Fast algorithms with preprocessing for matrix–vector multiplication problems

I.Gohberg and V.Olshevsky

School of Mathematical Sciences
Raymond and Beverly Sackler Faculty of Exact Sciences
Tel Aviv University, Ramat Aviv 69978, Israel.

e–mail: gohberg@math.tau.ac.il, vadim@math.tau.ac.il.

Journal of Complexity, (1994).

Abstract

In this paper the problem of complexity of multiplication of a matrix with a vector is studied for Toeplitz, Hankel, Vandermonde and Cauchy matrices and for matrices connected with them (i.e. for transpose, inverse and transpose to inverse matrices). The proposed algorithms have complexities at most $O(n \log^2 n)$ flops and in a number of cases improve the known estimates. In these algorithms, in a separate *preprocessing* phase, are singled out all the actions on the preparation of a given matrix, which aimed at the reduction of the complexity of the second stage of computations directly connected with the multiplication by an arbitrary vector. Incidentally, the effective algorithms for computing the Vandermonde determinant and the determinant of a Cauchy matrix, are given.

0 Introduction

Let the matrix $A \in \mathbf{C}^{n\times n}$ be given by all its n^2 entries. The problem is to compute the product $A\mathbf{b}$ of a matrix A by input vector $\mathbf{b} \in \mathbf{C}^n$. Using the standard rule of a matrix times vector multiplication, the above problem can be solved in $2n^2 - n$ flops (i.e. float point operations of addition, subtraction, multiplication and division). In the following three simple examples the special structure of a matrix enables faster computation of the product by a vector.

Examples

1) *Band matrices.*

Let $A = (a_{ij})_{i,j=0}^{n-1}$ be a band matrix with the width of the band β (i.e., by definition, $a_{ij} = 0$ while $\mid i - j \mid > \frac{\beta-1}{2}$). Obviously, in this case the computing the product $A\mathbf{b}$ costs $(2\beta - 1)n$ flops.Ξ

2) *Small rank matrices.*

Let matrix $A \in \mathbf{C}^{n\times n}$ with rank $\alpha \in \mathbf{R}$ be given in the form of the outer sum of α terms :

$$A = \sum_{m=1}^{\alpha} \mathbf{h}_m \cdot \mathbf{g}_m^T \qquad (\mathbf{h}_m, \mathbf{g}_m \in \mathbf{C}^n). \tag{0.1}$$

The representation (0.1) shows that A can be multiplied by an arbitrary vector in $(4\alpha-1)n-\alpha$ flops.Ξ

3) *Semiseparable matrices.*

Such a matrix $A = (a_{ij})_{i,j=0}^{n-1}$ is defined by the equalities

$$a_{ij} = \begin{cases} \sum_{m=1}^{\alpha} f_{m,i} \cdot g_{m,i} & i < j \\ 0 & i \geq j \end{cases},$$

where $\mathbf{f}_m = (f_{m,i})_{i=0}^{n-1}$, $\mathbf{g}_m = (g_{m,i})_{i=0}^{n-1}$ $(m = 1, 2, ..., \alpha)$ are given vectors from $\mathbf{C}^n$. In this case

$$A = \sum_{m=1}^{\alpha} \mathrm{diag}(\mathbf{f}_m) \cdot \begin{bmatrix} 0 & 1 & 1 & \cdots & 1 \\ 0 & 0 & 1 & \cdots & 1 \\ \vdots & & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & 1 \\ 0 & \cdots & \cdots & \cdots & 0 \end{bmatrix} \cdot \mathrm{diag}(\mathbf{g}_m), \tag{0.2}$$

where by $\mathrm{diag}(\mathbf{f})$ is denoted the diagonal matrix, whose entries on the main diagonal equal the coordinates of the vector $\mathbf{f} \in \mathbf{C}^n$. The product of a semiseparable matrix by an arbitrary vector can be computed using (0.2) in $(4\alpha-1)n-\alpha$ flops. Obviously, the analogous estimate holds for the transpose to the matrices of the form (0.2).Ξ

Let matrix $A \in \mathbf{C}^{n\times n}$ be given. The task is to compute the products $A\mathbf{b}_0, A\mathbf{b}_1, ...$ of the matrix A by input vectors $\mathbf{b}_0, \mathbf{b}_1, ... \in \mathbf{C}^n$ in the smallest possible time. Such a situation arises naturally in a number of computational problems, for example, when we have to carry out the iterations with a given matrix. The problem of computing the product of a matrix

by a matrix can also be solved in the above framework. In the latter case the columns of a second matrix are interpreted as input vectors.

In accordance with the accepted scheme, we single out all the computations which are not dependent upon the input vectors $\mathbf{b}_0, \mathbf{b}_1, ...$ Accordingly, the proposed algorithms will be divided into two separate phases :

I. Preprocessing for matrix $A \in \mathbf{C}^{n \times n}$,

II. Application to the vector,

The first phase also contains the preparation of the given matrix, which enables the second phase to be accomplished more effectively.

In the present paper we embody this scheme and propose a number of fast algorithms for matrices with a certain structure, namely for transposed Vandermonde matrix, for transpose to inverse of Vandermonde matrix, for Cauchy matrices and for matrices connected with them (i.e. for transpose, inverse and transpose to inverse matrices).

1 Background

1.1 Basic algorithms

In this paper a limited number of well known algorithms is used intensively. These algorithms are listed below and accompanied by the estimates of their complexities. The particular implementations of these *basic algorithms* and complexity analysis for them can be found in various sources (see, for example, Aho, Hopcroft and Ullman (1976)).

(BA1) *Evaluation algorithm.*
An algorithm of evaluation of a $(n-1)$ degree polynomial at n points. The complexity of this algorithm will be denoted by $\varepsilon(n)$. It is well known that

$$\varepsilon(n) \leq O(n \ \log^2 n).$$

(BA2) *Interpolation algorithm.*
An algorithm of interpolation of a $(n-1)$ degree polynomial from its values at n points. The complexity of the interpolation algorithm is denoted by $\iota(n)$ and, as it is well known,

$$\iota(n) \leq O(n \ \log^2 n).$$

(BA3) *Fast Fourier Transform algorithm.*
The discrete Fourier transform of the vector $\mathbf{r} = (r_i)_{i=0}^{n-1} \in \mathbf{C}^n$ is by definition the vector $(\sum_{k=0}^{n-1} r_k \omega^{ik})_{i=0}^{n-1} \in \mathbf{C}^n$, where ω is the primitive n-th root from unity. The discrete Fourier transform can be computed by well known methods, collectively named *Fast Fourier Transform* (FFT). It is well known that for the complexity $\phi(n)$ of computing one FFT of order n, the following estimate holds

$$\phi(n) \leq O(n \ \log n).$$

1.2 Basic matrices

In this subsection some matrices, which are known to have a lower than $O(n^2)$ flops complexity of multiplication with a vector, are considered. Note that the methods for fast computing of the products of these *basic matrices* by vectors are based on the algorithms (BA1) - (BA3).

(BM1) *Vandermonde matrix.*

By definition, the Vandermonde matrix $V(\mathbf{t})$ is the matrix of the form

$$V(\mathbf{t}) = \begin{bmatrix} 1 & t_0 & \cdots & t_0^{n-1} \\ 1 & t_1 & \cdots & t_1^{n-1} \\ \vdots & \vdots & & \vdots \\ 1 & t_{n-1} & \cdots & t_{n-1}^{n-1} \end{bmatrix}, \quad \text{with} \quad \mathbf{t} = (t_i)_{i=0}^{n-1} \in \mathbf{C}^n.$$

Obviously, the problem of computing the product $V(\mathbf{t})\mathbf{b}$ of the matrix $V(\mathbf{t})$ by a vector $\mathbf{b} = (b_i)_{i=0}^{n-1}$ is equivalent to the problem of evaluation of $p(\lambda) = \sum_{i=0}^{n-1} b_i \cdot \lambda^i$ at n points $t_0,\ t_1,\ \ldots\ t_{n-1}$. Thus, the product $V(\mathbf{t})\mathbf{b}$ can be computed using the algorithm (BA1) in $\varepsilon(n)$ flops.

(BM2) *Inverse of Vandermonde matrix.*

Consider the problem of application to a vector of the matrix $A \in \mathbf{C}^{n\times n}$, which is defined as the inverse of the given Vandermonde matrix $V(\mathbf{t})$ ($\mathbf{t} \in \mathbf{C}^n$). It is easy to see that the product $A\mathbf{b}$ can be computed using the algorithm (BA2). Thus, for the inverse of Vandermonde matrix, the application to a vector costs $\iota(n)$ flops.

(BM3) *Fourier matrix and its inverse.*

Let $\omega = e^{\frac{2\pi i}{n}}$ be primitive n-th root from the unity. Consider the Fourier matrix $\mathcal{F} = \frac{1}{\sqrt{n}} \cdot V(\ \ )$ with $\ \ = (\omega^i)_{i=0}^{n-1}$ and its inverse $\mathcal{F}^{-1} = \mathcal{F}^*$, where superscript * means conjugate transpose. The products $\mathcal{F}\mathbf{b}$ and $\mathcal{F}^*\mathbf{b}$ can be computed using the algorithm (BA3) in $\phi(n)$ flops.

(BM4) *Factor circulant.*

By $\mathrm{Circ}_\varphi(\mathbf{r})$ will be denoted *φ-circulant* with the first column $\mathbf{r} = (r_i)_{i=0}^{n-1}$, i.e. matrix of the form

$$\mathrm{Circ}_\varphi(\mathbf{r}) = \begin{bmatrix} r_0 & \varphi r_{n-1} & \cdots & \cdots & \varphi r_1 \\ r_1 & r_0 & \varphi r_{n-1} & & \vdots \\ \vdots & r_1 & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \varphi r_{n-1} \\ r_{n-1} & \cdots & \cdots & r_1 & r_0 \end{bmatrix}.$$

The matrix $\mathrm{Circ}_1(\mathbf{r})$ is referred to as a *circulant*. It is known (see Cline, Plemmons and Worm (1974)) that the matrix $\mathrm{Circ}_\varphi(\mathbf{r})$ admits the following decomposition :

$$\mathrm{Circ}_\varphi(\mathbf{r}) = D_\varphi^{-1} \cdot \mathcal{F}^* \cdot \Lambda_\varphi(\mathbf{r}) \cdot \mathcal{F} \cdot D_\varphi, \tag{1.1}$$

where $\mathcal{F}$ is the Fourier matrix, $\Lambda_\varphi(\mathbf{r}) = \mathrm{diag}(\mathcal{F}D_\varphi\mathbf{r})$ and $D_\varphi = \mathrm{diag}((\xi^i)_{i=0}^{n-1})$ with $\xi \in \mathbf{C}$ satisfying the condition $\xi^n = \varphi$. From (1.1) it follows that if the coordinates of

the vector $= (\xi^i)_{i=0}^{n-1} \in \mathbf{C}^n$ are given, then the product of the matrix $\mathrm{Circ}_\varphi(\mathbf{r})$ by an arbitrary vector can be computed in

$$2\phi(n) + O(n)$$

flops. In this case the preprocessing phase consists of computation of the central factor $\Lambda_\varphi(\mathbf{r})$ in the right-hand side of (1.1) and costs

$$\phi(n) + O(n)$$

flops.

(BM5) *Toeplitz matrices*

The product of Toeplitz matrix $A = (a_{i-j})_{i,j=0}^{n-1}$ by a vector can be computed by embedding of the matrix A in the circulant of double size. Correspondingly, the amount of operations is

$$2\phi(2n) + O(n) = 4\phi(n) + O(n)$$

flops after

$$\phi(2n) + O(n) = 2\phi(n) + O(n)$$

preprocessing.

(BM6) *Inverses of Toeplitz matrices*

In computations with the inverses of Toeplitz matrices, the Gohberg-Semencul formula (see Gohberg and Semencul (1972) or Gohberg and Feldman (1974)) is useful. This formula represents the inverse of a Toeplitz matrix in the form of the sum of products of triangular Toeplitz matrices. Namely, if for the Toeplitz matrix T the equations

$$T\mathbf{x} = \mathbf{e}_0, \qquad T\mathbf{y} = \mathbf{e}_{n-1} \tag{1.2}$$

with $\mathbf{e}_0 = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}^T$, $\mathbf{e}_{n-1} = \begin{bmatrix} 0 & \cdots & 0 & 1 \end{bmatrix}^T$ have solutions $\mathbf{x} = (x_i)_{i=0}^{n-1}$, $\mathbf{y} = (y_i)_{i=0}^{n-1}$ and $x_0 \neq 0$, then T is invertible and

$$T^{-1} = \frac{1}{x_0}\Bigg(\begin{bmatrix} x_0 & 0 & \cdots & \cdots & 0 \\ x_1 & x_0 & & & \vdots \\ \vdots & x_1 & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ x_{n-1} & \cdots & \cdots & x_1 & x_0 \end{bmatrix} \cdot \begin{bmatrix} y_{n-1} & y_{n-2} & \cdots & \cdots & y_0 \\ 0 & y_{n-1} & y_{n-2} & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & y_{n-2} \\ 0 & \cdots & \cdots & 0 & y_{n-1} \end{bmatrix} -$$

$$- \begin{bmatrix} 0 & 0 & \cdots & \cdots & 0 \\ y_0 & 0 & & & \vdots \\ \vdots & y_0 & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ y_{n-2} & \cdots & \cdots & y_0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & x_{n-1} & \cdots & \cdots & x_1 \\ 0 & 0 & x_{n-1} & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & x_{n-1} \\ 0 & \cdots & \cdots & \cdots & 0 \end{bmatrix} \Bigg). \tag{1.3}$$

On the basis of formula (1.3) the number of fast algorithms for the inversion of Toeplitz matrices was elaborated (see Brent, Gustavson and Yun (1980), de Hoog (1987), Chun and Kailath (1991), where the methods for solving the equations of the form (1.2) are suggested). Denote by $\tau(n)$ the complexity of solving one equation of the form (1.2). According to Brent, Gustavson and Yun (1980), de Hoog (1987), Chun and Kailath (1991)

$$\tau(n) \le O(n\ \log^2 n).$$

Thus, if matrix $A \in \mathbf{C}^{n\times n}$ is the inverse of the given Toeplitz matrix, then from the above arguments follows that after $2\tau(n) + 8\phi(n)$ flops preprocessing the product of A by an arbitrary vector can be computed by (1.3) in $16\phi(n) + O(n)$ flops. Below we show how this complexity can be reduced.

If for Toeplitz matrix $T \in \mathbf{C}^{n\times n}$ the equations (1.2) have solutions $\mathbf{x} = (x_i)_{i=0}^{n-1}$, $\mathbf{y} = (y_i)_{i=0}^{n-1}$ and $x_0 \neq 0$, then

$$T^{-1} = \frac{1}{x_0(\varphi - \psi)}(\mathrm{Circ}_\psi(\mathbf{x}) \cdot \mathrm{Circ}_\varphi(Z_\varphi \mathbf{y}) - \mathrm{Circ}_\psi(Z_\psi \mathbf{y}) \cdot \mathrm{Circ}_\varphi(\mathbf{x})), \qquad (1.4)$$

where numbers φ and ψ $(\neq \varphi)$ are arbitrary and

$$Z_\varphi = \begin{bmatrix} 0 & \cdots & \cdots & 0 & \varphi \\ 1 & 0 & & & 0 \\ 0 & 1 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & 0 \end{bmatrix} \qquad (\varphi \neq 0)$$

is the φ-cyclic lower shift matrix. Formula (1.4) was obtained in Ammar and Gader (1990) for positive definite Toeplitz matrices and $\varphi = 1,\ \ \psi = -1$, and was extended to the general case in Gohberg and Olshevsky (1992). Note, in the latter paper one can also find other factor circulant decompositions for the inverses of Toeplitz matrices, which are useful in the fast computations with Toeplitz matrices. Furthermore, representing each factor circulant in the left-hand side of (1.4) in the form of (1.1), we finally get

$$T^{-1} = \frac{1}{x_0(\varphi - \psi)} D_\psi^{-1} \cdot \mathcal{F}^* \cdot \Big(\Lambda_\psi(\mathbf{x}) \cdot \mathcal{F} \cdot D_\psi \cdot D_\varphi^{-1} \cdot \mathcal{F}^* \cdot \Lambda_\varphi(Z_\varphi \mathbf{y}) -$$

$$- \Lambda_\psi(Z_\psi \mathbf{y}) \cdot \mathcal{F} \cdot D_\psi \cdot D_\varphi^{-1} \cdot \mathcal{F}^* \cdot \Lambda_\varphi(\mathbf{x}) \Big) \cdot \mathcal{F} \cdot D_\varphi, \qquad (1.5)$$

where $\mathcal{F}$ is a Fourier matrix and the diagonal matrices D_φ and $\Lambda_\varphi(\mathbf{r})$ $(\mathbf{r} \in \mathbf{C}^n)$ are defined as in (BM4). From (1.5) it follows that after

$$2\tau(n) + 4\phi(n) + O(n) \qquad (1.6)$$

flops preprocessing, the product of the inverse of the Toeplitz matrix by an arbitrary vector can be computed in

$$6\phi(n) + O(n) \qquad (1.7)$$

flops. The last estimate is obtained here without the additional requirement for matrix A to be Hermitian. In the Hermitian case this result is obtained in the earlier paper Ammar and Gader (1990). Note, (1.6) and (1.7) imply that the product of the inverse of Toeplitz matrix by an arbitrary matrix can be computed in $6\phi(n)n + O(n^2)$ flops.

(BM7) *Hankel matrices.*

Let us recall that an arbitrary Hankel matrix $H = (a_{i+j})_{i,j=0}^{n-1}$ can be transformed in the Toeplitz matrix by means of multiplication from the right by the reverse identity matrix

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

Therefore the complexity of the multiplication with a vector for the class of Hankel matrices coincides with the same complexity for the class of Toeplitz matrices.

2 Transpose to Vandermonde and to inverse of Vandermonde matrices

2.1 Relations between Vandermonde matrices and Bezoutians

Let $n \in \mathbf{R}$ be the maximal degree of two polynomials $f(\lambda)$ and $g(\lambda)$, then the bilinear form

$$\mathcal{B}_{f,g}(\lambda,\mu) = \frac{f(\lambda)\cdot g(\mu) - f(\mu)\cdot g(\lambda)}{\lambda - \mu} = \sum_{i,j=0}^{n-1} b_{ij}\lambda^i\mu^j$$

is called the *Bezoutian* of $f(\lambda)$ and $g(\lambda)$. The matrix

$$B_{f,g} = (b_{ij})_{i,j=0}^{n-1} \quad ,$$

whose entries are determined by the coefficients of the bilinear form $\mathcal{B}_{f,g}(\lambda,\mu)$, will be referred to as a *Bezout matrix* which corresponds to the polynomials $f(\lambda)$ and $g(\lambda)$. Obviously,

$$\begin{bmatrix} 1 & \lambda & \lambda^2 & \cdots & \lambda^{n-1} \end{bmatrix} \cdot B_{f,g} \cdot \begin{bmatrix} 1 \\ \mu \\ \mu^2 \\ \vdots \\ \mu^{n-1} \end{bmatrix} = \mathcal{B}_{f,g}(\lambda,\mu). \tag{2.1}$$

The equality (2.1) yields the following useful property of the Bezout matrix :

$$V(\mathbf{s}) \cdot B_{f,g} \cdot V(\mathbf{t})^T = (\mathcal{B}_{f,g}(s_i,t_j))_{i,j=0}^{n-1} \quad , \tag{2.2}$$

where $\mathbf{s} = (s_i)_{i=0}^{n-1}$, $\mathbf{t} = (t_i)_{i=0}^{n-1} \in \mathbf{C}^n$. From (2.2) and obvious equality

$$\mathcal{B}_{f,g}(\lambda, \lambda) = f'(\lambda) \cdot g(\lambda) - f(\lambda) \cdot g'(\lambda)$$

it follows that if $t_0, t_1, ..., t_{n-1}$ are n simple roots of the polynomial $f(\lambda)$, then

$$V(\mathbf{t}) \cdot B_{f,g} \cdot V(\mathbf{t})^T = \mathrm{diag}((f'(t_i) \cdot g(t_i))_{i=0}^{n-1}). \tag{2.3}$$

The latter formula appeared in Lander (1974) and can also be found in Heinig and Rost (1984). On the basis of (2.3) in the next subsection the fast algorithm for multiplication of transpose Vandermonde matrix by a vector is presented.

2.2 Transpose to Vandermonde matrix and its inverse

For vector $\mathbf{t} = (t_i)_{i=0}^{n-1}$ $(t_i \neq t_j$ while $i \neq j)$ set

$$f(\lambda) = \prod_{i=0}^{n-1} (\lambda - t_i) = \lambda^n + \sum_{i=0}^{n-1} r_i \cdot \lambda^i, \quad \text{and} \quad g(\lambda) = \varphi - \lambda^n, \tag{2.4}$$

where φ is such that $\varphi \neq t_i^n$ $(i = 0, 1, ..., n-1)$. Under these conditions the matrix in the right-hand side of (2.3) is invertible. Hence, all the matrices in the left-hand side of (2.3) are also invertible, and moreover

$$V(\mathbf{t})^T = B_{f,g}^{-1} \cdot V(\mathbf{t})^{-1} \cdot \mathrm{diag}((f'(t_i) \cdot g(t_i))_{i=0}^{n-1}). \tag{2.5}$$

Furthermore, matrix $B_{f,g}$ is the Bezout matrix of two polynomials, one of them having the special form $g(\lambda) = \varphi - \lambda^n$. This fact allows to receive for $B_{f,g}$ and for its inverse a representation involving a factor circulant. To prove this, let us compute the Bezoutian $\mathcal{B}_{f,g}(\lambda, \mu)$ of the polynomials $f(\lambda)$ and $g(\lambda)$, defined by (2.4) :

$$\mathcal{B}_{f,g}(\lambda, \mu) = \varphi \frac{f(\lambda) - f(\mu)}{\lambda - \mu} - \frac{f(\lambda) \cdot \mu^n - \lambda^n \cdot f(\mu)}{\lambda - \mu} =$$

$$= \varphi \sum_{i=1}^{n} r_i (\lambda^{i-1} + \lambda^{i-2}\mu + ... + \mu^{i-1}) + \sum_{i=0}^{n-1} r_i (\lambda^i \mu^{n-1} + \lambda^{i+1}\mu^{n-2} + ... + \lambda^{n-1}\mu^i).$$

with $r_n = 1$. From the last identity it can be easily seen that the corresponding Bezout matrix $B_{f,g}$ has the special form

$$B_{f,g} = \begin{bmatrix} \varphi r_1 & \varphi r_2 & \cdots & \varphi r_{n-1} & r_0 + \varphi \\ \varphi r_2 & & \iddots & \iddots & r_1 \\ \vdots & \iddots & \iddots & \iddots & \vdots \\ \varphi r_{n-1} & \iddots & \iddots & & r_{n-2} \\ r_0 + \varphi & r_1 & \cdots & r_{n-2} & r_{n-1} \end{bmatrix} = \mathrm{Circ}_\varphi(\mathbf{r} + \varphi \mathbf{e}_0) \cdot J \ , \tag{2.6}$$

where J is, as above, the reverse identity matrix. Substituting (2.6) in (2.5), we have

$$V(\mathbf{t})^T = J \cdot \mathrm{Circ}_\varphi(\mathbf{r} + \varphi \mathbf{e}_0)^{-1} \cdot V(\mathbf{t})^{-1} \cdot \mathrm{diag}((f'(t_i) \cdot g(t_i))_{i=0}^{n-1}). \tag{2.7}$$

On the basis of the last formula the following algorithm is proposed.

Algorithm 2.1. Computing the product of transpose Vandermonde matrix with an arbitrary vector.

I. Preprocessing.

Complexity : $\iota(n) + \varepsilon(n) + \phi(n) + 2n\log n + O(n)$ flops.

Input : Vector $\mathbf{t} = (t_i)_{i=0}^{n-1} \in \mathbf{C}^n \quad (t_i \neq t_j \text{ while } i \neq j)$.

Output : Parameters of the representation $V(\mathbf{t})^T$ in the form

$$V(\mathbf{t})^T = J \cdot D_\varphi \cdot \mathcal{F} \cdot \Lambda_\varphi(\mathbf{r} + \varphi\mathbf{e}_0)^{-1} \cdot \mathcal{F}^* \cdot D_\varphi^{-1} \cdot V(\mathbf{t})^{-1} \cdot D_{f,g}. \qquad (2.8)$$

1. Compute the coefficients of the polynomial $f(\lambda) = \prod_{i=0}^{n-1}(\lambda - t_i) = \lambda^n + \sum_{i=0}^{n-1} r_i \cdot \lambda^i$ in $\iota(n)$ flops.
2. Compute in $O(n)$ flops the coefficients of the polynomial $f'(\lambda)$.
3. Evaluate in $\varepsilon(n)$ flops the values $f'(t_i)$ $(i = 0, 1, ..., n-1)$.
4. Evaluate in at most $2n\log n + n$ flops the values $g(t_i) = \varphi - t_i^n \quad (i = 0, 1, ..., n-1)$ using, for example, the algorithms from §4.6.3 in Knuth (1969).
5. Compute in n flops the entries of the matrix $D_{f,g} = \mathrm{diag}((f'(t_i) \cdot g(t_i))_{i=0}^{n-1})$.
6. Set $\xi = 1 + \max_{0 \leq i \leq n-1} t_i$. Compute in $O(n)$ flops the numbers ξ^i $(i = 0, 1, ..., n)$. Set $\varphi = \xi^n$ and $D_\varphi = \mathrm{diag}((\xi^i)_{i=0}^{n-1})$.
7. Compute in $O(n)$ flops the entries of the matrix D_φ^{-1}.
8. Compute the coordinates of the vector $\mathcal{F}D_\varphi\mathbf{y}$ with $\mathbf{y} = \mathbf{r} + \varphi\mathbf{e}_0$ in $\phi(n) + O(n)$ flops using (BM3), where $\mathbf{r} = (r_i)_{i=0}^{n-1}$.
9. Compute in $O(n)$ flops the inverse of the diagonal matrix $\Lambda_\varphi(\mathbf{r} + \varphi\mathbf{e}_0) = \mathrm{diag}(\mathcal{F}D_\varphi\mathbf{y})$.

II. Application to the vector.

Complexity : $\iota(n) + 2\phi(n) + O(n)$ flops.

Input : Vector $\mathbf{b} \in \mathbf{C}^n$.

Output : Vector $V(\mathbf{t})^T\mathbf{b} \in \mathbf{C}^n$.

1. Compute vector $V(\mathbf{t})^T\mathbf{b}$ by (2.8) using (BM2) and (BM3).Ξ

The product of transposed Vandermonde matrix with an arbitrary matrix can be computed using algorithm 2.1 in

$$\iota(n)n + 2\phi(n)n + O(n^2)$$

flops.

Note that representations of the matrix $V(\mathbf{t})^T$ in the form of the product of matrix $V(\mathbf{t})^{-1}$ by some matrices with lower complexity of multiplication with vectors, could be found in Heinig and Rost (1984) and Canny, Kaltofen and Yagati (1989). In Canny, Kaltofen and Yagati (1989), for example, the matrix $V(\mathbf{t})^T$ is represented in the form

$$V(\mathbf{t})^T = H \cdot V(\mathbf{t})^{-1}, \tag{2.9}$$

where matrix $H = V(\mathbf{t})^T \cdot V(\mathbf{t})$ turned out to be a Hankel matrix. For computing the entries of H it is proposed in Canny, Kaltofen and Yagati (1989) to interpolate the n-th degree polynomial from its zeros at n points, and then to solve an equation of the type $T\mathbf{x} = \mathbf{b}$ with Toeplitz matrix T from $\mathbf{C}^{2n\times 2n}$. Let $\tau(n)$ be as above the complexity of solving one Toeplitz equation of the form (1.2). Thus, computing the entries of the matrix H in (2.9) by the scheme in Canny, Kaltofen and Yagati (1989) and afterwards the preparation of the Hankel matrix H according to (BM5) costs $2\tau(2n) + \iota(n) + 2\phi(n) + O(n)$ flops. This complexity exceeds the complexity of preprocessing in the algorithm 2.1. Furthermore, the appearance of the factor circulant in (2.7) instead of the Hankel matrix in (2.9) is reflected in the additional economy in $2\phi(n)$ flops at the stage of application to the vector.

Formula (2.7) also enables the proposition of fast algorithm for computing the product of matrix $V(\mathbf{t})^{-T}$ by a vector. Moreover, the preprocessing phase consists of computing the parameters of the representation of $V(\mathbf{t})^{-T}$ in the form analogous to the representation (2.8) of $V(\mathbf{t})^T$ and costs

$$\iota(n) + \varepsilon(n) + \phi(n) + 2n\log n + O(n)$$

flops. Afterwards, the application to the vector costs

$$\varepsilon(n) + 2\phi(n) + O(n)$$

flops. The product of the matrix $V(\mathbf{t})^{-T}$ by an arbitrary matrix can be computed in

$$\varepsilon(n)n + 2\phi(n)n + O(n^2)$$

flops.

Formulas (2.3) and (2.6) yield the following representation for the inverse to a Vandermonde matrix :

$$V(\mathbf{t})^{-1} = \mathrm{Circ}_{\varphi}(\mathbf{r} + \varphi\mathbf{e}_0) \cdot J \cdot V(\mathbf{t})^T \cdot \mathrm{diag}(\Big(\frac{1}{f'(t_i)\cdot(\varphi - t_i^n)}\Big)_{i=0}^{n-1}).$$

Here, as above, J stands for the reverse identity matrix, $f(\lambda) = \prod_{i=0}^{n-1}(\lambda - t_i) = \lambda^n + \sum_{i=0}^{n-1} r_i\cdot\lambda^i$ and $\mathbf{r} = (r_i)_{i=0}^{n-1}$. The last formula can be used for computing the entries of the inverse of Vandermonde matrix in $O(n^2)$ flops.

3 Cauchy matrices and matrices connected with them

By definition, the Cauchy matrix $C(\mathbf{s},\mathbf{t})$ has the form

$$C(\mathbf{s},\mathbf{t}) = \Big(\frac{1}{s_i - t_j}\Big)_{i,j=0}^{n-1},$$

where $\mathbf{s} = (s_i)_{i=0}^{n-1}$ and $\mathbf{t} = (t_i)_{i=0}^{n-1}$ are given vectors from $\mathbf{C}^n$ such that t_i, s_i $(i = 0, 1, ..., n-1)$ are $2n$ different complex numbers. As in the previous section set

$$f(\lambda) = \prod_{i=0}^{n-1}(\lambda - t_i) = \lambda^n + \sum_{i=0}^{n-1} r_i \cdot \lambda^i, \qquad g(\lambda) = \varphi - \lambda^n,$$

where φ is such that $\varphi \neq t_i^n$ $(i = 0, 1, ..., n-1)$. Formula (2.2) yields

$$V(\mathbf{s}) \cdot B_{f,g} \cdot V(\mathbf{t})^T = \Big(\frac{f(s_i) \cdot g(t_j)}{s_i - t_j}\Big)_{i,j=0}^{n-1} = \mathrm{diag}((f(s_i))_{i=0}^{n-1}) \cdot C(\mathbf{s}, \mathbf{t}) \cdot \mathrm{diag}((g(t_i))_{i=0}^{n-1}).$$

From here follows

$$C(\mathbf{s}, \mathbf{t}) = \mathrm{diag}(\Big(\frac{1}{f(s_i)}\Big)_{i=0}^{n-1}) \cdot V(\mathbf{s}) \cdot B_{f,g} \cdot V(\mathbf{t})^T \cdot \mathrm{diag}(\Big(\frac{1}{g(t_i)}\Big)_{i=0}^{n-1}).$$

This equality and (2.5) yields the following representation of the Cauchy matrix :

$$C(\mathbf{s}, \mathbf{t}) = \mathrm{diag}(\Big(\frac{1}{f(s_i)}\Big)_{i=0}^{n-1}) \cdot V(\mathbf{s}) \cdot V(\mathbf{t})^{-1} \cdot \mathrm{diag}((f'(t_i))_{i=0}^{n-1}). \tag{3.1}$$

Formula (3.1) enables us to propose of the following algorithm.

Algorithm 3.1. Computing the product of the Cauchy matrix by a vector

I. Preprocessing.

Complexity : $\iota(n) + 2\varepsilon(n) + O(n)$ flops.

Input : Vectors $\mathbf{s} = (s_i)_{i=0}^{n-1}$ and $\mathbf{t} = (t_i)_{i=0}^{n-1}$ $\quad (t_i \neq t_j$ while $i \neq j)$.

Output : Parameters of the representation $C(\mathbf{s}, \mathbf{t})$ in the form (4.1).

1. Compute the coefficients of the polynomial $f(\lambda) = \prod_{i=0}^{n-1}(\lambda - t_i)$ in $\iota(n)$ flops.
2. Evaluate in $\varepsilon(n) + O(n)$ flops the values $\frac{1}{f(s_i)}$ $(i = 0, 1, ..., n-1)$.
3. Compute in $O(n)$ flops the coefficients of the polynomial $f'(\lambda)$.
4. Evaluate in $\varepsilon(n)$ flops the values $f'(t_i)$ $(i = 0, 1, ..., n-1)$.

II. Application to the vector.

Complexity : $\iota(n) + \varepsilon(n) + O(n)$ flops.

Input : Vector $\mathbf{b} \in \mathbf{C}^n$.

Output : Vector $C(\mathbf{s}, \mathbf{t})\mathbf{b} \in \mathbf{C}^n$.

1. Compute $C(\mathbf{s}, \mathbf{t})\mathbf{b}$ by (3.1) using (BM1) and (BM2).Ξ

In an earlier paper of Gerasoulis (1987) another algorithm for computing the product of Cauchy matrix with a vector was proposed. Moreover, this algorithm does not restrict itself to Cauchy matrices and is valid for more general classes of matrices. Gerasoulis algorithm applied for computing the product $C(\mathbf{s},\mathbf{t})\mathbf{b}$ requires $2\iota(n)+3\varepsilon(n)+O(n)$ flops of which $\iota(n)+2\varepsilon(n)+O(n)$ can be incorporated into a preprocessing stage.

Note, the product of $C(\mathbf{s},\mathbf{t})$ by an arbitrary matrix can be computed using algorithm 2.1 in

$$\iota(n)n+\varepsilon(n)n+O(n^2)$$

flops.

Formula (3.1) also enables the fast algorithms of multiplication with vectors for matrices $C(\mathbf{s},\mathbf{t})^T$, $C(\mathbf{s},\mathbf{t})^{-1}$ and $C(\mathbf{s},\mathbf{t})^{-T}$ to be elaborated. For example, formula (3.1) implies the following representation for the inverse of Cauchy matrix :

$$C(\mathbf{s},\mathbf{t})^{-1}=\mathrm{diag}(\Big(\frac{1}{f'(t_i)}\Big)_{i=0}^{n-1})\cdot V(\mathbf{t})\cdot V(\mathbf{s})^{-1}\cdot\mathrm{diag}((f(s_i))_{i=0}^{n-1}).$$

This formula yields that product of the inverse of Cauchy matrix by a vector can be computed in

$$\iota(n)+\varepsilon(n)+O(n)$$

flops after the same preprocessing as in algorithm 3.1. In fact the latter proposition can be formulated as follows. The system of linear equations with invertible Cauchy matrix can be solved in $O(n\log^2 n)$ flops.

4 Determinants of Vandermonde and Cauchy matrices

In the recent paper Pan (1990) an outline of some algorithms for computing up to a sign the determinants of Vandermonde and Cauchy matrices with real entries is suggested. For computing the Vandermonde determinant it is proposed therein first to compute using the algorithm from Canny, Kaltofen and Yagati (1989) the entries of the Hankel matrix H in (2.9). This step costs $2\tau(2n)+\iota(n)+O(n)$ flops. Then compute $\det H=(\det V(\mathbf{t}))^2$. This seems to be quite expensive. For the determinant of a Cauchy matrix in Pan (1990) a complicated procedure of reducing this problem to the analogous problem for some close-to-Toeplitz matrix is proposed. It turned out that well-known formulas lead to simple and significantly more effective algorithms for computing the determinants of Vandermonde and Cauchy matrices with complex entries. Indeed, as is well known,

$$\det V(\mathbf{t})=\prod_{0\le i<j\le n-1}(t_j-t_i). \tag{4.1}$$

The square of the expression in the right-hand side of (4.1) is, by definition, the discriminant $D(f)$ of the polynomial $f(\lambda)=\prod_{i=0}^{n-1}(\lambda-t_i)$. It is well known (see, for example, van der Warden (1971)) and can easily be seen, that $D(f)=(-1)^{\frac{n(n-1)}{2}}\prod_{i=0}^{n-1}f'(t_i)$. Thus,

$$\Big(\det V(\mathbf{t})\Big)^2=(-1)^{\frac{n(n-1)}{2}}\prod_{i=0}^{n-1}f'(t_i).$$

This formula enables the following algorithm to be proposed.

Algorithm 4.1. Computing the Vandermonde determinant.

Complexity: $\iota(n) + \varepsilon(n) + O(n)$ flops plus one extraction of a square root.

Input : Vector $\mathbf{t} = (t_i)_{i=0}^{n-1} \in \mathbf{C}^n \quad (t_i \neq t_j \text{ while } i \neq j)$.

Output : Value $\det V(\mathbf{t})$ up to a sign.

1. Compute in $\iota(n)$ flops the coefficients of the polynomial $f(\lambda) = \prod_{i=0}^{n-1}(\lambda - t_i)$.
2. Compute in $O(n)$ flops the coefficients of the polynomial $f'(\lambda)$.
3. Evaluate the values $f'(t_i)$ $(i = 0, 1, ..., n-1)$ in $\varepsilon(n)$ flops.
4. Compute $\det\left(V(\mathbf{t})\right)^2 = (-1)^{\frac{n(n-1)}{2}} \prod_{i=0}^{n-1} f'(t_i)$ in $O(n)$ flops.
5. Recover $\det V(\mathbf{t})$ up to a sign in one operation of the extraction of a square root.Ξ

Furthermore, if all the coordinates of the vector $\mathbf{t}$ are real numbers (this situation was considered in Pan (1990)), then the explicit expression in the right-hand side of (4.1) allows in $O(n \log n)$ time to come to a conclusion about the sign of $V(\mathbf{t})$. Indeed, in this case the sign of $V(\mathbf{t})$ is determined by the quantity of the pairs (t_i, t_j) satisfying the condition $t_i > t_j$ while $i < j$. Thus, the sign of $\det V(\mathbf{t})$ is determined by the number of inversions in the permutation $k = (k_0, k_1, ..., k_{n-1})$, where k_i is the position of the element t_i in the sequence, which is derived from $\{t_i\}_{i=0}^{n-1}$ by the rearrangement of all the elements in increasing order. To compute this number of inversions, we have to rearrange the two-component sequence $\{(t_i, i)\}_{i=0}^{n-1}$ in the increasing order of the first components and then extract the second components into the separate sequence $m = (m_0, m_1, ..., m_{n-1})$. It is easy to see that m is the inverse of the permutation k and consequently has the same number of inversions. Therefore, to determine the sign of $\det V(\mathbf{t})$ it remains to compute the number $In(m)$ of inversions in the permutation m. The algorithm based on these arguments is proposed below. Note that here we deviate from the rule accepted in the present paper, and estimate the complexity in terms time, and not in flops. The reason for this is that we use here the algorithms of sorting and of computing the number of inversions in a permutation. These two algorithms do not involve float point operations and involve cheaper operations of comparison and exchange.

Algorithm 4.2. Determination of the sign of Vandermonde determinant.

Complexity : $O(n \log n)$ time.

Input : Vector $\mathbf{t} = (t_i)_{i=0}^{n-1} \in \mathbf{R}^n \quad (t_i \neq t_j \text{ while } i \neq j)$.

Output : Sign of $\det V(\mathbf{t})$.

1. Sort in $O(n \log n)$ time the sequence of the pairs $\eta = \{(t_i, i)\}_{i=0}^{n-1}$ in the increasing order of the first components, using, for example, an algorithm from §5.2.6 in Knuth (1973).

2. For the permutation m, formed from the second components of the sorted sequence η, compute in $O(n \log n)$ time the number $In(m)$ of its inversions, using, for example, the algorithm from Knuth (1973), problem 5.1.1.6.
3. Set sign $V(\mathbf{t}) = 1$ if the number $In(m)$ is even and sign $V(\mathbf{t}) = -1$ in the opposite case.Ξ

Furthermore, for the determinant of Cauchy matrix the following expression is well-known :

$$\det C(\mathbf{s}, \mathbf{t}) = \frac{\prod_{0 \le i < j \le n-1}(s_j - s_i) \cdot \prod_{0 \le i < j \le n-1}(t_i - t_j)}{\prod_{0 \le i,j \le n-1}(s_i - t_j)}$$

(see, for example Polya and Szego (1972), where it is attributed to Cauchy (1891)). Using (4.1), the latter formula can be rewritten in the form :

$$\det C(\mathbf{s}, \mathbf{t}) = (-1)^{\frac{n(n-1)}{2}} \cdot \frac{\det V(\mathbf{s}) \cdot \det V(\mathbf{t})}{\prod_{i=0}^{n-1} f(s_i)}, \tag{4.2}$$

where as above $f(\lambda) = \prod_{i=0}^{n-1}(\lambda - t_i)$. Using this formula and the algorithm 4.1, one can compute the value $\det C(\mathbf{s}, \mathbf{t})$ up to a sign in

$$2\iota(n) + 3\varepsilon(n) + O(n)$$

flops plus one operation of the extraction of a square root.

Furthermore, in the case when s_i and t_i are real, the formula (4.2) and the algorithm 4.2 allow to determine the sign of the value $\det C(\mathbf{s}, \mathbf{t})$ in $O(n \log n)$ time.

References

1. Aho,A.V., Hopcroft,J.E. and Ullman,J.D. (1976), "The design and analysis of computer algorithms", Addison-Wesley, Reading, Mass.

2. Ammar,G. and Gader,P. (1990), New decompositions of the inverse of a Toeplitz matrix, *in* "Signal processing, Scattering and Operator Theory, and Numerical Methods, Proc. Int. Symp. MTNS-89, vol. III", (M.A.Kaashoek, J.H. van Schruppen and A.C.M.Ran. Eds), Birkhauser, Boston, pp. 421 – 428.

3. Brent,R., Gustavson,F. and Yun,D. (1980), Fast solutions of Toeplitz systems of equations and computation of Pade approximants, *Journal of Algorithms*, **1**, 259 – 295.

4. Canny,J.F., Kaltofen,E. and Yagati,L. (1989), Solving systems of non-linear equations faster, *in* "Proc. ACM-SIGSAM 1989 Internat. Symp. Symbolic Algebraic Comput.", ACM, New York, pp. 34 – 42.

5. Chun,J. and Kailath,T. (1991), Divide-and-conquer solutions of least-squares problems for matrices with displacement structure, *SIAM Journal of Matrix Analysis Appl.*, **12** (No. 1), 128 – 145.

6. Cline,R.E., Plemmons,R.J. and Worm,G. (1974), Generalized inverses of certain Toeplitz matrices, *Linear Algebra Appl.*, **8**, 25 – 33.

7. Gerasoulis,A. (1987), A fast algorithm for the multiplication of generalized Hilbert matrices with vectors, *Math. of Computation*, **50** (No. 181), 179 – 188.

8. Gohberg,I. and Feldman,I. (1974), "Convolution equations and projection methods for their solutions", Translations of Mathematical Monographs, **41**, Amer. Math. Soc.

9. Gohberg,I. and Olshevsky, V. (1992), Circulants, displacements and decompositions of matrices, *Integral Equations and Operator Theory*, **15** (No. 5), 730 – 743.

10. Gohberg,I. and Semencul,A. (1972), On the inversion of finite Toeplitz matrices and their continuous analogs (in Russian), *Matem. Issled.*, Kishinev, **7** (No. 2), 201 – 233.

11. Heinig,G., Rost, K. (1984), "Algebraic methods for Toeplitz-like matrices and operators", Operator Theory, vol. 13, Birkauser Verlag, Basel.

12. de Hoog,F. (1987), A new algorithm for solving Toeplitz system of equations, *Linear Algebra Appl.*, **88/89**, 122 – 138.

13. Knuth,D. (1969), "The art of computer programming", vol. **2**, "Seminumerical algorithms", Addison-Wesley, Reading, Mass.

14. Knuth,D. (1973), "The art of computer programming", vol. **3**, "Sorting and searching", Addison-Wesley, Reading, Mass.

15. Lander,F.I. (1974), The Bezoutian and the inversion of Hankel and Toeplitz matrices (in Russian), *Matem. Issled.*, Kishinev, **9** (No. 2), 69 – 87.

16. Pan,V. (1990), On computations with dense structured matrices, *Math. of Computation*, **55** (No. 191), 179 – 190.

17. Polya,G. and Szego, G. (1972), "Problems and theorems in analysis", Springer Verlag, Berlin, New York.

18. Van der Warden,B.L. (1971), "Algebra I", Springer Verlag, Berlin, New York.