6-10-2016

Fast Algorithms for Structured Matrices and Laurent Polynomials

David Miller
University of Connecticut - Storrs, david.j.miller@uconn.edu

Fast Algorithms for Structured Matrices and Laurent Polynomials

David J. Miller, Ph.D.

University of Connecticut, 2016

ABSTRACT

The Vandermonde matrix and Cauchy matrix are classical and are encountered in polynomial and rational interpolation computation respectively. The structure of these matrices lead to fast inversion algorithms and system solvers. We look to extend these properties to other structured matrices, including Cauchy-Vandermonde matrices and systems involving Laurent polynomials.

Fast Algorithms for Structured Matrices and Laurent Polynomials

David J. Miller

M.S., University of Connecticut, Storrs, CT, 2012

B.A., Salve Regina University, Newport, RI, 2010

A Dissertation

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

at the

University of Connecticut

2016

APPROVAL PAGE

Doctor of Philosophy Dissertation

Fast Algorithms for Structured Matrices and Laurent Polynomials

Presented by

David J. Miller, B.A. Math. , M.S. Math.

Major Advisor _____
Vadim Olshevsky

Associate Advisor _____
Alexander Teplyaev

Associate Advisor _____
Dmitriy Leykekhman

University of Connecticut

2016

ACKNOWLEDGMENTS

I would like to begin by thanking my advisor, Professor Vadim Olshevsky. I arrived at the University of Connecticut without any direction or knowledge of what I wanted to study. The first class I took with Vadim, Numerical Analysis I, was the first of many classes that captured my full interest. Since then, Vadim's guidance and inspiration has formed my research career. For that, I will always be grateful.

I would like to thank Professors Alexander (Sasha) Teplyaev and Dmitriy Leykekhman for their time while on my dissertation committee. Also, thank you to Professors Reed Solomon, Keith Conrad and David Gross for their guidance and support during my job search.

I have had the pleasure of meeting and working with a number of incredible people at UConn. Thank you to all of my friends who have made these past several years special. Specifically, Monique Roy has guided me every step of the way, and I truly do not know where I would be without her support. Professor Amit Savkar has helped shape the teacher I am, and has spent countless hours working with me. His support is unmatched, and I am truly grateful to have Amit as a friend and colleague.

Finally, to my family, my mother Debbie, father Dave, stepfather Pete and stepmother Maria. You all have given unlimited love and support throughout my career at UConn. I am lucky and blessed to have all of you in my life. A final thank you to Krystin, Rocky and Thor for your love and support. I am excited to start the next of many chapters of our lives together.

Contents

Chapter 1

Introduction

This thesis consists of two distinct parts devoted to structured matrices and their uses in deriving fast algorithms. One theme that is consistent through both parts is the study of the Vandermonde matrix and the fast algorithms used to both invert, and solve a system involving it. The two parts are written as self-contained research papers, and thus could be read independently or out of order.

Many problems in signal procession, coding theory, system theory, orthogonal polynomials can be reduced in terms of matrices. Often, these matrices have structure inherited from the original problem. By exploiting this structure, it is often possible to attain more accurate results, using fewer operations than standard, structure-ignoring methods.

1.1 Relationship between matrices and polynomials

The relationship between polynomials and structured matrices is a well-studied topic. In the context of polynomial computations, typically matrices with Toeplitz, Hankel, Vandermonde

and related structures were of specific interest. Recently, a different class of quasiseparable matrices has been the focus of much research, and the problems resulting from them are quiet different than Toeplitz and Hankel matrices. We start by indicating the difference between the two types of problems.

1.1.1 Classical polynomial families and their moment matrices

Orthogonal polynomials are polynomials that are orthogonal with respect to an inner product $\langle \cdot, \cdot \rangle$. Real orthogonal polynomials are evaluated over a real interval $[a, b]$ of the form

$$\langle p(x), q(x) \rangle = \int_a^b p(x)q(x)w^2(x)dx, \tag{1.1.1}$$

where $w^2(x)$ is some weight function. These polynomials are classical, and they arise in problems in scientific computing such as numerical integration and solving differential equations. It is well-known that the *moment matrices* H corresponding to real orthogonal polynomials have Hankel structure, which is that H has has constant values along the antidiagonals, displayed below

$$H = \begin{bmatrix} h_0 & h_1 & h_2 & \dots & h_{n-1} \\ h_1 & h_2 & & \iddots & \vdots \\ h_2 & & \iddots & & h_{2n-3} \\ \vdots & \iddots & & h_{2n-3} & h_{2n-2} \\ h_{n-1} & \dots & h_{2n-3} & h_{2n-2} & h_{2n-1} \end{bmatrix}. \tag{1.1.2}$$

Similarly, applications in signal processing and system theory and control give rise to polynomials that are orthogonal with respect to an inner product $\langle \cdot, \cdot \rangle$ defined with integration on the unit circle,

$$\langle p(x), q(x) \rangle = \int_{-\pi}^{\pi} p\left(e^{i\theta}\right) \overline{q\left(e^{i\theta}\right)} w^2(\theta) d\theta, \tag{1.1.3}$$

where $w^2(\theta)$ is some weight function. These polynomials are called the *Szegö polynomials*, and the moment matrices T corresponding to them have Toeplitz structure (i.e., T has constant values along diagonals) displayed below

$$T = \begin{bmatrix} c_0 & c_{-1} & & \dots & c_{-n+1} \\ c_1 & c_0 & c_{-1} & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & c_{-1} \\ c_{n-1} & \dots & \dots & c_1 & c_0 \end{bmatrix}. \tag{1.1.4}$$

Moment matrices of these forms are shift-invarient, meaning that they have constant values along their antidiagonals and diagonals, respectively. This can be deduced from their corresponding inner products $\langle \cdot, \cdot \rangle$, along with the interpretation of the elements of a moment matrix $M = \begin{bmatrix} m_{kj} \end{bmatrix}$ as

$$\langle x^k, x^j \rangle. \tag{1.1.5}$$

For the real line case, it follows from (1.1.1) that

$$m_{kj} = \int_a^b x^{k+j} w^2(x) dx, \tag{1.1.6}$$

hence m_{kj} depends only on the sum of the row and column indices and thus M has Hankel structure. The structure of the Toeplitz matrix is deduced similarly from (1.1.3) and the

fact that on the unit circle we have

$$\overline{x^j} = \overline{e^{ij\theta}} = e^{-ij\theta} = x^{-j}, \tag{1.1.7}$$

and we see that the moments m_{kj} depend on the difference of indices. The shift-invariant structure of H and T means that these matrices are *structured*, and are defined by only $\mathcal{O}(n)$ parameters.

1.1.2 Classical polynomials ant their recurrence matrices

In addition to Hankel and Toeplitz matrices, there are other classes of matrices associated with real orthogonal and Szegö polynomials, called tridiagonal and unitary Hessenberg matrices. They are of the form

$$T = \begin{bmatrix} \delta_1 & \gamma_2 & 0 & \dots & 0 \\ \beta_2 & \delta_2 & \gamma_3 & \ddots & \vdots \\ 0 & \beta_3 & \delta_3 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \gamma_n \\ 0 & \dots & 0 & \beta_n & \delta_n \end{bmatrix} \tag{1.1.8}$$

and

$$U = \begin{bmatrix} -\rho_1\rho_0{}^* & -\rho_2\mu_1\rho_0{}^* & \dots & -\rho_n\mu_{n-1}\dots\mu_1\rho_0{}^* \\ \mu_1 & -\rho_1\rho_0{}^* & \dots & -\rho_n\mu_{n-1}\dots\mu_2\rho_1{}^* \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & -\rho_n\mu_{n-1}\rho_{n-2}{}^* \\ 0 & \dots & \mu_{n-1} & -\rho_n\rho_{n-1}{}^* \end{bmatrix} \tag{1.1.9}$$

where $\mu_{k,j} = \mu_k \mu_{k+1} \dots \mu_j$, respectively.

As we had for the Hankel and Toeplitz matrices (1.1.2) and (1.1.4), the $n \times n$ matrices (1.1.8) and (1.1.9) are *structured* in that they are defined by $\mathcal{O}(n)$ parameters. The difference, however, is how the entries are defined. Instead of being defined through an inner product, the entries of the tridiagonal and Unitary Hessenberg matrices are defined through the *recurrence relations* that define the polynomials. For example, real orthogonal polynomials satisfy the *three-term recurrence relation*

$$r_k(x) = (x - \delta_k) \cdot r_{k-1}(x) - \gamma_k^2 \cdot r_{k-2}(x), \tag{1.1.10}$$

and the terms $\{\delta_k, \gamma_k\}$ form the parameters in (1.1.8). Similarly, the terms $\{\rho_k, \mu_k\}$ that define the parameters of (1.1.9) are formed through the two-term recurrence relations satisfied by the Szegö polynomials,

$$\begin{bmatrix} \phi_k(x) \\ \phi_k^{\#}(x) \end{bmatrix} = \frac{1}{\mu_k} \begin{bmatrix} 1 & -\rho_k^* \\ -\rho_k & 1 \end{bmatrix} \begin{bmatrix} \phi_{k-1}(x) \\ x \cdot \phi_{k-1}^{\#}(x) \end{bmatrix}. \tag{1.1.11}$$

Perhaps unsurprisingly, these matrices are generalized as being *recurrence matrices* for their relationship with orthogonal polynomials. Recurrence matrices of this type are also referred to as *confederate matrices* (see, i.e., [3, 4]). Table 1.1.1 below gives known polynomial systems and their recurrence matrices.

1.2 Quasiseparable matrices and polynomials

Many nice results originally derived for the tridiagonal and Unitary Hessenberg matrices, such as fast multiplication algorithms and explicit inversion formulas (see, i.e., [2, 13, 22]).

TABLE 1.1.1: Systems of polynomials and corresponding recurrence relations.

Polynomial system	Confederate matrix
monomials $r_k(x) = x \cdot r_{k-1}(x)$	$\begin{bmatrix} 0 & 0 & \dots & 0 & 0 \\ 1 & \ddots & & 0 & 0 \\ 0 & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & \dots & 0 & 1 & 0 \end{bmatrix}$ lower shift matrix
Real orthogonal polynomials $r_k(x) = (x - \delta_k) \cdot r_{k-1}(x) - \gamma_k^2 \cdot r_{k-2}(x)$	$\begin{bmatrix} \delta_1 & \gamma_2 & 0 & \dots & 0 \\ \beta_2 & \delta_2 & \gamma_3 & \ddots & \vdots \\ 0 & \beta_3 & \delta_3 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \gamma_n \\ 0 & \dots & 0 & \beta_n & \delta_n \end{bmatrix}$ tridiagonal matrix
Szegö polynomials $\begin{bmatrix} \phi_k(x) \\ \phi_k^{\#}(x) \end{bmatrix} = \frac{1}{\mu_k} \begin{bmatrix} 1 & -\rho_k^* \\ -\rho_k & 1 \end{bmatrix} \begin{bmatrix} \phi_{k-1}(x) \\ x \cdot \phi_{k-1}^{\#}(x) \end{bmatrix}$	$\begin{bmatrix} -\rho_1{\rho_0}^* & -\rho_2\mu_1{\rho_0}^* & \dots & -\rho_n\mu_{n-1}\cdots\mu_1{\rho_0}^* \\ \mu_1 & -\rho_1{\rho_0}^* & \dots & -\rho_n\mu_{n-1}\cdots\mu_2{\rho_1}^* \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & -\rho_n\mu_{n-1}{\rho_{n-2}}^* \\ 0 & \dots & \mu_{n-1} & -\rho_n{\rho_{n-1}}^* \end{bmatrix}$ Unitary Hessenberg matrix

Recently, a superclass of matrices called *quasiseparable matrices* and their associated polynomials, *quasiseparable polynomials* was studied to generalize these results (see, i.e., [3, 4]). In this study, a relationship between polynomials and matrices was established.

Proposition 1.2.1. *Let $\mathcal{H}$ be the set of all strongly upper Hessenberg matrices ($a_{i+1,i} \neq 0$ for $i = 1, \dots, n-1$ and $a_{i,j} = 0$ for $i > j+1$), and $\mathcal{P}$ be the set of all polynomial systems $\{r_k(x)\}$ satisfying deg $r_k(x) = k$. For any strongly upper Hessenberg $n \times n$ matrix $A \in \mathcal{H}$, define the function f via the relation*

$$f(A) = P, \quad P = \{r_k(x)\}_{k=0}^{n}, \quad r_k(x) = \frac{1}{a_{2,1}a_{3,2}\dots a_{k,k-1}} \det(x \cdot I - A)_{k\times k}, \tag{1.2.1}$$

where $A = \left[a_{i,j}\right]$, *and* $A_{k\times k}$ *denotes the* $k \times k$ *leading submatrix of* A.

The previous proposition states that associated with each strongly upper Hessenberg matrix H is a polynomial system consisting of the characteristic polynomials of the principal submatrices of H. It is clear that this provides a mapping from matrices to polynomial systems; $f : \mathcal{H} \to \mathcal{P}$. It turns out that the matrices in Table 1.1.1 are special cases of a more general class of matrices, called *quasiseparable matrices*, defined next.

Definition 1.2.2 ($(H, 1)$-quasiseparable matrices)**.** A matrix $A = \left[a_{i,j}\right]$ is called $(H, 1)-$*quasiseparable* if

1. it is strongly upper Hessenberg (nonzero along first subdiagonal), and
2. $\max(\text{rank } A_{12}) = 1$, where the maximum is taken over all symmetric partitions of the form

$$A = \left[\begin{array}{c|c} * & A_{12} \\ \hline * & * \end{array}\right].$$

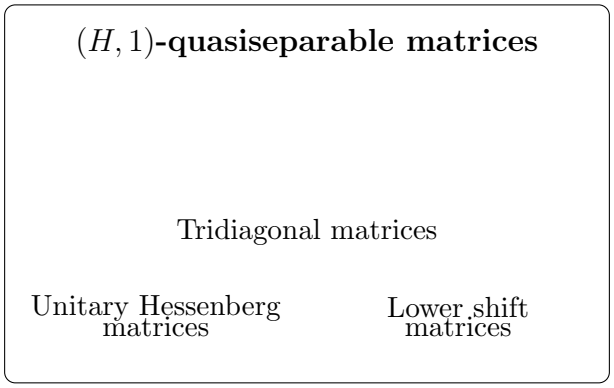
The class of $(H, 1)$-quasiseparable matrices includes the recurrence matrices associated with the monomials, Chebyshev polynomials, real orthogonal polynomials and Szegö polynomials (or all of the polynomials in Table 1.1.1) as special cases. This can be seen by investigating all of the matrices, and can be illustrated by Figure 1.2.1.

Proposition 1.2.3. *The lower shift matrix is* $(H, 0)$*-quasiseparable, Tridiagonal matrices are* $(H, 1)$*-quasiseparable and Unitary Hessenberg matrices are* $(H, 1)$*-quasiseparable.*

Proof. If A is the lower shift matrix, then any submatrix A_{12} is a zero matrix. If A is tridiagonal, then the submatrix A_{12} has the form $(\gamma_j)e_k e_1^T$, which has rank 1.

Finally, if A corresponds to the Szegö polynomials, then the submatrix A_{12} is also rank 1 since the rows are scalar multiples of each other. □

FIGURE 1.2.1: Quasiseparable matrices



A quasiseparable matrix is also uniquely defined by its *generator definition*.

Definition 1.2.4. A matrix A is called $(H,1)-$quasiseparable if it can be represented in the form

$$A = \begin{bmatrix} d_1 & g_1h_2 & g_1b_2h_3 & \dots & \dots & g_1b_1\dots b_{n-1}h_n \\ q_1 & d_2 & g_2h_3 & \dots & \dots & g_2b_2\dots b_{n-1}h_n \\ 0 & q_2 & d_3 & \dots & \dots & g_3b_3\dots b_{n-1}h_n \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & q_{n-2} & d_{n-1} & g_{n-1}h_n \\ 0 & \dots & \dots & 0 & q_{n-1} & d_n \end{bmatrix} \tag{1.2.2}$$

where $\{q_k \neq 0, d_k, g_k, b_k, h_k\}$ are called the *generators* of A.

Example 1.2.5. The lower shift matrix

$$Z_0 = \begin{bmatrix} 0 & 0 & \dots & 0 & 0 \\ 1 & \ddots & & 0 & 0 \\ 0 & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & \dots & 0 & 1 & 0 \end{bmatrix} \tag{1.2.3}$$

is $(H, 0)$-quasiseparable with generators $\{q_k = 1, d_k = g_k = b_k = h_k = 0\}$.

For brevity, a quaiseparable matrix will be depicted in the form

$$A = \begin{array}{|ccccc|}\hline d_1 & & & & g_i b_{ij}^{\times} h_j \\ q_1 & \ddots & & & \\ & \ddots & & & \\ 0 & & q_{n-1} & & d_n \\ \hline \end{array} \tag{1.2.4}$$

where $b_{ij}^{\times} = b_{i+1} \dots b_{j-1}$ for $j > i+1$ and $b_{ij}^{\times} = 1$ for $j = i+1$. Related to quasiseparable matrices are *quasiseparable polynomials* defined next.

Definition 1.2.6. Let $A = [a_{ij}]$ be an $(H, 1)-$ quasiseparable matrix. For $\alpha_i = \dfrac{1}{a_{i+1,i}}$, then the system of polynomials related to A via

$$r_k(x) = \alpha_1 \dots \alpha_k \det(xI - A)_{k \times k} \tag{1.2.5}$$

is called a system of $(H, 1)-$*quasiseparable polynomials*.

1.2.1 Quasiseparable polynomials

Many problems can be reduced to computations with systems of polynomials orthogonal to either the real-line or the unit circle in the complex plane. Algorithms involving these systems of polynomials often exploit the sparse recurrence relations satisfied by each class. As mentioned before, real-orthogonal polynomials are known to satisfy the three-term recurrence relations (1.1.10), and the Szegö polynomials satisfy the two-term recurrence relations (1.1.11).

Motivated by the computational savings enabled by the sparse recurrence relations, generalizations of these recurrence relations, in particular the general two-term recurrence relations

$$\begin{bmatrix} F_k(x) \\ r_k(x) \end{bmatrix} = \frac{1}{q_k} \begin{bmatrix} q_k b_k & -q_k g_k \\ h_k & x - d_k \end{bmatrix} \begin{bmatrix} F_{k-1}(x) \\ r_{k-1}(x) \end{bmatrix}, \tag{1.2.6}$$

where $\{F_k(x)\}$ are auxiliary polynomials, were formed.

As stated earlier, the relationship between real-orthogonal polynomials and Szegö polynomials and tridiagonal and unitary Hessenberg matrices is classical. A super class of matrices, quasiseparable matrices, was introduced to generalize these matrices. It was shown in [3] that the class of matrices related to systems of polynomials satisfying (1.2.6) is exactly the class of $(H, 1)-$quasiseparable matrices.

1.2.2 Digital filter structures

The well-known *Markel-Grey* filter design is an important result in signal processing, and is used to realize a system of Szegö polynomials via the two-term recurrence relations (1.1.11), which correspond to the ladder structure shown in Figure 1.2.2.

FIGURE 1.2.2: Markel-Grey filter structure: signal flow graph to realize the Szegö polynomials using two-term recurrence relations (1.1.11).

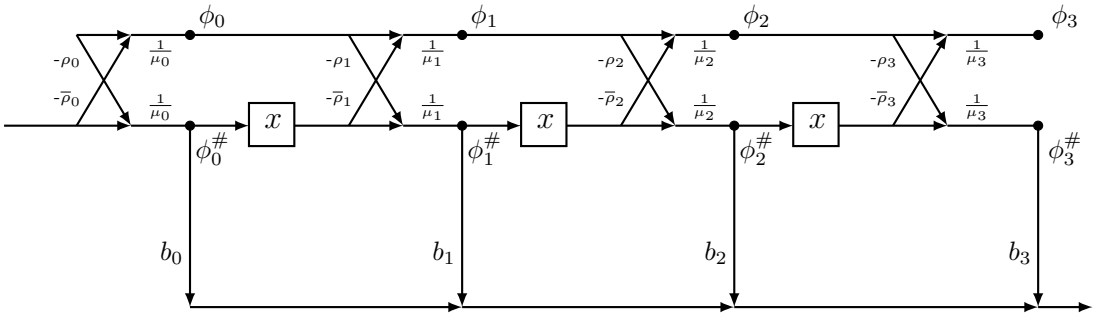
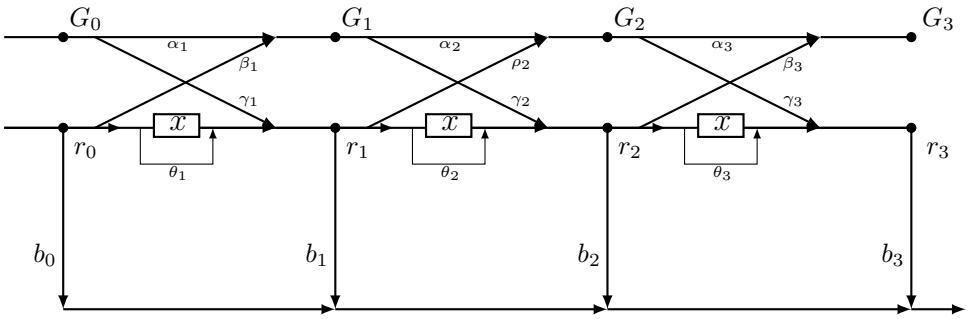


FIGURE 1.2.3: Quasiseparable filter structure: signal flow graph for polynomials R using the recurrence relations (1.2.6).



The recurrence relations (1.2.6) lead to the *quasiseparable filter structure*, shown in [9] and in Figure 1.2.3.

Chapter 2

Parts I and II: Laurent polynomials and the Cauchy-Vandermonde matrix

2.1 The Vandermonde matrix

Vandermonde matrices of the form

$$V(x_{1:n}) = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^n \\ 1 & x_2 & x_2^2 & \dots & x_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{bmatrix} \tag{2.1.1}$$

are classical, and explicit expression for their determinants and inverses are well known. The structure (2.1.1) can be exploited to speed-up computations involving $V(x)$, allowing one to design *fast algorithms*. These algorithms have complexity $\mathcal{O}(n^2)$, which is an order of magnitude less than that of standard, structure-ignoring methods. The following sections

explain the *Traub algorithm* and the *Björck-Pereyra algorithm*, which are an $\mathcal{O}(n^2)$ inversion algorithm and system-solver for $V(x)$ respectively.

2.1.1 The Traub algorithm for classical Vandermonde matrices

We first consider the numerical inversion of Vandermonde matrices of the form (2.1.1). Traub proposed the algorithm in [26], which is a fast method to compute all n^2 entries of $V(x)^{-1}$ in only $6n^2$ flops. Let

$$P(x) := \prod_{k=1}^{n}(x - x_k) = x^n + \sum_{k=0}^{n-1} a_k \cdot x^k. \tag{2.1.2}$$

be the *master polynomial*, whose zeros are the nodes of $V(x)$. Following [26], consider the divided difference

$$P[t, x] = \frac{P(t) - P(x)}{t - x}, \tag{2.1.3}$$

and define the polynomials $\{q_k(x)\}_{k=0}^{n-1}$ by

$$P[t, x] = \sum_{k=0}^{n-1} q_{n-k-1}(x) \cdot t^k. \tag{2.1.4}$$

The polynomials (2.1.4) are called the *associated* (or *Horner*) *polynomials* of $P(x)$. Substituting (2.1.2) into (2.1.3) shows that $P[t, x]$ has Hankel structure:

$$P[t, x] = \sum_{k=1}^{n} a_k \cdot \frac{t^k - x^k}{t - x} = \sum_{k=1}^{n} a_k \cdot \left(t^{k-1} + t^{k-2}x + \ldots + tx^{k-2} + x^{k-1}\right), \tag{2.1.5}$$

which implies that the associated polynomials are given by

$$q_k(x) = x^k + a_{n-1} \cdot x^{k-1} + \ldots + a_{n-k+1} \cdot x + a_{n-k}. \tag{2.1.6}$$

Equation (2.1.6) implies that the associated polynomials satisfy the recurrence relations

$$q_0(x) = 1, \quad q_k(x) = x \cdot q_{k-1}(x) + a_{n-k} \tag{2.1.7}$$

for $k = 1, \ldots, n-1$.

From (2.1.2), (2.1.3), (2.1.5) and the trivial identity

$$P[x, x] = P'(x), \tag{2.1.8}$$

we obtain what Traub called the *basic orthonormality relation*:

$$\frac{P[x_j, x_k]}{P'(x_k)} = \sum_{k=0}^{n-1} x_j^k \cdot \frac{q_{n-1-k}(x)}{P'(x_k)} = \delta_{jk}. \tag{2.1.9}$$

Equation (2.1.9) implies that the inverse of a Vandermonde matrix (2.1.1) is given by

$$V(x)^{-1} = \begin{bmatrix} q_{n-1}(x_1) & \ldots & q_{n-1}(x_n) \\ q_{n-2}(x_1) & \ldots & q_{n-2}(x_n) \\ \vdots & & \vdots \\ q_0(x_1) & \ldots & q_0(x_n) \end{bmatrix} diag\left(\left[\frac{1}{P'(x_k)} \right]_{k=1}^{n} \right). \tag{2.1.10}$$

Traub exploited formula (2.1.10) to derive a fast algorithm for inversion of Vandermonde matrices. First, observe that (2.1.7) allows one to compute the entries of the first factor in (2.1.10), while the second entries $P'(x)$ of the second factor are computed by

$$q_1'(x) = a_{n-1}, \quad q_k'(x) = q_{k-1}(x) + x \cdot q_{k-1}'(x), \tag{2.1.11}$$

which are obtained by differentiating (2.1.7). Thus the Traub algorithm can be summarized

as follows.

Algorithm 2.1.1 (The Traub algorithm)**. Input:** n distinct nodes, $\{x_k\}$. **Cost:** $6n^2$ flops. **Output:** Entries for $V(x)^{-1}$.

1. Compute the coefficients of $P(x)$ in (2.1.2) via nested polynomial multiplication:

$$\begin{bmatrix} a_0^{(1)} \\ a_1^{(1)} \end{bmatrix} = \begin{bmatrix} -x_1 \\ 1 \end{bmatrix}, \quad \begin{bmatrix} a_0^{(k)} \\ a_1^{(k)} \\ \vdots \\ a_k^{(k)} \end{bmatrix} = \begin{bmatrix} 0 \\ a_0^{(k-1)} \\ \vdots \\ a_{k-1}^{(k-1)} \end{bmatrix} - x_k \cdot \begin{bmatrix} a_0^{(k-1)} \\ \vdots \\ a_{k-1}^{(k-1)} \\ 0 \end{bmatrix}, \tag{2.1.12}$$

   with $a_j = a_j^{(n)}$.

2. For $j = 1, \ldots, n$,

   (a) Compute $q_k(x_j)$ via (2.1.7) for $k = 0, \ldots, n-1$.

   (b) Using these quantities, comput $P'(x_j)$ via (2.1.11).

   (c) Compute the j-th column $\left[\dfrac{q_k(x_j)}{P'(x_j)}\right]$ of $V(x)^{-1}$.

The Traub algorithm computes all n^2 entries of $V(x)$ in only $6n^2$ flops, which compares favorably with the complexity $\mathcal{O}(n^3)$ flops of structure-ignoring methods (i.e., Gaussian elimination). It was also noted that $P'(x_j)$ can be computed as

$$P'(x_j) = (x_j - x_1) \cdot \ldots \cdot (x_j - x_{j-1}) \cdot (x_j - x_{j+1}) \cdot \ldots \cdot (x_j - x_n), \tag{2.1.13}$$

which is used in other Traub-like algorithms in [3, 4].

2.1.2 The Björck-Pereyra algorithm

Björck and Pereyra described in [5] a fast algorithm that solves a Vandermonde linear system

$$V(x) \cdot a = f \tag{2.1.14}$$

in only $5n^2/2$ flops. Their algorithm exploits another explicit expression for $V(x)^{-1}$,

$$V(x)^{-1} = U_1^{-1} \cdot U_2^{-1} \cdot \ldots \cdot U_{n-1}^{-1} \cdot L_{n-1}^{-1} \cdot \ldots \cdot L_1^{-1}, \tag{2.1.15}$$

where

$$U_k^{-1} = \left[\begin{array}{c|ccc} I_{k-1} & & & \\ \hline & 1 & -x_k & \\ & & \ddots & \ddots \\ & & & 1 \end{array}\right], \tag{2.1.16}$$

$$L_k^{-1} = \left[\begin{array}{c|ccc} I_k & & & \\ \hline & \frac{1}{x_{k+1}-x_k} & & \\ & & \ddots & \\ & & & \frac{1}{x_n - x_{n-1}} \end{array}\right] \left[\begin{array}{c|ccc} I_{k-1} & & & \\ \hline & 1 & & \\ & -1 & 1 & \\ & & \ddots & \ddots \\ & & & -1 \quad 1 \end{array}\right], \tag{2.1.17}$$

This formula yields the following algorithm for solving a linear system (2.1.14):

Algorithm 2.1.2 (Björck-Pereyra algorithm)**. Input:** n distinct nodes, $\{x_k\}$ and entries of vector $f : f_k$. **Cost:** $5n^2/2$ flops. **Output:** Entries for $a : a_k$.

```
function [a]=BP(x,f)
```

```
a=f;
n=max(size(x));
for k=1:n-1
for i=n:-1:k+1
a(k)=(a(k)-a(k-1))/(x(k)-x(i-k-1));
end
end
for k=n-1:-1:1
for i=k:n-1
a(i)=a(i)-a(i+1)*x(k);
end
end
```

This algorithm solves one linear Vandermonde system of the form (2.1.14) in only $5n^2/2$ flops. It is known to provide, for special configurations of the nodes $\{x_k\}$, more accurate results than standard numerically stable algorithms.

2.1.3 Stability of the Björck-Pereyra algorithm

Björck and Pererya in [5] observed that their algorithm frequently produces more accurate solutions than could be expected from the condition number of the coefficient matrix. In [14] Higham analyzed Vandermonde matrices with positive and monotonically ordered nodes,

$$0 < x_1 < x_2 < \cdots < x_n, \tag{2.1.18}$$

and found that the Björck-Pereyra algorithm is guaranteed to compute a remarkably accurate solution $\widehat{a}$:

$$|\widehat{a} - a| \leq 5nu \cdot |a| + \mathcal{O}(u^2), \tag{2.1.19}$$

where a is the exact solution and u is the machine precision. It was further shown in [6] that the BP algorithm is also backward stable:

$$|\Delta V| \leq 12n^2 u V(x_{1:n}) + \mathcal{O}(u^2). \tag{2.1.20}$$

Here the computed solution $\widehat{a}$ is the exact solution of a nearby system $(V + \Delta V)\widehat{a} = f$.

2.1.4 Different ordering of nodes.

It is the experience of many that the numerical behavior of many algorithms depend on the ordering of the interpolation nodes. The orderings that are often considered are

- *Random ordering.*
- *Monotonic ordering* (2.1.18).
- *Leja ordering.* The points x_k are ordered so that

$$|x_1| = \max_{1 \leq k \leq n} |x_k|, \quad \prod_{j=1}^{k-1} |x_k - x_j| - \max_{k \leq l \leq n} \prod_{j=1}^{k-1} |x_l - x_j| \quad \text{for } 2 \leq k \leq n. \tag{2.1.21}$$

The latter ordering is related to Leja's work on interpolation (see [25]). In [16], Higham showed how to reorder x_k so that (2.3.15) holds in n^2 flops, so that incorporating Leja ordering will not slow down any fast $\mathcal{O}(n^2)$ algorithms. Higham also showed (2.3.15) mimics

row permutation of $V(x)$, which would be obtained by applying *partial pivoting* to $V(x)$. This fact is used later in generalizing this method of ordering of nodes for other matrix types.

2.2 Part I: Laurent polynomials

In this section we consider the problem of inverting Laurent polynomial-Vandermonde matrices, which is an extension of a subclass of Hessenberg-quaisiseparable-Vandermonde matrices. For a set of n distinct nodes $\{x_k\}_{k=1}^n$, the classical Vandermonde matrix $V(x) = \left[x_i^{j-1}\right]$ is known to be invertible, provided the nodes are distinct. One can generalize the structure by evaluating a basis other than the monomials at the nodes. That is, for a set of n polynomials $R = \{r_k(x)\}_{k=0}^{n-1}$ and n distinct nodes $\{x_k\}$, the matrix of the form

$$V_R(x) = \begin{bmatrix} r_0(x_1) & \dots & r_{n-1}(x_1) \\ r_0(x_2) & \dots & r_{n-1}(x_2) \\ \vdots & & \vdots \\ r_0(x_n) & \dots & r_{n-1}(x_n) \end{bmatrix} \tag{2.2.1}$$

is called a polynomial-Vandermonde matrix. In the simplest case where $M = \{1, x, \dots, x^{n-1}\}$ is the monomial basis, the matrix $V_M(x)$ reduces to a classical Vandermonde matrix and the inversion algorithm is due to Traub [26].

As in the monomial case, the desired inverse $V_R(x)^{-1}$ is given by the formula

$$V_R(x)^{-1} = \tilde{I} \cdot V_{\widehat{R}}^T(x) \cdot \mathrm{diag}(c_1, \dots, c_n), \tag{2.2.2}$$

with

$$c_i = \prod_{\substack{k=1 \\ k \neq i}}^{n} (x_k - x_i)^{-1}. \tag{2.2.3}$$

There has been much work on the inversion algorithm for different polynomial-Vandermonde matrices, which is depicted in Table 2.2.1. In this section, we are most concerned in the case where $G = \{\psi_k(x)\}_{k=0}^{n-1}$ is a system of Laurent polynomials, which we will define shortly.

TABLE 2.2.1: Fast $\mathcal{O}(n^2)$ algorithms for $V_R(x)$

Matrix $V_R(x)$	**System** R	$\mathcal{O}(n^2)$ **inversion**
Classic Vandermonde	Monomials	Traub [26]
Chebyshev-Vandermonde	Chebyshev Polynomials	Gohberg-Olshevsky [13]
Three-Term Vandermonde	Real Orthogonal Polynomials	Calvetti-Reichel [11]
Szegö - Vandermonde	Szegö Polynomials	Olshevsky [22]
$(H, 1)$-quasiseparable-V	$(H, 1)$-quasiseparable	BEGOT[1] [3]
(H, m)-quasiseparable-V	(H, m)-quasiseparable	BEGOTZ[2] [4]

2.2.1 Confederate matrices

We use Algorithm 2.1.10 as the motivation behind finding fast algorithms for various polynomial-Vandermonde matrices. We look at polynomial systems $R = \{r_0(x), \ldots, r_{n-1}(x), r_n(x)\}$ that are specified by the general $n-$term recurrence relations

$$r_k(x) = (\alpha_k \cdot x - a_{k-1,k}) \cdot r_{k-1}(x) - a_{k-2,k} \cdot r_{k-2}(x) - \ldots - a_{0,k} \cdot r_0(x), \quad \alpha_k \neq 0 \tag{2.2.4}$$

[1] Bella-Eidelman-Gohberg-Olshevsky-Tyrtyshnikov

[2] Bella-Eidelman-Gohberg-Olshevsky-Tyrtyshnikov-Zhlobich

for $k > 0$ and r_0 is a constant. It was first introduced in [19] that these polynomials are associated to a **confederate matrix** with respect to the basis R, defined by

$$C_R(r_n) = \begin{bmatrix} \frac{a_{0,1}}{\alpha_1} & \frac{a_{0,2}}{\alpha_2} & \frac{a_{0,3}}{\alpha_3} & \dots & \frac{a_{0,k}}{\alpha_k} & \dots & \dots & \frac{a_{0,n}}{\alpha_n} \\ \frac{1}{\alpha_1} & \frac{a_{1,2}}{\alpha_2} & \frac{a_{1,3}}{\alpha_3} & \dots & \frac{a_{1,k}}{\alpha_k} & \dots & \dots & \frac{a_{1,n}}{\alpha_n} \\ 0 & \frac{1}{\alpha_2} & \frac{a_{2,3}}{\alpha_3} & \dots & \vdots & \dots & \dots & \frac{a_{2,n}}{\alpha_n} \\ 0 & 0 & \frac{1}{\alpha_3} & \ddots & \frac{a_{k-2,k}}{\alpha_k} & \ddots & & \vdots \\ \vdots & \vdots & \ddots & \ddots & \frac{a_{k-1,k}}{\alpha_k} & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \frac{1}{\alpha_k} & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & \dots & \dots & 0 & \frac{1}{\alpha_{n-1}} & \frac{a_{n-1,n}}{\alpha_n} \end{bmatrix} \tag{2.2.5}$$

Notice the coefficients of the recurrence relations for the k^{th} polynomial $r_k(x)$ from (2.2.4) are contained in the k^{th} column of $C_R(r_n)$. We refer to [19] for many useful properties of the confederate matrix and only recall here that

$$r_0(x) = \lambda_0, r_k(x) = \lambda_0\lambda_1 \dots \lambda_k \det(xI - H_{k\times k}). \tag{2.2.6}$$

A table of well-known polynomial systems and their related Hessenberg matrices is given in [4].

In the classical Traub algorithm, we have the master polynomial (2.1.2), which is decomposed under the monomial basis $M = \{1, x, \dots, x^n\}$ with recurrence relation

$$r_k(x) = x \cdot r_{k-1}(x). \tag{2.2.7}$$

The confederate matrix for $P(x)$ with respect to the basis M is given by

$$C_M(P) = \begin{bmatrix} 0 & 0 & \dots & 0 & -P_0 \\ 1 & 0 & \dots & 0 & -P_1 \\ 0 & 1 & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & \dots & 0 & 1 & -P_{n-1} \end{bmatrix}. \tag{2.2.8}$$

The associated polynomials $\widehat{r}_k(x) = q_k(x)$ (2.1.6) have a well-known recurrence relation,

$$\widehat{r}_0(x) = 1, \quad \widehat{r}_k(x) = x \cdot \widehat{r}_{k-1}(x) + P_{n-k}, \tag{2.2.9}$$

which implies that the confederate matrix with respect to the basis $\widehat{M} = \{\widehat{r}_k(x)\}$ is given by

$$C_{\widehat{R}}(P) = \begin{bmatrix} -P_{n-1} & -P_{n-2} & \dots & -P_1 & -P_0 \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & \dots & 0 & 1 & 0 \end{bmatrix}. \tag{2.2.10}$$

The relation between confederate matrices $C_M(P)$ and $C_{\widehat{M}}(P)$ is given as

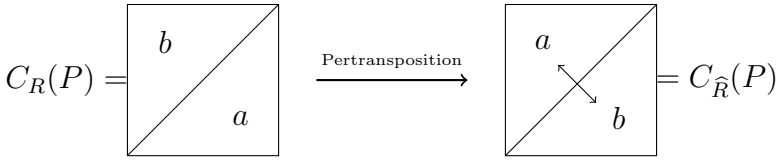
$$C_{\widehat{M}}(P) = \widetilde{I} \cdot C_M(P)^T \cdot \widetilde{I} \tag{2.2.11}$$

where

$$\widetilde{I} = \begin{bmatrix} 0 & \dots & 0 & 1 \\ \vdots & & 1 & 0 \\ 0 & \iddots & & \vdots \\ 1 & 0 & \dots & 0 \end{bmatrix} \tag{2.2.12}$$

is the antidiagonal matrix. The passage from $C_R(P)$ to $C_{\widehat{R}}(P)$ in (2.2.11) is called a *pertransposition*, or reflection across the antidiagonal. A visual representation of the pertransposition property is shown in Figure 2.2.1.

FIGURE 2.2.1: Pertransposition



2.2.2 Traub-like algorithm for quasiseparable polynomials

It was shown in [22] that the pertransposition property holds for any polynomial system, and in [4] it was shown that the recurrence relations for a given system of polynomials along with (2.2.11) allow fast evaluation of the polynomials $\widehat{R}$ at the nodes x_k, which is a required step in a fast Traub-like algorithm.

Consider the system $R = \{r_k(x)\}_{k=0}^n$ of $(H, 1)-$quasiseparable polynomials, which satisfy the recurrence relations (1.2.6). Now consider the nodes $\{x_k\}_{k=1}^n$ and write the master polynomial $P(x)$ with respect to the quasiseparable basis,

$$P(x) = \prod_{k=1}^{n}(x - x_k) = \sum_{k=0}^{n} P_k \cdot r_k(x).$$

Then the confederate matrix C of P with respect to R is

$$C_R(P) = \begin{bmatrix} d_1 & & g_i b_{ij}^{\times} h_j \\ q_1 & \ddots & \\ & \ddots & \\ 0 & q_{n-1} & d_n \end{bmatrix} - \frac{1}{P_n} \left[\begin{array}{c|c} & P_0 \\ 0 & \vdots \\ & P_{n-1} \end{array}\right]. \tag{2.2.13}$$

Applying *pertransposition* (2.2.11) gives us the confederate matrix for the associated polynomials $\widehat{R}$ as

$$C_{\widehat{R}}(P) = \begin{bmatrix} d_n & & g_{n-j} b_{n-j,n-i}^{\times} h_{n-1} \\ q_{n-1} & \ddots & \\ & \ddots & \\ 0 & q_1 & d_1 \end{bmatrix} - \frac{1}{P_n} \left[\begin{array}{ccc} P_{n-1} & \cdots & P_0 \\ \hline & 0 & \end{array}\right]. \tag{2.2.14}$$

Introducing the notation

$$\begin{aligned} \widehat{q}_k &= q_{n-k+1} \quad \widehat{d}_k = d_{n-k+1} \\ \widehat{g}_k &= h_{n-k+1} \quad \widehat{b}_k = b_{n-k+1} \quad \widehat{h}_k = g_{n-k+1} \end{aligned}$$

yields the [EGO05]-type recurrence relations

$$\begin{bmatrix} \widehat{F}_0(x) \\ \widehat{r}_0(x) \end{bmatrix} = \begin{bmatrix} 0 \\ P_n \end{bmatrix}, \quad \begin{bmatrix} \widehat{F}_k(x) \\ \widehat{r}_k(x) \end{bmatrix} = \underbrace{\frac{1}{\widehat{q}_k} \begin{bmatrix} \widehat{q}_k \widehat{b}_k & -\widehat{q}_k \widehat{g}_k \\ \widehat{h}_k & x - \widehat{d}_k \end{bmatrix} \begin{bmatrix} \widehat{F}_{k-1}(x) \\ \widehat{r}_{k-1}(x) \end{bmatrix}}_{\text{typical terms}} + \underbrace{\frac{1}{\widehat{q}_k} \begin{bmatrix} 0 \\ P_{n-k} \end{bmatrix}}_{\text{perturbation}} \tag{2.2.15}$$

where $\left\{\widehat{F}_k(x)\right\}$ are auxiliary polynomials.

Note that in order to use the recurrence relations (2.2.15) it is necessary to compute the coefficients of the master polynomial P_k. To this end, an efficient method of calculating these was shown in [3, 4], and is as follows.

Algorithm 2.2.1 (Coefficients of the master polynomial in the R basis)**. Input:** A quasiseparable confederate matrix $C_R(r_n)$ and n distinct nodes $\{x_k\}$.

1. Set $\begin{bmatrix} P_0^{(0)} & \dots & P_n^{(0)} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix}$
2. For $k = 1 : n$,

$$\begin{bmatrix} P_0^{(k)} \\ \vdots \\ P_n^{(k)} \end{bmatrix} = \left(\left[\begin{array}{ccc|c} & C_{\overline{R}}(x \cdot r_{n-1}(x)) & & 0 \\ \hline 0 & \dots & 0 \quad 1 & 0 \end{array}\right] - x_k \cdot I \right) \cdot \begin{bmatrix} P_0^{(k-1)} \\ \vdots \\ P_n^{(k-1)} \end{bmatrix}$$

   where $\overline{R} = \{r_0(x), \dots, r_{n-1}(x), x \cdot r_{n-1}(x)\}$.
3. Take $\begin{bmatrix} P_0 & \dots & P_n \end{bmatrix} = \begin{bmatrix} P_0^{(n)} & \dots & P_n^{(n)} \end{bmatrix}$

Now that we have an algorithm to compute the coefficients of the master polynomial, we can use them and the recurrence relations (2.2.15) to evaluate the associated polynomials at the nodes, using the following algorithm.

Algorithm 2.2.2 (Evaluate $\widehat{R}$ at $\{x_k\}$)**. Input:** Generators $\{p_k, q_k, d_k, g_k, b_k, h_k\}$ of a quasiseparable matrix $C_R(r_n)$ and n distinct nod es $\{x_k\}$.

1. Set $V_{\widehat{R}}(:, 1) = P_n \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}, \widehat{F_1} = \begin{bmatrix} 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{bmatrix}$

2. For $k = 1 : n-1$, compute

$$V_{\widehat{R}}(:, k+1) = \frac{1}{\widehat{q}_k} \left(\widehat{h}_k \widehat{F_k} + \left(\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} - \widehat{d}_k \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \right) V_{\widehat{R}}(:, k) + P_{n-k} \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \right)$$

and

$$\widehat{F}_{\widehat{R}}(:, k+1) = \frac{1}{\widehat{q}_k} (\widehat{q}_k \widehat{b}_k \widehat{F_k} - \widehat{q}_k \widehat{g}_k V_{\widehat{R}}(:, k)^T)$$

We are now in place to compute the inverse of a quasiseparable-Vandermonde matrix, $V_R(x)^{-1}$, using the following Traub-like algorithm.

Algorithm 2.2.3 (Traub-like inversion algorithm)**. Input:** Generators $\{p_k, q_k, d_k, g_k, b_k, h_k\}$ of a quasiseparable matrix $C_R(r_n)$ and n distinct nodes $\{x_k\}$.

1. Compute entries of $\text{diag}(c_1, \ldots, c_n)$, with $c_i = \prod_{\substack{k=1 \\ k \neq i}}^{n} (x_k - x_i)^{-1}$.
2. Compute the coefficients $\{P_k\}_{k=0}^n$ of the master polynomial $P(x)$ using Algorithm 2.2.1.
3. Evaluate the n polynomials of $\widehat{R}$ at the n nodes $\{x_k\}_{k=1}^n$ to form $V_{\widehat{R}}(x)$ using Algorithm 2.2.2.
4. Compute $V_R(x)^{-1} = \tilde{\text{I}} \cdot V_{\widehat{R}}^T(x) \cdot \text{diag}(c_1, \ldots, c_n)$.

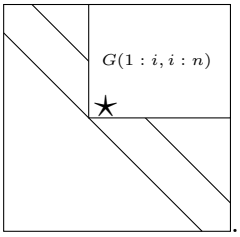
We will use the outline of Algorithm 2.2.3 when we look to invert a Laurent-Vandermonde system, defined shortly.

2.2.3 Main Tool: Green's matrices and Laurent Polynomials

There has been much discussion on fast inversion of polynomial-Vandermonde systems for various polynomial systems. Table 2.2.1 gives a representation of the different polynomial systems and the work done in those areas, with the most recent work concentrating on the inversion of $(H, 1)$-quasiseparable-Vandermonde matrices in [3] and (H, m)-quasiseparable-Vandermonde matrices in [4]. We use this previous work and look at a subclass of $(H, 1)$-quasiseparable matrices, called *Green's matrices*, and relate these matrices and their polynomials with Laurent polynomials. Green's matrices are first introduced in [23].

Definition 2.2.4. A strictly upper Hessenberg matrix G is called Green's $(H, 1)$-qs (or simply Green's matrix) if $\max_{1 \leq i \leq n} \text{rank } G(1 : i, i : n) = 1$.

The difference between $(H, 1)$-qs matrices and Green's matrices do not include the diagonal while submatrices $G(1 : i, i : n)$ do. Visually,

$G =$ 

.

It is discussed in [23] that a Green's matrix is $(H, 1)$-qs, which implies it has a generator definition. A strictly upper Hessenberg matrix G is Green's $(H, 1)$-qs if it can be represented in the form

TABLE 2.2.2: $(H,1)$-qs generators via Green's matrix $\widehat{G}$ generators

q_k	d_k	g_k	b_k	h_k
$\widehat{\sigma}_k$	$\tau_{k-1}\widehat{\tau}_k$	$\tau_{k-1}\widehat{\sigma}_k$	σ_k	$\widehat{\tau}_k$

$$G = \begin{bmatrix} \hat{\tau}_0\tau_1 & \hat{\tau}_0\sigma_1\tau_2 & \hat{\tau}_0\sigma_1\sigma_2\tau_3 & \dots & \dots & \hat{\tau}_0\sigma_1\dots\sigma_{n-1}\tau_n \\ \hat{\sigma_1} & \hat{\tau}_1\tau_2 & \hat{\tau}_1\sigma_2\tau_3 & \dots & \dots & \hat{\tau}_1\sigma_2\dots\sigma_{n-1}\tau_n \\ 0 & \hat{\sigma_2} & \hat{\tau}_2\tau_3 & \dots & \dots & \hat{\tau}_2\sigma_3\dots\sigma_{n-1}\tau_n \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \hat{\sigma}_{n-2} & \hat{\tau}_{n-2}\tau_{n-1} & \hat{\tau}_{n-2}\sigma_{n-1}\tau_n \\ 0 & \dots & \dots & 0 & \hat{\sigma}_{n-1} & \hat{\tau}_{n-1}\tau_n \end{bmatrix} \tag{2.2.16}$$

where $\{\sigma_k, \tau_k, \widehat{\sigma}_k \neq 0, \widehat{\tau}_k\}$ are called the generators of G. Table 2.2.2 gives the conversion formula from Green's generators to quasiseparable generators. A unique characteristic of a Green's matrix is that it can be decomposed as a product of n matrices, where a general $(H,1)$-quasiseparable matrix cannot.

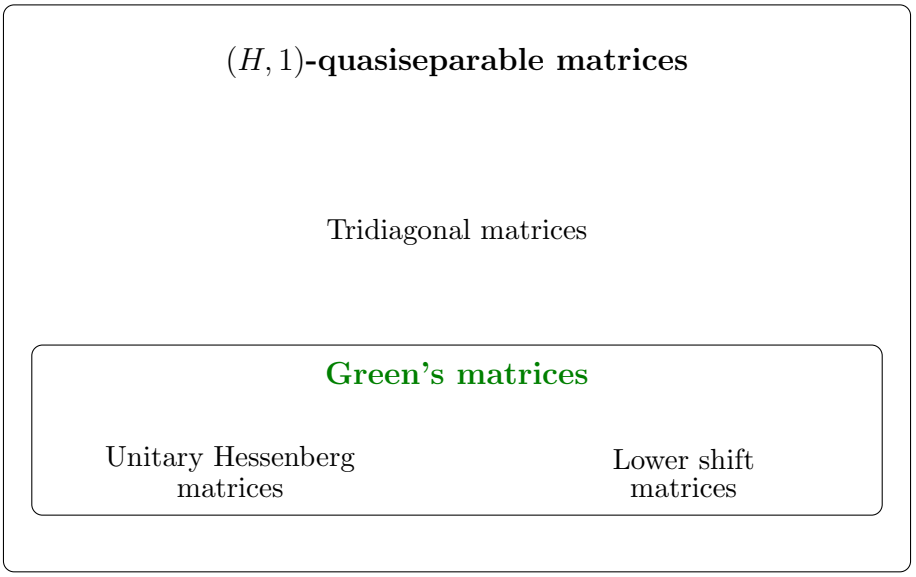
Definition 2.2.5. Let G be a Green's matrix. Then it has decomposition

$$G = \Theta_0\Theta_1\dots\Theta_n \tag{2.2.17}$$

where

$$\Theta_0 = \left[\begin{array}{c|c} \widehat{\tau}_0 & \\ \hline & I_{n-1} \end{array}\right], \Theta_k = \left[\begin{array}{c|cc|c} I_{k-1} & & & \\ \hline & \tau_k & \sigma_k & \\ & \widehat{\sigma}_k & \widehat{\tau}_k & \\ \hline & & & I_{n-k-1} \end{array}\right], \Theta_n = \left[\begin{array}{c|c} I_{n-1} & \\ \hline & \tau_n \end{array}\right]. \tag{2.2.18}$$

FIGURE 2.2.2: Green's matrices



Example 2.2.6 (Unitary Hessenberg decomposition)**.** Let U be the unitary Hessenberg matrix (1.1.9). It is well-known that it can be written as the product of Givens rotations, $U = \Gamma_0\Gamma_1 \dots \Gamma_n$ where

$$\Gamma_0 = \left[\begin{array}{c|c} \rho_0^* & \\ \hline & I_{n-1} \end{array}\right] \quad \Gamma_k = \left[\begin{array}{c|cc|c} I_{k-1} & & & \\ \hline & -\rho_k & \mu_k & \\ & \mu_k & -\rho_k^* & \\ \hline & & & I_{n-1} \end{array}\right], \quad \Gamma_n = \left[\begin{array}{c|c} I_{n-1} & \\ \hline & -\rho_n \end{array}\right]. \quad (2.2.19)$$

Figure 2.2.2 depicts the relationship between Green's matrices and Quasiseparable matrices.

Example 2.2.7 (Non-factorable $(H,1)-$qs matrix)**.** Consider the 3×3 tridiagonal matrix

$$A=\begin{bmatrix}1 & 1 & 0\\ 1 & 1 & 1\\ 0 & 1 & 1\end{bmatrix}.$$

Since A is tridiagonal, it is $(H,1)$-quasiseparable. Now assume it has the factorization

$$A=\begin{bmatrix}a & b & 0\\ c & d & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0\\ 0 & e & f\\ 0 & g & h\end{bmatrix}=\begin{bmatrix}a & bf & bg\\ c & df & dg\\ 0 & h & e\end{bmatrix}.$$

This leads to $bf=dg=1$ when $bg=0$, which is an inconsistent system of equations.

Definition 2.2.8. A system of polynomials $R=\{r_k(x)\}_{k=0}^{n}$ is called a system of *Green's polynomials* if it is related to a Green's matrix G via (2.2.6) with $\lambda_k=1/\widehat{\sigma}_k$.

It was shown in [23] that if a system of polynomials are Green's polynomials, then they satisfy the recurrence relation

$$\begin{bmatrix}f_0(x)\\ r_0(x)\end{bmatrix}=\begin{bmatrix}\widehat{\tau}_0\\ 1\end{bmatrix},\qquad \begin{bmatrix}f_k(x)\\ r_k(x)\end{bmatrix}=\frac{1}{\widehat{\sigma}_k}\begin{bmatrix}\widehat{\sigma}_k\sigma_k-\widehat{\tau}_k\tau_k & \widehat{\tau}_k\\ -\tau_k & 1\end{bmatrix}\begin{bmatrix}f_{k-1}(x)\\ x\cdot r_{k-1}(x)\end{bmatrix} \tag{2.2.20}$$

where $\{f_k(x)\}$ are auxiliary polynomials. We use Green's polynomials and the auxiliary $f_k(x)$s to define Laurent polynomials.

Definition 2.2.9. Let $J=(j_1,j_2,\ldots,j_n)$ be a sequence of binary digits. We define a sequence of **Laurent polynomials** $\Psi=\{\psi_k(x)\}_{k=0}^{n-1}$ as

$$\psi_k(x) = \begin{cases} x^{-\sum_{m=1}^{k+1} j_m} r_k(x) & \text{if } j_{k+1} = 0 \\ x^{-\sum_{m=1}^{k+1} j_m} f_k(x) & \text{if } j_{k+1} = 1 \end{cases} . \tag{2.2.21}$$

Definition 2.2.10. Let Ψ be a system of Laurent polynomials as in (3.5.5). Then $V_\Psi(x)$, given by

$$V_\Psi(x) = \begin{bmatrix} \psi_0(x_1) & \dots & \psi_{n-1}(x_1) \\ \psi_0(x_2) & \dots & \psi_{n-1}(x_2) \\ \vdots & & \vdots \\ \psi_0(x_n) & \dots & \psi_{n-1}(x_n) \end{bmatrix} \tag{2.2.22}$$

is called a Laurent-Vandermonde matrix.

2.3 Part II: Cauchy-Vandermonde matrices.

Cauchy-Vandermonde matrices. A Cauchy-Vandermonde matrix of order n is a matrix W of the form

$$W = [\ C \quad V\ ], \tag{2.3.1}$$

where the first ℓ $(1 \le \ell \le n)$ columns form a Cauchy matrix and the last $n-\ell$ columns form a Vandermonde matrix:

$$W(x_{1:n}, y_{1:\ell}) = \begin{bmatrix} \frac{1}{x_1 - y_1} & \dots & \frac{1}{x_1 - y_\ell} & 1 & x_1 & \dots & x_1^{n-\ell-1} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{x_n - y_1} & \dots & \frac{1}{x_n - y_\ell} & 1 & x_n & \dots & x_n^{n-\ell-1} \end{bmatrix} . \tag{2.3.2}$$

Cauchy-Vandermonde matrices are encountered in applied problems related to rational-

polynomial interpolation.

2.3.1 The Vandermonde and Cauchy matrices

Linear systems with Vandermonde matrices (2.1.1) and Cauchy matrices,

$$C(x_{1:n}, y_{1:n}) = \begin{bmatrix} \frac{1}{x_1-y_1} & \cdots & \frac{1}{x_1-y_n} \\ \vdots & \ddots & \vdots \\ \frac{1}{x_n-y_1} & \cdots & \frac{1}{x_n-y_n} \end{bmatrix} \tag{2.3.3}$$

are classical. They are encountered in many applied problems related to polynomial and rational function computations. Vandermonde and Cauchy matrices have many similar properties, among them is the existence of explicit formulas for their determinants and inverses, see, e.g., [7] and references therein. Along with many interesting algebraic properties, these matrices have several remarkable numerical properties, often producing more accurate computations than those based on the use of structure-ignoring algorithms, such as Gaussian Elimination with pivoting. At the same time, such favorable numerical properties are much better understood for Vandermonde and related matrices (see, for example [5, 10, 11, 14, 15, 16, 24].)

The Björck-Pereyra algorithm for Vandermonde systems. The Björck-Pereyra algorithm is based on the decomposition of the inverse of a Vandermonde matrix into a product of bidiagonal factors,

$$V^{-1}(x_{1:n}) = U_1^{-1} \cdot \ldots \cdot U_{n-1}^{-1} \cdot L_{n-1}^{-1} \cdot \ldots \cdot L_1^{-1}, \tag{2.3.4}$$

where the terms U_k^{-1} and L_k^{-1} are given in (2.1.17) and (2.1.16) respectively. This description allows one to solve the associated linear systems in only $\mathcal{O}(n^2)$ operations, which is by an

order of magnitude less than the complexity $\mathcal{O}(n^3)$ of general methods. Moreover, the algorithm requires only $\mathcal{O}(n)$ locations of memory.

2.3.2 BP-type algorithm for Cauchy matrices

The "bidiagonal" BKO Algorithm for general Cauchy systems. A fast Björck-Pereyra-type algorithm for Cauchy matrices was established in [7]. Again, it is based off the decomposition the inverse of a Cauchy matrix into a product of bidiagonal factors,

$$C^{-1}(x_{1:n}, y_{1:n}) = U_1^{-1} \cdot \ldots \cdot U_{n-1}^{-1} \cdot D^{-1} \cdot L_{n-1}^{-1} \cdot \ldots \cdot L_1^{-1} \tag{2.3.5}$$

This description again allows one to solve the associated linear systems in only $\mathcal{O}(n^2)$ operations. Moreover, the algorithm requires only $\mathcal{O}(n)$ locations of memory. Also in [7], it was shown that the sparsity of the L_k, U_k and D factors in (2.3.5) results in favorable upper bounds when computing forward stability of the algorithm.

2.3.3 Stability of BP-type algorithms

It was showin in [6] that the ordering of the nodes

$$y_n < \cdots < y_1 < 0 < x_1 < \cdots < x_n \tag{2.3.6}$$

is an appropriate analog of (2.1.18) for Cauchy matrices. This allows for the computation of the error bounds for the BP-type algorithm are similar to (2.1.19) and (2.1.20),

$$|a - \widehat{a}| \leq 5(2n+1)u \cdot |a| + \mathcal{O}(n^2), \tag{2.3.7}$$

$$|\Delta C| \leq 20n(2n-1)u \cdot C(x_{1:n}, y_{1:n}) + \mathcal{O}(n^2). \tag{2.3.8}$$

An interesting result is that the conditions (2.1.18) and (2.3.6) imply *total positivity* of $V(x_{1:n})$ and $C(x_{1:n}, y_{1:n})$ respectively. Totally positive matrices are usually extremely ill-conditioned, so that the Gaussian elimination procedure often fails to compute even one correct digit in the computed solution. The bounds (2.3.7) and (2.3.8) indicate that also in such cases the BP-type algorithm can produce, for special right hand sides, all possible results with relative precision.

Limitations for non-totally positive matrices. Reordering of the nodes $\{x_k\}$ and poles $\{y_k\}$ is equivalent to row and column permutation of $C(x_{1:n}, y_{1:n})$ respectively. If the two sets are separated from each other,

$$y_k < x_j, \quad 1 \leq k, j \leq n \tag{2.3.9}$$

the error bound (2.3.7) suggests we reorder the nodes monotonically as in (2.3.6) and apply the BP-type algorithm in [6]. However, numerical experiments show that in the general case (i.e. when (2.3.6) does not hold) the backward error of the fast BP-type algorithm of [6] may be worse that that of the slow Gaussian elimination with pivoting.

An examination of the error analysis of the BP-type algorithm in [6] indicates that the corresponding backward bound involves the quantity

$$|L_1| \cdot \ldots \cdot |L_{n-1}| \cdot |U_{n-1}| \cdot \ldots \cdot |U_1|, \tag{2.3.10}$$

which can be much higher than the more attractive quantity $|L| \cdot |U|$ because of a non-cancellation property $|M| \cdot |N| \geq |M \cdot N|$. In totally positive cases, the entries of L_k and U_k in (2.3.10) are nonnegative, allowing us to remove the moduli to replace (2.3.10) with

$C(x_{1:n}, y_{1:n})$ in the favorable bound (2.3.7). Unfortunately, the bidiagonal structure of the L_k and U_k in (2.3.5) does not allow us to remove the moduli. These limitations suggests a need for a new backward stable Cauchy solver that may be used to develop new algorithms with fewer restrictions on the ordering of the nodes.

2.3.4 "Lambda" BKO algorithm for Cauchy matrices.

In [8], a new LU-factorization of Cauchy matrices was introduced. This new algorithm is similar to the Björck-Pereyra algorithm, but used a "lambda" shape matrix rather than the bidiagonal factorization. The Cauchy matrix was factored as

$$C(x_{1:n}, y_{1:n})^{-1} = U_1^{-1} \cdot \ldots \cdot U_{n-1}^{-1} \cdot D^{-1} \cdot L_{n-1}^{-1} \cdot \ldots \cdot L_1^{-1}, \tag{2.3.11}$$

where

$$L_k^{-1} = \left[\begin{array}{c|ccc} I_k & & & \\ \hline & \frac{1}{x_{k+1}-x_k} & & \\ & & \ddots & \\ & & & \frac{1}{x_n - x_k} \end{array}\right] \left[\begin{array}{c|ccc} I_{k-1} & & & \\ \hline & 1 & & \\ & -(x_k - y_k) & (x_{k+1} - y_k) & \\ & \vdots & & \ddots \\ & -(x_k - y_k) & & (x_n - y_k) \end{array}\right], \tag{2.3.12}$$

$$U_k^{-1} = \left[\begin{array}{c|ccccc} I_{k-1} & & & & \\ \hline & 1 & -(x_k - y_k) & \ldots & -(x_k - y_k) \\ & & (x_k - y_{k+1}) & & \\ & & & \ddots & \\ & & & & (x_k - y_n) \end{array}\right] \left[\begin{array}{c|ccc} I_k & & & \\ \hline & \frac{1}{y_k - y_{k+1}} & & \\ & & \ddots & \\ & & & \frac{1}{y_k - y_n} \end{array}\right] \tag{2.3.13}$$

and

$$D^{-1} = diag\{(x_1 - y_1), \ldots, (x_n - y_n)\}. \tag{2.3.14}$$

2.3.5 Leja ordering for Vandermonde and Cauchy matrices.

It is important to note that the BKO algorithm is not invariant to permutations of the points defining the Cauchy matrix. Different configurations of $\{x_{1:n}\}$ yield different decompositions for $V(x_{1:n})$ and $C(x_{1:n}, y_{1:n})$, though all of the form (2.3.4) and (2.3.11) respectively. In [16], Higham proposed a reordering of the nodes for Vandermonde matrices:

$$|x_1| = \max_{1\le j\le n} |x_j|, \prod_{i=1}^{k-1} |x_k - x_i| = \max_{k\le j\le n} \prod_{i=1}^{k-1} |x_j - x_i| \text{ for } 2 \le k \le n, \tag{2.3.15}$$

and in [25] it was called *Leja ordering*. We suggest to call it *Leja-Vandermonde ordering* as to distinguish it from orderings associated with other matrices.

Through numerical examples in [7], it was shown that Leja-Vandermonde ordering of the nodes (2.3.15) improves the backward stability of the BKO algorithm. This implies the need to investigate the affects of Leja-type ordering on all BKO-type algorithms. The ordering (2.3.15) mimics the row interchange that would occur when Gaussian elimination with partial pivoting is applied. In addition to (2.3.15), a similar pivoting method for Cauchy matrices was introduced in [8]. Here, it is implied that partial pivoting on a Cauchy matrix, $C(x_{1:n}, y_{1:n})$, is equivalent to successive maximization of the quantities

$$|d_i| = \left| \frac{\prod_{j=1}^{i-1}(x_i - x_j)\prod_{j=1}^{i-1}(y_i - y_j)}{(x_i - y_i)\prod_{j=1}^{i-1}(x_i - y_j)\prod_{j=1}^{i-1}(x_j - y_i)} \right|, \tag{2.3.16}$$

for $i = 1, \ldots, n$. In [8] this ordering is called *predictive partial pivoting*, or *rational Leja ordering* by analogy with Leja ordering of [16]. We suggest to call it *Leja-Cauchy ordering*.

TABLE 2.3.1: Leja Ordering

Matrix	**Leja-type ordering**
Vandermonde	Higham [16], Reichel [25]
Cauchy	BKO [8]
Cauchy-Vandermonde	**??**

2.3.6 Previous work on Cauchy-Vandermonde matrices.

In the simplest classical Cauchy case (2.3.3), the lambda-factorization is, in many instances, superior to the bidiagonal one. This indicates the need to derive the lambda-shaped factorization for general Cauchy-Vandermonde matrices as well as its numerical properties.

In [20], a decomposition of the inverse of Cauchy-Vandermonde matrices into bidiagonal factors was established, however the lambda factorization was not derived, and the corresponding analog of Leja ordering was not explored. In [21], a fast, BP-type algorithm is introduced for solving a Cauchy-Vandermonde system. In addition, it was proven in [21] that $W(x_{1:n}, y_{1:\ell})$ is *totally positive* when

$$y_\ell < \cdots < y_1 < 0 < x_1 < \cdots < x_n. \tag{2.3.17}$$

Finally, a numerical example is given to indicate the stability of the algorithm, however the affect of the ordering of the nodes was not shown. Tables 2.3.1 and 2.3.2 summarize the known results for Leja ordering and $\mathcal{O}(n^2)$ system solvers respectively.

TABLE 2.3.2: Fast $\mathcal{O}(n^2)$ System Solvers

Matrix	**Fast System Solver Bidiagonal**	**Fast System Solver "Lambda"**
Vandermonde	Björck and Pereyra [5]	Boros, Kailath, Olshevsky[6]
Cauchy	Boros, Kailath, Olshevsky [7]	Boros, Kailath, Olshevsky [8]
C-V	Martinez and Peña [21]	**??**

2.4 Main Results

2.4.1 Laurent polynomials

In this section, we look to invert a Laurent-Vandermonde matrix, $V_\Psi(x)$, using a Traub-like algorithm. To do so, we will:

- Establish a Traub-like inversion algorithm for a Vandermonde matrix with respect to a Power system:

$$P_{[-m,n]} = \{x^{-m}, \ldots, x^{-1}, 1, x, \ldots, x^n\}. \tag{2.4.1}$$

- Introduce the concept of an **admissible system**, and find a general relation between their multiplication operators (a subclass of confederate matrices) and that of the associated polynomials.

- Use the findings for the admissible system to establish the **pertransposition** property for associated-Laurent polynomials.

- Use the recurrence relations found in the previous step to create $\widehat{\Psi} = \left\{\widehat{\psi}_0(x), \ldots, \widehat{\psi}_n(x)\right\}$. Using these polynomials, we have the equation for the inverse:

$$V_{\Psi}^{-1}(x) = \widehat{I} \cdot V_{\widehat{\Psi}}^{T}(x) \cdot diag(c_1, \ldots, c_n). \tag{2.4.2}$$

2.4.2 Cauchy-Vandermonde matrices

New "lambda" algorithm. We derive an alternative to the BKO-type algorithm based on the factorization

$$W^{-1}(x_{1:n}, y_{1:\ell}) = U_1^{-1} \cdot \ldots \cdot U_{n-1}^{-1} \cdot D^{-1} \cdot L_{n-1}^{-1} \cdot \ldots \cdot L_1^{-1} \tag{2.4.3}$$

where the L factors have the "lambda" shape as in (2.3.13). The new algorithm is provided to facilitate the computation of the upper bounds when performing backward error analysis.

CV-Leja ordering of nodes. After deriving the BKO-type algorithm, we use the well known equation for the determinant of a Cauchy-Vandermonde matrix to maximize the $det(W_{1:k,1:k})$ by maximizing the quantities

$$|d_i| = \begin{cases} \left| \dfrac{\prod_{j=1}^{i-1}(x_i - x_j)\prod_{j=1}^{i-1}(y_i - y_j)}{(x_i - y_i)\prod_{j=1}^{i-1}(x_i - y_j)\prod_{j=1}^{i-1}(x_j - y_i)} \right|, & \text{for } i = 1, \ldots, \ell \\ \\ \left| \dfrac{\prod_{j=1}^{i-1}(x_i - x_j)}{\prod_{j=1}^{\ell}(x_i - y_j)} \right|, & \text{for } i = \ell + 1, \ldots, n-1. \end{cases} \tag{2.4.4}$$

Error analysis. After deriving the BKO-type algorithm for Cauchy-Vandermonde matrices, we provide nice upper bounds on the unit roundoff error for the computed solution $\hat{a}$.

Numerical experiments. Finally, we present several numerical experiments on the

algorithm. In these experiments, we test different orderings of the nodes, including monotonic and CV-Leja, to see their affect on the stability of the algorithm.

Chapter 3

Fast Inversion Algorithm for Laurent-Vandermonde Matrices

3.1 Power basis

To study an inversion algorithm for Laurent polynomials, we must first look at some results for a simpler case. First, let us define the space of polynomials and the basis we are working in.

Definition 3.1.1. Let $\Pi_{[-m,n]} = span\left(x^{-m}, x^{-m+1}, \ldots, x^{-1}, 1, x, \ldots, x^{n}\right)$ be the **power space** of degree $(-m, n)$, and the set

$$P_{[-m,n]} = \left\{x^{-m}, x^{-m+1}, \ldots, x^{-1}, 1, x, \ldots, x^{n}\right\} \tag{3.1.1}$$

be the **power basis** of $\Pi_{-m,n}$.

We can use this set of polynomials to create a matrix similar to the classic Vandermonde

matrix, called the **Power-Vandermonde matrix**.

Definition 3.1.2. Let $P := P_{[-m,n-1]}$ be the power basis of $\Pi_{[-m,n]}$ as in (3.1.1) and $\{x_k\}$ be a set of distinct nodes for $k = 1, \ldots, n+m$ with $x_k \neq 0$ for all k. Then the **power-Vandermonde matrix**, $V_P(x_{1:n+m})$, is defined as

$$V_P(x_{1:n+m}) = \begin{bmatrix} x_1^{-m} & \cdots & x_1^{-1} & 1 & x_1 & \cdots & x_1^{n-1} \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ x_{n+m}^{-m} & \cdots & x_{n+m}^{-1} & 1 & x_{n+m} & \cdots & x_{n+m}^{n-1} \end{bmatrix}. \tag{3.1.2}$$

3.1.1 Traub-like algorithm for the power-Vandermonde matrix

We now look to invert (3.1.2) using a Traub-like algorithm. The Traub algorithm is referenced in equation (2.1.10), and any Traub-like algorithm begins by constructing the *master polynomial*, which is used to define the *associated polynomials*, $\{\widehat{r}_k(x)\}$.

Definition 3.1.3. Let $P_{[-m,n]}$ be the power basis (3.1.1) and $\{x_k\}$ be a set of $n+m$ distinct nodes with $x_k \neq 0$. Define the **master polynomial** as

$$P(x) = \prod_{k=1}^{m} \left(\frac{1}{x} - \frac{1}{x_k} \right) \cdot \prod_{k=m+1}^{n} (x - x_k) = \sum_{k=-m}^{n} P_k \cdot x^k. \tag{3.1.3}$$

We use the master polynomial (3.1.3) in the computation of the divided difference

$$Q[t, x] = \frac{P(t) - P(x)}{t - x} \tag{3.1.4}$$

and the associated polynomials as follows.

Definition 3.1.4. Let the master polynomial $P(x)$ be as in (3.1.3) and consider the divided

difference (3.1.4). Define the **associated polynomials**, $\{\widehat{r}_k(x)\}$ as

$$Q[t,x] = \sum_{k=-m}^{-1} \widehat{r}_k(x) \cdot t^{-m-k-1} + \sum_{k=0}^{n-1} \widehat{r}_k(x) \cdot t^{n-k-1}. \tag{3.1.5}$$

We can now obtain what Traub called the basic orthonormality relation.

Lemma 3.1.5. *Let $\{x_k\}$ be a set of $n+m$ distinct nodes with $x_k \neq 0$, the master polynomial $P(x)$ as in (3.1.3) and the divided difference $Q[t,x]$ be as in (3.1.4). Then*

$$\frac{Q[x_j,x_k]}{P'(x_k)} = \sum_{k=-m}^{-1} x_j^k \cdot \frac{\widehat{r}_{-m-k-1}(x_k)}{P'(x_k)} + \sum_{k=0}^{n-1} x_j^k \cdot \frac{\widehat{r}_{n-k-1}(x_k)}{P'(x_k)} = \delta_{j,k} \tag{3.1.6}$$

where

$$\delta_{i,j} = \begin{cases} 0 & \text{for } k \neq j \\ 1 & \text{for } k = j \end{cases} \tag{3.1.7}$$

Proof. Straightforward from (3.1.3), (3.1.4), (3.1.5) and the trivial identity

$$Q[x,x] = P'(x). \tag{3.1.8}$$

□

The latter relation implies that the inverse of the power-Vandermonde matrix (3.1.2) is given by

$$V_P(x_{1:n+m})^{-1} = \begin{bmatrix} \widehat{r}_{-1}(x_1) & \widehat{r}_{-1}(x_2) & \dots & \widehat{r}_{-1}(x_{n+m}) \\ \vdots & \vdots & & \vdots \\ \widehat{r}_{-m}(x_1) & \widehat{r}_{-m}(x_2) & \dots & \widehat{r}_{-m}(x_{n+m}) \\ \widehat{r}_{n-1}(x_1) & \widehat{r}_{n-1}(x_2) & \dots & \widehat{r}_{n-1}(x_{n+m}) \\ \vdots & \vdots & & \vdots \\ \widehat{r}_0(x_1) & \widehat{r}_0(x_2) & \dots & \widehat{r}_0(x_{n+m}) \end{bmatrix} \cdot diag\left(\left[\frac{1}{P'(x_k)} \right]_{k=1}^{n+m} \right) \tag{3.1.9}$$

The equation (3.1.9) gives a relation between the inverse of a power-Vandermonde matrix and the associated polynomials defined in (3.1.5). This method of computing (3.1.9) is known to be efficient when the structure of the associated polynomials $\{\widehat{r}_k(x)\}$ is exploited.

3.1.2 Recurrence relations for associated polynomials

The ability to construct the associated polynomials $\{\widehat{r}_k(x)\}$ efficiently will allow the fast inversion of a power-Vandermonde matrix. It is for this reason that we look at how to build the associated polynomials using their recurrence relations.

Lemma 3.1.6. *Let $\{x_k\}$ be a set of $n+m$ distinct nodes with $x_k \neq 0$ and the master polynomial $P(x)$ as in (3.1.3). Then associated polynomials (3.1.5) exist and satisfy*

$$\widehat{r}_0(x) = P_n, \quad \widehat{r}_k(x) = x \cdot \widehat{r}_{k-1}(x) + P_{n-k}, \tag{3.1.10}$$

for $k = 1, \dots, n-1$ and

$$\widehat{r}_{-1}(x) = -P_{-m} \cdot x^{-1}, \quad \widehat{r}_{-k}(x) = x^{-1} \cdot (\widehat{r}_{-k+1}(x) - P_{-m+k-1}), \tag{3.1.11}$$

for $k = 2, \ldots, m$.

Proof. The proof is immediate after plugging (3.1.3) into (3.1.4):

$$\begin{aligned} Q[t,x] &= \frac{P(t)-P(x)}{t-x} = \sum_{k=1}^{n} P_k \frac{t^k - x^k}{t-x} - \sum_{k=1}^{m} P_{-k} \frac{t^k - x^k}{t^k x^k (t-x)} && (3.1.12) \\ &= \sum_{k=1}^{n-1} P_k (t^{k-1} + t^{k-2}\cdot x + \ldots + t \cdot x^{k-2} + x^{k-1}) && (3.1.13) \\ &- \sum_{k=1}^{m} P_{-k} (t^{-1}\cdot x^{-k} + t^{-2}\cdot x^{-k+1} + \ldots + t^{-k+1}\cdot x^{-2} + t^{-k}\cdot x^{-1}). \end{aligned}$$

Equation (3.1.13) implies that the associated polynomials are given by

$$\widehat{r}_k(x) = x^k + P_{n-1}\cdot x^{k-1} + \ldots + P_{n-k+1}\cdot x + P_{n-k}, \tag{3.1.14}$$

for $k = 1, \ldots, n-1$ and

$$\widehat{r}_{-k}(x) = -P_{-m}x^{-k} - P_{-m+1}\cdot x^{-k+1} - \ldots - P_{-m+k-1}\cdot x^{-1}, \tag{3.1.15}$$

for $k = 1, \ldots, m$. Equations (3.1.14) and (3.1.15) imply the result. □

3.2 Multiplication operator

As discussed earlier, we can define a polynomial system, $R = \{r_k(x)\}_{k=0}^{n}$ and its recurrence relations by its associated confederate matrix, $C_R(r_n)$. One property of any confederate matrix is that it satisfies

$$x \cdot \begin{bmatrix} r_0(x) & \ldots & r_{n-1}(x) \end{bmatrix} = \begin{bmatrix} r_0(x) & \ldots & r_{n-1}(x) \end{bmatrix} C_R(r_n). \tag{3.2.1}$$

Example 3.2.1. For the monomial basis $R = \{x^k\}$ for $k = 0, \ldots, n-1$, the confederate matrix is the standard lower shift matrix

$$C_R(x^n) = \begin{bmatrix} 0 & 0 & \ldots & \ldots & 0 \\ 1 & 0 & \ldots & \ldots & 0 \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & 1 & 0 \end{bmatrix}. \tag{3.2.2}$$

The monomial basis clearly satisfies the recurrence relations $x^k = x \cdot x^{k-1}$, and thus satisfies

$$x \cdot \begin{bmatrix} 1 & x & \ldots & x^{n-1} \end{bmatrix} = \begin{bmatrix} 1 & x & \ldots & x^{n-1} \end{bmatrix} \begin{bmatrix} 0 & 0 & \ldots & \ldots & 0 \\ 1 & 0 & \ldots & \ldots & 0 \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & 1 & 0 \end{bmatrix}. \tag{3.2.3}$$

The confederate matrices allow the definition of polynomial systems through their recurrence relations. However, if we are not looking at strictly polynomial systems, then we cannot use a confederate matrix to define them. Instead, we will use the property (3.2.4), called a *multiplication operator.*

Definition 3.2.2. For a system of polynomials R, the matrix C_R is called a **multiplication operator** if it satisfies

$$x \cdot \begin{bmatrix} r_0(x) & r_1(x) & r_2(x) & \ldots \end{bmatrix} = \begin{bmatrix} r_0(x) & r_1(x) & r_2(x) & \ldots \end{bmatrix} C_R. \tag{3.2.4}$$

Example 3.2.3. For the infinite monomial basis $R = \{x^k\}$ for $k = 0, 1, \ldots$, the multiplication operator is the infinite-dimensional lower-shift matrix:

$$C_R = \begin{bmatrix} 0 & 0 & \ldots & \ldots \\ 1 & 0 & \ldots & \ldots \\ 0 & 1 & \ddots & \\ \vdots & & \ddots & \ddots \end{bmatrix}. \tag{3.2.5}$$

The concept of a multiplication operator is useful because we can extend the confederate matrix property of capturing recurrence relations to systems that are not directly related to a confederate matrix via (2.2.6). The next example shows how this can be achieved.

Example 3.2.4. Let $P_{[-m]}$ be the infinite power basis $\{x^{-m}, \ldots, x^{-1}, 1, x, x^2, \ldots\}$. One property that is immediate is that they satisfy the same recurrence relations as the monomial basis:

$$x^k = x \cdot x^{k-1} \tag{3.2.6}$$

for all $k = -m, \ldots, -1, 0, 1, \ldots, n$. Thus the multiplication operator, $C_{P_{[-m]}}$ is the same as C_R defined in (3.2.5).

3.2.1 Truncated multiplication operator

When looking at a polynomial-Vandermonde matrix, we use a set of n polynomials (or $m+n$ in the power basis case) instead of an infinite set. For this reason, we must look to *truncate* the indefinite multiplication operators to finite dimension.

Definition 3.2.5. Let $R = \{r_k(x)\}$ be a system of polynomials and C_R be its associated multiplication operator. Then if λ is a root of $r_n(x)$ and $C_R(r_n)$ is of size n, then it is called

a **truncated multiplication operator**, and satisfies

$$\lambda \cdot \begin{bmatrix} r_0(\lambda) & \dots & r_{n-1}(\lambda) \end{bmatrix} = \begin{bmatrix} r_0(\lambda) & \dots & r_{n-1}(\lambda) \end{bmatrix} C_R(r_n) \tag{3.2.7}$$

By definition, λ is an eigenvalue of $C_R(r_n)$ and we have the vector of polynomials evaluated at λ is a left-eigenvector. Truncating the multiplication operator is useful when we look to apply our Traub-like algorithm to a polynomial-Vandermonde matrix.

Lemma 3.2.6 (Truncated multiplication operators.)**.** *Let $R = \{r_k(x)\}$ be an infinite set of polynomials and C_R be its multiplication operator. Let $\{x_k\}$ for $k = 1, \dots, n$ be n distinct nodes and*

$$P(x) = \prod (x - x_k) = \sum_{k=0}^{n-1} P_k \cdot r_x(x) + P_n \cdot x \cdot r_{n-1}(x) \tag{3.2.8}$$

be the master polynomial. Then

$$x_k \cdot \begin{bmatrix} r_0(x_k) & \dots & r_{n-1}(x_k) \end{bmatrix} = \begin{bmatrix} r_0(x_k) & \dots & r_{n-1}(x_k) \end{bmatrix} \left(C_R(1:n, 1:n) - \frac{1}{P_n} \begin{bmatrix} 0 & \dots & 0 & P_0 \\ 0 & \dots & 0 & P_1 \\ \vdots & & \vdots & \vdots \\ 0 & \dots & 0 & P_{n-1} \end{bmatrix} \right) \tag{3.2.9}$$

where $C_R(1:n, 1:n)$ is the $n \times n$ leading sub diagonal of C_R and $\{P_k\}$ are the coefficients of the master polynomial (3.2.8).

The proof is immediate from the definition of the master polynomial (3.2.8) and (3.2.7). It is significant to note that that when using the truncated multiplication operator, the we use the polynomial set $\bar{R} = \{r_0(x), \dots, r_{n-1}(x), x \cdot r_{n-1}(x)\}$ instead of the conventional $R = \{r_0(x), \dots, r_n(x)\}$ in the definition of the master polynomial. The motivation behind using this set will be discussed in a later section as it applies to the Laurent polynomials.

For the remainder of this section, we will refer to the right hand side of (3.2.9), the truncated multiplication operator with respect to the master polynomial, $P(x)$, as $C_R(P)$.

Example 3.2.7. If $R = \{1, x, \ldots, x^{n-1}, P(x)\}$ where $P(x)$ is the master polynomial

$$P(x) = \prod_{k=1}^{n}(x - x_k) = P_0 + P_1 \cdot x + \cdots + P_{n-1}x^{n-1} + P_n \cdot x^n \tag{3.2.10}$$

Then

$$C_R(P) = \begin{bmatrix} 0 & 0 & \ldots & 0 & -\dfrac{P_0}{P_n} \\ 1 & 0 & \ldots & 0 & -\dfrac{P_1}{P_n} \\ 0 & 1 & \ldots & 0 & -\dfrac{P_2}{P_n} \\ & \ddots & \ddots & \vdots & \vdots \\ & & 0 & 1 & -\dfrac{P_{n-1}}{P_n} \end{bmatrix}. \tag{3.2.11}$$

Example 3.2.8. For the power basis $P_{[-m,n-1]}$ as in (3.1.1) and the master polynomial $P(x)$ defined in (3.1.3), the multiplication operator, $C_{P_{[-m,n-1]}}(P)$ is given by

$$C_{P_{[-m,n-1]}}(P) = \begin{bmatrix} 0 & 0 & & & \ldots & & & 0 & -P_{-m}/P_n \\ 1 & 0 & & & \ldots & & & 0 & -P_{-m+1}/P_n \\ 0 & \ddots & \ddots & & & & & \vdots & \vdots \\ & \ddots & \ddots & \ddots & & & & \vdots & \vdots \\ & & 0 & 1 & & & & 0 & -P_{-1}/P_n \\ & & & 0 & 1 & & & 0 & -P_0/P_n \\ & & & & \ddots & \ddots & & \vdots & \vdots \\ & & & & & 0 & 1 & 0 & -P_{n-2}/P_n \\ & & & & & & 0 & 1 & -P_{n-1}/P_n \end{bmatrix}. \tag{3.2.12}$$

The previous lemma and most recent example leads immediately to a connection between the power basis' truncated multiplication operator and the power-Vandermonde matrix (3.1.2).

Lemma 3.2.9. *Let $P := P_{[-m,n-1]}$ be the power basis as in (3.1.1) along with the master polynomial $P(x)$ defined in (3.1.3), and $C_P(P)$ be its multiplication operator (3.2.12). Let $\{x_k\}$ for $k = 1, \ldots, n+m$ be $n+m$ distinct nodes with $x_k \neq 0$. Then the following relation holds:*

$$D_x \cdot V_P(x) = V_P(x) \cdot C_P(P) \tag{3.2.13}$$

where $D_x = diag\{x_k\}$ and $V_P(x)$ is the power-Vandermonde matrix (2.2.1).

The proof, again, is immediate from the previous definitions. The relation (3.2.13) holds for all polynomial systems through the previous definitions.

3.2.2 Truncated multiplication operator for associated polynomials

In the previous Traub-like algorithms (see, e.g. [2, 3, 4]), the confederate matrix for a given polynomial system, $C_R(P)$, is used to generate the confederate matrix for the system of associated (or horner) polynomials, $C_{\widehat{R}}(P)$, via (2.2.11). We now will generalize the property (2.2.11) for the power basis through the truncated multiplication operator.

Theorem 3.2.10. *Let $P := P_{[-m,n-1]}$ be the power basis (3.1.1) and $\widehat{P} = \{\widehat{r}_k(x)\}_{k=-m}^{n-1}$ be the associated polynomials (3.1.5). Also let $\{x_k\}$ be $n+m$ distinct nodes with $x_k \neq 0$ for all $k = 1, \ldots, n+m$. Finally, define the master polynomial as in (3.1.3). Then*

$$C_{\widehat{P}}(P) = \tilde{I} \cdot C_P^T(P) \cdot \tilde{I}, \tag{3.2.14}$$

where $\tilde{I}$ is the anti-diagonal matrix (2.2.12) and $\widehat{P}$ is the associated-power polynomials, written as

$$\left[\widehat{r}_0(x_k) \quad \widehat{r}_1(x_k) \quad \dots \quad \widehat{r}_{n-1}(x_k) \quad \widehat{r}_{-m}(x_k) \quad \dots \quad \widehat{r}_{-1}(x_k)\right]. \tag{3.2.15}$$

Proof. Let $\widehat{P}$ be given as in (3.1.5), satisfying recurrence relations (3.1.10) and (3.1.11). Rearranging the recurrence relations with respect to $x \cdot \widehat{r}_k(x)$ gives

$$\widehat{r}_0(x) = P_n, \quad x \cdot \widehat{r}_{k-1}(x) = \widehat{r}_k - P_{n-k} \tag{3.2.16}$$

for $k = 1, \dots, n-1$ and

$$x \cdot \widehat{r}_{-1}(x) = -P_{-m}, \quad x \cdot \widehat{r}_{-k}(x) = \widehat{r}_{-k+1} - P_{-m+k-1} \tag{3.2.17}$$

for $k = 2, \dots, m$.

After noting that each coefficient of the master polynomial P_k can be written as

$$P_k = \frac{P_k}{P_n} \cdot \widehat{r}_0(x) \tag{3.2.18}$$

and writing the vector of associated polynomials evaluated at each node x_k as

$$\left[\widehat{r}_0(x_k) \quad \widehat{r}_1(x_k) \quad \dots \quad \widehat{r}_{n-1}(x_k) \quad \widehat{r}_{-m}(x_k) \quad \dots \quad \widehat{r}_{-1}(x_k)\right], \tag{3.2.19}$$

the multiplication operator $C_{\widehat{P}}(P)$ is given by

$$C_{\widehat{P}}(P) = \begin{bmatrix} -\frac{P_{n-1}}{P_n} & -\frac{P_{n-2}}{P_n} & \cdots & -\frac{P_1}{P_n} & -\frac{P_0}{P_n} & -\frac{P_{-1}}{P_n} & \cdots & -\frac{P_{-m+1}}{P_n} & -\frac{P_{-m}}{P_n} \\ 1 & 0 & \cdots & & & & & & 0 \\ & \ddots & \ddots & & & & & & \vdots \\ & & \ddots & \ddots & & & & & \vdots \\ & & & 1 & 0 & & & & \\ & & & & 1 & 0 & & & \\ & & & & & \ddots & \ddots & & \vdots \\ & & & & & & \ddots & 0 & \\ & & & & & & & 1 & 0 \end{bmatrix}. \tag{3.2.20}$$

Finally, for the power basis $P_{[-m,n]}$ and master polynomial $P(x)$, the truncated multiplication operator $C_P(P)$ is given by (3.2.12), proving the relation. □

The previous theorem may seem redundant since we already have recurrence relations for the associated polynomials $\widehat{R}$. However, we will use the property (3.2.14) to generalize the connection between multiplication operators for polynomial systems and related associated polynomials.

Lemma 3.2.11. *Let $P := P_{[-m,n-1]}$ be the power basis (3.1.1) and $\widehat{P} = \{\widehat{r}_k(x)\}_{k=-m}^{n-1}$ be the associated polynomials (3.1.5). Also let $\{x_k\}$ be $n+m$ distinct nodes with $x_k \neq 0$ for all $k = 1, \ldots, n+m$. Define the power-Vandermonde matrix, $V_P(x)$, as in (3.1.2). Then the inverse, $V_P(x)^{-1}$, is given by*

$$V_P(x)^{-1} = \tilde{I} \cdot V_{\widehat{P}}^T(x) \cdot diag\left(\left[\frac{1}{P'(x_k)}\right]_{k=1}^{n+m}\right) \tag{3.2.21}$$

where $V_{\widehat{P}}(x)$ *is the* associated-Vandermonde matrix

$$V_{\widehat{P}}(x) = \begin{bmatrix} \widehat{r}_0(x_1) & \widehat{r}_1(x_1) & \dots & \widehat{r}_{n-1}(x_1) & \widehat{r}_{-m}(x_1) & \dots & \widehat{r}_{-1}(x_1) \\ \widehat{r}_0(x_2) & \widehat{r}_1(x_2) & \dots & \widehat{r}_{n-1}(x_2) & \widehat{r}_{-m}(x_2) & \dots & \widehat{r}_{-1}(x_2) \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ \widehat{r}_0(x_{n+m}) & \widehat{r}_1(x_{n+m}) & \dots & \widehat{r}_{n-1}(x_{n+m}) & \widehat{r}_{-m}(x_{n+m}) & \dots & \widehat{r}_{-1}(x_{n+m}) \end{bmatrix} \tag{3.2.22}$$

The proof is straightforward from equations (3.1.9) and (3.2.22) .

3.3 Multiplication operator for a general system

Now that we have show how to invert a power-Vandermonde matrix and find the recurrence relations associated with that system, we can apply those results to a general set of polynomials.

3.3.1 Admissible system

We begin by defining a system of polynomials guaranteed to have an associated multiplication operator, called an *admissible system*.

Definition 3.3.1. Let $\Pi_{[-m,n]} = span\{x^k, k = -m, \dots, -1, 0, 1, \dots, n\}$ be the space of polynomials of degree $-m$ to n. The set of polynomials $L = \{\ell_k(x)\}$ for $k = -m, \dots, n-1$ and is called **admissible** if they satisfy the following:

1. $\ell_0(x) \in \Pi_0$
2. $\ell_k(x) \in \Pi_{[-m,n-1]}$ for all $k = -m \dots, n-1$.

3. $\{\ell_k(x)\}_{k=-m}^{n-1}$ form a basis of $\Pi_{[-m,n-1]}$

4. $\ell_{n-1}(x) \in \Pi_{[-m,n-1]}$ and $\notin \Pi_{[-m,n-2]}$.

Example 3.3.2 (Admissible systems)**.** The following are examples of admissible systems

1. The monomial basis $\{1, x, \ldots, x^{n-1}\}$.

2. $R = \{r_k(x)\}_{k=0}^{n-1}$ with $deg(r_k(x)) = k$.

3. The power basis $P_{[-m,n-1]}$.

The restrictions on an admissible system guarantee that $x \cdot \ell_k(x) \in \Pi_{[-m+1,n]}$, in which case we can write it as

$$x \cdot \ell_k(x) = \sum_{j=-m}^{n-1} a_{j,k}\ell_j(x) + a_{n,k} \cdot x \cdot \ell_{n-1}(x) \tag{3.3.1}$$

After defining the master polynomial $P(x)$

$$P(x) = \prod_{k=1}^{m} \left(\frac{1}{x} - \frac{1}{x_k}\right) \cdot \prod_{k=m+1}^{n} (x - x_k), \tag{3.3.2}$$

equation (3.3.1) leads to the existence of a truncated multiplication operator, $C_L(P)$. In addition to a multiplication operator, the system L of admissible polynomials, along with distinct $\{x_k\}_{k=1}^{n+m+1}$ admits an admissible-Vandermonde matrix, $V_L(x)$.

Definition 3.3.3. Let L be an admissible system and $\{x_k\}_{k=1}^{n+m}$ be $n+m$ distinct nodes where $\ell_k(x_j)$ exists for all j, k. Then the admissible-Vandermonde matrix, $V_L(x)$, is given by

$$V_L(x) = \begin{bmatrix} \ell_{-m}(x_1) & \dots & \ell_{n-1}(x_1) \\ \ell_{-m}(x_2) & \dots & \ell_{n-1}(x_2) \\ \vdots & & \vdots \\ \ell_{-m}(x_{n+m}) & \dots & \ell_{n-1}(x_{n+m}) \end{bmatrix}. \tag{3.3.3}$$

As before, we look to invert the admissible-Vandermonde matrix, (3.3.3) using a Traub-like algorithm:

Algorithm 3.3.4 (Admissible-Vandermonde inverse)**.** Let $V_L(x)$ be an admissible-Vandermonde matrix (3.3.3) and $\{x_k\}_{k=1}^{n+m+1}$ be $n+m+1$ distinct nodes. The, $V_L^{-1}(x)$ can be evaluated via:

1. Decompose the master polynomial $P(x)$ in the L basis:

$$P(x) = \prod_{k=1}^{m} \left(\frac{1}{x} - \frac{1}{x_k} \right) \cdot \prod_{k=m+1}^{n} (x - x_k) = \sum_{k=-m}^{n-1} P_k \cdot \ell_k(x) + P_n \cdot x \cdot \ell_{n-1}(x). \tag{3.3.4}$$

2. Define the associated polynomials $\widehat{\ell}_k(x)$ via the divided difference:

$$Q[t,x] = \frac{P(t) - P(x)}{t - x} = \sum_{k=-m}^{-1} \widehat{\ell}_{-m-k-1}(x) \cdot \ell_k(t) + \sum_{k=0}^{n-1} \widehat{\ell}_{n-k-1}(x) \cdot \ell_k(t) \tag{3.3.5}$$

3. The inverse of an admissible-Vandermonde matrix is given by

$$V_L(x)^{-1} = \tilde{I} \cdot V_{\widehat{L}}^T(x) \cdot diag\left(\left[\frac{1}{P'(x_k)} \right]_{k=1}^{n+m} \right) \tag{3.3.6}$$

where $V_{\widehat{L}}(x)$ is the associated-admissible-Vandermonde matrix

$$V_{\widehat{L}}(x) = \begin{bmatrix} \widehat{\ell}_0(x_1) & \widehat{\ell}_1(x_1) & \dots & \widehat{\ell}_{n-1}(x_1) & \widehat{\ell}_{-m}(x_1) & \dots & \widehat{\ell}_{-1}(x_1) \\ \widehat{\ell}_0(x_2) & \widehat{\ell}_1(x_2) & \dots & \widehat{\ell}_{n-1}(x_2) & \widehat{\ell}_{-m}(x_2) & \dots & \widehat{\ell}_{-1}(x_2) \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ \widehat{\ell}_0(x_{n+m}) & \widehat{\ell}_1(x_{n+m}) & \dots & \widehat{\ell}_{n-1}(x_{n+m}) & \widehat{\ell}_{-m}(x_{n+m}) & \dots & \widehat{\ell}_{-1}(x_{n+m}) \end{bmatrix}. \tag{3.3.7}$$

3.4 Associated-admissible polynomials.

Now that an algorithm for inverting any admissible-Vandermonde matrix has been established, we look to identify a pertransposition-type relationship between the truncated-multiplication operators for the system, $C_L(P)$ and the associated admissible $C_{\widehat{L}}(P)$. To do so, we first must establish the concept of a change-of-basis matrix.

3.4.1 Passing from the power basis to the admissible basis.

Along with the power basis $P_{[-m,n]}$ defined by (3.1.1), consider another set of admissible polynomials, defined by

$$\ell_k(x) = \sum_{j=-m}^{n-1} a_{j,k} \cdot x^j. \tag{3.4.1}$$

Let

$$S_{PL} = [s_{ij}] \tag{3.4.2}$$

be the matrix corresponding to passing from the basis P to the basis L in the linear space $\Pi_{[-m,n-1]}$. Clearly, the entries of the upper-triangular matrix S_{PL} are specified by (3.4.1)

and therefore

$$V_L(x) = V_P(x) \cdot S_{PL}. \tag{3.4.3}$$

3.4.2 Change of the confederate matrix

The following lemma shows how the truncated multiplication operator changes under the passage to another basis.

Lemma 3.4.1. *Let S_{PL} be the upper triangular matrix defined by (3.4.2). Also, let $C_P(P)$ be the the truncated multiplication operator (3.2.11) and $C_L(P)$ be the truncated multiplication operator associated with the admissible system. Then,*

$$C_L(P) = S_{PL}^{-1} \cdot C_P(P) \cdot S_{PL} \tag{3.4.4}$$

Proof. By definition of the truncated multiplication operator, we have

$$D_x \cdot V_P(x) = V_P(x) \cdot C_P(P). \tag{3.4.5}$$

Multiplying on the right of (3.4.5) by S_{PL} gives

$$D_x \cdot V_P(x) \cdot S_{PL} = V_P(x) \cdot S_{PL} \cdot S_{PL}^{-1} \cdot C_P(P) \cdot S_{PL}. \tag{3.4.6}$$

Applying (3.4.3) gives the result. □

3.4.3 Multiplication operator

The previous lemmas and definitions allow us to establish a generalized relationship between truncated multiplication operators for an admissible system and its associated polynomials.

Theorem 3.4.2. *Let $L = \{\ell_k(x)\}_{k=-m}^{n-1}$ be an admissible system and $\{x_k\}_{k=1}^{n+m}$ be distinct nodes. Let $V_L(x)$ be the admissible-Vandermonde matrix (3.3.3) and $P(x)$ be the master polynomial (3.3.4). Also let $C_L(P)$ be the truncated multiplication operator satisfying (3.4.5). Finally, let $\widehat{L} = \left\{\widehat{\ell}_k(x)\right\}_{k=-m}^{n-1}$ be the associated polynomials satisfying (3.3.5) and (3.3.6) and $C_{\widehat{L}}(P)$ be its truncated multiplication operator. Then*

$$C_{\widehat{L}}(P) = \tilde{I} \cdot C_L^T(P) \cdot \tilde{I}. \tag{3.4.7}$$

Proof. Let $V_P(x)$ be the power-Vandermonde matrix defined by (3.1.2) and S_{PL} be the change of basis matrix defined in (3.4.2). Applying S_{PL} and its inverse to $V_P(x)$ in the definition of V_P^{-1}, (3.2.21) gives

$$V_P \cdot S_{PL} \cdot S_{PL}^{-1} \cdot \tilde{I} \cdot V_{\widehat{P}}^T \cdot diag\left(\left[\frac{1}{P'(x_k)}\right]_{k=1}^{n+m}\right) = I. \tag{3.4.8}$$

Applying (3.4.3) to (3.4.8) gives

$$V_L \cdot S_{PL}^{-1} \cdot \tilde{I} \cdot V_{\widehat{P}}^T \cdot diag\left(\left[\frac{1}{P'(x_k)}\right]_{k=1}^{n+m}\right) = I. \tag{3.4.9}$$

Equations (3.3.6) and (3.4.9) imply that

$$\tilde{I} \cdot V_{\widehat{L}}^T = S_{PL}^{-1} \cdot \tilde{I} \cdot V_{\widehat{P}}^T. \tag{3.4.10}$$

Solving (3.4.10) for $V_{\widehat{L}}$ gives

$$V_{\widehat{L}} = V_{\widehat{P}} \cdot \tilde{I} \cdot S_{PL}^{-T} \cdot \tilde{I}. \tag{3.4.11}$$

Now, by the definition of the multiplication operator, applying (3.2.14) gives

$$D_x \cdot V_{\widehat{P}} = V_{\widehat{P}} \cdot \tilde{I} \cdot C_P^T \cdot \tilde{I}. \tag{3.4.12}$$

Applying (3.4.4) to (3.4.12) gives

$$\begin{aligned} D_x \cdot V_{\widehat{P}} &= V_{\widehat{P}} \cdot \tilde{I} \cdot \left(S_{PL} \cdot C_L \cdot S_{PL}^{-1}\right)^T \cdot \tilde{I} \\ &= \left(V_{\widehat{P}} \cdot \tilde{I} \cdot S_{PL}^{-T} \cdot \tilde{I}\right) \cdot \left(\tilde{I} \cdot C_L^T \cdot \tilde{I}\right) \cdot \tilde{I} \cdot S_{PL}^T \cdot \tilde{I}. \end{aligned} \tag{3.4.13}$$

Multiplying both sides of (3.4.13) on the right by $\tilde{I} \cdot S_{PL}^{-T} \cdot \tilde{I}$ and applying (3.4.11) gives the result. □

The above theorem is a powerful tool in any Traub-like algorithm for an admissible system. It allows us to generate the multiplication operator, and thus the recurrence relations for the set of associated-admissible polynomials. Also, this theorem is a generalization of known results for polynomials, for if $m = 0$ we have a polynomial system R with associated polynomials $\widehat{R}$, and

$$C_{\widehat{R}}(P) = \tilde{I} \cdot C_R^T(P) \cdot \tilde{I}, \tag{3.4.14}$$

which is specifically the pertransposition property in [4].

3.5 Twisted Green's matrices and Laurent polynomials

In this section we look to find the inverse of a Laurent-Vandermonde matrix using a Traub-like algorithm. First, we revisit the definitions of Green's matrices and polynomials, and use them to define the Laurent polynomials. Finally, we will apply (3.4.7) to generate recurrence relations for the associated-Laurent polynomials used to invert the Laurent-Vandermonde matrix.

3.5.1 Green's matrices and polynomials

Definition 3.5.1. A strictly upper Hessenberg matrix G is called Green's (H,1)-qs (or simply Green's matrix) if $\max\limits_{1\leq i\leq n} \text{rank } G(1:i, i:n) = 1$.

It is discussed in [23] that a Green's matrix is $(H,1)$-qs, which it has a generator definition. A strictly upper Hessenberg matrix G is Green's $(H,1)$-qs if it can be represented in the form

$$G = \begin{bmatrix} \hat{\tau}_0\tau_1 & \hat{\tau}_0\sigma_1\tau_2 & \hat{\tau}_0\sigma_1\sigma_2\tau_3 & \dots & \dots & \hat{\tau}_0\sigma_1\dots\sigma_{n-1}\tau_n \\ \hat{\sigma_1} & \hat{\tau}_1\tau_2 & \hat{\tau}_1\sigma_2\tau_3 & \dots & \dots & \hat{\tau}_1\sigma_2\dots\sigma_{n-1}\tau_n \\ 0 & \hat{\sigma_2} & \hat{\tau}_2\tau_3 & \dots & \dots & \hat{\tau}_2\sigma_3\dots\sigma_{n-1}\tau_n \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \hat{\sigma}_{n-2} & \hat{\tau}_{n-2}\tau_{n-1} & \hat{\tau}_{n-2}\sigma_{n-1}\tau_n \\ 0 & \dots & \dots & 0 & \hat{\sigma}_{n-1} & \hat{\tau}_{n-1}\tau_n \end{bmatrix} \tag{3.5.1}$$

where $\{\sigma_k, \tau_k, \widehat{\sigma}_k \neq 0, \widehat{\tau}_k\}$ are called the generators of G. It is also discussed in [23] that a unique characteristic of a Green's matrix is that it can be decomposed as a product of n matrices.

Definition 3.5.2. Let G be a Green's matrix. Then it has decomposition

$$G = \Theta_0 \Theta_1 \dots \Theta_n \tag{3.5.2}$$

where

$$\Theta_0 = \left[\begin{array}{c|c} \widehat{\tau}_0 & \\ \hline & I_{n-1} \end{array}\right], \Theta_k = \left[\begin{array}{c|cc|c} I_{k-1} & & & \\ \hline & \tau_k & \sigma_k & \\ & \widehat{\sigma}_k & \widehat{\tau}_k & \\ \hline & & & I_{n-k-1} \end{array}\right], \Theta_n = \left[\begin{array}{c|c} I_{n-1} & \\ \hline & \tau_n \end{array}\right]. \tag{3.5.3}$$

Definition 3.5.3. A system of polynomials $R = \{r_k(x)\}_{k=0}^n$ is called a system of *Green's polynomials* if it is related to a Green's matrix G via (2.2.6) with $\lambda_k = 1/\widehat{\sigma}_k$.

It was shown in [23] that if a system of polynomials are Green's polynomials, then they have the recurrence relations

$$\begin{bmatrix} f_0(x) \\ r_0(x) \end{bmatrix} = \begin{bmatrix} \widehat{\tau}_0 \\ 1 \end{bmatrix}, \quad \begin{bmatrix} f_k(x) \\ r_k(x) \end{bmatrix} = \frac{1}{\widehat{\sigma}_k} \begin{bmatrix} \widehat{\sigma}_k \sigma_k - \widehat{\tau}_k \tau_k & \widehat{\tau}_k \\ -\tau_k & 1 \end{bmatrix} \begin{bmatrix} f_{k-1}(x) \\ x \cdot r_{k-1}(x) \end{bmatrix} \tag{3.5.4}$$

where $\{f_k(x)\}$ are auxiliary polynomials.

3.5.2 Laurent polynomials

We use Green's polynomials and the auxiliary $f_k(x)$s to define Laurent polynomials:

Definition 3.5.4. Let $J = (j_1, j_2, \dots, j_n)$ be a sequence of binary digits. We define a

sequence of **Laurent polynomials** $\Psi = \{\psi_k(x)\}_{k=0}^{n-1}$ as

$$\psi_k(x) = \begin{cases} x^{-\sum_{m=1}^{k+1} j_m} r_k(x) & \text{if } j_{k+1} = 0 \\ x^{-\sum_{m=1}^{k+1} j_m} f_k(x) & \text{if } j_{k+1} = 1 \end{cases} . \tag{3.5.5}$$

where $r_k(x)$ are Green's polynomials and $f_k(x)$ are the auxiliary polynomials in (3.5.4).

To use the previous work on the Laurent polynomials, we must first show that they form an admissible system. To do so, we must enforce a few conditions on the Green's matrix G and the sequence of binary digits, J.

Lemma 3.5.5. *Let G be a Green's matrix defined by the factorization (3.5.2) and let k_0 be the first index such that Θ_{k_0} is singular. Then for any pattern $J = (j_1, \ldots, j_n)$ with $j_k = 0$ for $k > k_0$, Laurent polynomials $\{\psi_k(x)\}_{k=0}^{n}$ defined in (3.5.5) are linearly independent.*

The proof is direct from [23]. This lemma allows us to put the restriction that if the matrix Θ_k defined in (3.5.2) is nonsingular for all $k = 0, \ldots, n-1$, then the Laurent polynomials Ψ form a basis in $\text{span}\{\psi_k(x)\}$.

Theorem 3.5.6. *Let G be a Green's matrix defined by the factorization (3.5.2) with Θ_k is nonsingular for $k = 0, \ldots, n-1$. Also, let $J = (j_1, j_2, \ldots, j_n)$ be a sequence of binary digits with $j_1 = j_n = 0$. Then the Laurent polynomials $\{\psi_k(x)\}$ for $k = 0, \ldots, n-1$, as in (3.5.5) form an admissible system in the space $\Pi_{[-m,n-m]}$ where $m = \sum_{i=1}^{n} j_i$.*

Proof. Let $\{r_k(x)\}$ be the Green's polynomials associated with the Green's matrix G and $\{f_k(x)\}$ be the auxiliary polynomials defined in (3.5.4). Let $\{\psi_k(x)\}$ be the Laurent polynomials defined by (3.5.5). The fourth condition of an admissible system is satisfied since Θ_k are nonsingular for $k = 1, \ldots, n-1$. The condition that $j_1 = 0$ and the condition requirement

on $r_k(x)$ leads to

$$\psi_0(x) = r_0(x) \in \Pi_0, \tag{3.5.6}$$

satisfying the first condition of an admissible system.

Letting $j = \sum_{i=1}^{n} j_i$ together with the condition that $j_n = 0$, we have that

$$\psi_{n-1}(x) = x^{-j} \cdot r_{n-1}(x) \in \Pi_{[-j,n-j-1]} \tag{3.5.7}$$

which leads to $x \cdot \psi_{n-1}(x) \in \Pi_{[-j,n-j]}$ and $x \cdot \psi_{n-1}(x) \notin \Pi_{[-j,n-j-1]}$. □

Our next task is to create a two-term recurrence relations for the Laurent polynomials defined in (3.5.5). In doing so, we will use the known two-term recurrence relations for Green's polynomials (3.5.4) and the relationship between Green's polynomials and Laurent polynomials (3.5.5).

Theorem 3.5.7. *Let G be a Green's matrix with generators $\{\tau_k, \widehat{\tau}_k, \sigma_k, \widehat{\sigma}_k \neq 0\}$ and $r_k(x)$ be the Green's polynomials associated with G via (2.2.6). Let $f_k(x)$ be the auxiliary polynomials associated with $r_k(x)$ via (3.5.4). Then given a binary sequence of digits $J = (j_1, j_2, \ldots, j_n)$*

with $j_1 = j_n = 0$, the Laurent polynomials defined by (3.5.5) satisfy the recurrence relations

$$\begin{bmatrix} \varphi_0(x) \\ \psi_0(x) \end{bmatrix} = \begin{bmatrix} \widehat{\tau}_0 \\ 1 \end{bmatrix}, \quad \begin{cases} \begin{bmatrix} \varphi_k(x) \\ \psi_k(x) \end{bmatrix} = \dfrac{1}{\widehat{\sigma}_k} \begin{bmatrix} \widehat{\sigma}_k\sigma_k - \widehat{\tau}_k\tau_k & \widehat{\tau}_k \\ -\tau_k & 1 \end{bmatrix} \begin{bmatrix} \varphi_{k-1}(x) \\ x \cdot \psi_{k-1}(x) \end{bmatrix} & \text{if } j_k = 0, j_{k+1} = 0 \\ \begin{bmatrix} x \cdot \psi_k(x) \\ x \cdot \varphi_k(x) \end{bmatrix} = \dfrac{1}{\widehat{\sigma}_k} \begin{bmatrix} \widehat{\sigma}_k\sigma_k - \widehat{\tau}_k\tau_k & \widehat{\tau}_k \\ -\tau_k & 1 \end{bmatrix} \begin{bmatrix} \varphi_{k-1}(x) \\ x \cdot \psi_{k-1}(x) \end{bmatrix} & \text{if } j_k = 0, j_{k+1} = 1 \\ \begin{bmatrix} \varphi_k(x) \\ \psi_k(x) \end{bmatrix} = \dfrac{1}{\widehat{\sigma}_k} \begin{bmatrix} \widehat{\sigma}_k\sigma_k - \widehat{\tau}_k\tau_k & \widehat{\tau}_k \\ -\tau_k & 1 \end{bmatrix} \begin{bmatrix} \psi_{k-1}(x) \\ x \cdot \varphi_{k-1}(x) \end{bmatrix} & \text{if } j_k = 1, j_{k+1} = 0 \\ \begin{bmatrix} x \cdot \psi_k(x) \\ x \cdot \varphi_k(x) \end{bmatrix} = \dfrac{1}{\widehat{\sigma}_k} \begin{bmatrix} \widehat{\sigma}_k\sigma_k - \widehat{\tau}_k\tau_k & \widehat{\tau}_k \\ -\tau_k & 1 \end{bmatrix} \begin{bmatrix} \psi_{k-1}(x) \\ x \cdot \varphi_{k-1}(x) \end{bmatrix} & \text{if } j_k = 1, j_{k+1} = 1 \end{cases} \tag{3.5.8}$$

where $\varphi_k(x)$ are auxiliary Laurent polynomials, and are defined by

$$\varphi_k(x) = \begin{cases} x^{-\sum_{m=1}^{k+1} j_m} f_k(x) & \text{if } j_{k+1} = 0 \\ x^{-\sum_{m=1}^{k+1} j_m} r_k(x) & \text{if } j_{k+1} = 1 \end{cases}. \tag{3.5.9}$$

Proof. $j_1 = 0$ implies that $\psi_0(x) = r_0(x)$. Defining $\varphi_0(x) = f_0(x)$ via (3.5.9) and the first term in the Green's recurrence relations (3.5.4) gives the first term in (3.5.8). Now, assume that $j_2 = j_3 = \cdots = j_{k-1} = 0$. Let $\Psi = \{\psi_{k-1}, \psi_k\}$ and $\Phi = \{\varphi_k, \varphi_{k+1}\}$. The first two of the remaining four cases are broken down separately:

1. $\underline{j_k = 0, j_{k+1} = 0}$: Equations (3.5.5) and (3.5.9) give that $\Psi = \{r_{k-1}(x), r_k(x)\}$ and $\Phi = \{f_{k-1}(x), f_1(x)\}$. The Green's recurrence relation (3.5.4) verifies the result.

2. $\underline{j_k = 0, j_{k+1} = 1}$: Equations (3.5.5) and (3.5.9) imply that

$$\Psi = \{r_{k-1}(x), x^{-1} f_k(x)\}, \quad \Phi = \{f_{k-1}(x), x^{-1} r_k(x)\}. \tag{3.5.10}$$

Multiplying both of the second terms in (3.5.10) by x give that

$$x \cdot \psi_k(x) = f_k(x) \text{ and } x \cdot \varphi_k(x) = r_k(x). \tag{3.5.11}$$

Plugging terms in equations (3.5.10) and (3.5.11) into (3.5.4) give the result.

The final two cases are immediate from the definitions in the previous two. □

The recurrence relations (3.5.8) give a fast method of computing the Laurent polynomials without computing the Green's polynomials. It should be noted that in the recurrence relations, if $j_{k+1} = 1$, we are finding the term $x \cdot \psi_k(x)$, so to have an equation for $\psi_k(x)$, we must multiply the result by x^{-1}.

Example 3.5.8 (Building the Power basis)**.** Let $\tau_0 = 1$, $\tau_k = \widehat{\tau}_k = 0, \sigma_k = \widehat{\sigma}_k = 1$ for $k = 1, \ldots, 5$ and $j = (0, 1, 0, 1, 0)$. Using (3.5.8) we can build the Power basis $\Psi = \{1, x^{-1}, x, x^{-2}, x^2\}$ via:

$$\begin{bmatrix} \varphi_0(x) \\ \psi_0(x) \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix},$$

$$\begin{bmatrix} x \cdot \psi_1(x) \\ x \cdot \varphi_1(x) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ x \end{bmatrix},$$

$$\begin{bmatrix} \varphi_2(x) \\ \psi_2(x) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x^{-1} \\ x \end{bmatrix},$$

$$\begin{bmatrix} x \cdot \psi_3(x) \\ x \cdot \varphi_3(x) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x^{-1} \\ x^2 \end{bmatrix},$$

$$\begin{bmatrix} \varphi_4(x) \\ \psi_4(x) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x^{-2} \\ x^2 \end{bmatrix}.$$

Solving each case for $\psi_k(x)$ gives the result.

It is interesting (and useful for future applications) that if $j_{k+1} = 1$, then the recurrence relations (3.5.8) produce both $\psi_k(x)$ and $x \cdot \varphi_k(x)$. The second term $(x \cdot \varphi_k(x))$ is directly used in the next term in the recurrence relations.

3.5.3 Laurent multiplication operator

In order to apply the previous results, we first need a multiplication operator associated with the Laurent polynomials. It was established in [23] that the Green's matrix associated with the polynomials used to create the Laurent polynomials plays an important role in creating the multiplication operator, namely a **twisted-Green's matrix**

Definition 3.5.9. Let G be a Green's matrix with factorization (3.5.2) and $J = (j_1, j_2, \dots)$ be an infinite sequence of binary digits. The a **twisted Green's matrix**, G_J is defined by the recursion

$$G_0 = \Theta_0, \quad G_k = \begin{cases} G_{k-1}\Theta_k & \text{if } j_k = 0, \\ \Theta_k G_{k-1} & \text{if } j_k = 1, \end{cases} \quad G_J = G_\infty \tag{3.5.12}$$

Matrices G_J are related to the same polynomials $\{r_k(x)\}$ as G. Moreover, it was shown in [23] that the matrix G_J acts as a multiplication operator for the Laurent polynomials defined in (3.5.5).

Theorem 3.5.10. *Let G_J be a twisted Green's matrix of pattern $J = (j_1, j_2, \dots)$ defined by (3.5.12) and $\{\psi_k(x)\}_{k\geq 0}$ be the Laurent polynomials (3.5.5). Then*

$$x \cdot \begin{bmatrix} \psi_0(x) & \psi_1(x) & \psi_2(x) & \dots \end{bmatrix} = \begin{bmatrix} \psi_0(x) & \psi_1(x) & \psi_2(x) & \dots \end{bmatrix} \cdot G_J \tag{3.5.13}$$

The proof is found in [23]. This result is useful because it gives a multiplication operator for an infinite set of Laurent polynomials. This leads us to create the truncated multiplication operator with respect to the master polynomial.

Definition 3.5.11. Let $J = (j_1, j_2, \dots, j_n)$ be a finite sequence of binary digits with $j_1 = j_n = 0$. Also, let $\{x_k\}$ be n distinct nodes. Let $m = \sum j_k$. The we define the master polynomial, $P(x)$ as

$$\begin{aligned} P(x) &= \prod_{k=1}^{m} \left(\frac{1}{x} - \frac{1}{x_k}\right) \prod_{k=m+1}^{n} (x - x_k) \\ &= P_0\psi_0(x) + P_1\psi_1(x) + \ldots + P_{n-1}\psi_{n-1}(x) + P_n x \cdot \psi_{n-1}(x). \end{aligned} \tag{3.5.14}$$

Lemma 3.5.12. *Let G be a Green's matrix with factorization (3.5.2) and $J = (j_1, j_2, \dots, j_n)$ be a finite sequence of binary digits with $j_1 = j_n = 0$. Also, let $\{x_k\}$ be n distinct nodes and the master polynomial $P(x)$ be as in (3.5.14). Then the Laurent polynomials defined in (3.5.5) satisfy*

$$x_k \cdot \begin{bmatrix} \psi_0(x_k) & \dots & \psi_{n-1}(x_k) \end{bmatrix} = \begin{bmatrix} \psi_0(x_k) & \dots & \psi_{n-1}(x_k) \end{bmatrix} \cdot (G_J(1:n, 1:n) - P) \tag{3.5.15}$$

where $G_J(1:n, 1:n)$ *is the* $n \times n$ *leading submatrix of* G_J *and* P *is a rank 1 matrix containing the coefficients of the master polynomial* $P(x)$ *in the* $n-$*th column:*

$$P = \frac{1}{P_n} \begin{bmatrix} 0 & \dots & 0 & P_0 \\ 0 & \dots & 0 & P_1 \\ \vdots & & \vdots & \vdots \\ 0 & \dots & 0 & P_{n-1} \end{bmatrix} \tag{3.5.16}$$

It is important to notice that $P(x)$ is written using the Laurent basis

$$\overline{\Psi} = \{\psi_0(x), \dots, \psi_{n-1}(x), x \cdot \psi_{n-1}(x)\}, \tag{3.5.17}$$

which implies that $\tau_n = 0$. This is a direct result of the necessary $j_n = 0$, and results in $G_J(i, n) = 0$ for all i. The previous lemma establishes a truncated multiplication operator associated with the Laurent polynomials, which we will call $G_J(P)$ for brevity.

3.6 Laurent-Vandermonde matrix

Now that we have established the Laurent polynomials defined by (3.5.5) form an admissible system, we can form the Laurent-Vandermonde matrix and apply the methods in previous sections to find its inverse.

Definition 3.6.1. Let Ψ be a system of Laurent polynomials as in (3.5.5). Then $V_\Psi(x)$,

given by

$$V_\Psi(x) = \begin{bmatrix} \psi_0(x_1) & \dots & \psi_{n-1}(x_1) \\ \psi_0(x_2) & \dots & \psi_{n-1}(x_2) \\ \vdots & & \vdots \\ \psi_0(x_n) & \dots & \psi_{n-1}(x_n) \end{bmatrix} \tag{3.6.1}$$

is called a Laurent-Vandermonde matrix.

We can use the recurrence relations (3.5.9) to build the Laurent-Vandermonde matrix using the following algorithm produced in MATLAB.

Algorithm 3.6.2 (Laurent-Vandermonde matrix)**. Input:** A sequence of binary digits $J = (j_1, j_2, \dots, j_n)$ with $j_1 = j_n = 0$, generators $\{\tau_k, \widehat{\tau}_k, \sigma_k, \widehat{\sigma}_k\}$ for a twisted-Green's matrix G_J with respect to J and n nodes $\{x_k \neq 0\}$.

```
function [ V ] = LaurentVand(j, tau, tauhat, sigma, sigmahat, x)
n=numel(j);
V=ones(n);
F=zeros(n);
F(:,1)=tauhat(1)*V(:,1);
for k=1:n-1
    if j(k+1)==0
        if j(k)==0
        V(:,k+1)=1/sigmahat(k)*(-tau(k)*F(:,k)+x'.*V(:,k));
        F(:,k+1)=1/sigmahat(k)*((sigmahat(k)*sigma(k)-tau(k)*tauhat(k+1))*F(:,k)+tauhat(k+1)*x'.*V(:,k));
        else
        V(:,k+1)=1/sigmahat(k)*(-tau(k)*V(:,k)+x'.*F(:,k));
        F(:,k+1)=1/sigmahat(k)*((sigmahat(k)*sigma(k)-tau(k)*tauhat(k+1))*V(:,k)+tauhat(k+1)*x'.*F(:,k));
        end
    else
        if j(k)==0
        V(:,k+1)=1/sigmahat(k)*1./x'.*((sigmahat(k)*sigma(k)-tau(k)*tauhat(k+1))*F(:,k)+tauhat(k+1)*x'.*V(:,k));
        F(:,k+1)=1/sigmahat(k)*1./x'.*(-tau(k)*F(:,k)+x'.*V(:,k));
        else
        V(:,k+1)=1/sigmahat(k)*1./x'.*((sigmahat(k)*sigma(k)-tau(k)*tauhat(k+1))*V(:,k)+tauhat(k+1)*x'.*F(:,k));
```

```
        F(:,k+1)=1/sigmahat(k)*1./x'.*(-tau(k)*V(:,k)+x'.*F(:,k));
        end
    end
end
```

3.6.1 Laurent polynomials as an admissible system

Before applying the results, we need to address the order in which the Laurent polynomials are written. In an admissible system, the polynomials are ordered $\{\ell_{-m}, \ldots, \ell_{-1}, \ell_0, \ell_1, \ldots, \ell_n\}$. For this section alone, we will re-index the Laurent polynomials in a way so that we can relate them to an admissible system.

Definition 3.6.3. Let $J = (j_1, j_2, \ldots, j_n)$ be a finite sequence of binary digits with $j_1 = j_n = 0$ and the Laurent polynomials defined as in (3.5.5). let $m_k = \sum_{i=1}^{k+1} j_i$ and $m = m_n$. Then we re-index the Laurent polynomials as

$$\psi_k^{(k-m_k)}(x) \text{ if } j_{k+1} = 0 \tag{3.6.2}$$

$$\psi_k^{(-m_k)}(x) \text{ if } j_{k+1} = 1. \tag{3.6.3}$$

This re-indexing allows us to distinguish Laurent polynomials that are associated with $j_k = 0$ and $j_k = 1$, as $\psi_k^{(-i)}(x)$ represents the $i-$th polynomial such that $j_k = 1$ for $i = 1, \ldots, m$.

Example 3.6.4. Let $J = (0, 1, 0, 1, 0)$, then the Laurent polynomials will be indexed $\left\{\psi_0(x), \psi_1^{(-1)}(x), \psi_2^{(1)}(x), \psi_3^{(-2)}(x), \psi_4^{(2)}(x)\right\}$. The index in the superscript coincides with the indices of an admissible system.

We also use this re-indexing method on the coefficients of the master polynomial P_k in (3.5.14).

Example 3.6.5. Let $J = (0, 1, 0, 1, 0)$, then the Laurent polynomials will be indexed $\left\{\psi_0(x), \psi_1^{(-1)}(x), \psi_2^{(1)}(x), \psi_3^{(-2)}(x), \psi_3^{(2)}(x)\right\}$ and the master polynomial will be written as

$$P(x) = P_3^{(-2)}\psi_3^{(-2)}(x) + P_1^{(-1)}\psi_1^{(-1)}(x) + P_0\psi_0(x) + P_3^{(1)}\psi_3^{(1)}(x) + P_4^{(2)}\psi_2(x) + P_5 x \cdot \psi_2(x).$$

This leads to the matrix P to be given as

$$P = \frac{1}{P_5}\begin{bmatrix} & & P_0 \\ & & P_1^{(-1)} \\ & 0 & P_2^{(1)} \\ & & P_3^{(-2)} \\ & & P_4^{(2)} \end{bmatrix}.$$

Note that since $j_1 = j_n = 0$, there is no need for a superscript in P_0 or P_n. Now that the Laurent polynomials are indexed the same way as admissible polynomials, we must order them in the same way. The current ordering with respect to the truncated multiplication operator, $G_J(P)$, is

$$\begin{bmatrix} \psi_0(x_k) & \dots & \psi_{n-1}(x_k) \end{bmatrix}. \tag{3.6.4}$$

For this, we introduce a permutation matrix B so that

$$\begin{bmatrix} \psi_0(x_k) & \dots & \psi_{n-1}(x_k) \end{bmatrix} \cdot B = \begin{bmatrix} \psi^{(-m)}(x_k) & \psi^{(-m+1)}(x_k) & \dots & \psi_0(x_k) & \psi^{(1)}(x_k) & \dots & \psi^{(n-m-1)}(x_k) \end{bmatrix}, \tag{3.6.5}$$

where $m = \sum_{k=1}^{n} j_k$.

Definition 3.6.6. Let $J = (j_1, j_2, \dots, j_n)$ with $j_1 = j_n = 0$. We define the permutation

matrix B recursively via

$$B_0 = I \quad B_k = \begin{cases} B_{k-1} & \text{if } j_k = 0 \\ B_{k-1} \cdot E_k & \text{if } j_k = 1 \end{cases}, \quad B = B_n, \tag{3.6.6}$$

where

$$E_k = \begin{bmatrix} e_{j_k} & e_1 & \dots & e_{j_k-1} & e_{j_k+1} & \dots & e_n \end{bmatrix}. \tag{3.6.7}$$

and e_k is the standard notation of the $k-$th column of the identity matrix.

Example 3.6.7. Let $J = (0, 1, 0, 1, 0)$. Then $B = \begin{bmatrix} e_4 & e_2 & e_1 & e_3 & e_5 \end{bmatrix}$ and

$$\begin{bmatrix} \psi_0(x) & \psi_1^{(-1)}(x) & \psi_2^{(1)}(x) & \psi_3^{(-2)}(x) & \psi_4^{(2)}(x) \end{bmatrix} \cdot B = \begin{bmatrix} \psi_3^{(-2)}(x) & \psi_1^{(-1)}(x) & \psi_0(x) & \psi_2^{(1)}(x) & \psi_4^{(2)}(x) \end{bmatrix}.$$

Theorem 3.6.8. *Let G be a Green's matrix with factorization (3.5.2) and $J = (j_1, j_2, \dots, j_n)$ be a finite sequence of binary digits with $j_1 = j_n = 0$. Also, let $\{x_k\}$ be n distinct nodes and the master polynomial $P(x)$ be as in (3.5.14). Let $\Psi = \{\psi_k(x)\}$ be the Laurent polynomials and $G_J(P)$ be the twisted-Green's matrix associated with J. Then the truncated multiplication operator for the associated-Laurent polynomials, $\widehat{\Psi}$ is given by*

$$G_{\widehat{J}}(P) = \tilde{I} \cdot G_J^T(P) \cdot \tilde{I} \tag{3.6.8}$$

where $\widehat{J} = (\widehat{j}_1, \widehat{j}_2, \dots, \widehat{j}_n)$ is the binary sequence associated with the associated-Laurent polynomials with

$$\widehat{j}_k = j_{n-k+1}. \tag{3.6.9}$$

Proof. As shown in (3.5.15), $G_J(P)$ acts as a truncated multiplication operator for Ψ, or

$$x_k \cdot \begin{bmatrix} \psi_0(x_k) & \dots & \psi_{n-1}(x_k) \end{bmatrix} = \begin{bmatrix} \psi_0(x_k) & \dots & \psi_{n-1}(x_k) \end{bmatrix} \cdot G_J(P). \tag{3.6.10}$$

Define $m = \sum_{k=1}^{n} j_k$. Let the matrix B be the permutation matrix defined by (3.6.6). Multiplying the matrix B in (3.6.10) and that $B^{-1} = B^T$ leads to

$$x_k \cdot \begin{bmatrix} \psi^{(-m)}(x_k) & \dots & \psi^{(n-m-1)}(x_k) \end{bmatrix} = \begin{bmatrix} \psi^{(-m)}(x_k) & \dots & \psi^{(n-m-1)}(x_k) \end{bmatrix} \cdot B^T \cdot G_J(P) \cdot B, \tag{3.6.11}$$

showing that $B^T \cdot G_J(P) \cdot B$ is the multiplication operator for the Laurent polynomials ordered as in an admissible system. Applying the pertransposition property (3.4.7) gives

$$\begin{aligned} x_k \quad & \cdot \begin{bmatrix} \widehat{\psi}^{(0)}(x_k) & \dots & \widehat{\psi}^{(n-m-1)}(x_k) & \widehat{\psi}^{(-m)}(x_k) & \dots & \widehat{\psi}^{(-1)}(x_k) \end{bmatrix} \\ = \quad & \begin{bmatrix} \widehat{\psi}^{(0)}(x_k) & \dots & \widehat{\psi}^{(n-m-1)}(x_k) & \widehat{\psi}^{(-m)}(x_k) & \dots & \widehat{\psi}^{(-1)}(x_k) \end{bmatrix} \cdot \tilde{I} \cdot B^T \cdot G_J^T(P) \cdot B \cdot \tilde{I}. \end{aligned} \tag{3.6.12}$$

Multiplying on the right of (3.6.12) by $\left(\tilde{I} \cdot B^T \cdot \tilde{I}\right)$ gives the result. □

The previous theorem gives a direct relation between the recurrence relations for the Laurent polynomials, Ψ and their associated polynomials, $\widehat{\Psi}$.

We will use these recurrence relations in the next section in the Traub-like algorithm for the Laurent polynomials.

Remark 3.6.9. The requirement that $j_n = 0$ and the fact that we use the basis (3.5.17) implies that $\tau_n = 0$ for any system of Laurent polynomials. The requirement that $j_n = 0$ implies that any twisted Green's matrix G_J will be of the form

$$G_J = G_{n-1}\Theta_n, \tag{3.6.13}$$

where

$$\Theta_n = \left[\begin{array}{c|c} I_{n-1} & \\ \hline & \tau_n \end{array}\right]. \tag{3.6.14}$$

Applying the pertransposition operation on G_J implies that

$$G_{\widehat{J}}(P) = \tilde{I} \cdot \Theta_n^T \cdot \tilde{I} \cdot \tilde{I} \cdot G_{k-1} \cdot \tilde{I}, \tag{3.6.15}$$

where

$$\tilde{I} \cdot \Theta_n^T \cdot \tilde{I} = \left[\begin{array}{c|c} \tau_n & \\ \hline & I_{n-1} \end{array}\right]. \tag{3.6.16}$$

Equation (3.6.16) and $\tau_n = 0$ implies that $G_{\widehat{J}}(1:i) = 0$ for $i = 1, \ldots, n$.

3.6.2 Recurrence relations for associated-Laurent polynomials

We are now in a position to apply our results to build the associated-Laurent polynomials $\widehat{\Psi}$ that are used in the inverse of a Laurent-Vandermode matrix (3.6.1). For the remainder of this section, assume that we are given the generators for a Green's matrix $\{\tau_k, \widehat{\tau}_k, \sigma_k, \widehat{\sigma}_k \neq 0\}$ and a sequence of binary digits $J = (j_1, j_2, \ldots, j_n)$ with $j_1 = j_n = 0$.

Conjecture 3.6.10 (Recurrence relations for associated-Laurent polynomials)**.** *Let G be a Green's matrix with generators $\{\tau_k, \widehat{\tau}_k, \sigma_k, \widehat{\sigma}_k\}$ and $J = (j_1, j_2, \ldots, j_n)$ be a sequence of binary digits with $j_1 = j_n = 0$. Let $m_k = \sum_{j=1}^{k+1} j_k, i_k = k - m_k$ and $m = m_{n+1}$ Then the associated Laurent polynomials $\widehat{\Psi}$ satisfy the recurrence relations*

$$\begin{bmatrix} \widehat{\varphi}_0(x) \\ \widehat{\psi}_0(x) \end{bmatrix} = \begin{bmatrix} 0 \\ P_n \end{bmatrix}, \tag{3.6.17}$$

$\underline{\widehat{j}_k = 0, \widehat{j}_{k+1} = 0}$*:*

$$\begin{bmatrix} \widehat{\varphi}_k^{(i_k)}(x) \\ \widehat{\psi}_k^{(i_k)}(x) \end{bmatrix} = \frac{1}{\widehat{\sigma}_{n-k}} \begin{bmatrix} \widehat{\sigma}_{n-k}\sigma_{n-k} - \widehat{\tau}_{n-k}\tau_{n-k} & \tau_{n-k} \\ -\widehat{\tau}_{n-k} & 1 \end{bmatrix} \begin{bmatrix} \widehat{\varphi}_{k-1}^{(i_{k-1})}(x) \\ x \cdot \widehat{\psi}_{k-1}^{(i_{k-1})}(x) + P^{(n-m-i_k)} \end{bmatrix}$$

$\underline{\widehat{j}_k = 0, \widehat{j}_{k+1} = 1}$*:*

$$\begin{bmatrix} x \cdot \widehat{\psi}_k^{(-m_k)}(x) \\ x \cdot \widehat{\varphi}_k^{(-m_k)}(x) \end{bmatrix} = \frac{1}{\widehat{\sigma}_{n-k}} \begin{bmatrix} \widehat{\sigma}_{n-k}\sigma_{n-k} - \widehat{\tau}_{n-k}\tau_{n-k} & \tau_{n-k} \\ -\widehat{\tau}_{n-k} & 1 \end{bmatrix} \begin{bmatrix} \widehat{\varphi}_{k-1}^{(i_{k-1})}(x) \\ x \cdot \widehat{\psi}_{k-1}^{(i_{k-1})}(x) + P^{(n-m-i_k)} \end{bmatrix} - \begin{bmatrix} P^{(-m+m_k-1)} \\ 0 \end{bmatrix}$$

$\underline{\widehat{j}_k = 1, \widehat{j}_{k+1} = 0}$*:*

$$\begin{bmatrix} \widehat{\varphi}_k^{(i_k)}(x) \\ \widehat{\psi}_k^{(i_k)}(x) \end{bmatrix} = \frac{1}{\widehat{\sigma}_{n-k}} \begin{bmatrix} \widehat{\sigma}_{n-k}\sigma_{n-k} - \widehat{\tau}_{n-k}\tau_{n-k} & \tau_{n-k} \\ -\widehat{\tau}_{n-k} & 1 \end{bmatrix} \begin{bmatrix} \widehat{\psi}_{k-1}^{(i_{k-1})}(x) \\ x \cdot \widehat{\varphi}_{k-1}^{(i_{k-1})}(x) \end{bmatrix}$$

$\underline{\widehat{j}_{n-k} = 1, \widehat{j}_{k+1} = 1}$*:*

$$\begin{bmatrix} x \cdot \widehat{\psi}_k^{(-m_k)}(x) \\ x \cdot \widehat{\varphi}_k^{(-m_k)}(x) \end{bmatrix} = \frac{1}{\widehat{\sigma}_{n-k}} \begin{bmatrix} \widehat{\sigma}_{n-k}\sigma_{n-k} - \widehat{\tau}_{n-k}\tau_{n-k} & \tau_{n-k} \\ -\widehat{\tau}_{n-k} & 1 \end{bmatrix} \begin{bmatrix} \widehat{\psi}_{k-1}^{(-m_{k-1})}(x) \\ x \cdot \widehat{\varphi}_{k-1}^{(-m_{k-1})}(x) \end{bmatrix} - \begin{bmatrix} P^{(-m+m_k-1)} \\ 0 \end{bmatrix}.$$

where $\widehat{\varphi}_k(x)$ *are auxiliary polynomials and* $\widehat{j}_k = j_{n-k}$*.*

Example 3.6.11 (Building the associated-Power polynomials)**.** Let $\tau_0 = 1$, $\tau_k = \widehat{\tau}_k = 0, \sigma_k = \widehat{\sigma}_k = 1$ for $k = 1, \ldots, 5$ and $j = (0, 1, 0, 1, 0)$. Example 3.5.8 showed that $\Psi = \{1, x^{-1}, x, x^{-2}, x^2\}$. Using the recurrence relations (3.6.17), we can build the associated-Power polynomials $\widehat{\Psi} = \left\{\widehat{\psi}_0(x), \widehat{\psi}_1^{(-1)}(x), \widehat{\psi}_2^{(1)}(x), \widehat{\psi}_3^{(-2)}(x), \widehat{\psi}_4^{(2)}(x)\right\}$ via,

$$\begin{bmatrix} \widehat{\varphi}_0(x) \\ \widehat{\psi}_0(x) \end{bmatrix} = \begin{bmatrix} 0 \\ P_5^{(3)} \end{bmatrix},$$

$$\begin{bmatrix} x \cdot \widehat{\psi}_1^{(-1)}(x) \\ x \cdot \widehat{\varphi}_1^{(-1)}(x) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ P_5^{(3)}x + P_4^{(2)} \end{bmatrix} - \begin{bmatrix} P_3^{(-2)} \\ 0 \end{bmatrix},$$

$$\begin{bmatrix} \widehat{\varphi}_2^{(1)}(x) \\ \widehat{\psi}_2^{(1)}(x) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -P_3^{(-2)}x^{-1} \\ P_5^{(3)}x + P_4^{(2)} \end{bmatrix},$$

$$\begin{bmatrix} x \cdot \widehat{\psi}_3^{(-2)}(x) \\ x \cdot \widehat{\varphi}_3^{(-2)}(x) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -P_4^{(-2)}x^{-1} \\ P_5^{(3)}x^2 + P_4^{(2)}x + P_2^{(1)} \end{bmatrix} - \begin{bmatrix} P_1^{(-1)} \\ 0 \end{bmatrix},$$

$$\begin{bmatrix} \widehat{\varphi}_4^{(2)}(x) \\ \widehat{\psi}_4^{(2)}(x) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -P_3^{(-2)}x^{-2} - P_1^{(-1)}x^{-1} \\ P_5^{(3)}x^2 + P_4^{(2)}x + P_2^{(1)} \end{bmatrix}.$$

Solving each case for $\widehat{\psi}_k(x)$ gives the system

$$\begin{aligned} \widehat{\psi}_0(x) &= P_5^{(3)} \\ \widehat{\psi}_1^{(-1)}(x) &= -P_3^{(-2)}x^{-1} \\ \widehat{\psi}_2^{(1)}(x) &= P_5^{(3)}x + P_4^{(2)} \\ \widehat{\psi}_3^{(-2)}(x) &= -P_3^{(-2)}x^{-2} - P_1^{(-1)}x^{-1} \\ \widehat{\psi}_4^{(2)}(x) &= P_5^{(3)}x^2 + P_4^{(2)}x + P_2^{(1)}. \end{aligned}$$

When the superscripts are the focus, these are the exact associated polynomials produced using the direct method (3.1.10), (3.1.11).

3.6.3 Computing the coefficients of the master polynomial

Note that in order to use the recurrence relations of the previous section it is necessary to decompose the master polynomial $P(x)$ as in (3.5.14) into the $\overline{\Psi}$ basis (3.5.17). This can be done recursively by setting $\psi_{n-1}^{(0)}(x) = 1$ and then for $k = 1, \ldots, m$, updating

$$\psi_{n-1}^{(k)}(x) = \left(\frac{1}{x} - \frac{1}{x_k}\right) \cdot r_{n-1}^{(k-1)}(x), \tag{3.6.18}$$

where $m = \sum_{k=1}^{n} j_k$, and for $k = m+1, \ldots, n$, updating

$$\psi_{n-1}^{(k)}(x) = (x - x_k) \cdot r_{n-1}^{(k-1)}(x). \tag{3.6.19}$$

Lemma 5.1 in [4] suggests the following algorithm for computing coefficients $\{P_0, P_1, \ldots, P_n\}$ of the master polynomial.

Algorithm 3.6.12 (Computing the coefficients of the master polynomial in the Ψ basis)**. Input:** A sequence of binary digits $J = (j_1, j_2, \ldots, j_n)$ with $j_1 = j_n = 0$, generators $\{\tau_k, \widehat{\tau}_k, \sigma_k, \widehat{\sigma}_k\}$ for a twisted-Green's matrix G_J with respect to J and n nodes $\{x_k \neq 0\}$.

1. Set $\begin{bmatrix} P_0^{(0)} & \ldots & P_n^{(0)} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \ldots & 0 \end{bmatrix}$

2. For $k = 1 : m$,

$$x_k \cdot \left[\begin{array}{ccccc|c} & & G_J & & & 0 \\ \hline 0 & \ldots & 0 & 1 & 0 & \end{array}\right] \begin{bmatrix} P_0^{(k)} \\ \vdots \\ P_n^{(k)} \end{bmatrix} = \left(x_k \cdot I - \left[\begin{array}{ccccc|c} & & G_J & & & 0 \\ \hline 0 & \ldots & 0 & 1 & 0 & \end{array}\right] \right) \cdot \begin{bmatrix} P_0^{(k-1)} \\ \vdots \\ P_n^{(k-1)} \end{bmatrix}$$

where $m = \sum_{k=1}^{n} j_k$.

3. For $k = m+1 : n$,

$$\begin{bmatrix} P_0^{(k)} \\ \vdots \\ P_n^{(k)} \end{bmatrix} = \left(\left[\begin{array}{ccccc|c} & & G_J & & & 0 \\ \hline 0 & \dots & 0 & 1 & & 0 \end{array} \right] - x_k \cdot I \right) \cdot \begin{bmatrix} P_0^{(k-1)} \\ \vdots \\ P_n^{(k-1)} \end{bmatrix}$$

4. Take $\begin{bmatrix} P_0 & \dots & P_n \end{bmatrix} = \begin{bmatrix} P_0^{(n)} & \dots & P_n^{(n)} \end{bmatrix}$.

Clearly, the computational burden in this algorithm is the multiplication of the matrix G_J with a vector (Steps 2-3) and solving a linear system involving G_J (Step 2). The cost of each such step is $\mathcal{O}(m(n))$, where $m(n)$ is the cost of multiplying an $n \times n$ twisted-Green's matrix by a vector (or solving a linear system with it), thus the cost of computing the n coefficients is $\mathcal{O}(m(n) \times n)$. In [23], it was shown that any twisted-Green's matrix is also $(1,1)-$ quasiseparable. Using a fast $\mathcal{O}(n)$ algorithm for multiplication of a quasiseparable matrix by a vector first derived in [12] and the fast $\mathcal{O}(n)$ algorithm for solving a linear system in [1], the cost of this algorithm is $\mathcal{O}(n^2)$.

It is clear from Conjecture 3.6.10 that the sets of coefficients of the master polynomial associated with ψ_k when $j_{k+1} = 0$ and $j_{k+1} = 1$ are used differently when building the associated-Laurent polynomials. For this reason, we separate the vector containing the coefficients into two vectors in the following matlab function that computes the coefficients.

```
function [ p, q ] = MasterP(j,tau,tauhat,sigma,sigmahat, x)
n=numel(j);
J=sum(j);
G=Greens(j,tau,tauhat,sigma,sigmahat);

GG=blkdiag(G,0);
GG(n+1,n)=1;
```

```
P=eye(n+1,1);
for i=1:J
    Gg=GG(1:n+1,1:n);
    A=x(i).*Gg;
    Q=(x(i).*eye(n+1) - GG)*P;
    P=linsolve(A,Q);
    P(n+1)=0;
end
for i=J+1:n
    P=(GG-x(i).*eye(n+1))*P;
end

%Separate coefficients
%p contains coeffs associated with j=0
%q contains coeffs associated with j=1
i=1;J=1;
for k=1:n
    if j(k)==0
        p(i)=P(k);
        i=i+1;
    else
        q(J)=P(k);
        J=J+1;
    end
    p(i)=P(n+1);
end
end
```

3.6.4 Overall Traub-like algorithm for a Laurent-Vandermonde matrix

In this section we recall the formula that will be used to invert the Laurent-Vandermonde matrix (3.6.1), where Ψ is a system of Laurent polynomials associated with a sequence of binary digits $j = (j_1, \ldots, j_n)$ with $j_1 = j_n = 0$ via (3.5.5). The inverse of this matrix is given

by the formula

$$V_{\Psi}(x)^{-1} = \tilde{I} \cdot V_{\widehat{\Psi}}^{T}(x) \cdot \mathrm{diag}(c_1, \ldots, c_n) \tag{3.6.20}$$

where

$$c_i = \begin{cases} -(x_i)^2 \cdot \displaystyle\prod_{\substack{k=1 \\ k \neq i}}^{m} \left(\frac{1}{x_i} - \frac{1}{x_k} \right)^{-1} \prod_{k=m+1}^{n} (x_i - x_k)^{-1} & \text{for } i = 1, \ldots, m \\ \displaystyle\prod_{k=1}^{m} \left(\frac{1}{x_i} - \frac{1}{x_k} \right)^{-1} \prod_{\substack{k=m+1 \\ k \neq i}}^{n} (x_i - x_k)^{-1} & \text{for } i = m+1, \ldots, n \end{cases} \tag{3.6.21}$$

where $m = \sum_{k=1}^{n} j_k$.

The algorithm takes as its inputs the generators $\{\tau_k, \widehat{\tau}_k, \sigma_k, \widehat{\sigma}_k \neq 0\}$ of a Green's matrix G with $\tau_n = 0$, binary digits $j = (j_1, \ldots, j_n)$ with $j_1 = j_n = 0$ and n distinct nodes $\{x_k \neq 0\}$.

Algorithm 3.6.13 (Traub-like inversion algorithm)**. Input:** generators $\{\tau_k, \widehat{\tau}_k, \sigma_k, \widehat{\sigma}_k \neq 0\}$ of a Green's matrix G with $\tau_n = 0$, binary digits $j = (j_1, \ldots, j_n)$ with $j_1 = j_n = 0$ and n distinct nodes $\{x_k \neq 0\}$.

1. Compute the entries of $\mathrm{diag}(c_1, \ldots, c_n)$ via (3.6.21).

2. Compute the coefficients of the master polynomial $P(x)$ and separate them as in Algorithm 3.6.12.

3. Evaluate the n polynomials $\widehat{\Psi}$ at the n nodes for form $V_{\widehat{\Psi}}(x)$. Assertion 3.6.10 provides an algorithm for this, seen in the following MATLAB function

```
function [ V ] = LaurentVandHat(j, tau, tauhat, sigma, sigmahat, x)
n=numel(j);
[p,q]=MasterP(j,tau,tauhat,sigma,sigmahat, x);
P=numel(p);
Q=numel(q);
```

```
V=ones(n);
F=zeros(n);
V(:,1)=p(P)*V(:,1);
ip=1;
iq=0;
for k=1:n-1
    if j(n-k)==0
        if j(n-k+1)==0
        V(:,k+1)=1/sigmahat(n-k)*(-tauhat(n-k+1)*F(:,k)+x'.*V(:,k)+p(P-ip)*ones(n,1));
        F(:,k+1)=1/sigmahat(n-k)*((sigmahat(n-k)*sigma(n-k)-tau(n-k)*tauhat(n-k+1))*F(:,k)
                + tau(n-k)*(x'.*V(:,k)+p(P-ip)*ones(n,1)));
        ip=ip+1;
        else
        V(:,k+1)=1/sigmahat(n-k)*(-tauhat(n-k+1)*V(:,k)+x'.*F(:,k));
        F(:,k+1)=1/sigmahat(n-k)*((sigmahat(n-k)*sigma(n-k)-tau(n-k)*tauhat(n-k+1))*V(:,k)
                 +tau(n-k)*x'.*F(:,k));
        end
    else
        if j(n-k+1)==0
        V(:,k+1)=1/sigmahat(n-k)*1./x'.*((sigmahat(n-k)*sigma(n-k)-tau(n-k)*tauhat(n-k+1))*F(:,k)
                +tau(n-k)*(x'.*V(:,k)+p(P-ip)*ones(n,1)))-q(Q-iq)./x';
        F(:,k+1)=1/sigmahat(n-k)*1./x'.*(-tauhat(n-k+1)*F(:,k)+x'.*V(:,k)+p(P-ip)*ones(n,1));
ip=ip+1;
iq=iq+1;
        else
        V(:,k+1)=1/sigmahat(n-k)*1./x'.*((sigmahat(n-k)*sigma(n-k)-tau(n-k)*tauhat(n-k+1))*V(:,k)
                 +tau(n-k)*x'.*F(:,k))-q(Q-iq)./x';
        F(:,k+1)=1/sigmahat(n-kk)*1./x'.*(-tauhat(n-k+1)*V(:,k)+x'.*F(:,k));
iq=iq+1;
        end
    end
end
```

4. Compute $V_\Psi(x)^{-1}$ as in (3.6.20).

Chapter 4

Fast System-Solver for Cauchy-Vandermonde Matrices

4.1 Derivation of the BKO-type algorithm for a Cauchy-Vandermonde matrix

Following the methods in [7], we derive a similar decomposition of a Cauchy-Vandermonde matrix $W(x_{1:n}, y_{1:\ell})$ in [20] via purely matrix arguements.

Theorem 4.1.1. *The inverse of the Cauchy-Vandermonde matrix $W(x_{1:n}, y_{1:\ell})$ from (2.3.2) can be decomposed as*

$$W(x_{1:n}, y_{1:\ell})^{-1} = U_1^{-1} \cdot \ldots \cdot U_\ell^{-1} \cdot \hat{U}_{\ell+1}^{-1} \cdot \ldots \cdot \hat{U}_{n-1}^{-1} \cdot D^{-1} \cdot \hat{L}_{n-1}^{-1} \cdot \ldots \cdot \hat{L}_{\ell+1}^{-1} \cdot L_\ell^{-1} \cdot \ldots \cdot L_1^{-1} \quad (4.1.1)$$

where

$$L_k^{-1} = \left[\begin{array}{c|ccc} I_{k-1} & & & \\ \hline & 1 & & \\ & & \frac{1}{x_{k+1}-x_k} & \\ & & & \ddots \\ & & & \frac{1}{x_n-x_k} \end{array}\right] \left[\begin{array}{c|ccc} I_{k-1} & & & \\ \hline & 1 & & \\ & -(x_k-y_k) & (x_{k+1}-y_k) & \\ & \vdots & & \ddots \\ & -(x_k-y_k) & & (x_n-y_k) \end{array}\right] \tag{4.1.2}$$

for $k = 1 : \ell$,

$$\hat{L}_k^{-1} = \left[\begin{array}{c|ccc} I_{k-1} & & & \\ \hline & 1 & & \\ & & \frac{1}{x_{k+1}-x_k} & \\ & & & \ddots \\ & & & \frac{1}{x_n-x_k} \end{array}\right] \left[\begin{array}{c|ccc} I_{k-1} & & & \\ \hline & 1 & & \\ & -1 & 1 & \\ & \vdots & & \ddots \\ & -1 & & 1 \end{array}\right] \tag{4.1.3}$$

for $k = \ell + 1 : n - 1$,

$$\hat{U}_k^{-1} = \left[\begin{array}{c|ccc} I_{k-1} & & & \\ \hline & 1 & -x_k & \\ & & \ddots & \ddots \\ & & & 1 & -x_k \\ & & & & 1 \end{array}\right] \tag{4.1.4}$$

for $k = \ell + 1 : n - 1$

$$U_k^{-1} = \left[\begin{array}{c|cccccccc} I_{k-1} & & & & & & & & \\ \hline & 1 & -\frac{(x_k - y_k)}{(y_k - y_{k+1})} & \cdots & -\frac{(x_k - y_k)}{(y_k - y_\ell)} & -(x_k - y_k) & -y_k(x_k - y_k) & \cdots & -(y_k)^{n-\ell-1}(x_k - y_k) \\ & & \frac{(x_k - y_{k+1})}{y_k - y_{k+1}} & & & & & & \\ & & & \ddots & & & & & \\ & & & & \frac{(x_k - y_\ell)}{y_k - y_\ell} & & & & \\ & & & & & 1 & -(x_k - y_k) & \cdots & -(y_k)^{n-\ell-2}(x_k - y_k) \\ & & & & & & \ddots & \ddots & \vdots \\ & & & & & & & 1 & -(x_k - y_k) \\ & & & & & & & & 1 \end{array}\right] \tag{4.1.5}$$

for $k = 1 : \ell$, *and*

$$D^{-1} = \begin{bmatrix} (x_1 - y_1) & & & & & \\ & \ddots & & & & \\ & & (x_\ell - y_\ell) & & & \\ & & & 1 & & \\ & & & & \ddots & \\ & & & & & 1 \end{bmatrix}. \tag{4.1.6}$$

Proof. The proof uses induction on n. Note that if $\ell = 0$, we have a Vandermonde matrix, and $\ell = n$ we have a Cauchy matrix. In both cases, the factorization holds.

For $n = 1$ there is nothing to show. Assume that $2 \leq n, 1 \leq \ell < n$ and that the factorization holds for all matrices of size $n - 1$. Now let $W(x_{1:n}, y_{1:\ell}) \in M_n$ as below

$$W(x_{1:n}, y_{1:\ell}) = \begin{bmatrix} \frac{1}{x_1 - y_1} & \frac{1}{x_1 - y_2} & \dots & \frac{1}{x_1 - y_\ell} & 1 & x_1 & \dots & (x_1)^{n-\ell-1} \\ \frac{1}{x_2 - y_1} & \frac{1}{x_2 - y_2} & \dots & \frac{1}{x_2 - y_\ell} & 1 & x_2 & \dots & (x_2)^{n-\ell-1} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ \frac{1}{x_n - y_1} & \frac{1}{x_n - y_2} & \dots & \frac{1}{x_n - y_\ell} & 1 & x_n & \dots & (x_n)^{n-\ell-1} \end{bmatrix}$$

We want to show that

$$L_1^{-1} W U_1^{-1} = \left[\begin{array}{c|c} * & \\ \hline & W(x_{2:n}, y_{2:\ell}) \end{array}\right]$$

where $W(x_{2:n}, y_{2:\ell}) \in M_{n-1}$ is a Cauchy-Vandermonde matrix of size $n-1$. Note that since $\ell < n$, we are not concerned with any $\hat{L}$ or $\hat{U}$, since those factors affect only the Vandermonde part of W.

The factor L_1^{-1} clearly eliminates the first column (other than the pivot). To see the

affect of U_1^{-1}, it helps to look at U_k^{-1} as the product of two matrices, $U_{k1}^{-1}U_{k2}^{-1}$, where

$$U_{k1}^{-1} = \left[\begin{array}{c|cccccccc} I_{k-1} & & & & & & & & \\ \hline & 1 & -(x_k - y_k) & \dots & -(x_k - y_k) & -(x_k - y_k) & 0 & \dots & 0 \\ & & (x_k - y_{k+1}) & & & & & & \\ & & & \ddots & & & & & \\ & & & & (x_k - y_\ell) & & & & \\ & & & & & 1 & -x_k & & \\ & & & & & & \ddots & \ddots & \\ & & & & & & & 1 & -x_k \\ & & & & & & & & 1 \end{array}\right]$$

and

$$U_{k2}^{-1} = \left[\begin{array}{c|cccccccc} I_{k-1} & & & & & & & & \\ \hline & 1 & & & & & & & \\ & & \frac{1}{y_k - y_{k+1}} & & & & & & \\ & & & \ddots & & & & & \\ & & & & \frac{1}{y_k - y_\ell} & & & & \\ & & & & & 1 & y_k & (y_k)^2 & \dots & (y_k)^{n-\ell-1} \\ & & & & & & 1 & y_k & \ddots & \vdots \\ & & & & & & & \ddots & \ddots & (y_k)^2 \\ & & & & & & & & \ddots & y_k \\ & & & & & & & & & 1 \end{array}\right].$$

Under this decomposition, it is clear that U_{11}^{-1} eliminates the first row of W, producing

$$L_1^{-1} W U_{11}^{-1} = \left[\begin{array}{c|cc} \dfrac{1}{x_1 - y_1} & 0 & \\ \hline & & \\ 0 & \tilde{C} & \tilde{V} \end{array}\right] \tag{4.1.7}$$

where

$$\tilde{C} = \begin{bmatrix} \dfrac{y_1 - y_2}{(x_2 - y_2)} & \cdots & \dfrac{y_1 - y_\ell}{(x_2 - y_\ell)} \\ \vdots & & \vdots \\ \dfrac{y_1 - y_\ell}{(x_n - y_2)} & \cdots & \dfrac{y_1 - y_\ell}{(x_n - y_\ell)} \end{bmatrix}$$

and

$$\tilde{V} = \begin{bmatrix} 1 & x_1 - y_1 & (x_1)^2 - x_1 \cdot y_1 & \cdots & (x_1)^{n-\ell-1} - (x_1)^{n-\ell-2} \cdot y_1 \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n - y_1 & (x_n)^2 - x_n \cdot y_1 & \cdots & (x_n)^{n-\ell-1} - (x_n)^{n-\ell-2} \cdot y_1 \end{bmatrix}.$$

Multiplying (4.1.7) on the right by U_{k2}^{-1} produces

$$\begin{aligned} L_1^{-1} \cdot W \cdot U_{11}^{-1} \cdot U_{12}^{-1} &= \left[\begin{array}{c|ccccccc} \dfrac{1}{x_1 - y_1} & 0 & \cdots & & & & & 0 \\ \hline & & & & & & & \\ 0 & \dfrac{1}{(x_2 - y_2)} & \cdots & \dfrac{1}{(x_2 - y_\ell)} & 1 & x_2 & \cdots & (x_2)^{n-\ell-1} \\ \vdots & \vdots & & \vdots & & & & \\ 0 & \dfrac{1}{(x_n - y_2)} & \cdots & \dfrac{1}{(x_n - y_\ell)} & 1 & x_n & \cdots & (x_n)^{n-\ell-1} \end{array}\right] \\ &= \left[\begin{array}{c|c} * & \\ \hline & W(x_{2:n}, y_{2:\ell}) \end{array}\right] \end{aligned} \tag{4.1.8}$$

Where $W(x_{2:n}, y_{2:\ell}) \in M_{n-1}$ is a Cauchy-Vandermonde matrix. By inductive assumption, it has the given factorization. Note that the first ℓ factors (L_k^{-1}, U_k^{-1}) complete the Cauchy

part of the matrix. After those, we are left with a submatrix of size $n - \ell$ that is strictly Vandermonde. The remaining $n - \ell$ factors $(\hat{L}_k^{-1}, \hat{U}_k^{-1})$ are the standard Björck-Pereyra factors of a Vandermonde matrix.

□

It should be noted that if $\ell = 0$, the resulting matrix $W(x_{1:n})$ is a full Vandermonde matrix. In this case, the factors L_k^{-1} and U_k^{-1} of (4.1.1) do not appear, and $D^{-1} = \mathrm{I}$, leaving

$$W(x_{1:n})^{-1} = \hat{U}_1^{-1} \cdot \ldots \cdot \hat{U}_{n-1}^{-1} \cdot \hat{L}_{n-1}^{-1} \cdot \ldots \cdot \hat{L}_1^{-1} \tag{4.1.9}$$

where $\hat{L}_k^{-1}$ and $\hat{U}_k^{-1}$ are (4.1.3) and (4.1.4) respectively. This is the exact decomposition of a Vandermonde matrix found in [5], with lambda L_k^{-1} factors rather than bidiagonal.

Similarly, If $\ell = n$, the resulting matrix $W(x_{1:n}, y_{1:n})$ is a full Cauchy matrix. In this case, the factors $\hat{L}_k^{-1}$ and $\hat{U}_k^{-1}$ of (4.1.1) do not appear, and $D^{-1} = \mathrm{diag}(x_k - y_k)$ for $k = 1 : n$, leaving

$$W(x_{1:n}, y_{1:n})^{-1} = U_1^{-1} \cdot \ldots \cdot U_{n-1}^{-1} \cdot D^{-1} \cdot L_{n-1}^{-1} \cdot \ldots \cdot L_1^{-1} \tag{4.1.10}$$

where L_k^{-1} is (4.1.2) and U_k^{-1} is the first $\ell \times \ell$ block diagonal of (4.1.5). This is the exact decomposition of a Cauchy matrix found in [8].

The representation for the inverse matrix $W(x_{1:n}, y_{1:\ell})^{-1}$ obtained from the above leads to the following algorithm for solving $Wa = f$.

Algorithm 4.1.2.

```
function [a]=MO(x,y,f)
a=f;
n=max(size(x));
l=max(size(y));

if l<n
if l>0
for k=1:l
```

```
    for j=k+1:n
        a(j)=(a(j)*(x(j)-y(k))-a(k)*(x(k)-y(k)))/(x(j)-x(k));
    end
end
end

for k=l+1:n-1
    for j=k+1:n
        a(j)=(a(j)-a(k))/(x(j)-x(k));
    end
end

for j=1:l
    a(j)=a(j)*(x(j)-y(j));
end

for k=n-1:-1:l+1
    for j=k:n-1
        a(j)=a(j)-x(k)*a(j+1);
    end
end

if l>0
for k=l:-1:1

    for j=k+1:l
        a(j)=a(j)/(y(k)-y(j));
    end

        for j=l+1:n-1
        for i=1:n-j
            a(j)=a(j)+a(j+i)*y(k)^i;
        end
        end
    for i=1:l-k+1
        a(k)=a(k)+a(k+i)*(y(k)-x(k));
    end
```

```
     for j=k+1:l
        a(j)=a(j)*(x(k)-y(j));
     end
    for i=l+1:n-1
        a(i)=a(i)-a(i+1)*x(k);
    end
end
end
end
    if l==n
        for k=1:n-1
            for j=k:n
                a(j)=a(j)*(x(j)-y(k));
            end
            for j=k+1:n
                a(j)=a(j)-a(k);
            end
            for j=k+1:n
                a(j)=a(j)/(x(j)-x(k));
            end
        end
        for k=1:n-1
            a(k)=a(k)/(x(k)-y(k));
        end
        a(n)=a(n)*(x(n)-y(n));
for k=n-1:-1:1
           for j=k+1:n
            a(j)=a(j)/(y(k)-y(j));
        end

    tmp=0;
    for j=n:-1:k+1
        tmp=tmp+a(j);
    end
    a(k)=a(k)-tmp;
```

```
    for j=k:n
        a(j)=a(j)*(x(k)-y(j));
    end
end
    end
end
```

4.2 Positivity of L and U

We produce an error analysis for the new BKO-type algorithm, producing backward error bounds involving the quantity

$$|L_1| \cdot \ldots \cdot |L_\ell| \cdot |\hat{L}_{\ell+1}| \cdot \ldots \cdot |\hat{L}_{n-1}| \cdot |D| \cdot |\hat{U}_{n-1}| \cdot \ldots \cdot |\hat{U}_{\ell+1}| \cdot |U_\ell| \cdot \ldots \cdot |U_1|. \tag{4.2.1}$$

The BKO-type algorithm produces a "lambda" shape for each L_k and $\hat{L}_k$ term. In contrast to the bidiagonal factors in [21] of the BP-type algorithm, the sparsity patterns of (4.1.2) and (4.1.3) immediately imply the equality

$$|L| = |L_1| \cdot \ldots \cdot |L_\ell| \cdot |\hat{L}_{\ell+1}| \cdot \ldots \cdot |\hat{L}_{n-1}|. \tag{4.2.2}$$

4.2.1 Partial Pivoting and CV-Leja ordering

The equality in (4.2.2) holds for all orderings of the nodes, so we look for the ordering that reduces the size of $|L|$. Here we assume that the two sets of nodes $\{x_k\}_{k=1:n}$ and $\{y_k\}_{k=1:\ell}$ are not separated from each other, and that the ordering of $\{x_k\}_{k=1:n}$ can be computed in advanced using $\mathcal{O}(n^2)$ flops.

The objective of *partial pivoting* is the successive maximization of the pivots of L in the

decomposition

$$W(x_{1:n}, y_{1:\ell}) = L \cdot U. \tag{4.2.3}$$

The factorization (4.1.1) of the L factors (4.1.2 and 4.1.3) of $W(x_{1:n}, y_{1:\ell})$ lead to partial pivoting of $W(x_{1:n}, y_{1:\ell})$ to be the successive maximization of

$$|\ell_{ii}| = \begin{cases} \left| \dfrac{\prod_{j=1}^{i-1}(x_i - x_j)}{(x_i - y_i)\prod_{j=1}^{i-1}(x_j - y_i)} \right|, & \text{for } i = 2, \ldots, \ell \\ \\ \left| \prod_{j=1}^{i-1}(x_i - x_j) \right|, & \text{for } i = \ell + 1, \ldots, n-1. \end{cases} \tag{4.2.4}$$

Note that the first $\ell \times \ell$ submatrices of $W(x_{1:n}, y_{1:\ell})$ are classic Cauchy matrices, so the quantities of $|\ell_{ii}|$ are that of the Cauchy-Leja ordering from [8]. We shall call this procedure *CV-Leja* ordering, and can be computed in advance by the following algorithm.

Algorithm 4.2.1. CV-Leja ordering

```
function [x,f] = CV_L(x,y)
n = max(size(x));
l=max(size(y));
dist = 0; m = 1; aux = zeros(1,n);
for i = 1:n
aux(i) = abs(1 / (x(i) - y(1)));
if dist<aux(i) m = i; dist = aux(i);
end
end
x = swap(x,1,m); aux(m) = aux(1);
f = swap(f,1,m);
if n<=2 return; end

for i = 2:l
dist = 0; m = i;
for j = i:n
aux(j) = aux(j) * abs((x(j) - x(i-1)) / (x(j) - y(i)));
```

```
if dist<aux(j) m = j; dist = aux(j);
end
end
x = swap(x,i,m); aux(m) = aux(i);
end
if l<n-1
for i=l+1:n-1
dist = 0; m=i;
for j=i:n
 aux(j)=aux(j)*abs(x(j)-x(i));
if dist<aux(j) m = j; dist = aux(j);
end
end
end
x = swap(x,i,m); aux(m) = aux(i);
end
```

Numerical example: In this example, consider the Cauchy-Vandermonde matrix $W(x_{1:n}, y_{1:\ell})$ with n random nodes: $\{x_k : 0 < x_k < 1\}$ and $\ell = \lfloor n/2 \rfloor$ random poles: $\{y_k : 0 < y_k < 1\}$. We calculate $||L||_2$ for:

- **rand:** Random ordering of the nodes
- **mon:** Monotonic ordering of the nodes
- **CV-L:** CV-Leja ordering of the nodes (4.2.1)

The results in Table 4.2.1 confirmed that the ordering (4.2.1) of the nodes significantly reduces the size of $|L|$. In addition, monotonic ordering produces much larger values for $|L|$. This is significant because the equality (4.2.2) holds for the bidiagonal factors in [21] only when the nodes are ordered monotonically (see, e.g., [6]).

TABLE 4.2.1: $||L||_2$

n	**rand**	**mon**	**CV-L**
10	7.5e+01	1.9e+03	2.2e+00
15	2.2e+02	2.4e+03	2.9e+00
20	1.7e+05	1.5e+04	4.2e+00
25	5.1e+02	1.0e+06	4.6e+00
30	6.6e+03	1.1e+09	3.2e+00

4.2.2 Conditions for positivity of U

We progress to produce the error analysis for the new BKO-type algorithm involving the quantity (4.2.1). The sparsity pattern of (4.1.4) immediately leads to

$$|\hat{U}| = |\hat{U}_{\ell+1}| \cdot \ldots \cdot |\hat{U}_{n-1}|. \tag{4.2.5}$$

Finally, the sparsity pattern of U_k^{-1} (4.1.5) does not have a direct relationship with positivity. However, the next theorem shows the conditions under which we have the equality

$$|U_\ell \cdot \ldots \cdot U_1| = |U_\ell| \cdot \ldots \cdot |U_1|. \tag{4.2.6}$$

Theorem 4.2.2. *Let U_k^{-1} be given as in (4.1.5). Then U_k has a similar sparsity pattern and satisfies (4.2.6) under the conditions:*

$$x_k \cdot y_k < 0 \tag{4.2.7}$$

or

$$0 < |x_k| - |y_k| \tag{4.2.8}$$

for all $k = 1 : \ell$.

Proof. The equation for U_k is given as followed:

$$U_k = \left[\begin{array}{c|cccccccc} I_{k-1} & & & & & & & & \\ \hline & 1 & \frac{(x_k - y_k)}{(x_k - y_{k+1})} & \cdots & \frac{(x_k - y_k)}{(x_k - y_\ell)} & (x_k - y_k) & x_k(x_k - y_k) & \cdots & (x_k)^{n-\ell-1}(x_k - y_k) \\ & & \frac{y_k - y_{k+1}}{(x_k - y_{k+1})} & & & & & & \\ & & & \ddots & & & & & \\ & & & & \frac{y_k - y_\ell}{(x_k - y_\ell)} & & & & \\ & & & & & 1 & (x_k - y_k) & \cdots & (x_k)^{n-\ell-2}(x_k - y_k) \\ & & & & & & \ddots & \ddots & \vdots \\ & & & & & & & 1 & (x_k - y_k) \\ & & & & & & & & 1 \end{array}\right]. \tag{4.2.9}$$

which clearly has the same sparsity pattern as U_k^{-1} in (4.1.5). As for positivity, we will show that if x_k, y_k and x_{k-1}, y_{k-1} satisfy (4.2.7) or (4.2.8), then

$$|U_k \cdot U_{k-1}| = |U_k| \cdot |U_{k-1}| \tag{4.2.10}$$

The structure of the first $\ell - 1$ rows of (4.2.9) are "lambda" shaped, and thus equality holds for all nodes (see, e.g., [8]). When looking at the second block-diagonal submatrix, we consider the cases:

1. $x_k \cdot y_k < 0$ (4.2.7):

(a) $x_k > 0$: Equality (4.2.10) is immediate.

(b) $x_k < 0$: U_k and U_{k-1} submatrices have "checkerboard pattern", for example:

$$U_k = \begin{bmatrix} 1 & - & + & - & + & - \\ & 1 & - & + & - & + \\ & & 1 & - & + & - \\ & & & 1 & - & + \\ & & & & 1 & - \\ & & & & & 1 \end{bmatrix} \tag{4.2.11}$$

Multiplication of the matrix $\hat{I}$:

$$\hat{I} = \begin{bmatrix} 1 & & & & & \\ & -1 & & & & \\ & & 1 & & & \\ & & & -1 & & \\ & & & & 1 & \\ & & & & & -1 \end{bmatrix} \tag{4.2.12}$$

yields that $|U_k| = \hat{I} \cdot U_k \cdot \hat{I}$. If this pattern holds for U_{k-1} also, then the equality (4.2.10) is immediate.

2. $0 < |x_k| - |y_k|$ (4.2.8). All possibilities are either totally positive as in (4.2.7) (a) or checkerboard as in (4.2.7) (b).

□

So we have found that

$$|L|\cdot|DU| = |L_1|\cdot\ldots\cdot|L_\ell|\cdot|\hat{L}_{\ell+1}|\cdot\ldots\cdot|\hat{L}_{n-1}|\cdot|D|\cdot|\hat{U}_{n-1}|\cdot\ldots\cdot|\hat{U}_{\ell+1}|\cdot|U_\ell|\cdot\ldots\cdot|U_1| \quad (4.2.13)$$

for a set of nodes and poles that is more general than the condition of total positivity:

$$y_\ell < \cdots < y_1 < 0 < x_1 < \cdots < x_n \quad (4.2.14)$$

required in the bidiagonal case in [21].

4.3 Rounding error analysis

We look to find a bound for the computed solution $\hat{a}$, which is the exact solution of a nearby system $(W(x_{1:n}, y_{1:\ell})+\Delta W)\hat{a} = f$. It is well-known that the bounds for Gaussian elimination are

$$|\Delta R| \le 2\gamma_n|\hat{L}||\hat{U}|, \qquad \text{where} \qquad \gamma_n = \frac{nu}{1-nu} \quad (4.3.1)$$

see, e.g., p. 175 in [17]. Here $\hat{L}$ and $\hat{U}$ denote computed triangular factors. These bounds are the motivation to produce the bounds in the form of

$$|\Delta W| \le d_n u|L||DU|. \quad (4.3.2)$$

The equality (4.2.13) in general leads to the following backward error bound.

4.3.1 Backward Stability of Algorithm 4.1.2

Theorem 4.3.1. *Assume that Algorithm 4.1.2 (BKO-type algorithm) is carried out in floating point arithmetic with a unit roundoff u, and that no overflows were encountered during the computation. If the poles $\{y_k\}$ and the interpolation nodes $\{x_k\}$ satisfy (4.2.7) or (4.2.8) for $k = 1 : \ell$, then the computed solution $\hat{a}$ solves a nearby system*

$$(W(x_{1:n}, y_{1:\ell}) + \Delta W)\hat{a} = (U + \Delta U)(\hat{U} + \Delta\tilde{U})(D + \Delta D)(\hat{L} + \Delta\hat{L})(L + \Delta L)\hat{a} = f \quad (4.3.3)$$

with

$$|W(x_{1:n}, y_{1:\ell})\hat{a} - f| \leq ((6n^2 + n - 8n\ell + 2\ell^2 - 3)u + \mathcal{O}(n^2))|L||DU||\hat{a}| \quad (4.3.4)$$

$$|\Delta W| \leq ((6n^2 + n - 8n\ell + 2\ell^2 - 3)u + \mathcal{O}(n^2))|L||DU| \quad (4.3.5)$$

with

$$|\Delta L| \leq \gamma_{5\ell}|L|, \quad |\Delta\hat{L}| \leq \gamma_{3(n-\ell-1)}|\hat{L}|, \quad (4.3.6)$$

$$|\Delta D| \leq \gamma_2|D|, \quad (4.3.7)$$

$$|\Delta\hat{U}| \leq \gamma_{2n^2-2(\ell+2)n+2(\ell+1)}|\hat{U}| \quad (4.3.8)$$

$$|\Delta U| \leq \gamma_{2n+4}|U|. \quad (4.3.9)$$

Proof. Let us recall that algorithm 4.1.2 solves a Cauchy-Vandermonde linear system by computing

$$a = W(x_{1:n}, y_{1:\ell})^{-1} f = U_1^{-1} \dots U_\ell^{-1} \hat{U}_{\ell+1}^{-1} \dots \hat{U}_{n-1}^{-1} D \hat{L}_{n-1}^{-1} \dots \hat{L}_{\ell+1}^{-1} L_\ell \dots L_1^{-1} \cdot f \quad (4.3.10)$$

Where the $\{L_i^{-1}, \hat{L}_i^{-1}, D^{-1}, U_i^{-1}, \hat{U}_i^{-1}\}$ factors are given in (4.1.2)-(4.1.6).

First, we apply the standard error analysis for each elementary matrix-vector multiplication in (4.3.10) to show the computed solution $\hat{a}$ satisfies

$$\hat{a} = (U_1^{-1} * \delta U_1^{-1}) \dots (U_\ell^{-1} * \delta U_\ell^{-1})(\hat{U}_{\ell+1}^{-1} * \delta \hat{U}_{\ell+1}^{-1}) \dots (\hat{U}_{n-1}^{-1} * \delta \hat{U}_{n-1}^{-1}) \cdot \tag{4.3.11}$$
$$\cdot (D^{-1} * \delta D^{-1})(\hat{L}_{n-1}^{-1} * \delta \hat{L}_{n-1}^{-1}) \dots (\hat{L}_{\ell+1}^{-1} * \delta \hat{L}_{\ell+1}^{-1})(L_\ell^{-1} * \delta L_\ell^{-1}) \dots (L_1^{-1} * \delta L_1^{-1}) f$$

where the asterisk $*$ denotes the Hadamard (or componentwise) product.

Next, the bounds obtained for the $\delta L_k^{-1}, \delta \hat{L}_k^{-1}, \delta U_k^{-1}, \delta \hat{U}_k^{-1}, \delta D^{-1}$ will be used to deduce further bounds for $\Delta L_k, \Delta \hat{L}_k, \Delta U_k, \Delta \hat{U}_k, \Delta D$ defined by

$$(L_k * \delta L_k) = L_k + \Delta L_k, \quad (\hat{L}_k * \delta \hat{L}_k) = \hat{L}_k + \Delta \hat{L}_k, \tag{4.3.12}$$

$$(D * \delta D) = D + \Delta D, \tag{4.3.13}$$

$$(U_k * \delta U_k) = U_k + \Delta U_k, \quad (\hat{U}_k * \delta \hat{U}_k) = \hat{U}_k + \Delta \hat{U}_k. \tag{4.3.14}$$

Finally, inverting (4.3.1) we shall obtain (4.3.3) which will lead to the desired bounds in (4.3.6)-(4.3.9).

Lower triangular factors. We start with obtaining bounds for $\delta L_k, \delta \hat{L}_k, \Delta L_k$ and $\Delta \hat{L}_k$. The sparse nature of these matrices, see (4.1.2), (4.1.3), imply the following bound for the (i, j) entry of δL_k and $\delta \hat{L}_k$

$$(\delta L_k)_{i,j} \leq (1 + \delta)^5 \tag{4.3.15}$$

$$(\delta \hat{L}_k)_{i,j} \leq (1 + \delta)^3. \tag{4.3.16}$$

Since the L_k and $\hat{L}_k$ all have the exact same sparsity patterns as their inverses, equations

(4.3.12), (4.3.15) and (4.3.16) imply that

$$|\Delta L_k| \leq ((1+\delta)^5 - 1)|L_k| \tag{4.3.17}$$

$$|\Delta \hat{L}_k| \leq ((1+\delta)^3 - 1)|\hat{L}_k|. \tag{4.3.18}$$

Diagonal Factor The simple structure of D immediately implies

$$|\Delta D| \leq ((1+\delta)^2 - 1)|D|. \tag{4.3.19}$$

Upper triangular factors. Similar to the analysis of the lower triangular factors, the sparse nature of $\hat{U}_k$, see (4.1.4) implies the following bound for the (i,j) entry of $\Delta\hat{U}_k$:

$$(\delta\hat{U}_k)_{i,j} \leq (1+\delta)^2. \tag{4.3.20}$$

The well-known pattern for $\hat{U}_k$ yields

$$|\Delta\hat{U}_k| \leq \left((1+\delta)^{2(n-k)} - 1\right)|\hat{U}_k|. \tag{4.3.21}$$

Finally, U_k is given by (4.2.9). The inner product corresponding to the k-th row of U_k is evaluated from the last to the $k-$th entry, then the error in the (k,j) entry for $2 \leq j \leq \ell+1$ is bounded by

$$(\delta U_k)_{k,j} \leq (1+\delta)^{n-j+3}. \tag{4.3.22}$$

The error in the (k,j) entry for $\ell+2 \leq j \leq n$ is bounded by

$$(\delta U_k)_{k,j} \leq (1+\delta)^{n-\ell+1}. \tag{4.3.23}$$

Specifically, the $k \times k$ entry of δU_{k1} is bounded by

$$(\delta U_k)_{k,k} \leq (1+\delta)^{n-k}. \tag{4.3.24}$$

And for $k < i \leq \ell$, we have

$$(\delta U_k)_{i,i} \leq (1+\delta)^4. \tag{4.3.25}$$

Equations (4.3.22)-(4.3.25) imply

$$|\Delta U_k| \leq ((1+\delta)^{2n+2} - 1)|U_k| \tag{4.3.26}$$

To prove (4.3.6)-(4.3.9), we use the easily verified fact (see [17]). Let $|\Delta X_k| \leq \delta |X|$ for $k = 1, \ldots, m$, then

$$\left| \prod_{k=1}^{m} (X_k + \Delta X_k) - \prod_{k=1}^{m} X_k \right| \leq ((1-\delta)^m - 1) \prod_{k=1}^{m} |X_k|. \tag{4.3.27}$$

Equations (4.3.27) and (4.3.17) imply that

$$|\Delta L| = |(L_1 + \Delta L_1) \ldots (L_\ell + \Delta L_\ell) - L_1 \ldots L_\ell| \leq ((1+\delta)^{5\ell} - 1)|L_1| \ldots |L_\ell| \tag{4.3.28}$$

The sparsity pattern of L_k and $\hat{L}_k$ allows us to remove the moduli in the product on the right-hand side of (4.3.28), implying the first bound in (4.3.6). The second bound in (4.3.6) follows from (4.3.18) and (4.3.27). The bound in (4.3.9) follows from (4.3.26). The bound in (4.3.7) follows from (4.3.19) easily. Under the conditions (4.2.7) or (4.2.8), we can remove the moduli on the U_k factors, implying equations (4.3.8) and (4.3.5).

□

4.4 Full pivoting

Similar to partial pivoting, the objective of full pivoting is the successive maximization of the determinants of the leading submatrices, or the pivots. The factorization of $W(x_{1:n}, y_{1:\ell})$ leads to the following formula for its pivots.

Theorem 4.4.1. *Let $W(x_{1:n}, y_{1:\ell})$ be a Cauchy-Vandermonde matrix as in (2.3.2). The its pivots, d_i for $i = 1 : n$ are given by*

$$d_i = \begin{cases} \dfrac{1}{x_i - y_i} & \text{for } i = 1 \\ \dfrac{\prod_{j=1}^{i-1}(x_i - x_j)\prod_{j=1}^{i-1}(y_i - y_j)}{(x_i - y_i)\prod_{j=1}^{i-1}(x_i - y_j)\prod_{j=1}^{i-1}(x_j - y_i)}, & \text{for } i = 2, \ldots, \ell \\ \dfrac{\prod_{j=1}^{i-1}(x_i - x_j)}{\prod_{j=1}^{\ell}(x_i - y_j)}, & \text{for } i = \ell + 1, \ldots, n-1. \end{cases} \tag{4.4.1}$$

Proof. Immediate from (4.4.1)[1]. □

This definition of the pivots leads to the full pivoting of W as the successive maximization of

$$|d_i| = \begin{cases} \left| \dfrac{\prod_{j=1}^{i-1}(x_i - x_j)\prod_{j=1}^{i-1}(y_i - y_j)}{(x_i - y_i)\prod_{j=1}^{i-1}(x_i - y_j)\prod_{j=1}^{i-1}(x_j - y_i)} \right|, & \text{for } i = 2, \ldots, \ell \\ \left| \dfrac{\prod_{j=1}^{i-1}(x_i - x_j)}{\prod_{j=1}^{\ell}(x_i - y_j)} \right|, & \text{for } i = \ell + 1, \ldots, n-1. \end{cases} . \tag{4.4.2}$$

[1]This formula also follows immediately from the formula of the determinant of $W(x_{1:n}, y_{1:\ell})$ by Martinez and Peña in [20].

Similar to partial pivoting, note that the first $\ell \times \ell$ submatrices of $W(x_{1:n}, y_{1:\ell})$ are classic Cauchy matrices, so the quantities of $|d_{ii}|$ are that of the Cauchy-Leja ordering from [8]. We shall call this procedure *Full CV-Leja* ordering, and can be computed in advance by the following $\mathcal{O}(n^3)$ algorithm.

Algorithm 4.4.2. Full CV-Leja ordering

```
function [x,y] = CVLejaFull(x,y)
n = max(size(x));
l=max(size(y));
if l>0
dist = 0; m = 1; aux = zeros(n,l);
for i = 1:n
    for j=1:l
    aux(i,j) = abs(1 / (x(i) - y(j)));
if dist<aux(i,j)
    m = i;
    M = j;
    dist = aux(i,j);
end
    end
end
x = swap(x,1,m); aux(m,:)=aux(1,:);
y = swap(y,1,M); aux(:,M)=aux(:,1);

for i=2:l
    dist=0; m=i; M=i;
    for j=i:n
        for k=i:l
        aux(j,k)=aux(j,k)*abs((x(j)-x(i-1))/(x(j)-y(i-1))*(y(k)-y(i-1))/(x(i-1)-y(k)));
        if dist<aux(j,k)
            m = j;
            M = k;
            dist = aux(j,k);
        end
        end
    end
```

```
x = swap(x,i,m); aux(m,:)=aux(i,:);
y = swap(y,i,M); aux(:,M)=aux(:,i);
end

if l<n-1
for i=l+1:n-1
    dist = 0; m=i;
    for j=i:n
        aux(j,l)=aux(j,l)*abs(x(j)-x(i));
        if dist<aux(j,l)
            m = j;
            dist = aux(j,l);
        end
    end
end
x = swap(x,i,m); aux(m,:) = aux(i,:);
end
end
return
```

4.4.1 Numerical Illustrations

We performed numerous numerical tests for the algorithm suggested and analyzed in this paper. The results confirm theoretical results, as perhaps should be expected.

Consider the linear system

$$W(x_{1:n}, y_{1:\ell})a = f \tag{4.4.3}$$

where W is an $n \times n$ Cauchy-Vandermonde matrix, $f = \begin{bmatrix} -1 & 1 & -1 & \dots & (-1)^{n-1} & (-1)^n \end{bmatrix}^T$. We used the ordering:

- Random nodes $x = \{x_k : 0 < x_k < 2\}$, the random poles $y = \{y_k : 0 < y_k < 1\}$ for $\ell = \lfloor n/2 \rfloor$.

and tested the following algorithms:

1. **MO:** Algorithm 4.1.1

2. **MO+CVL:** 4.1.1 with CV-Leja ordering using algorithm 4.2.1.

3. **GEPP:** Gaussian elimination with partial-pivoting.

For each value n we solved the linear system by running the algorithm in MATLAB using double precision (unit roundoff $\approx 2.22 \times 10^{-16}$). We also calculated the backward error bound using the formula

$$b_i = \frac{||f - W \cdot \hat{a}_i||_2}{||W||_2 \cdot ||\hat{a}_i||_2}. \tag{4.4.4}$$

The results are found in Table 4.4.1.

TABLE 4.4.1: Backward error

n	Cond(W)	MO	MO+CVL	GEPP
10	5.6e+08	3.9e-17	7.9e-19	7.8e-19
15	2.2e+12	3.75e-17	1.5e-17	6.3e-18
20	1.3e+16	3.2e-14	8.6e-18	3.7e-18
25	1.7e+19	1.8e-12	3.4e-18	2.2e-18
30	2.9e+19	1.4e-13	1.5e-18	8.8e-18

Finally, we computed $||L||_2 \cdot ||DU||_2$ where $W = LDU$ is an $n \times n$ Cauchy-Vandermonde matrix with $\ell = \lfloor n/2 \rfloor$. We used the nodes $x_k = k$ and interlaced poles $y_k = \dfrac{x_{k+\ell/2} - x_{k+\ell/2-1}}{2}$ for the following orderings:

1. **mon:** monotonic ordering

2. **CVL:** CV-Leja ordering using algorithm 4.2.1

TABLE 4.4.2: $||L||_2 \cdot ||DU||_2$

n	**mon**	**CVL**	**Full CVL**
10	7.7e+06	1.2e+05	2.0e+04
15	8.7e+11	2.9e+09	2.2e+08
20	9.6e+17	1.1e+13	7.2e+11
25	7.2e+23	1.5e+18	7.5e+16
30	2.2e+29	1.5e+22	6.1e+20

3. **Full CVL:** Full CV-Leja ordering using algorithm 4.4.2.

Comparing the data in Tables 4.4.1 and 4.4.2 indicates the ordering of the nodes has a profound influence on the accuracy of the algorithm produced in this paper. The CV-Leja ordering (4.4.1) reduces the size of $|L| \cdot |DU|$.

Bibliography

[1] T. Bella, Y. Eidelman, I. Gohberg, V. Olshevsky, *Computations with quasiseparable polynomials and matrices.*

[2] T. Bella, Y. Eidelman, I. Gohberg, I. Koltratch, V. Olshevsky, *A fast Bjorck-Pereyra like algorithm for solving Hessenberg-quasiseparable-Vandermonde systems*, submitted to SIAM Journal of Matrix Analysis, (2007)

[3] T. Bella, Y. Eidelman, I. Gohberg, V. Olshevsky, E. Tyrtyshnikov, *Fast inversion of polynomial-Vandermonde matrices for polynomial systems related to order one quasiseparable matrices*, Advances in Structured Operator Theory and Related Areas, **237**, (2013), 79-106.

[4] T. Bella, Y. Eidelman, I. Gohberg, V. Olshevsky, E. Tyrtyshnikov, P. Zhlobich, *A Traub-like algorithm for Hessenberg-quasiseparable-VNdermonde matrices of arbitrary order*, (2010).

[5] A. Björck and V. Pereyra. *Solution of Vandermonde systems of equations.* Math. Comp, 24:893-903, 1970.

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

[7] T. Boros, T.Kailath and V.Olshevsky, *A Fast Parallel Björk-Pereyra-type Algorithm for Solving Cauchy Linear Equations*

[8] T. Boros, T.Kailath and V.Olshevsky, *Pivoting and Backward Stability of Fast Algorithms for solving Cauchy Linear Equations*

[9] T. Bella, V. Olshevsky, and P. Zhlobich, *Signal Flow Graph Approach to Inversion of (H. m)-quasiseparable Vandermonde Matrices and New Filter Structures*

[10] D. Calvetti and L. Reichel. *A Chebyshev-Vandermonde solver*. Linear Algebra and its Applications, 172(219-229), 1992.

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

[12] Y. Eidelman and I. Gohberg, *Linear complexity inversion algorithms for a class of structured matrices,* Integral Equations and Operator Theory, **35** (1999), 28-52.

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

[14] N.J. Higham. *Error analysis of the Björck-Pereyra algorithms for solving Vandermonde systems*. Numerische Mathematik, 50:613-632, 1987.

[15] N.J. Higham. *Fast solution of Vandermonde-like systems, involving orthogonal polynomials*. IMA J. Numerical Analysis, 8:473-486, 1988.

[16] N.J. Higham. *Stability analysis of algorithms for solving confluent Vandermonde-like systems.* SIAM, 11:23-41, 1990.

[17] N.J. Higham. *Accuracy and stability of numerical algorithms.* SIAM, 1996

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

[19] J. Maroulas and S. Barnett, *Polynomials with respect to a general basis.* I. Theory, J. of Math. Analysis and Appl., **72** (1979), 177-194.

[20] J.J. Martinez, J.M. Peña, *Factorizations of Cauchy-Vandermonde Matrices*

[21] J.J. Martinez, J.M. Peña, *Fast algorithms of Björck-Pereyra-type for solving Cauchy-Vandermonde linear systems*

[22] V. Olshevsky, *Associated polynomials, unitary Hessenberg matrices and fast generalized Parker-Traub and Bjorck-Pereyra algorithms for Szego-Vandermonde matrices* invited chapter in the book "Structured Matrices: Recent Developments in Theory and Computation", 67-78, NOVA Science Publ., USA. (2001).

[23] V. Olshevsky, G. Strang, P. Zhlobich, *Green's matrices*, Linear Algebra and its Applications, (2010).

[24] L. Reichel and G. Opfer. *Chebyshev-Vandermonde systems.* Math Comp., 57:703-721, 1991.

[25] L. Reichel. *Newton interpolation at Leja points.* BIT, 23-31, 1990

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