T(X_k^*-C^{-1})+(X_k^*-C^{-1})AB^TC^{-1}, \tag{4.9}$$

Hereafter, we use the column-norm of matrices,

$$\|W\| = \|W\|_1 = \max_j \sum_i |w_{ij}|,$$

where $W = (w_{ij})_{m\times n}$ (see [GL], p.57).

Proposition 4.2 *Let* $e_k^* = \|X_k^* - C^{-1}\|$. *Then*

$$\|E_k\| = \|U_kW_k^T - UW^T\| \le \|AB^T\|\, e_k^*\,(e_k^* + 2\|C^{-1}\|).$$

Proof: Proposition 4.2 immediately follows from (4.9).

Proposition 4.3 *Let* $e_k = \|X_k - C^{-1}\|$, *for nonsingular matrix* C, *and let* X_{k+1}^* *be defined by (4.3), for* $k = 0, 1, \ldots$. *Then we have*

$$e_{k+1}^* \le \|C\|\, e_k^2. \tag{4.10}$$

Proof: Due to (4.3), we have $I - CX_{k+1}^* = (I - CX_k)^2, \quad k = 0, 1, \ldots$.

It follows that

$e_{k+1}^* = \|X_{k+1}^* - C^{-1}\|$

$= \|C^{-1}(I - CX_{k+1}^*)\|$

$= \|C^{-1}(I - CX_k)^2\|$

$= \|C^{-1}(I - CX_k)CC^{-1}(I - CX_k)\|$

$= \|(C^{-1} - X_k)C(C^{-1} - X_k)\| \le \|C\|\, e_k^2.$

Proposition 4.4 *For any $k = 1,2, \ldots,$ we have*

$$e_k \leq s_k e_k^*, \quad e_{k+1}^* \leq (s_k e_k^*)^2 \, \|C\|$$

where

$$s_k = \rho \, \|AB^T\| \, (e_k^* + 2\|C^{-1}\|), \quad \rho = \max_{i,j} \frac{1}{|s_i - t_j|}. \tag{4.11}$$

Proof: We recall (4.4), (4.7)–(4.9), and proposition 4.2 and obtain that

$$\begin{aligned} e_k = \|X_k - C^{-1}\| &= \max_j \sum_i \frac{|(u_i^{(k)})^T w_j^{(k)} - u_i^T w_j|}{|s_i - t_j|} \\ &\leq \rho \max_j \sum_i |(u_i^{(k)})^T w_j^{(k)} - u_i^{(k)} w_j| \\ &= \rho \, \|E_k\| \leq \rho \, \|AB^T\| \, (e_k^* + 2\|C^{-1}\|) \, e_k^* = s_k e_k^*. \end{aligned}$$

for s_k of (4.11). Combining the latter bound on e^k with (4.10) gives us proposition 4.4.

Proposition 4.5

$$If \quad e_1^* \leq 1 \tag{4.12}$$

and

$$(e_1^*)^\theta \|C\| s_1^2 \leq 1, \quad for \quad \theta \leq 1 \tag{4.13}$$

and for s_1 of proposition 4.4, Then

$$e_{k+1}^* \leq (e_k^*)^{2-\theta} \leq \ldots \leq (e_1^*)^{(2-\theta)^k}, \quad for \; k = 1, 2, \ldots. \tag{4.14}$$

Proof: By virtue of proposition 4.4, we have

$$e_2^* \leq (e_1^*)^2 \, \|C\| \, s_1^2$$

Combine this bound with (4.13) and obtain that

$$e_2^* \le (e_1^*)^{2-\theta}. \tag{4.15}$$

Since

$$\theta \le 1, \quad 2-\theta \ge 1,$$

we obtain from the latter inequality and (4.12) the bounds

$$e_2^* \le e_1^* \le 1, \tag{4.16}$$

which extend (4.12). Substitute the first inequality of (4.16) into (4.11) and obtain that $s_2 \le s_1$. Substitute the latter bound and the bound $e_2^* \le e_1^*$ of (4.16) into (4.13) and obtain that $(e_2^*)^\theta \|C\| s_2^2 \le 1$, which extends (4.13). Inductive application of this argument enables us to extend (4.12)–(4.15) to the bounds

$$e_k^* \le 1, \quad (e_k^*)^\theta \|C\| s_k^* \le 1, \quad e_k^* \le (e_{k-1}^*)^{2-\theta}$$

for $k = 3, 4, \ldots$, and we arrive at proposition 4.5.

We will next restate proposition 4.5, by replacing e_1^* by $e_0^2\|C\|$, based on proposition 4.3 for $k = 0$, $r_0 = \|I - CX_0\|$

$$\|C^{-1}\| \le \frac{\|X_0\|}{1-r_0}.$$

follows from the next inequalities,

$$\|C^{-1}\| - \|X_0\| \le \|C^{-1} - X_0\| \le \|C^{-1}\| r_0.$$

Proposition 4.6 *If* $e_0\sqrt{\|C\|} \leq 1$ *and if*

$$e_0^{2\theta}\|C\|^{1+\theta}s^2 \leq 1, \tag{4.17}$$

for $\theta \leq 1$, *and*

$$s = \rho\|AB^T\|(e_0^2\|C\| + 2\|C^{-1}\|),$$

and for ρ *of (4.11), then*

$$e_{k+1}^* \leq (e_0^2\|C\|)^{(2-\theta)^k},$$

for $k = 0, 1, \ldots$.

Proof: Proposition 4.3 for $k = 0$ implies that

$$e_1^* \leq \|C\|e_0^2. \tag{4.18}$$

Therefore, the bound $e_1^* \leq 1$ of (4.12) holds if $e_0\sqrt{\|C\|} \leq 1$, and we also have that $s_1 \leq s$ for s_1 of (4.11). Combining the latter bound, (4.17) and (4.18) gives us (4.13). Therefore, the assumptions of proposition 4.6 imply the ones of proposition 4.5, and consequently, imply (4.14). Subtitute (4.18) into (4.14) and obtain proposition 4.6.

We are not supposed to have the values e_0 and $\|C^{-1}\|$ readily available, when we are given the matrices C and X_0, but we may use more readily available parameters. Indeed

$$e_0 \leq r_0\|C^{-1}\|$$

$$\text{where } r_0 = \|I - X_0C\|,$$

and

$$\|C^{-1}\| \leq \frac{\|X_0\|}{(1-r_0)} \text{ if } r_0 < 1.$$

(The latter implication immediately follows from the next inequalities:

$$\|C^{-1}\| - \|X_0\| \leq \|C^{-1} - X_0\| \leq \|C^{-1}\| r_0.)$$

We are going to substitute the latter estimates into the statement of proposition 4.6. We write

$$e_0^+ = \frac{r_0\|X_0\|}{1-r_0} \tag{4.19}$$

$$s_{k+1}^+ = \rho\,\|AB^T\| \left[((e_0^+)^2\|C\|)^{(2-\theta)^k} + \frac{2\|X_0\|}{1-r_0}\right], \tag{4.20}$$

$k = 0, 1, \ldots,$ so that $e_0^+ \geq e_0$ and $s_{k+1}^+ \geq s_{k+1}$

if $s_{k+1}^* \leq ((e_0^+)^2\|C\|)^{(2-\theta)^k}$,

$$s_{k+1}^+ \leq \rho\|AB^T\| \left(1 + \frac{2\|X_0\|}{1-r_0}\right), \tag{4.21}$$

for all k, if $e_0^+\sqrt{\|C\|} \leq 1$. We now summarize our results also by taking into account the bound $e_{k+1} \leq s_{k+1}e_{k+1}^*$ (from the proof of proposition 4.4), which enable us to estimate e_{k+1} as soon as we estimate e_{k+1}^*.

Corollary 4.1 *Let*

$$r_0 = \|I - CX_0\| < 1, \quad e_0^+\sqrt{\|C\|} \leq 1,\ \theta \leq 1, \tag{4.22}$$

$$(e_0^+))^{2\theta}\|C\|^{1+\theta}(s_0^+)^2 \leq 1, \tag{4.23}$$

for e_0^+ and s_k^+ of (4.19) and (4.20) respectively. Then

$$e_{k+1}^* \leq ((e_0^+)^2\|C\|)^{(2-\theta)^k}, \quad k = 0, 1, \ldots,$$

and

$$e_{k+1} \leq s_{k+1}^* e_{k+1}^*, \quad k = 0, 1, \ldots.$$

Let us write

$$k^* = \left\lceil \frac{\log \frac{\log \epsilon^*}{\log((e_0^+)^2\|C\|)}}{\log(2-\theta)} \right\rceil, \tag{4.24}$$

$$\hat{k} = \left\lceil \frac{\log \frac{\log \epsilon}{\log((e_0^+)^2\|C\|)}}{\log(2-\theta)} \right\rceil. \tag{4.25}$$

where $\theta < 1$, e_0^+ and s_{k+1}^+ are defined by (4.19) and (4.20), so that (4.21) holds. Then under the assumptions of corollary 4.1, it suffices to perform $k+1 \geq k^*+1$ recursive steps of the iteration (4.3), (4.4) in order to ensure that

$$e_{k+1}^* = \|X_{k+1}^* - C^{-1}\| \leq \epsilon^*,$$

and it suffices to perform $k+1 \geq \hat{k}+1$ recursive steps in order to insure that

$$e_{k+1} = \|X_{k+1} - C^{-1}\| \leq \epsilon\, s_{k+1}^+.$$

5 Polynomial Evaluation and Interpolation

Let

$$w(x) = \sum_{i=0}^{n} w_i x^i,$$

be a polynomial of degree n.

Polynomial Evaluation:

Given the coefficient of polynomial $w(x)$ of degree n, Evaluate the values $v_i = w(u_i)$ at points $u_0, \ldots, u_{m-1}, m > n$.

Polynomial Interpolation:

Given several points $u_0, \ldots, u_n$ and values of the polynomial at these points $v_i = w(u_i)$, Calculate the coefficients of the polynomial $w(x)$.

5.1 Background

It is known that interpolation and evaluation (on any set of n nodes) of an n-th degree polynomial can be performed in $O(n \log^2 n)$ (arithmetic) operations [AHU], but recursive application of several polynomial divisions is involved, and this makes the resulting algorithms highly unstable numerically (thus, restricting their application to the case of computers performing exact rational arithmetic). Fast Fourier transform (FFT) enables us to solve both of these problems in $O(n \log n)$ operations [AHU] with no numerical stability problems [GS], in the special case where the nodes are roots of unity, and similarly in some other special cases [BP]. In particular, for the input nodes lying

on a bounded real interval, the estimate $O(n \log^2 n)$ can be improved [P90a, Ro88]. However, for a general set of nodes on the complex plane, the order of n^2 operations is required in the known numerically stable solutions.

In this section, we propose a new approach to both problems, which we solve approximately, within a given tolerance $K\epsilon$ to the error, K denoting the condition number of some auxiliary computational problem [see subsection 5.3]. Then Solution time is proportional to $n(\log^2 n + \log(1/\epsilon))$, which turns into $O(n \log^2 n)$ if $\log(1/\epsilon) = O(\log^2 n)$. Polynomial divisions are avoided in this approach, based on the application of computations with structured matrices.

The solution involves Toeplitz-like linear systems, which for many inputs may still cause some numerical stability problems [BUN], not as devastating, however, as ones caused by recursive polynomial divisions. In particular, we may shift to symmetrized systems which are still of Toeplitz-like , and if they remain sufficiently well-conditioned after their symmetrization, then numerical stability problems are avoided [BUN].

Our algorithms have the feature of many iterative algorithms: their output errors and their running time decrease with the condition number of the auxiliary linear system (in our case of Toeplitz type) to which we reduce the solution. Furthermore, we may try to use a random transformation of some input parameters in order to improve the

condition of the latter linear system (see remark 5.1 and the derivation of the complexity estimates based on our first algorithm of subsection 5.3).

It is known that the Vandermonde matrix (defining the interpolation problem) is ill-conditioned for a very large class of sets of interpolation nodes, and the parameter K of equation (5.6), defining the approximation error of our computations, tends to be large. Thus we cannot, as of now, recommend our algorithms for practical computations, except for the special cases of node sets defining well-conditioned Vandermonde matrices.

From the theoretical point of view, however, the new algorithms may be of interest since they demonstrate some previously hidden correlations between computations with polynomials and with structured matrices. Specifically, we represent the original computational problems of polynomial evaluation and interpolation in the form of operations with a Vandermonde matrix. Then we apply the techniques of [P90] for computations with structured matrices and reduce the original problem with any set of nodes to the case of roots of unity as the nodes; in this case FFT applies. The reduction involves Toeplitz–like computations (with matrices having displacement rank at most 3) and a single multiplication of a generalized Hilbert matrix by a vector, at which stage we apply the fast approximation algorithm of [Ro85], known to be effective and reliable in numerous computations, in par-

ticular, for integral equations and n-body mechanics. We present and analyze our algorithms in subsection 5.3 after some definitions and auxiliary results.

5.2 Auxiliary Results

We recall the following simple and/or well-known results :

$$C(\mathbf{u},\mathbf{v}) = -C^T(\mathbf{v},\mathbf{u}). \tag{5.1}$$

$$V(\mathbf{r}) = V^T(\mathbf{r}), \quad V^H(\mathbf{r})V(\mathbf{r}) = nI. \tag{5.2}$$

Fact 5.1 *The values of a polynomial* $w(x) = \sum_{i=0}^{n-1} w_i x^i$ *(with coefficient vector* $\mathbf{w}$ *) on the set of points* $u_0, \ldots, u_{n-1}$ *are given by the vector*

$$\mathbf{v} = V(\mathbf{u})\mathbf{w}. \tag{5.3}$$

Fact 5.1 defines the problem of interpolation and multipoint evaluation of a polynomial $w(x)$ in terms of a vector equation (compare Problem in 5.4).

Remark 5.1 It is simple to shift from polynomial $w(x)$ to $t(x) = w(ax+b)$ for any fixed complex numbers a and b, and vice versa, at the cost of $O(n \log n)$ operations [ASU]. Even simpler is the transition to the reverse polynomial $w_{\text{rev}}(x) = x^n w(1/x)$. These transformations enable us to vary (to our convenience) the input matrix $V(\mathbf{u})$ of the

problems of polynomial interpolation and multipoint evaluation. For evaluation, we may also vary the input node set, for instance, by partitioning it into two or several subsets, complementing each subset to n points at our choice and solving two or several evaluation problems.

Fact 5.2 [AHU]. *Given vectors* $\mathbf{u}$ *and* $\mathbf{r}$, *it suffices to use* $O(n \log n)$ *arithmetic operations to compute* $V(\mathbf{r})\mathbf{u}$ *and* $V^H(\mathbf{r})\mathbf{u}$ *(that is, to perform the forward and inverse DFT of a vector* $\mathbf{u}$ *).*

Fact 5.3 *a)* [AHU] *Given a vector* $\mathbf{v}$ *and a Toeplitz matrix* T, *it suffices to use* $O(n \log n)$ *arithmetic operations to compute the vector* $T\mathbf{v}$.

b) [AG], [BA], [BGY], [DH] *Furthermore,* $O(n \log^2 n)$ *arithmetic operations suffice to compute* $T^{-1}\mathbf{v}$ *if* T *is nonsingular.*

Fact 5.4 *The estimates of the previous fact hold even if the matrix* T *is replaced by any matrix of the form* $V^T(\mathbf{u})V(\mathbf{u})$, $W(\mathbf{u},\mathbf{r})$ *and* $W^T(\mathbf{u},\mathbf{r})$, *where,* $W(\mathbf{u},\mathbf{r}) = V^T(\mathbf{r})C(\mathbf{r},\mathbf{u})V(\mathbf{u})$.

Proof. The extension of fact 5.3 to the matrix $V^T(\mathbf{u})V(\mathbf{u}) = \left[\sum_{k=0}^{n-1} u_k^{i+j}\right]$ follows since this is a Hankel matrix. The extension of fact 5.3 to $W(\mathbf{u},\mathbf{r})$ and $W^T(\mathbf{u},\mathbf{r})$ follows from [P90]. (Specifically, proposition 6.1 of [P90] implies that $W(\mathbf{u},\mathbf{r})$, $W^T(\mathbf{u},\mathbf{r})$ are Toeplitz–like matrices defined with their $n \times 3$ displacement generator matrices

(see definitions in [CKLA, P90]), and to such matrices both parts of fact 5.3 can be extended [see [CKLA] and/or [MUS] on the extension of part a) and [BA] or [MUS] on the extension of part b)]).

Propositon 5.1 [Ro85]. *Given a natural n, positive a, q, s, and ϵ and three complex vectors* $\mathbf{u}$, $\mathbf{v}$ *and* $\mathbf{y}$ *of dimension n, it suffices to use $(5n-1)s$ arithmetic operations to compute, within the error bound ϵ, the values*

$$(C(\mathbf{v},\mathbf{u})\mathbf{y})_i = \sum_{k=0}^{n-1} \frac{y_k}{v_i - u_k},$$

for $i = 0, 1, \ldots, n-1$, provided that

$$s \geq \frac{\log(an) - \log((1-q)\epsilon)}{\log(1/q)}, \tag{5.4}$$

$$a \geq |\frac{y_k}{u_k}|, \quad 1 > q > |\frac{v_i}{u_k}| \quad \textit{for all } i \textit{ and } k. \tag{5.5}$$

Proof (see appendix A).

Remark 5.2 *a) If, say, $q < 1/2$, $\log a = O(\log n)$, $\log(1/\epsilon) = O(\log n)$, then* (5.4) *can be satisfied for $s = O(\log n)$.*

b) In our algorithms of the next section, $\mathbf{v} = \mathbf{r}$, *and the vector* $\mathbf{u}$ *can be linearly transformed, according to remark 5.1. This enables us to insure the last inequality of (5.5) for $q = 1/2$ and for all i and k.*

5.3 Interpolation and Multipoint Evaluation

Problem 5.1 *Interpolation.*

Input: *vectors* $\mathbf{u}$ *and* $\mathbf{v}$.

Note that, by assumption about vectors $\mathbf{u}$ and $\mathbf{r}$, $V(\mathbf{u})$ is non-singular, and $C(\mathbf{r}, \mathbf{u})$ can be defined. Let $C(\mathbf{r}, \mathbf{u})$ be non-singular (until the end of this paper).

Solution: successively compute the three vectors:

1. $\mathbf{f} = C(\mathbf{r}, \mathbf{u})\mathbf{v}$,

2. $\mathbf{g} = V^T(\mathbf{r})\mathbf{f} = V(\mathbf{r})\mathbf{f}$,

3. $\mathbf{w} = W^{-1}(\mathbf{u}, \mathbf{r})\mathbf{g}$.

The *correctness* of this algorithm follows since for the computed vectors $\mathbf{w}$ and $\mathbf{g}$, we have:

$$W(\mathbf{u}, \mathbf{r})\mathbf{w} = W(\mathbf{u}, \mathbf{r})W^{-1}(\mathbf{u}, \mathbf{r})\mathbf{g} = \mathbf{g} = V^T(\mathbf{r})C(\mathbf{r}, \mathbf{u})\mathbf{v},$$

and (5.3) follows since $V(\mathbf{r})$ and $C(\mathbf{r}, \mathbf{u})$ are nonsingular.

We apply the algorithm for approximate evaluation of $\mathbf{w}$, so as to decrease the estimated *computational complexity*.

At stage 1, approximating the components of $\mathbf{f}$ (within $\epsilon > 0$) requires $(5n - 1)s$ operations, for s defined by (5.4);

Stage 2 requires $O(n \log n)$ operations (see fact 5.2);

Finally, at stage 3, we need to compute the matrix $W(\mathbf{u},\mathbf{r})$, of Toeplitz-like , or more precisely, to compute its displacement generator of length (at most) 3. For this, we just need to compute the products $\mathbf{w}(\ell) = \left(W(\mathbf{u},\mathbf{r}) - ZW(\mathbf{u},\mathbf{r})Z^T\right)\mathbf{v}(\ell)$, $\ell = 1,2,3$, for three general vectors $\mathbf{v}(1)$, $\mathbf{v}(2)$, $\mathbf{v}(3)$.

Moreover, we may choose these vectors in the form

$$\mathbf{v}(\ell) = \mathbf{b}(\ell) = [1, b(\ell), (b(\ell))^2, \ldots, (b(\ell))^{n-1}]^T,$$

where $b(\ell)$ is a random parameter, $\ell = 1,2,3$. [Indeed, the $n \times n$ matrix $B = [(b(i))^j]$ has rank n, if all the $b(i)$ are distinct numbers, $i,j = 0,1,\ldots,n-1$; therefore, with a high probability (see [SCHW]), for a random choice of $b(i)$, the matrix $\left(W(\mathbf{u},\mathbf{r}) - ZW(\mathbf{u},\mathbf{r})Z^T\right)B$ has rank 3, and so has any of its random $n \times 3$ submatrices. The vector $V(\mathbf{u})\mathbf{b}(\ell)$, $\ell = 1,2,3$, can be computed in $O(n \log n)$ operations, since the evaluation of $V(\mathbf{u})\mathbf{b}(\ell)$ amounts to evaluation of the polynomial

$$\frac{1-(b(\ell)x)^n}{1-b(\ell)x} = \sum_{i=0}^{n-1}(b(\ell)x)^i$$

at the points $u_0,\ldots,u_{n-1}$.

Therefore, the complexity of approximate evaluation of the vectors $\mathbf{w}(\ell)$, for $\ell = 1,2,3$, is still within the bounds of $O(n(s+\log n))$, for s defined by (5.4), provided that ϵ denotes the tolerance to the errors of the appproximate multiplications of $C(\mathbf{r},\mathbf{u})$ by vectors in the process of the evaluation of $\mathbf{w}(\ell)$.

Given $\mathbf{w}(\ell)$, $\ell = 1,2,3$, we can, by using the algorithms of facts 5.3 or 5.4, find $W^{-1}\mathbf{g}$ at the cost $O(n\log^2 n)$. Therefore, stage 3 can be done at the cost of $O(n(s+log^2 n))$, s defined by (5.4).

The bound ϵ on the approximation errors of all the multiplications of vectors by the matrix $C(\mathbf{u},\mathbf{r})$ is not substantially magnified in the subsequent multiplications of the resulting vectors by the matrix $V(\mathbf{r})$ (having 2-norm equal to 1) but may be substantially increased in the evaluation of $W^{-1}\mathbf{g}$ unless

$$K = \operatorname{cond} W \tag{5.6}$$

is small.

Problem 5.2 *Evaluation.*

Input: *vectors* $\mathbf{u}$ *and* $\mathbf{w}$.

Output: *vector* $\mathbf{v}$ *satisfying* (5.3).

Solution: successively compute

1. $U(\mathbf{u}) = V^T(\mathbf{u})V(\mathbf{u})$,

2. $\mathbf{e} = U(\mathbf{u})\mathbf{w}$,

3. $\mathbf{x} = W^T(\mathbf{u},\mathbf{r})^{-1}\mathbf{e}$,

4. $\mathbf{y} = V(\mathbf{r})\mathbf{x}$,

5. $\mathbf{v} = C^T(\mathbf{r},\mathbf{u})\mathbf{y}$.

Correctness.

Premultiply both sides of (5.3) by $V^T(\mathbf{u})$ and obtain that

$$\mathbf{e} = V^T(\mathbf{u})V(\mathbf{u})\mathbf{w} = V^T(\mathbf{u})\mathbf{v}.$$

Then substitute $\mathbf{v} = C^T(\mathbf{r},\mathbf{u})\mathbf{y} = C^T(\mathbf{r},\mathbf{y})V(\mathbf{r})\mathbf{x}$, obtain that

$$W^T(\mathbf{u},\mathbf{r})\mathbf{x} = \mathbf{e},$$

and thus verify the correctness of the above solution.

Complexity.

Follow [CKL] to perform stage 1 in $O(n\log^2 n)$ operations. Specifically, first compute at this cost [AHU] the coefficients of the polynomial $\prod_{k=0}^{n}(u-u_k)$. Then obtain the power sums $\sum_{k=0}^{n-1} u_k^s$ in $O(n\log n)$ operations from the system of Newton's identities (see e.g. [P90a], appendix A).

We need $O(n\log n)$ operations at stage 2 and $O(n(s+\log^2 n))$ at stage 3, for s defined in Proposition 5.1 [use fact 5.4 and the algorithm for the evaluation of $W(\mathbf{u},\mathbf{r})$ shown above], and $O(n\log n)$ operations at stage 4 (due to fact 5.2).

At stage 5, we approximate all the components of the vector $\mathbf{v}$ within the error bound ϵ at the cost $(5n-1)s$, where s is defined by Proposition 5.1.

A Proof of Proposition 5.1

Substitute the expressions

$$\frac{1}{(v-u_k)} = -\frac{1}{u_k}\sum_{h=0}^{\infty}\left(\frac{v}{u_k}\right)^h$$

and obtain the power series representation

$$p(v) = \sum_{h=0}^{\infty} p_h v^h = \sum_{k=0}^{n-1} \frac{y_k}{v-u_k}$$

$$= -\sum_{k=0}^{n-1} \frac{y_k}{u_k} \frac{1}{1-(v/u_k)},$$

$$p_h = -\sum_{k=0}^{n-1} \frac{y_k}{u_k^{h+1}}, \quad h = 0, 1, \ldots \quad .$$

$p(v)$ converges when $|v|$ is small enough. Approximate $p(v)$ by the s-term partial sum and estimate the error in terms of s, $|v_i/u_k|$ and a. Due to (5.4) and (5.5), we obtain:

$$\left| p(v_i) - \sum_{h=0}^{s-1} p_h v_i^h \right| \leq \frac{anq^s}{1-q} \leq \epsilon, \quad \text{for all } i$$

It remains to compute first $p_0, \ldots, p_{s-1}$, by using $(3n-1)s$ operations, and then

$\sum_{h=0}^{s-1} p_h v_i^h$ for $i = 0, 1, \ldots, s-1$,

by using $2ns$ operations.

B Extension to the Inversion of Vandermonde–like and Chebyshev–Vandermonde–like matrices

Let V be an $n \times n$ Vandermonde–like matrix such that

$$\nabla_{\{D_{\frac{1}{x}}, Z_1^T\}}(V) = D_{\frac{1}{x}} V - V Z_1^T = G B^T,$$

where $G \in C^{n \times \beta}$, $B \in C^{\beta \times n}$. Then, due to a result from [GKO] , VF^* is a Cauchy–like matrix such that

$$\nabla_{\{D_{\frac{1}{x}}, D_1^*\}}(VF^*) = GH^T.$$

Here, $H^T = B^T F^*$, $\quad D_{\frac{1}{x}} = diag(\frac{1}{x_1}, \ldots, \frac{1}{x_n})$,

$D_1^* = diag(1, e^{\frac{-2\pi i}{n}}, \ldots, e^{\frac{-2\pi i(n-1)}{n}})$,

$$F = \frac{1}{\sqrt{n}} \left[e^{\frac{2\pi i}{n}(k-1)(j-1)} \right]_{k,j=1}^{n},$$

$$Z_1^T = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 1 & 0 & 0 & \cdots & 0 \end{bmatrix},$$

so that F stands for the (normalized) matrix of Discrete Fourier Transform (DFT) , and F^* is the Hermitian tranpose of the matrix F. $F^* = F^{-1} = \sqrt{n}$.

Let X_0, C, and θ satisfy the assumption of thr corollary(4.1). Let $\epsilon = \frac{\epsilon^*}{\sqrt{n}}$ and let $X_{k^*}^*$ denotee the matrix obtained in $k^* + 1$ steps (4.3),

(4.4) [for k^* defined by (4.24)] at the arithmetic computational cost $O(k^*\beta^2 n \log^2 n)$. Then we have

$$\|(VF^*)^{-1} - X^*_{k^*+1}\| < \epsilon\sqrt{n}.$$

Next, we consider a Chebyshev–Vandermonde–like matrix R , which has generators G , B such that

$$\nabla_{\{2D_x, Z_1+Z_1^T\}}(R) = 2D_xR - R(Z_1 + Z_1^T) = GB^T,\ G, B \in C^{\beta\times n}.$$

By the virtue of a result from [KO], $C = RF^*$ is a Cauchy–like matrix such that

$$\nabla_{\{2D_x, D_{cos}\}}(RF^*) = 2D_xRF^* - RF^*D_{cos} = G\,B^*F^*,$$

where, F, D_x, Z_1 are as above, and

$D_{cos} = diag(2, 2\cos(\frac{\pi}{n}), \ldots, 2\cos(\frac{n-1}{n}\pi))$.

By using the latter equations, we may extend the inversion of C to the inversion of R, similarly to the case of a Vandermonde–like matrix V.

Finally, we recall how to reduce the inversion of Toeplitz–like matrices to the Cauchy–like case.

Proposition B.1[GKO]. Let $T \in C^{n\times n}$ be a Toeplitz–like matrix , such that

$$\nabla_{\{Z_1, Z_{-1}\}}(T) = Z_1T - TZ_{-1} = GB^T,$$

where $G \in C^{n\times\beta}$ and $B \in C^{n\times\beta}$. Then $C = FTD_0^{-1}F^*$ is a Cauchy–like

matrix:

$$\nabla_{\{D_1,D_{-1}\}}(FTD_0^{-1}F^*) = D_1(FTD_0^{-1}F^*) - (FTD_0^{-1}F^*)D_{-1} = \hat{G}H^T.$$

Here D_{-1}, F, F^*, and Z_1 are defined above,

$$D_1 = diag(1, e^{\frac{2\pi i}{n}}, \ldots, e^{\frac{2\pi i(n-1)}{n}}),$$

$$D_0 = diag(1, e^{\frac{\pi i}{n}}, \ldots, e^{\frac{\pi i(n-1)}{n}}),$$

$$\hat{G} = FG, \text{ and } \hat{H}^T = B^T D_0^{-1} F^*.$$

Suppose that X_0, C, and θ satisfy the assumptins of the corollary (4.1) , let $\epsilon = \epsilon^*/\sqrt{n}$, and let $X^*_{k^*+1}$ be the matrix obtained in $k^* + 1$ steps (4.3),(4.4) for k^* of (4.24) and such that

$$\|C^{-1} - X^*_{k^*}\| \leq \epsilon/n.$$

Then we have

$$\|(FTD_0^{-1}F^*)^{-1} - X^*_{k^*+1}\| \leq \epsilon/n.$$

We have $(FTD_0^{-1}F^*)^{-1} = FD_0T^{-1}F^*$,

$$F^*D_0^{-1}[(FTD_0^{-1}F^*)^{-1} - X^*_{k^*+1}]F = T^{-1} - F^*D_0^{-1}X^*_{k^*+1}F,$$

and therefore, by writing $\hat{X}_{k^*+1} = F^*D_0^{-1}X^*_{k^*+1}F$, we obtain that

$\|T^{-1} - \hat{X}_{k^*+1}\| \leq \|F^*\|\|D_0^{-1}\|\|F\|\epsilon/n = \epsilon,$

since $\|D_0^{-1}\| = 1, \|F\| = \|F^*\| = \sqrt{n}.$

Thus, in $k^* + 1$ steps (4.3), (4.4), at the arithmatic computational cost $O(k^*\beta^2 n \log^2 n)$, we will arrive at a desired matrix $\hat{X}_{k^*+1} = F^*D_0^{-1}X^*_{k^*}F$ approximating T^{-1} within the error norm ϵ.

References

[AHU] A. Aho, J. Hopcroft and J. Ullman, *The Design and Analysis of Computer Algorithms* (Addison-Wessley, Reading, MA, 1974).

[AG] G. Ammar, W.G. Gragg, Superfast Solution of Real Positive Definite Toeplitz Systems, *SIAM J. Matrix Anal. Appl.*, 9, 1, 61–76, 1988.

[ASU] A. Aho, K. Steiglitz and J. Ullman, Evaluating Polynomials at Fixed Set of Points, *SIAM J. on Computing* 4 (1975),533–539

[BA] R. R. Bitmead, B. D. O. Anderson, Asymptotically Fast Solution of Toeplitz and Related System of Linear Equations, *Linear Algebra Appl.*, 41, 111-130, 1981.

[BGY] R. P. Brent, F. G. Gustavson, D. Y. Y. Yun, Fast Solution of Toeplitz Systems of Equations and Computation of Pade Approximations, *J. of Algorithms*, 1, 259–295, 1980.

[BP] D . Bini , V. Y. Pan, Polynomial and Matrix Computation, vol. 1, *Fundamental Algorithms*, Birkhauser, Boston , 1994.

[BUN] J. R. Bunch, Stability of Methods for Solving Toeplitz Systems of Equations, *SIAM J. on Scientific aand Statist. Computing* 6 (1985), 349–364.

[C1] R. H. Chan, Circulant Preconditioners for Hermitian Toeplitz System , *SIAM J. Matrix Anal. Appl.*, 10, 4, 542–550.

[C2] R. H. Chan, Toeplitz Preconditioners for Toeplitz Systems with Nonnegative Generating Functions, *IMA J. of Numerical Analysis*, 11, 333–345, 1991.

[CB] G. Cybenko, M. Berry, Hyperbolic Householder Algorithm for Factoring Structured Matrices, *SIAM J. Matrix Anal.*, 11, 4, 499–520, 1990.

[CK] J. Chan, T. Kailath, Divide and Conquer Solution of Least-Squares Problems for Matrices with Displacement Structure, *SIAM J. Matrix Anal. Appl.*, 12, 1, 128–145, 1991.

[CKL] J. F. Canny, E. Kaltofen and Y. Lakssman, Solving Systems of Non-linear Polynomial Equations Faster, *Proc.ACM-SIGSAM Int. Symp. on Symb. and Algebraic Computing*, 121–128, 1989.

[CKLA] J. Chun, T. Kailath, H. Lev-Ari, Fast Parallel Algorithm for QR-Factorization of Structured Matrices, *SIAM J. Sci. Stat. Comput.*,8, 6, 899–913, 1987.

[CS] R. H. Chan, G. Strang, Toeplitz Equations by Conjugate Gradients with Circulant Preconditioner. *SIAM J. Sci. Stat. Comput.*, 10, 1, 104–119, 1989 .

[DH] F. R. deHoog, On the Solution of Toeplitz Systems, *Linear Algebra Appl.*, 88/89, 123–138, 1987.

[GKK] I. Gohhberg, T. Kailath, I. Koltracht, Efficient Solution of Linear Systems of Equations with Recursive Structure, *Linear Algebra Appl.*, 80, 81–113, 1986.

[GKKL] I. Gohhberg, T. Kailath, I. Koltracht, P. Lancaster , Linear Complexity Parallel Algorithms for Linear Systems of Equations with Recursive Structure, *Linear Algebra Appl.*, 88/89, 271–315, 1987.

[GKO] I. Gohberg, T. Kailath, V. Olshevsky, *Fast Gaussian Elimination with Partial Pivoting for Matrices with Displacement Structure*, submitted , 1994.

[GL] G. H. Golub, C.F. VanLoan, *Matrix Computation*, Johns Hopkins Univ. Press, Baltimore, Maryland, 1989.

[GO] I. Gohberg and V. Olshevsky, Complexity of Multiplication with Vectors for Structured Matrices,*Linear Algebra Appl.*, 202, 163–192, 1994.

[GO1] I.Gohberg, V.Olshevsky, Fast Inversion of Chebyshev– Vandermonde Matrices, *Numerical Math.*, 67, 1, 71–92, 1994.

[GO2] I. Gohberg, V. Olshevsky, Complexity of Mutiplication with Vectors for Structured matrices, *Linear Algebra Appl.*, 202, 163–192, 1994.

[GO3] I. Gohberg, V. Olshevsky, Fast Algorithm for Matrix Nehari Problem, *Proceedings of MTNS-93.*

[GO4] I. Gohberg, V. Olshevsky, Fast State Space Algorithms for Matrix Nehari and Nehari-Takagi Interpolation Problems, *Integral equations and Operator Theory*, 1994.

[GS] W. Gentleman, and G. Sande, Fast Fourier Transforms for Fun and Profit, *Proc. AFIPS Fall Joint Comput. Conf.*, 29 (1966), 563–578.

[H] G. Heinig, Inversion of Generalized Cauchy Matrices and the Other Classes of Structured Matrices, *Linear Algebra foe Signal Processing , IMA volume in Math. andIts Applications*, 69, 95–114, Springer, 1994.

[HR] G. Heinig, K. Rost, Algebraic Methods for Toeplitz-like Matrices and Operators, *Operation Theory*, 13, Birkhauser, 1984.

[KKM] T. Kailath, S. Y. Kung, M. Morf, Displacement Ranks of Matrices and Linear Equations, *J. Math. Ana. Appl.*, 68, 2, 395–407, 1978.

[KO] T. Kailath and V. Olshevsky, Displacement Structured approach to Chebyshev–Vandermonde and related Matrices, submitted 1994.

[KS] T. Kailath, A. Sayed, Displacement Structure: Theory and Applications, *SIAM Review*, 1994.

[KVM] T. Kailath, A. Viera, M. Morf, Inverses of Toeplitz Operators, Innovations, and Orthogonal Polynomials, *SIAM Review*, 20, 1, 106–119, 1978.

[LV] E. Linzer, M. Vetterli, Iterative Toeplitz Solvers with Local Quadratic Convergence, *Computing*, 49, 339–347, 1993.

[MORF] M. Morf, Doubling Algorithms for Toeplitz and Related Equations, *Proc, IEEE Internat. Conf. on ASSP*, 954–959, IEEE Computer Society Press, 1980.

[MUS] B. R. Musicus, Levinson and Fast Choleski Algorithms for Toeplitz and Almost Toeplitz Matrices, *Internal Report, M.I.T. Res. Lab. for Electronics*, 1981.

[P90] V. Y. Pan, Computation with Dense Structured Matrices, *Math. of Computation*, 55, 179–190, 1990.

[P90a] V. Pan, Parallel Least-Square Solution of General and Toeplitz-like Linear Systems, *Proc. 2nd Ann. ACM Symp. on Parallel Algorithms and Architecture* (1990), 244–253.

[P92e] Pan, V., Approximate Evaluation of a Polynomial on a Set of Real Points, Tech. Report, International Computer Science Institute, Berkeley, CA (1992).

[P92] V. Y. Pan, Parallel Solution of Toeplitz–like Linear Systems, *J. of Complexity* , 8, 1–21, 1992.

[P93] V. Y. Pan, Concurrent Iterative Algorithms for Toeplitz–like Linear Systems, *IEEE Trans. on Parallel and Distributive Systems*, 4, 5, 592–600, 1993.

[P93a] V.Y.Pan, Decreasing the Displacement Rank of a Matrix, *SIAM J. on Matrix Analysis and Applications*, 14, 1, 118–121, 1993.

[PS] V. Y. Pan, R. Schreiber. A Fast, Preconditioned Conjugate Gradient Solver. *Computer Math. Applic.* 24 , 7, 17–24, 1992.

[Ro85] V. Rokhlin, Rapid Solution of Integral Equations of Classical Potential Theory, *J. of Comput. Physics* **60** (1985), 187-207.

[Ro88] V. Rokhlin, A Fast Algorithm for the Discrete Laplace Transformation, *J. of Complexity* **4** (1988), 12-32.

[S] G. Strang, A Proposal for Toeplitz Matrix Calculations, *Studies in Appl. Math*, 74, 171–176, 1986.

[SCHW] Schwartz, J. T., Fast Probabilistic Algorithms for Verification of Polynomial Identities, *J. of ACM* **27** (1980), 701-717.

[T] W. F. Trench, A Note on a Toeplitz Inversion Formula, *Linear Algebra Appl.*, 29, 55–61, 1990.

[W] D.H. Wood, Product Rules for the Displacement of Near–Toeplitz Matrices,*Linear Algebra Appl.*, 188, 189, 641–663,1993.

[ZIP] R. E. Zippel, Probabilistic Algorithms for Sparse Polynomials, Springer Lecture Notes in Computer Science **72** (1979), 216-226.