Fast inversion of Chebyshev–Vandermonde matrices[1]

I.Gohberg and **V.Olshevsky**[2]

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.

Summary. This paper contains two fast algorithms for inversion of Chebyshev–Vandermonde matrices of the first and second kind. They are based on special representations of the Bezoutians of Chebyshev polynomials of both kinds. The paper also contains the results of numerical experiments which show that the algorithms proposed here are not only much faster, but also more stable than other algorithms available. It is also efficient to use the above two algorithms for solving Chebyshev–Vandermode systems of equations with preprocessing.

Mathematics Subject Classifications (1991) : 15A09, 15A23, 47B35, 65F05, 65T10, 65Y05, 68Q25.

0 Introduction

In this paper we consider polynomial Vandermonde matrix of the form

$$\mathrm{V}_P(\mathbf{t}) = \begin{bmatrix} P_0(t_0) & P_1(t_0) & \cdots & P_{n-1}(t_0) \\ P_0(t_1) & P_1(t_1) & \cdots & P_{n-1}(t_1) \\ \vdots & & & \vdots \\ P_0(t_{n-1}) & P_1(t_{n-1}) & \cdots & P_{n-1}(t_{n-1}) \end{bmatrix}, \qquad (\mathbf{t} = (t_i)_{i=0}^{n-1}),$$

where $t_0, t_1, ..., t_{n-1}$ are different complex numbers and system $\{P\} = \{P_0(\lambda), P_1(\lambda), ..., P_{n-1}(\lambda)\}$ forms the basis in the linear space $\mathbf{C}_{n-1}[\lambda]$ of all complex polynomials in λ whose degree does not exceed $n-1$. The concept of polynomial Vandermonde matrix is a generalization of the ordinary Vandermonde matrix, where $P_i(\lambda) = \lambda^i$. For ordinary Vandermonde matrices the number of fast algorithms is available for solving linear systems (Björck and Pereyra (1970), Tang and Golub (1981); Gohberg, Kailath, Koltracht and Lancaster (1987)) and for inversion (Traub (1966), Heinig and Rost (1984), Gohberg and Koltracht (1991). More general classes of polynomial Vandermonde matrices appears in different areas and were considered and analyzed in literature (see, for example, Karlin and Szegö (1960), Verde-Star

[1]In Numerische Mathematik, **67**, No. 1 (1994), 71 – 92.

[2]The second author is currently with Information Systems Laboratory, Stanford University, Stanford CA 94305 – 4055, U.S.A., e-mail : olshevsk@rascals.stanford.edu

(1988)). The analysis of the numerical properties and algorithms connected with polynomial Vandermonde matrices can be found in various sources. In Gautschi (1983) the condition number of polynomial Vandermonde matrix was estimated for various choices of the system $\{P\}$. In Higham (1988), Higham (1990), Reichel and Opfer (1991) some algorithms for solving linear systems with polynomial Vandermonde matrices were deduced and analyzed.

In the present paper the use of the Bezoutian allows to deduce the following equality which connects two polynomial Vandermonde matrices corresponding to two systems of polynomials $\{P\}$ and $\{Q\}$:

$$\mathrm{V}_P(\mathbf{t})^{-1} = B \cdot \mathrm{V}_Q(\mathbf{t})^\tau \cdot \mathrm{diag}((\frac{1}{(\prod_{\substack{i=0 \\ i\neq k}}^{n-1}(t_k - t_i))})_{i=0}^{n-1}), \tag{0.1}$$

where b_{ij} are determined from the equality

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

with $F(\lambda) = \prod_{k=0}^{n-1}(\lambda - t_k)$, and

$$B = (b_{ij})_{i,j=0}^{n-1}.$$

Numerical features of the algorithm for computing $\mathrm{V}_P(\mathbf{t})^{-1}$ based on equality (0.1) depend essentially on the complexity of the matrix B. We consider in this paper the case where $\{P\}$ and $\{Q\}$ are systems of Chebyshev polynomials. In the latter case it turns out that the matrix B is a Hankel (or almost a Hankel) matrix. This leads to the fact that the algorithm for inversion of Chebyshev–Vandermonde matrices based on the equality (0.1) has complexity $7n^2$ operations. The proposed algorithms compare favorably in both time and error accumulation with algorithms from Higham (1988),Reichel and Opfer (1991) and also with Gaussian elimination.

The paper consists of two sections. The first contains the deduction of the formulae used later for the design of the algorithms. The second section contains the detailed description of the algorithms and presents the results of numerical experiments.

After the submission for publication of the present paper the authors received a preprint of Calvetti and Reichel (1993), which contains another algorithm for inversion of polynomial Vandermonde matrices. This algorithm is a generalization of the scheme for ordinary Vandermonde matrices, suggested earlier in Traub (1966). The Calvetti and Reichel algorithm does not restrict itself to Chebyshev-Vandermonde case and is valid for more general classes of polynomial Vandermonde matrices. At the same time it is a little slower and requires $12n^2 + O(n)$ arithmetic operations.

1 Formulae for inverses of Chebyshev–Vandermonde matrices

1.1 Notations, definitions and relevant facts

Matrices. The algebra of all $n \times n$ matrices with complex entries will be denoted by $\mathbf{C}^{n\times n}$.

Transpose. Transpose of any matrix $A \in \mathbf{C}^{n\times n}$ will be denoted by A^{τ}.

Polynomials. The ring of all polynomials in λ over $\mathbf{C}$ whose degrees do not exceed $n \in \mathbf{N}$ will be denoted by $\mathbf{C}_n[\lambda]$.

Chebyshev polynomials. Let

$$\{T\} = \{T_k(\lambda\}_{k=0}^{n-1}, \quad \{U\} = \{U_k(\lambda\}_{k=0}^{n-1}$$

stand for the families of Chebyshev polynomials

$$T_k(\lambda) = \cos(k \cdot \arccos(\lambda)), \quad U_k(\lambda) = \frac{\sin((k+1)\cdot \arccos(\lambda))}{\sin(\arccos(\lambda))},$$

of the first and of the second kind, respectively.

Recurrence relations for Chebyshev polynomials. As it is well known, Chebyshev polynomials can also be defined by the following recurrence relations :

$$T_0(\lambda) = 1, \qquad T_1(\lambda) = \lambda, \qquad 2\lambda \cdot T_{n-1}(\lambda) = T_n(\lambda) + T_{n-2}(\lambda); \tag{1.1}$$

$$U_0(\lambda) = 1, \qquad U_1(\lambda) = 2\lambda, \qquad 2\lambda \cdot U_{n-1}(\lambda) = U_n(\lambda) + U_{n-2}(\lambda); \tag{1.2}$$

Connections between Chebyshev polynomials. It is well known and immediately follows from (1.1) and (1.2) that

$$T_n(\lambda) = \frac{1}{2}(U_n(\lambda) - U_{n-2}(\lambda)). \tag{1.3}$$

$$U_n(\lambda) = 2 \cdot \sum_{j=0}^{[\frac{n}{2}]} T_{n-2j}(\lambda) - \frac{(-1)^n + 1}{2}. \tag{1.4}$$

Associated vectors. For each vector $\mathbf{t} = (t_k)_{k=0}^{n-1} \in \mathbf{C}^n$ will be associated two vectors

$$\mathbf{a}(\mathbf{t}) = (a_{k+1}(\mathbf{t}))_{k=0}^{n-1} \quad \text{and} \quad \mathbf{b}(\mathbf{t}) = (b_{k+1}(\mathbf{t}))_{k=0}^{n-1} \tag{1.5}$$

whose coordinates are determined from the following equalities :

$$\prod_{k=0}^{n-1}(\lambda - t_k) = \sum_{k=0}^{n} a_k(\mathbf{t}) \cdot T_k(\lambda) = \sum_{k=0}^{n} b_k(\mathbf{t}) \cdot U_k(\lambda).$$

Bezoutian. Let $F(\lambda)$, $G(\lambda)$ be two polynomials from $\mathbf{C}_n[\lambda]$. The bilinear form

$$\mathcal{B}_{(F,G)}(\lambda,\mu)=\frac{F(\lambda)\cdot G(\mu)-F(\mu)\cdot G(\lambda)}{\lambda-\mu}=\sum_{i,j=0}^{n-1}q_{ij}\cdot\lambda^i\cdot\mu^j \tag{1.6}$$

is referred to as the *Bezoution* of $F(\lambda)$, $G(\lambda)$.

Polynomial Vandermonde matrix. Let $\mathbf{t}=(t_k)_{k=0}^{n-1}\in\mathbf{C}^n$ and some system of polynomials $\{P\}=\{P_0(\lambda),P_1(\lambda),...,P_{n-1}(\lambda)\}$ is given. The matrix

$$\mathrm{V}_P(\mathbf{t})=\begin{bmatrix} P_0(t_0) & P_1(t_0) & \cdots & P_{n-1}(t_0)\\ P_0(t_1) & P_1(t_1) & \cdots & P_{n-1}(t_1)\\ \vdots & & & \vdots\\ P_0(t_{n-1}) & P_1(t_{n-1}) & \cdots & P_{n-1}(t_{n-1})\end{bmatrix}$$

will be referred to as *polynomial Vandermonde matrix*, corresponding to the system $\{P\}$ and to vector $\mathbf{t}$.

Chebyshev–Vandermonde matrices. Matrices $\mathrm{V}_T(\mathbf{t})$ and $\mathrm{V}_U(\mathbf{t})$ will be referred to as Chebyshev–Vandermonde matrices of the first and of the second kind, respectively.

Hankel matrix. For some vector $\mathbf{c}=(c_k)_{k=0}^{n-1}\in\mathbf{C}^n$ by $\mathrm{H}(\mathbf{c})$ will be denoted the following triangular Hankel matrix

$$\mathrm{H}(\mathbf{c})=\begin{bmatrix} c_0 & c_1 & \cdots & c_{n-2} & c_{n-1}\\ c_1 & & \iddots & \iddots & 0\\ \vdots & c_{n-2} & \iddots & \iddots & \vdots\\ c_{n-2} & c_{n-1} & \iddots & & \vdots\\ c_{n-1} & 0 & \cdots & \cdots & 0\end{bmatrix}.$$

Greatest integer in n is denoted by $[n]$.

1.2 Main results

Theorem 2.1 *Let* $\mathbf{t}=(t_k)_{k=0}^{n-1}$ *and* $t_0,t_1,...,t_{n-1}$ *be* n *different complex numbers. Then the following formulae hold :*

$$\mathrm{V}_T(\mathbf{t})^{-1}=4\cdot D\cdot\mathrm{H}(\mathbf{d}(\mathbf{t}))\cdot D\cdot\mathrm{V}_T(\mathbf{t})^{\tau}\cdot\mathrm{diag}(\mathbf{c}(\mathbf{t})), \tag{2.1}$$

$$\mathrm{V}_U(\mathbf{t})^{-1}=\mathrm{H}(\mathbf{e}(\mathbf{t}))\cdot\mathrm{V}_U(\mathbf{t})^{\tau}\cdot\mathrm{diag}(\mathbf{c}(\mathbf{t})), \tag{2.2}$$

$$\mathrm{V}_T(\mathbf{t})^{-1}=2\cdot D\cdot\mathrm{H}(\mathbf{a}(\mathbf{t}))\cdot\mathrm{V}_U(\mathbf{t})^{\tau}\cdot\mathrm{diag}(\mathbf{c}(\mathbf{t})), \tag{2.3}$$

$$\mathrm{V}_U(\mathbf{t})^{-1}=2\cdot\mathrm{H}(\mathbf{a}(\mathbf{t}))\cdot D\cdot\mathrm{V}_T(\mathbf{t})^{\tau}\cdot\mathrm{diag}(\mathbf{c}(\mathbf{t})), \tag{2.4}$$

where

$D = \mathrm{diag}(\frac{1}{2}, 1, 1, ..., 1)$, *vector* $\mathbf{a}(\mathbf{t})$ *is as in* (1.5) *and coordinates of vectors* $\mathbf{c}(\mathbf{t}) = (c_i(\mathbf{t}))_{i=0}^{n-1}$, $\mathbf{d}(\mathbf{t}) = (d_i(\mathbf{t}))_{i=0}^{n-1}$, $\mathbf{e}(\mathbf{t}) = (e_i(\mathbf{t}))_{i=0}^{n-1}$ *are defined as follows :*

$$c_k(\mathbf{t}) = \frac{1}{\prod_{\substack{i=0 \\ i\neq k}}^{n-1}(t_k - t_i)}, \quad d_k(\mathbf{t}) = \sum_{i=0}^{[\frac{n-k-1}{2}]} a_{k+1+2i}(\mathbf{t}) \quad (k = 0, 1, ..., n-1),$$

$$e_{n-1}(\mathbf{t}) = a_n(\mathbf{t}), \quad e_{n-2}(\mathbf{t}) = a_{n-1}(\mathbf{t}), \quad e_k(\mathbf{t}) = a_{k+1}(\mathbf{t}) - a_{k+3}(\mathbf{t}) \quad (k = 0, 1, ..., n-3).$$

Theorem 2.2 *Let* $\mathbf{t} = (t_k)_{k=0}^{n-1}$ *and* $t_0, t_1, ..., t_{n-1}$ *be* n *different complex numbers. Then the following formulae hold :*

$$\mathrm{V}_T(\mathbf{t})^{-1} = 8 \cdot D \cdot \mathrm{H}(\mathbf{f}(\mathbf{t})) \cdot D \cdot \mathrm{V}_T(\mathbf{t})^{\tau} \cdot \mathrm{diag}(\mathbf{c}(\mathbf{t})), \tag{2.5}$$

$$\mathrm{V}_U(\mathbf{t})^{-1} = 2 \cdot \mathrm{H}(\mathbf{b}(\mathbf{t})) \cdot \mathrm{V}_U(\mathbf{t})^{\tau} \cdot \mathrm{diag}(\mathbf{c}(\mathbf{t})), \tag{2.6}$$

$$\mathrm{V}_T(\mathbf{t})^{-1} = 4 \cdot D \cdot \mathrm{H}(\mathbf{g}(\mathbf{t})) \cdot \mathrm{V}_U(\mathbf{t})^{\tau} \cdot \mathrm{diag}(\mathbf{c}(\mathbf{t})), \tag{2.7}$$

$$\mathrm{V}_U(\mathbf{t})^{-1} = 4 \cdot \mathrm{H}(\mathbf{g}(\mathbf{t})) \cdot D \cdot \mathrm{V}_T(\mathbf{t})^{\tau} \cdot \mathrm{diag}(\mathbf{c}(\mathbf{t})), \tag{2.8}$$

where $\mathbf{b}(\mathbf{t})$ *is from* (1.5)*; matrix* $D \in \mathbf{C}^{n\times n}$ *and vector* $\mathbf{c}(\mathbf{t}) \in \mathbf{C}^n$ *are the same as in the theorem* 2.1. *The coordinates of the vectors* $\mathbf{f}(\mathbf{t}) = (f_i(\mathbf{t}))_{i=0}^{n-1}$ *and* $\mathbf{g}(\mathbf{t}) = (g_i(\mathbf{t}))_{i=0}^{n-1}$ *are defined by*

$$f_k(\mathbf{t}) = \sum_{i=0}^{[\frac{n-k-1}{2}]} (i+1) \cdot b_{k+1+2i}(\mathbf{t}); \quad g_k(\mathbf{t}) = \sum_{i=0}^{[\frac{n-k-1}{2}]} b_{k+1+2i}(\mathbf{t}) \quad (k = 0, 1, ..., n-1).$$

The proof of both above theorems is based on interrelations between Chebyshev polynomials which are given in the next theorem.

Theorem 2.3 *For* $n \geq 1$ *the following formulae hold :*

$$\frac{T_n(\lambda) - T_n(\mu)}{\lambda - \mu} = 4 \cdot \sum_{k=0}^{[\frac{n-3}{2}]} \sum_{j=1}^{n-2-2k} T_{n-1-2k-j}(\lambda) \cdot T_j(\mu) +$$

$$2 \cdot \sum_{k=0}^{[\frac{n-2}{2}]} T_{n-1-2k}(\lambda) + 2 \cdot \sum_{k=0}^{[\frac{n-2}{2}]} T_{n-1-2k}(\mu) + \frac{1 + (-1)^{n-1}}{2}; \tag{2.9}$$

$$\frac{T_n(\lambda) - T_n(\mu)}{\lambda - \mu} = \sum_{j=0}^{n-1} U_{n-1-j}(\lambda) \cdot U_j(\mu) - \sum_{j=0}^{n-3} U_{n-3-j}(\lambda) \cdot U_j(\mu); \tag{2.10}$$

$$\frac{T_n(\lambda)-T_n(\mu)}{\lambda-\mu}=2\cdot\sum_{j=0}^{n-2}T_{n-1-j}(\lambda)\cdot U_j(\mu)+T_0(\lambda)\cdot U_{n-1}(\mu); \tag{2.11}$$

$$\frac{U_n(\lambda)-U_n(\mu)}{\lambda-\mu}=8\cdot\sum_{k=0}^{[\frac{n-3}{2}]}(k+1)\cdot\sum_{j=1}^{n-2-2k}T_{n-1-2k-j}(\lambda)\cdot T_j(\mu)+$$

$$4\cdot\sum_{k=0}^{[\frac{n-2}{2}]}(k+1)\cdot T_{n-1-2k}(\lambda)+4\cdot\sum_{k=0}^{[\frac{n-2}{2}]}(k+1)\cdot T_{n-1-2k}(\mu)+\frac{1+(-1)^{n-1}}{2}\cdot(n+1); \tag{2.12}$$

$$\frac{U_n(\lambda)-U_n(\mu)}{\lambda-\mu}=2\cdot\sum_{j=0}^{n-1}U_{n-1-j}(\lambda)\cdot U_j(\mu); \tag{2.13}$$

$$\frac{U_n(\lambda)-U_n(\mu)}{\lambda-\mu}=4\cdot\sum_{k=0}^{[\frac{n-1}{2}]}\sum_{j=0}^{n-2-2k}T_{n-1-2k-j}(\lambda)\cdot U_j(\mu)+2\cdot\sum_{k=0}^{[\frac{n-1}{2}]}T_0(\lambda)\cdot U_{n-1-2k}(\mu). \tag{2.14}$$

Theorems 2.1 – 2.2 and theorem 2.3 will be proved in the next two subsections.

1.3 Proof of theorems 2.1 and 2.2

Let $F(\lambda)$ and $G(\lambda)$ be any two polynomials from $\mathbf{C}_{n-1}[\lambda]$, and

$$\{P\}=\{P_i(\lambda)\}_{i=0}^{n-1},\qquad \{Q\}=\{Q_i(\lambda)\}_{i=0}^{n-1}$$

be two arbitrary systems of polynomials, each of them forms the basis in the linear space $\mathbf{C}_{n-1}[\lambda]$. Let us represent the Bezoutian (1.6) of $F(\lambda)$ and $G(\lambda)$ in the following form

$$\mathcal{B}_{(F,G)}(\lambda,\mu)=\frac{F(\lambda)\cdot G(\mu)-F(\mu)\cdot G(\lambda)}{\lambda-\mu}=\sum_{i,j=0}^{n-1}b_{ij}\cdot P_i(\lambda)\cdot Q_j(\mu). \tag{3.1}$$

The equation (3.1) can be rewritten in a matrix form as follows

$$\mathcal{B}_{(F,G)}(\lambda,\mu)=\begin{bmatrix}P_0(\lambda) & P_1(\lambda) & P_2(\lambda) & \cdots & P_{n-1}(\lambda)\end{bmatrix}\cdot B_{(\{P\},\{Q\},F,G)}\cdot\begin{bmatrix}Q_0(\mu)\\ Q_1(\mu)\\ Q_2(\mu)\\ \vdots\\ Q_{n-1}(\mu)\end{bmatrix}, \tag{3.2}$$

where the matrix

$$B_{(\{P\},\{Q\},F,G)} = (b_{ij})_{i,j=0}^{n-1}$$

composed from the coefficients of $\mathcal{B}_{(F,G)}(\lambda,\mu)$ will be referred to as a *Bezout matrix* of the polynomials $F(\lambda)$ and $G(\lambda)$ with respect to the bases $\{P\}$ and $\{Q\}$ in $\mathbf{C}_{n-1}[\lambda]$. This concept generalized the canonical notion of Bezout matrix, where $P_i(\lambda) = Q_i(\lambda) = \lambda^i$.

From (3.2) it follows that for arbitrary $\mathbf{s} = (s_i)_{i=0}^{n-1}, \quad \mathbf{t} = (t_i)_{i=0}^{n-1} \in \mathbf{C}^n$ the following formula holds:

$$\mathrm{V}_P(\mathbf{s}) \cdot B_{(\{P\},\{Q\},F,G)} \cdot \mathrm{V}_Q(\mathbf{t})^\tau = (\mathcal{B}_{f,g}(s_i,t_j))_{i,j=0}^{n-1}, \tag{3.3}$$

In the case where the numbers $t_0, t_1, ..., t_{n-1}$ are n roots of the polynomial $F(\lambda)$ equality (3.3) and obvious identity

$$\mathcal{B}_{(F,G)}(\lambda,\lambda) = F'(\lambda) \cdot G(\lambda) - F(\lambda) \cdot G'(\lambda)$$

imply

$$\mathrm{V}_P(\mathbf{t}) \cdot B_{(\{P\},\{Q\},F,G)} \cdot \mathrm{V}_Q(\mathbf{t})^\tau = \mathrm{diag}((F'(t_i) \cdot G(t_i))_{i=0}^{n-1}). \tag{3.4}$$

The latter formula is a generalization of a result due to Lander (1974), where it appeared for ordinary Vandermonde matrices. Now suppose that $t_0, t_1, ..., t_{n-1}$ are n different complex numbers and set

$$F(\lambda) = \prod_{k=0}^{n-1}(\lambda - t_k), \qquad G(\lambda) = 1. \tag{3.5}$$

In this case (3.4) gives the following representation of the inverse polynomial Vandermonde matrix :

$$\mathrm{V}_P(\mathbf{t})^{-1} = B_{(\{P\},\{Q\},F,G)} \cdot \mathrm{V}_Q(\mathbf{t})^\tau \cdot \mathrm{diag}((\frac{1}{(\prod_{\substack{i=0 \\ i\neq k}}^{n-1}(t_k - t_i))})_{i=0}^{n-1}). \tag{3.6}$$

The numerical features for computing $\mathrm{V}_P(\mathbf{t})^{-1}$ by the formula (3.6) essentially depend on the complexity of the matrix $B_{(\{P\},\{Q\},F,G)}$. In case $P_i(\lambda) = Q_i(\lambda) = \lambda^i$ this matrix turned out to be a Hankel (cf. Gohberg and Olshevsky (1993)). In general $B_{(\{P\},\{Q\},F,G)}$ does not possess the Hankel structure, for example, even for Legendre, Hermite and Laguerre systems of polynomials $\{P\}$ and $\{Q\}$. However, the situation is different for the families of Chebyshev polynomials of the first and of the second kind. In the latter case matrix $B_{(\{P\},\{Q\},F,G)}$ turned out to be Hankel or almost Hankel. For example, from the formulae (2.9) – (2.14) it is easy to see that if $\{P\}$ and $\{Q\}$ are chosen to be systems of Chebyshev polynomials $\{T_i(\lambda)\}_{i=0}^{n-1}$ or $\{U_i(\lambda)\}_{i=0}^{n-1}$ then the Bezout matrix of polynomial $T_n(\lambda)$ (or $U_n(\lambda)$) and $G(\lambda) = 1$ has a Hankel or almost Hankel structure. Let us illustrate that fact and write down these matrices for $n = 5$. From (2.9) – (2.14) we deduce that

$$B_{(\{T\},\{T\},T_5(\lambda),1)} = D \cdot \begin{bmatrix} 4 & 0 & 4 & 0 & 4 \\ 0 & 4 & 0 & 4 & 0 \\ 4 & 0 & 4 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 \end{bmatrix} \cdot D; \qquad B_{\{U\},\{U\},T_5(\lambda),1)} = \begin{bmatrix} 0 & 0 & -1 & 0 & 1 \\ 0 & -1 & 0 & 1 & 0 \\ -1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix};$$

$$B_{(\{T\},\{U\},T_5(\lambda),1)} = D\cdot\begin{bmatrix} 0 & 0 & 0 & 0 & 2\\ 0 & 0 & 0 & 2 & 0\\ 0 & 0 & 2 & 0 & 0\\ 0 & 2 & 0 & 0 & 0\\ 2 & 0 & 0 & 0 & 0 \end{bmatrix};\qquad B_{\{T\},\{T\},U_5(\lambda),1)} = D\cdot\begin{bmatrix} 24 & 0 & 16 & 0 & 8\\ 0 & 16 & 0 & 8 & 0\\ 16 & 0 & 8 & 0 & 0\\ 0 & 8 & 0 & 0 & 0\\ 8 & 0 & 0 & 0 & 0 \end{bmatrix}\cdot D;$$

$$B_{(\{U\}\{U\},U_5(\lambda),1)} = \begin{bmatrix} 0 & 0 & 0 & 0 & 2\\ 0 & 0 & 0 & 2 & 0\\ 0 & 0 & 2 & 0 & 0\\ 0 & 2 & 0 & 0 & 0\\ 2 & 0 & 0 & 0 & 0 \end{bmatrix};\qquad B_{(\{T\},\{U\},U_5(\lambda),1)} = D\cdot\begin{bmatrix} 4 & 0 & 4 & 0 & 4\\ 0 & 4 & 0 & 4 & 0\\ 4 & 0 & 4 & 0 & 0\\ 0 & 4 & 0 & 0 & 0\\ 4 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Here

$$D = \mathrm{diag}(\frac{1}{2}, 1, 1, ..., 1).$$

According to the formula (3.6) to obtain the representations (2.1) – (2.4) of the inverse Chebyshev-Vandermonde matrices it remains to compute in each case the Bezoutian (3.1) for the polynomials $F(\lambda)$ and $G(\lambda)$ from (3.5). We have

$$\mathcal{B}_{(F,G)}(\lambda,\mu) = \frac{F(\lambda)-F(\mu)}{\lambda-\mu} = \sum_{k=1}^{n} a_i(\mathbf{t})\frac{T_i(\lambda)-T_i(\mu)}{\lambda-\mu}, \tag{3.7}$$

where $\mathbf{a}(\mathbf{t})$ defined as in (1.5).

The latter equality and the explicit expressions for $\frac{T_i(\lambda)-T_i(\mu)}{\lambda-\mu}$ computed in the theorem 2.3 imply desired formulae (2.1) – (2.4). Indeed, in order to obtain the equality (2.3) one has to substitute the expression (2.11) in (3.7). Then we have

$$\mathcal{B}_{(F,G)}(\lambda,\mu) = 2\cdot\sum_{i=1}^{n}\sum_{j=0}^{i-2} a_i(\mathbf{t})\cdot T_{i-1-j}(\lambda)\cdot U_j(\mu) + \sum_{i=1}^{n} a_i(\mathbf{t})\cdot T_0(\lambda)\cdot U_{i-1}(\lambda) =$$

$$\begin{bmatrix} T_0(\lambda) & T_1(\lambda) & \cdots & T_{n-1}(\lambda) \end{bmatrix}\cdot\begin{bmatrix} a_1(\mathbf{t}) & a_2(\mathbf{t}) & \cdots & a_{n-1}(\mathbf{t}) & a_n(\mathbf{t})\\ 2\cdot a_2(\mathbf{t}) & & 2\cdot a_{n-1}(\mathbf{t}) & 2\cdot a_n(\mathbf{t}) & 0\\ \vdots & \iddots & \iddots & \iddots & \vdots\\ 2\cdot a_{n-1}(\mathbf{t}) & \iddots & \iddots & & \vdots\\ 2\cdot a_n(\mathbf{t}) & 0 & \cdots & \cdots & 0 \end{bmatrix}\cdot\begin{bmatrix} U_0(\lambda)\\ U_1(\lambda)\\ \vdots\\ U_{n-1} \end{bmatrix}.$$

Hence $B_{(\{T\},\{U\},F(\lambda),G(\lambda))} = D\cdot \mathrm{H}(\mathbf{a}(\mathbf{t}))$ and formula (2.3) follows from (3.6).

Analogously, let us substitute the expression (2.10) in (3.7). Then

$$\mathcal{B}_{(F,G)}(\lambda,\mu) = \sum_{i=1}^{n} a_i(\mathbf{t})\cdot(\sum_{j=0}^{i-1} U_{i-1-j}(\lambda)\cdot U_j(\mu) - \sum_{j=0}^{i-3} U_{i-3-j}(\lambda)\cdot U_j(\lambda)) =$$

$$a_n(\mathbf{t})\cdot\sum_{j=0}^{n-1} U_{n-1-j}(\lambda)\cdot U_j(\mu) + a_{n-1}(\mathbf{t})\cdot\sum_{j=0}^{n-2} U_{n-2-j}(\lambda)\cdot U_j(\mu) +$$

$$\sum_{i=1}^{n-3}\sum_{j=0}^{i-1}(a_i(\mathbf{t})-a_{i+2}(\mathbf{t}))\cdot U_{i-1-j}(\lambda)\cdot U_j(\mu)=$$

$$=\begin{bmatrix} T_0(\lambda) & T_1(\lambda) & \cdots & T_{n-1}(\lambda)\end{bmatrix}\cdot\begin{bmatrix} e_0(\mathbf{t}) & e_1(\mathbf{t}) & \cdots & e_{n-2}(\mathbf{t}) & e_{n-1}(\mathbf{t})\\ e_1(\mathbf{t}) & & \iddots & \iddots & 0\\ \vdots & e_{n-2}(\mathbf{t}) & \iddots & \iddots & \vdots\\ e_{n-2}(\mathbf{t}) & e_{n-1}(\mathbf{t}) & \iddots & & \vdots\\ e_{n-1}(\mathbf{t}) & 0 & \cdots & \cdots & 0\end{bmatrix}\cdot\begin{bmatrix} U_0(\lambda)\\ U_1(\lambda)\\ \vdots\\ U_{n-1}\end{bmatrix},$$

where $e_k(\mathbf{t})$ are defined as in the formulation of theorem 2.1. Hence $B_{(\{U\},\{U\},F(\lambda),G(\lambda))}=\mathrm{H}(\mathbf{e}(\mathbf{t}))$ which implies (2.2).

Analogously, formula (2.1) can be deduced from (3.6), (3.7) and (2.9). Formula (2.4) is an immediate consequence of (2.3). Thus theorem 2.1 is now proved.

The assertion of theorem 2.2 follows from the equality (3.6) with completely similar arguments. Here Bezoutian of $F(\lambda)$ and $G(\lambda)$ is represented as

$$\mathcal{B}_{(F,G)}(\lambda,\mu)=\frac{F(\lambda)-F(\mu)}{\lambda-\mu}=\sum_{k=0}^{n}b_i(\mathbf{t})\frac{U_i(\lambda)-U_i(\mu)}{\lambda-\mu},\tag{3.8}$$

where $\mathbf{b}(\mathbf{t})$ is defined in (1.5). The latter equality and (2.12) imply formula (2.5). Formula (2.6) follows from (3.8) and (2.13), and on the other hand, formula (2.7) follows from (3.8) and (2.14). Finally let us remark, that formula (2.8) is a straightforward consequence of (2.7).□

1.4 Proof of theorem 2.3

The proof of the formulae (2.9) – (2.14) is mainly based on recurrence relations (1.1) and (1.2).

First we will prove the formula (2.9). To this end let us show that the following equality holds :

$$\frac{T_n(\lambda)-T_n(\mu)}{\lambda-\mu}-\frac{T_{n-2}(\lambda)-T_{n-2}(\mu)}{\lambda-\mu}=$$

$$4\cdot\sum_{j=1}^{n-2}T_{n-1-j}(\lambda)\cdot T_j(\mu)+2\cdot T_{n-1}(\lambda)+2\cdot T_{n-1}(\mu).\tag{4.1}$$

Indeed, let A stand for the expression in the right-hand side of (4.1), then

$$(\lambda-\mu)\cdot A=$$

$$4\cdot(\lambda-\mu)\cdot\sum_{j=1}^{n-2}T_{n-1-j}(\lambda)\cdot T_j(\mu)+2\cdot(\lambda-\mu)\cdot T_{n-1}(\lambda)+2\cdot(\lambda-\mu)\cdot T_{n-1}(\mu)=$$

$$4\cdot\sum_{j=1}^{n-2}\lambda\cdot T_{n-1-j}(\lambda)\cdot T_j(\mu)+2\lambda\cdot T_{n-1}(\lambda)+2\lambda\cdot T_{n-1}(\mu)-$$

$$4 \cdot \sum_{j=1}^{n-2} \mu \cdot T_{n-1-j}(\lambda) \cdot T_j(\mu) - 2\mu \cdot T_{n-1}(\mu) - 2\mu \cdot T_{n-1}(\lambda).$$

Then, using recurrence relations (1.1), we have

$$(\lambda - \mu) \cdot A =$$

$$2 \cdot \sum_{j=1}^{n-2} (T_{n-j}(\lambda) + T_{n-2-j}(\lambda)) \cdot T_j(\mu) + T_n(\lambda) + T_{n-2}(\lambda) + 2\lambda \cdot T_{n-1}(\mu) -$$

$$2 \cdot \sum_{j=1}^{n-2} T_{n-1-j}(\lambda) \cdot (T_{j+1}(\mu) + T_{j-1}(\mu)) - T_n(\mu) - T_{n-2}(\mu) - 2\mu \cdot T_{n-1}(\lambda) =$$

$$2 \cdot (\sum_{j=1}^{n-2} T_{n-j}(\lambda) \cdot T_j(\mu) - \sum_{j=1}^{n-2} T_{n-1-j}(\lambda) \cdot T_{j+1}(\mu)) +$$

$$2 \cdot (\sum_{j=1}^{n-2} T_{n-2-j}(\lambda) \cdot T_j(\mu) - \sum_{j=1}^{n-2} T_{n-1-j}(\lambda) \cdot T_{j-1}(\mu)) +$$

$$T_n(\lambda) - T_n(\mu) + T_{n-2}(\lambda) - T_{n-2}(\mu) + 2\lambda \cdot T_{n-1}(\mu) - 2\mu \cdot T_{n-1}(\lambda) =$$

$$2 \cdot (T_{n-1}(\lambda) \cdot T_1(\mu) - T_1(\lambda) \cdot T_{n-1}(\mu)) - 2 \cdot (T_{n-2}(\lambda) - T_{n-2}(\mu)) +$$

$$T_n(\lambda) - T_n(\mu) + T_{n-2}(\lambda) - T_{n-2}(\mu) + 2\lambda \cdot T_{n-1}(\mu) - 2\mu \cdot T_{n-1}(\lambda) =$$

$$(T_n(\lambda) - T_n(\mu)) - (T_{n-2}(\lambda) - T_{n-2}(\mu)),$$

and (4.1) follows. It is straightforward to check that formula (2.9) is obviously valid for $n = 1$. For arbitrary odd n this formula follows from equality (4.1) by induction. Analogously, direct computation shows that formula (2.9) holds for $n = 2$. For arbitrary even n this formula is deduced from equality (4.1) using induction.

Now let us turn to the proof of formula (2.11). Denoting by B the expression in its right-hand side, we have

$$(\lambda - \mu) \cdot B =$$

$$2 \cdot (\lambda - \mu) \cdot \sum_{j=0}^{n-2} T_{n-1-j}(\lambda) \cdot U_j(\mu) + (\lambda - \mu) \cdot T_0(\lambda) \cdot U_{n-1}(\mu)) =$$

$$2\lambda \cdot T_{n-1}(\lambda) + \lambda \cdot U_{n-1}(\mu) - 2\mu \cdot T_{n-1}(\lambda) - \mu \cdot U_{n-1}(\mu) +$$

$$2 \cdot \sum_{j=1}^{n-2} \lambda \cdot T_{n-1-j}(\lambda) \cdot U_j(\mu) - 2 \cdot \sum_{j=1}^{n-2} \mu \cdot T_{n-1-j}(\lambda) \cdot U_j(\mu).$$

Then, using recurrence relations (1.1), we have

$$(\lambda - \mu) \cdot B =$$

$$2\lambda \cdot T_{n-1}(\lambda) + \lambda \cdot U_{n-1}(\mu) - 2\mu \cdot T_{n-1}(\lambda) - \mu \cdot U_{n-1}(\mu) +$$

$$\sum_{j=1}^{n-2}(T_{n-j}(\lambda)+T_{n-2-j}(\lambda))\cdot U_j(\mu)-\sum_{j=1}^{n-2}T_{n-1-j}(\lambda)\cdot(U_{j+1}(\mu)+U_{j-1}(\mu))=$$

$$2\lambda\cdot T_{n-1}(\lambda)+\lambda\cdot U_{n-1}(\mu)-2\mu\cdot T_{n-1}(\lambda)-\mu\cdot U_{n-1}(\mu)+$$

$$\sum_{j=1}^{n-2}T_{n-j}(\lambda)\cdot U_j(\mu)-\sum_{j=1}^{n-2}T_{n-1-j}(\lambda)\cdot U_{j+1}(\mu)+$$

$$\sum_{j=1}^{n-2}T_{n-2-j}(\lambda)\cdot U_j(\mu)-\sum_{j=1}^{n-2}T_{n-1-j}(\lambda)\cdot U_{j-1}(\mu)=$$

$$2\lambda\cdot T_{n-1}(\lambda)+\lambda\cdot U_{n-1}(\mu)-2\mu\cdot T_{n-1}(\lambda)-\mu\cdot U_{n-1}(\mu)+$$

$$T_{n-1}(\lambda)\cdot U_1(\mu)-T_1(\lambda)\cdot U_{n-1}(\mu)+T_0(\lambda)\cdot U_{n-2}(\mu)-T_{n-2}(\lambda)\cdot U_0(\mu)=$$

$$2\lambda\cdot T_{n-1}(\lambda)+\lambda\cdot U_{n-1}(\mu)-2\mu\cdot T_{n-1}(\lambda)-\mu\cdot U_{n-1}(\mu)+$$

$$2\mu\cdot T_{n-1}(\lambda)-\lambda\cdot U_{n-1}(\mu)+U_{n-2}(\mu)-T_{n-2}(\lambda)=$$

$$2\lambda\cdot T_{n-1}(\lambda)-T_{n-2}(\lambda)-\mu\cdot U_{n-1}(\mu)-U_{n-2}(\mu).$$

From here, again using (1.1) and afterwards using (1.3) we have

$$(\lambda-\mu)\cdot B=$$

$$T_n(\lambda)-\frac{1}{2}\cdot(U_n(\mu)-U_{n-2}(\mu))=$$

$$T_n(\lambda)-T_n(\mu)$$

This proves (2.11).

Now let us turn to the proof of (2.13). Similarly, denoting by C the expression in the right-hand side of (2.13), we have

$$(\lambda-\mu)\cdot C=$$

$$2(\lambda-\mu)\cdot\sum_{j=0}^{n-1}U_{n-1-j}(\lambda)\cdot U_j(\mu)=$$

$$2\lambda\cdot\sum_{j=0}^{n-2}U_{n-1-j}(\lambda)\cdot U_j(\mu)+2\lambda\cdot U_0(\lambda)\cdot U_{n-1}(\mu)-$$

$$2\mu\cdot\sum_{j=1}^{n-1}U_{n-1-j}(\lambda)\cdot U_j(\mu)-2\mu\cdot U_{n-1}(\lambda)\cdot U_0(\mu).$$

From here and from (1.2) it follows that

$$(\lambda-\mu)\cdot C=$$

$$\sum_{j=0}^{n-2}(U_{n-j}(\lambda)+U_{n-2-j}(\lambda))\cdot U_j(\mu)+2\lambda\cdot U_{n-1}(\mu)-$$

$$\sum_{j=1}^{n-1} U_{n-1-j}(\lambda)\cdot(U_{j+1}(\mu)+U_{j-1}(\mu))+2\mu\cdot U_{n-1}(\lambda)=$$

$$(\sum_{j=0}^{n-2} U_{n-j}(\lambda)\cdot U_j(\mu)-\sum_{j=1}^{n-1} U_{n-1-j}(\lambda)\cdot U_{j+1}(\mu))+$$

$$(\sum_{j=0}^{n-2} U_{n-2-j}(\lambda)\cdot U_j(\mu)-\sum_{j=1}^{n-1} U_{n-1-j}(\lambda)\cdot U_{j-1}(\mu))+$$

$$2\lambda\cdot U_{n-1}(\mu)-2\mu\cdot U_{n-1}(\lambda)=$$

$$U_n(\lambda)+U_{n-1}(\lambda)\cdot U_1(\mu)-U_1(\lambda)\cdot U_{n-1}(\mu)-U_n(\mu)+2\lambda\cdot U_{n-1}(\mu)-2\mu\cdot U_{n-1}(\lambda)=$$

$$U_n(\lambda)-U_n(\mu).$$

This proves (2.13).

Now let us prove formulae (2.10), (2.12) and (2.14). Formula (2.10) is a straightforward consequence of (2.13) and (1.3). Formula (2.12) follows from (2.9) and (1.4). Analogously, (2.11) and (1.5) imply formula (2.14).□

2 Algorithms

Below we describe in detail the algorithms derived from the formulae proved in the first part of this paper. While all these formulae are valid for complex numbers, below only a real case is considered.

2.1 Computing the inverses of Chebyshev–Vandermonde matrices

As it is well known, the standard method of matrix inversion and solving linear systems of equations, Gaussian elimination, requires $O(n^3)$ arithmetic operations. In Higham (1988), Reichel and Opfer (1991) it was shown, that using a special structure of matrix $\mathrm{V}_T(\mathbf{t})$ one can solve the corresponding system of linear equations in $4n^2$ arithmetic operations. Here and henceforth we neglect the lower terms in the estimates for the complexity of computations. The main result of the present subsection is that all n^2 elements of the matrix $\mathrm{V}_T(\mathbf{t})^{-1}$ can be computed in $7n^2$ arithmetic operations. The analogous assertion is also valid for computing the elements of the matrix $\mathrm{V}_U(\mathbf{t})^{-1}$. These estimates follow from the algorithms based on the formulae (2.3) and (2.2) and given at the end of this subsection. Let us turn to the derivation of these algorithms.

Let $\mathbf{t}=(t_k)_{k=0}^{n-1}$ and $t_0,t_1,...,t_{n-1}$ be n different real numbers. Computing the elements of $\mathrm{V}_T(\mathbf{t})^{-1}$ by the formula (2.3) can be divided into the following steps.

Step 1. Computing the numbers $c_k(\mathbf{t})=\frac{1}{\prod_{\substack{i=0\\ i\neq k}}^{n-1}(t_k-t_i)}$.

This task can be solved in $1.5n^2$ operations.

Step 2. Computing the numbers $a_1(\mathbf{t}), a_2(\mathbf{t}), ..., a_n(\mathbf{t})$.

Set

$$F_0(\lambda) = 1; \qquad F_k(\lambda) = \prod_{i=0}^{k-1}(\lambda - t_i) = \sum_{i=0}^{n} a_i^{(k)} \cdot T_i(\lambda) \qquad (k = 1, 2, ..., n).$$

Then using recurrence relations, we have

$$F_{k+1}(\lambda) = (\lambda - t_k) \cdot F_k(\lambda) = \sum_{i=0}^{k-1} a_i^{(k)} \cdot \lambda \cdot T_i(\lambda) - \sum_{i=0}^{k-1} t_k \cdot a_i^{(k)} \cdot T_i(\lambda) =$$

$$\frac{1}{2} \cdot (2 \cdot a_0^{(k)} \cdot T_1(\lambda) + \sum_{i=1}^{k-1} a_i^{(k)} \cdot T_{i-1}(\lambda) + \sum_{i=1}^{k-1} a_i^{(k)} \cdot T_{i+1}(\lambda) - \sum_{i=0}^{k-1} (2 \cdot t_k) \cdot T_i(\lambda)).$$

Therefore, the numbers $a_i(\mathbf{t})$ $(= a_i^{(n)})$ can be computed by the following scheme :

1. $$\begin{bmatrix} a_0^{(0)} \\ a_1^{(0)} \\ a_2^{(0)} \\ \vdots \\ a_n^{(0)} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

2. $$\begin{bmatrix} a_0^{(k+1)} \\ a_1^{(k+1)} \\ a_2^{(k+1)} \\ \vdots \\ a_n^{(k+1)} \end{bmatrix} = \frac{1}{2} \cdot \left(\begin{bmatrix} 0 \\ 2 \cdot a_0^{(k)} \\ a_1^{(k)} \\ \vdots \\ a_{n-1}^{(k)} \end{bmatrix} + \begin{bmatrix} a_1^{(k)} \\ a_2^{(k)} \\ \vdots \\ a_n^{(k)} \\ 0 \end{bmatrix} - (2 \cdot t_k) \cdot \begin{bmatrix} a_0^{(k)} \\ a_1^{(k)} \\ a_2^{(k)} \\ \vdots \\ a_n^{(k)} \end{bmatrix} \right).$$

Taking into account that only first $k+1$ from the numbers $a_0(\mathbf{t})^{(k)}, a_1(\mathbf{t})^{(k)}, ..., a_n(\mathbf{t})^{(k)}$ can differ from zero, and that multiplications and divisions by 2 can be dropped out, the above scheme can be implemented in $1.5n^2$ operations.

Step 3. Computing the matrix product $R = (r_{ij})_{i,j=0}^{n-1} = \mathrm{H}(\mathbf{a}(\mathbf{t})) \cdot \mathrm{V}_U(\mathbf{t})^\tau$.

This task is reduced to n matrix times vector multiplications

$$\begin{bmatrix} r_{0,k} \\ r_{1,k} \\ r_{2,k} \\ \vdots \\ r_{n-1,k} \end{bmatrix} = \begin{bmatrix} a_1(\mathbf{t}) & a_2(\mathbf{t}) & \cdots & a_{n-1}(\mathbf{t}) & a_n(\mathbf{t}) \\ a_2(\mathbf{t}) & & \cdot\cdot & \cdot\cdot & 0 \\ \vdots & a_{n-1}(\mathbf{t}) & \cdot\cdot & \cdot\cdot & \vdots \\ a_{n-1}(\mathbf{t}) & a_n(\mathbf{t}) & \cdot\cdot & & \vdots \\ a_n(\mathbf{t}) & 0 & \cdots & \cdots & 0 \end{bmatrix} \cdot \begin{bmatrix} U_0(t_k) \\ U_1(t_k) \\ U_2(t_k) \\ \vdots \\ U_{n-1}(t_k) \end{bmatrix}$$

$(k = 0, 1, ..., n-1)$. Obviously, $r_{n-1,k} = a_n(\mathbf{t})$, $r_{n-2,k} = a_{n-1}(\mathbf{t}) + a_n(\mathbf{t}) \cdot U_1(t_k)$, and recurrence relations (1.2) imply that

$$r_{n-j,k} = a_{n-j+1}(\mathbf{t}) + (2 \cdot t_k) \cdot r_{n-j+1,k} - r_{n-j+2,k} \qquad (j = 3, 4, ..., n-1).$$

Thus, elements of R can be computed in $3n^2$ operations.

Step 4. Computing the output matrix $\mathrm{V}_T(\mathbf{t})^{-1}$.
Here the matrix R from the previous step multiplied from the right by diagonal matrix $\mathrm{diag}(2\mathbf{c}(\mathbf{t}))$ and the first row of the result is halved. It requires n^2 arithmetic operations.□

Below this scheme is implemented in MATLAB language, especially expressive when it comes to matrix and vector manipulations. The reader may consult MATLAB User's Guide (1991).

Algorithm 1.1 Inversion of Chebyshev–Vandermonde matrix of the first kind.
Input: $\mathbf{t} = (t_i)_{i=0}^{n-1} \in \mathbf{R}^n$.
Output: The inverse G of $\mathrm{V}_T(\mathbf{t})$.
Complexity: $7n^2$ flops.
Basis: Formula (2.3)

```
for i = 1:(n–1)                                                    Step 1.
    for k = (i+1):n
        G(i,k) = t(i) – t(k);
    end
end
sign = 1;
c = 2∗ones(1,n);
for i = 1:n
    for k = 1:(i–1)
        c(i) = c(i)/G(k,i);
    end
    for k = (i+1):n
        c(i) = c(i)/G(i,k);
    end
    c(i) = sign∗c(i);
    sign = –sign;
end
t_d = 2∗t;                                                         Step 2.
a = [1   0   0];
for i = 1:n
    a = [0   2∗a(1)   a(2:(i+1))   0] + [a(2:(i+2)   0   0] – t_d(i)∗[a   0];
end
a = a/(2^n);
for i = 1:n                                                        Step 3.
```

```
    G(n,i) = a(n+1);
    G(n–1,i) = a(n) + t_d(i)*a(n+1);
    for k = (n–2):–1:1
        G(k,i) = a(k+1) + t_d(i)*G(k+1,i) – G(k+2,i);
    end
end
G = G*diag(c);                                                    Step 4.
G(1,:) = G(1,:)/2;
return;
```

With completely analogous arguments, formula (2.2) leads to the algorithm for computing the inverse of Chebyshev–Vandermonde matrix of the second kind. The derivation of this algorithm differ from the derivation of algorithm 1.1 only in nonessential details. Here is its MATLAB version.

Algorithm 1.2 Inversion of Chebyshev–Vandermonde matrix of the second kind.
Input: $\mathbf{t} = (t_i)_{i=0}^{n-1} \in \mathbf{R}^n$.
Output: The inverse G of $V_U(\mathbf{t})$.
Complexity: $7n^2$ flops.
Basis: Formula (2.2)

```
for i = 1:(n–1)
    for k = (i+1):n
        G(i,k) = t(i) – t(k);
    end
end
sign = 1;
c = ones(1,n);
for i = 1:n
    for k = 1:(i–1)
        c(i) = c(i)/G(k,i);
    end
    for k = (i+1):n
        c(i) = c(i)/G(i,k);
    end
    c(i) = sign*c(i);
    sign = –sign;
end
t_d = 2*t;
a = [1   0   0];
```

```
for i = 1:n
    a = [0   2*a(1)   a(2:(i+1))   0] + [a(2:(i+2))   0   0] - t_d(i)*[a   0];
end
a = a/(2^n);
for i = 1:(n–2)
    e(i) = a(i+1) – a(i+3);
end
e(n) = a(n+1);
e(n–1) = a(n);
for i = 1:n
    G(n,i) = e(n);
    G(n–1,i) = e(n–1) + t_d(i)*e(n);
    for k = (n–2):–1:1
        G(k,i) = e(k) + t_d(i)*G(k+1,i) – G(k+2,i);
    end
end
G = G*diag(c);
return;
```

Both algorithms use the place of the output matrix G for the storage of intermediate quantities and therefore require only $3n$ additional locations of memory.

It is easy to see, that algorithms 1.1 and 1.2 admit parallel realization. Indeed, each of steps $1 - 4$ can be implemented in the form of two nested loops. The computations in each inner loop are independent of each other and therefore can be performed in parallel. The thorough operation count proves the following result.

Theorem 1.3 *The inverse of each Chebyshev–Vandermonde matrices* $\mathrm{V}_T(\mathbf{t})$, $\mathrm{V}_U(\mathbf{t}) \in \mathbf{C}^{n\times n}$ *can be computed in* $8.5n$ *parallel arithmetic operations using* n *processors.*

2.2 Numerical experiments

We have carried out a wide variety of computer experiments to test the algorithms presented in subsection 2.1 and to compare them with other algorithms available. All the tests were performed using Borland C++ on a IBM PC AT compatible machine. Borland C++ uses IEEE–standard single and double precision arithmetic for which unit roundoffs are $u_{sp} = 2^{-23} \approx 1.19 \times 10^{-7}$ and $u_{dp} = 2^{-52} \approx 2.22 \times 10^{-16}$, respectively. For each choice of nodes $t_0, t_1, ..., t_{n-1}$ specified below, we computed the matrix $\mathrm{V}_T(\mathbf{t})^{-1}$ using the following algorithms :

(1) G–O Algorithm 1.1 of the present paper.
(2) NH Algorithm 1 from Higham (1988).
(3) R–O Algorithm 1 from Reichel and Opfer (1991).
(4) Gauss Gaussian elimination with full pivoting. C routine from Press et.al. (1988).

All the algorithms listed above were implemented in single and double precision versions. For each choice of the nodes $\mathbf{t} = (t_k)_{k=0}^{n-1}$ we computed the inverse of $V_T(\mathbf{t})$ by each algorithm from those listed above. The inverse matrices computed by single and double precision versions are denoted by $A^{(sp)} = (a_{ij}^{(sp)})_{ij=0}^{n-1}$ and $A^{(dp)} = (a_{ij}^{(dp)})_{ij=0}^{n-1}$, respectively. In each example matrices $A^{(dp)}$, computed by double precision versions of all four algorithms, turned out very close to each other. This fact led to consider the double precision answers as exact. In each example we measured the infinite–norm and componentwise relative errors :

$$\mathrm{ERR} = \frac{\|A^{(dp)} - A^{(sp)}\|_\infty}{\|A^{(dp)}\|_\infty}, \qquad \mathrm{COMP\_ERR} = \max_{0 \le i,j \le n-1} \frac{|a_{ij}^{(dp)} - a_{ij}^{(sp)}|}{|a_{ij}^{(dp)}|}.$$

It is known (see Higham (1990), Reichel and Opfer (1991)) that accuracy of the algorithms NH and R–O often improves when reordering of the nodes is used. With respect to the problem of matrix inversion reordering is equivalent to the interchanging of the rows of the matrix and of the columns of the inverse matrix. Therefore it has no influence on solving the matrix inversion problem. In Higham (1990) the following order of nodes was considered :

$$|t_0 - t_1| = \max_{0 \le j,k \le n-1} |t_j - t_k|,$$

$$\prod_{j=0}^{k-1} |t_k - t_j| = \max_{k \le s \le n-1} \prod_{j=0}^{k-1} |t_s - t_j| \qquad (2 \le k \le n-1).$$

Moreover in Higham (1990) it was shown that such reordering can be performed in n^2 arithmetic operations and $\frac{1}{2}n^2$ comparisons. This will be referred to as algorithm REORDER. It turned out that REORDER improves to even greater extent the numerical behavior of the algorithm G–O.

In the following table the data about time paid by each algorithm mentioned are given. The time required for the reordering of the given nodes using algorithm REORDER is not included in these counts and is given in a separate column. The last column of table 1 contains the time needed for computing one matrix times vector product. It should be noted, that the time listed for the algorithms NH and R–O pays for the solving of one system of linear equations, or for finding one column of the inverse matrix, while G–O and Gauss compute the whole inverse of Chebyshev–Vandermonde matrix.

n	G–O (inversion)	NH (one system)	R–O (one system)	Gauss (inversion)	REORDER	M×V product
10	–	0.05	–	0.05	–	–
15	0.11	0.05	–	0.55	0.05	–
20	0.16	0.11	0.05	1.21	0.05	–
25	0.22	0.16	0.11	2.31	0.05	0.05
30	0.33	0.22	0.22	3.90	0.11	0.11
35	0.49	0.27	0.27	6.15	0.11	0.16
40	0.60	0.38	0.38	9.07	0.16	0.16
45	0.82	0.49	0.44	12.91	0.22	0.22
50	0.99	0.60	0.60	17.58	0.27	0.22
55	1.21	0.77	0.66	23.30	0.33	0.33
60	1.43	0.88	0.88	30.16	0.38	0.33

Table 1. Time (seconds).

The data on errors accompanied each algorithm from the four considered are given below. In all examples the nodes are sorted using REORDER.

Clustered nodes on (–1, 1) :

$$t_i = -1 + 2(\frac{i}{n-1})^2, \qquad (i = 0, 1, ..., n-1).$$

	——————-COMP_ERR——————-				——————ERR——————			
n	G–O	NH	R–O	Gauss	G–O	NH	R–O	Gauss
5	1.1e–07	1.5e–06	1.1e–06	5.3e–07	4.4e–08	7.8e–08	6.5e–08	8.1e–08
10	1.8e–06	4.3e–03	4.3e–03	9.1e–04	4.4e–07	7.5e–07	5.6e–07	1.9e–05
15	1.9e–06	1.2e+00	1.2e+00	7.4e–01	5.2e–07	8.9e–07	8.0e–07	9.0e–03
20	4.8e–04	3.2e+02	3.2e+02	6.0e+01	1.1e–06	5.0e–06	4.9e–06	1.5e+00
30	9.7e–06	1.9e+09	1.9e+09	2.9e+05	2.7e–06	6.6e–06	6.6e–06	1.0e+00
40	2.7e–05	1.9e+09	1.9e+09	2.1e+04	3.8e–06	9.6e–06	9.9e–06	1.0e+00
50	2.6e–05	2.5e+10	2.5e+10	4.4e+04	4.3e–06	2.3e–05	2.2e–05	1.0e+00

Table 2. Clustered nodes on (–1,1).

Equidistant nodes on (–1, 1) :

$$t_i = -1 + 2 \cdot \frac{i}{n-1}, \qquad (i = 0, 1, ..., n-1).$$

	COMP_ERR				ERR			
n	G–O	NH	R–O	Gauss	G–O	NH	R–O	Gauss
5	3.0e–08	6.0e–07	1.2e–07	1.2e–07	3.0e–08	2.2e–07	3.7e–08	3.7e–08
10	5.7e–06	3.2e–05	6.5e–06	9.2e–06	3.1e–07	4.8e–07	2.3e–07	1.4e–07
15	4.0e–06	4.1e–05	6.7e–06	1.4e–04	8.2e–07	1.6e–06	8.5e–07	1.7e–06
20	9.5e–06	4.6e–04	4.6e–04	2.7e–03	3.7e–07	1.9e–06	5.0e–07	3.7e–05
30	8.6e–05	7.2e–01	7.2e–01	1.8e+00	5.8e–07	2.6e–06	1.0e–06	1.9e–02
40	9.8e–04	2.6e+01	2.6e+01	1.8e+03	2.7e–06	4.0e–06	3.6e–06	1.6e+00
50	9.0e–04	2.5e+04	2.5e+04	1.5e+04	1.7e–06	4.7e–06	2.7e–06	1.0e+00

Table 3. Equidistant nodes on (–1,1).

	COMP_ERR				ERR			
n	G–O	NH	R–O	Gauss	G–O	NH	R–O	Gauss
5	3.5e–06	5.3e–06	1.9e–06	5.7e–07	1.0e–07	9.4e–08	7.5e–08	8.4e–08
10	9.5e–07	7.7e–06	7.7e–06	1.8e–05	1.4e–07	3.2e–07	3.1e–07	9.2e–07
15	6.1e–07	3.3e–05	3.3e–05	4.1e–01	1.2e–07	5.3e–07	5.4e–07	1.2e–01
20	1.5e–05	4.5e–02	4.5e–02	1.0e+02	3.5e–07	3.6e–07	5.2e–07	9.8e–01
30	3.7e–05	7.7e+01	7.7e+01	4.9e+01	2.3e–07	4.2e–07	4.6e–07	1.0e+00
40	9.7e–05	1.4e+06	1.4e+06	2.3e+03	2.6e–07	1.1e–06	1.1e–06	1.0e+00
50	1.8e–06	1.1e+05	1.1e+05	7.1e+04	6.7e–07	2.5e–06	2.3e–06	1.0e+00
55	9.0e–06	1.3e+07	1.3e+07	5.3e+04	3.5e–07	2.0e–06	2.1e–06	1.0e+00

Table 4. Random nodes on (–1,1). Typical picture.

	COMP_ERR				ERR			
n	G–O	NH	R–O	Gauss	G–O	NH	R–O	Gauss
5	4.6e–06	2.6e–05	1.2e–05	1.3e–05	2.3e–07	2.0e–07	1.2e–07	1.0e–05
10	5.6e–05	3.2e–01	3.2e–01	5.8e–01	3.6e–07	8.6e–07	8.7e–07	2.2e–01
15	1.6e–03	5.8e+02	5.8e+02	1.1e+02	1.0e–06	1.4e–06	1.4e–06	9.6e–01
20	4.3e–03	5.8e+02	5.8e+02	3.3e+02	1.0e–06	2.2e–06	1.4e–06	1.1e+00
30	1.3e–03	5.7e+03	5.7e+03	5.2e+03	2.4e–06	5.2e–06	5.3e–06	1.1e+00
40	1.3e–03	1.0e+10	1.0e+10	3.5e+04	2.9e–06	7.1e–06	7.0e–06	1.2e+00
50	7.0e–03	1.1e+10	1.1e+10	5.6e+05	1.9e–05	2.0e–05	2.9e–05	1.0e+00

Table 5. Random nodes on (–1,1). The worst cases over 10 runs.

	——————-COMP_ERR——————-				——————ERR——————			
n	G–O	NH	R–O	Gauss	G–O	NH	R–O	Gauss
5	4.0e–07	7.0e–07	4.6e–07	4.6e–07	1.6e–07	2.4e–07	1.9e–07	1.7e–07
10	4.5e–06	7.8e–06	4.2e–06	3.4e–06	5.8e–07	6.7e–07	6.1e–07	6.1e–07
15	6.4e–06	9.8e–06	8.0e–06	4.5e–06	1.0e–06	1.4e–06	1.2e–06	9.1e–07
20	3.5e–05	5.2e–05	3.6e–05	3.3e–05	1.6e–06	2.5e–06	1.8e–06	1.6e–06
30	4.2e–05	1.7e–04	1.8e–04	4.7e–05	2.7e–06	5.2e–06	4.5e–06	3.8e–06
40	2.8e–04	2.6e–04	3.0e–04	2.8e–04	4.1e–06	8.4e–06	5.3e–06	4.5e–06
50	6.8e–04	9.9e–04	1.0e–03	5.4e–04	8.5e–06	1.2e–05	1.1e–05	6.7e–06
55	2.2e–04	5.0e–04	4.8e–04	2.6e–04	3.1e–06	1.2e–05	9.1e–06	4.3e–06

Table 6. Zeros of $T_n(\lambda)$.

2.3 Conclusions

Table 1 shows that time paid for the computing of the whole inverse Chebyshev–Vandermonde matrix by the algorithm G–O does not exceed the time of computing two columns of the above matrix using algorithm NH or R–O, and also is significantly smaller in comparison with Gauss. Simultaneously, the error accumulation accompanying the algorithm G–O is favorable with respect to other algorithms considered. From the tables 2 – 6 it is seen, that algorithm G–O has stable numerical behavior. For the case when $t_0, t_1, ..., t_{n-1}$ are zeros of Chebyshev polynomial $T_n(\lambda)$ (table 6), it is known that corresponding Chebyshev–Vandermonde matrix is well–conditioned (see Gautschi (1983)). In this case algorithm G–O produces approximately the same error as Gauss (and slightly lower than NH and R–O), while it works significantly faster. In other examples considered algorithm Gauss failed. In all of these cases (tables 2 – 5) the data show that algorithm G–O works and is accompanied by smaller infinite–norm relative error, in comparison with the algorithms NH and R–O. At the same time the data on componentwise relative errors collected in the tables 2 – 5 show that G–O is the only reliable algorithm among the algorithms compared.

2.4 Solving Chebyshev–Vandermonde systems

The following problem is considered below. Let matrix $A \in \mathbf{R}^{n\times n}$ be given. The task is to solve as fast as possible the systems of linear equations

$$Ax = f_i \qquad (i = 0, 1, 2, ...)$$

for the sequence of input vectors $f_0, f_1, f_2, ...$ from $\mathbf{R}^n$. In accordance with the accepted scheme, all the computations which are not dependent upon the input vectors $f_0, f_1, ...$ are singled out in a separate preprocessing stage. Thus, the algorithms for the solution of the above problem are divided into the following two stages :

I. Preprocessing of matrix $A \in \mathbf{R}^n$.

II. Processing of vector $f \in \mathbf{R}^n$.

The aim of the first stage is to reach more effective realization of the second stage. Algorithms presented in the subsection 2.1 enable to implement this scheme for Chebyshev–Vandermonde matrices of the first and second kinds as well as for their transposes. Indeed, in the latter case the preprocessing stage consists of computing the inverse matrix A^{-1}. Using algorithms 1.1 and 1.2 it can be done by the cost of $7n^2$ operations. Then the second stage is reduced to computing the matrix times vector product $A^{-1}f$, which requires only $2n^2$ operations. Thus, when the number of input vectors exceeds two, then use of the algorithms 1.1 and 1.2 became more efficient rather than NH or R–O.

2.5 Numerical experiments with systems

The two–staged scheme described in the previous subsection was implemented for the solution of Chebyshev–Vandermonde system

$$\mathrm{V}_T(\mathbf{t})x = f \tag{5.1}$$

of the first kind. Firstly the inverse matrix $\mathrm{V}_T(\mathbf{t})^{-1}$ was computed. The time required for this preprocessing stage one can find in the second column of table 1. The last column of this table gives information on the time performance of the second stage, where the matrix times vector product $\mathrm{V}_T(\mathbf{t})^{-1}f$ was computed. The analysis of the data from table 1 supports the theoretical conclusion that algorithm G–O became preferable in comparison with NH and R–O when one has to solve more than two linear systems with the same Chebyshev–Vandermonde matrix.

We also compared the accuracy in the computed solutions of (5.1), determined by G–O, NH, R–O and Gauss. Again, let $\mathbf{x}^{(sp)} = (x_i^{(sp)})_{i=0}^{n-1}$ and $\mathbf{x}^{(dp)} = (x_i^{(dp)})_{i=0}^{n-1}$ stand for the solution computed in single and double precision arithmetic and

$$\mathrm{err} = \frac{\|\mathbf{x}^{(dp)} - \mathbf{x}^{(sp)}\|_\infty}{\|\mathbf{x}^{(dp)}\|_\infty}, \qquad \mathrm{comp\_err} = \max_{0 \le i \le n-1} \frac{|x_i^{(dp)} - x_i^{(sp)}|}{|x^{(dp)}|} \tag{5.2}$$

denote the infinite norm and componentwise relative errors in the computed solution. For all four algorithms discussed we have computed solutions of a large number of systems (5.1) with various choices of nodes $t_0, t_1, ..., t_{n-1}$ and right-hand sides f. In each case we determine the relative errors (5.2) accompanying each algorithm from four mentioned. The conclusion which follows from the results of numerical experiments is that for reasonable values of n algorithm G–O computes the solution of (5.1) accurately. For the solution of Chebyshev–Vandermonde systems with preprocessing G–O should be preferred algorithm of those considered. Here are some of the computed examples.

n	comp_err				err			
	G–O	NH	R–O	Gauss	G–O	NH	R–O	Gauss
5	1.1e-16	0.0e+00	0.0e+00	0.0e+00	5.6e-17	0.0e+00	0.0e+00	0.0e+00
10	2.3e–07	3.0e–07	2.3e–07	2.3e–06	2.0e–07	7.4e–07	1.6e–07	5.3e–07
15	4.9e–06	6.6e–05	3.5e–06	4.5e–06	3.6e–07	1.9e–06	8.9e–07	1.0e–06
20	4.6e–05	7.9e–05	4.3e–05	1.9e–03	7.7e–07	2.6e–06	3.0e–06	1.2e–04
30	6.3e–06	2.3e–05	1.2e–05	2.3e–02	1.3e–06	4.9e–06	9.2e–07	6.7e–03
40	1.6e–04	7.5e–04	1.8e–04	1.9e+00	2.9e–06	1.3e–05	7.4e–06	2.2e+00
50	7.0e–05	1.2e–04	5.3e–05	1.0e+00	1.6e–06	7.3e–06	1.2e–06	3.0e+04

Table 7. Equidistant nodes on (–1,1). Right hand side : $\mathbf{f} = ((-1)^i)_{i=0}^{n-1}$.

n	comp_err				err			
	G–O	NH	R–O	Gauss	G–O	NH	R–O	Gauss
5	5.9e–08	9.0e–07	1.1e–07	1.2e–07	3.2e–08	5.5e–07	1.1e–07	7.8e–08
10	1.2e–06	6.6e–06	1.4e–06	2.3e–06	1.2e–07	1.2e–06	1.3e–07	6.6e–07
15	2.5e–06	2.5e–05	4.7e–06	2.1e–06	5.7e–07	1.4e–06	5.2e–07	8.3e–07
20	1.0e–05	2.1e–04	8.6e–06	4.7e–04	4.5e–07	3.1e–06	6.8e–07	3.1e–05
30	1.5e–05	1.3e–04	2.3e–05	5.3e–02	8.8e–07	3.4e–06	1.9e–06	1.1e–02
40	4.1e–04	8.2e–04	4.4e–04	2.8e+00	1.8e–06	6.9e–06	1.8e–06	2.5e+00
50	2.9e–04	4.0e–04	2.9e–04	1.0e+00	1.5e–06	8.8e–06	1.3e–06	1.0e+04

Table 8. Equidistant nodes on (–1,1). Coordinates of the right-hand side $\mathbf{f}$: random integers on (–10,10).

n	comp_err				err			
	G–O	NH	R–O	Gauss	G–O	NH	R–O	Gauss
5	2.4e–07	6.6e–07	1.8e–07	1.7e–07	1.6e–07	5.6e–07	1.7e–07	1.6e–07
10	1.6e–06	2.0e–06	2.3e–06	8.9e–07	2.2e–07	2.5e–07	1.9e–07	1.2e–07
15	3.0e–06	1.4e–05	1.4e–05	4.9e–06	3.1e–07	5.8e–07	4.0e–07	8.9e–08
20	1.5e–05	4.3e–05	3.9e–05	3.2e–06	7.0e–07	2.7e–06	1.9e–06	2.3e–07
30	1.1e–04	1.3e–04	5.0e–05	2.8e–06	1.7e–06	8.2e–06	8.1e–06	2.9e–07
40	3.1e–05	2.3e–04	2.5e–04	1.2e–05	7.6e–07	3.9e–06	4.3e–06	2.5e–07
50	8.0e–05	2.9e–03	2.9e–03	8.6e–05	1.2e–06	2.3e–05	2.1e–05	3.8e–07

Table 9. Nodes – extrema of $T_n(\lambda)$. Coordinates of the right-hand side $\mathbf{f}$: random integers on (–10,10).

We also determined the residial infinite–norm and componentwise errors

$$\text{res} = \frac{\||V_T(\mathbf{t}) \cdot \mathbf{x}^{(sp)} - \mathbf{f}\|_\infty}{\|\mathbf{f}\|_\infty}, \qquad \text{comp\_res} = \max_{0 \le i \le n-1} \frac{|g_i - f_i|}{|f_i|} \tag{5.3}$$

by accumulation sums in double precision arithmetic. Here $\mathbf{g} = (g_i)_{i=0}^{n-1} = \mathrm{V}_T(\mathbf{t}) \cdot \mathbf{x}^{(sp)}$. Here are some of the computed examples.

	——————-comp_res——————-				——————-res——————-			
n	G–O	NH	R–O	Gauss	G–O	NH	R–O	Gauss
5	1.2e–07	5.8e–07	1.5e–07	2.3e–07	5.7e–08	5.8e–07	9.4e–08	1.2e–07
10	8.5e–07	5.6e–06	8.6e–07	7.1e–07	3.9e–07	6.8e–07	4.2e–07	1.6e–07
15	3.6e–06	1.1e–05	4.8e–06	1.2e–06	4.3e–07	4.3e–07	3.7e–07	3.1e–07
20	5.6e–06	2.5e–05	6.5e–06	1.8e–06	5.5e–07	4.5e–06	1.5e–06	4.1e–07
30	4.3e–06	6.6e–05	1.1e–05	2.0e–06	3.9e–06	1.3e–05	3.9e–06	5.9e–07
40	3.0e–05	9.4e–05	6.9e–05	3.6e–06	2.6e–05	9.6e–06	1.4e–05	7.4e–07
50	1.9e–05	1.5e–04	4.5e–05	1.4e–05	3.0e–06	2.2e–05	2.3e–05	6.6e–07

Table 10. Nodes : zeros of $T_n(\lambda)$. Coordinates of right-hand side $\mathbf{f}$: random integers on (–10, 10).

	——————-comp_res——————-				——————-res——————-			
n	G–O	NH	R–O	Gauss	G–O	NH	R–O	Gauss
5	1.4e–07	3.0e–07	6.2e–08	7.1e–08	1.4e–07	3.0e–07	6.2e–08	7.1e–08
10	3.8e–07	9.6e–07	2.7e–07	3.3e–07	3.8e–07	9.6e–07	2.7e–07	3.3e–07
15	7.6e–07	1.4e–06	5.8e–07	2.3e–07	7.6e–07	1.4e–06	5.8e–07	2.3e–07
20	9.2e–07	3.0e–06	6.2e–07	1.1e–06	9.2e–07	3.0e–06	6.2e–07	1.1e–06
30	1.0e–06	8.2e–06	8.0e–06	6.5e–07	1.0e–06	8.2e–06	8.0e–06	6.5e–07
40	6.9e–06	7.5e–06	6.3e–06	1.7e–06	6.9e–06	7.5e–06	6.3e–06	1.7e–06
50	1.2e–05	1.6e–05	1.5e–05	1.7e–06	1.2e–05	1.6e–05	1.5e–05	1.7e–06

Table 11. Nodes : zeros of $T_n(\lambda)$. Right hand side : $\mathbf{f} = ((-1)^i)_{i=0}^{n-1}$.

	comp_res				res			
n	G–O	NH	R–O	Gauss	G–O	NH	R–O	Gauss
5	1.6e–07	6.1e–07	2.2e–07	2.2e–07	1.6e–07	5.4e–07	2.0e–07	1.1e–07
10	3.5e–07	6.9e–07	6.9e–07	2.2e–06	3.2e–07	6.2e–07	6.2e–07	2.1e–07
15	2.7e–06	7.8e–06	2.2e–06	3.3e–06	4.3e–07	1.0e–06	6.8e–07	1.6e–07
20	2.2e–06	8.4e–06	8.3e–06	1.7e–06	8.0e–07	5.2e–06	5.2e–06	2.2e–07
30	4.9e–06	3.3e–05	3.2e–05	2.0e–06	2.3e–06	2.0e–05	2.0e–05	4.7e–07
40	6.6e–06	2.7e–05	2.8e–05	3.0e–06	2.2e–06	1.8e–05	1.9e–05	5.0e–07
50	2.6e–05	1.6e–04	1.6e–04	3.6e–06	2.4e–06	7.8e–05	7.8e–05	5.4e–07

Table 12. Nodes : extrema of $T_n(\lambda)$. Coordinates of right-hand side $\mathbf{f}$: random integers on (–10,10).

References

1. Björck A. and Pereyra V. (1970) : Solution of Vandermonde systems of linear equations, Math. of Computation, **24**, 893 – 903.

2. Calvetti D. and Reichel L. (1993) : Fast inversion of Vandermonde-like matrices involving orthogonal polynomials, preprint.

3. Gautschi W. (1983) : The condition of Vandermonde–like matrices, involving orthogonal polynomials, Linear Algebra Appl., **52 / 52**, 293 – 300.

4. Gohberg I., Kailath T., Koltracht I. and Lancaster P. (1987) : Linear complexity parallel algorithms for linear systems of equations with recursive structure, Linear Algebra Appl., **88/89**, 271 – 315.

5. Gohberg I. and Koltracht I. (1991) : Mixed, componentwise and structured condition numbers , preprint.

6. Gohberg I. and Olshevsky V. (1993) : Fast algorithms with preprocessing for multiplication of transpose Vandermonde matrix and Cauchy matrix with vectors, submitted.

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

8. Higham N. (1988) : Fast solution of Vandermonde–like systems involving orthogonal polynomials, IMA J. Numerical Analysis, **8**, 473 – 486.

9. Higham N. (1990) : Stability analysis of algorithms for solving confluent Vandermonde–like systems, SIAM J. Matrix Analysis, **11** No. 1, 23 – 41.

10. Karlin S. and Szegö G. (1960) : On certain determinants whose elements are orthogonal polynomials. In Gabor Szegö Collected papers – volume 3 (1982), Birkhauser, Boston, 603 – 762.

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

12. MATLAB User's Guide (1991) : The Math Works, Inc., Natick MA.

13. Press W., Flannery B., Teykolsky S. and Vetterling W. (1988) : Numerical recipes in C, Cambridge University Press, Cambridge, 1988.

14. Reichel L. and Opfer G. (1991) : Chebyshev–Vandermonde systems, Math. of Computation, **57** No. 196, 703 – 721.

15. Tang W. and Golub G. (1981) : The block decomposition of a Vandermonde matrix and its applications, BIT, **21**, 505 – 517.

16. Traub J. (1966) : Associated polynomials and uniform methods for the solution of linear problems, SIAM Review, **8**, 277 – 301.

17. Verde-Star L. (1988) : Inverses of generalized Vandermonde matrices, J. Math. Anal. Appl., **131**, 341 – 353.