A Björck-Pereyra-type algorithm for Szegö-Vandermonde matrices based on properties of unitary Hessenberg matrices[1]

Tom Bella

Department of Mathematics, University of Connecticut, 196 Auditorium Road, Storrs, CT, 06269-3009, USA

Yuli Eidelman

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

Israel Gohberg

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

Israel Koltracht

Department of Mathematics, University of Connecticut, 196 Auditorium Road, Storrs, CT, 06269-3009, USA

Vadim Olshevsky

Department of Mathematics, University of Connecticut, 196 Auditorium Road, Storrs, CT, 06269-3009, USA

AMS classification: 15A06; 65F05

Abstract

In this paper we carry over the Björck-Pereyra algorithm for solving Vandermonde linear systems to what we suggest to call Szegö-Vandermonde systems $V_\Phi(x)$, i.e., polynomial-Vandermonde systems where the corresponding polynomial system Φ is the *Szegö polynomials*. The properties of the corresponding *unitary Hessenberg* matrix allow us to derive a fast $O(n^2)$ computational procedure. We present numerical experiments that indicate that for ill-conditioned matrices the new algorithm yields better forward accuracy than Gaussian elimination.

Key words: Bjorck-Pereyra algorithm, Szego polynomials, Unitary Hessenberg matrices, Vandermonde matrices, polynomial-Vandermonde matrices, fast algorithms

1 Introduction

Vandermonde matrices of the form

$$V(x) = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} \end{bmatrix} \qquad (1.1)$$

are classical, and explicit expressions for their determinants and inverses are well known. The structure in (1.1) can be exploited to speed-up computations involving $V(x)$ allowing one to design *fast algorithms*, i.e., algorithms whose complexity is at least an order of magnitude less than that of standard (structure-ignoring) methods.

For example, solving a Vandermonde linear system using methods that ignore this special structure requires $O(n^3)$ operations, whereas the now well known Björck-Pereyra algorithm (see [BP70], [GVL96], [O03]) solves the system in $O(n^2)$ operations. Moreover, it was shown that the Björck-Pereyra algorithm is not only faster but it is often more accurate than the standard methods, see, e.g. [H87] for the forward stability and [BKO99] for the backward stability analyses.

Classical Vandermonde matrices (1.1) appear in polynomial computations exploiting the monomial basis $\{1, x, x^2, \ldots, x^{n-1}\}$. A slightly more general class of matrices arise by following the same essential structure in (1.1), but permitting polynomials in place of the monomials. Such matrices are polynomial-Vandermonde matrices of the form

$$V_R(x) = \begin{bmatrix} r_0(x_1) & r_1(x_1) & \cdots & r_{n-1}(x_1) \\ r_0(x_2) & r_1(x_2) & \cdots & r_{n-1}(x_2) \\ \vdots & \vdots & & \vdots \\ r_0(x_n) & r_1(x_n) & \cdots & r_{n-1}(x_n) \end{bmatrix}, \qquad (1.2)$$

where $R = \{r_0(x), r_1(x), \ldots, r_{n-1}(x)\}$ is a given system of polynomials. The Björck-Pereyra algorithm for solving linear systems as well as the Traub algorithm for inversion have been extended to polynomial-Vandermonde matrices in several notable special cases of the polynomial system R. The resulting fast algorithms along with corresponding references are listed in Table 1.

Email addresses: bella@math.uconn.edu (Tom Bella), eideyu@post.tau.ac.il (Yuli Eidelman), gohberg@post.tau.ac.il (Israel Gohberg), koltracht@math.uconn.edu (Israel Koltracht), olshevsky@math.uconn.edu (Vadim Olshevsky).

[1] Supported by the NSF grant 0242518.

Coefficient matrix	Inversion algorithm	Algorithm solving linear system
Vandermonde	Parker-Forney-Traub algorithm [P64], [F66], [T66]	Björck-Pereyra algorithm [BP70]
Chebyshev-Vandermonde	Gohberg-Olshevsky algorithm [GO94]	Reichel-Opfer algorithm [RO91]
three-term Vandermonde	Calvetti-Reichel algorithm [CR93]	Higham algorithm [H90]
Szegö-Vandermonde	Olshevsky algorithm [O01]	???

Table 1. Fast $O(n^2)$ algorithms for three-term Vandermonde matrices.

However, many problems involve computations with the Szegö polynomials $\Phi = \{\phi_k^{\#}(x)\}$; that is, polynomials orthonormal on the unit circle with respect to a suitable inner product,

$$< p(x), q(x) >= \frac{1}{2\pi}\int_{-\pi}^{\pi} p(e^{i\theta}) \cdot [q(e^{i\theta})]^* w^2(\theta) d\theta. \tag{1.3}$$

It is well known that the Szegö polynomials are completely described by the two-term recurrence relations

$$\begin{bmatrix} \phi_0(x) \\ \phi_0^{\#}(x) \end{bmatrix} = \frac{1}{\mu_0}\begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \begin{bmatrix} \phi_{k+1}(x) \\ \phi_{k+1}^{\#}(x) \end{bmatrix} = \frac{1}{\mu_{k+1}}\begin{bmatrix} 1 & -\rho_{k+1}^* \\ -\rho_{k+1} & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & x \end{bmatrix}\begin{bmatrix} \phi_k(x) \\ \phi_k^{\#}(x) \end{bmatrix}, \tag{1.4}$$

see [GS58], [G48]. The parameters $\{\rho_0, \rho_1, \dots, \rho_n\}$ (with $\rho_0 := -1$), are called *reflection coefficients* (the names *parcor coefficients* and *Schur parameters* are also in use). The numbers $\mu_k = \sqrt{1 - |\rho_k|^2}$ are called the *complementary parameters* ($\mu_k := 1$ if $|\rho_k| = 1$), and $\phi_k(x) = x^k[\phi^{\#}(\frac{1}{x^*})]^*$.

We use the notation $\phi_k^{\#}(x)$ for the Szegö polynomials following [O98], [O01] and the engineering references therein where $\phi_k(x)$ are called *backward predictor polynomials*, and the Szegö polynomials $\phi_k^{\#}(x)$ are obtained from them.

In this paper we present a Björck-Pereyra-type algorithm to solve linear systems where the coefficient matrix is a polynomial-Vandermonde matrix whose corresponding system of polynomials are Szegö polynomials. We call such a matrix a Szegö-Vandermonde matrix. This algorithm will be based on the Hessenberg matrix

$$H = \begin{bmatrix} -\rho_1\rho_0^* & -\rho_2\mu_1\rho_0^* & -\rho_3\mu_2\mu_1\rho_0^* & \cdots & -\rho_{n-1}\mu_{n-2}\cdots\mu_1\rho_0^* & -\rho_n\mu_{n-1}\cdots\mu_1\rho_0^* \\ \mu_1 & -\rho_2\rho_1^* & -\rho_3\mu_2\rho_1^* & \cdots & -\rho_{n-1}\mu_{n-2}\cdots\mu_2\rho_1^* & -\rho_n\mu_{n-1}\cdots\mu_2\rho_1^* \\ 0 & \mu_2 & -\rho_3\rho_2^* & \cdots & -\rho_{n-1}\mu_{n-2}\cdots\mu_3\rho_2^* & -\rho_n\mu_{n-1}\cdots\mu_3\rho_2^* \\ \vdots & \ddots & \mu_3 & & \vdots & \vdots \\ \vdots & & \ddots & \ddots & -\rho_{n-1}\rho_{n-2}^* & -\rho_n\mu_{n-1}\rho_{n-2}^* \\ 0 & \cdots & \cdots & 0 & \mu_{n-1} & -\rho_n\rho_{n-1}^* \end{bmatrix} \tag{1.5}$$

that was used in many areas, see, e.g, [G82], [BC92], [R95], and the references therein. This matrix has many nice properties. For instance, it is well known that H differs from a unitary matrix only in the last column. Additionally, it can be seen that if H_k is the leading principal $k \times k$ submatrix of H, then

$$\det(xI - H_k) = \frac{1}{\mu_1 \cdots \mu_k}\phi_k^{\#}(x) \tag{1.6}$$

see, for instance, [G82].

Such almost-unitary Hessenberg matrices have been studied in many contexts, notably in the fields of numerical linear algebra, operator theory, and signal processing. In numerical linear algebra, matrices H are related to Gaussian quadrature on the unit circle, as well as direct and inverse unitary eigenvalue problems, and efficient algorithms for several problems can be found in [G82], [G86], [GR90], [AGR93], [ACR96], among others. In operator theory literature the structure in (1.5) is associated with the *Naimark dilation*, see, e.g., [C84], Sec. 2.4 in [BC92], and Sec. 6.7 in [FF89]. Szegö polynomials can often be found in signal processing literature, because they describe the state-space structure for lattice digital filters, see, e.g., [R95], [ML80], [KP83], [TKH83], [K85].

The paper is structured as follows. In the next section we recall the classical Björck-Pereyra algorithm for solving a classical Vandermonde linear system. In Section 3, after a brief bit of background the main result is presented, a recursive factorization of the inverse of a Szegö-Vandermonde matrix, and formulas for the resulting fast algorithm for solving the corresponding linear system. In Section 4 several interesting numerical experiments are performed, as the proposed algorithm yields good forward error results while solving ill-conditioned systems.

Acknowledgement. The authors would like to sincerely thank the anonymous referee for very carefully reading the paper, and for a number of suggestions that allowed us to improve the presentation.

2 The classical Björck-Pereyra algorithm

Recall that in [BP70], the authors derive a representation for the inverse $V(x)^{-1}$ of an $n \times n$ Vandermonde matrix (1.1) as the product of bidiagonal matrices, that is,

$$V(x)^{-1} = U_1 \cdots U_{n-1} \cdot \tilde{L}_{n-1} \cdots \tilde{L}_1 \tag{2.1}$$

and used this result to solve the linear system $V(x)a = f$ by computing the solution vector

$$a = U_1 \cdots U_{n-1} \cdot \tilde{L}_{n-1} \cdots \tilde{L}_1 f \tag{2.2}$$

which solves the linear system in $\frac{5}{2}n^2$ operations. This is an order of magnitude improvement over Gaussian elimination, which is well known to require $O(n^3)$ operations in general. This favorable complexity results from the fact that the matrices U_k and $\tilde{L}_k$ are sparse. More specifically, if $V(x)$ is given by (1.1), the factors U_k and $\tilde{L}_k$ are given by

$$U_k = \left[\begin{array}{c|cccc} I_{k-1} & & & & \\ \hline & 1 & -x_k & & \\ & & 1 & \ddots & \\ & & & \ddots & -x_k \\ & & & & 1 \end{array}\right], \tag{2.3}$$

$$\tilde{L}_k = \left[\begin{array}{c|ccc} I_k & & & \\ \hline & \frac{1}{x_{k+1}-x_1} & & \\ & & \ddots & \\ & & & \frac{1}{x_n - x_{n-k}} \end{array}\right] \cdot \left[\begin{array}{c|cccc} I_{k-1} & & & & \\ \hline & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \end{array}\right]. \tag{2.4}$$

In the next section we will present our algorithm for solving Szegö-Vandermonde systems by obtaining a counterpart of the above factorization.

3 Factorization formula

In order to derive an analog of the Björck-Pereyra algorithm for the Szegö case, we will use the concept of associated polynomials, defined next.

3.1 Associated (generalized Horner) polynomials

Following [KO97] define the *associated polynomials* $\widehat{R} = \{\widehat{r}_0(x), \ldots, \widehat{r}_n(x)\}$ for a given system of polynomials $R = \{r_0(x), \ldots, r_n(x)\}$ via the relation

$$\frac{r_n(x) - r_n(y)}{x - y} = \begin{bmatrix} r_0(x) & r_1(x) & r_2(x) & \cdots & r_{n-1}(x) \end{bmatrix} \cdot \begin{bmatrix} \hat{r}_{n-1}(y) \\ \hat{r}_{n-2}(y) \\ \vdots \\ \hat{r}_1(y) \\ \hat{r}_0(y) \end{bmatrix}. \tag{3.1}$$

with additionally $\widehat{r}_n(x) = r_n(x)$.

However, before proceeding we first clarify the existence of such polynomials. Firstly, the polynomials associated with the monomials exist. Indeed, if P is the system of $n+1$ polynomials $P = \{1, x, x^2, ..., x^{n-1}, r_n(x)\}$, then

$$\frac{r_n(x) - r_n(y)}{x - y} = \begin{bmatrix} 1 & x & x^2 & \cdots & x^{n-1} \end{bmatrix} \cdot \begin{bmatrix} \hat{p}_{n-1}(y) \\ \hat{p}_{n-2}(y) \\ \vdots \\ \hat{p}_1(y) \\ \hat{p}_0(y) \end{bmatrix} = \sum_{i=0}^{n-1} x^i \cdot \hat{p}_{n-1-i}(y), \tag{3.2}$$

and in this case the associated polynomials $\widehat{P}$ can be seen to be the classical Horner polynomials (see, e.g., [KO97, Section 3.]).

Secondly, given a system of polynomials $R = \{r_0(x), r_1(x), \ldots, r_{n-1}(x), r_n(x)\}$, there is a corresponding system of polynomials $\hat{R} = \{\hat{r}_0(x), \hat{r}_1(x), \ldots, \hat{r}_{n-1}(x), \hat{r}_n(x)\}$ (with $\hat{r}_n(x) = r_n(x)$) satisfying (3.1). One can see that, given a polynomial system R with $\deg(r_k) = k$, the polynomials in R can be obtained from the monomial basis by

$$\begin{bmatrix} 1 & x & x^2 & \cdots & x_{n-1} \end{bmatrix} S = \begin{bmatrix} r_0(x) & r_1(x) & r_2(x) & \cdots & r_{n-1}(x) \end{bmatrix} \tag{3.3}$$

where S is an $n \times n$ upper triangular invertible matrix capturing the recurrence relations of the polynomial system R. Inserting SS^{-1} into (3.2) between the row and column vectors and using (3.3), we see that the polynomials associated with R are

$$\begin{bmatrix} \hat{r}_{n-1}(y) \\ \hat{r}_{n-2}(y) \\ \vdots \\ \hat{r}_1(y) \\ \hat{r}_0(y) \end{bmatrix} = S^{-1} \begin{bmatrix} \hat{p}_{n-1}(y) \\ \hat{p}_{n-2}(y) \\ \vdots \\ \hat{p}_1(y) \\ \hat{p}_0(y) \end{bmatrix} \tag{3.4}$$

where $\widehat{P} = \{\widehat{p}_0(x), \ldots, \widehat{p}_{n-1}(x)\}$ are the classical Horner polynomials and S is from (3.3).

3.2 Factorization formula

In this and subsequent sections we consider the Szegö polynomials $\Phi = \{\phi_k^{\#}(x)\}$, i.e. the polynomials orthogonal on the unit circle satisfying the two-term recurrence relations (1.4). The following lemma will be useful in finding the factorization formula below.

Lemma 1 *Let $\Phi = \{\phi_k^{\#}(x)\}_{k=0}^{n-1}$ be a system of Szegö polynomials corresponding to the reflection coefficients $\{\rho_k\}_{k=0}^{n}$ as defined in Section 1. For $k = 1, 2, \ldots, n-1$ denote by $\Phi^{(k)}$*

the system of polynomials $\Phi^{(k)} = \{\hat{\phi}_0^{(k)}(x), \ldots, \hat{\phi}_k^{(k)}(x)\}$ *associated with the truncated system* $\{\phi_0^{\#}(x), \ldots, \phi_k^{\#}(x)\}$*. Then*

$$\begin{bmatrix} \hat{\phi}_0^{(1)}(x) & \hat{\phi}_1^{(2)}(x) & \cdots & \hat{\phi}_{n-2}^{(n-1)}(x) \\ & \hat{\phi}_0^{(2)}(x) & \cdots & \hat{\phi}_{n-3}^{(n-1)}(x) \\ & & \ddots & \vdots \\ & & & \hat{\phi}_0^{(n-1)}(x) \end{bmatrix}^{-1} = \begin{bmatrix} \mu_1 & -x-\rho_2\rho_1^* & -\rho_3\mu_2\rho_1^* & \cdots & -\rho_{n-1}\mu_{n-2}\cdots\mu_2\rho_1^* \\ & \mu_2 & -x-\rho_3\rho_2^* & \cdots & -\rho_{n-1}\mu_{n-2}\cdots\mu_3\rho_2^* \\ & & \mu_3 & \ddots & \vdots \\ & & & \ddots & -x-\rho_{n-1}\rho_{n-2}^* \\ & & & & \mu_{n-1} \end{bmatrix} \tag{3.5}$$

Proof. It was shown in [O98, eq. (4.8)] that the n-term recurrence relations for the truncated associated polynomials $\widehat{\Phi}^{\#}$ are

$$\mu_{k-m}\widehat{\phi}_m^{(k)}(x) = x \cdot \widehat{\phi}_{m-1}^{(k)}(x) + \rho_{k-m+1}\rho_{k-m}^* \cdot \widehat{\phi}_{m-1}^{(k)}(x) + \rho_{k-m+2}\mu_{k-m+1}\rho_{k-m}^* \cdot \widehat{\phi}_{m-2}^{(k)}$$
$$+ \ldots + \rho_k\mu_{k-1}\cdots\mu_{k-m+1}\rho_{k-m}^* \cdot \widehat{\phi}_0^{(k)}(x), \tag{3.6}$$

for $m = 1, \ldots, k-1$, and also

$$\mu_k\widehat{\phi}_0^{(k)} = 1. \tag{3.7}$$

Now consider the product

$$\begin{bmatrix} \mu_1 & -x-\rho_2\rho_1^* & -\rho_3\mu_2\rho_1^* & \cdots & -\rho_{n-1}\mu_{n-2}\cdots\mu_2\rho_1^* \\ & \ddots & & & \vdots \\ & & \mu_i \;\; -x-\rho_{i+1}\rho_i^* & -\rho_{i+2}\mu_{i+1}\rho_i^* \;\; \cdots & -\rho_{n-1}\mu_{n-2}\cdots\mu_{i+1}\rho_i^* \\ & & \ddots & & \vdots \\ & & & & \mu_{n-1} \end{bmatrix} \begin{bmatrix} \widehat{\phi}_0^{(1)}(x) & \cdots & \widehat{\phi}_{j-1}^{(j)}(x) & \cdots & \widehat{\phi}_{n-2}^{(n-1)}(x) \\ & & \widehat{\phi}_{j-2}^{(j)}(x) & & \widehat{\phi}_{n-3}^{(n-1)}(x) \\ & \ddots & \vdots & & \vdots \\ & & \widehat{\phi}_0^{(j)}(x) & & \widehat{\phi}_1^{(n-1)}(x) \\ & & & \ddots & \widehat{\phi}_0^{(n-1)}(x) \end{bmatrix}, \tag{3.8}$$

where the (i, j) entry of this product defined by the highlighted row and column can be seen as (3.6) with $k = j, m = j - i$ if $i \neq j$ and (3.7) with $k = i$ if $i = j$. Thus this product is the identity, implying (3.5). □

Our algorithm involves a recursive factorization of the inverse matrix $V_R(x)^{-1}$. In the following we use the notation $x_{1:n} = (x_1, \ldots, x_n)$, etc.

Lemma 2 *Let* $\Phi = \{\phi_k^{\#}(x)\}_{k=0}^{n-1}$ *be a system of Szegö polynomials corresponding to the reflection coefficients* $\{\rho_k\}_{k=0}^n$ *and complementary parameters* $\{\mu_k\}_{k=0}^n$ *as defined in Section 1, and let* $x_{1:n}$ *be* n *distinct points. Then the inverse of* $V_R(x_{1:n})$ *admits a decomposition*

$$V_R(x_{1:n})^{-1} = U_1 \cdot \begin{bmatrix} 1 & 0 \\ 0 & V_R(x_{2:n})^{-1} \end{bmatrix} L_1, \tag{3.9}$$

with

$$U_1 = \begin{bmatrix} \frac{1}{\mu_0} & -x_1 - \rho_1\rho_0^* & -\rho_2\mu_1\rho_0^* & \cdots & \cdots & -\rho_{n-1}\mu_{n-2}\cdots\mu_1\rho_0^* \\ & \frac{1}{\mu_1} & -x_1 - \rho_2\rho_1^* & \cdots & \cdots & -\rho_{n-1}\mu_{n-2}\cdots\mu_2\rho_1^* \\ & & \frac{1}{\mu_2} & \ddots & & -\rho_{n-1}\mu_{n-2}\cdots\mu_3\rho_2^* \\ & & & \ddots & \ddots & \vdots \\ & & & & \frac{1}{\mu_{n-2}} & -x_1 - \rho_{n-1}\rho_{n-2}^* \\ & & & & & \frac{1}{\mu_{n-1}} \end{bmatrix}, \tag{3.10}$$

$$L_1 = \begin{bmatrix} 1 & & & \\ & \frac{1}{x_2-x_1} & & \\ & & \ddots & \\ & & & \frac{1}{x_n-x_1} \end{bmatrix} \begin{bmatrix} 1 & & & \\ -1 & 1 & & \\ \vdots & & \ddots & \\ -1 & & & 1 \end{bmatrix}. \tag{3.11}$$

Proof. Performing one step of Gaussian elimination on $V_R(x_{1:n})$ yields

$$V_R(x_{1:n}) = \begin{bmatrix} 1 & & & \\ 1 & 1 & & \\ \vdots & & \ddots & \\ 1 & & & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & & & \\ & x_2 - x_1 & & \\ & & \ddots & \\ & & & x_n - x_1 \end{bmatrix}.$$

$$\cdot \begin{bmatrix} 1 & 0 \\ 0 & \bar{R} \end{bmatrix} \cdot \left[\begin{array}{c|c} \phi_0^{\#}(x_1) & \phi_1^{\#}(x_1) \ \cdots \ \phi_{n-1}^{\#}(x_1) \\ \hline 0 & I \end{array}\right], \tag{3.12}$$

where the matrix $\bar{R} = \left[\frac{\phi_j^{\#}(x_{i+1}) - \phi_1^{\#}(x_1)}{x_{i+1}-x_1} \right]$ consists of divided differences. By the discussion above, associated with the system Φ is the system $\widehat{\Phi} = \{\hat{\phi}_k(x)\}$. Following the notation of Lemma 1, denote by $\widehat{\Phi}^{(k)} = \{\hat{\phi}_0^{(k)}(x), \ldots, \hat{\phi}_k^{(k)}(x)\}$ the system of polynomials associated with the truncated system $\{\phi_0(x), ..., \phi_k(x)\}$. By the definition of the associated polynomials we have for $k = 1, 2, \ldots, n-1$

$$\frac{\phi_k^{\#}(x) - \phi_k^{\#}(y)}{x - y} = \begin{bmatrix} \phi_0^{\#}(x) & \phi_1^{\#}(x) & \phi_2^{\#}(x) & \cdots & \phi_{k-1}^{\#}(x) \end{bmatrix} \cdot \begin{bmatrix} \hat{\phi}_{k-1}^{(k)}(y) \\ \hat{\phi}_{n-2}^{(k)}(y) \\ \vdots \\ \hat{\phi}_1^{(k)}(y) \\ \hat{\phi}_0^{(k)}(y) \end{bmatrix} = \sum_{i=0}^{k-1} \phi_i^{\#}(x) \cdot \hat{\phi}_{k-1-i}^{(k)}(y).$$

Finally, denoting by $\widehat{\Phi}^{(k)} = \{\hat{\phi}_0^{(k)}(x), \ldots, \hat{\phi}_k^{(k)}(x)\}$ the system of polynomials associated with the truncated system $\{\phi_0(x), ..., \phi_k(x)\}$ we can further factor $\bar{R}$ as

$$\bar{R} = V_R(x_{2:n}) \cdot \begin{bmatrix} \hat{\phi}_0^{(1)}(x_1) & \hat{\phi}_1^{(2)}(x_1) & \cdots & \hat{\phi}_{n-2}^{(n-1)}(x_1) \\ & \hat{\phi}_0^{(2)}(x_1) & \cdots & \hat{\phi}_{n-3}^{(n-1)}(x_1) \\ & & \ddots & \vdots \\ & & & \hat{\phi}_0^{(n-1)}(x_1) \end{bmatrix}. \tag{3.13}$$

The last matrix on the right-hand side of (3.13) can be inverted by Lemma 1. Therefore, inverting (3.12) and substituting (3.5) results in (3.9). $\square$

3.3 *New Björck-Pereyra type algorithm*

Like the classical Björk-Pereyra algorithm, the recursive nature of the formula (3.9) allows a decomposition

$$V_R(x_{1:n})^{-1} = U_1 \cdot ... \cdot U_{n-1} \cdot L_{n-1} \cdot ... \cdot L_1, \tag{3.14}$$

with the upper and lower triangular factors given via recursively applying Lemma 2, arriving at

$$U_k = \left[\begin{array}{c|c} I_{k-1} & 0 \\ \hline 0 & \widetilde{U}_k \end{array}\right] = \left[\begin{array}{c|cccccc} I_{k-1} & & & & & & \\ \hline & \frac{1}{\mu_0} & -x_k - \rho_1\rho_0^* & -\rho_2\mu_1\rho_0^* & \cdots & \cdots & -\rho_{n-k}\mu_{n-k-1}\cdots\mu_1\rho_0^* \\ & & \frac{1}{\mu_1} & -x_k - \rho_2\rho_1^* & \cdots & \cdots & -\rho_{n-k}\mu_{n-k-1}\cdots\mu_2\rho_1^* \\ & & & \frac{1}{\mu_2} & \ddots & & -\rho_{n-k}\mu_{n-k-1}\cdots\mu_3\rho_2^* \\ & & & & \ddots & \ddots & \vdots \\ & & & & & \frac{1}{\mu_{n-k-1}} & -x_k - \rho_{n-k}\rho_{n-k-1}^* \\ & & & & & & \frac{1}{\mu_{n-k}} \end{array}\right], \tag{3.15}$$

$$L_k = \left[\begin{array}{c|cccc} I_{k-1} & & & & \\ \hline & 1 & & & \\ & & \frac{1}{x_{k+1}-x_k} & & \\ & & & \ddots & \\ & & & & \frac{1}{x_n-x_k} \end{array}\right] \left[\begin{array}{c|cccc} I_{k-1} & & & & \\ \hline & 1 & & & \\ & -1 & 1 & & \\ & \vdots & & \ddots & \\ & -1 & & & 1 \end{array}\right]. \tag{3.16}$$

Remark 1 *It is worth noting that there is a difference between the factors L_k in (3.16) and the factors $\tilde{L}_k$ in (2.4). From the uniqueness of the L factor in the LU-factorization, the formula is valid with either choice.*

The associated linear system can be solved by multiplying (3.14) by the right-hand side vector f in the linear system $V_R(x_{1:n})x = f$. However, unlike the classical Björck-Pereyra algorithm, the matrices U_k involved in the proposed algorithm are not sparse. The sparseness of these factors in the classical Vandermonde case (in fact they are even bidiagonal, see (2.3)) is exactly what reduces the complexity by an order of magnitude to $O(n^2)$. In fact, for a general polynomial system R, a similar derivation of an algorithm is possible, however the overall cost is again $O(n^3)$.

Although the matrices U_k are not sparse, a method of fast multiplication of U_k by a vector will allow the desired reduction in complexity. Denote by H_k the $k \times k$ leading submatrix of H given in (1.5) as in Section 1, and further denote

$$H_k(x) = H_k - xI; \tag{3.17}$$

that is, $H_k(x)$ is the leading $k \times k$ submatrix of H with entries on the main diagonal shifted by x. With this notation, by comparing (3.15) and (1.5), we make the observation that the matrix $\widetilde{U}_k$ given in (3.15) can be written as

$$U_k = \left[\begin{array}{c|c} I_{k-1} & 0 \\ \hline 0 & \widetilde{U}_k \end{array}\right], \qquad \widetilde{U}_k = \left[\begin{array}{c|c} \frac{1}{\mu_0} & \\ 0 & \\ \vdots & H_{n-k}(x_k) \\ 0 & \\ \hline 0 & 0 \cdots 0 \ \frac{1}{\mu_{n-k}} \end{array}\right]. \tag{3.18}$$

This observation reduces the problem of fast multiplication of U_k by a vector to that of fast multiplication of H_k by a vector. Fast multiplication of H_k by a vector is easily achieved by using the well known decomposition of H_k as

$$H_k = G_k(\rho_1) \cdot G_k(\rho_2) \cdot \ldots \cdot G_k(\rho_{k-1}) \cdot \tilde{G}_k(\rho_k), \tag{3.19}$$

where

$$G_k(\rho_j) = \operatorname{diag}\{I_{j-1}, \begin{bmatrix} \rho_j & \mu_j \\ \mu_j & -\rho_j^* \end{bmatrix}, I_{k-j-1}\}, \quad j = 1, 2, \ldots, k-1 \tag{3.20}$$

and

$$\tilde{G}_k(\rho_k) = \operatorname{diag}\{I_{k-1}, \rho_k\} \tag{3.21}$$

see, for instance, [G82], [BC92], or [R95].

Therefore, the factorization (3.19) reduces multiplication of H_k by a vector to $k-1$ circular rotations, thus suggesting an efficient $O(n^2)$ implementation for our Björck-Pereyra like algorithm for Szegö-Vandermonde matrices.

4 Numerical Illustrations

To check the numerical performance of the proposed algorithm, the following numerical experiments were performed. All algorithms have been implemented in MATLAB v6 in double precision. These results were compared with exact solutions calculated in Maple v7 using software-implemented 40 digits of precision (or more as needed as detailed below).

Herein matrices of size 30×30 were used. In the tables, BP-SV denotes the proposed Björck-Pereyra like algorithm for Szegö-Vandermonde systems implemented using of the previous Section. The factors L_k from (3.16) were used. GE2t and GE3t indicate MATLAB's Gaussian elimination, performed on the Szegö-Vandermonde matrix generated using the two-term and three-term [2] recurrence relations, respectively. Finally, $\mathrm{cond}(V)$ denotes the condition number of the matrix V computed via the MATLAB command `cond()`.

It is known (see [RO91], [H90]) that reordering the nodes for polynomial Vandermonde matrices, which corresponds to a permutation of the rows, can affect the accuracy of related algorithms. In particular, ordering the nodes according to the *Leja ordering*

$$|x_1| = \max_{1\le i\le n} |x_i|, \qquad \prod_{j=1}^{k-1} |x_k - x_j| = \max_{k\le i\le n} \prod_{j=1}^{k-1} |t_i - t_j|, \quad k = 2,\dots,n-1 \tag{4.1}$$

(see [RO91], [H90], [O03]) improves the performance of many similar algorithms. As the performance of the suggested BP-SV algorithm is improved by this ordering, we include only experiments where Leja ordering is used. A counterpart of this ordering is known for Cauchy matrices, see [BKO02].

In all experiments, we compare the forward accuracy of the algorithm, defined by

$$e = \frac{\|x - \hat{x}\|_2}{\|x\|_2} \tag{4.2}$$

where x is the solution computed by each algorithm in MATLAB in double precision, and $\hat{x}$ is what is taken as the exact solution. This exact solution is the result of solving the system in Maple using 40 digits (or more) of software-implemented precision.

The results are arranged as follows. In the first four experiments, we compare the forward accuracy of the algorithms with that of Gaussian elimination under varying conditions. We present the interesting relationship between the accuracy of the algorithms and the condition numbers of the matrices involved. This relationship is shown in Figure 1. The dependence of the algorithm on the direction of the right-hand side vector is investigated in Experiment 5. Finally, to address the case where the accuracy is not as good as Gaussian elimination, we present a first step in the direction of better results under these conditions using one step of iterative refinement.

Experiment 1. In the first experiment (Table 2), the values for $\rho_k, k = 1,\dots,n$ were chosen randomly (complex) in the unit disc, similarly for the nodes $x_k, k = 1,\dots,n$, and the entries of the right hand side vector $b_k, k = 1,\dots,n$. Choosing such parameters, the condition of the matrices is large, however, the BP-SV algorithm still produces excellent forward accuracy. Ten trial results are listed in Table 2.

[2] For the three-term recurrence relations for the Szegö polynomials see for instance [O98] and references therein.

#	cond(V)	BP-SV	GE2t	GE3t
1	7e+14	5e-15	8e-06	1e-05
2	1e+15	2e-15	5e-05	8e-05
3	3e+15	3e-15	6e-04	2e-04
4	1e+18	2e-15	1e-01	7e-01
5	2e+15	4e-15	4e-04	4e-04
6	5e+17	1e-14	5e-02	3e-02
7	1e+16	4e-15	2e-05	4e-05
8	1e+18	1e-15	2e-02	1e-02
9	1e+18	2e-15	1e-00	1e-00
10	9e+18	6e-16	5e-01	7e-01

Table 2. Random $\{\rho_k\}, \{x_k\}, \{b_k\}$ on the unit disc.

Experiment 2. In the second experiment (Table 3), the values for x_k and b_k were chosen in the same manner, but the reflection coefficients are chosen randomly within the unit disc subject to $.999 \leq |\rho_k| < 1$; that is, close to the unit circle. This has the effect of producing even more ill-conditioned matrices. Due to this increase in condition numbers, in order to maintain accuracy, this experiment was compared with solutions computed using 80 digits of software-implemented precision. The proposed algorithm still produces very good forward accuracy, as seen in Table 3.

#	cond(V)	BP-SV	GE2t	GE3t
1	1e+47	5e-14	9e-02	1e-01
2	4e+51	2e-15	1e-00	1e-00
3	5e+48	3e-15	2e-02	1e-02
4	1e+50	6e-15	9e-02	9e-02
5	1e+48	3e-15	2e-04	3e-04
6	5e+51	3e-15	3e-01	6e-02
7	8e+48	1e-14	6e-03	7e-03
8	9e+52	3e-15	6e-01	7e-01
9	3e+51	4e-15	1e-00	1e-00
10	8e+50	1e-15	9e-01	9e-01

Table 3. Random $\{\rho_k\}, \{x_k\}, \{b_k\}$ on the unit disc, with $.999 \leq |\rho_k| < 1$.

Experiment 3. The third experiment consisted of selection of random values within the unit disc for ρ_k and b_k as in the first experiment, and then selecting the nodes x_k as the roots of the polynomial $\phi_n(x)$ defined by the reflection coefficients $\{\rho_k\}_{k=0}^{n}$. Choosing the parameters in this manner results in a more well-conditioned matrix than the previous experiments. The BP-SV algorithm does not perform well compared with GE in this case. See Table 4.

#	cond(V)	BP-SV	GE2t	GE3t
1	4e+08	3e-11	1e-12	3e-12
2	2e+07	8e-11	2e-14	5e-14
3	9e+05	1e-09	9e-15	3e-14
4	3e+08	3e-11	2e-13	2e-13
5	6e+09	2e-09	1e-13	3e-13
6	1e+09	3e-11	3e-14	8e-15
7	5e+05	3e-11	1e-14	3e-14
8	3e+06	7e-11	1e-13	7e-14
9	7e+05	1e-08	2e-14	6e-14
10	2e+06	1e-10	1e-13	1e-13

Table 4. Random $\{\rho_k\}, \{b_k\}$ on the unit disc, $\{x_k\}$ chosen as roots of $\phi_n(x)$ corresponding to $\{\rho_k\}$.

Experiment 4. Further investigation of the behavior of the algorithms on the well-conditioned matrices in the previous experiment were done by fixing the values $\{\rho_k\}$ and $\{x_k\}$, and slowly perturbing the x_k by increasing amounts such that the condition number of the involved matrix grows. Some results for the condition number of the resulting matrix and how the algorithms performed are given in Table 5.

Experiments 1-4 are essentially summarized by Figure 1. As opposed to the expectation that poorly conditioned matrices will result in less accuracy as with Gaussian elimination, this is not the case with this algorithm. The algorithm performs well for ill-conditioned matrices, and poorly for better conditioned matrices.

#	cond(V)	BP-SV	GE2t	GE3t
1	1e+06	6e-10	6e-15	1e-14
2	6e+06	4e-10	6e-14	7e-14
3	9e+07	7e-11	3e-13	1e-13
4	1e+09	1e-10	2e-12	2e-12
5	3e+09	3e-10	3e-12	7e-12
6	6e+09	8e-12	2e-13	4e-13
7	7e+11	8e-12	8e-10	2e-10
8	3e+12	5e-12	2e-10	2e-10
9	9e+11	2e-12	4e-11	6e-11
10	2e+12	8e-12	1e-11	2e-10
11	1e+12	2e-12	1e-11	1e-11
12	3e+13	2e-12	2e-09	1e-09
13	6e+13	9e-13	2e-10	8e-11
14	2e+13	2e-12	4e-11	9e-11
15	3e+16	2e-13	3e-07	2e-07
16	7e+15	1e-13	8e-07	2e-06
17	5e+16	3e-13	7e-07	6e-07
18	6e+16	1e-13	1e-07	1e-07
19	3e+16	1e-13	1e-08	1e-08
20	1e+18	1e-14	2e-05	1e-05

Table 5. Randomly chosen (fixed throughout all 20 trials) $\{\rho_k\}, \{b_k\}$ on the unit disc, $\{x_k\}$ chosen as roots of $\phi_n(x)$ corresponding to $\{\rho_k\}$, then the roots $\{x_k\}$ are slowly perturbed.

Remark 2 *This behavior is analogous to that of the classical Björck-Pereyra algorithm, which also performs well for ill-conditioned systems. Classical Vandermonde matrices with real nodes are always ill-conditioned as shown in [T94]. As opposed to this case, Szegö-Vandermonde matrices can be better conditioned.*

Experiment 5. In [CF88], [BKO99] it was shown that the behavior of BKO-type algorithms can depend on the direction of the right hand side vector. We include a similar experiment here where the outcome is consistent with observations in [BKO99]. This is illustrated in Figure 2, which shows the results given a fixed set of $\{\rho_k\}$ and $\{x_k\}$ on the unit circle, and the results of applying

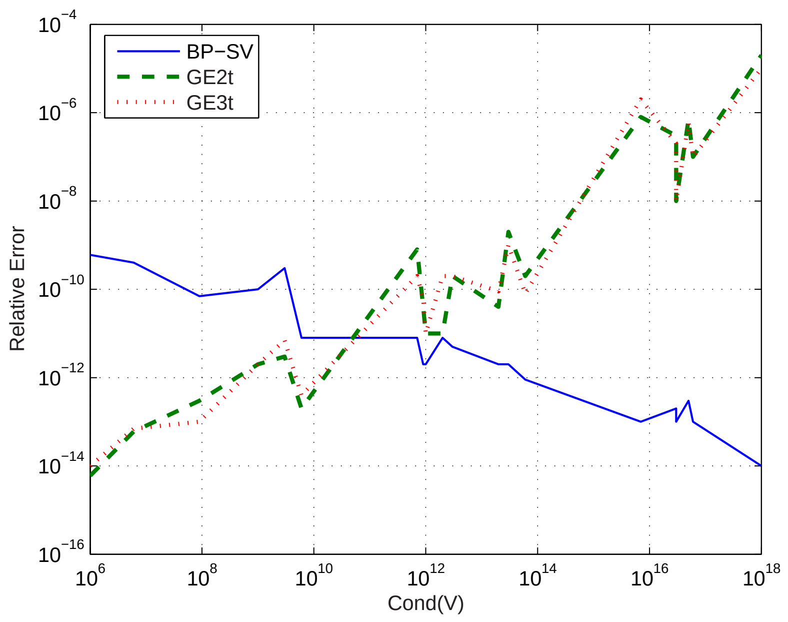

Fig. 1. Effects of conditioning.

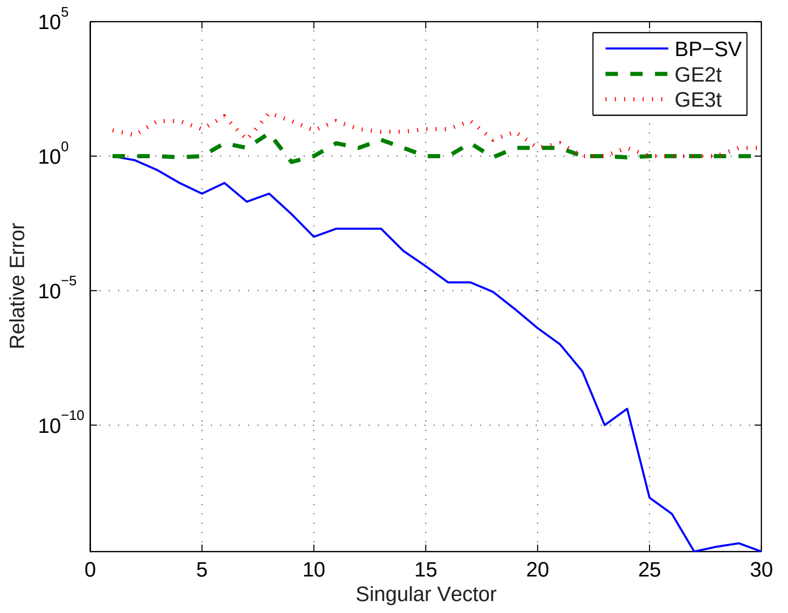

Fig. 2. Effects of different left singular vectors as right hand side.

the various algorithms to solve the system with each (left) singular vector as the right hand side. The results suggest the dependence of the performance of the BP-SV algorithm to the direction of the right hand side vector, as opposed to GE which is insensitive to this, as the accuracy of the algorithm significantly improves when passing from the first to the last singular vectors.

Experiment 6. In the final experiment, we attempt to correct the problem of the algorithm not performing well compared to Gaussian elimination for the better conditioned matrices. We implement a single step of iterative refinement; that is, after solving the system $V_\Phi x_1 = f$ using the algorithm, we then formulate and solve the system $V_\Phi x_2 = (f - V_\Phi x_1)$ using the algorithm and take our solution to the original problem to be the sum $x_1 + x_2$. The results of this experiment are given in Table 6, where BP-SV IR indicates that one step of iterative refinement has taken place.

The results are positive; although the above experiments indicate good accuracy for ill-conditioned matrices but not as good results for well-conditioned matrices, this drawback can be fixed by one-step of iterative refinement, as the results in Table 6 indicate.

#	cond(V)	BP-SV	BP-SV IR	GE2t	GE3t
1	1e+07	2e-09	1e-14	4e-13	6e-13
2	3e+07	2e-10	4e-14	4e-14	8e-14
3	1e+08	2e-10	4e-14	9e-13	2e-13
4	6e+08	6e-11	3e-14	4e-13	4e-13
5	1e+07	3e-10	4e-15	1e-13	2e-13
6	5e+05	5e-10	1e-15	4e-14	7e-14
7	3e+06	3e-10	1e-15	2e-14	2e-13
8	1e+08	7e-10	3e-15	4e-14	3e-14
9	2e+07	8e-10	2e-15	1e-13	9e-14
10	3e+07	5e-10	3e-14	8e-14	1e-13

Table 6. Implementing one step of iterative refinement.

Conclusions. These initial numerical experiments indicate that the proposed Björck-Pereyra like algorithm performs quite well for ill-conditioned systems, for which it showed behavior superior to Gaussian elimination with partial pivoting (GEPP) not only in speed but also in accuracy. For better conditioned system the new algorithm, while faster, may not give the same accuracy as GEPP. The experiments suggest that in this case the use of one step of iterative refinement can provide the same accuracy as that of GEPP. These observations are preliminary, and more practical experience may be needed. Additionally, a round-off error analysis may provide more insights.

5 Conclusions

In this paper an analog of the well known Björck-Pereyra algorithm was presented for Szegö-Vandermonde matrices. This algorithm was derived by exploiting the properties of the related unitary Hessenberg matrix, which resulted in the small computational complexity $O(n^2)$ operations, as opposed to the usual $O(n^3)$ of standard (structure-ignoring) methods. Initial numerical experiments using this algorithm indicate good performance for ill-conditioned systems, in fact better results than Gaussian elimination for the same systems.

References

[ACR96] G.Ammar, D.Calvetti and L.Reichel, *Continuation methods for the computation of zeros of Szegö polynomials*, Linear Algebra and Its Applications, **249** (1996), 125-155.

[AGR93] G.Ammar, W.Gragg and L.Reichel, *An analogue for the Szegö polynomials of the Clenshaw algorithm*, J. Computational Appl. Math., 46 (1993) pp., 211-216.

[BC92] M.Bakonyi and T.Constantinescu, *Schur's algorithm and several applications*, in Pitman Research Notes in Mathematics Series, vol. 61, Longman Scientific and Technical, Harlow, 1992.

[BKO99] T.Boros, T.Kailath and V.Olshevsky, *Fast Bjorck-Pereyra-type algorithm for parallel solution of Cauchy linear equations*, Linear Algebra and Its Applications, 302-303 (1999), p.265-293.

[BKO02] T.Boros, T.Kailath, V.Olshevsky, *Pivoting and backward stability of fast algorithms for solving Cauchy linear equations*, Special issue on structured and infinite systems of linear equations. Linear Algebra Appl. 343/344 (2002), 63–99.

[BP70] A.Björck and V.Pereyra, *Solution of Vandermonde Systems of Equations*, Math. Comp., **24** (1970), 893-903.

[C84] T.Constantinescu, *On the structure of the Naimark dilation*, J. of Operator Theory, **12**: 159-175 (1984).

[CF88] T.Chan and D.Foulser, *Effectively well-conditioned linear systems*, SIAM J. Sci. Stat. Computations, 9 (1988), 963 – 969.

[CR93] Calvetti, D. and Reichel, L. : *Fast inversion of Vandermonde-like matrices involving orthogonal polynomials*, BIT, 1993.

[F66] G.Forney, *Concatenated codes*, The M.I.T. Press, 1966, Cambridge.

[FF89] C.Foias and A.E.Frazho, *The Commutant Lifting Approach to Interpolation Problems*, Birkhauser Verlag, 1989.

[G82] W.B.Gragg, *Positive definite Toeplitz matrices, the Arnoldi process for isometric operators, and Gaussian quadrature on the unit circle* (in Russian). In : E.S. Nikolaev (Ed.), *Numerical methods in Linear Algebra*, pp. 16-32, Moskow University Press, 1982.
English translation in : J. Comput. and Appl. Math., **46**(1993), 183-198.

[G86] W.B.Gragg, *The QR algorithm for unitary Hessenberg matrices*, J.Comput. Appl. Math., **16** (1986), 1-8.

[G48] L.Y.Geronimus, *Polynomials orthogonal on a circle and their applications*, Amer. Math. Translations, **3** p.1-78, 1954 (Russian original 1948).

[GO94] I.Gohberg and V.Olshevsky, *Fast inversion of Chebyshev-Vandermonde matrices*, Numerische Mathematik, **67**, No. 1 (1994), 71 – 92.

[GR90] W.B.Gragg and L.Reichel, *A divide and conquer method for unitary and orthogonal eigenproblems*, Numer. Math., **57** (1990), 695-718.

[GS58] U.Grenader and G.Szegö, *Toeplitz forms and Applications*, University of California Press, 1958.

[GVL96] G.Golub and C.van Loan. Matrix Computations. Johns Hopkins University Press, 3rd edition, 1996.

[H87] N.J.Higham, *Error analysis of the Björck–Pereyra algorithms for solving Vandermonde systems*, Numerische mathematic, 50 (1987), 613 – 632.

[H90] N.J.Higham, *Stability analysis of algorithms for solving confluent Vandermonde-like systems*, SIAM J. Matrix Anal. Appl., **11**(1) (1990), 23–41.

[K85] H.Kimura, *Generalized Schwartz Form and Lattice-Ladder Realizations for Digital Filters*, IEEE Transactions on Circuits and Systems, **32**, No 11 (1985), 1130-1139.

[KO97] T.Kailath and V.Olshevsky, *Displacement structure approach to polynomial Vandermonde and related matrices*, Linear Algebra and Its Applications, **261** (1997), 49-90.

[KP83] T.Kailath and B.Porat, *State-space generators for orthogonal polynomials*, in Prediction theory and harmonic analysis, The Pesi Masani Volume, V.Mandrekar and H.Salehi (eds.), pp.131-163, North-Holland Publishing Company, 1983.

[ML80] M.Morf and D.T.Lee, *State-space structure of ladder canonical forms*, Proc. 18th Conf. on Control and Design, Dec. 1980, 1221-1224.

[O98] V.Olshevsky, *Eigenvector computation for almost unitary Hessenberg matrices and inversion of Szego-Vandermonde matrices via discrete transmission lines,* Linear Algebra and Its Applications, 285 (1998), 37-67.

[O01] V.Olshevsky, *Associated polynomials, unitary Hessenberg matrices and fast generalized Parker-Traub and Bjorck-Pereyra algorithms for Szego-Vandermonde matrices,* “Structured Matrices: Recent Developments in Theory and Computation,” 67-78, (D.Bini, E. Tyrtyshnikov, P. Yalamov., Eds.), 2001, NOVA Science Publ.

[O03] V. Olshevsky, *Pivoting for structured matrices and rational tangential interpolation*, in Fast Algorithms for Structured Matrices: Theory and Applications, CONM/323, p. 1 - 75, AMS publications, May 2003.

[P64] F.Parker, *Inverses of Vandermonde matrices*, Amer. Math. Monthly, 71 (1964), 410 - 411.

[R90] L.Reichel, *Newton interpolation at Leja points*, BIT, 30(1990), 23 – 41.

[R95] P.A.Regalia, *Adaptive IIR filtering in signal processing and control*, Marcel Dekker, New York, 1995.

[RO91] L.Reichel and G.Opfer, *Chebyshev-Vandermonde systems*, Math. of Comp., **57** (1991), 703-721.

[T66] J. Traub, *Associated polynomials and uniform methods for the solution of linear problems*, SIAM Review, 8, No. 3 (1966), 277 – 301.

[T94] EE Tyrtyshnikov. *How bad are Hankel matrices?* Numerische Mathematik, (1994) 67:261–269

[TKH83] M.Takizawa, Hisao Kishi and N.Hamada, *Synthesis of Lattice Digital Filter by the State Space Variable Method*, Trans. IECE Japan, vol. J65-A (1983), 363-370.