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

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

STUDIES IN

ALGORITHMS FOR FAST STRUCTURED MATRICES COMPUTATIONS AND THEIR APPLICATIONS

by

AI LONG ZHENG

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

1998

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

UMI Number: 9820599

Copyright 1998 by
Zheng, Ai Long

All rights reserved.

UMI Microform 9820599
Copyright 1998, by UMI Company. All rights reserved.

This microform edition is protected against unauthorized
copying under Title 17, United States Code.

UMI
300 North Zeeb Road
Ann Arbor, MI 48103

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

© 1998

AI LONG ZHENG

All Rights Reserved

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

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.

10/14/97
Date

Victor Pan
Chair of Examining Committee

10/15/97
Date

Executive Officer

MICHAEL ANSHEL

MYONG-HI KIM

VICTOR Y. PAN

Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Abstract

STUDIES IN ALGORITHMS FOR STRUCTURED

MATRICES COMPUTATIONS AND THEIR APPLICATIONS

by

Ai Long Zheng

Advisor: Professor Victor Y. Pan

Dense structured matrices are special matrices that arise in numerous applications such as control, signal and image processing, coding, partial differential and integral equations, and a variety of algebraic computations. Scaling and displacement operators associated to structured matrices help us exploit the underlying structure of a matrix when we devise fast and numerical reliable algorithms for various computational problems. The underlying characteristic properties of the structured matrices, which distinguish them from unstructured matrices, in particular from general matrices, is the dramatic decrease of the rank of the associated matrices obtained as the images of scaling and displacement operators applied to the given matrices.

Such image matrices are called *scaling* and *displacement generators*. They relate the three parts of this dissertation to each other.

In part I, for a Toeplitz or Toeplitz-like matrix T, we define a preconditioning applied to the matrix $T^H T$. This enables us to accelerate the conjugate gradient algorithm for solving Toeplitz and Toeplitz-like linear systems, thus extending the previous results of [PS].

In part II, we specify some initial assumptions that guarantee rapid refinement of a rough initial approximation to the inverse of a Cauchy-like matrix, by means of our new modifications of Newton's iteration.

Finally, in part III, we extend the algorithm of [PSLT] by using the properties of var-

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

ious classes of structured matrices and the known correlations among them, to solve the problems of multipoint polynomial evaluation and interpolation. Unlike [R88] and [P95], this approach allows complex input points $x_0, \cdots, x_{n-1}$, the nodes of evaluation and interpolation.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Acknowledgements

First and foremost, I would like to thank my principal advisor, Professor Victor Y. Pan, for his consistent support and encouragement. It was a uniquely rewarding personal and educational experience to be part of his highly qualified research group. Under his guidance, I was introduced to a wealth of material that ultimately led to this dissertation. His constant enthusiasm, integrity and creativity will continue to be a model for me in the future.

I would like to express my deep gratitude to my thesis committee members, Professor Michael Anshel and Professor Myong-hi Kim, for their time and helpful advice.

Very special thanks go to Professor Esther Owens, SEEK LAB director at John Jay College of Criminal Justice, for her warm support and encouragement during my work as a college assistant. Without her consistent help and advice in the past three difficult years, this thesis would not have been possible.

Many, many thanks go to my aunt Ching Ngar Lau and my uncle Guochun Lau, who sponsored my studies in the U.S.A, and who gave me helpful advice during these years.

I wish to extend my sincere appreciation to Dr. Xiaohang Huang and to Mr. Steve Yu with whom I have closely collaborated. Thanks for their guidance in the typing of my dissertation.

I owe tremendous debt of gratitudes to my family, my wife, Weifang He, and my son, Steve Zheng, for their unconditional love, devotion, and understanding during this diffcult separation.

I am more than grateful to my sisters and brother for their taking care of our parents in which I should share my responsibility.

Finally, and most importantly, I would like to express my deepest gratitude to my

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

parents who always encouraged me to study.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Contents

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

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

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

1.1 Introduction

We present a new approach to preconditioning of an unsymmetric Toeplitz matrix T, which substantially improves the solution of unsymmetric Toeplitz linear systems of n equations, by means of the conjugate gradient method. The approach also works for the more general class of Toplitz-like linear systems too.

In contrast to the direct Toeplitz solvers using order of the n^2 or $n\log^2 n$ arithmetic operations [T], [CB], [CK], [AG], [BA], [BGY], [D], the conjugate gradient method requires $O(knlogn)$ 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 [CH89], [C91], [C89], [S]), works neither for the unsymmetric nor for Toeplitz-like case, which are also highly important in computational practice.

The present paper gives a desired extention of the algorithm of [PS] to these cases. The extention 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

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

explicit formulae expressing the displacement generators of the inverses of such 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 our presentation, we try to follow the line of [PS]. In the next section, we recall some relevant results on the displacement representation of Toeplitz-like matrices. In section 3 we show a general outline of the method. In section 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 section 5 we specify the more effective solver in the Toeplitz case.

1.2 Some Properties of Toeplitz-like Matrices

Definition 1.2.1 (compare [BP], deifnition 2.11.1) *Let $\nabla : \mathbf{F}^{m\times n} \mapsto \mathbf{F}^{m\times n}$ be an operator, let $A \in \mathbf{F}^{m\times n}$, and let $G \in \mathbf{F}^{m\times l}$, $H \in \mathbf{F}^{n\times l}$ denote two matrices such that $\nabla(A) = GH^T$. Then $l = rank(\nabla(A))$, the rank of the matrix $\nabla(A)$, is called the ∇-rank of A, and the pair of the matrices G and H is called an ∇-generator of A of length l.*

Given a scalar $\phi \neq 0$, an $m \times m$ matrix X and an $n \times n$ matrix Y, define the operator $\nabla_{(X,Y)}(A) = A - XAY$ and specify a displacement operator of Toeplitz-type as follows.

$$\nabla(A) = \nabla_{(Z_\phi, Z_{\frac{1}{\phi}}^T)}(A) = A - Z_\phi A Z_{\frac{1}{\phi}}^T, \tag{1.2.1}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$Z_\phi = \begin{pmatrix} 0 & \cdots & 0 & \phi \\ 1 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & 0 \end{pmatrix}.$$

Definition 1.2.2 *An $m \times n$ matrix is called a Toeplitz-like matrix if it has ∇-rank bounded from above by a constant independent of m and n, where ∇ is the operator defined in (1.2.1).*

Hereafter, let $\phi = 1$, $Z = Z_1$. We have the following basic lemmas.

Lemma 1.2.1 [BP] *Let $A \in \mathbf{F}^{n\times n}$, $B \in \mathbf{F}^{m\times m}$ be two Toeplitz-like matrices given with their ∇-generators of lengths l_A and l_B, respectively. Then AB is a Toeplitz-like matrix having an ∇-generator of length $l_{AB} \leq l_A + l_B$.*

Proof: follows from the observation that $\nabla(AB) = \nabla(A)B + ZAZ^T\nabla(B)$.

Lemma 1.2.2 (compare [KKM], [BP], [GO]) *Let A be a nonsingular Toeplitz-like matrix with an ∇-generator $\nabla(A) = G_1 H_1^T$, of length l_A. Then A^{-1} is a Toeplitz-like matrix with an ∇-generator equal to GH^T, where $G = -A^{-1}G_1$, $H^T = H_1^T Z A^{-1} Z^T$.*

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

Corollary 1.2.1 *Let T be an $n \times n$ Toeplitz-like matrix with an ∇-generator of length l_T. Then $B = T^H T + \mu I$, $C = I - \mu B^{-1}$ are Toeplitz-like matrices with $l_B \leq 2l$ and $l_C \leq 2l$, provided that $-\mu$ is not an eigenvalue of $T^H T$.*

Definition 1.2.3[BP]. *An $m \times n$ matrix $Circ_\phi(r) = Circ_{(\phi,m,n)}(r) = [Z_{i,j}]$, for a vector $r = [r_0, \cdots, r_{m-1}]^T$ and for a scalar $\phi \neq 0$, is called an ϕ-circulant matrix if $z_{i,j} = r_{(i-j) mod\ m}$ for $i \geq j$; $z_{i,j} = \phi r_{(i-j) mod\ m}$ for $i < j$.*

Hereafter, l will stand for l_T.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

1.3 A Condition-Improving Matrix Factorization

Lemma 1.3.1 [PS] *Let A be an $n \times n$ nonsingular 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

$$\alpha_n \leq \alpha_{n-1} \leq \cdots \leq \alpha_1 = \lambda(A),$$

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

$$\gamma_n \leq \gamma_{n-1} \leq \cdots \leq \gamma_1 = \lambda(C).$$

By the definition of B and C, we have

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

Lemma 1.3.2 [PS] *Let A, B, and C be as above and let $\mu > 0$. Then the condition numbers of B and C are given by*

$$k(B) = \frac{\alpha_1 + \mu}{\alpha_n + \mu} \tag{1.3.1}$$

and

$$k(C) = \frac{\alpha_1}{\alpha_n}\left(\frac{\alpha_n + \mu}{\alpha_1 + \mu}\right), \tag{1.3.2}$$

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

$$k(A) = k(B)k(C). \tag{1.3.3}$$

Lemma 1.3.3 [PS] *Let $\mu = \sqrt{\alpha_1\alpha_n}$. Then $k(B) = k(C) = \sqrt{k(A)}$.*

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

1.4 A Fast Toeplitz-like Solver

Consider the linear system

$$Tx = b, \tag{1.4.1}$$

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

$$T^H T x = T^H b. \tag{1.4.2}$$

Let $A = T^H T$, then A is an $n \times n$ h.p.d. Toeplitz matrix, $l_A \leq 2l$. 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 $l_B \leq 2l$ and $l_C \leq 2l$. By the results of [GO], B^{-1} is completely defined by its last row and its ∇-generator:

$$B^{-1} = Circlr + \frac{1}{1-\phi} \sum_{m=1}^{2l} Circ_{\phi}(r_m) Circ_1(s_m^T), \tag{1.4.3}$$

where ϕ is arbitrary, $\phi \neq 1$, $Circl_r$ is the 1-circulant matrix with the last row equal to y^T. Furthermore, r_m, s_m and y^T satisfy following equations:

$$Br_m = g_m, \tag{1.4.4}$$

$$Bt_m = -Z_1^T h_m, \tag{1.4.5}$$

$$s_m = -Z_1^T t_m, \ m = 1, 2, \cdots, 2l, \tag{1.4.6}$$

$$By = e_{n-1}, \ e_{n-1} = (0, 0, \cdots, 1)^T, \tag{1.4.7}$$

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

Algorithm 1.1

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

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

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Stage 1: *solve the equations (1.4.4), (1.4.5), (1.4.6) and (1.4.7).*

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 $2l + 1$ multiplications of f-circulant matrices by vectors for $f = 1$ and $f = \phi$ [see the representation (1.4.3)]. Therefore, by the well known results (see e.g. [GO]), the arithmetic cost of performing stage 1, i.e. the arithmetic cost of performing n_B steps of the CG iteraton on B, equals

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

and similarly at stage 3, we have

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

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

1.4.1 The optimal shift

We will next follow [PS] by chosing the optimal μ such that the total work $[(4l + 1)(4l + 3)n_B + (4l + 3)n_C]\phi(n)$ is minimized, where n_B and n_C are the numbers of steps of the CG iteration at stages 1 and 3, respectively. Let

$$n_B = F\sqrt{k(B)}, \tag{1.4.8}$$

$$n_C = F\sqrt{k(C)}, \tag{13}$$

where F is a constant. Then by (1.3.3),

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

Define

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

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

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

$$n_B = \sqrt{\frac{M}{L}},\ n_C = Ln_B. \tag{1.4.9}$$

In view of (1.4.8)-(1.4.10), we choose μ satisfying

$$k(C) = L^2 k(B). \tag{1.4.11}$$

Use (1.3.1), (1.3.2) and let $\mu = m\sqrt{\alpha_1\alpha_n}$. We have the following equation:

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

so

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

where $k(A) = \frac{\alpha_1}{\alpha_n}$, $L = 4l + 1$. Since $L \geq 5$, $k(A) \geq 1$, we have $m_- > 0$ only for $k(A) > L^2$.

Lemma 1.4.1 [PS]. *Let $\mu = m\sqrt{\alpha_1\alpha_n}$, where $m = m_-$ (see above). Then*

$$k(B) = L^{-1}\sqrt{k(A)}, \tag{1.4.12}$$

$$k(C) = L\sqrt{k(A)}. \tag{1.4.13}$$

Now assume (1.4.10) and choose $\mu = m_-\sqrt{\alpha_1\alpha_n}$. Then the total cost is

$$\begin{aligned}
&(4l + 3)[(4l + 1)n_B + n_C]\phi(n) \\
&= (4l + 3)(Ln_B + n_C)\phi(n) \\
&= 2(4l + 3)F\sqrt{k(C)}\phi(n) \\
&= 2(4l + 3)\sqrt{4l + 1}k^{\frac{1}{4}}F\phi(n).
\end{aligned} \tag{1.4.14}$$

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

$$Cost(CG) = (4l + 3)n_{CG}\phi(n) = (4l + 3)k^{\frac{1}{2}}(A)F\phi(n). \tag{1.4.15}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Comparing with (1.4.14) we can see an improvement for $k(A) > 16(4l+1)^2$.

1.4.2 Recursive Preconditioning

We may use the factorization $A = T^H T = BC$ recursively. In particular, we may solve equation (1.4.4), (1.4.5) and (1.4.7) at stage 1 of algorithm 1.1 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 1.1. Since we have $l_B \leq 2l$, $l_C \leq 2l$ (where l_W denotes the length of an ∇-generator of W, for $W = B$, $W = C$), it follows from (1.4.14), that the total computational cost of performing stages 1 and 3 is bounded by

$$2(8l+1)(8l+3)\sqrt{8l+1}k^{\frac{1}{4}}(B)F\phi(n) \tag{1.4.16}$$

and

$$2(8l+3)\sqrt{8l+1}k^{\frac{1}{4}}(C)F\phi(n), \tag{1.4.17}$$

respectively. Now we choose μ so as to minimize the sum of (1.4.16) and (1.4.17). Since $k(A) = k(B)k(C)$, we have the solution $k(B) = \frac{k^{\frac{1}{2}}(A)}{(8l+1)^2}$, $k(C) = (8l+1)^2 k^{\frac{1}{2}}(A)$,

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

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

$$4(8l+1)(8l+3)F\phi(n)k^{\frac{1}{8}}(A) \tag{1.4.18}.$$

This is less than the cost (1.4.14) of non-recursive preconditioning for

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

and is also less than the cost of application of the unpreconditioned (CG) method to $Ax = b$ (see (1.4.15)) when $k(A) > [\frac{4(8l+1)(8l+3)}{(4l+3)}]^{\frac{8}{3}}$.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

For $l = 2,\ 3$, we compute the estimates (1.4.14), (1.4.15) and (1.4.18) and show the results in the following. For $l = 2$,

$$CG\ method = 11k^{\frac{1}{2}}(A)F\phi(n),$$

$$non - recursive = 66k^{\frac{1}{4}}(A)F\phi(n),$$

$$recursive = 1292k^{\frac{1}{8}}(A)F\phi(n).$$

For $l = 3$,

$$CG\ method = 15k^{\frac{1}{2}}(A)F\phi(n),$$

$$non - recursive = 30\sqrt{13}k^{\frac{1}{4}}(A)F\phi(n),$$

$$recursive = 2700k^{\frac{1}{8}}(A)F\phi(n).$$

1.5 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 (l=2). Since $B = T^HT + \mu I$, multiplying the matrix B by a vector cost $8\phi(n) + O(n)$. 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

$$(72n_B + 11n_C)\phi(n) = 11(qn_B + n_C)\phi(n),\ q = 72/11,$$

where n_B and n_C denote the number of the CG iterations at stages 1 and 3, respectively. Assume the optimal value of $\mu = m_-\sqrt{\alpha_1\alpha_n}$, where

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

Then, similarly to (1.4.13), we derive the following cost bound for the entire computation:

$$22n_C\phi(n) = 12\sqrt{22}k^{\frac{1}{4}}(A)F\phi(n). \tag{1.5.1}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

We may compare the bound of (1.5.1) to the cost of the solution via the CG method (without preconditioning), which is estimated similarly to (1.4.15) and is bounded by

$$8k^{\frac{1}{2}}(A)F\phi(n). \tag{1.5.2}$$

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

$$108\sqrt{22}k^{\frac{1}{4}}(B)F\phi(n), \tag{1.5.3}$$

At stage 3, choose μ_2 so as to decrease the cost to

$$4620k^{\frac{1}{8}}(C)F\phi(n), \tag{1.5.4}$$

[compare (1.4.18)]. Now we choose μ so as to minimize the sum of (1.5.3) and (1.5.4). Then we obtain that

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

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

and the overall cost is bounded by

$$Ek^{\frac{1}{12}}(A)F\phi(n), \tag{1.5.5}$$

where

$$E = 400,993.268...$$

[compare (1.4.18)]. Therefore, the recursive method is superior to the nonrecursive method only if $k(A)$ is large enough $k(A) > (\frac{E}{12\sqrt{22}})^6$. We also compare (1.5.5) and (1.5.2) and conclude taht the recursive method improves the unpreconditioned CG method only for extremely large $k(A)$, $k(A) > (\frac{E}{8})^{\frac{12}{5}}$.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2 Newton's Iteration for Inversion of Cauchy-like and Other Structured Matrices

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2.1 Introduction

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 "short" *generators* associated with some operators of displacement (shift) and/or scaling [KVM], [KKM], [CKL-A], [GKK], [GKK1], [HR], [P90], [GO], [BP], [H], [GKO], [GO1], [KO], [KS].

Such a generator is a pair (G, H) of matrices associated with a given structured $n \times n$ matrix A, satisfying the following matrix equations,

$$\Delta_{\{R,S\}}(A) = A - RAS = GH^T \tag{2.1.1}$$

or

$$\nabla_{\{R,S\}}(A) = RA - AS = GH^T, \tag{2.1.2}$$

for a fixed natural α, called the *length* of the generator, for two fixed matrices R and S, representing scaling and/or displacement, and for some $n \times \alpha$ matrices G and H. Here and hereafter, W^T denotes the transpose of a vector or a matrix W, and we note that the rank of the matrix GH^T does not exceed the length α of the generator G, H. We also note that the operators $\Delta_{\{R,S\}}(A)$ and $\nabla_{\{R,S\}}(A)$ are closely related to each other if the matrices R and/or S are nonsingular. In this paper, we will only deal with $\nabla_{\{R,S\}}$-generators of A, of the form (2.1.2), except for one case in section 2.6, and will specify R and S according to the equations (2.1.3) – (2.1.5) below. Scaling is represented by diagonal matrices

$$D_x = diag(\, x_1, x_2, \cdots, x_n \,), \tag{2.1.3}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

for fixed $x_1, x_2, \cdots, x_n$, whereas such matrices as

$$Z_f = \begin{pmatrix} 0 & 0 & \cdots & 0 & f \\ 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix}, \tag{2.1.4}$$

or

$$aZ_f + bZ_\varphi^T + cZ_\psi^{-1}, \tag{2.1.5}$$

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

The key-idea is that the original matrix A can be easily recovered from its generators (2.1.1) or (2.1.2); 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 that are 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 (2.1.1) and (2.1.2), for some appropriate choices of the matrices R and S among the matrices of the classes (2.1.3) and (2.1.4), is as small as 1 or 2. The three 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 circulant, Sylvester, subresultant, Hankel, Hankel-like, Lowener, Bezout, and Chebyshev-Vandermonde matrices [HR], [BP], [H], [GO1], [KO].) It was observed in [P90] that some correlations among the operators associated with the matrices of the three cited classes can be exploited in order to reduce the computations for matrices of any of the three classes to computations with matrices of the class for which most effective algorithms are available.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

In their original form, the reductions proposed in [P90] were not convenient for practical implementation because they involved operations with Vandermonde matrices, and such matrices are generally known to be ill-conditioned [GI].

On the other hand, in the important case of the transition from Toeplitz-like and Vandermonde-like matrices to Cauchy-like matrices, the reduction is dramatically simplified and becomes practically effective because, in this case, the Vandermonde matrices turn into the matrices of the discrete Fourier transforms, [H] and [GKO], and the reduction to Cauchy-like computations is performed via FFT, that is, in a fast and numerically stable way (cf. [BP]). Thus, effective algorithms for computations with Cauchy-like matrices should play most fundamental role.

In the present paper, we consider the solution of a nonsingular Cauchy-like linear system, $C\vec{x} = \vec{v}$, and the inversion of a nonsingular Cauchy-like matrix C. These operations can be immediately extended to the Vandermonde-like and Toeplitz-like cases, as well as to the Chebyshev-Vandermonde cases (see our section 2.5 or [GKO], [GO1], [H] and [KO]).

It is well-known that Newton's iteration rapidly improves 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], and [P93a], Newton's iteration has been modified so as 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 SVDs of the displacement matrices defined by such generators. This made all the iteration steps of the resulting algorithms computationally simple; moreover, such a simplification was achieved with no significant slowdown of the convergence, except for the initial stages, where the increase of

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

the approximation errors due to the latter chopping posed additional serious requirements on the quality of the initial approximation.

In the present part, 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. Our solution of 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]. In particular, we do not need to compute the SVDs of the auxiliary matrices and to chop-off the smallest singular values. Besides the resulting computational saving at each Newton's stage, this enables us to relax the respective constraints on the choice of the initial approximation. The cited advantages should motivate the reduction of the 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], and [P93a].

In spite of some facilitation of the choice of the initial approximation that we achieved in the Cauchy-like case (versus the Toplitz-like case), such a choice remains an open problem for the general Cauchy-like input, and we discuss this problem in our last section 2.7.

Otherwise, we present our results in the following order. In the next section, we describe our modification of a Newton's iteration for the refinement of an initial approximation to the inverse of a Cauchy-like matrix. In short section 2.3, we estimate the computational cost of each iteration step, performed by operating with short displacement generators of matrices, rather than with the matrices themselves. In section 2.4, we quantitatively specify the assumptions about the error norm of the initial approximation to the inverse that guarantee rapid convergence of our modification of Newton's iteration, and we also specify the number of iteration steps sufficient for convergence to an approximation within a fixed

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

output error bound. In section 2.5, we recall the reduction of the inversion of Toeplitz-like, Vandermoode-like, and Chebyshev-Vandermonde-like matrices to the inversion of Cauchy-like matrices, which enables us to extend our results for a Cauchy-like input to ones for Toeplitz-like, Vandermonde-like, and Chebyshev-Vandermonde-like inputs. In section 2.6, we recall an alternative approach to approximating C^{-1}, based on reduction of the problem to Toeplitz-like computations, and point out some major deficiences of this approach.

2.2 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

$$\nabla_{\{D_t,D_s\}}(C) = D_t C - C D_s = ZY^T , \tag{2.2.1}$$

$$C = \left[\frac{z_i^T y_j}{t_i - s_j} \right], \qquad Z^T = [\, z_1, \cdots, z_n \,], \qquad Y^T = [\, y_1, \cdots, y_n \,] , \tag{2.2.2}$$

where $z_i,\ y_j \in \mathbf{C}^{\alpha \times 1}$, $D_t = diag(\ t_1, \cdots, t_n\ )$, $D_s = diag(\ s_1, \cdots, s_n\ )$, $t_i \neq s_j$, for all $i,\ j = 1, 2, \cdots, n$ (cf. [H] and [GKO]). Then, we have the following fundamental result, showing that the inverse matrix C^{-1} is also a Cauchy-like matrix and relating its scaling generator to one of C.

Theorem 2.2.1 [H] *For a Cauchy-like matrix C satisfying* (2.2.1) *and* (2.2.2), *we have*

$$C^{-1} = -\left[\frac{u_i^T w_j}{s_i - t_j} \right] , \tag{2.2.3}$$

$$U = [\, u_1^T, \cdots, u_n^T \,]^T = C^{-1} Z , \qquad W^T = [\, w_1, \cdots, w_n \,] = Y^T C^{-1} , \tag{2.2.4}$$

where $Z^T = [\, z_1, \cdots, z_n \,]$ *and* $Y^T = [\, y_1, \cdots, y_n \,]$.

Note the reversion of the order of scaling operators: the opereator $\nabla_{\{D_t,D_s\}}$, which we associate to the matrix C corresponds to the operator $\nabla_{\{D_s,D_t\}}$, which we associate to the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

inverse C^{-1}. Assume that an initial approximation X_0 to C^{-1} is available with its $\nabla_{\{D_s,D_t\}}$-generator of a length at most α. Then, we recursively define matrices $X_1^{=}$, X_1, $X_2^{=}$, X_2, $\cdots$ as follows:

$$X_{k+1}^{=} = X_k \, (\, 2I - CX_k \,), \qquad k = 0, 1, \cdots, \tag{2.2.5}$$

$$X_{k+1} = - \left[\frac{(u_i^{k+1})^T w_j^{k+1}}{s_i - t_j} \right]_{i,j=1}^{n}, \qquad k = 0, 1, \cdots, \tag{2.2.6}$$

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

$$U_{k+1} = \begin{pmatrix} (u_1^{k+1})^T \\ \vdots \\ (u_n^{k+1})^T \end{pmatrix} = X_{k+1}^{=} Z \, , \qquad Z = \begin{pmatrix} z_1^T \\ \vdots \\ z_n^T \end{pmatrix} \, , \tag{2.2.7}$$

$$W_{k+1}^T = [\, w_1^{k+1}, \cdots, w_n^{k+1} \,] = Y^T X_{k+1}^{=} \, , \qquad Y^T = [\, y_1, \cdots, y_n \,] \, . \tag{2.2.8}$$

Equation (2.2.5) represents a step of Newton's iteration for matrix inversion (cf. e.g. [BP]), and equation (2.2.6) "corrects" the results $X_{k+1}^{=}$ of (2.2.5), so as to turn $X_{k+1}^{=}$ into a Cauchy-like matrix, associated with $\nabla_{\{D_s,D_t\}}$, that is , with the same scaling operator 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 2.2.1 *For all $k = 0, 1, \cdots$, the matrices $X_{k+1}^{=} = 2X_k - X_k C X_k$ are Cauchy-like matrices whose $\nabla_{\{D_s,D_t\}}$-generators have lengths of at most 3α.*

Proof: By the definition of the ∇- operator, we have

$$\nabla_{\{D_s,D_t\}}(X_k C X_k) = D_s X_k C X_k - X_k C X_k D_t \, , \tag{2.2.9}$$

$$\nabla_{\{D_s,D_t\}}(X_k) C X_k = (D_s X_k - X_k D_t) C X_k = D_s X_k C X_k - X_k D_t C X_k \, , \tag{2.2.10}$$

$$X_k \nabla_{\{D_t,D_s\}}(C) X_k = X_k (D_t C - C D_s) X_k = X_k D_t C X_k - X_k C D_s X_k \, , \tag{2.2.11}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$X_kC\nabla_{\{D_s,D_t\}}(X_k) = X_kC(D_sX_k - X_kD_t) = X_kCD_sX_k - X_kCX_kD_t \ . \qquad (2.2.12)$$

By combining (2.2.10), (2.2.11) and (2.2.12), we obtain that

$$\nabla_{\{D_s,D_t\}}(X_kCX_k) = \nabla_{\{D_s,D_t\}}(X_k)CX_k + X_k\nabla_{\{D_t,D_s\}}(C)X_k + X_kC\nabla_{\{D_s,D_t\}}(X_k) \ .$$

Therefore,

$$\begin{aligned}
\nabla_{\{D_s,D_t\}}(X_{k+1}^{=}) &= \nabla_{\{D_s,D_t\}}(\ 2X_k - X_kCX_k\ ) \\
&= 2\nabla_{\{D_s,D_t\}}(X_k) - \nabla_{\{D_s,D_t\}}(X_kCX_k) \\
&= 2\nabla_{\{D_s,D_t\}}(X_k) - \nabla_{\{D_s,D_t\}}(X_k)CX_k - X_k\nabla_{\{D_t,D_s\}}(C)X_k - X_kC\nabla_{\{D_s,D_t\}}(X_k) \\
&= \nabla_{\{D_s,D_t\}}(X_k)(2I - CX_k) - X_k\nabla_{\{D_t,D_s\}}(C)X_k - X_kC\nabla_{\{D_s,D_t\}}(X_k) \ .
\end{aligned}$$

We have

$$\nabla_{\{D_s,D_t\}}(X_k) = G_kH_k^T \ , \qquad \nabla_{\{D_t,D_s\}}(C) = GH^T \ ,$$

where $G_k,\ H_k,\ G,\ H\ \in \mathbf{C}^{n\times\alpha}$. Therefore, $\nabla_{\{D_s,D_t\}}(X_{k+1}^{=}) = G_{k+1}^{=}(H_{k+1}^{=})^T$, where

$$\begin{aligned}
G_{k+1}^{=} &= [\ G_k,\ -X_kG,\ -X_kCG_k\ ] \in \mathbf{C}^{n\times 3\alpha}, \\
H_{k+1}^{=} &= [\ (\ 2I - CX_k\ )^TH_k,\ X_k^TH,\ H_k\ ] \in \mathbf{C}^{n\times 3\alpha}. \qquad \square
\end{aligned}$$

2.3 Computational Complexity of an Iteration Step

Next, we will estimate the arithmetic cost of computing the matrices X_{k+1}, U_{k+1}, and W_{k+1} (the latter pair of matrices being the $\nabla_{\{D_s,D_t\}}$-generator for $X_{k+1}^{=}$), based on the equations (2.2.5)-(2.2.8). Given Cauchy-like matrices C and X_k, the computation of the $\nabla_{\{D_s,D_t\}}$-generator of length 3α for $X_{k+1}^{=}$ [according to (2.2.5)] uses $O(\alpha^2 n\log^2 n)$ ops (see e.g. [GO] or [BP], chapter 2, sections 4, 11, and 12). (Here and hereafter, "*ops*" stands for "arithmetic operations".) Therefore, (2.2.7) and (2.2.8) 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

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

of this computation is the economization of computer memory, that is, representation of all involved matrices by means of their short generators requires 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.

2.4 Estimating Convergence Rate of Newton's Iteration

In the following, we will estimate how fast X_k approaches C^{-1}. We recall from (2.2.6)–(2.2.8) and (2.2.4) that

$$\begin{aligned} X_k &= -\left[\frac{(u_i^k)^T w_j^k}{s_i - t_j} \right], \\ U_k &= X_k^* Z = (\, X_k^* - C^{-1}\,)Z + C^{-1}Z = (\, X_k^* - C^{-1}\,)Z + U\,, \\ W_k^T &= Y^T X_k^* = Y^T(\, X_k^* - C^{-1}\,) + W^T\,. \end{aligned}$$

Then, we obtain the following matrix equation:

$$U_k W_k^T = (X_k^* - C^{-1})ZY^T(X_k^* - C^{-1}) + UW^T + UY^T(X_k^* - C^{-1}) + (X_k^* - C^{-1})ZW^T.$$

We deduce from this equation and (2.2.4) that

$$\begin{aligned} E_k &= U_k W_k^T - UW^T \\ &= (X_k^* - C^{-1})ZY^T(X_k^* - C^{-1}) + C^{-1}ZY^T(X_k^* - C^{-1}) + (X_k^* - C^{-1})ZY^T C^{-1}. \end{aligned} \tag{2.4.1}$$

Hereafter, we will use the column-norm of matrices, writing

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

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

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Proposition 2.4.1 *Let* $e_k^{\blacksquare} = \| X_k^{\blacksquare} - C^{-1} \|$. *Then*

$$\|E_k\| = \| U_k W_k^T - UW^T \| \leq \|ZY^T\| e_k^{\blacksquare}(e_k^{\blacksquare} + 2\|C^{-1}\|) .$$

Proof: Proposition 2.4.1 immediately follows from (2.4.1) . □

Proposition 2.4.2 *Let* $e_k = \| X_k - C^{-1} \|$, *for a nonsingular matrix* C, *and let* $X_{k+1}^{\blacksquare}$ *be defined by* (2.2.5), *for* $k = 0, 1, 2, \cdots$. *Then we have*

$$e_{k+1}^{\blacksquare} \leq \|C\| e_k^2 . \tag{2.4.2}$$

Proof: Due to (2.2.5), we have

$$I - CX_{k+1}^{\blacksquare} = (I - CX_k)^2, \qquad k = 0, 1, 2, \cdots .$$

It follows that

$$\begin{aligned} e_{k+1}^{\blacksquare} &= \|X_{k+1}^{\blacksquare} - C^{-1}\| \\ &= \|C^{-1}(I - CX_{k+1}^{\blacksquare})\| \\ &= \|C^{-1}(I - CX_k)^2\| \\ &= \|C^{-1}(I - CX_k)CC^{-1}(I - CX_k)\| \\ &= \|(C^{-1} - X_k)C(C^{-1} - X_k)\| \\ &\leq \|C\| e_k^2 . \end{aligned}$$ □

Proposition 2.4.3 *For any* $k = 1, 2, \cdots$, *we have* $e_k \leq h_k\, e_k^{\blacksquare}$, $e_{k+1}^{\blacksquare} \leq (h_k e_k^{\blacksquare})^2 \|C\|$, *where*

$$h_k = \rho\|ZY^T\|(e_k^{\blacksquare} + 2\|C^{-1}\|), \qquad \rho = \max_{i,j} \frac{1}{|s_i - t_j|} . \tag{2.4.3}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Proof: We recall (2.2.3),(2.2.4),(2.2.6),(2.4.1), and proposition 4.1 and obtain that

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

for h_k of (2.4.3). Combining the latter bound on e_k with (2.4.6) gives us proposition 4.3. □

Proposition 2.4.4 *If*

$$e_1^* \leq 1 \tag{2.4.4}$$

and if

$$(e_1^*)^\theta ||C|| h_1^2 \leq 1 , \tag{2.4.5}$$

for $\theta \leq 1$ *and for* h_1 *of proposition* 4.3, *then*

$$e_{k+1}^* \leq (e_k^*)^{2-\theta} \leq \cdots \leq (e_1^*)^{(2-\theta)^k} , \qquad \text{for} \quad k = 1, 2, \cdots . \tag{2.4.6}$$

Proof: By the virtue of propositon 4.3, we have

$$e_2^* \leq (e_1^*)^2 ||C|| h_1^2 .$$

Combine this bound with (2.4.5) and obtain that

$$e_2^* \leq (e_1^*)^{2-\theta} . \tag{2.4.7}$$

Combine the latter inequality and (2.4.4), and since $\theta \leq 1$, $2 - \theta \geq 1$, obtain the bounds

$$e_2^* \leq e_1^* \leq 1 , \tag{2.4.8}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

which extend (2.4.4) . Substitute the first inequality of (2.4.8) into (2.4.3) and obtain that $h_2 \leq h_1$. Substitute the latter bound and the bound $e_2^* \leq e_1^*$ of (2.4.8) into (2.4.5) and obtain that $(e_2^*)^\theta ||C|| h_2^2 \leq 1$, which extends (2.4.5). Inductive application of this argument enables us to extend (2.4.4), (2.4.5), and (2.4.7) to the bounds

$$e_k^* \leq 1 \ , \qquad (e_k^*)^\theta ||C|| h_k^* \leq 1 \ , \qquad e_k^* \leq (e_{k-1}^*)^{2-\theta}$$

for $k = 3, 4, \cdots$, and we arrive at (2.4.6). □

We will next restate proposition 2.4.4, by replacing e_1^* by $e_0^2||C||$, based on proposition 2.4.2 for $k = 0$.

Proposition 2.4.5 *If* $e_0\sqrt{||C||} \leq 1$ *and if*

$$e_0^{2\theta} ||C||^{1+\theta} h^2 \leq 1 \ , \tag{2.4.9}$$

for $\theta \leq 1$, *for* $h = \rho ||ZY^T|| (e_0^2||C|| + 2||C^{-1}||)$, *and for* ρ *of* (2.4.3), *then*

$$e_{k+1}^* \leq (e_0^2||C||)^{(2-\theta)^k} \ . \qquad \text{for} \quad k = 0, 1, \cdots .$$

Proof: Proposition 2.4.2 for $k = 0$ implies that

$$e_1^* \leq ||C|| e_0^2 \ . \tag{2.4.10}$$

Therefore, the bound $e_1^* \leq 1$ of (2.4.4) holds if $e_0\sqrt{||C||} \leq 1$, and we also have that $h_1 \leq h$ for h_1 of (2.4.3). Combining the latter bound, (2.4.9), and (2.4.10) gives us (2.4.5). Therefore, the assumptions of proposition 4.5 imply the ones of proposition 4.4 and, consequently, imply (2.4.6). Substitute (2.4.10) into (2.4.6) and obtain proposition 4.5. □

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,

$$e_0 \leq r_0 ||C^{-1}|| \ , \quad \text{where} \quad r_0 = ||I - X_0 C|| \ ,$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

and

$$||C^{-1}|| \le \frac{||X_0||}{1 - r_0} \quad \text{if} \quad r_0 < 1 \ .$$

(The latter implication immediately follows from the next inequalities:

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

Computation of X_0C amounts to n multiplications of a Cauchy-like matrix by vectors, which takes $O((\alpha^2 n \log n)^2)$ ops (cf. [GO] or [BP], section 4, 11 and 12 of chapter 2). For any vector $v \neq 0$, we may compute a lower bound $n(v) = ||(I - X_0C)v||/||v||$ in $O(\alpha^2 n \log^2 n)$ ops, where $n(v) \le ||I - X_0C||$. (For a random choice of one or several vectors v, such lower bounds may give us a reasonably good approximation to $||I - X_0C||$.)

Next, we are going to substitute the above estimate for e_0 into the statement of proposition 4.5. We write

$$e_0^+ = \frac{r_0||X_0||}{1 - r_0} \ , \tag{2.4.11}$$

$$h_{k+1}^+ = \rho||ZY^T|| \ [\ (\ (e_0^+)^2||C|| \ )^{(2-\theta)^k} + \frac{2||X_0||}{1 - r_0} \ ] \ , \quad k = 0, 1, \cdots \ , \tag{2.4.12}$$

so that $e_0^+ \ge e_0$ and $h_{k+1}^+ \ge h_{k+1}$ if $h_{k+1}^- \le (\ (e_0^+)^2||C|| \ )^{(2-\theta)^k}$, and

$$h_{k+1}^+ \le \rho||ZY^T|| \ (\ 1 + \frac{2||X_0||}{1 - r_0} \ ) \ , \quad \text{for all } k, \quad \text{if} \quad e_0^+\sqrt{||C||} \le 1 \ . \tag{2.4.13}$$

We now summarize our results, including the bound $e_{k+1} \le h_{k+1}e_{k+1}^-$ (from proposition 2.4.3), which enables us to estimate e_{k+1} as soon as we estimate e_{k+1}^- .

Corollary 2.4.1 *Let*

$$r_0 = ||I - CX_0|| < 1, \quad e_0^+\sqrt{||C||} \le 1, \quad \theta \le 1, \tag{2.4.14}$$

$$(e_0^+)^{2\theta}||C||^{1+\theta} \ (h_1^+)^2 \le 1, \tag{2.4.15}$$

for e_0^+ *of* (2.4.11) *and* h_1^+ *of* (2.4.12). *Then we have*

$$e_{k+1}^- \le (\ (e_0^+)^2||C|| \ )^{(2-\theta)^k}, \qquad k = 0, 1, \cdots,$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$e_{k+1} \leq h_{k+1}^{+} e_{k+1}^{-}, \qquad k = 0, 1, \cdots .$$

Let us write

$$k^{-} = \lceil \, (\log \frac{\log \epsilon}{\log(\, (e_0^{+})^2 ||C|| \,)} \,) \, / \log(2 - \theta) \, \rceil \, , \tag{2.4.16}$$

where $\theta < 1$, e_0^{+} and h_{k+1}^{+} are defined by (2.4.11) and (2.4.12), so that (2.4.13) holds. Then, under the assumptions (2.4.14) and (2.4.15) of corollary 4.1, it suffices to perform $k + 1 \geq k^{-} + 1$ recursive steps of the iteration (2.2.5), (2.2.6) in order to ensure that

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

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

[Note that (2.4.14) implies, in particular, the bounds of (2.4.13).]

2.5 Extension to the Inversion of Toeplitz-like, Vandermonde-like, and Chebyshev-Vandermonde-like Matrices

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

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

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

$$\nabla_{\{D_{1/x}, D_1^{*}\}}(VF^{-}) = G H^T \, ,$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

where Z_1 is defined by (2.1.4) for $f = 1$,

$$Z_1 = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 1 & 0 & 0 & \cdots & 0 \end{pmatrix},$$

$$H^T = B^T F^*, \qquad D_{1/x} = diag\left(\frac{1}{x_1}, \cdots, \frac{1}{x_n} \right),$$

$$D_1^* = diag\left(1, e^{-\frac{2\pi i}{n}}, \cdots, e^{-\frac{2\pi i}{n}(n-1)} \right), \qquad F = \frac{1}{\sqrt{n}} \left[e^{\frac{2\pi i}{n}(k-1)(j-1)} \right]_{k,j=1}^{n},$$

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

Let X_0, C, and θ satisfy the assumptions of corollary 2.4.1. Let $\epsilon = \epsilon^*/\sqrt{n}$ and let $X_{k^*}^*$ denote the matrix obtained in $k^* + 1$ steps (2.2.5), (2.2.6) [for k^* defined by (2.4.16)] at the arithmetic computational cost $O(k^* \beta^2 n \log^2 n)$. Then we have

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

Similarly, consider a Chebyshev-Vandermonde-like matrix R, which has generators G, B such that

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

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

$$\nabla_{\{2D_x, D_{cos}\}}(RF^*) = 2D_x RF^* - RF^* D_{cos} = GB^T F^*,$$

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

$$D_{cos} = diag\left(2, 2cos(\tfrac{\pi}{n}), \cdots, 2cos(\tfrac{(n-1)\pi}{n}) \right).$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

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 2.5.1 [GKO]. *Let $T \in \mathbf{C}^{n\times n}$ be a Toeplitz-like matrix, such that*

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

where $G \in \mathbf{C}^{n\times\beta}$ and $B \in \mathbf{C}^{n\times\beta}$. 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,

$$D_1 = diag(\ 1, e^{\frac{2\pi i}{n}}, \cdots, e^{\frac{2\pi i}{n}(n-1)}\ ), \qquad D_0 = diag(\ 1, e^{\frac{\pi i}{n}}, \cdots, e^{\frac{(n-1)\pi i}{n}}\ ),$$

$$D_{-1} = diag(e^{\frac{\pi i}{n}}, e^{\frac{3\pi i}{n}}, \cdots, e^{\frac{(2n-1)\pi i}{n}}), \quad \hat{G} = FG\ , \quad and \quad \hat{H}^T = B^TD_0^{-1}F^*\ .$$

Suppose that X_0, $C = FTD_0^{-1}F^*$, and θ satisfy the assumptions of corollary 4.1, let $\epsilon = \epsilon^*/\sqrt{n}$, and let $X^*_{k^*+1}$ denote the matrix obtained in k^*+1 steps (2.2.5), (2.2.6), for k^* of (2.4.16) and such that

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

Then, we have

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

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

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

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

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

$$|| T^{-1} - \hat{X}_{k^*+1} || \le ||F^*|| \, ||D_0^{-1}|| \, ||F|| \, \epsilon/n = \epsilon \ , \text{ since } ||D_0^{-1}|| = 1, \ ||F|| = ||F^*|| = \sqrt{n} \ .$$

Thus, in $k^* + 1$ steps (2.2.5), (2.2.6) , at the arithmetic 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 bound ϵ.

In fact, the algorithm of the present paper for Cauchy-like inversion extends the algorithm of [P93] for Toeplitz-like inversion. Application of our present algorithm to Toeplitz-like inversion based on proposition 2.5.1 has some advantages over the direct solution, by means of the algorithm of [P93].. Namely, modification of Newton's iteration of (2.2.5)-(2.2.8) in the Toeplitz-like case requires an additional nonrational stage of computing the singular value decompositions of the product GH^T of the generator matrices G and H followed by the truncation by zeroing the smaller singular values (cf. [P93a]). Such a nonrational stage involves some additional computations and implies an increase of the approximation error norm bound by factors of order $3n$ at each recursive step, where 3 denotes the length of the displacement generator of the input matrix. To compensate for such an error norm increase, one needs to perform some extra Newton's steps, and this may substantially slow down or even ruin the convergence unless the required bound on the initial approximation error is strengthened, respectively. Our new Cauchy-like modification is free of such additional restrictions.

2.6 Reduction from Cauchy-like to Toeplitz-like Inversion

By following [P90], one may reduce inversion of any Cauchy-like matrix C of (2.2.2) to the inversion of Vandermonde and Toeplitz-like matrices. Let us recall such a reduction (cf.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[P90], [BP94], [GO]). We have that

$$C = V(t^{-1})TV^T(s)\ , \tag{2.6.1}$$

where

$$V(q) = \begin{pmatrix} 1 & q_0 & \cdots & q_0^{n-1} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & q_0 & \cdots & q_0^{n-1} \end{pmatrix}$$

(for $q = (q_i)_{i=0}^{n-1}$) is a Vandermonde matrix, T is a Toeplitz-like matrix, such that

$$\Delta_{\{Z_\varphi, Z_{1/\varphi}\}}(T) = T - Z_\varphi T Z_{1/\varphi}^T = \overline{GH}^T = [\overline{g}_1, \cdots, \overline{g}_{\alpha+2}][\overline{h}_1, \cdots, \overline{h}_{\alpha+2}]^T\ ,$$

[cf. (2.1.1)]:

$$\overline{g}_m = V^{-1}(\frac{1}{t})g_m, \quad \overline{h}_m = V^{-1}(s)h_m, \quad m = 1, 2, \cdots, \alpha;$$

$$[g_1, \cdots, g_\alpha] = D_{\frac{1}{t}}[z_1, \cdots, z_n]^T;$$

$$[h_1, \cdots, h_\alpha] = [y_1, \cdots, y_n]^T,$$

$$\overline{g}_{\alpha+1} = -v - \varphi e_0, \quad \overline{h}_{\alpha+1} = \Phi_\gamma V^{-1}(s)C^{-1}V^{-T}(\frac{1}{t})e_{n-1}\ ,$$

$$\overline{g}_{\alpha+2} = Z_\varphi V^{-1}(\frac{1}{t})CV^{-T}(s)e_{n-1}, \quad \overline{h}_{\alpha+2} = -\gamma - \frac{1}{\varphi}e_0\ ,$$

$$P(\lambda) = \prod_{i=0}^{n-1}(\lambda - s_i) = \lambda^n + \sum_{i=0}^{n-1}\gamma_i\lambda^i, \quad \gamma = (\gamma_i)_{i=0}^{n-1}\ ,$$

$$U(\lambda) = \prod_{i=0}^{n-1}(\lambda - \frac{1}{t_i}) = \lambda^n + \sum_{i=0}^{n-1}v_i\lambda^i, \quad v = (v_i)_{i=0}^{n-1}\ ,$$

$$\Phi_\gamma = \begin{pmatrix} 0 & 0 & \cdots & 0 & -\gamma_0 \\ 1 & 0 & \cdots & 0 & -\gamma_1 \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & -\gamma_{n-1} \end{pmatrix}$$

is the Frobenius (companion) matrix of the polynomial $P(\lambda)$, with coefficients γ_i, $i = 0, 1, \cdots, n-1$.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Due to (2.6.1), we have

$$C^{-1} = V^{-T}(s)T^{-1}V^{-1}(t^{-1}) , \qquad (2.6.2)$$

which, in principle, reduces Cauchy-like inversion to Toeplitz-like inversion and Vandermonde computations, as promised. In the previous section, however, we commented on some advantages of our present modification of Newton's iteration in the Cauchy-like case over the known one in the Toeplitz-like case. Involvement of operations with Vandermonde matrices is an additional burden in the latter case; furthermore, numerical implementation of the Vandermonde inversion stage is hard since Vandermonde matrices are known to be ill-conditioned (cf. [GI] and [GO], equation (3.5)), unlike the Fourier transform matrices F of section 2.5 (cf. [BP], proposition 3.4.1).

2.7 Discussion

The proposed algorithms for Cauchy-like inversion are not complete; they should be complemented by some recipes for obtaining initial approximations X_0, which should be good enough in order to guarantee sufficiently fast convergence, as we estimated in section 2.4. In some cases, a good initial approximation is readily available, for instance, if the original matrix arises from discretization of dynamically varying input parameters and if the objective is to maintain the originally available solution. Presently, however, we have no recipes for rapid computation of the initial approximations for the general case input, so that our present paper only supplies one half of a solution, leaving the initial guess problem open. A possible direction towards filling this gap is by extending the homotopy approach of [P92], originally developed for the Toeplitz-like case. According to this approach, one first approximates the inverse of a nearby Toeplitz-like matrix whose inversion is simple. Then, one recursively uses the computed approximations as the initial approximations X_0

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

to the inverses of the nearby matrices in their homotopic transformation to the original matrix. In [P92], this approach was specified and made effective for any well-conditioned Toeplitz-like input matrix.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3 Fast Multipoint Polynomial Evaluation and Interpolation via Computations with Structured Matrices

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3.1 Introduction

Suppose that we are given n points, $x_0, x_1, \cdots, x_{n-1}$, and the coefficients of a polynomial $p(x)$ of degree $n-1$. One may evaluate $p(x_0), p(x_1), \cdots, p(x_{n-1})$ in $2(n^2-n)$ arithmetic operations, by means of Horner's algorithm, or in $O(n\log^2 n)$ arithmetic operations, by means of the more recent algorithm of Moenck and Borodin (cf. [MB], [AHU], or [BP]), which, however, leads to numerical stability problems (due to recursive application of polynomial division). For $x_0, x_1, \cdots, x_{n-1}$ lying in a fixed real line interval, the alternative algorithms of [R88] and [P95] approximate the values $p(x_0), p(x_1), \cdots, p(x_{n-1})$ at the cost $O(n\log^2 n)$, with improved numerical stability. Confinement of the input points to a real line interval is essential in the approaches of [R88] and [P95] since they rely on a certain result on approximation of functions on such an interval.

In [PSLT], it was proposed to use the properties of various classes of structured matrices (that is, of Toeplitz-like, Vandermonde-like, and Cauchy-like matrices) and the known correlations among these classes [defined via some associated linear operators of scaling and displacement (shift)] in order to solve the problem of multipoint polynomial approximation and also the converse problem of approximate polynomial interpolation. The idea of exploiting the correlation among the above matrix classes in order to improve some fundamental matrix computations was first proposed in [P90] and later on used in [GKO]; as an alternative approach to multipoint polynomial evaluation and interpolation, this idea was first applied in [PSLT]. Unlike [R88] and [P95], the approach of [PSLT] allows complex input points $x_0, x_1, \cdots, x_{n-1}$.

Our present paper refines and extends the latter approach. Like [PSLT], we define the original evaluation and interpolation problems by the vector equation $\vec{v} = V(\vec{x})\,\vec{p}$, where $\vec{p}$ and $\vec{v}$ are the vectors of the coefficients of the polynomial and of its values at the given points

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$x_0, x_1, \cdots, x_{n-1}$, respectively, and $V(\vec{x})$ is the associated Vandermonde matrix. Then, instead of rather complicated reductions of [PSLT] to computations with Toeplitz-like and Cauchy-like matrices, we multiply $V(\vec{x})$ by a discrete Fourier transform matrix and arrive at a Cauchy-like matrix, for which approximate solutions to our original problems can be readily computed, due to a simple but highly effective technique of *summation reordering* (cf. [R85] or [BP], pp. 260-261).

We specify the number of arithmetic operations sufficient in order to guarantee the desired bound ϵ on the output approximation errors (this number is shown to be $O(n \log n)$ for a large class of input sets of points and for $\log(1/\epsilon) = O(\log n)$) and then show 2 extensions of the approach, in particular, to multipoint approximation of a polynomial given by its Chebyshev decomposition (section 3.7) and to approximation of the vector $\vec{v}$ for a given vector $\vec{p}$, where $\vec{v} = V\ \vec{p}$, for a Vandermonde-like matrix V (section 3.6).

We consider our present paper as a new improvement of the previous work , and we point out some directions for further modifications and potential improvements of our algorithms (see remarks 3.3.1, 3.3.2, 3.3.3, and 3.5.3). The major remaining challenge is the problem of extending our results to a larger class of inputs, that is, the problem of relaxing the restrictions on the class of input values for which our algorithms work effectively. (We specify these restrictions in the statements of our propositions 3.3.1, 3.5.1, 3.6.1, and 3.7.1.) Our numerical experiments, however, show that already in the present form our fast algorithms have substantial advantage in their reliability over the Moenck-Borodin algorithm of [MB], and unlike the algorithms of [R88] and [P95], they work without restricting the input nodes to a real line interval.

Apart from the cited material of sections 3.6 and 3.7, our paper is organized as follows. In sections 3.2 and 3.4, we present some definitions and auxiliary results. In sections 3.3

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

and 3.5, we treat multipoint evaluation and interpolation, respectively. In section 3.8, we show some results of our numerical tests.

3.2 Preliminaries

Definition 3.2.1 *Hereafter, $\vec{z}$ will denote the n-dimensional vector* $[\ z_0, \cdots, z_{n-1}\ ]^T \in C^{n\times 1}$, *$z_i$ will denote its i-th coordinate, for* $i = 0, 1, \cdots, n-1$, *and* $1/\vec{z} = \vec{z}^{-1}$ *will denote the vector* $[\ 1/z_0, \cdots, 1/z_{n-1}\ ]^T$, *provided that* $\prod_{i=0}^{n-1} z_i \neq 0$.

This definition also apply to the cases where p, r, s, t, u, v, w, x, and y replace z.

Given the coefficient vector $\vec{p} = [\ p_0, \cdots, p_{n-1}\ ]^T$ of a polynomial

$$p(x) = \sum_{i=0}^{n-1} p_i x^i,$$

the vector $\vec{v}$ of the values of $p(x)$ on the set of points $\{\ x_0, \cdots, x_{n-1}\ \}$ is defined by the equation,

$$\vec{v} = V(\vec{x})\ \vec{p}, \tag{3.2.1}$$

where definition 3.2.1 is applied, and

$$V = V(\vec{x}) = \begin{pmatrix} 1 & x_0 & \cdots & x_0^{n-1} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n-1} & \cdots & x_{n-1}^{n-1} \end{pmatrix}, \quad x_i \neq 0, \quad x_i \neq x_j,\ i \neq j,\ i, j = 0, \cdots, n-1\ .$$

$V(\vec{x})$ is called a *Vandermonde matrix.*

The vector equation (3.2.1) defines the problems of *multipoint evaluation and interpolation* for a polynomial $p(x)$, that is, the problem of computing the vector $\vec{v}$ of the values of $p(x)$, where the vectors $\vec{p}$ of the coefficients and $\vec{x}$ of the points (nodes of evaluation) are

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

given, and the problem of computing the coefficient vector $\vec{p}$, where the vectors $\vec{v}$ of the values and $\vec{x}$ of the points (nodes of interpolation) are given, respectively. We will approach these two problems by using scaling and displacement (shift) operators for representation of the Vandermonde matrix $V(\vec{x})$ of (3.2.1). By following [GKK], [P90], [GO], and [BP], we define such an operator for a Vandermonde matrix $V = V(\vec{x})$ as follows:

$$\nabla_{(D_{1/\vec{x}},Z_1^T)}(V) = D_{1/\vec{x}}V - VZ_1^T = \vec{g}\,\vec{e}^{\,T} . \tag{3.2.2}$$

Here and hereafter, we write

$$\left.\begin{array}{rcl} \vec{e}^{\,T} & = & [\,1,0,\cdots,0\,]\,, \\ D_{1/\vec{x}} & = & diag(\,\frac{1}{x_0},\cdots,\frac{1}{x_{n-1}}\,)\,, \\ \vec{g} & = & \left[\;\frac{1}{x_0}-x_0^{n-1},\cdots,\frac{1}{x_{n-1}}-x_{n-1}^{n-1}\;\right]^T, \end{array}\right\} \tag{3.2.3}$$

and

$$Z_1 = \begin{pmatrix} 0 & \cdots & 0 & 1 \\ 1 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & 0 \end{pmatrix} .$$

Definition3.2.2 [GO] *The matrix*

$$C(\vec{s},\vec{t}\,) = \left[\;\frac{1}{s_i - t_j}\;\right]_{i,j=0}^{n-1}, \qquad where\ \ s_i \neq t_j,\ \ for\ \ all\ \ pairs\ \ i,\ j, \tag{3.2.4}$$

is called a Cauchy matrix. An $n \times n$ matrix C is called a Cauchy-like matrix if

$$\nabla_{(D_{\vec{s}},D_{\vec{t}})}(C) = G\,H^T\,, \quad G \in C^{n\times\alpha}\,,\ H \in C^{n\times\alpha}\,. \tag{3.2.5}$$

Then, the pair of matrices $\{G, H^T\}$ is called a $(D_{\vec{s}}, D_{\vec{t}})$-generator (or a scaling generator) of C of length α, and the minimum α allowing the above representation (3.2.5) *is called the $(D_{\vec{s}}, D_{\vec{t}})$-rank (or the scaling rank) of C.*

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Cauchy matrix of (3.2.4) is a special case of Cauchy-like matrices having $(D_{\vec{s}}, D_{\vec{t}})$-rank 1.

Hereafter, M^H denotes the Hermitian transpose of a matrix M, and we write

$$w_k = exp(2\pi k\sqrt{-1}/n), \quad \vec{w} = [\ w_0, \cdots, w_{n-1}\ ]^T, \qquad k = 0, 1, \cdots, n-1\ . \tag{3.2.6}$$

It is known from proposition 5.1 of [GKO] that VF^H is a Cauchy-like matrix such that

$$\nabla_{(D_{1/\vec{x}}, D_1^H)}(VF^H) = \vec{g}\,\vec{h}^{\,T}\ ,$$

where $D_{1/\vec{x}}$ is defined by (3.2.3),

$$\left.\begin{aligned} \vec{h}^{\,T} &= \vec{e}^{\,T}F^H = \tfrac{1}{\sqrt{n}}[1, 1, \cdots, 1], \\ D_1 &= diag(w_k)_{k=0}^{n-1} = diag(exp(2\pi k\sqrt{-1}/n))_{k=0}^{n-1}, \\ D_1^H &= D^{-1} = diag(exp(-2\pi k\sqrt{-1}/n))_{k=0}^{n-1}, \end{aligned}\right\} \tag{3.2.7}$$

$F = \frac{1}{\sqrt{n}}(w_{j,k})_{j,k=0}^{n-1}$ denotes the (normalized) matrix of the discrete Fourier transform, $w_{j,k} = w_{jk}$, $F^H = F^{-1}$. Therefore, a well-known formula expresses the matrix VF^H via its generator (cf. e.g. theorem 3.1 of [GO] or fact 2.11.2 on page 179 of [BP]), which gives us the following matrix equation (cf. definitions 3.2.1 and 3.2.2):

$$V\ F^H = \frac{1}{\sqrt{n}}\ diag(\frac{1}{x_i} - x_i^{n-1})_{i=0}^{n-1}\ C(\vec{x}^{\,-1}, \vec{w}^{\,-1}\ ) \tag{3.2.8}$$

or, equivalently,

$$V = -\frac{1}{\sqrt{n}}\ diag(1 - x_i^n)_{i=0}^{n-1}\ C(\vec{x}, \vec{w}\ )\ diag(w_i)_{i=0}^{n-1}\ F\ . \tag{3.2.9}$$

Definition 3.2.3 *For a vector* $\vec{x} = [\ x_0, \cdots, x_{n-1}\ ]^T$, *let* $\Gamma_{\vec{x}}(u)$ *denote the polynomial*

$$\prod_{i=0}^{n-1}(u - x_i) = u^n + \sum_{i=0}^{n-1} r_i u^i$$

and let $\vec{r} = \vec{r}(\vec{x})$ *denote the vector* $[\ r_0, \cdots, r_{n-1}\ ]^T$.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

An alternative derivation of (3.2.8) and (3.2.9) may rely on the following matrix equation (see equation (3.1) of [GO] or equation (2.4.2) on page 131 of [BP]):

$$C(\vec{y},\vec{z}) = diag(1/\Gamma_{\vec{z}}(y_i))_{i=0}^{n-1}\ V(\vec{y})\ V^{-1}(\vec{z})\ diag(\Gamma'_{\vec{z}}(z_i))_{i=0}^{n-1}, \tag{3.2.10}$$

where $\Gamma_{\vec{z}}(y)$ is defined by definition 3.2.3. Substitute $\vec{y} = \vec{x}$, $\vec{z} = \vec{w}$,

$$V(\vec{y}) = V(\vec{x}) = V,\ \ V(\vec{z}) = V(\vec{w}) = \sqrt{n}F,\ \ \Gamma_{\vec{w}}(u) = \prod_{k=0}^{n-1}(u - w_k) = u^n - 1,\ \ \Gamma'_{\vec{w}}(u) = nu^{n-1}, \tag{3.2.11}$$

so that

$$\Gamma_{\vec{z}}(y_i) = x_i^n - 1, \quad \Gamma'_{\vec{z}}(z_i) = nw_i^{n-1} = n/w_i, \quad i = 0, 1, \cdots, n-1, \tag{3.2.12}$$

and obtain from (3.2.10) that

$$C(\vec{x},\vec{w}) = diag(\frac{1}{x_i^n - 1})_{i=0}^{n-1}\ V(\sqrt{n}F)^{-1}\ diag(n/w_i)_{i=0}^{n-1},$$

or equivalently,

$$V = \frac{1}{\sqrt{n}}\ diag(x_i^n - 1)_{i=0}^{n-1}\ C(\vec{x},\vec{w})\ diag(w_i)_{i=0}^{n-1}\ F.$$

which gives us (3.2.9).

Let us also show how (3.2.9) and, consequently, (3.2.8) can be alternatively deduced by applying the Lagrange interpolation formula.

$$p(u) = \Gamma_{\vec{w}}(u) \sum_{k=0}^{n-1} \frac{p(w_k)}{\Gamma'_{\vec{w}}(w_k)\ (u - w_k)}, \tag{3.2.13}$$

where $p(u) = \sum_{i=0}^{n-1} p_i u^i$, w_k are defined by (3.2.6), $\Gamma_{\vec{w}}(u)$ is defined by definition 3.2.3.

We express the vectors $p(\vec{w}) = (p(w_k))_{k=0}^{n-1}$ as $\sqrt{n}F\vec{p}$, $\vec{p} = (\ p_0, \cdots, p_{n-1}\ )^T$, and $p(\vec{x}) = (p(x_k))_{k=0}^{n-1}$ as $V\vec{p}$ and obtain from (3.2.11) – (3.2.13) that

$$\begin{aligned} V\ \vec{p} &= diag(\Gamma_{\vec{w}}(x_i))_{i=0}^{n-1}\ (\frac{1}{x_i - w_k})_{i,k=0}^{n-1}\ diag(1/\Gamma'_{\vec{w}}(w_k))_{k=0}^{n-1}\ \sqrt{n}\ F\ \vec{p} \\ &= \frac{1}{\sqrt{n}} diag(x_i^n - 1)_{i=0}^{n-1}\ (\frac{1}{x_i - w_k})_{i,k=0}^{n-1}\ diag(w_k)_{k=0}^{n-1}\ F\ \vec{p}\ , \end{aligned}$$

for any vector $\vec{p}$. By removing $\vec{p}$ on both sides, we arrive at (3.2.9).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3.3 Multipoint Polynomial Evaluation

We extend and strengthen the results of [PSLT] on fast approximate multipoint polynomial evaluation. The algorithm, its correctness proof, and the complexity analysis are presented in the statement and the proof of the following proposition:

Proposition 3.3.1 *Given the coefficients of a polynomial*

$$p(x) = \sum_{i=0}^{n-1} p_i x^i$$

and the set of points $\{x_0, \cdots, x_{n-1}\}$, *the vector* $\vec{v}$ *of the equation* (3.2.1) *can be approximated within a maximum error norm bound* ϵ (*that is, we may compute a vector* $\vec{v}^{\,*}$ *such that*

$$|| \vec{v}^{\,*} - \vec{v} || = \max_i | v_i^* - v_i | \leq \epsilon \quad)$$

by using $(4n-1)L + O(n \log n)$ *arithmetic operations, where*

$$L = \lceil \frac{\log(\alpha n b / ((1-q)\epsilon))}{\log(1/q)} \rceil, \quad q \text{ is a fixed constant}, \quad 1 > q \geq x_+ = \max_k |x_k|,$$

$$\alpha = \max_k |u_k|, \quad u_k = (\frac{1}{\sqrt{n}} F \vec{p} \,)_k, \quad \text{and} \quad b = \max_k |x_k^n - 1| \leq 1 + q^n \,,$$

so that $L = O(\log n)$ *if* $\log(\alpha/\epsilon) = O(\log n)$.

Proof: Due to (3.2.1) and (3.2.8), we have

$$\vec{v} = V(\vec{x})\, \vec{p} = diag(\frac{1}{x_i} - x_i^{n-1})\, C(\vec{s}, \vec{t}\,)\, \vec{u}\,, \tag{3.3.1}$$

where $\vec{u} = (1/\sqrt{n})\, F\, \vec{p}$, $\vec{s} = \vec{x}^{\,-1}$, $\vec{t} = \vec{w}^{\,-1}$

(cf. definition 3.2.1). By substituting (3.2.4), (3.2.9), (3.3.1) and the equation

$$\frac{1}{s_i - t_k} = x_i \sum_{j=0}^{\infty} (\frac{x_i}{w_k})^j, \quad \text{for all } i \text{ and } k,$$

we obtain that

$$v_i = v(x_i) = (\frac{1}{x_i} - x_i^{n-1}) \sum_{k=0}^{n-1} \frac{u_k}{s_i - t_k} = (1 - x_i^n) \sum_{j=0}^{\infty} A_j x_i^j \,, \qquad A_j = \sum_{k=0}^{n-1} \frac{u_k}{w_k^j} \,.$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Due to the summation reordering, we now fix a sufficiently large natural L, approximate v_i by

$$v_i^* = v_i^{(L)} = (1 - x_i^n) \sum_{j=0}^{L-1} A_j x_i^j ,$$

note that $|w_k| = 1$ for all k, so that $|A_j| \leq \alpha n$, for all j, and obtain that

$$E_L = || \vec{v}^{\,*} - \vec{v} || = \max_i | v_i^* - v_i | \leq \frac{\alpha n b}{1-q} q^L . \tag{3.3.2}$$

Therefore, for $L \geq \lceil \frac{\log(\alpha n b/((1-q)\epsilon))}{\log(1/q)} \rceil$, we have $E_L \leq \epsilon$.

Evaluation of the vectors $F\vec{p}$ and $\vec{u} = [\, u_0, \cdots, u_{n-1} \,]^T$ and of the scalars $1 - x_i^n$, $i = 0, 1, \cdots, n-1$, requires $O(n \log n)$ arithmetic operations; evaluation of A_j, for all j, $j = 0, 1, \cdots, L-1$, requires $L(2n-2)$ arithmetic operations, and the subsequent evaluation of

$$v_i^* = (1 - x_i^n) \sum_{j=0}^{L-1} A_j x_i^j, \qquad \text{for all } i,\ 0 \leq i \leq n-1 ,$$

requires $(1 + 2L)n$ arithmetic operations. Therefore, we obtain an approximation to the vector $V(\vec{x})\vec{p}$ by using $L(4n-1) + O(n \log n) + n$ arithmetic operations. This is $O(n \log n)$, for $L = O(\log n)$, and if we have

$$x_+ = \max_k |x_k| \leq q < 1 \ (\text{ for a fixed constant } q \ )$$

and $\log(\alpha/\epsilon) = O(\log n)$, then $L = O(\log n)$. □

Remark 3.3.1 *Proposition* 3.3.1 *exploits the simple but highly effective algorithm [R85] for the approximation of a Cauchy matrix by a vector. This algorithm has several other important applications, notably, to particle simulation [AGR], [CGR], and [GR]. In the case where the input points* $x_0, x_1, \cdots, x_{n-1}$ *are real, one may exploit some techniques based on Chebyshev expansion (cf. [DGR]), which in the real case converges faster than Taylor's expansion that we used in order to obtain* (3.3.2).

Remark 3.3.2 *The requirement* $q < 1$ *restricts the class of input values to which proposition* 3.3.1 *applies. Moreover, even if* $x_+ \leq q < 1$ *but if* q *and* x_+ *are close to* 1, *then the value*

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

L of proposition 3.1 *may substantially exceed the level of* $\log n$. *There are some recipes to counter this problem. In particular,* given any vector $\vec{x}$, we may choose two scalars $\gamma \neq 0$ and δ such that

$$|y_i| \leq q < 1 \tag{3.3.3}$$

for $y_i = (x_i - \delta)/\gamma$, $x_i = \gamma y_i + \delta$. Then, by scaling and shifting the variable x, we may turn it into a new variable $y = (x - \delta)/\gamma$ and thus transform the polynomial

$$p(x) = \sum_{i=0}^{n-1} p_i x^i$$

into the polynomial

$$p_{\gamma,\delta}(y) = \sum_{i=0}^{n-1} p_{\gamma,\delta,i}\, y^i = \sum_{i=0}^{n-1} p_i(\gamma y + \delta)^i = \sum_{i=0}^{n-1} \gamma^i y^i \sum_{h=i}^{n-1} p_h \delta^{h-i} \binom{h}{i},$$

whose coefficients can be computed by using $O(n \log n)$ operations, if we are given γ, δ, and the vector $\vec{p}$ (compare [BP]). We have $p(x_i) = p_{\gamma,\delta}(y_i)$. Due to (3.3.3), we may apply proposition 3.3.1 in order to approximate $p_{\gamma,\delta}(y_i)$, $i = 0, 1, \cdots, n-1$, and then extend this result to approximate evaluation of $p(x_0), p(x_1), \cdots, p(x_{n-1})$; the estimates of proposition 3.3.1 for the approximation errors should change since they will now depend on the coefficients

$$p_{\gamma,\delta,i} = \gamma^i \sum_{h=i}^{n-1} p_h \delta^{h-i} \binom{h}{i}$$

of the polynomial $p_{\gamma,\delta}(x)$.

Remark 3.3.3 *The comments of remark* 3.3.2 *also apply to our other results on polynomial multipoint evaluation and interpolation (see propositions* 3.5.1, 3.6.1, *and* 3.7.1 *). In the implementation, one may add some other special techniques to improve the performance and to extend the algorithm to a larger class of input data. In particular, in the case of multipoint evaluation, one may use the following simple trick: partition the input set*

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$X = \{ x_0, \cdots, x_{n-1} \}$ into three subsets X_-, X_0, and X_+, where $|x_i| < 1$, $|x_i| = 1$, and $|x_i| > 1$ for the points x_i of X_-, X_0, and X_+, respectively, and then evaluate $p(x)$ separately on each subset. On the subset X_-, we have $q < 1$, which enables us to apply proposition 3.3.1. On the subset X_+, we have $1/|x_i| < 1$, and we may apply proposition 3.3.1 *to the reverse polynomial $q(x) = x^n\, p(1/x)$, whereas on the subset X_0 (excluding the trivial points $x = 1$ and $x = -1$), we have $-1 < y = Re\ x < 1$, $x = y + \sqrt{-1}\ Im\ x$, $(Im\ x)^2 = 1 - y^2$, where $Re\ x$ and $Im\ x$ denote the real and imaginary parts of x, respectively. Therefore, we may rewrite $p(x)$ as $p_0(y) + (Im\ x)\ p_1(y)$ and reduce the original problem of the evaluation of $p(x)$ on X_0 to the evaluation of $p_0(y)$ and $p_1(y)$ for $-1 < y < 1$. By substituting the variable $z = \theta x$, for some constant θ, $|\theta| = 1$ and/or by further partitioning the set X_0 into two subsets, we may ensure , say, that $-1/\sqrt{2} < y < 1/\sqrt{2}$.*

3.4 Some Further Definitions and Auxiliary Results

In this section, we recall some definitions and auxiliary results from [BP], [GO], and [H].

Definition 3.4.1 *For a vector $\vec{y} = [\ y_0, y_1, \cdots, y_{n-1}\ ]^T$ and a scalar $\phi \neq 0$, an $n \times n$ matrix $Z_{\phi,n}(\vec{y}) = [z_{i,j}]$ is called a ϕ-circulant matrix if*

$$z_{i,j} = y_{(i-j) \bmod n} \quad for \quad i \geq j\ , \qquad z_{i,j} = \phi y_{(i-j) \bmod n} \quad for \quad i < j\ .$$

We will use the following well-known equation (compare [BP], page 144):

$$V^T(\vec{y}\ ) = J\ Z^{-1}_{f,n}(\vec{r} + f\vec{e}\ )\ V^{-1}(\vec{y}\ )\ diag(\ a_i\ )_{i=0}^{n-1}$$

or, equivalently,

$$V^{-1}(\vec{y}\ ) = Z_{f,n}(\vec{r} + f\vec{e}\ )\ J\ V^T(\vec{y}\ )\ diag(\ a_i^{-1}\ )_{i=0}^{n-1}\ , \tag{3.4.1}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

provided that $f \neq 0, \quad f \neq x_i^n, \quad i = 0, 1, \cdots, n-1,$

$$a_i = \Gamma'_{\vec{x}}(x_i)(\ f - x_i^n\ )\ , \qquad i = 0, 1, \cdots, n-1, \tag{3.4.2}$$

$\Gamma(u) = \Gamma_{\vec{x}}(u)$ and $\vec{r} = \vec{r}(\vec{x})$ are defined in definition 3.2.3,

$Z_{f,n}(\vec{r}+f\vec{e}\,)$ denotes the f-circulant $n \times n$ matrix with the first column $\vec{r}+f\vec{e}$, (3.4.3

and

$$J = \begin{pmatrix} 0 & 0 & \cdots & 1 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 1 & \cdots & 0 \\ 1 & 0 & \cdots & 0 \end{pmatrix}. \tag{3.4.4}$$

3.5 Polynomial Interpolation Algorithm

We will next exploit the representation of the polynomial interpolation problem based on the equations (3.2.1) and (3.4.1), according to which the interpolation problem is equivalent to computing the following vector:

$$\vec{p} = V^{-1}(\vec{x})\ \vec{v} = Z_{f,n}(\vec{r} + f\vec{e}\,)\ J\ V^T\ diag(a_i^{-1})_{i=0}^{n-1}\ \vec{v}\ . \tag{3.5.1}$$

Here a_i, $Z_{f,n}(\vec{r} + f\vec{e}\,)$, and J are defined by (3.4.2) – (3.4.4); V is a Vandermonde matrix satisfying (3.2.2), and $\vec{g}$, $\vec{e}$, and $D_{1/\vec{x}}$ are defined by (3.2.3). [Compare remark 3.5.1 at the end of this section on some alternative choices of the two basic equations, instead of (3.4.1) and (3.5.1).]

Since $F^T = F$, we deduce from (3.2.9) that

$$V^T = \frac{1}{\sqrt{n}}\ F\ diag(w_i)_{i=0}^{n-1}\ C(\vec{w}, \vec{x}\,)\ diag(1 - x_i^n)_{i=0}^{n-1}. \tag{3.5.2}$$

By combining the latter equation with (3.5.1), we obtain the following result:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Corollary 3.5.1

$$\vec{p} = -\frac{1}{\sqrt{n}}\, Z_{f,n}(\vec{r} + f\vec{e}\,)\; J\; F\; C(\vec{t},\vec{s}\,)\; diag(\; \frac{1 - x_i^n}{a_i x_i}\;)\; \vec{v}\;.$$

where $\vec{s}$ *and* $\vec{t}$ *are defined by* (3.3.1), $Z_{f,n}(\vec{r} + f\vec{e}\,)$ *is defined by* (3.4.3) *and* J *is defined by* (3.4.4).

Proposition 3.5.1 *Given* 3 *sets of values:*

i) $\{\; x_i \;:\; i = 0, 1, \cdots, n-1;\; x_i \neq x_j \;\; for \;\; i \neq j \;\}$,

ii) $\{\; v_i \;:\; i = 0, 1, \cdots, n-1 \;\}$, *and*

iii) $\{\; r_i \;:\; i = 0, 1, \cdots, n-1 \;\}$

(*where the values* $\{x_i\}$ *and* $\{r_i\}$ *are related according to definition* 3.4.1), *one may approximate within* ϵ *the coefficient vector* $\vec{p} = [\; p_0, \cdots, p_{n-1} \;]$ *of the polynomial*

$$p(x) = \sum_{i=0}^{n-1} p_i x^i$$

(*that is, one may compute a vector* $\vec{p}^{\,*}$ *such that* $\| \vec{p}^{\,*} - \vec{p} \| \leq \epsilon$) *by using* $O(n \log n)$ *arithmetic operations if*

$$\max_i |x_i| \leq q < 1, \quad q \text{ is a fixed constant}, \qquad \log(\; r\alpha/\epsilon \;) = O(\log n)\;,$$

$$\alpha = \max_i |y_i|\;, \qquad y_i = \frac{1 - x_i^n}{a_i x_i}\, v_i\;, \qquad \text{and} \qquad r = \max_{i>0} |r_i|\;,$$

for $r_1, \cdots, r_{n-1}$ *of definition* 3.4.1 .

Proof: Let us write

$$\vec{y} = diag(\; \frac{1 - x_i^n}{a_i x_i}\;)\; \vec{v} = [\; y_0, \cdots, y_{n-1} \;]^T, \quad \text{so that} \quad y_i = \frac{1 - x_i^n}{a_i x_i} v_i\;.$$

Substitute these equations into corollary 3.5.1, obtain that

$$\vec{p} = -\frac{1}{\sqrt{n}}\, Z_{f,n}(\; \vec{r} + f\vec{e}\,)\; J\; F\; C(\vec{t},\vec{s}\,)\; \vec{y}\;, \tag{3.5.3}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

for $\vec{s}$ and $\vec{t}$ of (3.3.1), and consider the following vector equation:

$$\vec{z} = \begin{pmatrix} z_0 \\ \vdots \\ z_{n-1} \end{pmatrix} = C(\vec{t},\vec{s}) \begin{pmatrix} y_0 \\ \vdots \\ y_{n-1} \end{pmatrix} = \begin{pmatrix} \frac{y_0}{t_0-s_0} + \cdots + \frac{y_{n-1}}{t_0-s_{n-1}} \\ \vdots \\ \frac{y_0}{t_{n-1}-s_0} + \cdots + \frac{y_{n-1}}{t_{n-1}-s_{n-1}} \end{pmatrix} . \qquad (3.5.4)$$

The i-th coordinate of the vector $\vec{z}$ can be re-written as follows:

$$z_i = \sum_{k=0}^{n-1} \frac{y_k}{t_i - s_k} = \sum_{k=0}^{n-1} \frac{y_k}{t_i - \frac{1}{x_k}} = -\sum_{k=0}^{n-1} \frac{y_k x_k}{1 - t_i x_k} = -\sum_{k=0}^{n-1} y_k x_k \sum_{j=0}^{\infty} (t_i x_k)^j \ .$$

We write $x_+ = \max_k |x_k|$. If $x_+ < 1$, we have $|t_i x_k| \le x_+ < 1$ for all pairs (i,k), $0 \le i,\ k \le n-1$, since $t_h = exp(-2\pi h\sqrt{-1}/n)$, $h = 0, 1, \cdots, n-1$. Then, the series $\sum_{j=0}^{\infty}(t_i x_k)^j$ converges for all k, and we have

$$| \sum_{j=0}^{\infty} (t_i x_k)^j | \le \sum_{j=0}^{\infty} x_+^j = \frac{1}{1-x_+} \ .$$

Now, we write

$$B_j = \sum_{k=0}^{n-1} y_k x_k^{j+1}, \qquad z_i = -\sum_{j=0}^{\infty}\sum_{k=0}^{n-1} y_k x_k^{j+1} t_i^j = -\sum_{j=0}^{\infty} B_j t_i^j \ .$$

By means of limiting the number of the summation terms $B_j t_i^j$ to a certain number L, we obtain an approximation of the vector $\vec{z}$ by the vector

$$\vec{z}^{\,*} = \vec{z}^{\,(L)} = [\ z_0^*, z_1^*, \cdots, z_{n-1}^*\ ]^T$$

such that

$$z_i^* = -\sum_{j=0}^{L-1} B_j t_i^j \ .$$

For the error norm,

$$\epsilon_L = \| \ z^* - z \ \| = \max_i | \ z_i^* - z_i \ | \ ,$$

we obtain the bound

$$\epsilon_L \le \frac{n\alpha q^{L+1}}{1-q} \ ,$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

similarly to (3.3.2). From (3.5.3) and (3.5.4), we obtain that

$$\vec{p} = -\frac{1}{\sqrt{n}} Z_{f,n}(\vec{r} + f\vec{e})\, J\, F\, \vec{z}\,.$$

We write

$$\vec{p}^{\,*} = -\frac{1}{\sqrt{n}} Z_{f,n}(\vec{r} + f\vec{e})\, JF\, \vec{z}^{\,*}\,, \qquad E_L^* = \| \vec{p}^{\,*} - \vec{p} \| = \max_i | p_i^* - p_i |$$

and obtain that $E_L^* \le \frac{1}{\sqrt{n}} \| Z_{f,n}(\vec{r} + f\vec{e}) \|_\infty \|J\|_\infty \|F\|_\infty \epsilon_L$,

where $\|A\|_\infty$ denotes the row norm of a matrix $A = [a_{ij}]$,

$$\|A\|_\infty = \max_i \sum_j |a_{ij}| \qquad (\text{compare [GL], p.57}).$$

We have

$$\|J\|_\infty = 1, \qquad \|F\|_\infty = \sqrt{n}, \qquad \| Z_f(\vec{r} + f\vec{e}) \|_\infty \le n \max_{i>0} |r_i|\,,$$

if we choose f so as to decrease or to cancel the first component of the vector $\vec{r} + f\vec{e}$. Therefore,

$$E_L^* \le \max_{i>0} |r_i|\; n\; \epsilon_L \le \frac{r n^2 \alpha q^{L+1}}{1-q}\,, \qquad \text{where} \quad r = \| r(\vec{x}) \|_\infty = \max_{i>0} |r_i|\,.$$

By assumption, $q < 1$. Therefore, we have $E_L^* < \epsilon$ for a fixed positive ϵ, if

$$L \ge \lceil \frac{\log(\, r\, \alpha\, n^2/(\,(1-q)\epsilon\,)\,)}{\log(1/q)} \rceil\,.$$

Thus, under the assumptions of proposition 3.5.2, we may choose $L = O(\log n)$.

We will now complete the proof of proposition 3.5.2 by showing that $O(n \log n + nL)$ arithmetic operations suffice for the evaluation of the vector $\vec{p}^{\,*}$. The transition from $\vec{z}^{\,*}$ to $\vec{p}^{\,*}$ is reduced to performing discrete Fourier transforms (by means of $O(n \log n)$ operations) [BP], and we will only need to estimate the arithmetic cost of computing the vector $\vec{z}^{\,*}$. We observe that $2n-1$ arithmetic operations suffice for computing each B_j. If all the B_j are

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

available, then for any fixed integer i, $2(L-1)$ arithmetic operations suffice for computing z_i^*, so that all z_i^*, $i = 0, 1, \cdots, n-1$, can be computed by using

$$2(L-1)n + L(2n-1) = 4Ln - 2n - L$$

arithmetic operations. □

Remark 3.5.1 *The readers may consider some variations of the presented construction, where the basic equation (3.4.1),* (3.5.1) *and/or (3.5.1) are replaced by some alternative equations. Here are 3 examples of such alternatives:*

a). One may rely on the equation

$$V\ J\ L(Jr)\ V^T = diag[\Gamma'(x_i)],$$

from [FHR] (compare page 144 of [BP]), rather than on (3.4.1).

b). Alternatively, one may rely on the equation (3.2.10),

$$C(\vec{y}, \vec{z}\ ) = diag(\ 1/\Gamma_{\vec{z}}(y_i))_{i=0}^{n-1}\ V(\vec{y})V^{-1}(\vec{z})\ diag(\Gamma'_{\vec{z}}(z_k)\ )_{k=0}^{n-1}.$$

In particular, one may combine (3.2.10) for $\vec{y} = \vec{x}^{\ -1}$, $\vec{z} = \vec{w}$, as a substitution for (3.5.2), with the next matrix equation, due to [FHR], Corollary 3.3.1:

$$C^{-1}(\vec{x}, \vec{y}\ ) = -diag(\ \frac{\Gamma_{\vec{x}}(y_i)}{\Gamma'_{\vec{y}}(y_i)}\ )_{i=0}^{n-1}\ C^T(\vec{x}, \vec{y}\ )\ diag(\ \frac{\Gamma_{\vec{y}}(x_i)}{\Gamma'_{\vec{x}}(x_i)}\ )_{i=0}^{n-1}\ , \qquad (3.5.5)$$

for any pair of vectors $\vec{x}$, $\vec{y}$ defining a nonsingular $n \times n$ matrix $C(\vec{x}, \vec{y}\ )$. On the other hand, from (3.2.9) we obtain that

$$V^{-1} = -\sqrt{n}F^H\ diag(t_i)_{i=0}^{n-1}\ C^{-1}(\vec{x}, \vec{w}\ )\ diag(\frac{1}{1-x_i^n}\ )_{i=0}^{n-1}.$$

Substitute (3.5.5) for $\vec{y} = \vec{w}$ and for $\vec{w}$ of (3.2.6) into the latter equation, apply (3.2.12), and deduce the following equation:

$$V^{-1} = -\frac{1}{\sqrt{n}}F^H\ diag(\Gamma_{\vec{x}}(w_i))_{i=0}^{n-1}\ C^T(\vec{x}, \vec{w}\ )\ diag(\frac{1}{\Gamma'_{\vec{x}}(x_i)}\ )_{i=0}^{n-1}.$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

c). We may rely on (3.2.10) but substitute distinct $\vec{y}$ and $\vec{z}$, namely, $\vec{y} = \vec{w}$, $\vec{z} = \vec{x}$.

Then we have

$$V(\vec{y}\,) = \sqrt{n}F, \qquad \Gamma_{\vec{x}}(u) = \prod_{j=0}^{n-1}(u - x_j),$$

$$\Gamma_{\vec{z}}(y_i) = \Gamma_{\vec{x}}(w_i) = \prod_{j=0}^{n-1}(w_i - x_j). \qquad \Gamma'_{\vec{z}}(z_k) = \Gamma'_{\vec{x}}(x_k) = \prod_{i=0, i\neq k}^{n-1}(x_k - x_i).$$

We deduce from (3.5.2) that

$$V^{-1} = \frac{1}{\sqrt{n}}F^H \; diag(\Gamma_{\vec{x}}(w_i))_{i=0}^{n-1} \; C^T(\vec{w}, \vec{x}\,) \; diag(\frac{1}{\Gamma'_{\vec{x}}(x_i)}\,)_{i=0}^{n-1}\,,$$

and, consequently,

$$\vec{p} = \frac{1}{\sqrt{n}}F^H \; diag(\Gamma_{\vec{x}}(w_i))_{i=0}^{n-1} \; C^T(\vec{w}, \vec{x}\,) \; diag(\frac{1}{\Gamma'_{\vec{x}}(x_i)}\,)_{i=0}^{n-1} \; \vec{v}.$$

Finally, we observe that all the cited modifications lead us to the same asymptotic estimates for the computational complexity of computing approximate solutions, though, perhaps, they may differ substantially in their practical performance.

3.6 Multiplication of a Vector by Vandermonde-like Matrix

In this section, we will extend the results of section 3 to the class of Vandermonde-like matrices, which includes Vandermonde matrices as a special subclass.

Definition 3.6.1 [GO] *V is an $n \times n$ Vandermonde-like matrix if*

$$\nabla_{(D_{1/\vec{x}}, Z_1^T)}(V) = D_{1/\vec{x}}V - VZ_1^T = GB^T\,, \tag{3.6.1}$$

where $G = [\,\vec{g}_1, \vec{g}_2, \cdots, \vec{g}_\beta\,] \in C^{n\times\beta}$, $B \in C^{n\times\beta}$, and $D_{1/\vec{x}}$ and Z_1^T are defined as in the previous sections. The pair of matrices $\{G, B^T\}$ (nonunique) is called a $(D_{1/\vec{x}}, Z_1^T)$-generator of V of length β, and the minimum possible β allowing the representation (3.6.1) *is the $(D_{1/\vec{x}}, Z_1^T)$-rank of V.*

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

It can be easily verified that for V satisfying (3.6.1), VF^H is a Cauchy-like matrix such that

$$\nabla_{(D_{1/x},D_1^H)}(VF^H) = GH^H,$$

where $H^H = B^TF^H = [\ \vec{h}_1,\cdots,\vec{h}_\beta\ ]^H$ and, as before, the superscript 'H' stands for the Hermitian transposes. By following [GKO], [GO], and [H], we obtain that

$$VF^H = \sum_{m=1}^{\beta} diag(\vec{g}_m)\ C(\vec{s},\vec{t}\,)\ diag(\vec{h}_m)\ ,$$

where we use definition 3.2.1 and equations (3.2.3), (3.2.7) and (3.3.1). Then, for any vector $\vec{p}\in C^{n\times 1}$, we have

$$V\vec{p} = \sum_{m=1}^{\beta} diag(\vec{g}_m)\ C(\vec{s},\vec{t}\,)\ diag(\vec{h}_m)\ F\ \vec{p}\ .$$

We write

$$V_m = diag(\vec{h}_m)\ F\ \vec{p} = \begin{pmatrix} v_{m,0} \\ \vdots \\ v_{m,n-1} \end{pmatrix}, \qquad \vec{g}_m = \begin{pmatrix} g_{m,0} \\ \vdots \\ g_{m,n-1} \end{pmatrix}, \qquad 1 \le m \le 3.$$

Then, we have

$$V\vec{p} = \begin{pmatrix} \sum_{m=1}^{\beta} g_{m,0} \sum_{k=0}^{n-1} \frac{v_{m,k}}{\frac{1}{x_0}-t_k} \\ \vdots \\ \sum_{m=1}^{\beta} g_{m,n-1} \sum_{k=0}^{n-1} \frac{v_{m,k}}{\frac{1}{x_{n-1}}-t_k} \end{pmatrix} = \begin{pmatrix} u(x_0) \\ \vdots \\ u(x_{n-1}) \end{pmatrix},$$

where

$$u(x_i) = \sum_{m=1}^{\beta} g_{m,i} \sum_{k=0}^{n-1} \frac{v_{m,k}}{\frac{1}{x_i}-t_k}\ .$$

Furthermore, if $|x_i| < q < 1$,

$$u(x_i) = \sum_{m=1}^{\beta} g_{m,i} \sum_{k=0}^{n-1} \frac{v_{m,k}x_i}{1-t_kx_i} = \sum_{m=1}^{\beta} g_{m,i} \sum_{k=0}^{n-1} v_{m,k}x_i \sum_{j=0}^{\infty}(t_kx_i)^j = \sum_{m=1}^{\beta} g_{m,i} \sum_{j=0}^{\infty} A_{j,m}x_i^{j+1}\ .$$

Write

$$A_{j,m} = \sum_{k=0}^{n-1} v_{m,k}\ t_k^j, \qquad m = 1,2,\cdots,\beta,$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

and define multipoint polynomial approximations as follows:

$$u^*(x_i) = \sum_{m=1}^{\beta} g_{m,i} \sum_{j=0}^{L-1} A_{j,m} x_i^{j+1} \ , \quad \text{and} \qquad E_L(x_i) = \mid u(x_i) - u^*(x_i) \mid .$$

We have

$$E_L(x_i) = \mid \sum_{m=1}^{\beta} g_{m,i} \sum_{j=L}^{\infty} A_{j,m} x_i^{j+1} \mid \leq \frac{n\alpha\beta b q^{L+1}}{1-q} \ ,$$

where

$$x_+ = \max_i |x_i| \leq q, \text{ as in sections 3.3 and 3.5}, \quad \alpha = \max_{m,k} |v_{m,k}|, \quad b = \max_{m,k} |g_{m,k}| \ .$$

Therefore, for

$$L \geq \lceil \frac{\log(\ n\alpha\beta b/(\ (1-q)\epsilon\ )\ )}{\log(1/q)} \rceil - 1 \ , \tag{3.6.2}$$

we have $E_L(x_i) \leq \epsilon$.

Next, we estimate the computational cost of approximation of the values $u(x_i)$. Computing $A_{j,m}$ for all $j = 0, 1, \cdots, L-1, \quad m = 1, 2, \cdots, \beta$, involves $L(2n-2)\beta$ arithmetic operations. Subsequent computation of $g_{mi} \sum_{j=0}^{L-1} A_{j,m} x_i^{j+1}$ involves $2L+1$ arithmetic operations. Thus, for all $1 \leq i \leq n$, $1 \leq m \leq \beta$, computing $u^*(x_i)$ uses

$$L(n-2)\beta + 2nL\beta + (\beta-1)n + O(\beta n \log n) + n\beta = (4n-2)L\beta + O(\beta n \log n)$$

arithmetic operations. Here, $O(n \log n)$ arithmetic operations are needed to compute V_m for each m, $m = 1, 2, \cdots, \beta$.

We have proved the following result:

Proposition 3.6.1 *For an $n \times n$ Vandermonde-like matrix V, given with its generator G and B, $G \in C^{n \times \beta}$, $B \in C^{n \times \beta}$, and for a given vector $\vec{p} \in C^{n \times 1}$, the vector $V\vec{p}$ can be approximated by $u^*(x_i)$, within the error bound $\| \vec{u}^{\,*} - \vec{u} \| \leq \epsilon$ (for any given $\epsilon > 0$), at the cost of performing*

$L(4n-2)\beta + O(\beta n \log n)$ arithmetic operations, for L satisfying (3.6.2) and $q < 1$. In

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

particular, if $\log(\alpha\beta b/\epsilon) = O(\log n)$, *for* q, α, *and* b *defined as in propositions* 3.3.1 *and* 3.5.2, *then, the cost bound is* $O(n\beta \log n)$.

3.7 Multiplication of a Vector by Chebyshev-Vandermonde or Chebyshev-Vandermonde-like Matrices

Definition 7.1 (compare [GO1]) *For* $n = 0, 1, 2, \cdots$, *recursively define Chebyshev polynomials of the first kind* $T_n(x)$ *and Chebyshev polynomials of the second kind* $U_n(x)$ *as follows:*

$$\begin{aligned} T_0(x) &= 1, \qquad T_1(x) = x \ , \\ T_n(x) &= T_{n-1}(x) - T_{n-2}(x) \ , \end{aligned}$$

$$\begin{aligned} U_0(x) &= 1, \qquad U_1(x) = 2x \ , \\ U_n(x) &= 2xU_{n-1}(x) - U_{n-2}(x) \ . \end{aligned}$$

Define the Chebyshev-Vandermonde matrices as follows:

$$V_T(\vec{x}) = \begin{pmatrix} T_0(x_0) & T_1(x_0) & \cdots & T_{n-1}(x_0) \\ T_0(x_1) & T_1(x_1) & \cdots & T_{n-1}(x_1) \\ \vdots & \vdots & \vdots & \vdots \\ T_0(x_{n-1}) & T_1(x_{n-1}) & \cdots & T_{n-1}(x_{n-1}) \end{pmatrix},$$

$$V_U(\vec{x}) = \begin{pmatrix} U_0(x_0) & U_1(x_0) & \cdots & U_{n-1}(x_0) \\ U_0(x_1) & U_1(x_1) & \cdots & U_{n-1}(x_1) \\ \vdots & \vdots & \vdots & \vdots \\ U_0(x_{n-1}) & U_1(x_{n-1}) & \cdots & U_{n-1}(x_{n-1}) \end{pmatrix}.$$

Fact 3.7.1 *The vectors* $V_T(\vec{x})\ \vec{p}$ *and* $V_U(\vec{x})\ \vec{p}$, *for 2 Chebyshev-Vandermonde matrices*

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$V_T(\vec{x})$ and $V_U(\vec{x})$, are the vectors of the values of the 2 polynomials,

$$P_T(x) = \sum_{k=0}^{n-1} p_k T_k(x) \qquad \text{and} \qquad P_U(x) = \sum_{k=0}^{n-1} p_k U_k(x) \ , \tag{3.7.1}$$

respectively, at the points $x_0, \cdots, x_{n-1}$.

[KO] defines linear operators associated with Chebyshev-Vandermonde matrices $V_T(\vec{x})$ and $V_U(\vec{x})$, which leads to operator representation of these matrices in the form GB^T for $G \in C^{n\times\alpha}$, $B \in C^{n\times\alpha}$, $\alpha \leq 2$. By applying discrete Fourier transform, we obtain Cauchy-like matrices $V_T(\vec{x})F^H$ and $V_U(\vec{x})F^H$ ([KO]). Then, application of the techniques of the previous section gives us fast algorithms for multipoint approximation of the polynomials $P_T(x)$ and $P_U(x)$ of (3.7.1). We will next specify this approach for the classes of polynomials $P_T(x)$, Chebyshev-Vandermonde matrices $V_T(\vec{x})$, and their extensions to Chebyshev-Vandermonde-like matrices. We will omit the similar treatment of polynomial $P_U(x)$, matrices $V_U(\vec{x})$, and their Chebyshev-Vandermonde-like extentions: furthermore, as a challenge for the readers and future researchers, we will leave elaboration of a real case modification that would rely on Chebyshev's expansion approach of the report [DGR], rather than on Taylor's expansion approach (compare our remark 3.3.1).

Definition 7.2 (compare [KO]) *Let a matrix $R \in C^{n\times n}$, satisfy the matrix equation*

$$\nabla_{(D_{1/x},W)}(R) = GB^T, \qquad for \quad G,\ B \in C^{n\times\alpha} \ , \tag{3.7.2}$$

where

$$W = 2\sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} (-1)^{i-1} \ (Z_0^T)^{2i-1}, \qquad \text{and} \qquad Z_0 = \begin{pmatrix} 0 & \cdots & 0 & 0 & 0 \\ 1 & \cdots & 0 & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & 1 & 0 & 0 \\ 0 & \cdots & 0 & 1 & 0 \end{pmatrix} .$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Then R is called a Chebyshev-Vandermonde-like matrix, the pair of matrices $\{G, B\}$ is called a $(D_{1/\vec{x}}, W)$-generator of R of length α (not uniquely defined for any R and α), and the smallest length α over all possible $(D_{1/\vec{x}}, W)$-generators is called the $(D_{1/\vec{x}}, W)$-rank of R.

At the end of this section, we will show that the matrix $R = V_T(\vec{x})$ satisfies (3.7.2) for $\alpha \leq 2$. The following lemma states how a Chebyshev-Vandermonde-like matrix can be transformed into a Cauchy-like matrix:

Lemma 3.7.1 (compare [KO]) *Let a matrix R be given by its ($2D_{\vec{x}}, Z_1 + Z_1^T$)-generator, so that*

$$\nabla_{(2D_{\vec{x}}, Z_1 + Z_1^T)}(R) = 2D_{\vec{x}}R - R(Z_1 + Z_1^T) = GB^H$$

for $G,\ B \in C^{n \times \alpha}$. Then RF^H is a Cauchy-like matrix such that

$$\nabla_{(2D_{\vec{x}}, 2D_{cos})}(RF^H) = 2(D_{\vec{x}}(RF^H) - (RF^H)D_{cos}) = GH^H,$$

where

$$G = [\ \vec{g}_1, \cdots, \vec{g}_\alpha\ ], \qquad H = [\ \vec{h}_1^H, \cdots, \vec{h}_\alpha^H\ ] = FB, \qquad D_{\vec{x}} = diag(\ x_0, \cdots, x_{n-1}\ )\ ;$$

Z_1, F^H, and F are defined as above, and

$$D_{cos} = diag(\ 1,\ cos(\tfrac{\pi}{n}),\ cos(\tfrac{\pi}{n}),\ \cdots,\ cos(\tfrac{(n-1)\pi}{n})\ )\ .$$

Next, we will approximate the vector $R\ \vec{p}$, where $\vec{p} \in C^{n \times 1}$, R is an $n \times n$ Chebyshev-Vandermonde-like matrix. We will apply the reduction to Cauchy-like matrices given by lemma 7.1. We have the following equations [GO]:

$$\begin{aligned} RF^H &= \frac{1}{2}\sum_{m=1}^{\alpha} diag(\vec{g}_m)\ C(\vec{x}, \vec{t}\ )\ diag(\vec{h}_m), \\ R\ \vec{p} &= \frac{1}{2}\sum_{m=1}^{\alpha} diag(\vec{g}_m)\ C(\vec{x}, \vec{t}\ )\ diag(\vec{h}_m)\ F\ \vec{p}\ , \end{aligned}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

where $\vec{x} = (x_i)_{i=0}^{n-1}$, $\vec{t} = (t_j)_{j=0}^{n-1}$ are defined as before (cf. definition 3.2.1) but now we write $t_j = 2cos(\frac{j-1}{n}\pi)$, instead of using (3.3.1).

Denote $\vec{v}_m = diag(\vec{h}_m)\ F\ \vec{p} = [\ v_{m,0}, \cdots, v_{m,n-1}\ ]$, $m = 1, 2, \cdots, \alpha$.

Then $O(n\alpha \log n)$ arithmetic operations suffice in order to compute $\vec{v}_m$ for all m. Since

$$C(\vec{x}, \vec{t}\ )\ \vec{v}_m = \begin{pmatrix} \sum_{k=0}^{n-1} \frac{v_{m,k}}{x_0 - t_k} \\ \vdots \\ \sum_{k=0}^{n} \frac{v_{m,k}}{x_{n-1} - t_k} \end{pmatrix} ,$$

we have

$$2R\ \vec{p} = \sum_{m=1}^{\alpha} diag(\vec{g}_m)\ C(\vec{x}, \vec{t}\ )\ \vec{v}_m = \begin{pmatrix} \sum_{m=1}^{\alpha} g_{m,0} \sum_{k=0}^{n-1} \frac{v_{m,k}}{x_0 - t_k} \\ \vdots \\ \sum_{m=1}^{\alpha} g_{m,n-1} \sum_{k=0}^{n-1} \frac{v_{m,k}}{x_{n-1} - t_k} \end{pmatrix} = \begin{pmatrix} u(x_0) \\ \vdots \\ u(x_{n-1}) \end{pmatrix} ,$$

where $\vec{g}_m = [\ g_{m,0}, \cdots, g_{m,n-1}\ ]^T$, and

$$u(x_i) = \sum_{m=1}^{\alpha} g_{m,i} \sum_{k=0}^{n-1} \frac{v_{m,k}}{x_i - t_k} .$$

If $|\frac{x_i}{t_k}| < q < 1$ for all i and k, then, we have

$$u(x_i) = -\sum_{m=1}^{\alpha} g_{m,i} \sum_{k=0}^{n-1} \frac{v_{m,k}}{t_k} \sum_{j=0}^{\infty} (\frac{x_i}{t_k})^j = -\sum_{m=1}^{\alpha} g_{mi} \sum_{j=0}^{\infty} A_{j,m} x_i^j ,$$

where

$$A_{j,m} = \sum_{k=0}^{n-1} \frac{v_{m,k}}{t_k^{j+1}} .$$

Let

$$A = \max_{m,i} |g_{m,i}|, \qquad b = \max_{m,k} |\frac{v_{m,k}}{t_k}| , \tag{3.7.3}$$

and

$$u^*(x_i) = -\sum_{m=1}^{\alpha} g_{m,i} \sum_{j=0}^{L-1} A_{j,m} x_i^j .$$

Then, we have

$$E_L(x_i) = |\ u(x_i) - u^*(x_i)\ | = |\sum_{m=1}^{\alpha} g_{m,i} \sum_{j=L}^{\infty} A_{j,m} x_i^j\ | \le \frac{\alpha A n b q^L}{(1-q)} \le \epsilon ,$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

for any

$$L \geq \lceil \frac{\log(\ \alpha Anb/(\ (1-q)\epsilon\ )\ )}{\log(1/q)} \rceil\ . \tag{3.7.4}$$

Now, we estimate the cost of computing the approximations $u^*(x_i)$ to $u(x_i)$, $i = 0, 1, \cdots, n-1$, where

$$u^*(x_i) = -\sum_{m=1}^{\alpha} g_{m,i} \sum_{j=0}^{L-1} A_{j,m} x_i^j\ .$$

Computing $A_{j,m}$ requires $(2n-1)L\alpha$ arithmetic operations for all j and m, computing

$$g_{m,i} \sum_{j=0}^{L-1} A_{j,m} x_i^j$$

takes on $n(2L-2)$ arithmetic operatons for all i, and therefore, the overall cost of computing the vector $[\ u^*(x_0), \cdots, u^*(x_{n-1})\ ]^T$, which approximates the vector $2R\vec{p}$, is

$$\alpha L(2n-2) + \alpha n(2L-1) + n(\alpha - 1) + O(n\alpha \log n) + n\alpha\ = 4\alpha nL + O(n\alpha \log n)\ .$$

This is $O(n\alpha \log n)$, for $L = O(\log n)$, and we may choose $L = O(\log n)$ satisfying (3.7.4) if $q < 1$ and if $\log(\alpha Ab/\epsilon) = O(\log n)$. We have proved the following proposition:

Proposition 3.7.1 *Given an $n \times n$ Chebyshev-Vandermonde-like matrix R and a vector $\vec{p} \in C^{n\times 1}$, let q is a fixed constant, $x_+ = \max_i |x_i| \leq q < 1$, as in propositions* 3.3.1, 3.5.2 *and* 3.6.1. *Then the vector $2R\vec{p}$ can be approximated by the vector*

$$\vec{u}^{\,*}(x) = [\ u^*(x_0), \cdots, u^*(x_{n-1})\ ]^T$$

such that $|\ u(x_i) - u^*(x_i)\ | \leq \epsilon$ for all i and for a fixed $\epsilon > 0$. The arithmetic complexity of computing such an approximation vector is bounded by $O(\alpha n \log n)$, if $\log(\alpha Anb/\epsilon) = O(\log n)$, where A and b are defined by (3.7.3)

If $R = V_T(x)$ is a Chebyshev-Vandermonde matrix, then

$$\nabla_{(2D_x, Z_1 + Z_1^T)}(R) = GB^T\ ,$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

where

$$G = \begin{pmatrix} x_0 - T_{n-2}(x_0) & T_{n-1}(x_0) - 1 \\ \vdots & \vdots \\ x_{n-1} - T_{n-2}(x_{n-1}) & T_{n-1}(x_{n-1}) - 1 \end{pmatrix}, \qquad B = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 1 \end{pmatrix}.$$

Therefore, in this case, we have $\alpha = 2$,

$$H = FB = \begin{pmatrix} 1 & 1 \\ 1 & e^{\frac{2\pi\sqrt{-1}}{n}(n-1)} \\ \vdots & \vdots \\ 1 & e^{\frac{2\pi\sqrt{-1}}{n}(n-1)^2} \end{pmatrix}, \qquad \vec{v}_m = diag(\vec{h}_m)\, F\, \vec{p} = diag(\vec{h}_m) \begin{pmatrix} b_0^* \\ \vdots \\ b_{n-1}^* \end{pmatrix}, \quad m = 1, 2,$$

$$\vec{v}_1 = [\, b_0^*, \cdots, b_{n-1}^* \,]^T, \qquad \vec{v}_2 = [\, b_0^*, \cdots, b_{n-1}^* e^{\frac{2\pi\sqrt{-1}}{n}(n-1)^2} \,]^T.$$

Thus, we have proved the following result:

Corollary 3.7.1 Given an $n \times n$ Chebyshev-Vandermonde matrix $V_T(\vec{x})$ and a vector $\vec{p} \in C^{n\times 1}$, let q be a fixed constant,

$$x_+ = \max_i |x_i| \le q < 1, \qquad b = \max_{m,k} \left| \frac{v_{m,k}}{t_k} \right|,$$

and $A = \max(\, |\, x_i - T_{n-1}(x_i)\, |,\ |\, T_n(x_i) - 1\, |\,)$ for $0 \le i \le n-1$. *Then, the vector $2V_T(\vec{x})\vec{p}$ can be approximated by a vector $\vec{u}^*(\vec{x})$ such that*

$$|\, u(x_i) - u^*(x_i)\, | \le \epsilon \qquad \textit{for a given } \epsilon > 0 \textit{ and for all } i\,,$$

and $O(\, n(L+\log n)\,)$ arithmetic operations suffice in order to compute $u^*(x_i)$, $i = 0, 1, \cdots, n-1$, where

$$L = \left\lceil \frac{\log(\, Abn/(\,(1-q)\epsilon\,)\,)}{\log(1/q)} \right\rceil.$$

Furthermore, $L = O(\log n)$, so that $O(n \log n)$ arithmetic operations suffice for the evaluation of $u^*(x_i)$, $i = 0, 1, \cdots, n-1$, if $\log(Ab/\epsilon) = O(\log n)$.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3.8 Numerical Experiments

In this section, we present the results of some numerical experiments for our algorithm of section 3.3, where we set $L = O(\log n)$, and for one of [MB]. These experiments do not cover all classes of the input values but, still, give some indication to the algorithm's behavior; in particular, they show that our algorithm is generally more reliable than one of [MB] and does not require to restrict the input nodes to a real line interval.

We have written the programs for our computation by using the C++ programming language and have compiled and linked these programs by using Microsoft Visual C++. The programs have been executed on a 386 and a Pentium PC, and identical results were obtained from both platforms. Evaluation of arithemetic functions have been performed by using Microsoft FP87 emulation library functions without utilizing math coprocessor.

In the following 3 examples, algorithm 1 denotes the algorithm of [MB] and algorithm 2 denotes the algorithm proposed in section 3.3. To control the computation errors, we also present the control computation results obtained either by using Horner's algorithm (actually due to I. Newton, cf. [Kn]) or, for the polynomials of the form

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

by computing $x^n - 1$ and dividing the results by $x - 1$. All the input, output, and intermediate results are stored by using double precision binary representation.

Example 1. Our first example shows that algorithm 1 (of [MB]) may fail even for low degree polynomials if we apply it to the input with substantial variation of the magnitude of the nodes $x_0, \cdots, x_{n-1}$. Algorithm 2 has no problems in treating such cases. We present the computational results of performing algorithm 1 step by step, to demonstrate the arising

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

difficulties. Notice that in steps 2 and 3, due to limited precision, the least significant digits of the intermediate results are lost, and this greatly affects the reliability of the final output of algorithm 1, i.e., two identical input values, $x_0 = x_2 = 1.5$, actually result in 2 different output values, $v_0 = 8.125$ and $v_2 = 6$.

Polynomial: $p(x) = x^3 + x^2 + x^1 + 1$.

Input vector: $x = (1.5, 20, 1.5, 1 \times 10^8)$.

Exact output: $v = (8.125, 8421, 8.125, 1 \times 10^{24})$.

Algorithm 1

Step 1: Compute coefficient vectors for $P_0(x) = (x - x_0)(x - x_1)$ and $P_1(x) = (x - x_2)(x - x_3)$:

$$\vec{P_0} = (\ 1, -21.5, 30\ ); \qquad \vec{P_1} = (\ 1, -1.000000015 \times 10^8, 1.5 \times 10^8\ ).$$

Step 2: Compute $M_i(x) = P(x)\ mod\ P_i(x),\ \ i = 0, 1.$

$$m_0 = (454.75, -674); \qquad m_1 = (1.000000025 \times 10^{16}, -1.5000000375 \times 10^{16}).$$

Step 3: Compute vector $\vec{v}$, where $v_i = M_{\lfloor \frac{i}{2} \rfloor}(x)\ mod\ (x - x_i),\ \ i = 0, 1, 2, 3$:

Output: $\vec{v} = [8.125, 8421, 6, 1 \times 10^{24}]^T$.

Algorithm 2

Output:

$$\vec{v} = \begin{pmatrix} 8.125 + 1.99483 \times 10^{-16} i \\ 8421 + 1.52282 \times 10^{-12} i \\ 8.125 + 1.99483 \times 10^{-16} i \\ 1 \times 10^{24} + 2.00008 \times 10^{8} i \end{pmatrix}.$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Our numerical experiments have showed similar behavior of algorithms 1 and 2 where $p(x) = (x^{16} - 1)/(x - 1)$ and the input points are 1.5, 100, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, and 15, that is, where the input polynomial has a higher degree, and the magnitudes of the input points vary on a much smaller scale.

Furthermore, the behavior remained similar where the polynomial coefficients have been selected more randomly.

Example 2:

Polynomial:

$$p(x) = 2.1x^{15} + x^{14} + 10x^{13} + 11x^{12} + 12x^{11} + 15x^{10} + 1.4x^9 + 5x^8 + 6x^7 + 7x^6 + 8x^5 + 6.3x^4 + 1.2x^3 + 3x^2 + 3.3x + 2.$$

Vector of input points:

$$\vec{x} = [1.5, 100, 2, 3, 50, 5, 30, 7, 20, 9, 10, 70, 77, 55, 42, 15]^T$$

Results of experiments:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Input points	Control Computation	Algorithm 1	Algorithm 2
1.5	6962.47	4.39805×10^{12}	$6962.47 - 8.45502 \times 10^{-12}i$
100	2.11101×10^{30}	2.11101×10^{30}	$2.11101 \times 10^{30} - 7.9277 \times 10^{15}i$
2	255709	1.28849×10^{10}	$255709 - 3.54152 \times 10^{-10}i$
3	5.97974×10^{7}	2.57698×10^{10}	$5.97974 \times 10^{7} - 1.08648 \times 10^{-7}i$
50	6.48221×10^{25}	6.48221×10^{25}	$6.48221 \times 10^{25} - 2.39968 \times 10^{11}i$
5	8.58207×10^{10}	8.58993×10^{10}	$8.58207 \times 10^{10} - 2.17322 \times 10^{4}i$
30	3.07765×10^{22}	3.07765×10^{22}	$3.07765 \times 10^{22} - 1.11633 \times 10^{8}i$
7	1.17973×10^{13}	1.17973×10^{13}	$1.17973 \times 10^{13} - 3.43946 \times 10^{-2}i$
20	7.13181×10^{19}	7.13181×10^{19}	$7.13181 \times 10^{19} - 2.51662 \times 10^{5}i$
9	4.84203×10^{14}	4.84232×10^{14}	$4.84203 \times 10^{14} - 1.5167 \times 10^{1}i$
10	2.31235×10^{15}	2.31282×10^{15}	$2.31235 \times 10^{15} - 7.41649 \times 10^{1}i$
70	1.00475×10^{28}	1.00475×10^{28}	$1.00475 \times 10^{28} - 3.75053 \times 10^{13}i$
77	4.19381×10^{28}	4.19381×10^{28}	$4.19381 \times 10^{28} - 1.56835 \times 10^{14}i$
55	2.70455×10^{26}	2.70455×10^{26}	$2.70455 \times 10^{26} - 1.00389 \times 10^{12}i$
42	4.75383×10^{24}	4.75383×10^{24}	$4.75383 \times 10^{24} - 1.74988 \times 10^{10}i$
15	9.69772×10^{17}	9.69777×10^{17}	$9.69772 \times 10^{17} - 3.3218 \times 10^{3}i$

These experiments show that the output values of $p(x)$ at points 1.5, 2, 3 computed by algorithm 1 are completely contaminated by the errors, whereas algorithm 2 closely approximates the values $p(x)$ at all the 16 points x, as comparison with the results of control computation shows.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Example 3: From the proof of proposition 3.3.1 and from remark 3.3.2, we can see that the value L must be increased (in order to ensure the desired output accuracy) as $\min_{i=0,1,\cdots,n-1} x_i$ decreases. On the other hand, increasing L makes algorithm 2 less efficient. In the following example, for a simple polynomial, where $n = 4$, algorithm 2 cannot obtain the results within desired relative error bound until L is far greater than n.

Polynomial: $P(x) = x^3 + x^2 + x + 1$.

Input points	Control computation	Algorithm 1	Algorithm 2 ($L = 4$)	Algorithm 2 ($L = 40$)
1.1	4.641	4.641	$(1.47113 - 2.29133 \times 10^{-17}i)$	$(4.53846 - 7.06877e - 017i$
1.2	5.368	5.368	$(3.2.77927 - 8.21258 \times 10^{-17}i)$	$(5.36435 - 1.58514e - 016i$
1.3	6.187	6.187	$(4.02076 - 1.69062 \times 10^{-16}i)$	$(6.18683 - 2.6014e - 016i)$
1.4	7.104	7.104	$(5.25477 - 2.79601 \times 10^{-16}i)$	$(7.10399 - 3.77997e - 016i$

In this particular case, however, the problem can be remedied by shifting the variable and transforming the coefficients of the original polynomial and the input points $x_0, \cdots, x_{n-1}$, respectively. The new resulting coefficients and new input points can be easily evaluated. We have obtained the output with the desired precision by choosing $L = 4$ after shifting $P(x)$ to $Q(x) = P(x - 50)$.

Polynomial: $Q(x) = P(x - 50) = x^3 - 149x^2 + 7401x - 122549$.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Input points	Control computation	Algorithm 1	Algorithm 2 ($L = 4$)
51.1	4.641	4.641	$(4.641 - 5.03155e - 007i)$
51.2	5.368	5.368	$(5.368 - 5.06079e - 007i)$
51.3	6.187	6.187	$(6.187 - 5.09014e - 007i)$
51.4	7.104	7.104	$(7.104 - 5.11961e - 007i)$

In the general case, numerical stability problems may arise due to the variable shift when we operate with a polynomial of a higher degree. Proper adaptation of shifting techniques to various input data could be an interesting topic for further study.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

References

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

[AGR] J. Ambrosiano, L. Greengard, and V. Rokhlin, The Fast Multipole Method for Gridless Particle Simulation, *Computer Physics Communications*, **48**, pp.115-125, 1988.

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

[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 and V. Y. Pan, *Polynomial and Matrix Computations, Volume 1: Fundamental Algorithms*, Birkhauser, Boston, 1994.

[C89] R. H. Chan, G. Strang, Toeplitz Equations by Conjugate Gradients with Circulant Preconditioner. *SIAM J. Sci. Stat. Comput.*, 10, 1, 104-119, 1989.

[C91] 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.

[CGR] J. Carier, L. Greengard, and V. Rokhlin, A Fast Adaptive Multipole Algorithm for Particle Simulation. *SIAM J. Sci. Stat. Comput.*, **9**, 4, pp.669-686, 1988.

[CH89] R. H. Chan, Circulant Preconditioners for Hermitian Toeplitz System, *SIAM J. Matrix Anal. Appl.*, 10, 4, 542-550.

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

[D] F. R. dehoog, On the Solution of Toeplitz Systems, *Linear Algebra Appl.* 88/89, 123-138, 1987.

[DGR] A. Dutt, M. Gu, and V. Rokhlin, Fast Algorithms for Polynomial Interpolation, Integration and Differentiation, Research Report RR-977, *Computer Science Department, Yale University*, 1993.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[FHR] T. Fink, G. Heinig, and K. Rost, An Inversion Formula and Fast Algorithms for Cauchy-Vandermonde Matrices, *Linear Algebra and Its Appl.*, **183**, pp.179-191, 1993.

[GI] W. Gautschi and G. Inglese, Lower Bounds for the Condition Number of Vandermonde Matrices, *Numerische Math.*, 52, 241-250, 1988.

[GKK] I. Gohberg, T. Kailath, and I. Koltracht, Efficient Solution of Linear Systems of Equations with Recursive Structure, *Linear Algebra and Its Appl.*, **80**, pp.81-113, 1986.

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

[GKO] I. Gohberg, T. Kailath, and V. Olshevsky, Fast Gaussian Elimination with Partial Pivoting for Matrices with Displacement Structure, *Math. Comp.*64, 1557-1576, 1995.

[GL] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, Maryland, 1989.

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

[GO1] I. Gohberg and V. Olshevsky, Fast Inversion of Chebyshev-Vandermonde Matrices, *Numerische Math.*, **67**, 1, pp.71-92, 1994.

[GR] L. Greengard and V. Rokhlin, A Fast Algorithm for Particle Simulations, *J. Comput. Phys.*, **73**, pp.325-348, 1987.

[H] G. Heinig, Inversion of Generalized Cauchy Matrices and Other Classes of Structured Matrices, *IMA Volume: Linear Algebra for Signal Processing in Math and Its Applications*, **69**, pp.95-114, Springer, 1994.

[HR] H. Heinig and K. Rost, *Algebraic Methods for Toeplitz-like Matrices and Operators, Operator Theory*, 13, Birkhaeuser, 1984.

[KKM] T. Kailath, S.Y. Kung, M. Morf, Displacement Ranks of Matrices and Linear Equations, *J.Math. Anal. Appl.*, 68, 2, 395-407, 1979.

[Kn] D. E. Knuth, *The Art of Computer Programming: Seminumerical Algorithms*, **2**, Addison-Wesley, Reading, Massachusetts, 1981.

[KO] T. Kailath and V. Olshevsky, Displacement Structure Approach to Chebyshev-Vandermonde and Related Matrices, *Integral Equations and Operator Theory*, 22, 65-92, 1995.

[KP91] E. Kaltofen, V. Y. Pan, Processor Efficient Parallel Solution of Linear Systems over an Abstract Field, *Proc. 3rd Ann. ACM Symp. on Parallel Algorithms and Architectures*, 180-191, ACM Press, New York, 1991.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[KP92] E. Kaltofen, V. Y. Pan, Processor Efficient Parallel Solution of Linear Systems II. The Positive Characteristic and Singular Cases, *Proc. 33rd Ann. IEEE Symp. on Foundation of Computer Science*, 714-723, IEEE Computer Society Press, 1992.

[KS] T. Kailath and A. Sayed, Displacement Structure: Theory and Applications, *SIAM Review*, 37, 3, 297-386, 1995.

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

[MB] R. Moenck and A. B. Borodin, Fast Modular Transform via Division, *Proc. 13th Ann. IEEE Symp. on Switching and Automata Theory*, pp.90-96, 1972.

[P87] V. Y. Pan, Complexity of Parallel Matrix Computations, *Theoretical Computer Science*, 54, 65-85, 1987.

[P89] V. Y. Pan, On Some Computations with Dense Structured Matrices, *Proc. ACM-SIGSAM Internat. Symp. on Symbolic and Algebriac Computing*, 34-42, ACM Press, New York, 1989.

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

[P92a] V. Y. Pan, Parallel Solution of Toeplitz-like Linear Systems, *J. of Complexity*, 8, pp. 1-21, 1992.

[P92b] V. Y. Pan, Parametrization of Newton's Iteration for Computations with Structured Matrices and Applications, *Computers and Mathematics (with applications)*, 24, 3, 61-75, 1992.

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

[P93a] V. Y. Pan, Decreasing the Displacement Rank of a Matrix, *SIAM Journal of Matrix Analysis and Appl.*, **14**, 1, pp.118-121, 1993.

[P95] V. Y. Pan, An Algebraic Approach to Approximate Evaluation of a Polynomial on a Set of Real Points, *Advances in Computational Math.*, **3**, pp.41-58, 1995.

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

[PSLT] V. Y. Pan, A. Sadikou, E. Landowne, and O. Tiga, A New Approach to Fast Polynomial Interpolation and Multipoint Evaluation, *Computers Math. with Appl.*, **25**, 9, pp.25-30, 1993.

[PZ97] Victor. Pan, Ailong Zheng, Fast Algorithms for Cauchy-like Comptations and an Extention, *Linear Algebra and Its Applications*, to appear, 1997.

[PZDH95] V. Y. Pan, A. L. Zheng, O. Dias, and X. Huang, A Fast, Preconditioned Conjugate Gradient Toeplitz and Toeplitz-like Solvers, *Computers and Mathematics (with applications)*, 30, 8, 57-63, 1995.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[PZHD97] V. Y. Pan, A. L. Zheng, X. Huang, and O. Dias, Newton's Iteration for Inversion of Cauchy-like and Other Structured Matricesnd Mathematics, *J. of Complexity* , 13, 108-124, 1997.

[PZHY97] V. Y. Pan, A. L. Zheng, X. Huang, and Y. Yu, Fast Multipoint Polynomial Evaluation and Interpolation via Computations with Structured Matrices, *Annals of Numerical Mathematics*, 4. 483-510, 1997.

[R85] V. Rokhlin, Rapid Solution of Integral Equations of Classical Potential Theory, *Journal of Computational Physics*, **60**, pp.187-207, 1985.

[R88] V. Rokhlin, A Fast Algorithm for the Discrete Laplace Transformation, *Journal of Complexity*, **4**, pp.12-32, 1988.

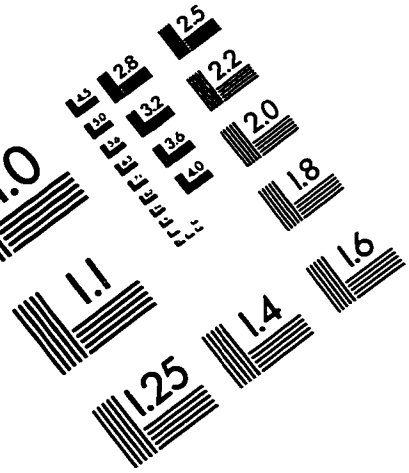
[S] G. Strang, A Proposal for Toeplitz Matrix Calculations, *Studies in Appl. Math*, 74, 171-176, 1986.

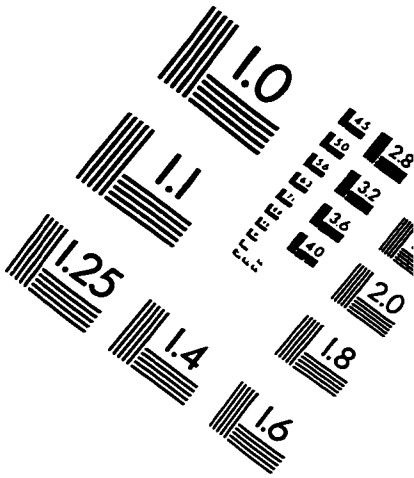
[T] W. F. Trench, A Note on a Toeplitz Inversion Formula, *Linear Algebra Appl.*, 29, 55-61,1990.

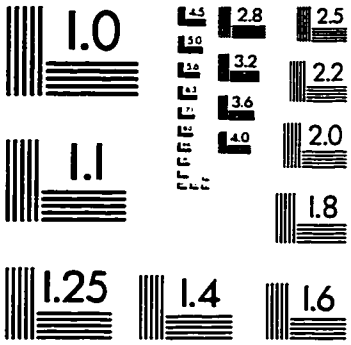
[W] D. H. Wood, Product Rules for the Displacement of Near-Toeplitz Matrices, *Linear Algebra Appl.*, 188, 189, pp. 641-663, 1993.

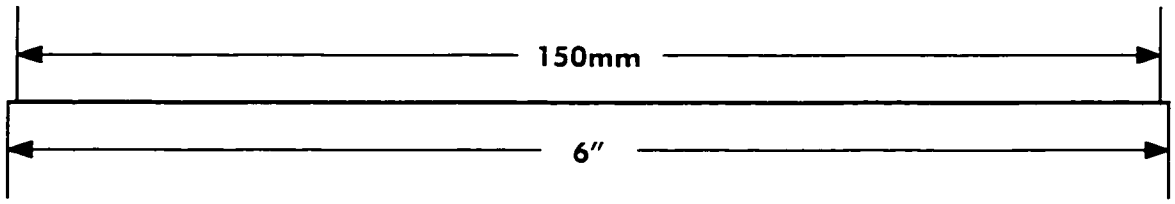
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

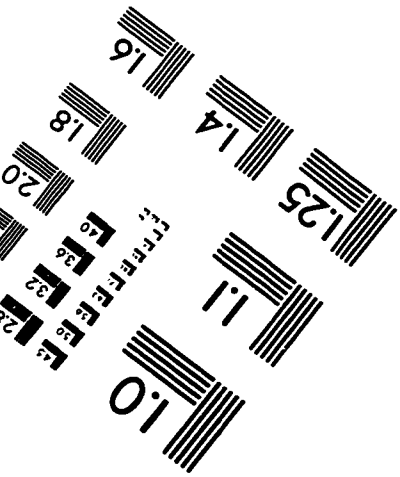
IMAGE EVALUATION TEST TARGET (QA-3)

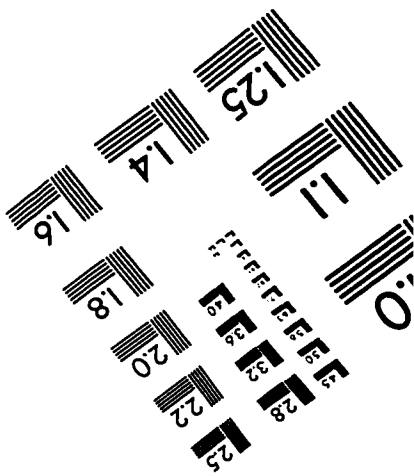












APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989

© 1993, Applied Image, Inc., All Rights Reserved

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.